Using a PK/PD model for simulation-based assessment of probability of technical success in drug development.

https://www.github.com/metrumresearchgroup

https://www.github.com/mrgsolve/examples

https://mrgsolve.github.io/user_guide

Contents

1 Probability of technical success

- $\Delta_i = f(\Theta, \Omega, \Sigma)$ a the measure of treatment effect of interest
  - Θ = fixed effects (population parameters)
  - Ω = covariance matrix of subject-level random effects
  - Σ = covariance matrix for within-subject random effects
- Δ_i could be
  - mean response
  - mean change from baseline
  - fraction of patients achieving some goal
  - median time to some event
  - relative measure comparing test and reference treatment
- $\mathrm{PTS} = \mathrm{P}(\Delta \geq \mathrm{TV})$
  - TV = a target value; the actual goal that you want to meet
  - Does not depend on the sample size, trial design, etc
  - Calculation based on information about Θ, Ω, Σ (e.g. from fitting model to data)

1.1 References

- Chuang-Stein, C., Kirby, S., French, J., Kowalski, K., Marshall, S., Smith, M.K., Bycott, P., Beltangady, M. *A Quantitative Approach for Making Go/No-Go Decisions in Drug Development.* Drug Information Journal, Vol 45. pp 187-202. 2011.
- Kowalski, K.G., French, J.L., Smith, M.K., Hutmacher, M.M. “A model-based framework for quantitative decision making in drug development”. ACOP, Tuscon, AZ. 2008. http://tucson2008.go-acop.org/pdfs/8-Kowalski_FINAL.pdf
- Smith, M.K., French, J., Kowalski, K., Ewy, W. *Enhanced Quantitative Decision Making - Reducing the likelihood of incorrect decisions.* PAGE Pre-Meeting Presentation, St. Petersburg, Russia. 2009.
- Smith MK, French JL, Kowalski KG, Hutmacher MM, and Ewy W. (2011) *Decision-Making in Drug Development: Application of a Model Based Framework for Assessing Trial Performance.* In Clinical Trial Simulations, Holly H. C. Kimko and Carl C. Peck (eds). Springer.

2 Setup

```
.libPaths("lib")
library(mrgsolve)
library(dplyr)
library(tidyr)
library(readr)
library(ggplot2)
library(parallel)
library(magrittr)
source("src/functions.R")
library(rmarkdown)
library(knitr)

RNGkind("L'Ecuyer-CMRG")
options(mc.cores=32)
mc.reset.stream()

theme_update(legend.position="top")
opts_chunk$set(comment='.',fig.align="center")
```

3 Read in (simulated) NHANES data set

We took an NHANES data set and simulated a new, larger data set based on:

S. J. Tannenbaum, N. H. Holford, H. Lee, C. C. Peck, and D. R. Mould. *Simulation of correlated continuous and categorical variables using a single multivariate distribution.* J Pharmacokinet.Pharmacodyn., 2006.

- Filter out people with more than mild renal dysfunction

```
ids <- readRDS(file="data/nhanes_large.RDS") %>% filter(RFST <= 2)
```

- Keep only people with `BMI [20,35]`

```
ids %<>% filter(BMI >=20 & BMI <= 35)
```

- Create a marker for people with `BMI >= 25`

```
ids %<>% mutate(BMIG = as.integer(BMI >= 25))
```

The data set

```
head(ids) %>% as.data.frame
```

```
.         WT      AGE      EGFR      BMI       HT     ALBU      ALT
. 1 85.96013 19.64425 167.60337 29.69030 170.1941 3.806922 12.18813
. 2 83.48701 19.53721 131.61773 28.04386 172.5715 4.736923 26.05577
. 3 62.20423 23.68561 113.84334 29.98625 144.0428 4.259215 14.75077
```

```
. 4 53.27747 35.97811  82.60383 23.30401 151.3161 4.243018 35.86039
. 5 70.70554 79.64432 105.93033 24.65716 169.2199 4.282462 13.75932
. 6 61.64187 55.86612 158.90641 27.05034 151.0003 4.352739 35.10589
.       TBILI SEX ETHNIC RFSTAGE RFST BLACK BMIG
. 1 0.4241158   1      5  Normal    1     0    1
. 2 0.8002647   1      1  Normal    1     0    1
. 3 0.4254764   1      3  Normal    1     0    1
. 4 0.5306805   1      3    Mild    2     0    0
. 5 0.8768458   0      5  Normal    1     0    0
. 6 0.4680404   1      5  Normal    1     0    1
```

```
ids %>% count(RFSTAGE,BMIG)
```

```
. Source: local data frame [4 x 3]
. Groups: RFSTAGE [?]
.
.   RFSTAGE  BMIG     n
.    (fctr) (int) (int)
. 1    Mild     0  8139
. 2    Mild     1 18196
. 3  Normal     0 16332
. 4  Normal     1 26228
```

3.1 Function to automate some of the data assembly

- Select only `BMI`, `EGFR`, `SEX`, `RFST`, and `BMIG`
- Randomly sample `n` patients and add columns for `BID` dosing `x20`
- Create a grid of `ID`, `amt` and join to the covariates
- Derive `dose` column for summarizing later

```
gen_data <- function(ids,n,amt) {

  BIG_N <- n*length(amt)

  ids %<>%
    dplyr::select(BMI,EGFR,SEX,RFST,BMIG) %>%
    sample_n(BIG_N) %>%
    mutate(ID = 1:n())

  doses <- expand.ev(ID=1:n,amt=amt,ii=12,addl=19)

  df <- left_join(doses,ids,by="ID") %>% mutate(dose=amt)

  return(as.data.frame(df))
}
```

3.1.1 One population from which to simulate

- All takers
- `set = 1`

```
data <- ids %>% gen_data(1000,500)  %>% mutate(set=1)
```

3.1.2 Another population

- Only patients with `BMI >= 25`
- `set = 2`

```
data2 <- ids %>% filter(BMI >= 25) %>% gen_data(1000,500) %>% mutate(set=2)
```

