METRUM
RESEARCH GROUP

mrgsolve: Simulation Examples

mrgsolve Workshop
March 12, 2016
San Diego, CA

Our focus is on `mrgsolve` issues

- What you need to do to get information into the model
- How to carry out different simulations
- For now, we'll overlook
  - Number of subjects in a simulation
  - Number of simulation replicates
  - How to turn your simulated output into a really effective table or plot

Agenda

1. Read in some NONMEM output
2. Fixed effect simulation from final estimates
3. Fixed effect simulation from posterior
4. Population simulation from final estimates
5. Population simulation from posterior

Load libraries

```
library(mrgsolve)
library(dplyr)
library(knitr)
library(readr)
library(magrittr)
library(ggplot2)
loadNamespace("metrumrg")
```

```
## <environment: namespace:metrumrg>
```

A population PK model

```
mod <- mread("poppk", "models")
mod
```

```
.
.
. -------- mrgsolve model object (unix) --------
.   Project: /Users/kyleb/CTS/script/models
.   source:        poppk.cpp
.   shared object: ae656192e2c4 (loaded)
.
.   compile date:  03/12 10:44
.   Time:          start: 0 end: 24 delta: 1
.   >              add: <none>
.   >              tscale: 1
.
.   Compartments:  GUT CENT PERIPH [3]
.   Parameters:    WT SEX EGFR BMI ALT BLACK
.   >              FORM FBIO THETA1 THETA2 THETA3 THETA4
.   >              THETA5 THETA6 THETA7 THETA8 THETA9 THETA10
.   >              THETA11 THETA12 THETA13 THETA14 THETA15 [23]
.   Omega:         4x4
.   Sigma:         2x2
.
.   Solver:        atol: 1e-08 rtol: 1e-08
.   >              maxsteps: 2000 hmin: 0 hmax: 0
```

$MAIN

```
mod %>% blocks(MAIN)
```

```
.
. Model file: poppk.cpp
.
. $MAIN
. _F(1) = 1;
. if(FORM==2) _F(1) = FBIO;
. double LTVCL = THETA1 + THETA6*log(BMI/25) + THETA8*SEX + THETA7*log(EGFR/100
. double LTVVC = THETA2 + THETA9*log(BMI/25) + THETA10*SEX;
. double LTVVP = THETA3 + THETA11*log(BMI/25);
. double LTVQ =  THETA4;
. double LTVKA = THETA5;
. double CL =   exp(LTVCL + ETA(1));
. double VC =   exp(LTVVC);
. double KA =   exp(LTVKA + ETA(3));
. double Q =    exp(LTVQ );
. double VP =   exp(LTVVP + ETA(2));
```

Parameters

```
param(mod)
```

```
.
.  Model parameters (N=23):
.  name     value     . name     value
.  ALT      0.5       | THETA14  2.22
.  BLACK    0         | THETA15  0.72
.  BMI      20        | THETA2   1.6
.  EGFR     100       | THETA3   4.34
.  FBIO     1         | THETA4   1.24
.  FORM     1         | THETA5   -0.078
.  SEX      0         | THETA6   0.366
.  THETA1   0.57      | THETA7   0.472
.  THETA10  -0.0638   | THETA8   0.0216
.  THETA11  0.793     | THETA9   0.48
.  THETA12  4.61      | WT       70
.  THETA13  3.82      | .        .
```

Read in the posterior

```
post <-
  read_table("nonmem/1001/1001.ext", skip=1) %>%
  filter(ITERATION > 0)
```

```
post[1:5,1:7] %>% as.data.frame
```

```
.   ITERATION   THETA1  THETA2  THETA3  THETA4   THETA5   THETA6
. 1         1 0.706956 1.98185 4.21065 1.18204 0.194938 0.977403
. 2         2 0.521959 1.99340 4.25382 1.18211 0.479422 0.937604
. 3         3 0.699798 2.01382 4.26981 1.18384 0.400331 0.760408
. 4         4 0.599968 2.04206 4.31422 1.18312 0.496337 0.863302
. 5         5 0.792689 2.04283 4.29784 1.19621 0.487024 0.575212
```

To just get the final estimates, take `ITERATION` -1E9 from 1001.ext

Get the median and update model object

```
mpost <-
  post %>%
  summarise_each(funs(median),2:ncol(post))
```

```
mod %<>% param(mpost)
```

```
mpost[1:6] %>% as.data.frame
```

```
.     THETA1  THETA2 THETA3   THETA4    THETA5    THETA6
. 1 0.695519 2.05737 4.2685 1.202385 0.4428255 0.7489165
```

