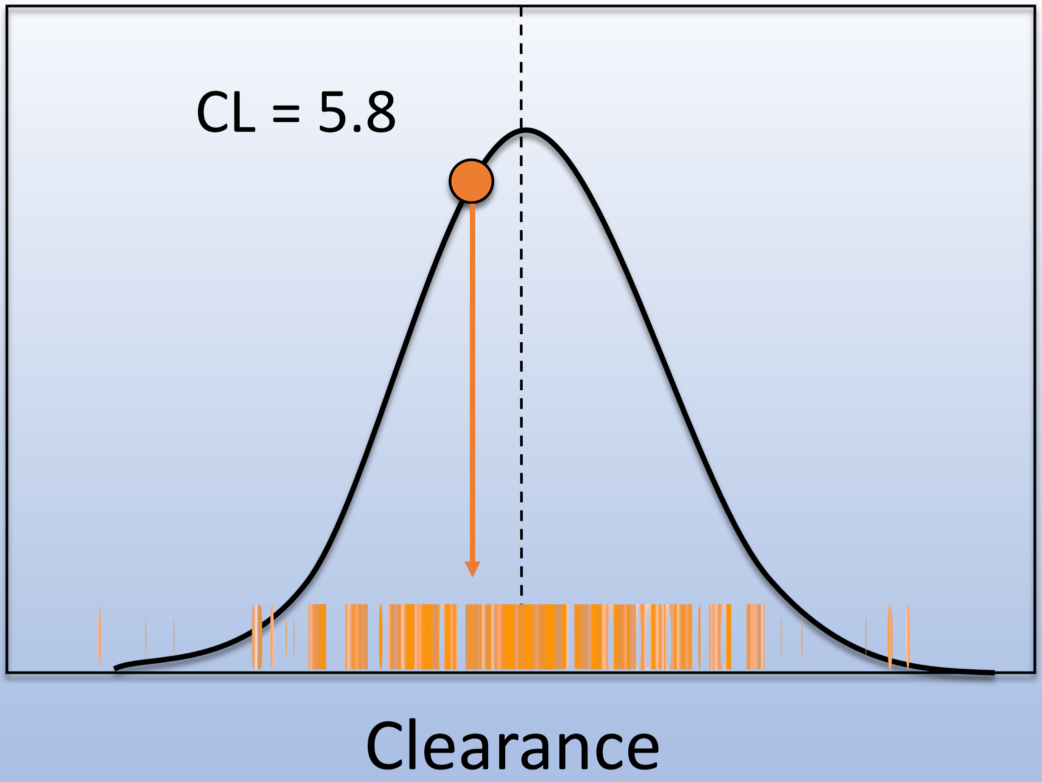
Monte Carlo Simulation

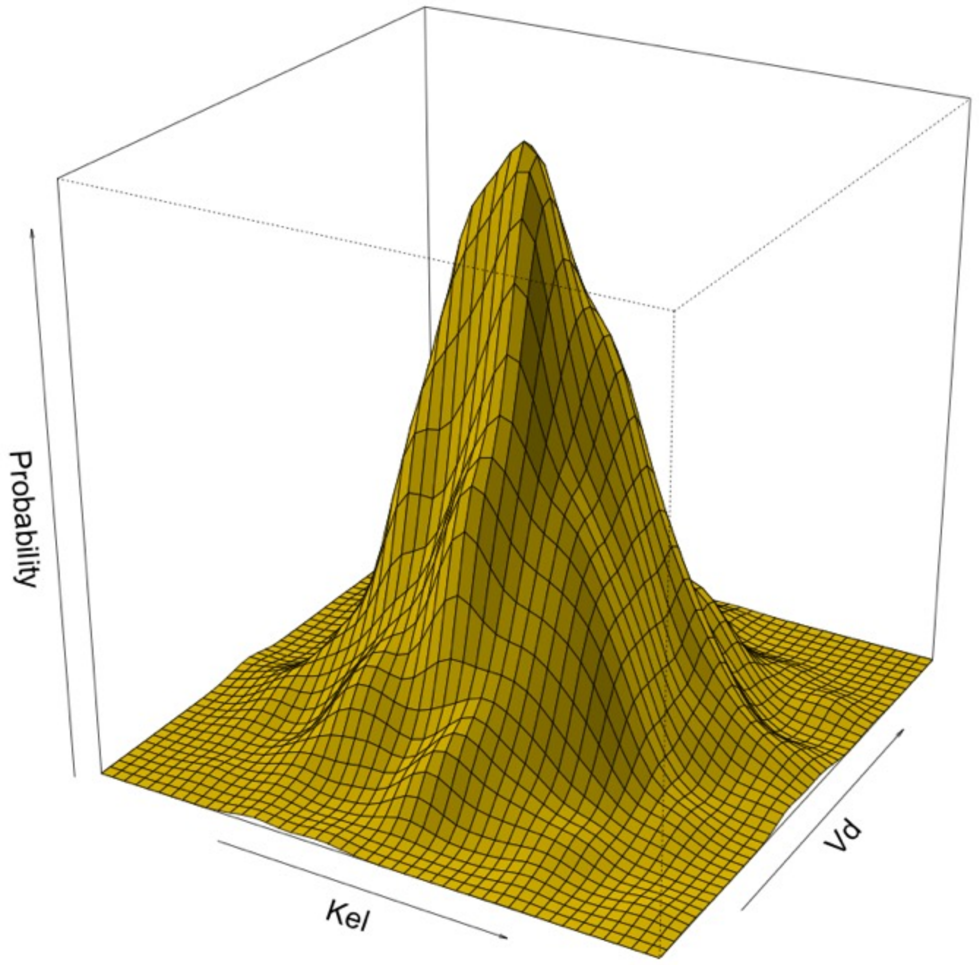
Michael Neely, MD
Professor of Pediatrics and Clinical Scholar
Director, Laboratory of Applied Pharmacokinetics and Bioinformatics
University of Southern California, Children's Hospital Los Angeles