3.1.3 Summarise

```
data <- bind_rows(data,data2)
data %>% count(set,RFST,BMIG,dose)
```

```
. Source: local data frame [6 x 5]
. Groups: set, RFST, BMIG [?]
.
.     set  RFST  BMIG  dose     n
.   (dbl) (dbl) (int) (dbl) (int)
. 1     1     1     0   500   242
. 2     1     1     1   500   400
. 3     1     2     0   500   100
. 4     1     2     1   500   258
. 5     2     1     1   500   610
. 6     2     2     1   500   390
```

4 The mrgsolve model

```
mod <- mread("popmodel", "models") %>% update(delta=12, end=240)

mod
```

```
.
.
. -------- mrgsolve model object (unix) --------
.   Project: /data/project.mrg/mrgsolve-demo/models
.   source:        popmodel.cpp
.   shared object: 1448cb48550 (loaded)
.
.   compile date:  06/24 03:57
.   Time:          start: 0 end: 240 delta: 12
.   >              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
```

```
see(mod)
```

```
.
. Model file:  popmodel.cpp
.  $PARAM
.  WT = 70, SEX=0, EGFR=100, BMI = 20, ALT = 0.5
.  BLACK=0, FORM=1, FBIO=1
.
.  $THETA
.  0.57 1.6 4.34
.  1.24 -0.078 0.3656 0.4720 0.0216 0.480
.  -0.0638141 0.79283 4.61 3.82 2.22 0.72
.
.  $CMT GUT CENT  PERIPH
.
.  $MAIN
.
.  F_GUT = 1;
.  if(FORM==2) F_GUT = 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));
.
.  double E0    = exp(THETA12 + ETA(4));
.  double EC50  = exp(THETA14);
.  double EMAX  = exp(THETA13);
.  double m     = exp(THETA15);
.
.  $OMEGA 0 0 0 0
.  $SIGMA 0 0
.
.  $ODE
.  dxdt_GUT   = -KA*GUT;
.  dxdt_CENT =  KA*GUT - (CL+Q)*CP + Q*CT;
.  dxdt_PERIPH = Q*(CP - CT);
.
```

```
.  $GLOBAL
.  double BASE = 0, base=0;
.  #define CT (PERIPH/VP)
.  #define CP (CENT/VC)
.  #define driver CP
.
.  $TABLE
.  double DV = CP*exp(EPS(1));
.  double IPRED = CENT/VC;
.
.  double EFF = E0 - EMAX*pow(driver,m)/(pow(EC50,m)+pow(driver,m));
.
.  if(NEWIND <=1) {
.    BASE = EFF;
.    base = EFF + EPS(2);
.  }
.
.  double dEFF = EFF - BASE;
.  double deff = EFF - base + EPS(2);
.
.  $CAPTURE CL EFF dEFF deff
.
.
```

```r
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     | .        .
```

Mention

- $THETA
- $MAIN
- $TABLE

5 The NONMEM model

- Read in the posterior

- Take only post-burnin iterations

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

Sample 1000 draws from the posterior

```
set.seed(101)
post %<>% sample_n(1000)
om <- as_bmat(post, "OMEGA")
sg <- as_bmat(post, "SIGMA")
```

5.1 The 3 data items we need to run the simulation

- post posterior samples for THETAn
- om list of OMEGA matrices
- sg list of SIGMA matrices

```
post
```

```
. Source: local data frame [1,000 x 35]
.
.    ITERATION   THETA1  THETA2  THETA3  THETA4   THETA5   THETA6   THETA7
.        <int>    <dbl>   <dbl>   <dbl>   <dbl>    <dbl>    <dbl>    <dbl>
. 1        986 0.623748 2.05548 4.30014 1.18659 0.469732 0.912680 0.601255
. 2        916 0.597378 2.05624 4.26490 1.17565 0.496322 0.756379 0.531457
. 3        273 0.775222 2.13074 4.29355 1.24915 0.400302 0.913668 0.527899
. 4        572 0.587915 2.02625 4.29448 1.22368 0.318081 0.905348 0.688503
. 5        594 0.719651 2.10929 4.33620 1.22720 0.380924 0.730739 0.387186
. 6        578 0.716189 2.05872 4.31671 1.21803 0.190726 0.594924 0.604107
. 7        328 0.661865 2.07204 4.23857 1.16023 0.475771 0.754144 0.456267
. 8        385 0.720112 2.00355 4.21791 1.24423 0.378419 1.014390 0.449391
. 9        447 0.598599 2.08730 4.31314 1.16673 0.458248 0.723430 0.486618
. 10        80 0.627769 1.94423 4.22879 1.18854 0.522763 0.636807 0.446237
. ..       ...      ...     ...     ...     ...      ...      ...      ...
. Variables not shown: THETA8 <dbl>, THETA9 <dbl>, THETA10 <dbl>, THETA11
.   <dbl>, THETA12 <dbl>, THETA13 <dbl>, THETA14 <dbl>, THETA15 <dbl>,
.   THETA16 <dbl>, THETA17 <dbl>, THETA18 <dbl>, THETA19 <dbl>, THETA20
.   <dbl>, SIGMA(1,1) <dbl>, SIGMA(2,1) <dbl>, SIGMA(2,2) <dbl>, OMEGA(1,1)
.   <dbl>, OMEGA(2,1) <dbl>, OMEGA(2,2) <dbl>, OMEGA(3,1) <dbl>, OMEGA(3,2)
.   <dbl>, OMEGA(3,3) <dbl>, OMEGA(4,1) <dbl>, OMEGA(4,2) <dbl>, OMEGA(4,3)
.   <dbl>, OMEGA(4,4) <dbl>, MCMCOBJ <dbl>.
```

```
om[[10]]
```

```
.             [,1]        [,2]       [,3]       [,4]
. [1,]  0.12238000  0.00599381 -0.1684630  0.0109533
. [2,]  0.00599381  0.24511000 -0.2919560 -0.0145303
. [3,] -0.16846300 -0.29195600  1.0169600  0.0297438
. [4,]  0.01095330 -0.01453030  0.0297438  0.0469669
```

```
sg[[100]]
```

```
.           [,1]    [,2]
. [1,] 0.0308353  0.0000
. [2,] 0.0000000 25.4227
```