Simulate a low/high doses for low/normal eGFR

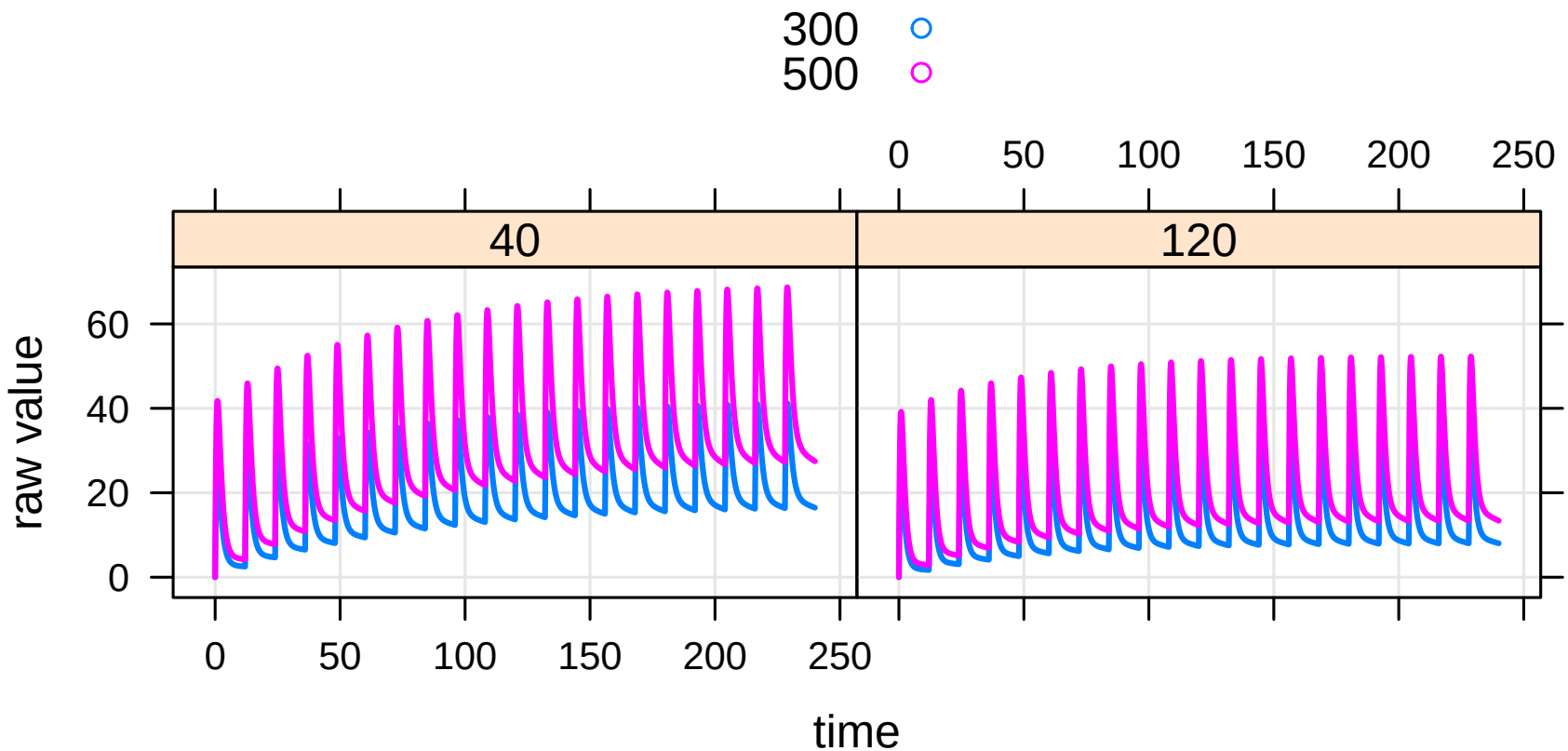
```
data <-
  expand.ev(amt=c(300,500),
            ii=12, addl=19,
            EGFR=c(40,120))

data %<>% mutate(dose=amt)

out <-
  mod %>%
  data_set(data) %>%
  drop.re %>%
  carry.out(EGFR,dose) %>%
  Req(CP) %>%
  mrgsim(end=240,delta=0.1)
```

