`mrgsolve` II

Ashwin Karanam

July 27, 2020

Contents

1. Overview of the session

As our penultimate session, this will be an intermediate to advanced level session which will utilize everything we have learnt up until now including functions, tidyverse and pipes. Things can get confusing, so feel free to ask questions.

---

2. Recap of last session:

- `mrgsolve` is an `R` package for simulation from ODE-based models
  - Free, OpenSource, GitHub, CRAN
- Language
  - Models written in `C++` inside model specification format
  - Simulation specifics in an `R` Script
- Simulations will a variation of one of the following:
  - `model` %>% `intervention` %>% `options` %>% Go! %>% . . .
  - `model` %>% `intervention` %>% `population` %>% Go! %>% . . .
  - `model` %>% `data-set` %>% Go! %>% . . .
    - ∗ where `data-set` = `intervention + population`

---

3. Real life example:

- "Population Pharmacokinetic Analysis and Dosing Regimen Optimization of Meropenem in Adult Patients"
  - Li et al. J Clin Pharmacol 2006
- Meropenem is broad-spectrum carbapenem antibiotic
  - Efficacy related to time above MIC
- IV dosing every 8 hours by infusion or bolus
  - bolus over 3 to 5 minutes
  - infusion over 15 to 30 minutes
- Authors were interested in seeing if a longer infusion duration can increase time above MIC
- Adapted from Kyle's workshop slides

The model from this publication is available on DDMoRe. `Simulated_DatasetMeropenem.csv` is a simulated dataset from the study provided by the authors. Let's take a look at this dataset.

```
data <-
  read_csv("Simulated_DatasetMeropenem.csv", na = '.') %>%
  mutate(CMT=1, DUR = AMT/RATE)
```

```
## Parsed with column specification:
## cols(
##   ID = col_double(),
##   TIME = col_double(),
##   GROUP = col_double(),
##   DV = col_double(),
##   MDV = col_double(),
##   EVID = col_double(),
##   AMT = col_double(),
##   RATE = col_double(),
##   AGE = col_double(),
##   WT = col_double(),
##   CLCR = col_double()
## )
```

```
n_distinct(data$ID)
```

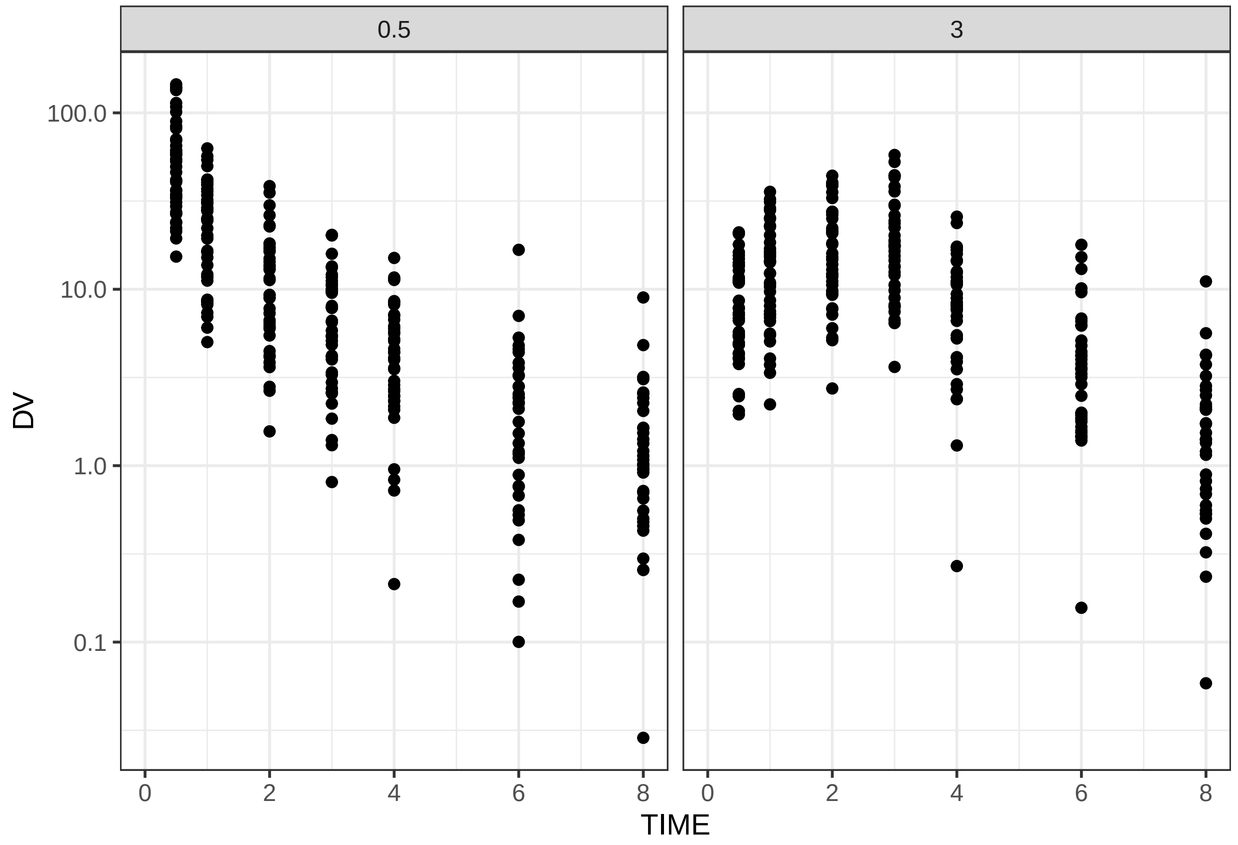
```
## [1] 79
```

```
head(data)
```

```
## # A tibble: 6 x 13
##      ID  TIME GROUP    DV   MDV  EVID   AMT  RATE   AGE    WT  CLCR   CMT
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1   0       1 NA        1     1   500  1000  29.7  63.8    83     1
## 2     1   0.5     1 31.1     NA     0    NA    NA  29.7  63.8    83     1
## 3     1   1       1 11.2     NA     0    NA    NA  29.7  63.8    83     1
## 4     1   2       1  6.21    NA     0    NA    NA  29.7  63.8    83     1
## 5     1   3       1  4.00    NA     0    NA    NA  29.7  63.8    83     1
## 6     1   4       1  2.33    NA     0    NA    NA  29.7  63.8    83     1
## # ... with 1 more variable: DUR <dbl>
```

