Defining and Simulating Populations

PumasAI

August, 2020

```
using Pumas, DataFrames, LinearAlgebra, Plots
```

1 Introduction

In this tutorial, we will cover the fundamentals of generating populations to simulate with Pumas. We will demonstrate how to specify dosage regimens and covariates, and then how to piece these together to form a population to simulate.

1.1 The model

Below is a Pumas model that specifies a 1-compartment oral absorption system with between-subject variability on all the parameters. Details of the model specification are provided in the introduction tutorial.

```
model = @model begin
  @param begin
    θ ∈ VectorDomain(4)
    Ω ∈ PSDDomain(3)
    σ_prop ∈ RealDomain(init=0.1)
  end

  @random begin
    η ~ MvNormal(Ω)
  end

  @covariates isPM Wt

  @pre begin
    TVCL = isPM == 1 ? θ[1] : θ[4]
    CL = θ[1]*(Wt/70)^0.75*exp(η[1])
    V = θ[2]*(Wt/70)^0.75*exp(η[2])
    Ka = θ[3]*exp(η[3])
  end

  @dynamics begin
    Depot'    = -Ka*Depot
```

```
    Central' =  Ka*Depot - Central*CL/V
  end

  @vars begin
    conc = Central/V
  end

  @derived begin
    dv ~ @.Normal(conc,sqrt(conc^2*σ_prop+ eps()))
  end

end
```

```
PumasModel
  Parameters: θ, Ω, σ_prop
  Random effects: η
  Covariates: isPM, Wt
  Dynamical variables: Depot, Central
  Derived: conc, dv
  Observed: conc, dv
```

1.2 Setting up parameters

Next we provide the initial estimates of the parameters to simulate from. The fixed effects are provided in the θ vector (CL, V, Ka) and the between-subject variability parameteres are provided in the Ω vector as variances. So, 0.04 variance on Ω*11 suggests a 20% coefficient of variation. Similarly,* σprop has a 20% proportional residual error.

```
fixeffs = (
  θ = [0.4,20,1.1,2],
  Ω = diagm(0 => [0.04,0.04,0.04]),
  σ_prop = 0.04
  )
```

```
(θ = [0.4, 20.0, 1.1, 2.0], Ω = [0.04 0.0 0.0; 0.0 0.04 0.0; 0.0 0.0 0.04],
 σ_prop = 0.04)
```

1.3 Single dose example

`DosageRegimen()` is the function that lets you construct a dosing regimen. The first argument of the `DosageRegimen` is `amt` and is not a named argument. All subsequent arguments need to be named. Lets try a simple example where you provide a 100 mg dose at `time=0`.

```
ev = DosageRegimen(100, time=0)
first(ev.data)
```

	time	cmt	amt	evid	ii	addl	rate	duration	ss
	Float64	Int64	Float64	Int8	Float64	Int64	Float64	Float64	Int8
1	0.0	1	100.0	1	0.0	0	0.0	0.0	0

As you can see above, we provided a single 100 mg dose. `DosageRegimen` provides some defaults when it creates the dataset, `time=0`, `evid=1`, `cmt=1`, `rate=0`, `ii=0` & `addl=0`. We can also provide units to the `amt` and any other variable that is derived from `amt`, e.g. `rate`, will have associated units. Handling of units will be covered in a different tutorial.

Note that `ev` is of type `DosageRegimen`. Specified like above, `DosageRegimen` is one of the four fundamental building block of a `Subject` (more on `Subject` below).

1.3.1 Building Subjects

Let's create a single subject

```
s1 = Subject(id=1,events=ev,covariates=(isPM=0, Wt=70))
for fn in fieldnames(Subject)
    x = getproperty(s1, fn)
    if !isa(x, Nothing)
        println(fn)
        println(x)
    end
end
```

```
id
1
covariates
Pumas.ConstantCovar{NamedTuple{(:isPM, :Wt),Tuple{Int64,Int64}}}((isPM = 0,
 Wt = 70))
events
Pumas.Event[Dose event
  dose amount = 100.0
  dose time = 0.0
  compartment = 1
  instantaneous
  interdose interval = 0.0
  infusion start time = 0.0
]
```

Note that each `Subject` is an individual composed of:

- `id`: an unique identifier
- `obs`: observations, represented by `Pumas.Observation[]`
- `cvs`: covariates
- `evs`: events, represented by `Pumas.Event[]`

In the example above, we only provided the `id`, `evs`, and the `cvs`. Since `obs` were not provided, they are represented by an empty array. Lets take a closer at the events for this subject 1.

```
s1.events
```

```
1-element Array{Pumas.Event,1}:
 Dose event
  dose amount = 100.0
  dose time = 0.0
  compartment = 1
  instantaneous
  interdose interval = 0.0
  infusion start time = 0.0
```

The events are presented by basic information such as the dose of drug and associated units if specified, the time of dose administration, the compartment number for administration and whether the dose is an instantaneous input or an infusion.

Below is how the covariates are represented

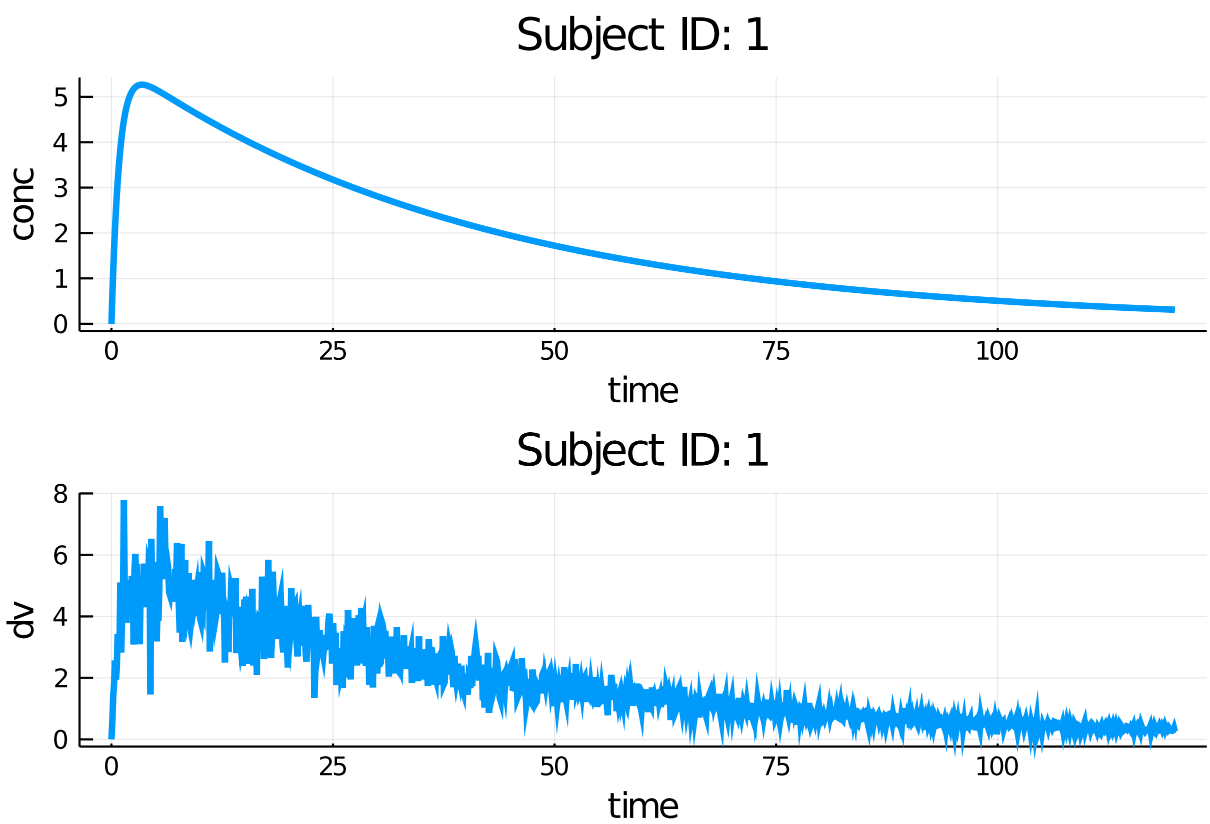
```
s1.covariates
```

```
Pumas.ConstantCovar{NamedTuple{(:isPM, :Wt),Tuple{Int64,Int64}}}((isPM = 0,
 Wt = 70))
```

(Note: defining distributions for covariates will be discussed in detail later.)