Plot

```
plot(out, CP~time|factor(EGFR),
     groups=dose,scales="same",auto.key=TRUE)
```



Now, let's simulate from the posterior

Parameters are in columns, iterations are in rows

```
post[1:6,1:6] %>% as.data.frame
```

```
.   ITERATION   THETA1  THETA2  THETA3  THETA4   THETA5
. 1         1 0.706956 1.98185 4.21065 1.18204 0.194938
. 2         2 0.521959 1.99340 4.25382 1.18211 0.479422
. 3         3 0.699798 2.01382 4.26981 1.18384 0.400331
. 4         4 0.599968 2.04206 4.31422 1.18312 0.496337
. 5         5 0.792689 2.04283 4.29784 1.19621 0.487024
. 6         6 0.677799 2.04260 4.37175 1.20389 0.308509
```

Simulate 5 replicates

```
out <- lapply(1:5, function(i) {
  mod %>%
    param(slice(post,i)) %>% # <--- ***
    data_set(data) %>%
    drop.re %>%
    Req(CP) %>%
    carry.out(dose,EGFR) %>%
    mrgsim(end=240,delta=0.25) %>%
    mutate(irep=i)

}) %>% bind_rows
```

The output

```
head(out,3) %>% as.data.frame
```

```
.   ID time dose EGFR       CP irep
. 1  1 0.00  300   40  0.00000    1
. 2  1 0.00  300   40  0.00000    1
. 3  1 0.25  300   40 12.27817    1
```

```
tail(out,3) %>% as.data.frame
```

```
.   ID   time dose EGFR       CP irep
. 1  4 239.50  500  120 11.09787    5
. 2  4 239.75  500  120 11.03365    5
. 3  4 240.00  500  120 10.96999    5
```

Now, parallelize the simulation

```
library(parallel)
options(mc.cores=4)
RNGkind("L'Ecuyer-CMRG")
set.seed(1556)
mc.reset.stream()
```

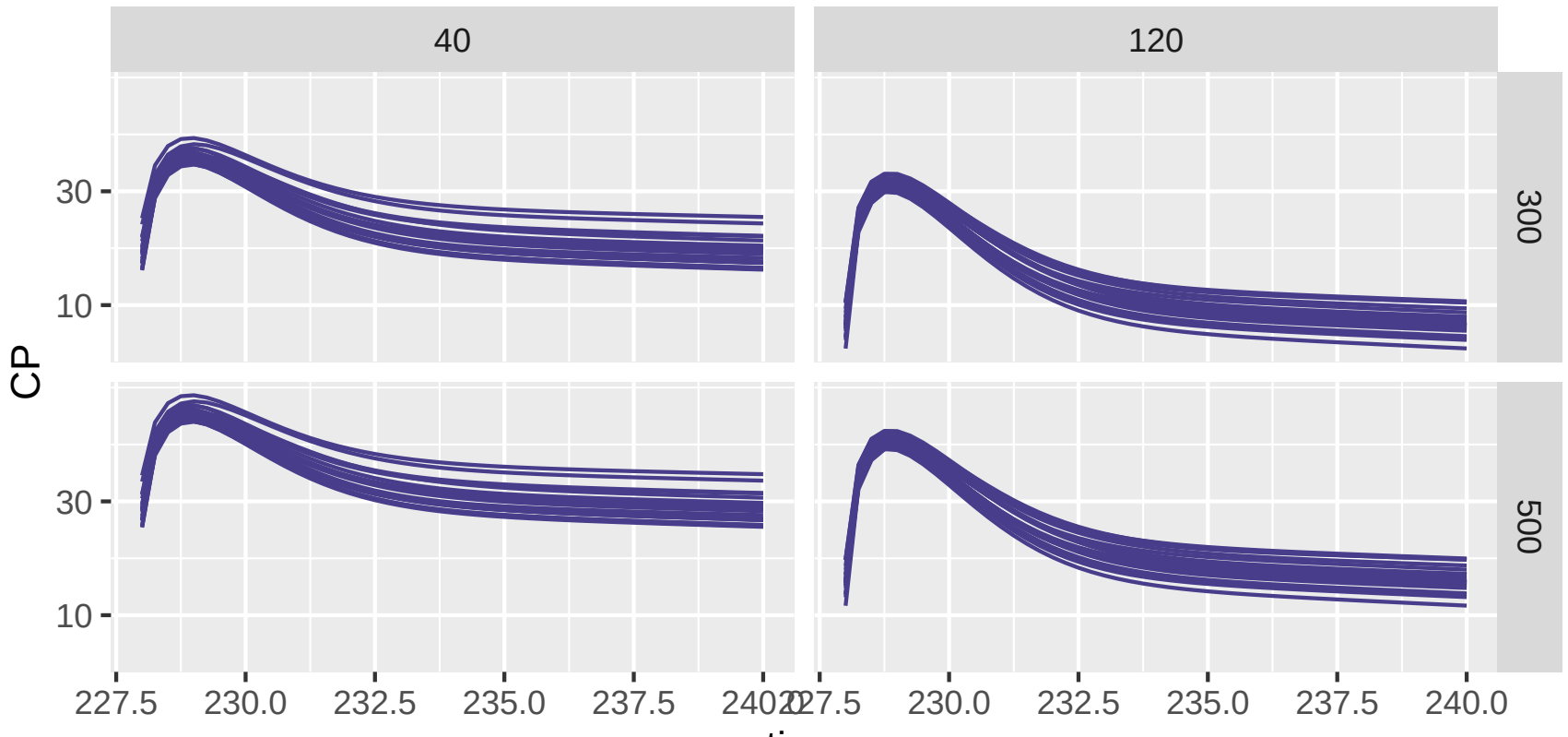
Use `doParallel` on Windows (example at the end of deck)