6 A function to simulate responses

- `i` current simulation replicate
- `post` data frame holding posterior
- `indata` a template data set (`data.frame`)
- For each replicate (`i`), take a new draw from the posterior distribution for fixed effect estimates
- Before returning, only take the day 10 value and label

```
sim <- function(i,post,indata,pop=FALSE) {

 if(pop) mod <- mod %>% omat(om[[i]]) %>% smat(sg[[i]])

  mod %>%
    data_set(indata) %>%
    param(slice(post,i)) %>%
    Req(deff) %>%
    carry.out(RFST,BMIG,dose,set) %>%
    mrgsim(end=-1,add=240) %>%
    filter(time==240) %>%
    mutate(irep=i)
}
```

Function for `qapply` Test the function

```
set.seed(2201)
system.time(test <- sim(11,post,data,TRUE))
```

```
.    user  system elapsed
.   1.860   0.000   1.757
```

7 Run the simulation

7.1 Parallel with `mclapply`

The sequence: - Draw one set of Θ, Ω, and Σ from posterior / bootstrap estimates - This is `i` or `irep` or `iter` - Simulate 1000 patients - Filter to day-10 effect (change from baseline) - Repeat for 320 iterations

```
set.seed(11002)
system.time(out <- mclapply(1:320, sim, post=post, indata=data, pop=TRUE) %>% bind_rows)
```

```
.    user  system elapsed
. 921.522  16.322  36.996
```

```
out
```

```
. Source: local data frame [640,000 x 8]
.
.        ID  time  RFST  BMIG  dose   set      deff  irep
.     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl> <int>
. 1       1   240     1     1   500     1 -25.53285     1
. 2       2   240     1     1   500     1 -14.75919     1
. 3       3   240     2     0   500     1 -50.42116     1
. 4       4   240     2     1   500     1 -32.69586     1
. 5       5   240     1     0   500     1 -45.79958     1
. 6       6   240     2     1   500     1 -38.71611     1
. 7       7   240     2     1   500     1 -20.05167     1
. 8       8   240     2     1   500     1 -44.37036     1
. 9       9   240     1     0   500     1 -40.77183     1
. 10     10   240     1     0   500     1 -34.08069     1
. ..    ...   ...   ...   ...   ...   ...       ...   ...
```

7.2 Parallel with `qapply`

- Requires grid engine

```
if(FALSE) {

  stopifnot(require(qapply))

  mod <- mread("popmodel", "models", soloc="so") %>% update(delta=12, end=240)

  out <- qapply(1:32,
                parSeed=c(1,3,2,4,2,1),
                tag="q1",
                FUN=function(i,...) {loadso(mod); sim(i,...)},
                commonData=list(mod=mod, om=om,sg=sg,sim=sim),
                fargs=list(post=post,indata=data,pop=TRUE)) %>% bind_rows
}
```

7.3 Parallel with `doParallel`

- This should work on Windows

```
if(FALSE) {

  stopifnot(require(doParallel))

  cl <- makeCluster(32); registerDoParallel(cl)

  clusterCall(cl, function() {
    .libPaths("lib"); library(mrgsolve); library(dplyr)
  })
```

```
  clusterExport(cl,c("sim", "mod", "om", "sg", "data", "post"))

  system.time({
    out. <- foreach(i=1:320) %dopar% {
      loadso(mod)
      sim(i,post=post,indata=data,pop=TRUE)
    } %>% bind_rows
  })
  stopCluster(cl)

}
```

8 Summarize simulations to get PTS

8.1 Summary: fraction of patients reaching a target value

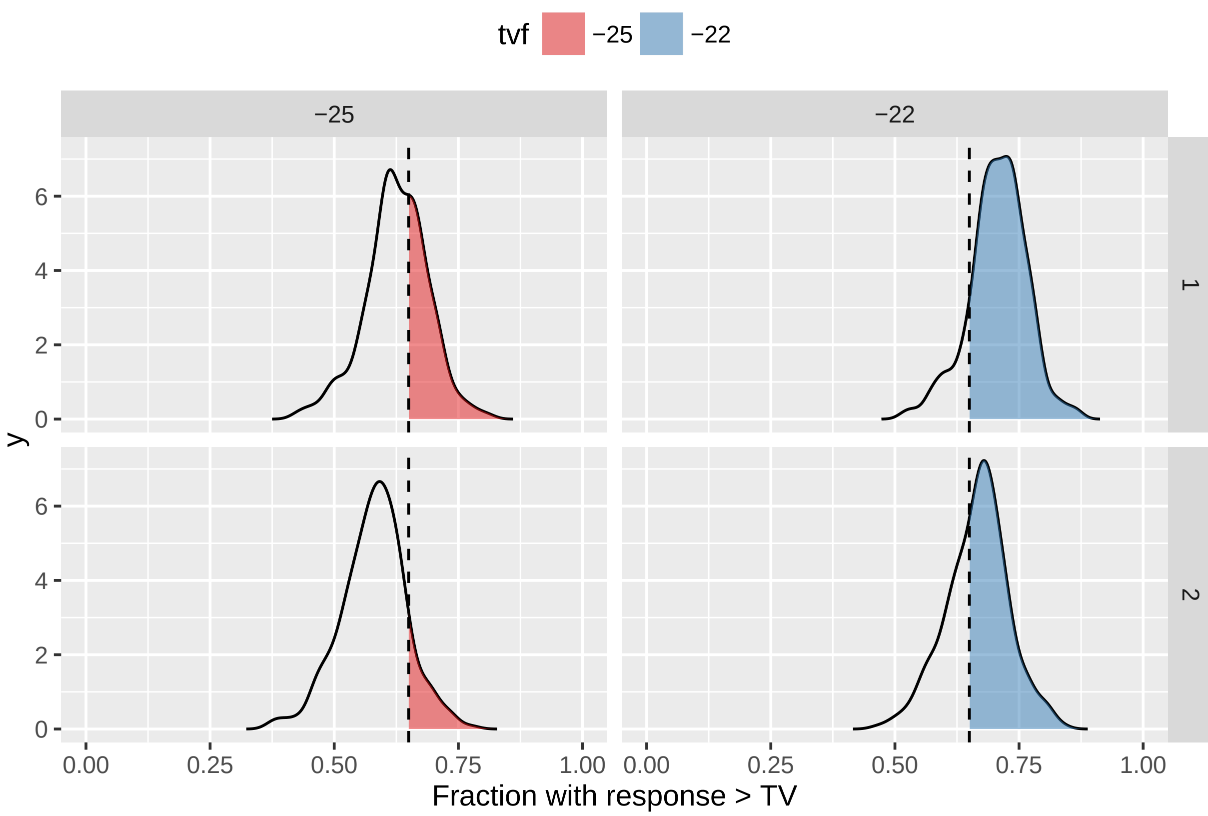
```
sum <- lapply(c(-25,-22), function(tv) {
  out %>%
    group_by(irep,dose,set) %>%
    summarise(frac=mean(deff < tv)) %>%
    mutate(tv=tv)
}) %>% bind_rows %>% mutate(tvf=factor(tv))
```

```
d <- sum %>% group_by(dose,tvf,tv,set) %>% do(.density(.$frac))

target.frac <- 0.65
```