Using this one subject, `s1`, let us simulate a simple concentration time profile using the model above:

```
obs = simobs(model,s1,fixeffs,obstimes=0:0.1:120)
plot(obs)
```



1.3.2 Building Populations

Now, lets create one more subject, `s2`.

```
s2 = Subject(id=2,events=ev,covariates=(isPM=1,Wt=70))
```

```
Subject
  ID: 2
  Events: 1
  Covariates: isPM, Wt
```

If we want to simulate both `s1` and `s2` together, we need to bring these subjects together to form a `Population`. A `Population` is essentially a collection of subjects.

```
twosubjs =  Population([s1,s2])
```

```
Population
  Subjects: 2
  Covariates: isPM, Wt
```

Let's see the details of the first and the second subject

```
twosubjs[1]
```

```
Subject
  ID: 1
  Events: 1
  Covariates: isPM, Wt
```

```
twosubjs[2]
```

```
Subject
  ID: 2
  Events: 1
  Covariates: isPM, Wt
```

Now, we can simulate this `Population` of 2 subjects as below

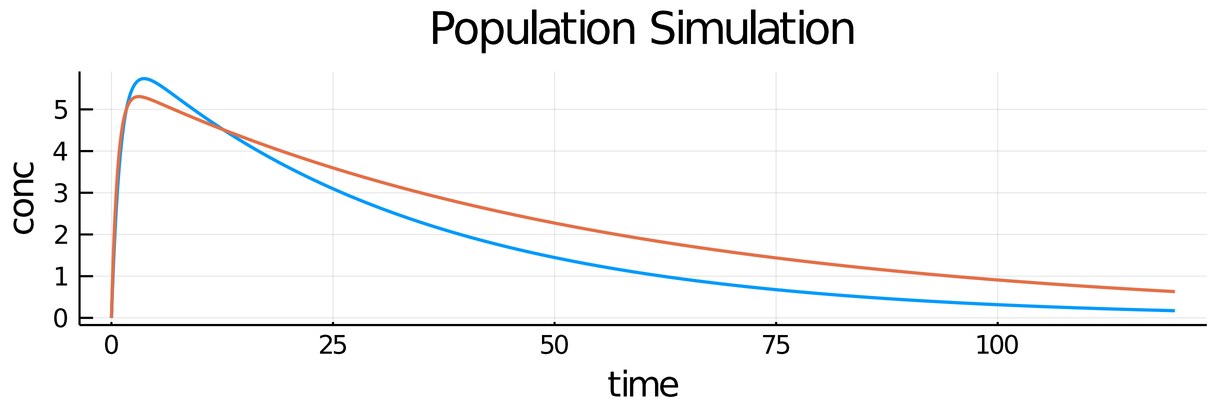
```
obs = simobs(model,twosubjs,fixeffs,obstimes=0:0.1:120)
```

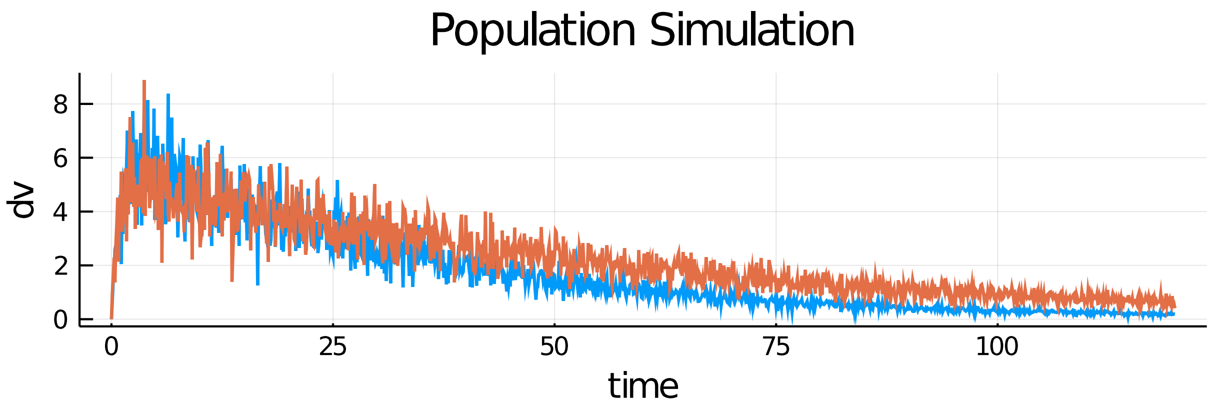
```
2-element Array{Pumas.SimulatedObservations{Pumas.Subject{Nothing,Pumas.Con
stantCovar{NamedTuple{(:isPM, :Wt),Tuple{Int64,Int64}}},Array{Pumas.Event,1
},Nothing},StepRangeLen{Float64,Base.TwicePrecision{Float64},Base.TwicePrec
```

```
ision{Float64}},NamedTuple{(:conc, :dv),Tuple{Array{Float64,1},Array{Float6
4,1}}}},1}:
 Pumas.SimulatedObservations{Pumas.Subject{Nothing,Pumas.ConstantCovar{Name
dTuple{(:isPM, :Wt),Tuple{Int64,Int64}}},Array{Pumas.Event,1},Nothing},Step
RangeLen{Float64,Base.TwicePrecision{Float64},Base.TwicePrecision{Float64}}
,NamedTuple{(:conc, :dv),Tuple{Array{Float64,1},Array{Float64,1}}}}(Subject
  ID: 1
  Events: 1
  Covariates: isPM, Wt
, 0.0:0.1:120.0, (conc = [0.0, 0.5920182875900496, 1.1275439970276668, 1.61
18023183386991, 2.049533707090369, 2.4450427919398416, 2.8022379594671207,
3.124658582542248, 3.4155267831089424, 3.6777679519283066  …  0.17676562786
67516, 0.17622858978954742, 0.1756931831850726, 0.1751594031079319, 0.17462
724462789678, 0.1740967028299049, 0.17356777281406022, 0.17304044969563348,
 0.1725147286050614, 0.1719906046879474], dv = [-2.3996761636767473e-8, 0.6
720982940199381, 1.4004185672819596, 1.5663314682074567, 2.595291024074222,
 2.7479225072534463, 2.7658956240046884, 2.652162897361175, 2.8407619840509
093, 3.6480023140598363  …  0.16920320831344973, 0.22720072719537696, 0.218
77218926629047, 0.17668727679325713, 0.182711180641346, 0.25333793415512074
, 0.1749047419484749, 0.17969352171765807, 0.1922542357873911, 0.1863963764
3803053]))
 Pumas.SimulatedObservations{Pumas.Subject{Nothing,Pumas.ConstantCovar{Name
dTuple{(:isPM, :Wt),Tuple{Int64,Int64}}},Array{Pumas.Event,1},Nothing},Step
RangeLen{Float64,Base.TwicePrecision{Float64},Base.TwicePrecision{Float64}}
,NamedTuple{(:conc, :dv),Tuple{Array{Float64,1},Array{Float64,1}}}}(Subject
  ID: 2
  Events: 1
  Covariates: isPM, Wt
, 0.0:0.1:120.0, (conc = [0.0, 0.7481966187204888, 1.395238196058453, 1.954
6169333125551, 2.438029631777846, 2.8556045059901813, 3.2161332172142996, 3
.5272174174682793, 3.7954543394560747, 4.026575797373808  …  0.640493901939
0935, 0.6393205153137747, 0.6381492796567342, 0.636980191504382, 0.63581324
7439136, 0.634648442206765, 0.6334857705256688, 0.6323252285353173, 0.63116
68123772246, 0.6300105181949518], dv = [-2.7218683323716026e-8, 0.983009801
1177977, 1.6033737055618054, 2.2904629285904865, 1.3691737371683141, 2.8361
89612043221, 3.552866158411832, 4.52255111502784, 2.1391312726920537, 4.439
617713229104  …  0.5303777624825339, 0.7397659440750053, 0.6491192230129273
, 0.7443655175538789, 0.5374973781608423, 0.4600806531591324, 0.85837985869
30989, 0.5282916511604071, 0.5745976017570928, 0.4069724018418258]))
```