Use mclapply rather than lapply

```
out <- mclapply(1:20, function(i) {
  mod %>%
    param(slice(post,i)) %>%
    data_set(data) %>%
    drop.re %>%
    Req(CP) %>%
    carry.out(dose,EGFR) %>%
    mrgsim(end=240,delta=0.25) %>%
    mutate(irep=i)
}) %>% bind_rows
```

Plot

```
out %>% filter(time >= 228) %>%
  ggplot(. ,aes(time,CP,group=irep)) +
  geom_line(col="darkslateblue") +
  scale_y_continuous(trans="log10", breaks=c(3,10,30)) +
  facet_grid(dose~EGFR)
```



Population simulation

- Let's deal with `IIV` for now
- Note that `$OMEGA` is 4x4
- We have filled this in with zeros for now and it needs updating

```
mod %>% blocks(OMEGA)
```

```
.
. Model file: poppk.cpp
.
. $OMEGA
. 0 0 0 0
```

Update $OMEGA (use omat)

Note that mpost contains OMEGA1.1, OMEGA2.1, OMEGA2.2, etc...

```
omega <- as_bmat(mpost, "OMEGA")
mod %<>% omat(omega)
omat(mod)
```

```
. $...
.              [,1]         [,2]        [,3]         [,4]
. 1:   0.12152550 -0.01149300 -0.1281720  0.01072455
. 2:  -0.01149300  0.18292500 -0.1671945 -0.01397405
. 3:  -0.12817200 -0.16719450  0.7275515  0.01362860
. 4:   0.01072455 -0.01397405  0.0136286  0.05274095
```

A single population simulation (no RUV)

```
data <- expand.ev(ID=1:20, amt=1000, cmt=2,
                  rate=200, ii=24, addl=9)
```