Overview

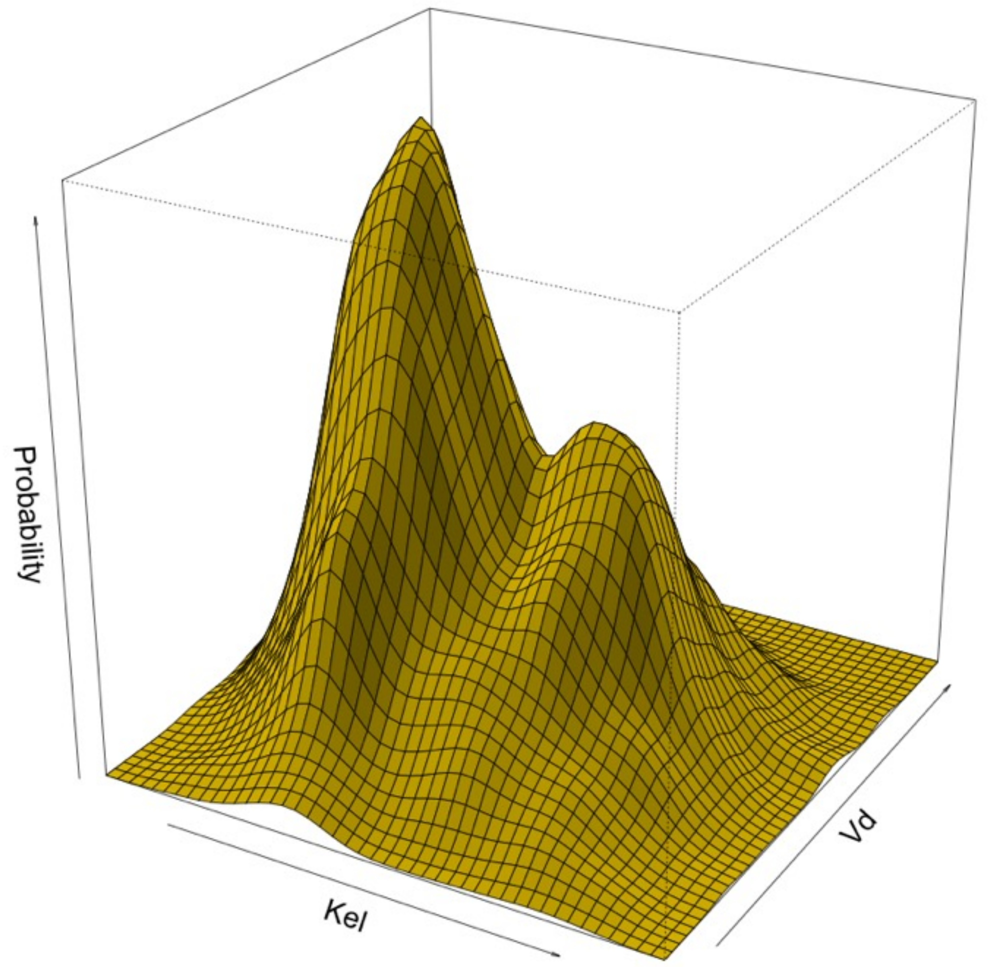
- Repeatedly draw random samples from a joint parameter probability distribution
- Calculate outputs given a set of inputs and the simulated parameter values

