Virtual Comparative Trials and Bioequivalence Assessment: From Data-Based to Probabilistic Assessment

Celine Brochot, Frederic Y. Bois

Certara UK Limited, Simcyp Division, Sheffield, UK. Certara UK Limited

Corresponding author: Frederic Y. Bois,
Simcyp Division, Level 2-Acero, 1 Concourse Way,
Sheffield, S1 2BJ, United Kingdom.
Email: frederic.bois@certara.com

Keywords

Bayesian inference, Comparative clinical trials, Virtual bioequivalence assessment, Paliperidone, Sensitivity analysis.

Abstract

The recent emergence of virtual comparative trials, in particular for virtual bioequivalence (VBE) assessment, implies a formalization of their analyses. In recent VBE assessments, pharmacokinetic models informed with *in vitro* data and verified with small clinical trials data were used to simulate otherwise unfeasibly large trials. Simulated VBE trials are assessed in a frequentist framework as if they were real, despite the unlimited number of virtual subjects they can use. This may control consumer risk adequately but imposes unnecessary risks to producers.

We propose a fully Bayesian model-integrated VBE assessment framework to control consumer and minimize producer risk, and compare its performance with a data-based VBE workflow. We illustrate our approach with a case study on a hypothetical paliperidone palmitate generic long-acting injectable suspension formulation, using a validated population pharmacokinetic model published for the reference formulation. Our study shows that the fully Bayesian workflow is efficient and rewards data gathering and model-integration to make the best use of prior information. Safe space analyses in the two workflows differ because the accuracy of the second one is higher and gives a clearer estimate of the parameter region in which BE is expected.

Introduction

Bioequivalence (BE) clinical trials analyses check that two drug formulations do not lead to different average rate and extent of drug absorption in patient populations or surrogate healthy volunteers. The development of complex generic products can be challenging: long-acting injectable (LAI) formulations, for example, may require clinical trials lasting years; high inter- or intra-subject variability may force to use very large numbers of trial participants; costs can be prohibitive for such products. Virtual bioequivalence (VBE) testing uses a model, *in vitro* and abbreviated trial data (obtained in small-size studies), to generate realistic BE trials simulations, and uses them to assess BE for a particular formulation [1,2]. This can be much less expensive and time-consuming [3,4]. Motivations and conditions for the US-FDA approval of a generic product on the basis of a VBE assessment instead of a comparative clinical trial were recently explained [1]. An extensive validation process was developed at the occasion of that assessment. A publication by Hsieh *et al*. [2] describes a partly Bayesian VBE workflow integrating evidence from *in vitro* experiments, scientific literature and clinical trials.

Regulatory acceptance of VBE is quite new, and VBE methodology still in flux. Fortunately, most ingredients of a sound VBE framework are available. Modeling and simulations to design or replace clinical trials are common [5–12]. Model-integrated BE assessment is progressing rapidly [13–15] and VBE can easily use model-integrated approaches. Since VBE uses minimal clinical data, it makes sense to integrate historical data and *in vitro* evidence (*e.g.*, on dissolution [16]) to quantify the differences between reference and generic products. Bayesian inference is currently the best way to do that, even with mechanistic models [2,17–19]. Bayesian analysis of clinical BE trials has been discussed [20–26].

So far, simulated VBE trials have been submitted to non-compartmental analysis and standard hypothesis testing as if they were real trials. However, unlimited sample size is available to a virtual trial. At the limit, standard statistical tests would need to operate with zero-length confidence intervals, and error analyses become moot. Limiting arbitrarily the size of virtual trials is also sub-optimal for decision making. It lowers power and affects both producers and consumers because a safe product, potentially less expensive, might not be approved when it could be. Nobody benefits from curtailing the power of VBE assessments. We show that the above difficulties disappear if we adopt a more coherent fully Bayesian approach [20–22], shifting from a statistical assessment based on asymptotics to a more coherent probabilistic assessment.

In the following, we describe a fully Bayesian workflow for VBE assessment and compare it to a partly Bayesian workflow. We apply those workflows in a case study using simulated abbreviated trial data. The reference and test formulations will be assumed to differ in terms of a single drug-release parameter. We discuss issues related to model-integrated VBE, power and type I error assessments, as well as safe space analysis (a form of sensitivity analysis to estimate the range by which a generic formulation's characteristics can vary while maintaining bioequivalence [2]).