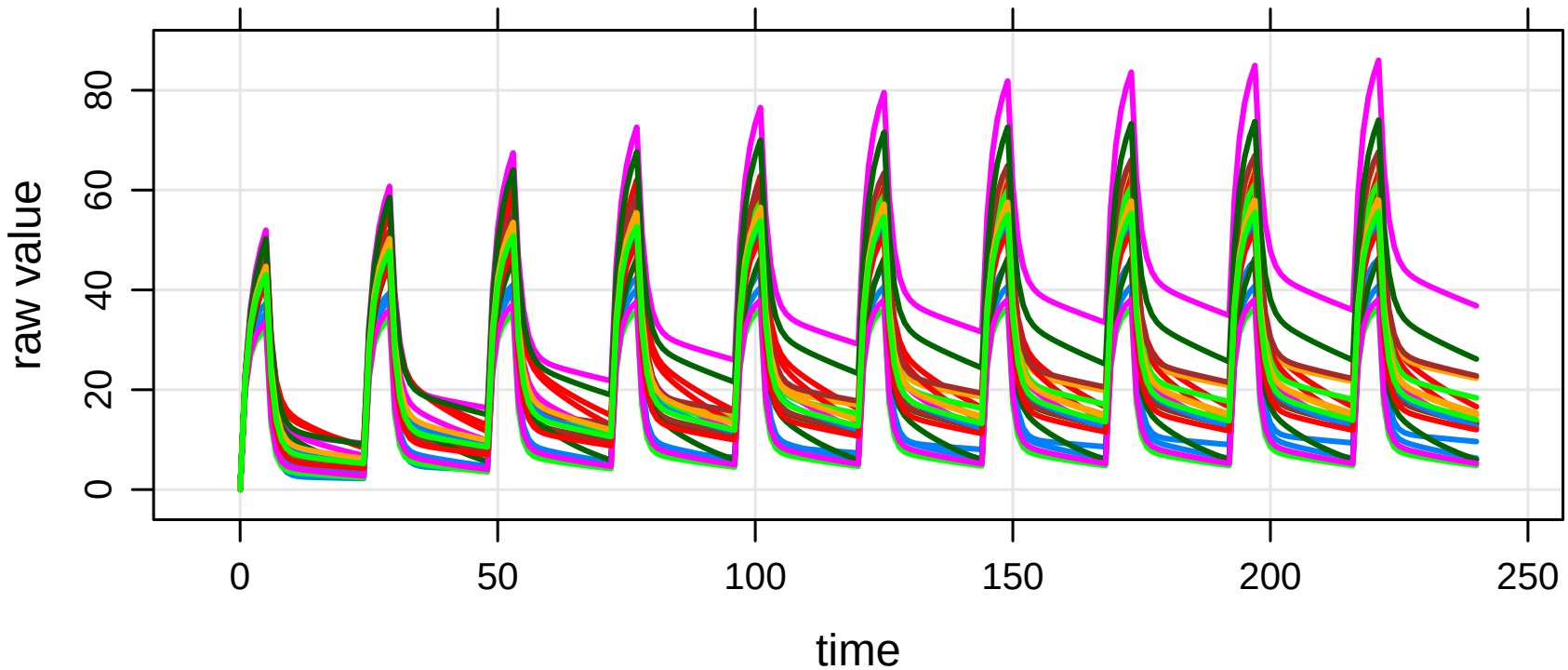
```
head(data, n=4)
```

```
.   ID  amt cmt rate ii addl evid time
. 1  1 1000   2  200 24    9    1    0
. 2  2 1000   2  200 24    9    1    0
. 3  3 1000   2  200 24    9    1    0
. 4  4 1000   2  200 24    9    1    0
```

```
set.seed(878)
out <-
  mod %>%
  data_set(data) %>%
  mrgsim(end=240,Req="CP")
```