Sampling

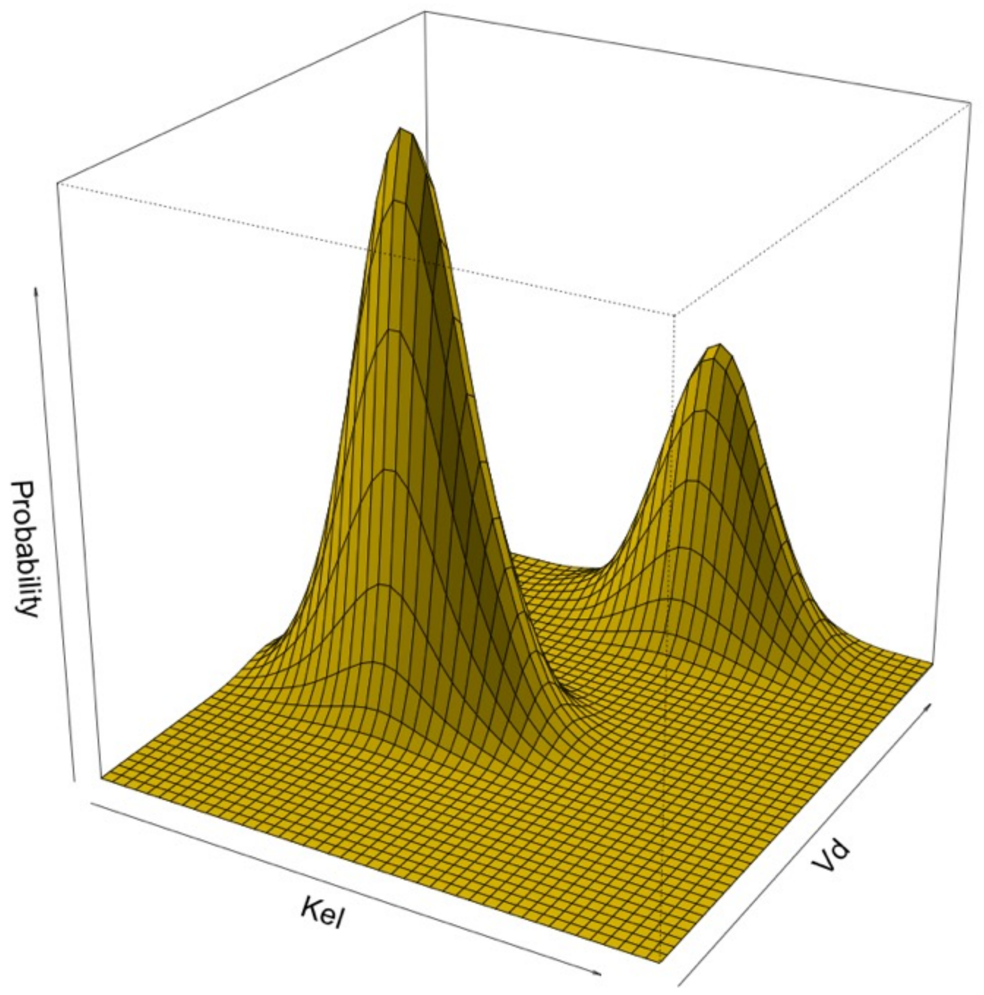




Unimodal **Unimodal**

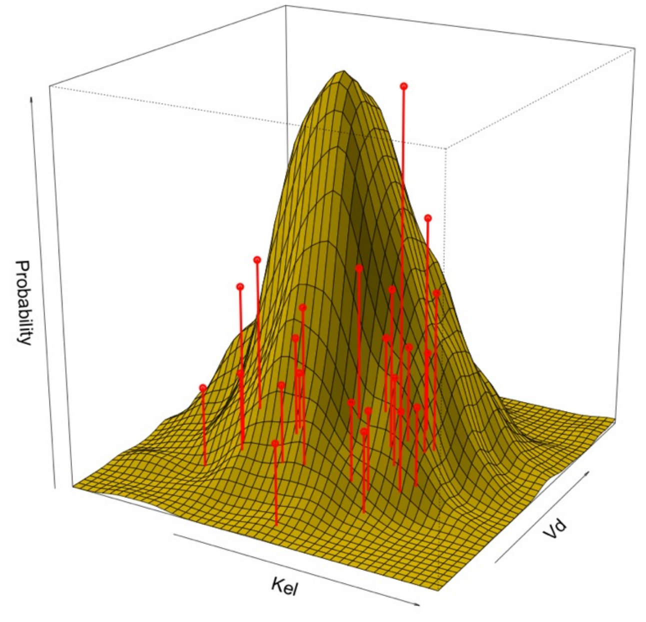


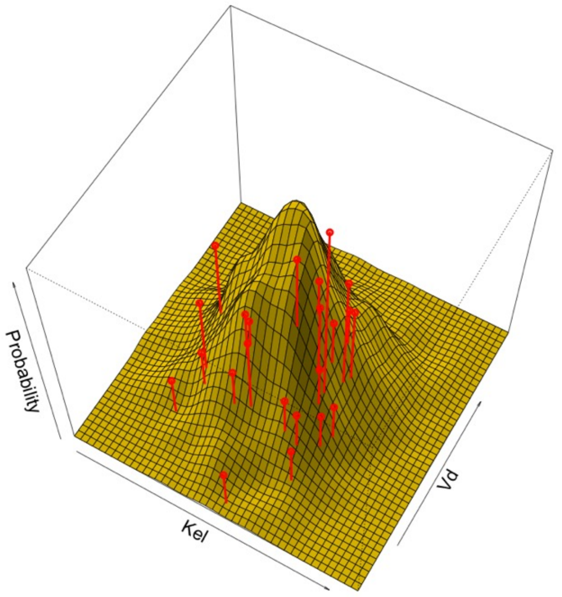
Bimodal **Unimodal**

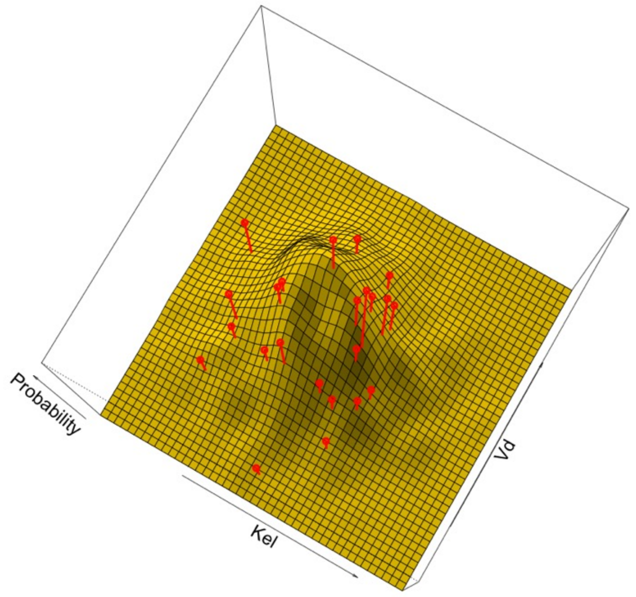


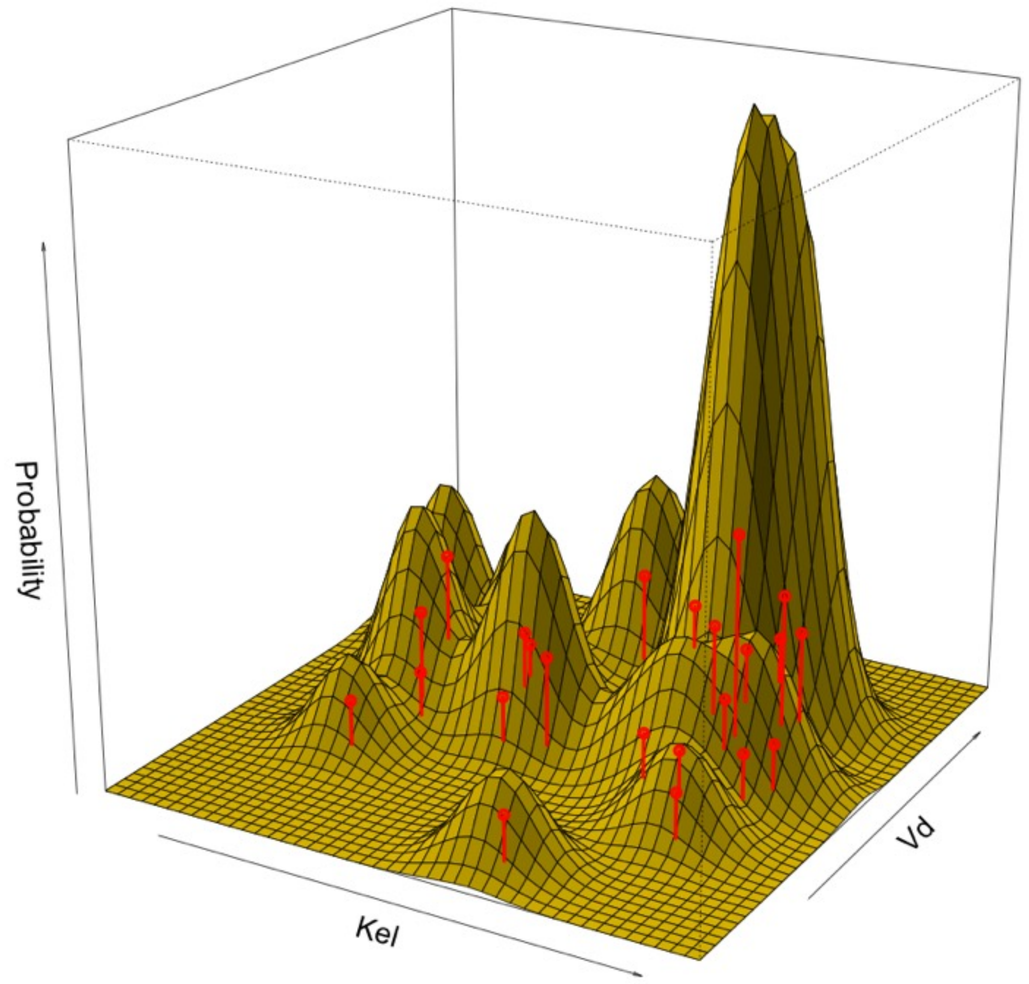
Bimodal **Bimodal**

NPAG support points

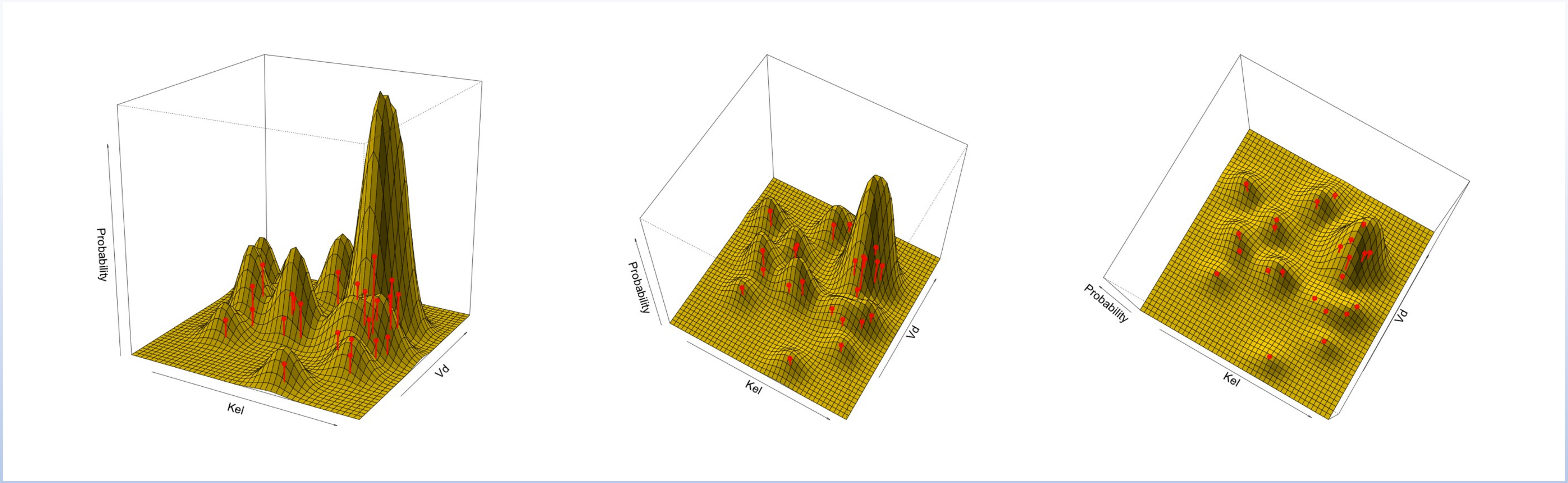








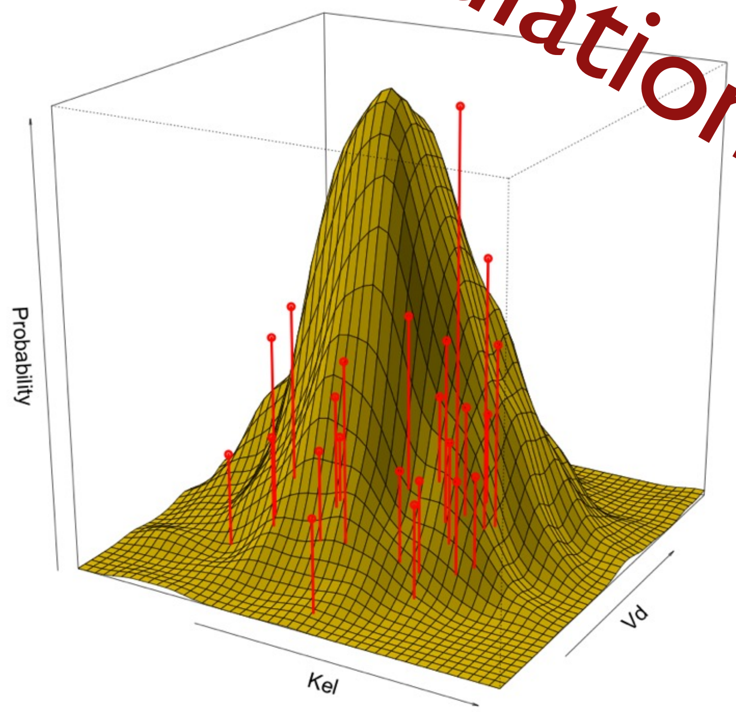
Multimodal **Multimodal**


Probability
Kel