The shaded area is PTS

```
ggplot(data=d, aes(x=x,y=y)) +
  geom_line() + facet_grid(set~tvf) + xlab("Fraction with response > TV") +
  geom_ribbon_density(d, "x >= 0.65",fill="tvf") + .fillSet1() +
  geom_vline(xintercept=target.frac,lty=2) + xlim(0,1)
```



Calculate the tail area for each cut

```
sum %>%
  group_by(set,tvf,dose) %>%
  summarise(PTS = mean(frac > target.frac))
```

```
. Source: local data frame [4 x 4]
. Groups: set, tvf [?]
.
.     set   tvf  dose      PTS
.   (dbl) (fctr) (dbl)    (dbl)
. 1     1   -25   500 0.368750
. 2     1   -22   500 0.859375
. 3     2   -25   500 0.106250
. 4     2   -22   500 0.640625
```

```
sum %>%
  group_by(tvf,tv,dose,set) %>%
  summarise(PTS = mean(frac > target.frac))
```

```
. Source: local data frame [4 x 5]
. Groups: tvf, tv, dose [?]
.
.      tvf    tv  dose   set      PTS
```

```
.    (fctr) (dbl) (dbl) (dbl)     (dbl)
. 1     -25   -25   500     1 0.368750
. 2     -25   -25   500     2 0.106250
. 3     -22   -22   500     1 0.859375
. 4     -22   -22   500     2 0.640625
```

8.2 Summary: mean response > target value

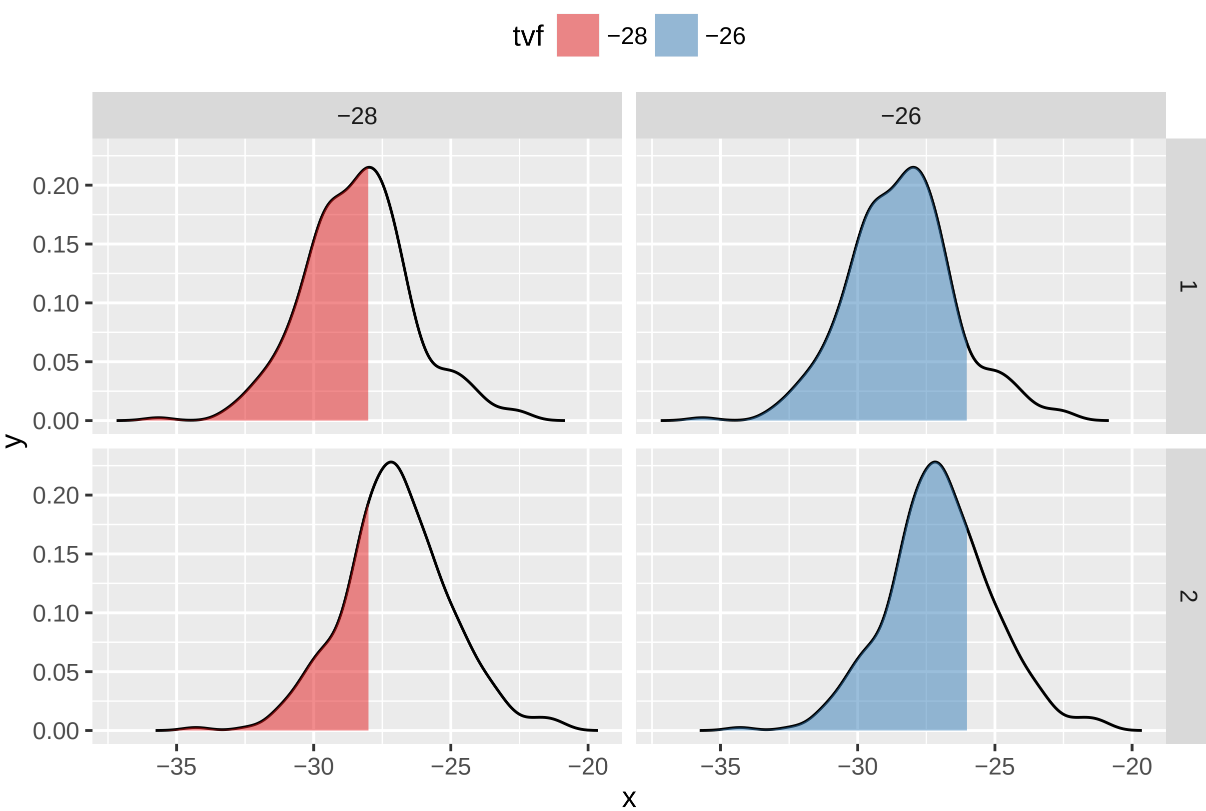
```
sum <-
  out %>%
  group_by(irep,dose,set) %>%
  summarise(mean = mean(deff))

sum <- lapply(c(-28,-26), function(tv) sum %>% mutate(tv = tv)) %>%
  bind_rows %>% mutate(tvf=factor(tv))
```

```
d <- sum %>% group_by(dose,tvf,tv,set) %>% do(.density(.$mean))
```