Methods

Bayesian workflows description

We investigate two Bayesian workflows (Figure 1). Workflow A steps mimic data-based BE assessment, except for the Bayesian calibration of a predictive model:

1. Definition of the model structure and prior parameter distributions, usually for the reference formulation. Mechanistic or empirical structural models can be used, but mechanistic should be preferred if *in vitro* data are available. The model should be sufficiently predictive of the key characteristics used to compare products: bioequivalence checks similar active drug absorption's rate and extent between test and reference formulations. Rate and extent are measured by peak plasma concentration (C_{max}) and area under the plasma concentration *vs*. time curve (*AUC*). It is therefore mandatory for those to be correctly simulated by the model for both test and reference.
2. Model recalibration with *in vitro* data and clinical data from an abbreviated BE trial provides estimates of the difference between test and reference formulations' critical quality attributes (CQA), part of the model parameters. An alternative is to first use the abbreviated trial data for model verification. If the model needs improvement, updating it one way or another is necessary and Bayesian recalibration using the abbreviated trial data can be tried. If the model does not need recalibration, it can be used directly to perform further simulations. Recalibration is mandatory for an empirical PK model, because there is no other way to inform the difference between test and reference.
3. Simulation of virtual trials of different sizes for BE assessment, type I, type II error and CQA safe-space analyses using data-based methods. Those methods are usually akin to the TOST test [27] with trial design-specific adaptations. Model-integrated methods have been proposed, whereby C_{max} and *AUC* are estimated by model-fitting [13]. The statistical tests

control type I error rate, the probability of declaring bioequivalent a formulation that is in fact not bioequivalent, which is clearly a consumer risk [15]. Type II error rate, the probability of wrongly declaring non-bioequivalent a formulation, is clearly a producer risk but also indirectly a consumer risk. Type II error depends on trial size and intra-group variances. The power of a trial (one minus type II error) is usually required to be at least 80% to avoid running wasteful trials for sponsors and participants. Since type II error and power strongly depend on the variability structure of C_{max} and AUC measures and on drug concentration measurements' uncertainty, the model should also be predictive of C_{max} and AUC variances.

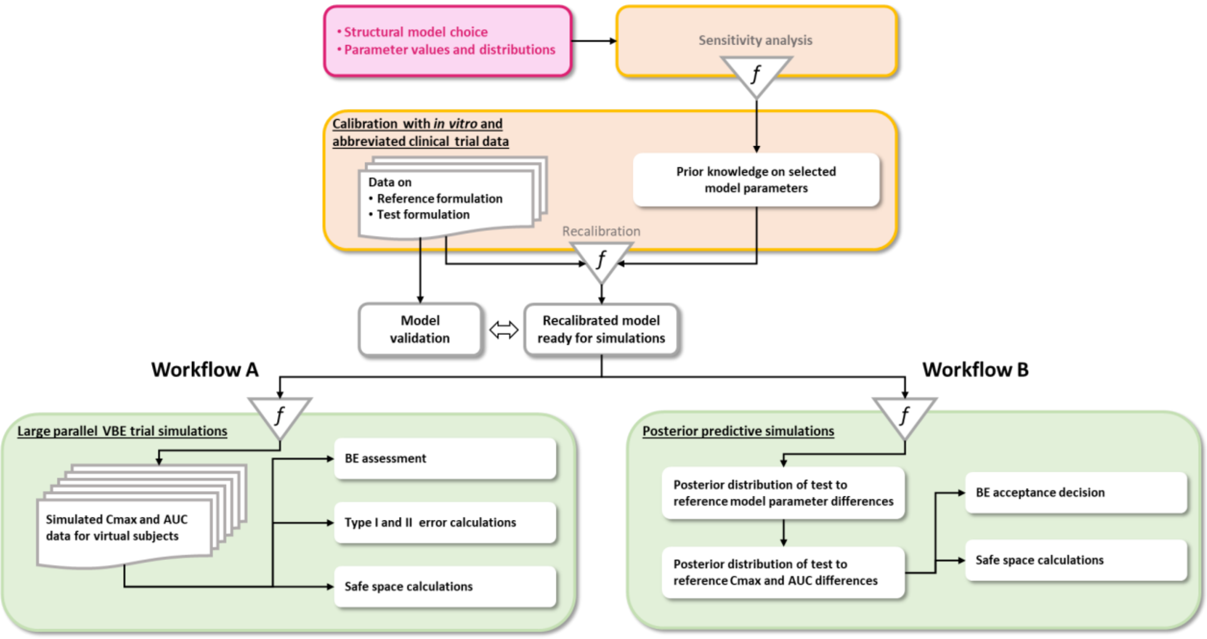
• Structural model choice
• Parameter values and distributions
Sensitivity analysis
f
Calibration with in vitro and abbreviated clinical trial data
Data on
• Reference formulation
• Test formulation
Prior knowledge on selected model parameters
Recalibration
f
Model validation
⇔
Recalibrated model ready for simulations
Workflow A
Workflow B
f
Large parallel VBE trial simulations
Simulated Cmax and AUC data for virtual subjects
BE assessment
Type I and II error calculations
Safe space calculations
f
Posterior predictive simulations
Posterior distribution of test to reference model parameter differences
Posterior distribution of test to reference Cmax and AUC differences
BE acceptance decision
Safe space calculations