Vd
Probability
Kel
Vd
Probability
Kel
Vd

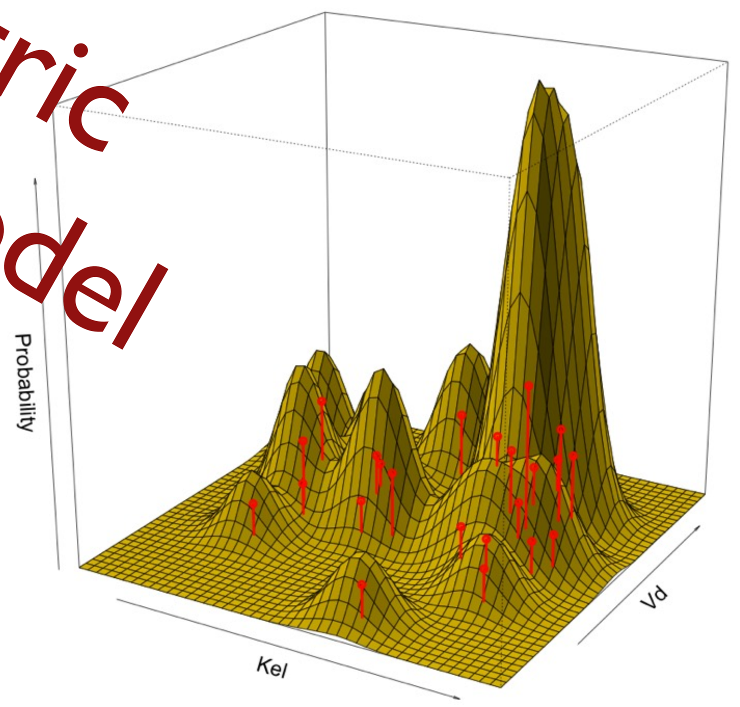
Two Simulation Choices

Kel	Vd	Probability
0.382617188	1.795898437	0.028571429
0.231738281	2.235839842	0.057142857
1.030468752	2.2421875	0.057938845
0.952148437	2.084960938	0.130811467
0.33701[illegible]	1.568347656	0.028571429
0.959960937	2.006835937	0.032164251

Nonparametric Population Model



Unimodal PM_sim$run(split=F)



Multimodal PM_sim$run(split=T)