The shaded area is PTS

```
ggplot(data=d, aes(x,y)) +
  geom_line() +
  facet_grid(set~tv) + .fillSet1() +
  geom_ribbon_density(d,"x <= tv",fill="tvf")
```



```
sum %>%
  group_by(dose,tv,set) %>%
  summarise(PTS = mean(mean < tv))
```

```
. Source: local data frame [4 x 4]
. Groups: dose, tv [?]
.
.    dose    tv   set      PTS
.   (dbl) (dbl) (dbl)    (dbl)
. 1   500   -28     1 0.578125
. 2   500   -28     2 0.275000
. 3   500   -26     1 0.906250
. 4   500   -26     2 0.709375
```

9 Simulate model parameters from covariance matrix (of the estimate)

```
nsimpar <- 2500
nwish <- 300
mc.cores <- 32
options(mc.cores=mc.cores)
```

Take iteration `-1E9` to get the "estimate"

```
est <- read.csv("nonmem/1001/1001.ext", header=TRUE, skip=1, sep="") %>%
  filter(ITERATION == -1E9)
```

Go into the `run.cov` file to get the covariance matrix

```
.cov <- read.csv("nonmem/1001/1001.cov", header=TRUE, skip=1, sep="")
.cov$NAME <- NULL
```

We only want the covariance matrix for `THETA`s; we'll handle `OMEGA` and `SIGMA` separately

```
take <- grep("THETA",names(.cov))
cov <- .cov[take,take]
signif(cov,3)
```

```
.       THETA1    THETA2    THETA3    THETA4    THETA5    THETA6    THETA7
. 1   4.14e-03  0.000858 -3.94e-04  9.73e-07 -0.001380 -0.003920  5.22e-04
. 2   8.58e-04  0.004230 -2.05e-04 -2.67e-04  0.005390 -0.001070 -3.22e-04
. 3  -3.94e-04 -0.000205  3.78e-03  3.92e-04 -0.003990  0.000986 -2.95e-05
. 4   9.73e-07 -0.000267  3.92e-04  4.91e-04 -0.001690 -0.000213 -8.95e-05
. 5  -1.38e-03  0.005390 -3.99e-03 -1.69e-03  0.029300  0.002490 -3.87e-04
. 6  -3.92e-03 -0.001070  9.86e-04 -2.13e-04  0.002490  0.031200 -9.79e-04
. 7   5.22e-04 -0.000322 -2.95e-05 -8.95e-05 -0.000387 -0.000979  9.98e-03
. 8  -3.14e-03 -0.000997  3.46e-05  4.95e-05 -0.001150  0.000166  9.38e-04
. 9  -2.22e-03 -0.004450 -1.69e-03 -4.66e-04  0.004330  0.010100  8.91e-04
. 10 -6.58e-04 -0.001710  2.30e-04  1.04e-04 -0.000263  0.000831  1.44e-04
```

```
. 11  6.70e-04 -0.000235 -5.00e-03 -3.24e-04  0.002910 -0.007580 -1.02e-04
. 12  6.65e-05 -0.000206 -3.42e-04  2.09e-05  0.000245  0.000132 -1.80e-04
. 13 -1.32e-04  0.000155  1.92e-04  2.43e-04  0.000377  0.000466 -3.71e-04
. 14  1.66e-04  0.000399 -1.19e-04  2.43e-04  0.001460  0.000425  5.07e-05
. 15  5.00e-04  0.000623 -1.42e-04 -4.23e-04  0.000260 -0.002010  4.66e-04
. 16  0.00e+00  0.000000  0.00e+00  0.00e+00  0.000000  0.000000  0.00e+00
. 17  0.00e+00  0.000000  0.00e+00  0.00e+00  0.000000  0.000000  0.00e+00
. 18  0.00e+00  0.000000  0.00e+00  0.00e+00  0.000000  0.000000  0.00e+00
. 19  0.00e+00  0.000000  0.00e+00  0.00e+00  0.000000  0.000000  0.00e+00
. 20  0.00e+00  0.000000  0.00e+00  0.00e+00  0.000000  0.000000  0.00e+00
.       THETA8    THETA9   THETA10   THETA11   THETA12   THETA13   THETA14
. 1  -3.14e-03 -0.002220 -0.000658  0.000670  6.65e-05 -0.000132  1.66e-04
. 2  -9.97e-04 -0.004450 -0.001710 -0.000235 -2.06e-04  0.000155  3.99e-04
. 3   3.46e-05 -0.001690  0.000230 -0.005000 -3.42e-04  0.000192 -1.19e-04
. 4   4.95e-05 -0.000466  0.000104 -0.000324  2.09e-05  0.000243  2.43e-04
. 5  -1.15e-03  0.004330 -0.000263  0.002910  2.45e-04  0.000377  1.46e-03
. 6   1.66e-04  0.010100  0.000831 -0.007580  1.32e-04  0.000466  4.25e-04
. 7   9.38e-04  0.000891  0.000144 -0.000102 -1.80e-04 -0.000371  5.07e-05
. 8   5.44e-03  0.001430  0.000899  0.000946  1.80e-04 -0.000108 -5.76e-04
. 9   1.43e-03  0.033100  0.001290  0.011200  3.02e-04  0.002150  2.66e-03
. 10  8.99e-04  0.001290  0.002470  0.000344  2.24e-04 -0.000204 -5.77e-04
. 11  9.46e-04  0.011200  0.000344  0.046300  2.50e-04  0.000416  3.22e-04
. 12  1.80e-04  0.000302  0.000224  0.000250  1.03e-03  0.000091 -4.07e-04
. 13 -1.08e-04  0.002150 -0.000204  0.000416  9.10e-05  0.008300  1.03e-02
. 14 -5.76e-04  0.002660 -0.000577  0.000322 -4.07e-04  0.010300  2.06e-02
. 15 -1.63e-04 -0.005200 -0.000396 -0.001760 -5.72e-04 -0.008890 -1.17e-02
. 16  0.00e+00  0.000000  0.000000  0.000000  0.00e+00  0.000000  0.00e+00
. 17  0.00e+00  0.000000  0.000000  0.000000  0.00e+00  0.000000  0.00e+00
. 18  0.00e+00  0.000000  0.000000  0.000000  0.00e+00  0.000000  0.00e+00
. 19  0.00e+00  0.000000  0.000000  0.000000  0.00e+00  0.000000  0.00e+00
. 20  0.00e+00  0.000000  0.000000  0.000000  0.00e+00  0.000000  0.00e+00
.      THETA15 THETA16 THETA17 THETA18 THETA19 THETA20
. 1   0.000500       0       0       0       0       0
. 2   0.000623       0       0       0       0       0
. 3  -0.000142       0       0       0       0       0
. 4  -0.000423       0       0       0       0       0
. 5   0.000260       0       0       0       0       0
. 6  -0.002010       0       0       0       0       0
. 7   0.000466       0       0       0       0       0
. 8  -0.000163       0       0       0       0       0
. 9  -0.005200       0       0       0       0       0
. 10 -0.000396       0       0       0       0       0
. 11 -0.001760       0       0       0       0       0
. 12 -0.000572       0       0       0       0       0
. 13 -0.008890       0       0       0       0       0
. 14 -0.011700       0       0       0       0       0
. 15  0.021600       0       0       0       0       0
. 16  0.000000       0       0       0       0       0
. 17  0.000000       0       0       0       0       0
. 18  0.000000       0       0       0       0       0
. 19  0.000000       0       0       0       0       0
. 20  0.000000       0       0       0       0       0
```

```
THETA
```

```
theta <- est[grepl("THETA",names(est))]
```