Result

```
plot(out)
```

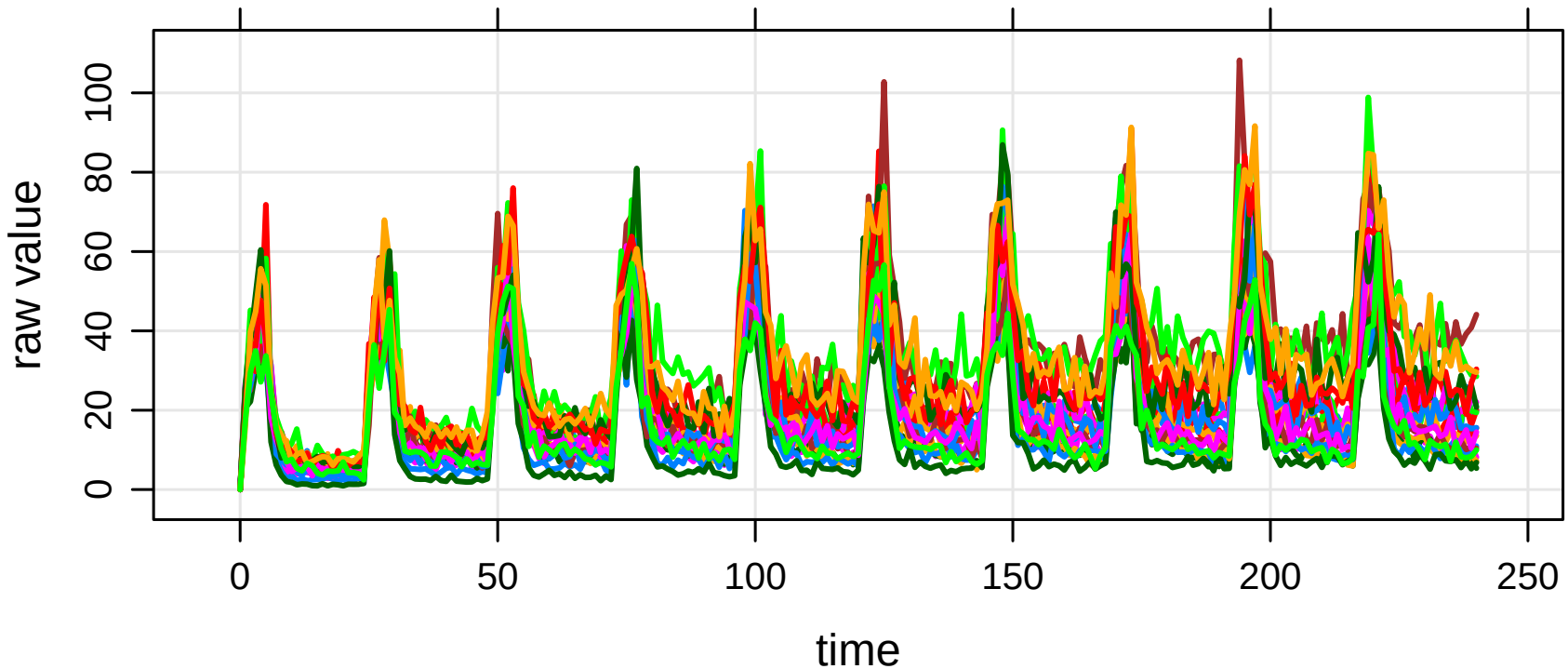


Update $SIGMA (use smat)

```
out <-
  mod %>%
  data_set(data) %>%
  smat(dmat(0.025,0.025)) %>%
  mrgsim(end=240,Req="DV")
```

Result

```
plot(out)
```



Population simulation from a posterior

```
post[1:3, 1:6] %>% as.data.frame
```

```
.   ITERATION   THETA1  THETA2  THETA3  THETA4   THETA5
. 1         1 0.706956 1.98185 4.21065 1.18204 0.194938
. 2         2 0.521959 1.99340 4.25382 1.18211 0.479422
. 3         3 0.699798 2.01382 4.26981 1.18384 0.400331
```

- Get `OMEGA` and `SIGMA` out of the posterior (as lists of matrices)
- Each iteration draws from a different row of the posterior, updating `THETA`, `SIGMA`, and `OMEGA` along the way

Get your OMEGA and SIGMA matrices

Looking for OMEGA1.1, OMEGA2.1, OMEGA2.2, SIGMA1.1, etc...

```
omegas <- as_bmat(post, "OMEGA")
sigmas <- as_bmat(post, "SIGMA")
```

These are both lists

```
omegas[[55]]
```

```
.              [,1]         [,2]         [,3]          [,4]
. [1,]  0.1295170 -0.02502250 -0.1680880  0.01591000
. [2,] -0.0250225  0.17792600 -0.1984400  0.00153978
. [3,] -0.1680880 -0.19844000  0.9779200 -0.01302450
. [4,]  0.0159100  0.00153978 -0.0130245  0.04564890
```