Two ways of simulating

- `simdata <- PM_result$sim(…)`
- `simdata <- PM_sim$run(poppar, data = "data.csv", model = model.txt, …)`

PM_sim$run

```
PM_sim$run(poppar, limits = NULL,
model = "model.txt", data =
"data.csv", split = T, include,
exclude, nsim = 1000, predInt = 0,
covariate, usePost = F, seed = -17,
ode = -4, obsNoise, doseTimeNoise =
rep(0, 4),doseNoise = rep(0, 4),
obsTimeNoise = rep(0, 4), makecsv,
outname, nocheck = F)
```

PM_sim$run: poppar

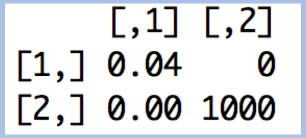
- poppar is the population parameter value distribution used to generate the random samples in the Monte Carlo simulation
- It can be a PMfinal object from NPAG or IT2B, e.g. `poppar <- PM_result$final`
  - This is the most common and easiest way to specify the prior
- Specify manually as a list of length 3 named any way, but in this order: weights, means, covariances
  - **weights**…a vector of length equal to the number of modes, with the weight of each mode. It must sum to one. E.g. `poppar <- list(wt=1,…)` or `poppar <- list(wt=c(0.4,0.6),…)`

PM_sim$run: poppar

- The second element of the poppar list is **mean**.
  - **mean**…a matrix with *n* rows and *m* columns, where n=number of modes, and m=number of random parameters to be simulated
  - E.g., `poppar <- list(wt=1, mean=c(0.4,100),…)` or `poppar <- list(wt=c(0.4,0.6), mean=matrix(c(0.4,100,0.8,100),nrow=2,byrow=T),…)`

PM_sim$run: poppar

- The final element of the poppar list is **cov**.
  - **cov**…An *m* x *m* matrix (regardless of the length of weights), where *m* is the number of random parameters, that contains the variances of the parameters on the diagonal, and covariances on the off-diagonal matrix members.
  - Typically when simulating from the literature, you will only have the diagonal members, which are the SD^2 (variance) for each parameter value distribution. You will usually not have the off-diagonal covariances.
  - E.g. `poppar <- list(wt=1, mean=c(0.4,100), cov <- diag(c(0.04,1000)))`



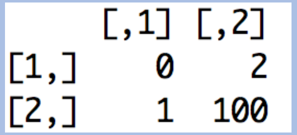
```
     [,1] [,2]
[1,] 0.04    0
[2,] 0.00 1000
```

PM_sim$run: limits

- Limits allows for truncated simulations
- Useful when mean is near 0 and SD is large, resulting in high probability of negative parameter values
- Also useful to prevent un-physiologically extreme parameter values

PM_sim$run: limits

- There are several ways to specify limits on the simulated parameter values
  - `limits=NULL`, the default, has no limits
  - `limits=NA`, use ranges in model file #PRI block
  - `limits=n`, use lower limit in model file #PRI block and multiply upper limit by n
  - `limits=c(m,n)`, multiply lower limit in model file #PRI block by m and upper limit by n
  - `limits=`*`matrix`*, create a custom matrix of npar rows and 2 columns, with the lower and upper limits for each random parameter in the first and second columns of each row, e.g. `limits=matrix(c(0,2,1,100),nrow=2,byrow=T)`



```
     [,1] [,2]
[1,]    0    2
[2,]    1  100
```

PM_sim$run: limits

- When limits are applied, any parameter set that contains values outside the limits will be discarded
- So there will usually be more total sets simulated than in the final set
- Pmetrics will report the total number simulated, the mean and covariance of all the simulated parameters as a simulation validity check (i.e. did you get the mean and covariance you requested)

PM_sim$run: model and data

- The model and data are supplied automatically when simulating from PM_result, e.g. `PM_result$sim()`.
- Can be customized, e.g. `PM_result$sim(data="new.csv")`
  - **model**…A PM_model object.
  - **data**…A PM_data object. May contain multiple subjects, each of whom can serve as a template for `nsim` versions. OUT values for EVID=0 rows will be replaced with simulated values (except -99 which will be retained as a missing value).

PM_sim$run: split

- **split**…Only applicable when `poppar` is a PM_final object from an NPAG run (<u>not</u> IT2B). Default is TRUE, which results in multi-modal, multi-variate prior, based on work of Goutelle et al.[1]
- Often a good way to simulate from NPAG prior as it preserves the non-parametric nature of the model

1. Goutelle S, Bourguignon L, Maire PH, Van Guilder M, Conte JE, Jelliffe RW. Population modeling and Monte Carlo simulation study of the pharmacokinetics and antituberculosis pharmacodynamics of rifampin in lungs. Antimicrob Agents Chemother **2009**; 53:2974–2981

PM_sim$run: include/exclude

- As for `PM_fit$run()` and others

PM_sim$run: nsim

- The number of simulations to make per subject in the template data file
- Default is 1000

PM_sim$run: predInt

- A powerful tool to shorten template data files by requesting simulated outputs at times other than those in the template. EVID=0 times in the template will always result in simulated output.

- Three ways to specify

  - `predInt=i`, From time 0 until max time for each template subject, simulate at interval of i

  - `predInt=c(n,m,i)`, Simulate an output starting at time n, continuing to time m, at interval i

  - `predInt=list(c(n1,m1,i1), c(n2,m2,i2),…)`, Simulate at discrete intervals

Simulating with covariates

- Pmetrics determines the means and variances of all covariates in the model, and the covariances between covariates and median Bayesian parameter values for each subject
- It then creates a new model file c_*model*.txt, where all non-fixed covariates have been moved from #COV to #PRI, causing them to be simulated

PM_sim$run: covariate

- A list that permits simulation with covariates, rather than fixing them to the same value for all simulated profiles. The list has 5 named components which must be named as such. Only the first item is mandatory.
  - **cov**…The name of a PM_cov object, e.g. `covariate=list(cov = run1$cov)`, which will be used to define the covariate-parameter covariances for simulation
  - **mean**…A named list of any covariate for which you wish to change the mean from the mean in the PMcov object, e.g. `covariate=list(cov = run1$cov, mean=list(wt=50))`

PM_sim$run:covariate

- **SD**…a named list of any covariate you wish to change the SD from that in the PM-cov object, e.g. `covariate = list(cov = run1$cov, mean = list(wt = 50), sd = list(wt = 10))`
- **limits**…similar to limits for parameters, here a named list with the lower and upper bounds you wish for a covariate, e.g. `covariate = list(cov = run1$cov, mean = list(wt = 50), sd = list(wt = 10), limits = list(wt = c(10, 100)))`

