Fitting models in Pumas

August 11, 2020

1 Fitting a PK model

In this tutorial we will go through the steps required to fit a model in Pumas.jl.

1.1 The dosage regimen

We start by simulating a population from a two compartment model with oral absorption, and then we show how to fit and do some model validation using the fit output.

```
using Pumas, Plots, Random
Random.seed!(0);
```

```
MersenneTwister(UInt32[0x00000000], Random.DSFMT.DSFMT_state(Int32[74839879
7, 1073523691, -1738140313, 1073664641, -1492392947, 1073490074, -162528183
9, 1073254801, 1875112882, 1073717145  ...  943540191, 1073626624, 1091647724
, 1073372234, -1273625233, -823628301, 835224507, 991807863, 382, 0]), [0.0
, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0  ...  0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0], UInt128[0x00000000000000000000000000000000, 0x000
00000000000000000000000000000, 0x00000000000000000000000000000000, 0x000000
00000000000000000000000000, 0x00000000000000000000000000000000, 0x000000000
00000000000000000000000, 0x00000000000000000000000000000000, 0x000000000000
00000000000000000000, 0x00000000000000000000000000000000, 0x000000000000000
00000000000000000  ...  0x00000000000000000000000000000000, 0x000000000000000
00000000000000000, 0x00000000000000000000000000000000, 0x000000000000000000
00000000000000, 0x00000000000000000000000000000000, 0x000000000000000000000
00000000000, 0x00000000000000000000000000000000, 0x000000000000000000000000
00000000, 0x00000000000000000000000000000000, 0x000000000000000000000000000
00000], 1002, 0)
```

The dosage regimen is an dose of 100 into `Depot` at time 0, followed by two additional (`addl=2`) doses every fourth hour

```
repeated_dose_regimen = DosageRegimen(100, time=0, ii=4, addl=2)
```

	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	4.0	2	0.0	0.0	0

As ususal, let's define a function to choose body weight randomly per subject

```
choose_covariates() = (Wt = rand(55:80),)
```

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

and generate a population of subjects with a random weight generated from the covariate function above

```
pop = Population(map(i -> Subject(id = i,
                                  events = repeated_dose_regimen,
                                  observations =(dv=Float64[],),
                                  covariates = choose_covariates()),
                                  1:24))
```

```
Population
  Subjects: 24
  Covariates: Wt
  Observables: dv
```

We now have 24 subjects equipped with a weight and a dosage regimen.

1.2 The PK model of drug concentration and elimination

To simulate a data set and attempt to estimate the data generating parameters, we have to set up the actual pharmacokinetics (PK) model and simulate the data. We use the closed form model called `Depots1Central1Periph1` which is a two compartment model with first order absorption. This requires `CL`, `Vc`, `Ka`, `Vp`, and `Q` to be defined in the `@pre`-block, since they define the rates of transfer between (and out of) the compartments

```
mymodel = @model begin
  @param   begin
    cl ∈ RealDomain(lower = 0.0, init = 1.0)
    tv ∈ RealDomain(lower = 0.0, init = 10.0)
    ka ∈ RealDomain(lower = 0.0, init = 1.0)
    q  ∈ RealDomain(lower = 0.0, init = 0.5)
    Ω   ∈ PDiagDomain(init = [0.9,0.07, 0.05])
    σ_prop ∈ RealDomain(lower = 0,init = 0.03)
  end

  @random begin
    η ~ MvNormal(Ω)
  end

  @covariates Wt

  @pre begin
    CL = cl * (Wt/70)^0.75 * exp(η[1])
    Vc = tv * (Wt/70) * exp(η[2])
    Ka = ka * exp(η[3])
    Vp = 30.0
    Q  = q
  end

  @dynamics Depots1Central1Periph1

  @derived begin
      cp := @. 1000*(Central / Vc) # We use := because we don't want simobs to store the
variable
      dv ~ @. Normal(cp, abs(cp)*σ_prop)
    end
end
```