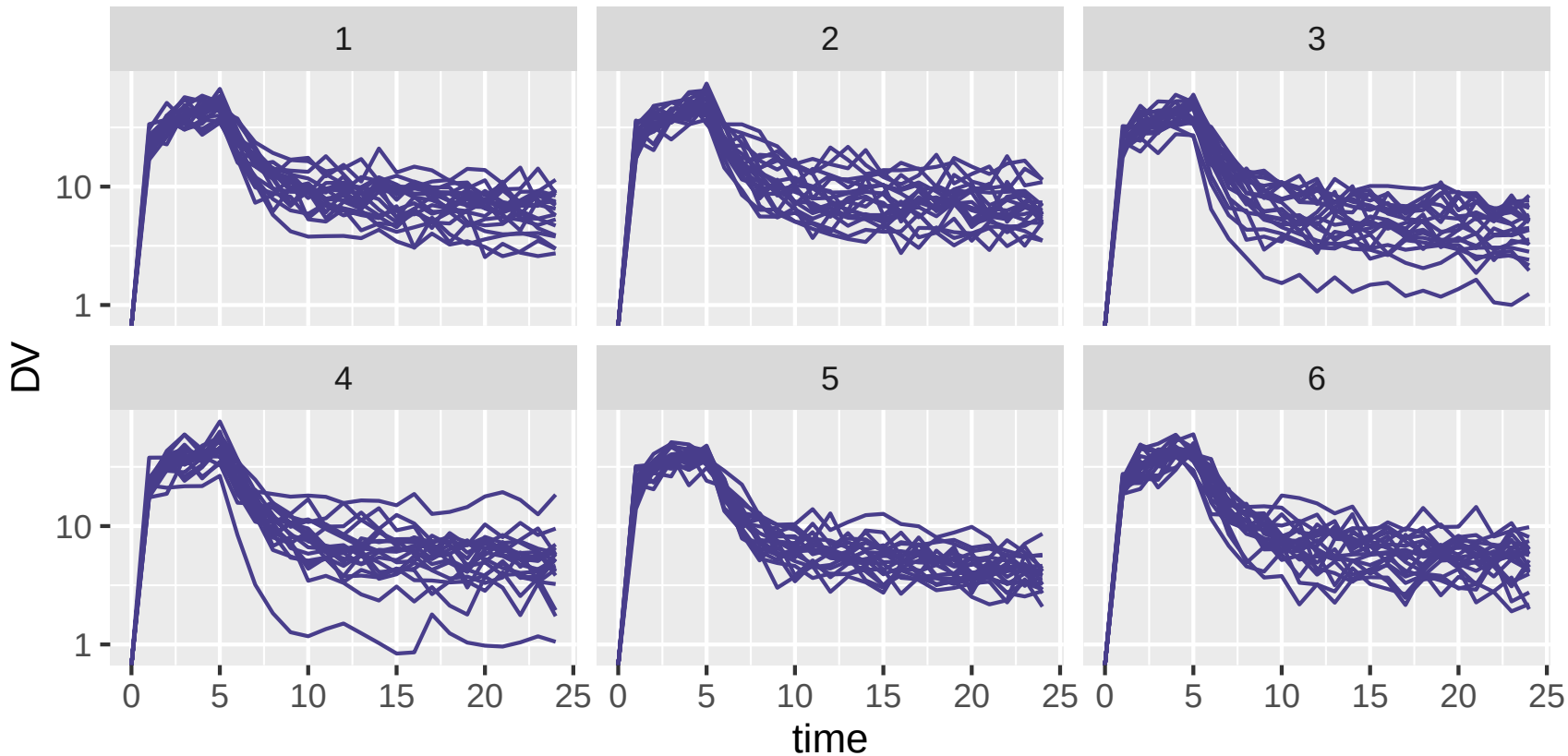
Let's go parallel

```
mcRNG()
set.seed(8712)
mc.reset.stream()

out <-
  mclapply(1:6, function(i) {
    mod %>%
      param(slice(post,i)) %>%    # <--- ***
      omat(omegas[[i]]) %>%       # <--- ***
      smat(sigmas[[i]]) %>%       # <--- ***
      data_set(data) %>%
      mrgsim(end=24,delta=1, Req="DV") %>%
      mutate(irep=i)
  }) %>% bind_rows
```

Result

```
ggplot(out, aes(time,DV, group=ID)) + facet_wrap(~irep) +
  geom_line(col="darkslateblue") + scale_y_log10()
```