PM_sim$run:covariate

- **fix**…a vector of covariate names that you wish to fix to the values in the template data file. Any covariate not in fix will be simulated, with mean/sd in cov.1 and no limits, unless mean, sd or limits are specified, e.g. `covariate = list(cov = run1$cov, fix = c(“wt”, “age”))`

PM_sim$run: usePost

- Boolean (T/F) with default as false
- Only if `poppar` is NPAG PM_final object
- Use each subject’s posterior distribution to simulate
- Number of data templates must match subjects included

PM_sim$run: usePost

$$\begin{tikzcd}
\text{Template 1} \arrow[r] & \text{Posterior 1} \\
\text{Template 2} \arrow[r] & \text{Posterior 2} \\
\text{…} \arrow[r] & \text{…} \\
\text{Template N} \arrow[r] & \text{Posterior N}
\end{tikzcd}$$

PM_sim$run: seed

- Set a value as the start of the random number generator to ensure random but reproducible results.
- The value will be recycled for each subject in a template data file, ensuring that the difference between simulations is not due to different sets of simulated parameter values.
- To force different parameter value sets (not common), set seed values for each subject in the template, e.g. for 4 subjects, `seed = c(-10,1,5,57)`.

PM_sim$run: ode

- Just as for `PM_fit$run()`
- Ignored for models without #DIF

PM_sim$run: Noise

- **obsNoise**…Noise on simulated output values, ~N(0, SD),
  - SD=C0 + C1 * sim + C2 * sim^2 + C3 * sim^3.
  - Default obsNoise is 0; `obsNoise = NA` will use values in model file. Can also specify directly, e.g. `obsNoise=c(1,0.4,0,0)`
- **doseTimeNoise, doseNoise, obsTimeNoise**…Noise on dose times and amounts or observation times. Default=0 for all. Specify as C0, C1, C2, C3, e.g. `doseTimeNoise = c(0.01,0.1,0,0)`

PM_sim$run: makecsv

- **makecsv**…name of .csv file to be made from simulated output. Optional.
- Will provide warning if nsim>50 as this can take a very long time.
- If obsNoise=0, will use values in model file for C0, …, C3.

PM_sim$run: outname

- **outname**…The name for the output files. One file will be generated for each subject ID in the template data file. Each file will contain *nsim* sets of parameter values and outputs.
- Default is “simout”. Files will be called simout1.txt, simout2.txt, simout3.txt, etc.
- e.g. `outname="mysim"` will result in mysim1.txt, mysim2.txt, etc.

PM_sim$run: nocheck

- If true, suppress checking of the data file with `PMcheck()`. Default is FALSE.

SIMparse

- SIMparse is called at the end of `PM_sim$run()` to get simulation results back into R.
- Class: PM_sim, list

PM_sim$run(): combine

- **combine**…Default is FALSE. This means that each file retrieved by SIMparse will be added to a list of PM_sim objects
  - If simulation template has *N*≥2 subjects, `simdata <- PM_sim$run(…)` will result in simdata as a list of *N* PM_sim objects. For example to plot the second template, use `simdata[[2]]$plot()`.
  - If template has only one subject or `PM_sim$run(…, combine = TRUE)`, you do not need the list subset, i.e. use `simdata$plot()`

PM_sim

- A list of the following objects
  - **obs**…A data frame of simulated observations with 4 columns: **id**, **time**, **out**, **outeq. id** is the number of the simulated subject, which will have a unique ending appended if simulations are combined, such that id will become x.y with x being the simulated profile number and y being the simulation template number**. time** is the time of the simulated output, **out** of output equation number **outeq.**
  - **amt**…A data frame of simulated amounts with 4 columns: **id**, **time**, **out**, **comp**. **id** is as for obs, **time** is the time of the simulated amount, **amt**, in compartment number **comp.**
  - **parValues**…A data frame of the simulated parameter values, combined across files as necessary