When using `simobs` on more than one subject, i.e., on a `Population`, the simulation is automatically parallelized across the subejcts.

```
plot(obs)
```





Similarly, we can build a population of any number of subjects. But before we do that, let's dive into covariate generation.

1.3.3 Covariates

As was discussed earlier, a `Subject` can also be provided details regarding covariates. In the model above, there are two covariates, `isPM` which stands for *is the subject a poor metabolizer* and takes a boolean of *yes* and *no*. The second covariate is a continuous covariate where body weight `Wt` impacts both `CL` and `V`. Let us now specify covariates to a population of 10 subjects.

```
choose_covariates() = (isPM = rand([1, 0]),
                       Wt = rand(55:80))
```

```
choose_covariates (generic function with 1 method)
```

`choose_covariates` will randomly choose a `isPM` and an `Wt` between 55-80 kgs

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

```
cvs = [ choose_covariates() for i in 1:10 ]
DataFrame(cvs)
```

	isPM	Wt
	Int64	Int64
1	1	73
2	0	70
3	0	66
4	1	57
5	0	68
6	0	79
7	1	78
8	1	75
9	0	77
10	1	74

Now, we add these covariates to the population as below. The `map(f,xs)` will return the result of `f` on each element of `xs`. Let's map a function that build's a subject with the randomly chosen covariates in order to build a population:

```
pop_with_covariates = Population(map(i ->
Subject(id=i,events=ev,covariates=choose_covariates()),1:10))
```

```
Population
  Subjects: 10
  Covariates: isPM, Wt
```

Simulate into the population

```
obs = simobs(model,pop_with_covariates,fixeffs,obstimes=0:0.1:120);
```

```
10-element Array{Pumas.SimulatedObservations{Pumas.Subject{Nothing,Pumas.Co
nstantCovar{NamedTuple{(:isPM, :Wt),Tuple{Int64,Int64}}},Array{Pumas.Event,
1},Nothing},StepRangeLen{Float64,Base.TwicePrecision{Float64},Base.TwicePre
cision{Float64}},NamedTuple{(:conc, :dv),Tuple{Array{Float64,1},Array{Float
64,1}}}},1}:
 Pumas.SimulatedObservations{Pumas.Subject{Nothing,Pumas.ConstantCovar{Name
dTuple{(:isPM, :Wt),Tuple{Int64,Int64}}},Array{Pumas.Event,1},Nothing},Step
RangeLen{Float64,Base.TwicePrecision{Float64},Base.TwicePrecision{Float64}}
,NamedTuple{(:conc, :dv),Tuple{Array{Float64,1},Array{Float64,1}}}}(Subject
  ID: 1
  Events: 1
  Covariates: isPM, Wt
, 0.0:0.1:120.0, (conc = [0.0, 0.7873451499721718, 1.4845313478512498, 2.10
16255654207754, 2.6475684105095443, 3.1303095879707317, 3.5568998255604742,
 3.9336198147043087, 4.266036657693111, 4.559090557764859  …  0.21791206464
05572, 0.21727183179849974, 0.2166334797744116, 0.21599700309367492, 0.2153
6239630750378, 0.21472965399294378, 0.21409877075287237, 0.2134697412159990
4, 0.21284256003686464, 0.21221722189584216], dv = [1.5688902450312924e-8,
0.9368668206956892, 1.430898213099687, 2.4663257855492797, 3.35930537035126
87, 3.6744990088305842, 2.1034160805619857, 4.2953241841314, 4.165814348882
736, 3.8659450749577497  …  0.22578444113992496, 0.263178162859136, 0.25087
01652038241, 0.24903979785597993, 0.17174936541072328, 0.24423638161952851,
 0.1613093491783673, 0.24963082982353543, 0.2092875999007101, 0.11843594563
```