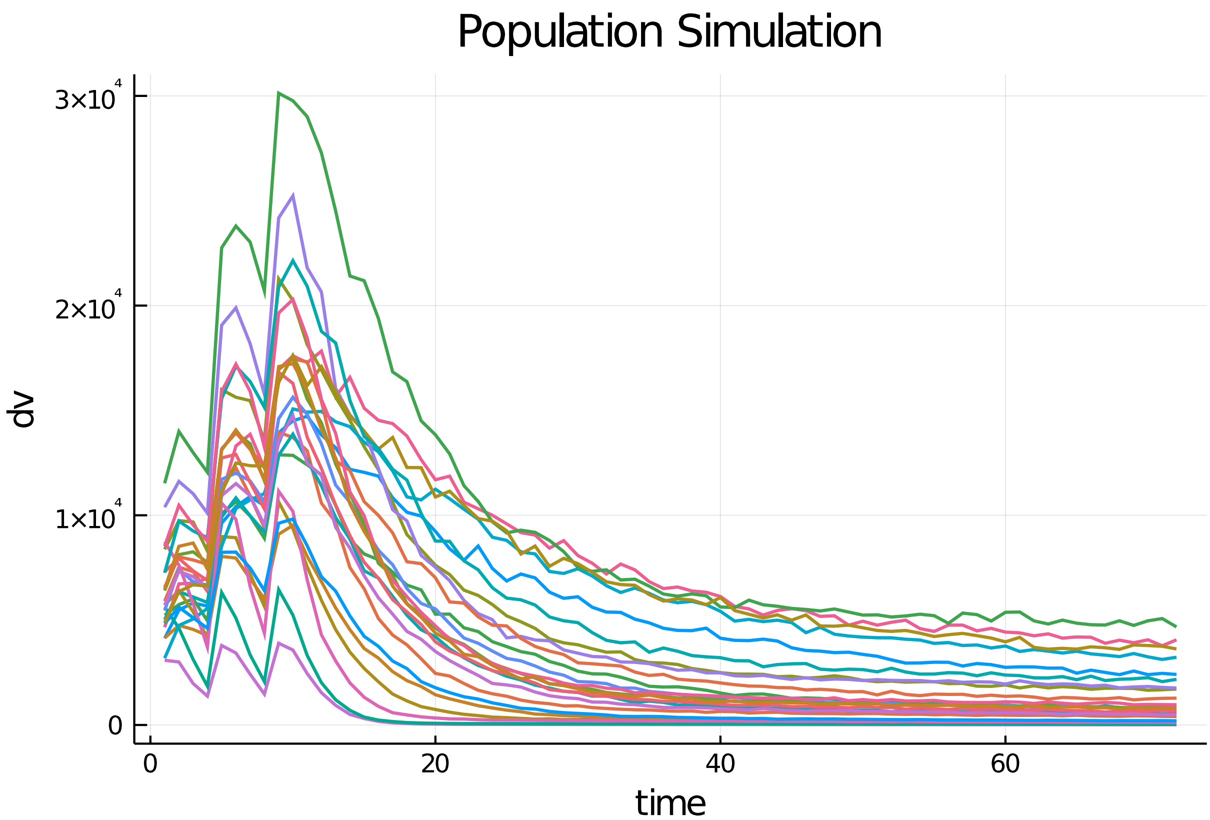
```
PumasModel
  Parameters: cl, tv, ka, q, Ω, σ_prop
  Random effects: η
  Covariates: Wt
  Dynamical variables: Depot, Central, Peripheral
  Derived: dv
  Observed: dv
```

Some parameters are left free by giving them domains in the `@param`-block, and one PK parameter (the volume of distribution of the peripheral compartment) is fixed to 20.0.

1.3 Simulating the indiviual observations

The `simobs` function is used to simulate individual time series. We input the model, the population of Subjects that currently only have dosage regimens and covariates, the parameter vector and the times where we want to simulate. Since we have a proportional error model we avoid observations at time zero to avoid degenerate distributions of the dependent variable. The problem is, that if the concentration is zero the variance in distribution of the explained variable will also be zero. Let's use the default parameters, as set in the `@param`-block, and simulate the data

```
param = init_param(mymodel)
obs = simobs(mymodel, pop, param, obstimes=1:1:72)
plot(obs)
```