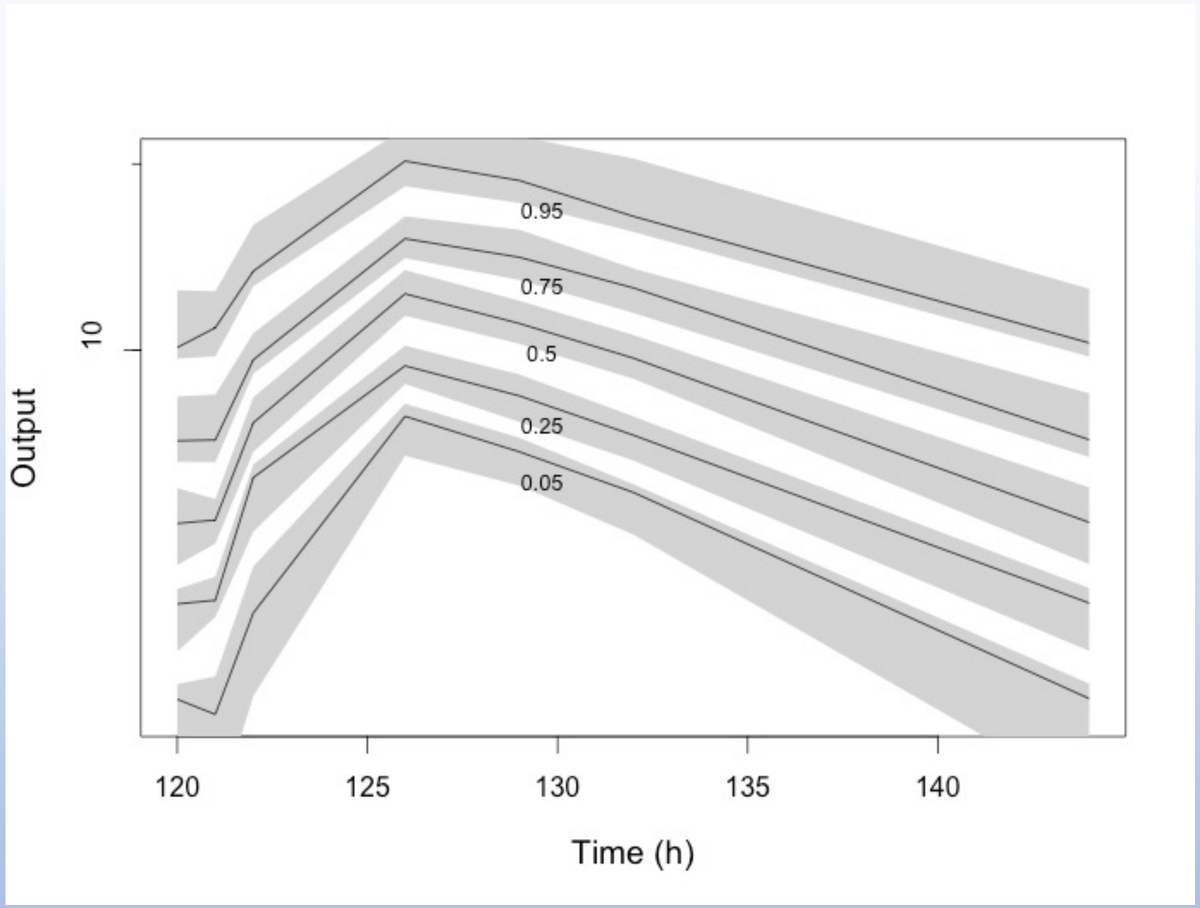
PM_sim

- **totalSets**…The total number of parameter sets simulated, which may be greater than the number of rows in parValues if some sets were discarded for being outside specified limits. For more than one file parsed, this will be the total number in all files.

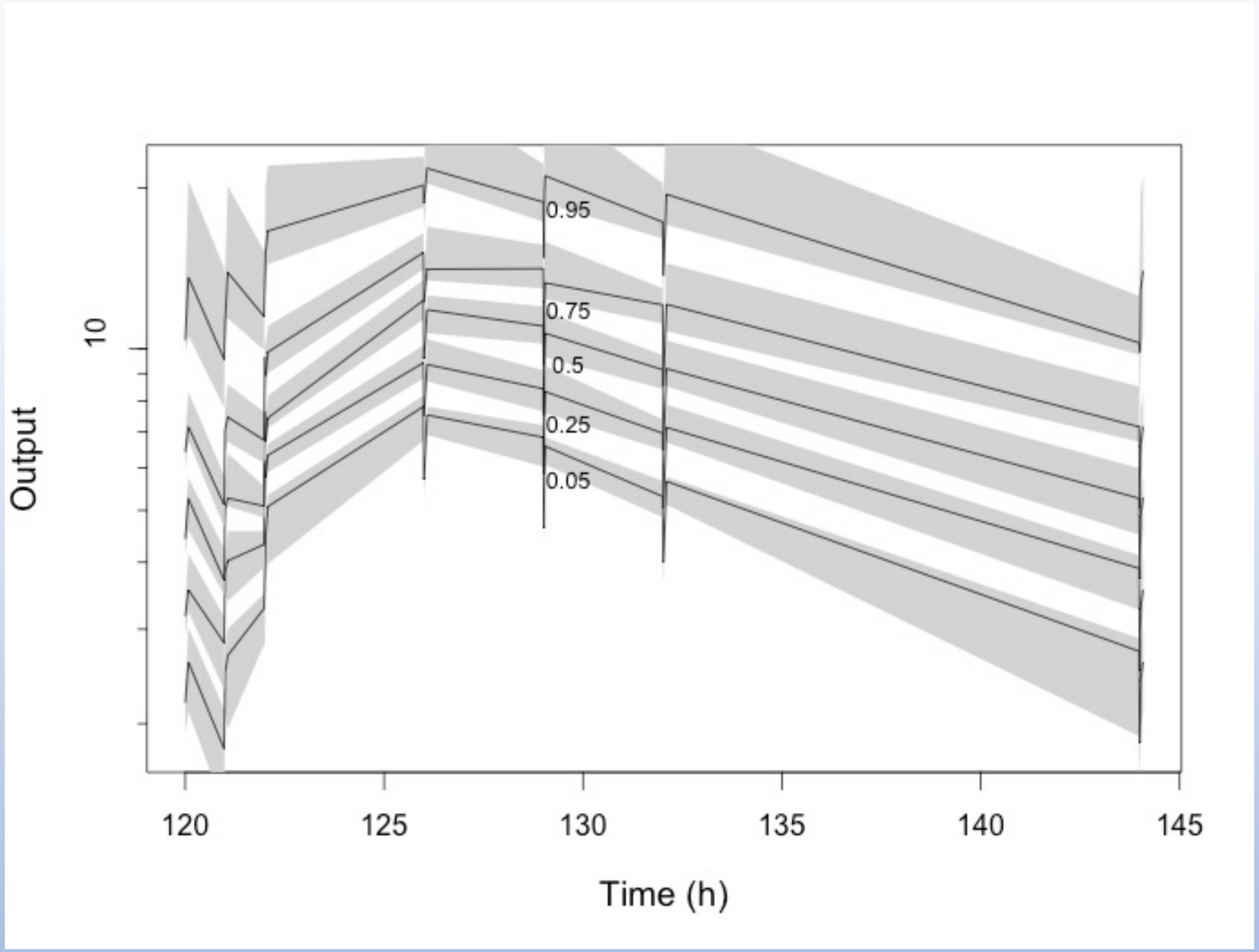
- **totalMeans**…The means of each simulated parameter based on all profiles in a given file (even those discarded for exceeding limits). For more than one file parsed, this will be the weighted averages for all simulations.

- **totalCov**…The covariances of the simulated parameter sets based on all profiles in a given file (even those discarded for exceeding limits). For more than one file parsed, this will be the weighted averages for all simulations.