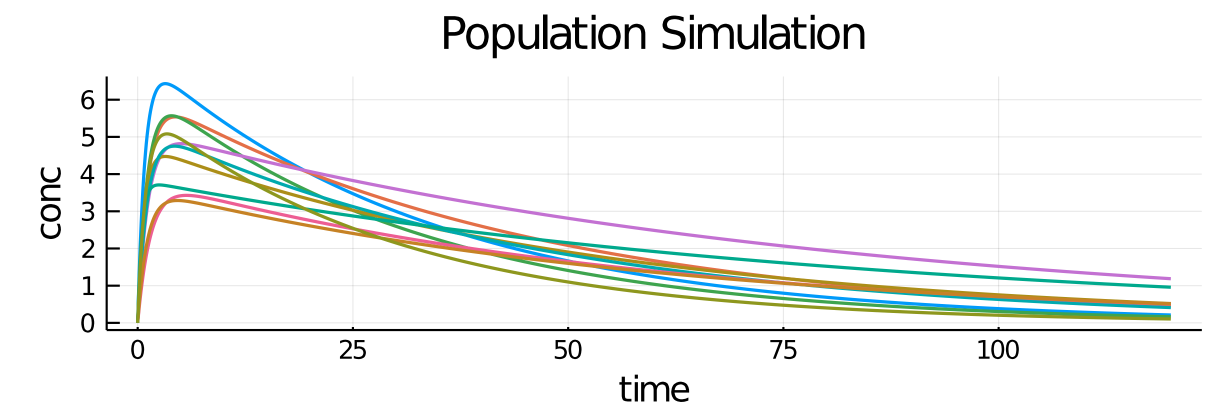
```
157775]))
 Pumas.SimulatedObservations{Pumas.Subject{Nothing,Pumas.ConstantCovar{Name
dTuple{(:isPM, :Wt),Tuple{Int64,Int64}}},Array{Pumas.Event,1},Nothing},Step
RangeLen{Float64,Base.TwicePrecision{Float64},Base.TwicePrecision{Float64}}
,NamedTuple{(:conc, :dv),Tuple{Array{Float64,1},Array{Float64,1}}}}(Subject
  ID: 2
  Events: 1
  Covariates: isPM, Wt
, 0.0:0.1:120.0, (conc = [0.0, 0.5019593591660826, 0.9614449748317604, 1.38
19598287801118, 1.766718580656869, 2.118668930165071, 2.4405199092158045, 2
.7347534719010507, 3.0036451895524308, 3.249289547031423  ...  0.454555423987
3277, 0.4535557344780326, 0.45255824350509605, 0.45156294625144566, 0.45056
983791259564, 0.449578913696646, 0.4485901688242819, 0.4476035985287755, 0.
4466191980559837, 0.44563696266435054], dv = [-8.90244015657306e-9, 0.45393
62245342933, 1.2605712418327566, 1.243762853375494, 1.7724069576556758, 2.2
18971972699213, 3.059963009764373, 2.4530578902964018, 2.7913318276327135,
 3.7355107084279933  ...  0.36320210669741104, 0.4750824730266186, 0.28384670
58377293, 0.4698820964365834, 0.3896813523781413, 0.4745006294239272, 0.502
2884994374331, 0.40971587103391277, 0.2876356543756866, 0.5657603755608475]
))
 Pumas.SimulatedObservations{Pumas.Subject{Nothing,Pumas.ConstantCovar{Name
dTuple{(:isPM, :Wt),Tuple{Int64,Int64}}},Array{Pumas.Event,1},Nothing},Step
RangeLen{Float64,Base.TwicePrecision{Float64},Base.TwicePrecision{Float64}}
,NamedTuple{(:conc, :dv),Tuple{Array{Float64,1},Array{Float64,1}}}}(Subject
  ID: 3
  Events: 1
  Covariates: isPM, Wt
, 0.0:0.1:120.0, (conc = [0.0, 0.5332321990303284, 1.0194775710537347, 1.46
27380447721843, 1.8666748951894823, 2.234636873778541, 2.5696920143976856,
2.874641367199387, 3.1520495124271997, 3.40426855863243  ...  0.1699149870124
5247, 0.16939581915991778, 0.1688782375406386, 0.16836223732738775, 0.16784
781371007898, 0.16733496189576655, 0.1668236771086453, 0.166313954590051, 0
.16580578959845987, 0.16529917740948902], dv = [8.34832775710865e-9, 0.5330
813384475555, 0.866196631250481, 1.5455853213993502, 2.3013921990485873, 1.
5982031012249927, 2.237427752704714, 2.0057415857159895, 2.321678070644651,
 2.5712516215736367  ...  0.19565863214804166, 0.1738715338993723, 0.14619188
984935644, 0.1396417499373395, 0.13014507987610904, 0.194514373624028 4, 0.1
5852975811247166, 0.18716847164092495, 0.22768058428005544, 0.1179855472292
5499]))
 Pumas.SimulatedObservations{Pumas.Subject{Nothing,Pumas.ConstantCovar{Name
dTuple{(:isPM, :Wt),Tuple{Int64,Int64}}},Array{Pumas.Event,1},Nothing},Step
RangeLen{Float64,Base.TwicePrecision{Float64},Base.TwicePrecision{Float64}}
,NamedTuple{(:conc, :dv),Tuple{Array{Float64,1},Array{Float64,1}}}}(Subject
  ID: 4
  Events: 1
  Covariates: isPM, Wt
, 0.0:0.1:120.0, (conc = [0.0, 0.41502396126367586, 0.7959420861472055, 1.1
455157014735844, 1.4662831959737612, 1.7605754182493127, 2.030538089468679,
 2.278141144562448, 2.505191991688914, 2.713357460483698  ...  1.200405691893
4582, 1.1989278052769254, 1.1974517387356243, 1.1959774901245104, 1.1945050
573060119, 1.19303443793197, 1.191565629056834 6, 1.1900986283950659, 1.1886
334337300448, 1.1871700428471261], dv = [-1.2885133053500581e-8, 0.35447098
983129527, 1.1372398666332741, 0.9594871095886857, 1.7477364831378919, 1.91
77036425054208, 2.0472707853874277, 2.398649728833799, 3.2182307462408613,
2.2140735256863264  ...  1.3177668216101053, 1.2926654902352328, 1.3244934702
957114, 0.8381721116638812, 1.0862269466160417, 0.6212908101999886, 1.10558
13491330033, 0.95970890231753, 0.6986969903690757, 1.240318526812011]))
 Pumas.SimulatedObservations{Pumas.Subject{Nothing,Pumas.ConstantCovar{Name
dTuple{(:isPM, :Wt),Tuple{Int64,Int64}}},Array{Pumas.Event,1},Nothing},Step
```