OMEGA

```
omega <- as_bmat(est,"OMEGA")[[1]]
```

SIGMA

```
sigma <- as_bmat(est,"SIGMA")[[1]]
```

9.1 Use simpar to simulate THETAs, OMEGAs, and SIGMAs

- ?simpar
- Distributional assumptions
  - $\Theta \sim$ multivariate normal
  - $\Omega \sim$ Inverse Wishart
  - $\Sigma \sim$ Inverse Wishart
- Arguments:
  - omega is the estimated OMEGA matrix
  - sigma is the estimated SIGMA matrix
  - odf: OMEGA degrees of freedom; odf must be greater than length(omega)
  - sdf: SIGMA degrees of freedom; sdf must be greater than length(sigma)
  - simpar returns a matrix; we'll coerce to data.frame
  - nsim number of sets of simulated values
- Return:
  - Matrix of THETAs, OMEGAs, and SIGMAs
  - One simulated set per row in the matrix

```
simpost <- metrumrg::simpar(n=1500,
                            theta=unlist(theta),
                            cov=cov,
                            omega=omega,
                            sigma=sigma,
                            odf=100,sdf=1000) %>% data.frame
```

In the output, each row is one draw from the variance-covariance matrix.

```
head(simpost)
```

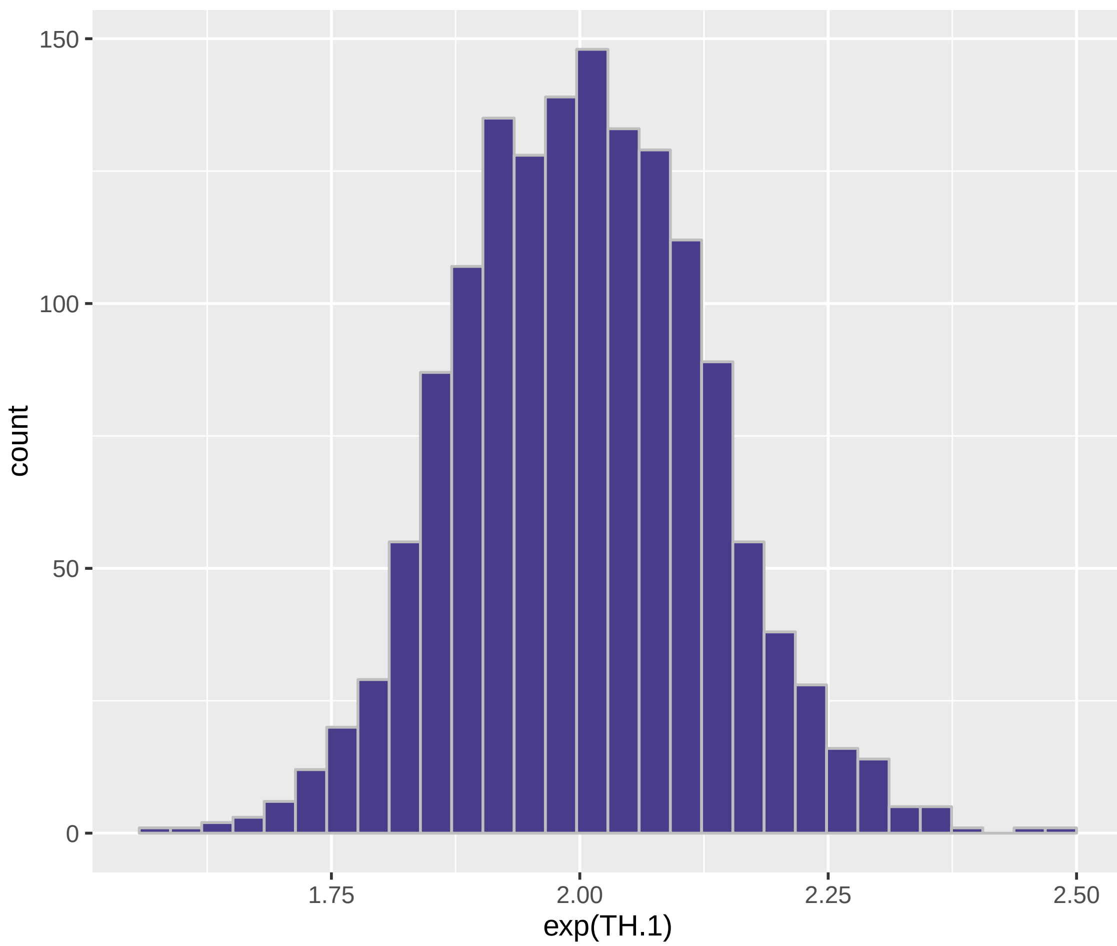
```
.     TH.1  TH.2  TH.3  TH.4     TH.5   TH.6   TH.7     TH.8   TH.9   TH.10
. 1 0.6065 2.181 4.316 1.183  0.53820 0.3765 0.5537 -0.14160 0.6860 -0.3887
. 2 0.7820 1.970 4.272 1.236 -0.04296 0.7952 0.5149 -0.18320 0.7087 -0.3778
. 3 0.7976 2.068 4.289 1.177  0.39900 0.8402 0.6314 -0.28230 1.1000 -0.3867
. 4 0.7379 2.074 4.406 1.178  0.34860 0.6758 0.4369 -0.26620 1.1320 -0.4383
. 5 0.6545 2.094 4.277 1.230  0.28930 0.4349 0.6289 -0.08600 0.6905 -0.4374