1.4 Fitting the model

To fit the model, we use the `fit` function. It requires a model, a population, a named tuple of parameters and a likelihood approximation method.

```
result = fit(mymodel, Subject.(obs), param, Pumas.FOCEI())
```

```
Iter     Function value   Gradient norm
     0     9.509662e+03     4.974134e+02
 * time: 4.00543212890625e-5
     1     9.509087e+03     5.235690e+01
 * time: 0.3228719234466553
     2     9.508605e+03     1.141497e+01
 * time: 0.474902868270874
     3     9.508415e+03     3.304623e+00
 * time: 0.5766148567199707
     4     9.508362e+03     3.055397e+00
 * time: 0.6750218868255615
     5     9.508092e+03     7.897585e+00
 * time: 0.7867660522460938
     6     9.507916e+03     7.176151e+00
 * time: 0.9132959842681885
     7     9.507822e+03     1.852197e+00
 * time: 1.0123238563537598
     8     9.507810e+03     1.416720e+00
 * time: 1.1087579727172852
     9     9.507794e+03     1.847344e+00
 * time: 1.2256789207458496
    10     9.507765e+03     3.646199e+00
 * time: 1.3224499225616455
    11     9.507727e+03     4.127054e+00
 * time: 1.4156739711761475
    12     9.507701e+03     2.259724e+00
 * time: 1.5351550579071045
    13     9.507695e+03     4.267957e-01
 * time: 1.631192922592163
    14     9.507695e+03     1.529258e-01
 * time: 1.7254600524902344
    15     9.507695e+03     1.458034e-01
 * time: 1.8348259925842285
    16     9.507695e+03     1.374069e-01
 * time: 1.922973871231079
    17     9.507694e+03     1.983480e-01
 * time: 2.019239902496338
    18     9.507694e+03     1.832846e-01
 * time: 2.1073968410491943
    19     9.507694e+03     8.495916e-02
 * time: 2.226806879043579
    20     9.507694e+03     1.434653e-02
 * time: 2.3228230476379395
    21     9.507694e+03     1.142185e-03
 * time: 2.4226839542388916
    22     9.507694e+03     1.139690e-03
 * time: 2.569169044494629
    23     9.507694e+03     4.554493e-02
 * time: 2.6666598320007324
FittedPumasModel

Successful minimization:                      false
```

```
Likelihood approximation:                   Pumas.FOCEI
Log-likelihood value:                        -9507.6936
Number of subjects:                                  24
Number of parameters:          Fixed      Optimized
                                   0              6
Observation records:          Active        Missing
    dv:                         1728              0
    Total:                      1728              0

---------------------
           Estimate
---------------------
cl          0.87552
tv          9.9608
ka          0.97115
q           0.4989
Ω_1,_1        0.99073
Ω_2,_2        0.061093
Ω_3,_3        0.037914
σ_prop      0.029516
---------------------
```