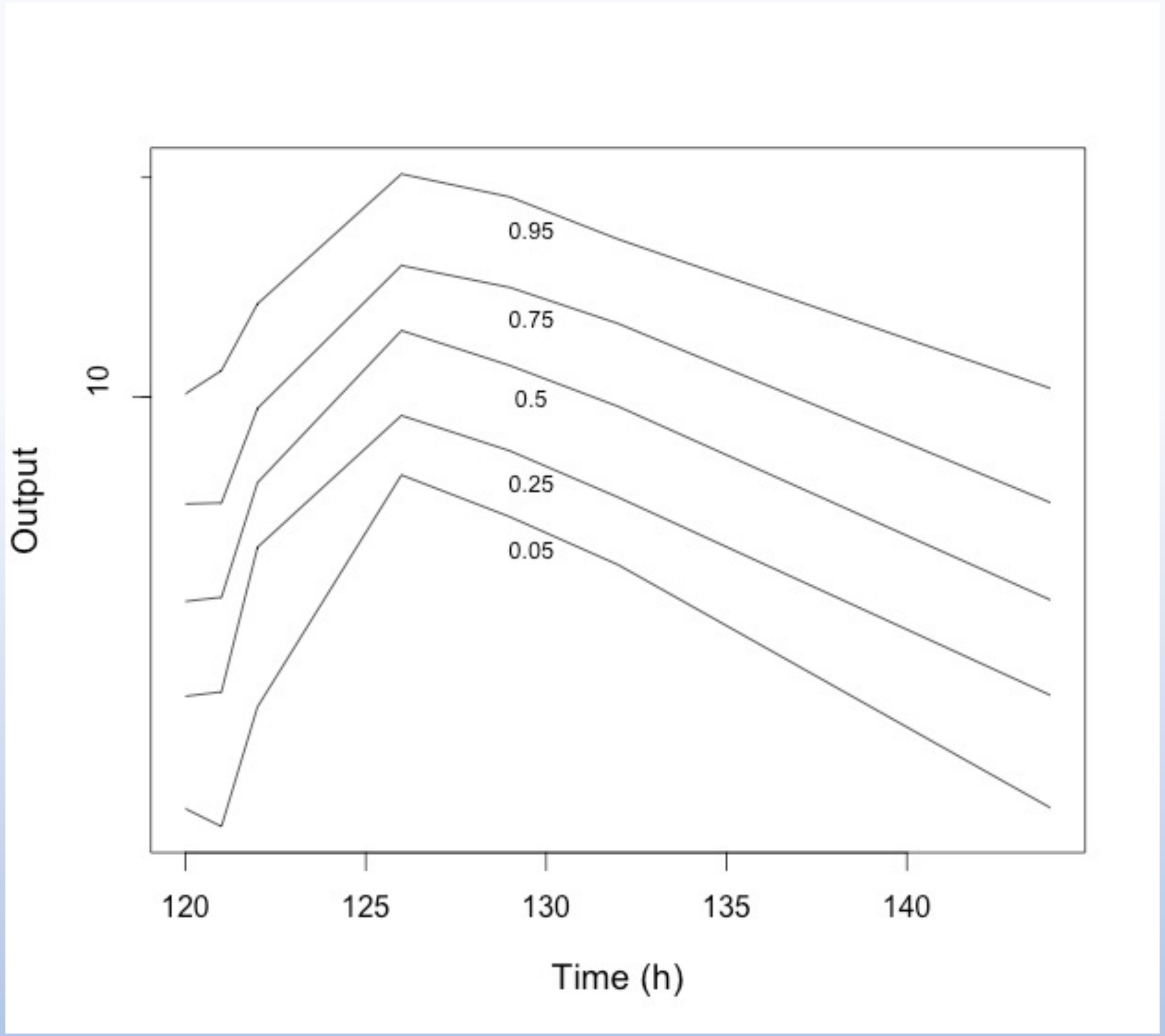
PM_sim$plot()



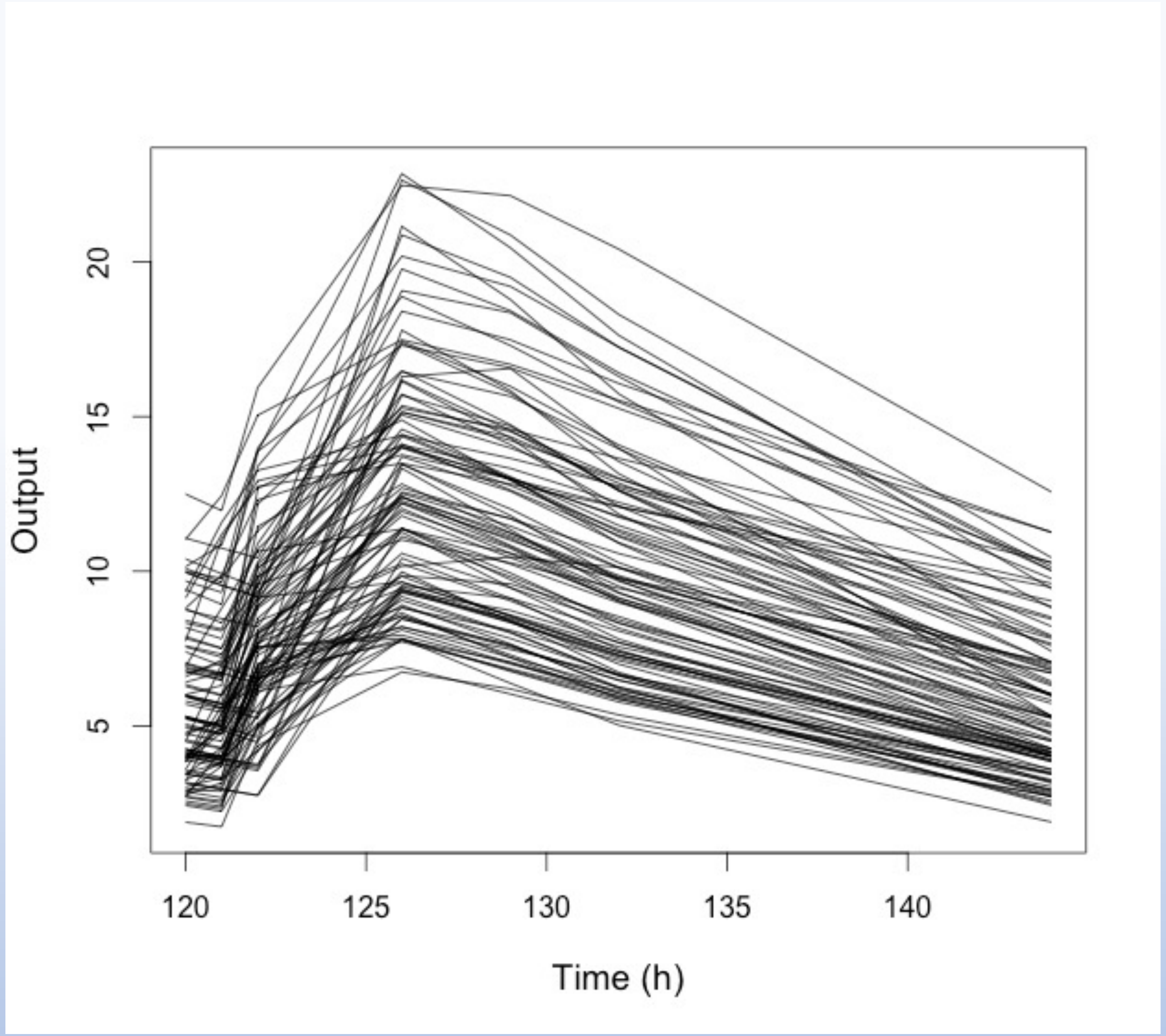
`Simdata$plot()`



```
simdata2 <- PM_sim$run(…,combine=T)
simdata2$plot()
```



`simdata$plot(ci=0)`



```
simdata$plot(probs=NA,log=F)
```