Let's plot the data

```
#Derive DURATION for each individual
# x %<>% is the same as x <- x %>%
data %<>%
  group_by(ID) %>%
  mutate(DUR = first(DUR[!is.na(AMT)])) %>%
  ungroup %>%
  mutate(DUR = round(DUR,1))

# Plot
ggplot(data, aes(TIME,DV)) +
  geom_point() +
  scale_y_log10() +
  facet_wrap(~DUR) + xlim(0,8) +
  theme_bw()
```



Look at distinct values of `CMT`, `EVID`, `DUR` in data

```
kable(count(data,CMT,EVID,DUR))
```

CMT	EVID	DUR	n
1	0	0.5	280
1	0	3.0	273
1	1	0.5	40
1	1	3.0	39

Now, let's take a look at the mrgsolve model file

```r
mod <- mread("meropenem")
```

```
## Building meropenem ... done.
```

```r
#Model overview
mod
```

```
##
##
## --------------  source: meropenem.cpp  --------------
##
##   project: C:/Users/ashwi/D... - mrgsolve 2
##   shared object: meropenem-so-b24c375eef
##
##   time:          start: 0 end: 8 delta: 0.1
##                  add: <none>
##
##   compartments:  CENT PERIPH [2]
##   parameters:    WT CLCR AGE THETA1 THETA2 THETA3 THETA4
##                  THETA5 THETA6 THETA7 THETA8 THETA9 [12]
##   captures:      CC Y [2]
##   omega:         4x4
##   sigma:         1x1
##
##   solver:        atol: 1e-08 rtol: 1e-08 maxsteps: 20k
## ------------------------------------------------------
```

```r
#Parameters
param(mod)
```

```
##
##  Model parameters (N=12):
##  name   value . name   value
##  AGE    35    | THETA5 -0.447
##  CLCR   83    | THETA6 0.82
##  THETA1 15    | THETA7 0.188
##  THETA2 12.7  | THETA8 0.476
##  THETA3 15.2  | THETA9 0.62
##  THETA4 12.4  | WT     70
```