Of course, we started the fitting at the true parameters, so let us define our own starting parameters, and fit based on those values

```
alternative_param = (
    cl = 0.5,
    tv = 9.0,
    ka = 1.3,
    q  = 0.3,
    Ω  = Diagonal([0.18,0.04, 0.03]),
    σ_prop = 0.04)

fit(mymodel, Subject.(obs), alternative_param, Pumas.FOCEI())
```

```
Iter     Function value   Gradient norm
     0     2.171648e+04     4.024499e+04
 * time: 4.00543212890625e-5
     1     1.374758e+04     1.846694e+04
 * time: 0.2893030643463135
     2     1.062678e+04     4.399382e+03
 * time: 0.4878559112548828
     3     1.035016e+04     1.328733e+03
 * time: 0.666126012802124
     4     1.028054e+04     1.295589e+03
 * time: 0.8571720123291016
     5     1.021065e+04     1.841083e+03
 * time: 1.1695630550384521
     6     9.949460e+03     6.144939e+03
 * time: 1.432831048965454
     7     9.595062e+03     3.239578e+03
 * time: 1.7042200565338135
     8     9.569951e+03     4.301083e+02
 * time: 1.9668200016021729
     9     9.568758e+03     1.318693e+02
 * time: 2.1882638931274414
    10     9.567892e+03     2.141681e+02
 * time: 2.471191883087158
    11     9.561440e+03     7.908701e+02
 * time: 2.7361960411071777
```

```
   12     9.552319e+03     1.086506e+03
* time: 2.983299970626831
   13     9.545273e+03     6.937697e+02
* time: 3.264453887939453
   14     9.543505e+03     1.872858e+02
* time: 3.5000860691070557
   15     9.543171e+03     5.062953e+01
* time: 3.7264058589935303
   16     9.542870e+03     1.524584e+02
* time: 3.947721004486084
   17     9.542118e+03     3.573733e+02
* time: 4.200770854949951
   18     9.540396e+03     6.307445e+02
* time: 4.450094938278198
   19     9.536735e+03     9.350193e+02
* time: 4.693774938583374
   20     9.530476e+03     1.095649e+03
* time: 4.973253011703491
   21     9.522366e+03     8.881636e+02
* time: 5.239232063293457
   22     9.516800e+03     3.390603e+02
* time: 5.493597984313965
   23     9.515844e+03     3.149418e+01
* time: 5.758120059967041
   24     9.515776e+03     3.246223e+01
* time: 5.995143890380859
   25     9.515691e+03     4.957381e+01
* time: 6.22666597366333
   26     9.515502e+03     5.679289e+01
* time: 6.5137529373168945
   27     9.515158e+03     3.770801e+01
* time: 6.764739990234375
   28     9.514635e+03     1.984292e+01
* time: 7.014884948730469
   29     9.514115e+03     5.875926e+01
* time: 7.296093940734863
   30     9.513726e+03     7.526339e+01
* time: 7.53692102432251
   31     9.513454e+03     5.726155e+01
* time: 7.779139995574951
   32     9.513370e+03     2.345084e+01
* time: 8.05837607383728
   33     9.513346e+03     1.495845e+01
* time: 8.291872024536133
   34     9.513305e+03     3.106059e+01
* time: 8.52426290512085
   35     9.513241e+03     6.757386e+01
* time: 8.769797086715698
   36     9.513059e+03     1.342898e+02
* time: 9.022075891494751
   37     9.512653e+03     2.234116e+02
* time: 9.264693975448608
   38     9.511771e+03     3.222847e+02
* time: 9.509826898574829
   39     9.510273e+03     3.589024e+02
* time: 9.76872992515564
   40     9.508599e+03     2.512825e+02
* time: 10.01951003074646
   41     9.507861e+03     8.450873e+01
```

```
 * time: 10.262578964233398
    42     9.507785e+03     1.614494e+01
 * time: 10.53338098526001
    43     9.507780e+03     3.390312e+00
 * time: 10.765872955322266
    44     9.507779e+03     1.391292e+00
 * time: 10.988980054855347
    45     9.507779e+03     1.398817e+00
 * time: 11.24770188331604
    46     9.507779e+03     1.402850e+00
 * time: 11.470831871032715
    47     9.507779e+03     1.407965e+00
 * time: 11.69336199760437
    48     9.507778e+03     1.411078e+00
 * time: 11.91158390045166
    49     9.507777e+03     1.405185e+00
 * time: 12.161458015441895
    50     9.507773e+03     1.368683e+00
 * time: 12.398000955581665
    51     9.507763e+03     1.726587e+00
 * time: 12.632549047470093
    52     9.507745e+03     2.360911e+00
 * time: 12.906507015228271
    53     9.507719e+03     2.370152e+00
 * time: 13.135526895523071
    54     9.507700e+03     1.436938e+00
 * time: 13.369709014892578
    55     9.507695e+03     4.604145e-01
 * time: 13.649132013320923
    56     9.507694e+03     2.545239e-01
 * time: 13.880127906799316
    57     9.507694e+03     8.748370e-02
 * time: 14.103055000305176
    58     9.507694e+03     8.781354e-02
 * time: 14.313617944717407
    59     9.507694e+03     8.779529e-02
 * time: 14.517945051193237
    60     9.507694e+03     8.780567e-02
 * time: 14.64756989479065
FittedPumasModel

Successful minimization:                      false

Likelihood approximation:               Pumas.FOCEI
Log-likelihood value:                     -9507.694
Number of subjects:                              24
Number of parameters:         Fixed      Optimized
                                  0              6
Observation records:         Active        Missing
    dv:                        1728              0
    Total:                     1728              0

---------------------
           Estimate
---------------------
cl          0.87564
tv          9.9607
ka          0.9711
q           0.49891
```

```
Ω_1,_1        0.99095
Ω_2,_2        0.061185
Ω_3,_3        0.037614
σ_prop      0.029516
---------------------
```

and we see that the estimates are essentially the same up to numerical noise.