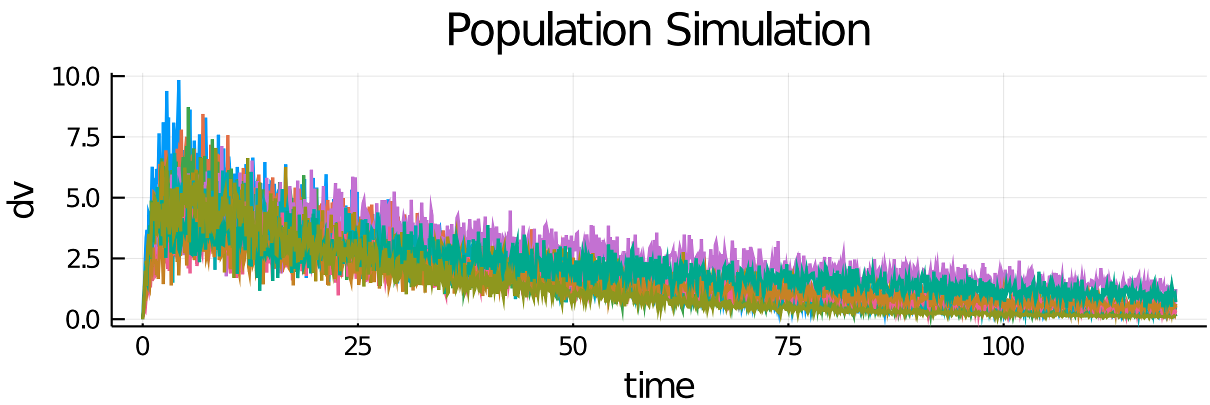
```
RangeLen{Float64,Base.TwicePrecision{Float64},Base.TwicePrecision{Float64}}
,NamedTuple{(:conc, :dv),Tuple{Array{Float64,1},Array{Float64,1}}}}(Subject
  ID: 5
  Events: 1
  Covariates: isPM, Wt
, 0.0:0.1:120.0, (conc = [0.0, 0.6029586800725552, 1.1281588316124678, 1.58
54858908392742, 1.9835706451454789, 2.3299424272282763, 2.6311794316718697,
 2.8930216303862135, 3.1204696163737684, 3.317910756265904  …  0.5282165951
761976, 0.5272374646849961, 0.5262601489280082, 0.5252846445906451, 0.52431
09483769189, 0.5233390570094435, 0.5223689672294329, 0.5214006757967035, 0.
5204341794896719, 0.5194694751053562], dv = [7.957687028004218e-9, 0.670250
397126378, 1.1063414314661493, 1.566768897989287, 2.4020164023507133, 1.803
378015075072, 1.7092707834467442, 2.3070558442852414, 3.4193470864052498, 2
.6956295971511337  …  0.6756372379841415, 0.5140946131085697, 0.50020109758
58334, 0.5124810856788442, 0.3078616857040175, 0.6631137205196289, 0.519052
8293822511, 0.4802658660625303, 0.3879348619054245, 0.38197900755514197]))
 Pumas.SimulatedObservations{Pumas.Subject{Nothing,Pumas.ConstantCovar{Name
dTuple{(:isPM, :Wt),Tuple{Int64,Int64}}},Array{Pumas.Event,1},Nothing},Step
RangeLen{Float64,Base.TwicePrecision{Float64},Base.TwicePrecision{Float64}}
,NamedTuple{(:conc, :dv),Tuple{Array{Float64,1},Array{Float64,1}}}}(Subject
  ID: 6
  Events: 1
  Covariates: isPM, Wt
, 0.0:0.1:120.0, (conc = [0.0, 0.44513485946247355, 0.8512163595450327, 1.2
215896788652105, 1.5593133783871214, 1.8671834308454607, 2.1477598566090768
, 2.4033783653780856, 2.6361766219401646, 2.8481138933674033  …  0.41912692
88518278, 0.4182327931729144, 0.41734056499018035, 0.4164502402797496, 0.41
55618150304681, 0.414675285243904, 0.41379064693434725, 0.41290789612881074
, 0.4120270288670285, 0.41114804120145754], dv = [-1.801765057721149e-8, 0.
5230743404347346, 0.9292659785833227, 1.8529793482229462, 1.687417920936237
2, 1.3620716473443193, 2.456018960826004, 2.765478603485309, 2.220540422091
992, 3.049047760173941  …  0.33926510690007033, 0.5004583215432229, 0.46823
89672724018, 0.41968848633494626, 0.37817142921239827, 0.37538019118434224,
 0.291671407743298, 0.44215199313140885, 0.44253073878501875, 0.49902514378
97656]))
 Pumas.SimulatedObservations{Pumas.Subject{Nothing,Pumas.ConstantCovar{Name
dTuple{(:isPM, :Wt),Tuple{Int64,Int64}}},Array{Pumas.Event,1},Nothing},Step
RangeLen{Float64,Base.TwicePrecision{Float64},Base.TwicePrecision{Float64}}
,NamedTuple{(:conc, :dv),Tuple{Array{Float64,1},Array{Float64,1}}}}(Subject
  ID: 7
  Events: 1
  Covariates: isPM, Wt
, 0.0:0.1:120.0, (conc = [0.0, 0.2418360152742771, 0.467758760373408, 0.678
7887254775345, 0.8758814864600662, 1.059931035931172, 1.2317733076171946, 1
.392192168140384, 1.5419213169571961, 1.681645408378093  …  0.5068232584972
944, 0.5059582400272963, 0.5050946979890208, 0.5042326298825487, 0.50337203
32133799, 0.502512905492432, 0.501655244236041, 0.5007990469659619, 0.49994
43112093672, 0.4990910344988486], dv = [-3.529439744005928e-9, 0.1849179585
6827153, 0.523832815195017, 0.43670632166081036, 0.8692490852883867, 1.2048
334640869638, 1.4396336258984421, 1.2444961911105261, 1.6607926335542313, 1
.307318918673845  …  0.35784823860470116, 0.4250449231104276, 0.52652152805
75782, 0.5986152368900748, 0.38160154516839884, 0.6124324958765818, 0.33340
00502730591, 0.4122886325560158, 0.24457157943742713, 0.34042087433370444])
)
 Pumas.SimulatedObservations{Pumas.Subject{Nothing,Pumas.ConstantCovar{Name
dTuple{(:isPM, :Wt),Tuple{Int64,Int64}}},Array{Pumas.Event,1},Nothing},Step
RangeLen{Float64,Base.TwicePrecision{Float64},Base.TwicePrecision{Float64}}
,NamedTuple{(:conc, :dv),Tuple{Array{Float64,1},Array{Float64,1}}}}(Subject
  ID: 8
```

```
  Events: 1
  Covariates: isPM, Wt
, 0.0:0.1:120.0, (conc = [0.0, 0.29907275667610683, 0.5724189999531937, 0.8
222107265037927, 1.0504366226272985, 1.2589167329093933, 1.4493199243856496
, 1.6231711989215358, 1.7818671567451898, 1.9266899627868255  ...  0.51898168
9812847, 0.518136770171651, 0.517293226033698, 0.5164510551644372, 0.515610
255332977, 0.514770824312083, 0.5139327598781785, 0.5130960598113458, 0.512
2607218953239, 0.5114267439175106], dv = [-2.2499737666044564e-8, 0.2850786
3432044034, 0.4975200017732515, 0.9035389938547675, 0.8703779250478206, 1.1
82783393874378, 1.5317571382094368, 1.4495413596185989, 1.3242852289813392,
 1.9622370525130937  ...  0.43691999468222725, 0.509469932829203, 0.477167216
6207264, 0.6121726625703893, 0.45572183191806354, 0.6260301718518901, 0.684
9335230352868, 0.48066777933801386, 0.5064898987195948, 0.63721628683293]))
 Pumas.SimulatedObservations{Pumas.Subject{Nothing,Pumas.ConstantCovar{Name
dTuple{(:isPM, :Wt),Tuple{Int64,Int64}}},Array{Pumas.Event,1},Nothing},Step
RangeLen{Float64,Base.TwicePrecision{Float64},Base.TwicePrecision{Float64}}
,NamedTuple{(:conc, :dv),Tuple{Array{Float64,1},Array{Float64,1}}}}(Subject
  ID: 9
  Events: 1
  Covariates: isPM, Wt
, 0.0:0.1:120.0, (conc = [0.0, 0.7242586282089044, 1.3101682079972872, 1.78
39984609692388, 2.1670294796381895, 2.476500037500731, 2.7263842684555053,
2.9279859314201677, 3.090483675209919, 3.2212989317958867  ...  0.96863670338
73954, 0.9675175998791006, 0.9663997899431035, 0.9652832729245637, 0.964168
0483037971, 0.9630541151071895, 0.9619414689149811, 0.9608301077349677, 0.9
597200301702286, 0.958611234814019], dv = [3.201826870030563e-9, 0.56534576
87176978, 1.2608971250470438, 1.7036123183854885, 1.7125805185509904, 1.573
060659074231, 3.0459094832556435, 2.636037926533856, 3.861419245157185, 2.4
829418296167383  ...  0.9987065009331054, 0.7465736133905885, 0.8549548316644
384, 1.1006652044581302, 0.5071992775996672, 1.2495287463755032, 1.03697054
58409253, 1.0136004660042544, 1.060025350777078, 0.7088758754583255]))
 Pumas.SimulatedObservations{Pumas.Subject{Nothing,Pumas.ConstantCovar{Name
dTuple{(:isPM, :Wt),Tuple{Int64,Int64}}},Array{Pumas.Event,1},Nothing},Step
RangeLen{Float64,Base.TwicePrecision{Float64},Base.TwicePrecision{Float64}}
,NamedTuple{(:conc, :dv),Tuple{Array{Float64,1},Array{Float64,1}}}}(Subject
  ID: 10
  Events: 1
  Covariates: isPM, Wt
, 0.0:0.1:120.0, (conc = [0.0, 0.5640454752408991, 1.0702424349552648, 1.52
43371403769008, 1.931503021071218, 2.2964048514805424, 2.62324012605657, 2.
915789747090548, 3.1774679322783834, 3.411339000458852  ...  0.10810655270108
753, 0.1077443608771798, 0.10738338258082372, 0.10702361388210611, 0.106665
05087727673, 0.10630768968874803, 0.10595152646509493, 0.10559655738105526,
 0.10524277863752934, 0.10489018646158045], dv = [-1.128523549498443e-9, 0.
42360808625376845, 1.2856365689345117, 1.9923795971910472, 1.36274806965007
82, 2.110860950775249, 3.0586667604153606, 3.508079408068059, 2.88734346230
63723, 2.7583527315442185  ...  0.11820163161398772, 0.09900759666775354, 0.1
0180886723165089, 0.08689930692636534, 0.070617415142437, 0.074628272487769
42, 0.10982665750215427, 0.13958361648012046, 0.10267880256853129, 0.091881
37033632932]))
```