Figure 1: Two VBE workflows. On the left (A), a partly Bayesian data-based assessment workflow, on the right (B), the fully Bayesian workflow we propose.

Workflow B differs workflow A at the last step. There is uncertainty about the difference between test and reference because information is imperfect and all the model parameters calibrated at step 2 of the workflow, even without recalibration with *in vivo* data, have a joint posterior probability distribution to which all components of variability and uncertainty contribute. Therefore, average C_{max} and AUC differences between test and reference have a joint posterior predictive distribution which can be estimated. With such a posterior distribution, the Bayesian strict equivalent of the current standard regulatory rule (focusing on population bioequivalence [28]) is to declare bioequivalence if the probability that both C_{max} and AUC differences fall within the 0.8 to 1.25 interval is equal or superior to 0.90. Questions about type

I and type II errors of a statistical test for a simulated trial disappear from our concerns. However, concerns about making a correct decision are still present and relate directly to model validation. Safe-space analyses are still possible and valid though. With nonlinear PK models, the posterior predictive distribution of C_{max} and *AUC* differences can be estimated by Monte Carlo simulations.

Case study: VBE of generics for long-acting injectable products

One of the most important problems in the management of schizophrenic patients is poor medication adherence [29]. LAI formulations, which can release drug over months, improve adherence to treatment. The first marketed LAI suspension of paliperidone palmitate, an antipsychotic agent [30,31], called INVEGA SUSTENA® or PP1M in the following, is usually injected once per month. A more recent formulation (INVEGA TRINZA®, PP3Mr in the following) can be injected every three months. Reliable population PK models have been developed and published for PP1M and PP3Mr [30,31]. Those models were developed using clinical data collected in phase I and phase III trials. The subjects received an injection of PP1M (dose range 50–150 mg eq.) every month for 4 months; and then switched to PP3Mr (dose range 175–525 mg eq.) with an injection every 3 months for one year (*i.e.*, 4 injections in total). We also developed a version of the model that can simulate cross-over clinical trials with sequences PP1M, PP3Mr, PP3Mt or PP1M, PP3Mt, PP3Mr. We will assess our VBE workflows with those models.

Models for PP long-acting injectables

The joint PP1M and PP3Mr models we use [30,31] is illustrated in Figure 2. It is a two-compartment model with a depot and a central compartment. The PP3Mr model has two saturable release processes (described by Hills equations) from the depot compartment. The joint model was developed and checked in a population framework with large clinical datasets of the innovator’s drug [31]. We use the same framework (see Appendix, sections Paliperidone palmitate long-acting injectable PK models and Hierarchical population model for details).