To augment the basic information listed when we print the results, we can use `infer` to provide RSEs and confidence intervals

```
infer(result)
```

```
Calculating: variance-covariance matrix. Done.
Asymptotic inference results

Successful minimization:                     false

Likelihood approximation:              Pumas.FOCEI
Log-likelihood value:                   -9507.6936
Number of subjects:                             24
Number of parameters:         Fixed      Optimized
                                  0              6
Observation records:         Active        Missing
    dv:                        1728              0
    Total:                     1728              0

---------------------------------------------------------------------
          Estimate           SE                      95.0% C.I.
---------------------------------------------------------------------
cl         0.87552         0.17807             [0.52651 ;  1.2245  ]
tv         9.9608          0.50239             [8.9761  ; 10.945   ]
ka         0.97115         0.038954            [0.8948  ;  1.0475  ]
q          0.4989          0.00103             [0.49689 ;  0.50092 ]
Ω_1,_1       0.99073         0.27442             [0.45288 ;  1.5286  ]
Ω_2,_2       0.061093        0.014388            [0.032894;  0.089292]
Ω_3,_3       0.037914        0.011946            [0.014501;  0.061328]
σ_prop     0.029516        0.00051089          [0.028515;  0.030517]
---------------------------------------------------------------------
```

So as we observed earlier, the parameters look like they have sensible values. The confidence intervals are a bit wide, and especially so for the random effect variability parameters. To see how we can use simulation to better understand the statistical properties of our model, we can simulate a much larger population and try again

```
pop_big = Population(map(i -> Subject(id = i,
                                  events = repeated_dose_regimen,
                                  observations =(dv=Float64[],),
                                  covariates = choose_covariates()),
                                  1:100))
obs_big = simobs(mymodel, pop_big, param, obstimes=1:1:72)
result_big = fit(mymodel, Subject.(obs_big), param, Pumas.FOCEI())
```

```
Iter     Function value   Gradient norm
     0     3.923167e+04     1.061584e+03
 * time: 6.508827209472656e-5
     1     3.923104e+04     5.439570e+01
 * time: 1.5991668701171875
     2     3.923062e+04     5.493562e+01
 * time: 2.5061988830566406
```

```
     3     3.922981e+04     7.061316e+00
 * time: 3.1530919075012207
     4     3.922971e+04     6.257481e+00
 * time: 3.5873680114746094
     5     3.922917e+04     8.285576e+00
 * time: 4.0845348834991455
     6     3.922915e+04     2.215932e+00
 * time: 4.767730951309204
     7     3.922913e+04     2.501836e+00
 * time: 5.57011604309082
     8     3.922910e+04     5.866463e+00
 * time: 6.473873853683472
     9     3.922909e+04     4.914102e+00
 * time: 7.347162961959839
    10     3.922908e+04     1.679291e+00
 * time: 8.229390859603882
    11     3.922908e+04     2.096004e-01
 * time: 8.847455978393555
    12     3.922908e+04     3.402351e-02
 * time: 9.259455919265747
    13     3.922908e+04     2.982270e-02
 * time: 9.729099035263062
    14     3.922908e+04     2.982270e-02
 * time: 10.487254858016968
    15     3.922908e+04     2.982270e-02
 * time: 11.344712018966675
    16     3.922908e+04     2.982270e-02
 * time: 12.16309905052185
```

```
infer(result_big)
```

```
Calculating: variance-covariance matrix. Done.
Asymptotic inference results

Successful minimization:                      true

Likelihood approximation:              Pumas.FOCEI
Log-likelihood value:                   -39229.078
Number of subjects:                            100
Number of parameters:         Fixed      Optimized
                                  0              6
Observation records:         Active        Missing
    dv:                        7200              0
    Total:                     7200              0

--------------------------------------------------------------------
          Estimate          SE                      95.0% C.I.
--------------------------------------------------------------------
cl         0.93233          0.093072          [0.74991 ;  1.1147  ]
tv        10.39             0.28913           [9.8235  ; 10.957   ]
ka         1.0051           0.022993          [0.96008 ;  1.0502  ]
q          0.50059          0.00052754        [0.49955 ;  0.50162 ]
Ω_1,_1       0.99768          0.15308           [0.69765 ;  1.2977  ]
Ω_2,_2       0.07766          0.015107          [0.048051;  0.10727 ]
Ω_3,_3       0.049499         0.010411          [0.029093;  0.069906]
σ_prop     0.029905         0.00023157        [0.029451;  0.030358]
--------------------------------------------------------------------
```

This time we see similar estimates, but much narrower confidence intervals across the board.

