# Data Generation for PuMaS

José Bayoán Santiago Calderón

2018-10-24

## 1 Introduction

This introduction to data generation for PuMaS will cover how to specify subjects and populations to use with PuMaS. You will learn how to read data, specify dosage regimens, covariates, observations, and generate populations to use with the software.

### 1.1 Installation

Because PuMaS is still unregistered, you will need to give the Git repository in order to add the package. To do this, use the command `]dev https://github.com/UMCTM/PuMaS.jl`. Doing it this way, PuMaS and its dependencies will install automatically. If one cannot authenticate for this command (since the repository is currently private!), then first clone the repository how you please, use `]add path/to/PuMaS.jl`, then then do `]build PuMaS`. Using the build command will download and install the dependencies.

### 1.2 Subjects

PuMaS unit of observation are subjects.

Each `Subject` is an individual composed of:

- `id`: an unique identifier
- `cvs`: covariates
- `obs`: observations
- `evs`: events

### 1.3 Population

A Population is a collection of subjects.

## 1.4 NMTRAN data format

The NMTRAN schema uses a single tabular representation to describe a `Population`

Consider a NMTRAN text file. We can read the data using the CSV package to read it, and the DataFrames package to work with the tabular representation.

Load the packages through `using`

```
using CSV, DataFrames, PuMaS
using DataFrames: head
```

We'll be working with an example NMTRAN data set in PuMaS

```
nmtran_data = example_nmtran_data("data1")
show(nmtran_data)
```

```
"/Users/jbsc/.julia/packages/PuMaS/Zktty/src/../examples/data1.csv"
```

We can read the text file using the `CSV` and `DataFrames` packages

```
data = DataFrame(CSV.File(nmtran_data)) # Make a DataFrame from the text file.
head(data) # Show the first six rows
```

| | id | time | mdv | evid | dv | amt | sex | wt | etn |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0.0 | 1 | 1 | 0.0 | 10000.0 | 1 | 51.6 | 1 |
| 2 | 1 | 0.0 | 0 | 0 | 1.2161 | 0.0 | 1 | 51.6 | 1 |
| 3 | 1 | 0.25 | 0 | 0 | 12.565 | 0.0 | 1 | 51.6 | 1 |
| 4 | 1 | 0.5 | 0 | 0 | 11.218 | 0.0 | 1 | 51.6 | 1 |
| 5 | 1 | 0.75 | 0 | 0 | 17.709 | 0.0 | 1 | 51.6 | 1 |
| 6 | 1 | 1.0 | 0 | 0 | 21.925 | 0.0 | 1 | 51.6 | 1 |

One can obtain a `Population` from a NMTRAN table through the `process_nmtran` function.

```
using PuMaS
@doc process_nmtran
```

```
process_nmtran(filename, cvs=[], dvs=[:dv])
process_nmtran(data, cvs=[], dvs=[:dv])
```

Import NMTRAN-formatted data.

- `cvs` covariates specified by either names or column numbers
- `dvs` dependent variables specified by either names or column numbers

Based on the documentation for `process_nmtran` it takes a `DataFrame`, an optional list of covariates, and a list of dependent variables which `[:dv]` as the default value. In our case, we want to specify `sex` and `wt` as covariates.

```
population = process_nmtran(data, [:sex, :wt])
show(population)
```

```
Population
  Subjects: 40
  Covariates: sex, wt
  Observables: dv
```

Just like that we have read NMTRAN data and ready it for analysis using PuMaS.

## 1.5 Interactive

PuMaS offers an alternative to the NMTRAN schema is to generate data

Let us simulate the some values for glucose as out observations

```
using Distributions
distribution = Normal(5.5, 1.5)
show(distribution)
```

```
Normal{Float64}(μ=5.5, σ=1.5)
```

```
glucose = rand(distribution, 5) # Five draws from the distribution
```

```
obs = DataFrame(time = 0:8:32, # Makes a sequence starting at 0 by 8 until 32
                glucose = glucose)
head(obs)
```

| | time | glucose |
|---|---|---|
| 1 | 0 | 4.3813 |
| 2 | 8 | 6.69199 |
| 3 | 16 | 2.93397 |
| 4 | 24 | 3.39894 |
| 5 | 32 | 5.22363 |

Next we generate the covariates for our subject

```
cvs = (sex = "male", age = 25)
show(cvs)
```

```
(sex = "male", age = 25)
```

Lastly, we can generate a series of dosage events through a `DosageRegimen`

```
@doc DosageRegimen
```

`DosageRegimen`

Lazy representation of a series of Events.

The first argument is an amount. `addl` specifies additional dosages and `ii` the interdose interval.

```
evs = DosageRegimen(5, addl = 3, ii = 8u"hr")
head(evs.data)
```

| | time | cmt | amt | evid | ii | addl | rate | ss |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.0 | 1 | 5.0 | 1 | 8.0 | 3 | 0.0 | 0 |

This regimen specifies 4 dosage events of five milligrams starting at `time = 0` with an interdose interval of 8 hours

We can now fully specify a subject bringing the covariates, observations, and regimen

```
subject = Subject(cvs = cvs, obs = obs, evs = evs)
show(subject)
```

```
Subject
  Events: 4
  Observations: 5
  Covariates:
    sex: male
    age: 25
  Observables: glucose
```

We can inspect the observations through

```
for obs in subject.observations
    println(obs)
end
```

```
Observation
  time of measurement = 0.0
  compartment = 1
  measurements
    glucose = 4.3812984872411285

Observation
  time of measurement = 8.0
  compartment = 1
  measurements
    glucose = 6.691994650606107

Observation
  time of measurement = 16.0
  compartment = 1
  measurements
    glucose = 2.9339680551385783

Observation
  time of measurement = 24.0
  compartment = 1
  measurements
    glucose = 3.39893966030675

Observation
  time of measurement = 32.0
  compartment = 1
  measurements
    glucose = 5.223625554223433
```