What does the cpp file look like?

```r
see(mod)
```

```
##
## Model file:  meropenem.cpp
## [PROB]
## Meropenem PopPK
##
```

```
## https://www.ncbi.nlm.nih.gov/pubmed/16988206
##
## [SET] delta=0.1, end=8, req=""
##
## [PKMODEL] cmt = "CENT, PERIPH"
##
## [PARAM] @annotated
## WT   : 70 : Weight (kg)
## CLCR : 83 : Creatinine clearance (ml/min)
## AGE  : 35 : Age (years)
##
## [THETA] @annotated
##  1.50E+01 : Typical value of clearance (L/h)
##  1.27E+01 : Typical value of volume 1 (L)
##  1.52E+01 : Intercompartmental clearance (L/h)
##  1.24E+01 : Typical value of volume 2 (L)
## -4.47E-01 : AGE on CL
##  8.20E-01 : WT on V1
##  1.88E-01 : Proportional error standard deviation
##  4.76E-01 : Additive error standard deviation
##  6.20E-01 : CLCR on CL
##
## [MAIN]
##
## double TVCL     = THETA1;
## double TVV1     = THETA2;
## double TVQ      = THETA3;
## double TVV2     = THETA4;
## double CL_AGE   = THETA5;
## double V1_WT    = THETA6;
## double RUV_PROP = THETA7;
## double RUV_ADD  = THETA8;
## double CL_CLCR  = THETA9;
##
## double LOGTWT = log((WT/70.0));
##
## double LOGTAGE = log((AGE/35.0));
##
## double LOGTCLCR = log((CLCR/83.0));
##
## double CL =  exp(log(TVCL) + CL_AGE * LOGTAGE + CL_CLCR * LOGTCLCR +  ETA(1)) ;
##
## double V1 =  exp(log(TVV1) + V1_WT * LOGTWT +  ETA(2)) ;
##
## double Q  =  exp(log(TVQ) +  ETA(3)) ;
##
## double V2 =  exp(log(TVV2) +  ETA(4));
##
## [OMEGA] @annotated
## ECL : 8.84E-02 : ETA on CL
## EV1 : 9.76E-02 : ETA on V1
## EQ  : 1.03E-01 : ETA on Q
## EV2 : 7.26E-02 : ETA on V2
##
```

```
## [SIGMA] @annotated
## EP : 1 : Not used
##
## [TABLE]
## capture CC = (CENT/V1);
## double IPRED = CC;
## double W = sqrt((RUV_ADD*RUV_ADD)+ (RUV_PROP*RUV_PROP*IPRED*IPRED));
## capture Y = IPRED+W*EPS(1);
```

Some additional helpful `C++` functions are:

```
if(a == 2) b = 2;
if(b <= 2) {
  c=3;
} else {
  c=4;
}
double d = pow(base,exponent);
double e = exp(3);
double f = fabs(-4);
double g = sqrt(5);
double h = log(6);
double i = log10(7);
double j = floor(4.2);
double k = ceil(4.2);
```

You can also refer to http://en.cppreference.com/w/cpp/numeric/math/tgamma for more details.

---

4. Creating visual predictive checks with `mrgsolve`

We will be using the meropenem simulated data and the model described earlier for creating visual predictive checks (VPCs). According to `A Tutorial on Visual Predictive Checks. Mats O. Karlsson & Nick Holford. Page 2008`:

> The visual predictive check (VPC) is a model diagnostic that can be used to: (i) allow comparison between alternative models, (ii) suggest model improvements, and (iii) support appropriateness of a model. The VPC is constructed from stochastic simulations from the model therefore all model components contribute and it can help in diagnosing both structural and stochastic contributions. In a typical VPC, the model is used to repeatedly (usually n >= 1000) simulate observations according to the original design of the study. Based on these simulations, percentiles of the simulated data are plotted versus an independent variable, usually time since start of treatment. It is then desirable that the same percentiles are calculated and plotted for the observed data to aid comparison of predictions with observations.

So, things we need for a VPC:

1. Observed dataset - Simulated meropenem data avaialable
2. Study design with doses - Also from the simulated dataset
3. Model from which iterative simulations can be performed - `meropenem.cpp` file
4. Iterative simulation code - R code
5. Plotting - also R code

4.1 Observed data

Let us subset the observed data which will be used for plotting at the end

```
obs <- filter(data, EVID==0)
head(obs)
```

```
## # A tibble: 6 x 13
##      ID  TIME GROUP     DV   MDV  EVID   AMT  RATE   AGE    WT  CLCR   CMT
##   <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1   0.5     1 31.1      NA     0    NA    NA  29.7  63.8    83     1
## 2     1   1       1 11.2      NA     0    NA    NA  29.7  63.8    83     1
## 3     1   2       1  6.21     NA     0    NA    NA  29.7  63.8    83     1
## 4     1   3       1  4.00     NA     0    NA    NA  29.7  63.8    83     1
## 5     1   4       1  2.33     NA     0    NA    NA  29.7  63.8    83     1
## 6     1   6       1  0.766    NA     0    NA    NA  29.7  63.8    83     1
## # ... with 1 more variable: DUR <dbl>
```