1.5 Estimating a misspecified model

To explore some of the diagnostics tools available in Julia, we can try to set up a model that does not fit out data generating process. This time we propose a one compartent model. The problem with estimating a one compartment model when the data comes from a two compartment model, is that we cannot capture the change in slope on the concentration profile you get with a two compartment model. This means that even if we can capture the model fit someone well on average, we should expect to see systematic trends in the residual diagnostics post estimation.

```
mymodel_misspec = @model begin
  @param   begin
    cl ∈ RealDomain(lower = 0.0, init = 1.0)
    tv ∈ RealDomain(lower = 0.0, init = 20.0)
    ka ∈ RealDomain(lower = 0.0, init = 1.0)
    Ω  ∈ PDiagDomain(init = [0.12, 0.05, 0.08])
    σ_prop ∈ RealDomain(lower = 0, init = 0.03)
  end

  @random begin
    η ~ MvNormal(Ω)
  end

  @pre begin
    CL = cl * (Wt/70)^0.75 * exp(η[1])
    Vc = tv * (Wt/70) * exp(η[2])
    Ka = ka * exp(η[3])
  end
  @covariates Wt

  @dynamics Depots1Central1

  @derived begin
      cp = @. 1000*(Central / Vc)
      dv ~ @. Normal(cp, abs(cp)*σ_prop)
    end
end

PumasModel
  Parameters: cl, tv, ka, Ω, σ_prop
  Random effects: η
  Covariates: Wt
  Dynamical variables: Depot, Central
  Derived: cp, dv
  Observed: cp, dv

alternative_param_no_q = (
    cl = 0.5,
    tv = 9.0,
    ka = 1.3,
    Ω  = Diagonal([0.18,0.04, 0.03]),
    σ_prop = 0.04)

result_misspec = fit(mymodel_misspec, Subject.(obs), alternative_param_no_q,
Pumas.FOCEI())

Iter     Function value   Gradient norm
     0     1.130666e+05     2.034877e+05
 * time: 3.910064697265625e-5
```

```
    1     2.587757e+04     2.631157e+04
* time: 0.24605798721313477
    2     2.252342e+04     1.917207e+04
* time: 0.5298690795898438
    3     1.742887e+04     7.806532e+03
* time: 0.7762081623077393
    4     1.586746e+04     3.869815e+03
* time: 0.9934890270233154
    5     1.514158e+04     1.626030e+03
* time: 1.1863300800323486
    6     1.491812e+04     6.143775e+02
* time: 1.404879093170166
    7     1.486465e+04     4.559947e+02
* time: 1.5935490131378174
    8     1.485667e+04     4.548559e+02
* time: 1.802685022354126
    9     1.485453e+04     4.529243e+02
* time: 1.9846131801605225
   10     1.485029e+04     4.470285e+02
* time: 2.2058281898498535
   11     1.484016e+04     4.302327e+02
* time: 2.4101920127868652
   12     1.481713e+04     3.885750e+02
* time: 2.647118091583252
   13     1.477276e+04     4.540622e+02
* time: 2.8553991317749023
   14     1.470854e+04     4.794530e+02
* time: 3.0642170906066895
   15     1.464361e+04     3.845968e+02
* time: 3.2304911613464355
   16     1.460286e+04     1.716652e+02
* time: 3.4344100952148438
   17     1.459619e+04     8.137585e+01
* time: 3.6015191078186035
   18     1.459564e+04     8.232241e+01
* time: 3.7881951332092285
   19     1.459533e+04     8.310168e+01
* time: 3.9598610401153564
   20     1.459522e+04     8.314237e+01
* time: 4.1117939949035645
   21     1.459360e+04     8.253116e+01
* time: 4.303216218948364
   22     1.459079e+04     7.990532e+01
* time: 4.480956077575684
   23     1.458301e+04     7.043052e+01
* time: 4.692941188812256
   24     1.456850e+04     6.365034e+01
* time: 4.8764941692352295
   25     1.454808e+04     5.721254e+01
* time: 5.078487157821655
   26     1.453536e+04     2.934679e+01
* time: 5.241946220397949
   27     1.453260e+04     1.249869e+01
* time: 5.433487176895142
   28     1.453239e+04     1.445128e+01
* time: 5.5779900550842285
   29     1.453238e+04     1.444180e+01
* time: 5.75766921043396
   30     1.453238e+04     1.416957e+01
```