and visualize the output

```
plot(obs)
```





1.4 Multiple dose example

The additional dosage regimen controls of the NMTRAN format are available in `DosageRegimen`. For example, `ii` defines the "interdose interval", or the time distance between two doses, while `addl` defines how many additional times to repeat a dose. Thus, let's define a dose of 100 that's repeated 7 times at 24 hour intervals:

```
md =  DosageRegimen(100,ii=24,addl=6)
```

	time	cmt	amt	evid	ii	addl	rate	duration	ss
	Float64	Int64	Float64	Int8	Float64	Int64	Float64	Float64	Int8
1	0.0	1	100.0	1	24.0	6	0.0	0.0	0

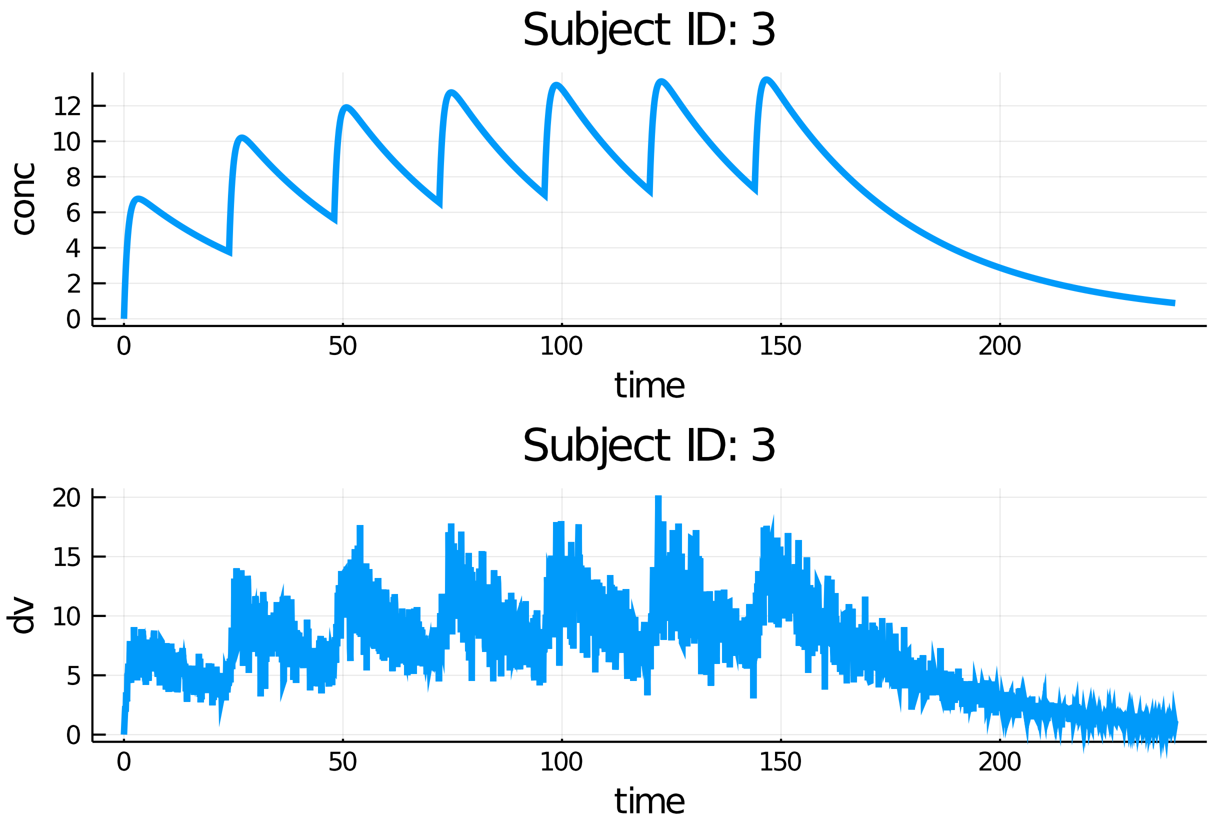
Let's create a new subject, `s3` with this dosage regimen:

```
s3 = Subject(id=3,events=md, covariates=(isPM=0,Wt=70))
```

```
Subject
  ID: 3
  Events: 7
  Covariates: isPM, Wt
```

and see the results:

```
obs = simobs(model, s3, fixeffs,obstimes=0:0.1:240)
plot(obs)
```



1.5 Combining dosage regimens

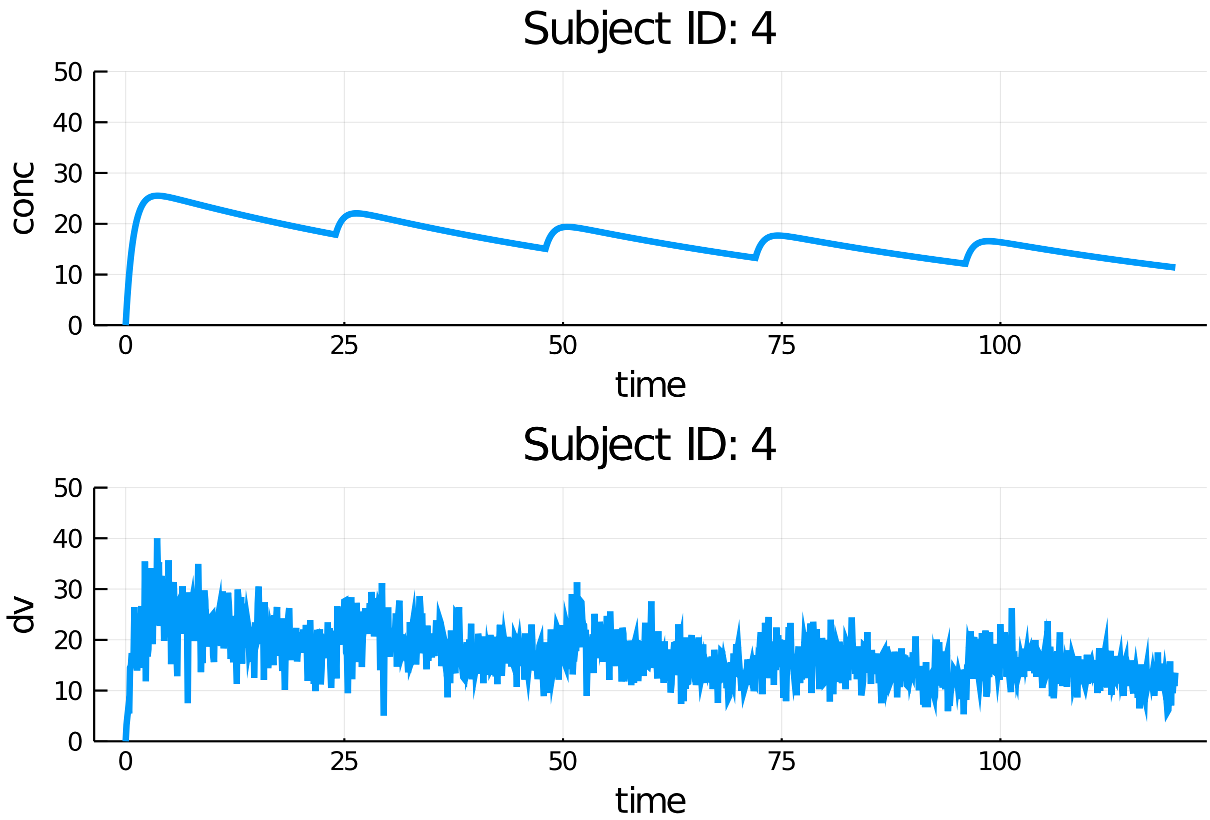
We can also combine dosage regimens to build a more complex regimen. Recall from the introduction that using arrays will build the element-wise combinations. Thus let's build a dose of 500 into compartment 1 at time 0, and 7 doses into compartment 1 of 100 spaced by 24 hours:

```
ldmd = DosageRegimen([500,100],cmt=1, time=[0,24], addl=[0,6],ii=[0,24])
```