4.2 Dosing data

Let us subset dosing data

```
vpcdose <-
  data %>%
  group_by(ID) %>%
  mutate(RATE = first(RATE))


head(vpcdose)
```

```
## # A tibble: 6 x 13
## # Groups:   ID [1]
##      ID  TIME GROUP    DV   MDV  EVID   AMT  RATE   AGE    WT  CLCR   CMT
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1   0       1 NA        1     1   500  1000  29.7  63.8    83     1
## 2     1   0.5     1 31.1     NA     0    NA  1000  29.7  63.8    83     1
## 3     1   1       1 11.2     NA     0    NA  1000  29.7  63.8    83     1
## 4     1   2       1  6.21    NA     0    NA  1000  29.7  63.8    83     1
## 5     1   3       1  4.00    NA     0    NA  1000  29.7  63.8    83     1
## 6     1   4       1  2.33    NA     0    NA  1000  29.7  63.8    83     1
## # ... with 1 more variable: DUR <dbl>
```

Alternatively, you could only subset dosing datasets and provide your own timepoints

```
vpcdose.test <- filter(data, EVID==1)

des1 <- tgrid(0,3.1,0.1)
des2 <- tgrid(0,8,1)
des <- c(des1,des2)
des
```

```
## start:  0  end:    3.1  delta:  0.1  offset: 0  min:    0   max:    3.1
## --------
## start:  0  end:    8  delta:  1  offset: 0  min:    0   max:    8
## --------
```

4.3 Iterative simulation function

We can do this two ways i) create simulation function and use `map` or ii) use `for` loops. We will create a function for simulation

```
vpc <- function(i){
  mod %>%
    data_set(vpcdose) %>% # Provide dataset with dose and times
    carry_out(DUR) %>% # Carry the duration out for plotting
    obsonly %>% # Only request the concentrations
    mrgsim() %>% # simulate
    filter(TIME > 0 & Y > 0) %>% # remove any simulations below 0
    mutate(irep = i) # add a column called irep
}
```

This function will return a simulated data set containing

- time as in the dataset
- `DUR` the infusion duration
- `TIME` $>$ 0 and `Y` $>$ 0
- Labeled with replicate number `i`

4.4 Iterative simulations

Now, we need to simulate `n` replicates using this function

```
iter <- 100
out <- map(1:iter, vpc) %>% bind_rows()

head(out)
```

```
## # A tibble: 6 x 6
##      ID  TIME   DUR    CC     Y  irep
##   <dbl> <dbl> <dbl> <dbl> <dbl> <int>
## 1     1   0.5   0.5 28.5  35.0      1
## 2     1   1     0.5 11.1  14.5      1
## 3     1   2     0.5  6.02  5.94     1
## 4     1   3     0.5  3.91  3.33     1
## 5     1   4     0.5  2.54  2.40     1
## 6     1   6     0.5  1.08  1.53     1
```

```
tail(out)
```

```
## # A tibble: 6 x 6
##      ID  TIME   DUR     CC      Y  irep
##   <dbl> <dbl> <dbl>  <dbl>  <dbl> <int>
## 1    79     1     3 16.9   17.4     100
## 2    79     2     3 21.5   16.4     100
## 3    79     3     3 23.6   17.1     100
## 4    79     4     3  7.67   6.10    100
## 5    79     6     3  1.64   1.98    100
## 6    79     8     3  0.368  0.430   100
```

4.5 Summarize the data and plot

Now we have the simulate data. All we need to do is summarize them. Let's define some functions that will helpus make summarizing easier.

```
qt <- function(x,y) unname(quantile(x,prob=y/100))
lo <- function(x) qt(x,5)
hi <- function(x) qt(x,95)
med <- function(x) qt(x,50)
loci <- function(x) qt(x,2.5)
hici <- function(x) qt(x,97.5)
```