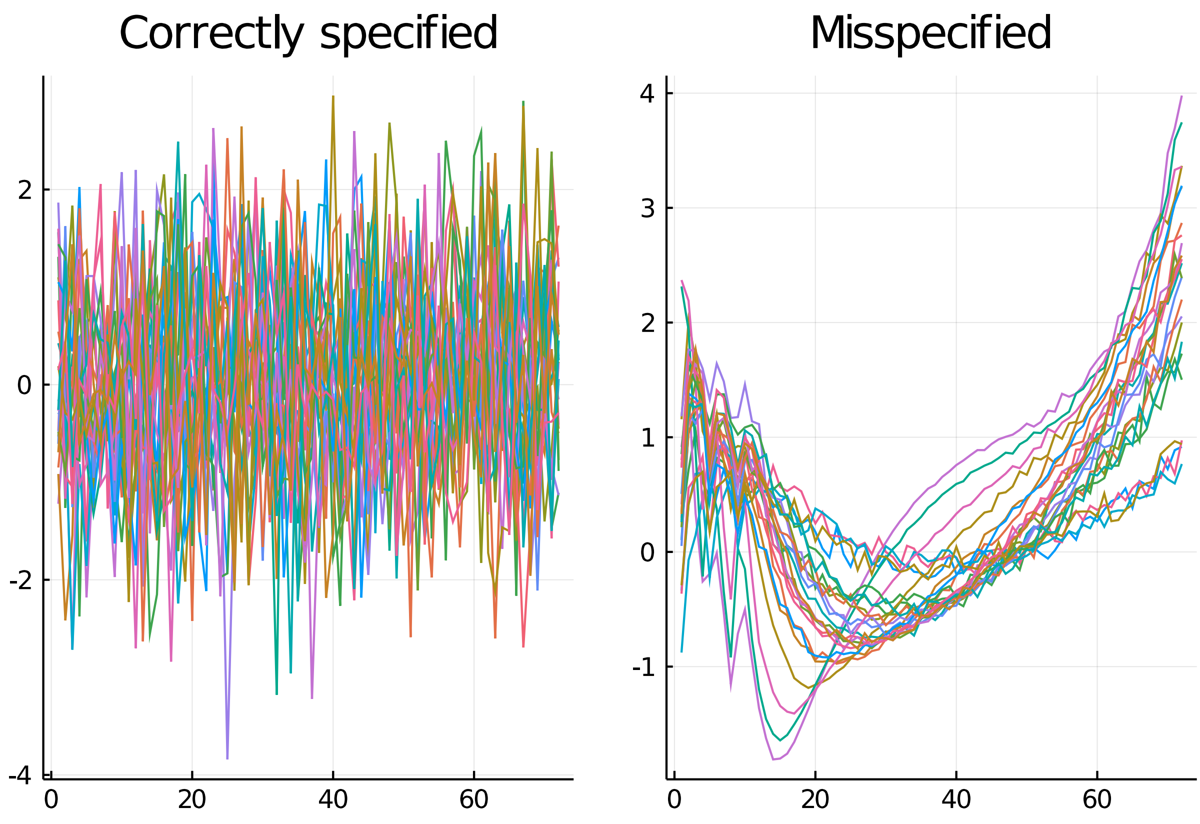
```
* time: 5.914283037185669
   31     1.453237e+04     1.384491e+01
* time: 6.068180084228516
   32     1.453237e+04     1.345241e+01
* time: 6.241929054260254
   33     1.453235e+04     1.278907e+01
* time: 6.398565053939819
   34     1.453231e+04     1.168377e+01
* time: 6.592168092727661
   35     1.453220e+04     9.925444e+00
* time: 6.753125190734863
   36     1.453193e+04     1.017005e+01
* time: 6.941609144210815
   37     1.453132e+04     9.562097e+00
* time: 7.107763051986694
   38     1.453020e+04     7.132143e+00
* time: 7.274438142776489
   39     1.452882e+04     6.237787e+00
* time: 7.512144088745117
   40     1.452773e+04     4.886911e+00
* time: 7.784556150436401
   41     1.452721e+04     2.309835e+00
* time: 8.00527811050415
   42     1.452709e+04     8.448814e-01
* time: 8.164470195770264
   43     1.452708e+04     1.028132e+00
* time: 8.3490571975708
   44     1.452708e+04     1.062585e+00
* time: 8.512262105941772
   45     1.452708e+04     1.067032e+00
* time: 8.723161220550537
   46     1.452708e+04     1.072589e+00
* time: 8.946467161178589
   47     1.452708e+04     1.075766e+00
* time: 9.074393033981323
   48     1.452708e+04     1.081692e+00
* time: 9.251603126525879
   49     1.452708e+04     1.089409e+00
* time: 9.402245044708252
   50     1.452708e+04     1.100283e+00
* time: 9.555345058441162
   51     1.452708e+04     1.111684e+00
* time: 9.736768007278442
   52     1.452707e+04     1.114455e+00
* time: 9.897705078125
   53     1.452707e+04     1.080389e+00
* time: 10.094945192337036
   54     1.452705e+04     1.087978e+00
* time: 10.250735998153687
   55     1.452703e+04     1.115299e+00
* time: 10.411971092224121
   56     1.452701e+04     6.293728e-01
* time: 10.595460176467896
   57     1.452700e+04     4.329917e-01
* time: 10.752167224884033
   58     1.452700e+04     3.560297e-01
* time: 10.92929720878601
   59     1.452700e+04     3.347408e-01
* time: 11.058487176895142
```

```
    60     1.452700e+04     3.332461e-01
 * time: 11.180670022964478
    61     1.452700e+04     3.309391e-01
 * time: 11.362521171569824
    62     1.452700e+04     3.289458e-01
 * time: 11.596994161605835
    63     1.452700e+04     3.262512e-01
 * time: 11.764774084091187
    64     1.452700e+04     3.228271e-01
 * time: 11.985912084579468
    65     1.452700e+04     3.177641e-01
 * time: 12.123960018157959
    66     1.452700e+04     3.097423e-01
 * time: 12.292476177215576
    67     1.452700e+04     5.004012e-01
 * time: 12.430918216705322
    68     1.452700e+04     8.731792e-01
 * time: 12.58197021484375
    69     1.452699e+04     1.396165e+00
 * time: 12.76498818397522
    70     1.452698e+04     1.994401e+00
 * time: 12.932569026947021
    71     1.452696e+04     2.426608e+00
 * time: 13.108297109603882
    72     1.452691e+04     2.240644e+00
 * time: 13.248215198516846
    73     1.452686e+04     8.847438e-01
 * time: 13.424806118011475