Likewise for the dosage events

```
for evs in subject.events
    println(evs)
end
```

```
Dose event
  dose amount = 5.0
  dose time = 0.0
  compartment = 1
  instantaneous
  interdose interval = 8.0
  infusion start time = 0.0

Dose event
  dose amount = 5.0
```

```
  dose time = 8.0
  compartment = 1
  instantaneous
  interdose interval = 8.0
  infusion start time = 8.0

Dose event
  dose amount = 5.0
  dose time = 16.0
  compartment = 1
  instantaneous
  interdose interval = 8.0
  infusion start time = 16.0

Dose event
  dose amount = 5.0
  dose time = 24.0
  compartment = 1
  instantaneous
  interdose interval = 8.0
  infusion start time = 24.0
```

We can build more complex regimens by passing a collection of arguments

```
regimens = DosageRegimen([0.005u"g", 0.0025u"g"], time = [0, 90u"minute"])
head(regimens.data)
```

| | time | cmt | amt | evid | ii | addl | rate | ss |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.0 | 1 | 5.0 | 1 | 0.0 | 0 | 0.0 | 0 |
| 2 | 1.5 | 1 | 2.5 | 1 | 0.0 | 0 | 0.0 | 0 |

Given that we have specified the amount in grams, these are converted to milligrams

Likewise, for time which was specified in minutes, these are converted to hours.

```
subject2 = Subject(id = 2,
                   evs = regimens)
show(subject2)
```

```
Subject
  Events: 2
  Observations: 0
  Covariates:
```

This `Subject` does not have any observations or covariates.

The dosage regimen can be inspected through

```
for evs in subject2.events
    println(evs)
end
```

```
Dose event
  dose amount = 5.0
  dose time = 0.0
  compartment = 1
  instantaneous
  interdose interval = 0.0
  infusion start time = 0.0

Dose event
```

```
  dose amount = 2.5
  dose time = 1.5
  compartment = 1
  instantaneous
  interdose interval = 0.0
  infusion start time = 1.5
```

One may construct a `Population` by passing a collection of subjects

```
@doc Population
```

```
Population(::AbstractVector{<:Subject})
```

A `Population` is a set of `Subject`s. It can be instanced passing a collection or `Subject`.

There will be six subjects

```
ids = 1:6 # There will be 6 subject 1, 2, 3, 4, 5, 6
```

We will simulate the covariates for these six subjects

```
choose_sex_age() = (sex = rand(["male", "female"]),
                    age = rand(21:25))
```

`choose_sex_age` will randomly choose a sex and an age between 21-25 years

We can make a list with covariates for six subjects through a list comprehension

```
cvs = [ choose_sex_age() for i in 1:6 ]
show(cvs)
```

```
NamedTuple{(:sex, :age),Tuple{String,Int64}}[(sex = "female", age = 23), (s
ex = "male", age = 21), (sex = "female", age = 25), (sex = "female", age =
21), (sex = "female", age = 23), (sex = "male", age = 22)]
```

We can generate a table with the time and observations for the dependent variable for each of the six subjects

```
obs = [ DataFrame(time = 0:4:8, dv = rand(3)) for id in 1:6 ]
head.(obs[1:2])
```

```
2-element Array{DataFrame,1}:
 3×2 DataFrame
 Row  time   dv
      Int64  Float64

 1    0      0.915873
 2    4      0.214413
 3    8      0.404113
 3×2 DataFrame
 Row  time   dv
      Int64  Float64

 1    0      0.728336
 2    4      0.683909
 3    8      0.474315
```

We will assign one regimen to the first three and another to the last three

```
regimen1 = DosageRegimen(15, addl = 3, ii = 4)
regimen2 = DosageRegimen(10, addl = 4, ii = 3)
regimens = vcat(repeat([regimen1], 3),
                repeat([regimen2], 3))
```

We can now generate the six subjects using a

```
subjects = Subject[]
for (id, cvs, evs) in zip(ids, cvs, regimens)
    push!(subjects, Subject(id = id, cvs = cvs, evs = evs))
end
show(subjects)
```

```
Subject[Subject
  Events: 4
  Observations: 0
  Covariates:
    sex: female
    age: 23
, Subject
  Events: 4
  Observations: 0
  Covariates:
    sex: male
    age: 21
, Subject
  Events: 4
  Observations: 0
  Covariates:
    sex: female
    age: 25
, Subject
  Events: 5
  Observations: 0
  Covariates:
    sex: female
    age: 21
, Subject
  Events: 5
  Observations: 0
  Covariates:
    sex: female
    age: 23
, Subject
  Events: 5
  Observations: 0
  Covariates:
    sex: male
    age: 22
]
```

Lastly, we can make a population with comprised of these subjects

```
population = Population(subjects)
show(population)
```

```
Population
  Subjects: 6
  Covariates: sex, age
```

```
for subject in population
    println(subject)
end
```

```
Subject
  Events: 4
  Observations: 0
  Covariates:
    sex: female
    age: 23

Subject
  Events: 4
  Observations: 0
  Covariates:
    sex: male
    age: 21

Subject
  Events: 4
  Observations: 0
  Covariates:
    sex: female
    age: 25

Subject
  Events: 5
  Observations: 0
  Covariates:
    sex: female
    age: 21

Subject
  Events: 5
  Observations: 0
  Covariates:
    sex: female
    age: 23

Subject
  Events: 5
  Observations: 0
  Covariates:
    sex: male
    age: 22
```