Now summarize by iteration

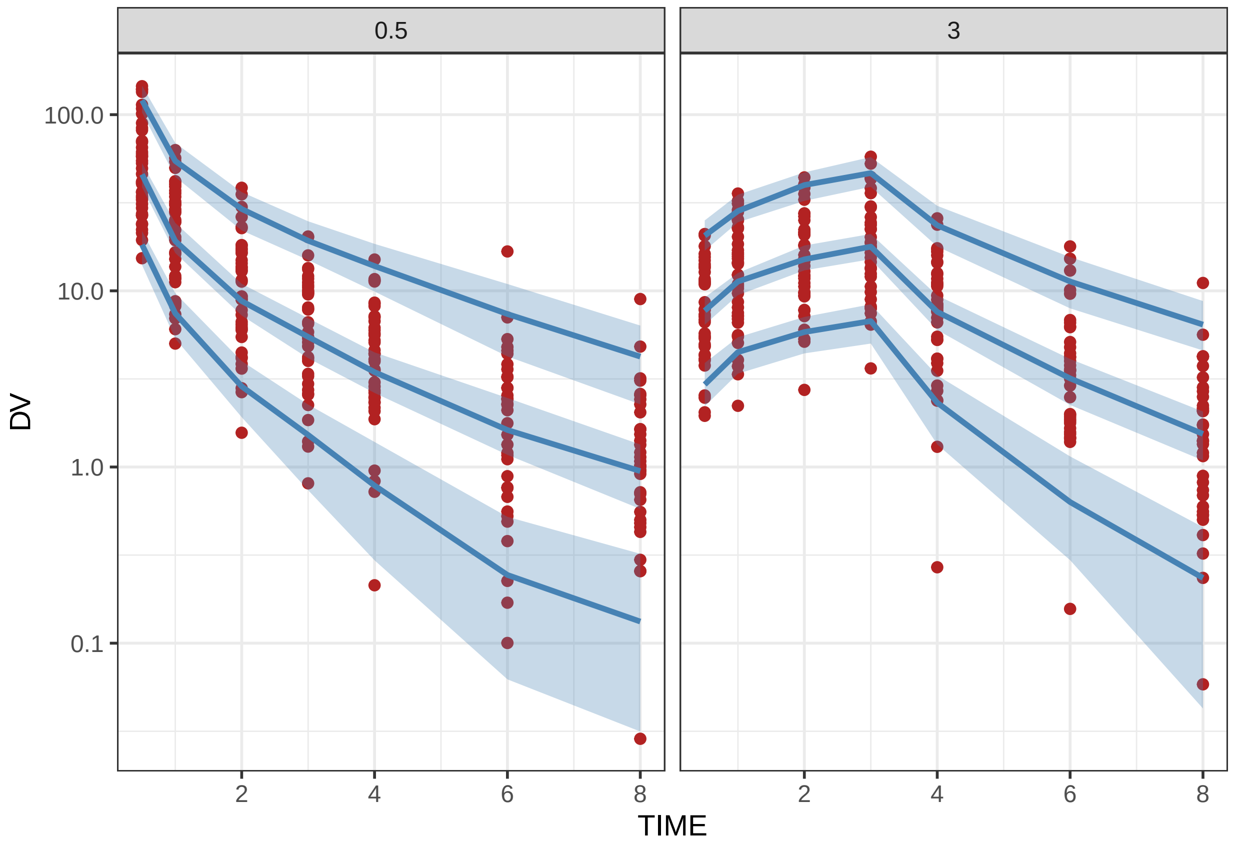
```
sum1 <-
  out %>%
  filter(Y > 0) %>%
  group_by(DUR,irep,TIME) %>%
  summarise(med=med(Y), lo=lo(Y), hi=hi(Y), N=n())
```

Next summarize this summary by ignoring iteration

```
sum2 <-
  sum1 %>%
  group_by(DUR,TIME) %>%
  summarise(medlo = loci(med), medmed = med(med), medhi = hici(med),
            lolo =  loci(lo),  lomed  = med(lo),  lohi  = hici(lo),
            hilo =  loci(hi),  himed  = med(hi),  hihi  = hici(hi))
```