    74     1.452683e+04     2.401597e-01
 * time: 13.58080506324768
    75     1.452683e+04     6.924838e-01
 * time: 13.734705209732056
    76     1.452682e+04     7.426297e-01
 * time: 13.913239002227783
    77     1.452682e+04     4.989830e-01
 * time: 14.056345224380493
    78     1.452682e+04     1.387091e-01
 * time: 14.23645305633545
    79     1.452681e+04     7.360207e-02
 * time: 14.384446144104004
    80     1.452681e+04     1.312669e-01
 * time: 14.537137031555176
    81     1.452681e+04     1.092161e-01
 * time: 14.696864128112793
    82     1.452681e+04     4.168175e-02
 * time: 14.83450698852539
    83     1.452681e+04     9.372777e-03
 * time: 14.984457015991211
    84     1.452681e+04     2.615442e-02
 * time: 15.205247163772583
    85     1.452681e+04     2.321991e-02
 * time: 15.329478025436401
    86     1.452681e+04     1.010919e-02
 * time: 15.460186004638672
    87     1.452681e+04     9.626878e-04
 * time: 15.611956119537354
FittedPumasModel

Successful minimization:                      true
```

```
Likelihood approximation:                    Pumas.FOCEI
Log-likelihood value:                         -14526.813
Number of subjects:                                   24
Number of parameters:               Fixed      Optimized
                                        0              5
Observation records:               Active        Missing
    dv:                              1728              0
    Total:                           1728              0

--------------------
           Estimate
--------------------
cl          1.2266
tv         24.739
ka     279250.0
Ω_1,_1        0.42901
Ω_2,_2        0.12159
Ω_3,_3        0.04714
σ_prop      0.49129
--------------------
```

First off, the absorption flow parameters `ka` is quite off the charts, so that would be a warning sign off the bat, but let us try to use a tool in the toolbox to asses the fit: the weighted residuals. We get these by using the `wresiduals` function

```
wres = wresiduals(result)
wres_misspec = wresiduals(result_misspec)
p1 = plot([w.wres.dv for w in wres], title="Correctly specified", legend=false)
p2 = plot([w.wres.dv for w in wres_misspec], title = "Misspecified", legend=false)
plot(p1, p2)
```



The weighted residuals should be standard normally distributed with throughout the time domain. We see that this is the case for the correctly specified model, but certainly not for the misspecified model. That latter has a very clear pattern in time. This comes from the fact that the one compartment model is not able to capture the change in slope as time progresses, so it can never accurately capture the curves generated by a two compartment model.

1.5.1 Conclusion

This tutorial showed how to use fitting in Pumas.jl based on a simulated data set. There are many more models and simulation experiments to explore. Please try out `fit` on your own data and model, and reach out if further questions or problems come up.