	time	cmt	amt	evid	ii	addl	rate	duration	ss
	Float64	Int64	Float64	Int8	Float64	Int64	Float64	Float64	Int8
1	0.0	1	500.0	1	0.0	0	0.0	0.0	0
2	24.0	1	100.0	1	24.0	6	0.0	0.0	0

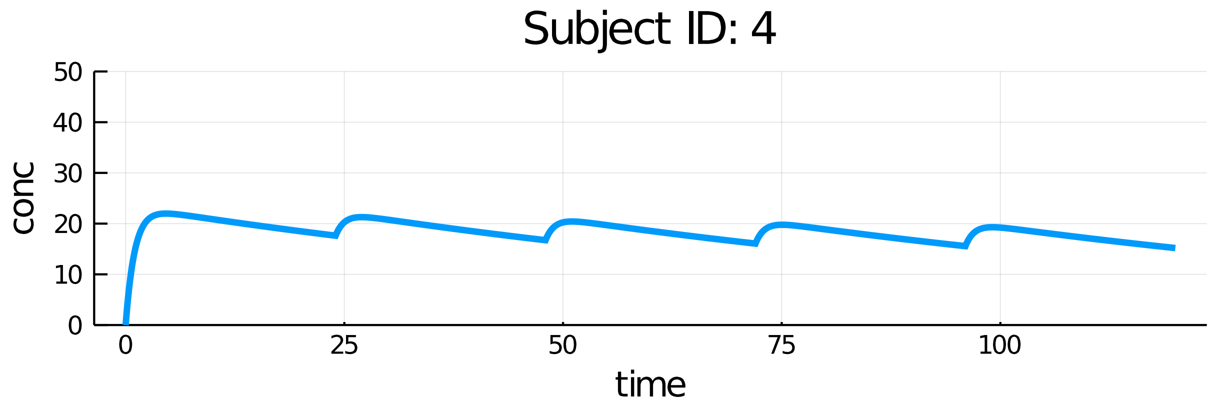
Let's see if this result matches our intuition:

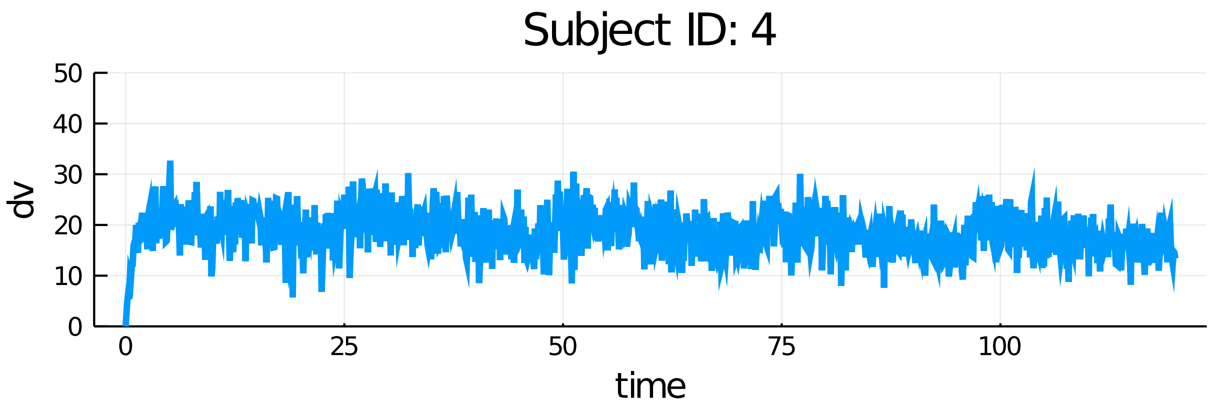
```
s4 = Subject(id=4, events=ldmd, covariates=(isPM=0,Wt=70))
obs = simobs(model, s4, fixeffs,obstimes=0:0.1:120)
plot(obs, ylims=(0,50))
```



Another way to build complex dosage regiments is to combine previously constructed regimens into a single regimen. For example:

```
e1 = DosageRegimen(500,cmt=1, time=0, addl=0,ii=0)
e2 = DosageRegimen(100,cmt=1, time=24, addl=6,ii=24)
evs = DosageRegimen(e1,e2)
obs = simobs(model, s4, fixeffs,obstimes=0:0.1:120)
plot(obs, ylims=(0,50))
```





is the same regimen as before.

Putting these ideas together, we can define a population where individuals with different covariates undergo different regimens, and simulate them all together with automatic parallelism:

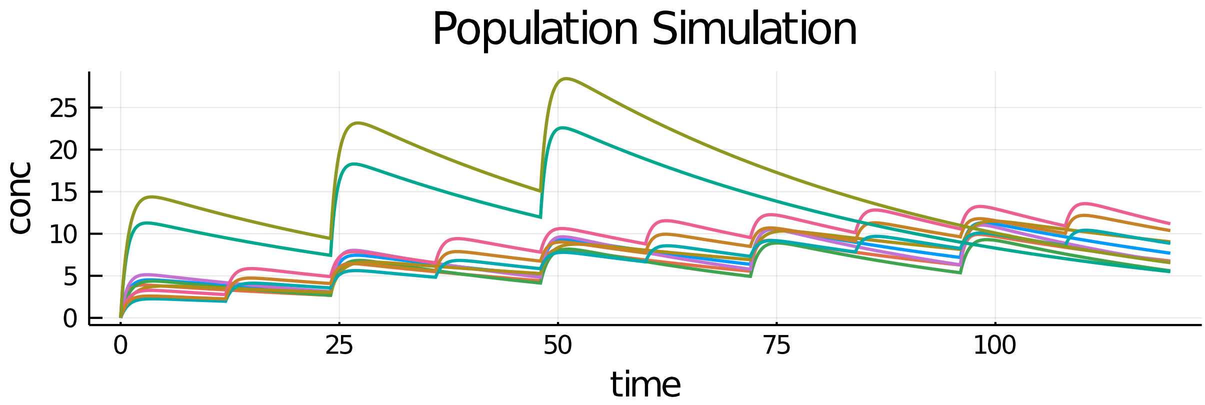
```
e1 = DosageRegimen(100, ii=24, addl=6)
e2 = DosageRegimen(50,  ii=12, addl=13)
e3 = DosageRegimen(200, ii=24, addl=2)
```