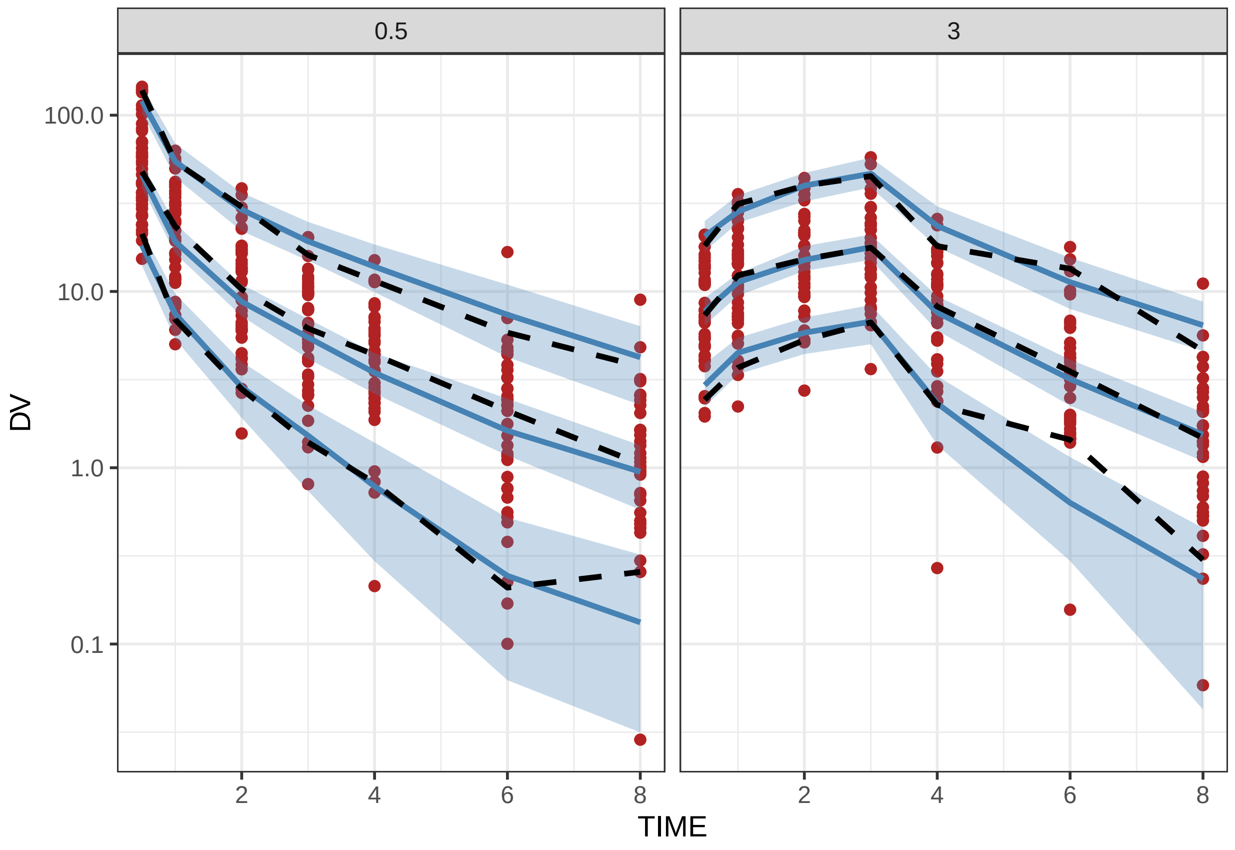
Now we can plot

```
col1 <- "steelblue"
col2 <- "firebrick"

p1 <-
  ggplot(data=sum2) +
  geom_point(data=obs, aes(TIME,DV),col=col2) + #add datapoints
  geom_line(aes(TIME,y=medmed), lwd=1,col=col1) + # median line
  geom_ribbon(aes(TIME,ymin=medlo, ymax = medhi),alpha=0.3,fill=col1) + # PI for median
  geom_line(aes(TIME,y=lomed), lwd=1,col=col1) + # low quantile line
  geom_ribbon(aes(TIME,ymin=lolo, ymax=lohi),alpha=0.3,fill=col1) + # PI for low quantile
  geom_line(aes(TIME,y=himed), lwd=1,col=col1) + # high quantile
  geom_ribbon(aes(TIME,ymin=hilo, ymax=hihi),alpha=0.3,fill=col1) + # PI for high quantile
  scale_y_continuous(trans="log", breaks=10^seq(-5,5)) +
  facet_wrap(~DUR)+
  theme_bw()

p1
```



We can also summarize the observed data and add them to the plot

```
obssum <-
  obs %>%
  filter(DV > 0) %>%
  group_by(DUR,TIME) %>%
  summarise(med=med(DV), lo=lo(DV), hi=hi(DV), N=n())

p1 +
  geom_line(data=obssum,aes(TIME,y=med),lty=2, lwd=1) +
  geom_line(data=obssum,aes(TIME,y=lo), lty=2, lwd=1) +
  geom_line(data=obssum,aes(TIME,y=hi), lty=2, lwd=1)
```


0.5
3
DV
TIME
100.0
10.0
1.0
0.1
2
4
6
8