. 6 0.6969 2.020 4.224 1.194  0.40390 0.6952 0.4130 -0.08365 0.9445 -0.3105
.    TH.11 TH.12 TH.13 TH.14  TH.15 TH.16 TH.17 TH.18 TH.19 TH.20  OM1.1
```

```
. 1 1.0840 4.561 3.729 2.191 0.6753     0     0     0     0     0 0.1208
. 2 0.7627 4.610 3.760 2.224 0.6523     0     0     0     0     0 0.1307
. 3 0.9747 4.623 3.871 2.275 0.4127     0     0     0     0     0 0.1046
. 4 0.8225 4.556 3.824 1.961 0.6915     0     0     0     0     0 0.1051
. 5 1.0010 4.623 3.708 1.889 0.8608     0     0     0     0     0 0.1171
. 6 0.9761 4.618 3.851 2.133 0.5323     0     0     0     0     0 0.1460
.       OM2.1  OM2.2    OM3.1   OM3.2  OM3.3     OM4.1     OM4.2     OM4.3
. 1 -0.007655 0.1927 -0.13450 -0.1904 0.8574  0.012090 -0.014840  0.030380
. 2  0.003527 0.1581 -0.13920 -0.1088 0.6588  0.011920 -0.014720  0.005142
. 3 -0.023020 0.1634 -0.08857 -0.1482 0.7390  0.007778 -0.032760  0.099240
. 4 -0.018180 0.2608 -0.14270 -0.2231 0.9691  0.011480 -0.005546 -0.014900
. 5 -0.018880 0.1721 -0.11980 -0.1678 0.7584 -0.000136 -0.026480  0.060660
. 6 -0.009546 0.1728 -0.14280 -0.1542 0.6511  0.014740 -0.005268 -0.000919
.     OM4.4   SG1.1    SG2.1 SG2.2
. 1 0.06039 0.03092  0.03467 29.62
. 2 0.05152 0.03369  0.01283 31.60
. 3 0.07829 0.03258 -0.03316 29.10
. 4 0.05857 0.03278 -0.01699 31.27
. 5 0.06337 0.03056  0.03105 29.01
. 6 0.04759 0.03091 -0.00738 28.39
```

Simulated **TVCL** distribution:

```r
ggplot(data=simpost) + geom_histogram(aes(x=exp(TH.1)), fill=.dsb, col="grey")
```

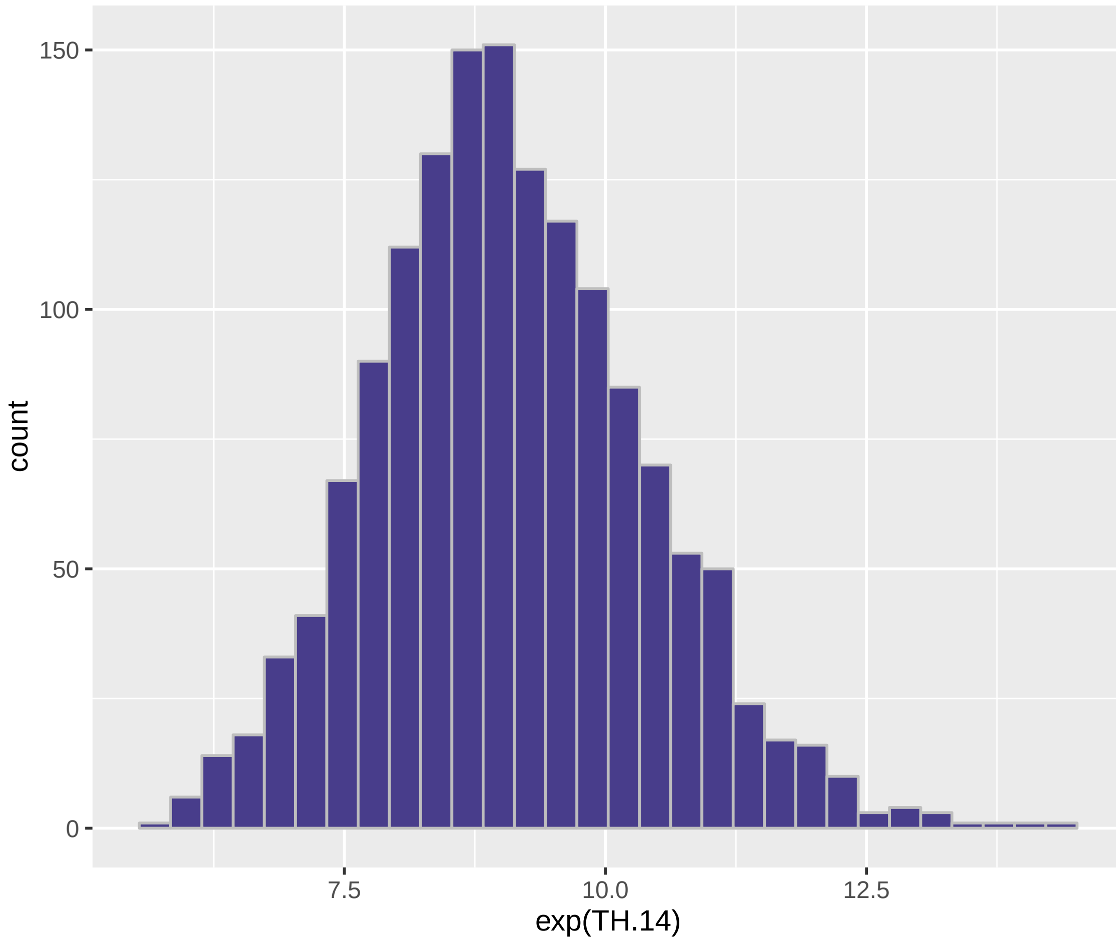
```
. `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Simulated **EC50** distribution:

```
ggplot(data=simpost) + geom_histogram(aes(x=exp(TH.14)), fill=.dsb, col="grey")
```

```
. `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



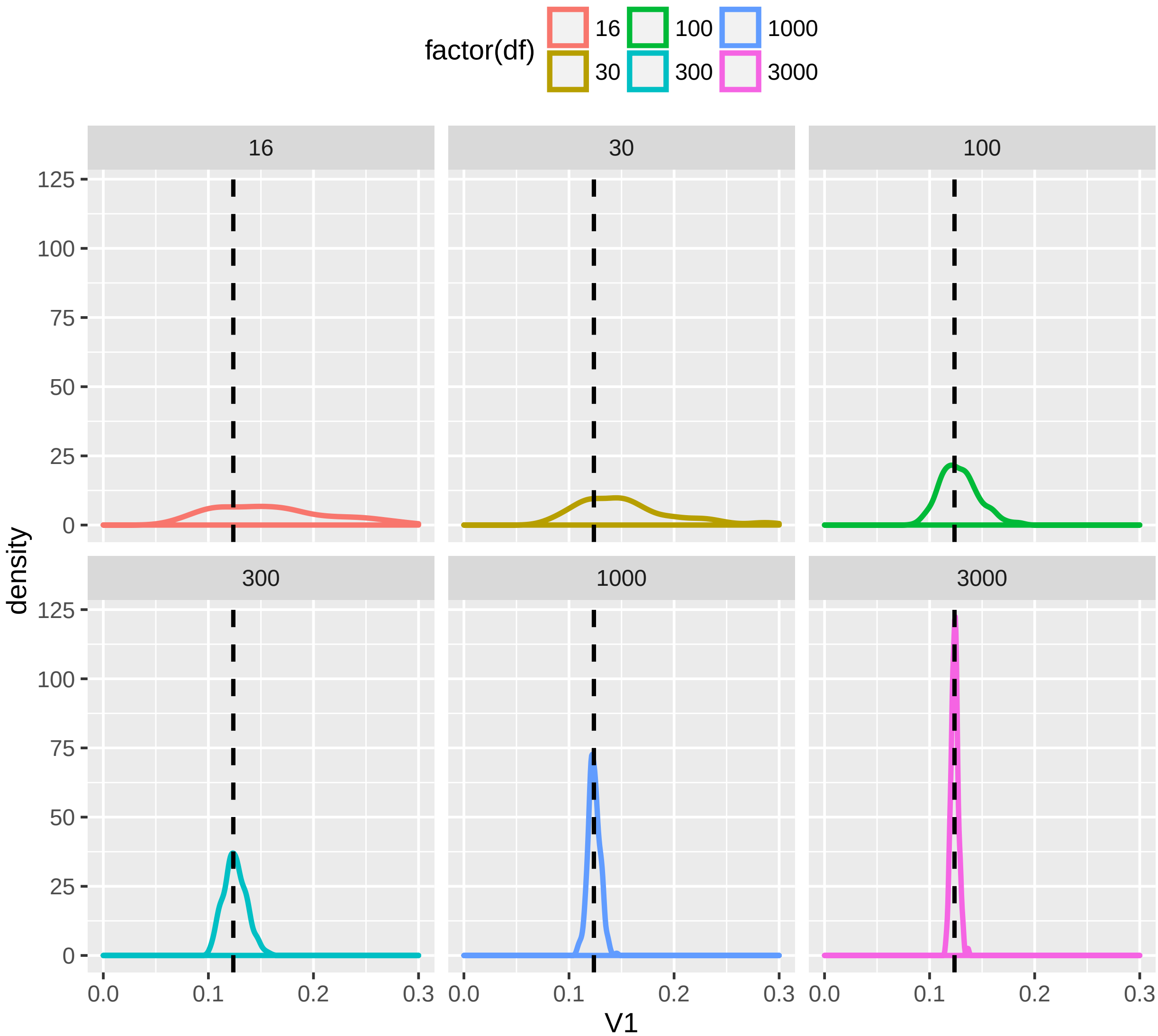
9.2 Explore how simulated random effect variances depend on `df`

- We are using the `simblock` function here (`?simblock`)
- Also, we will parallelize this calculation using `mclapply`

```
n <- length(unlist(omega))
x <- c(16,30,100,300,1000,3000)
sims <- mclapply(x, function(i) {
  metrumrg::simblock(nwish, df=i,cov=omega) %>%
    as.data.frame %>%
    mutate(df=i)
}) %>% bind_rows
```

Just look at **OMEGA_CL**:

```
ggplot(sims) + xlim(0,0.3) + facet_wrap(~df) +
  geom_density(aes(x=V1, col=factor(df), group=factor(df)), lwd=1) +
  geom_vline(xintercept=omega[1,1], col="black", lty=2, lwd=0.8)
```


factor(df)
16
30
100
300
1000
3000
16
30
100
300
1000
3000
density
125
100
75
50
25
0
0.0
0.1
0.2
0.3
V1