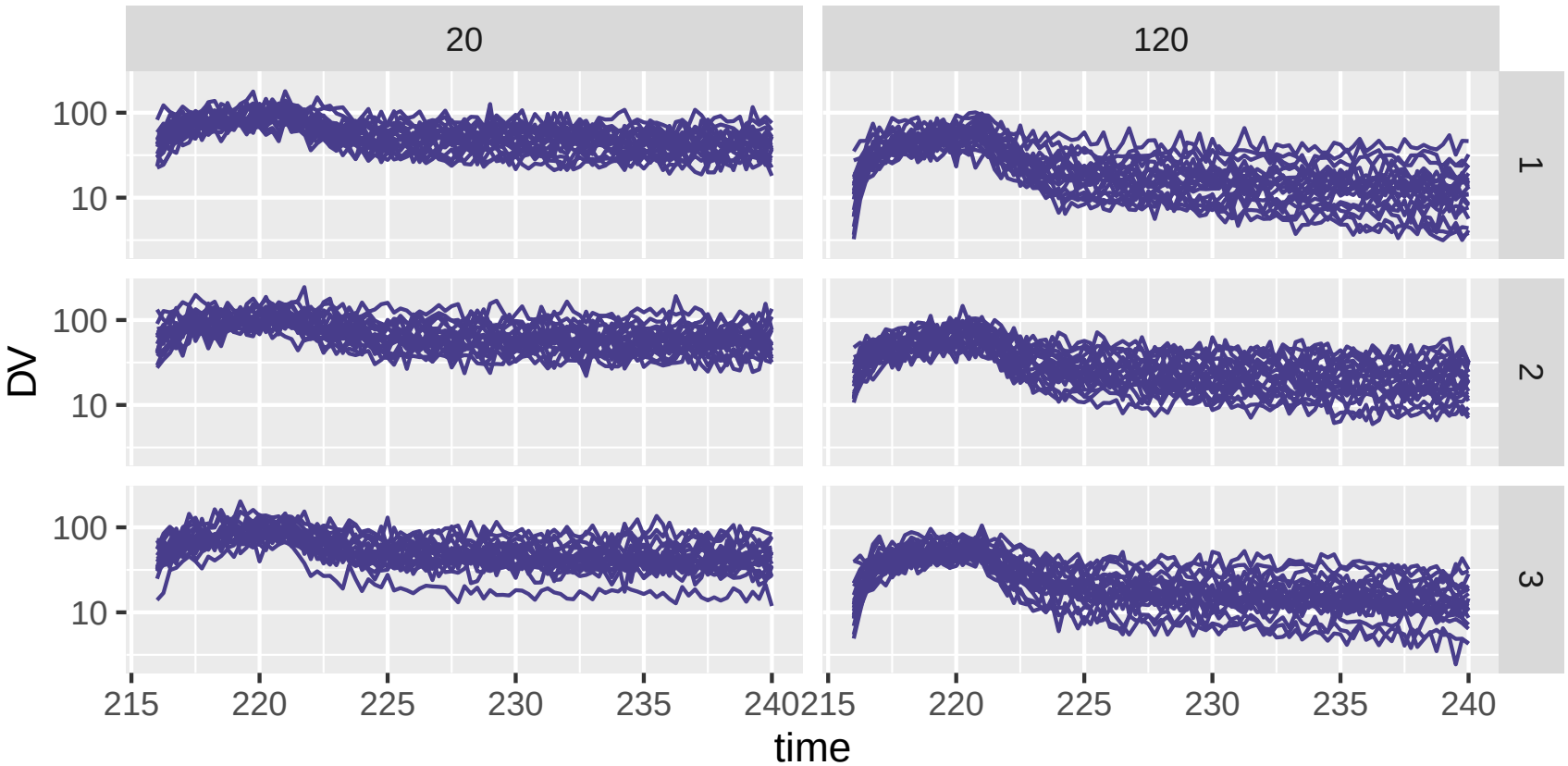
A population data set with covariates

```
data2 <- bind_rows(data %>% mutate(EGFR=20),
                   data %>% mutate(EGFR=120))
```

Only change the data set and replicate number

```
out <-
  mclapply(1:3, function(i) {
    mod %>%
      param(slice(post,i)) %>%
      omat(omegas[[i]]) %>%
      smat(sigmas[[i]]) %>%
      data_set(data2) %>%
      Req(DV) %>% carry.out(EGFR) %>%
      mrgsim(end=240,delta=0.25) %>%
      mutate(irep=i)
  }) %>% bind_rows()
```

Result



Why does this work?

Consistent naming in $PARAM & post & data

- The **posterior** had columns THETA1, THETA2, THETA3, ...
  - We also had THETA1, THETA2, THETA3, etc ... in the parameter list
  - mrgsolve will update THETAn as we draw from the posterior and update
- The **data set** had columns EGFR, WT, SEX etc ...
  - We also had EGFR, WT, SEX etc ... in the parameter list
  - mrgsolve will update those covariate values as it works through the data set

doParallel example

```
library(doParallel)
cl <- makeCluster(4); registerDoParallel(cl)
```

```
clusterCall(cl, function() {
  library(mrgsolve); library(dplyr)
})

mod. <- mrgsolve:::house()

out. <- foreach(i=1:100) %dopar% {
  mod. %>%
    ev(amt=1000,ii=8, addl=3) %>%
    mrgsim %>% mutate(irep=i)
} %>% bind_rows
```