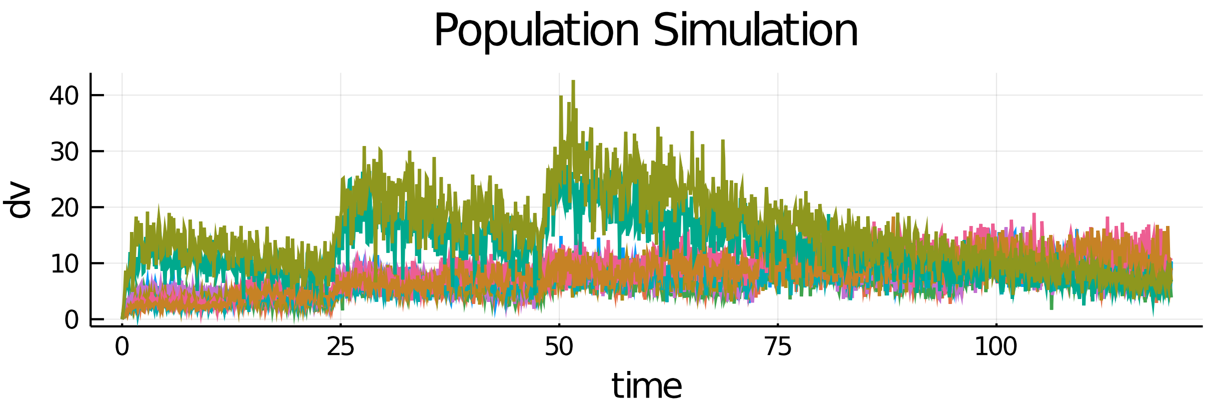
	time	cmt	amt	evid	ii	addl	rate	duration	ss
	Float64	Int64	Float64	Int8	Float64	Int64	Float64	Float64	Int8
1	0.0	1	200.0	1	24.0	2	0.0	0.0	0

```
pop1 = Population(map(i -> Subject(id=i,events=e1,covariates=choose_covariates()),1:5))
pop2 = Population(map(i -> Subject(id=i,events=e2,covariates=choose_covariates()),6:8))
pop3 = Population(map(i -> Subject(id=i,events=e3,covariates=choose_covariates()),9:10))
pop = Population(vcat(pop1,pop2,pop3))
```

```
Population
  Subjects: 10
  Covariates: isPM, Wt
```

```
obs = simobs(model,pop,fixeffs,obstimes=0:0.1:120)
plot(obs)
```





1.6 Defining Infusions

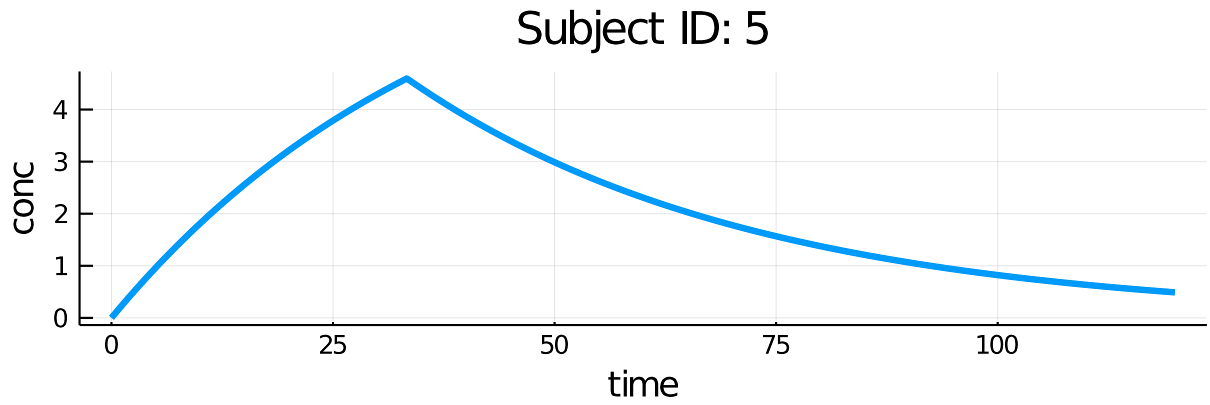
As specified in the NMTRAN format, an infusion is a dosage which is defined as having a non-zero positive rate at which the drug enters the system. Let's define a single infusion dose of total amount 100 with a rate of 3 into the second compartment:

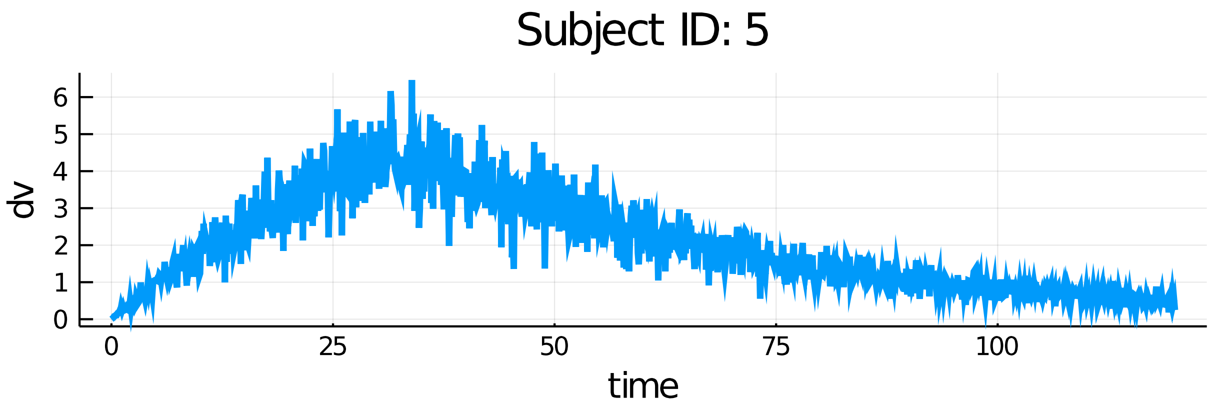
```
inf = DosageRegimen(100, rate=3, cmt=2)
```

	time	cmt	amt	evid	ii	addl	rate	duration	ss
	Float64	Int64	Float64	Int8	Float64	Int64	Float64	Float64	Int8
1	0.0	2	100.0	1	0.0	0	3.0	33.3333	0

Now let's simulate a subject undergoing this treatment strategy:

```
s5 = Subject(id=5, events=inf, covariates=(isPM=0,Wt=70))
obs = simobs(model, s5, fixeffs, obstimes=0:0.1:120)
plot(obs)
```





1.7 Final Note on Julia Programming

Note that all of these functions are standard Julia functions, and thus standard Julia programming constructions can be utilized to simplify the construction of large populations. We already demonstrated the use of `map` and a comprehension, but we can also make use of constructs like `for` loops.

1.8 Conclusion

This tutorial shows the tools for generating populations of infinite complexity, defining covariates and dosage regimens on the fly and simulating the results of the model.