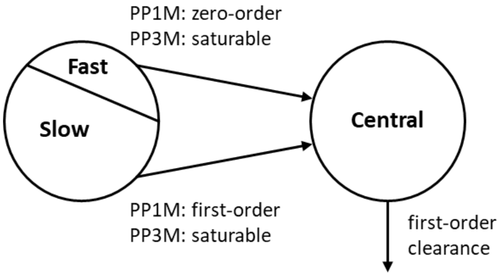

Figure 2: Structural part of the population PK models used for the innovator's PP 1-month (PP1M) and 3-month (PP3Mr) long-acting injectable products[28,29].

To model differences between the reference (PP3Mr) and test (PP3Mt) formulations, we introduced a vector of relative changes, δ, affecting the geometric means of the six drug-release and absorption parameters of the model, f_3 (the fraction of PP rapidly released), $k_{as3,max}$ (maximum release rate from the slow depot), $k_{ar3,max}$ (maximum release rate from the rapid depot), $k_{as3,50}$ (Hills coefficient for the slow-release depot), $k_{ar3,50}$ (Hills coefficient for the rapid depot), and γ (Hills power), in that order. Each mean ($\mu_{i,test}$ in the following equation) for the test formulation, given the reference formulation value $\mu_{i,ref}$ and the relative change δ_i, was computed as:

$$\mu_{i,test} = \delta_i \times \mu_{i,ref}, \text{with } i \in \{1, \dots, 6\} \qquad (1)$$

Recalibration of the model with simulated abbreviated parallel clinical trial data

In the absence of an actual abbreviated trial on PP3Mr and PP3Mt formulations, we simulated an abbreviated parallel BE clinical trial with 25 subjects per arm. All the model parameters with prior uncertainty or population variability where randomly sampled to generate virtual subjects. The same dosing regimen and sampling scheme as in [31] were applied. In both arms, a final PP dose of 525 mg eq. was tested; initial PP1M dose was 150 mg eq. Plasma concentrations simulated at 54, 55, 57, 59 and 63 weeks for each individual were used for model recalibration. The calibration dataset therefore consisted of 250 measurements from 50 individuals. Differences between PP3Mr and PP3Mt drug-release parameters were simulated at the population average level by setting the value of the second component of vector δ, *i.e.* δ_2, to 1.05 (5% difference). This increases $k_{as3,max}$ population mean by 5% for test above reference. The other components of δ were set to 1.0 (no difference). Parameter $k_{as3,max}$ was determined to be the most influential on C_{max} and (partial) *AUC* at steady state, during the last dosing period;

it conditions the rate of release from the slow depot compartment (see Appendix, section Sensitivity analysis of impact of drug-release parameters on *Cmax* and *AUC*).

The above simulated abbreviated trial is the only source of information we should consider to estimate the differences between parameters of the test and reference formulations. Estimating those differences is important to simulate realistic final BE trials. Because our population PK model has strong prior information on the reference formulation parameters, a Bayesian approach is well suited to estimate the value of the difference δ_2. Metropolis-Hastings Markov-chain Monte Carlo (MCMC) sampling was used to obtain a sample of parameter values from their joint posterior distribution given the abbreviated trial data [19]. We fixed the population means and variances to the central values (maximum likelihood estimate, MLE, values in [31]) of their prior distributions. We set residual error σ^2 to the MLE of [31]. It would not make sense to update those parameters on the basis of a small trial, and we know by construction that the subjects from the reference trial arm are drawn from those priors. Subject-level parameters f_1, f_3, $k_{a,1}$, $k_{ar3,50}$, $k_{as3,max}$, $k_{as3,50}$, *CL*, *V*, and $Q_{central}$ (0) were estimated jointly with the values of the difference test *vs*. reference parameter δ_2. A vague lognormal prior with geometric mean at 1 and geometric SD at 2 was assigned to δ_2. The prior of δ_2 can be seen on Figure 5, it approximately spans a factor 8.

Four MCMC chains of 10000 iterations were run, the first 2500 iterations were discarded. Convergence of the remaining 4×7500 was checked using Gelman and Rubin diagnostic [32].

Large parallel virtual trial simulation, BE assessment, type I and II errors analyses

At this step, workflow A and workflow B diverge. Workflow A is data-based: We simulated a “realistically large” virtual parallel BE trial and analyzed it as a standard BE trial. Since the trial design is parallel, a simple TOST test was used on the data-based estimates of C_{max} and partial *AUC* over the last dosing period (*AUC* was estimated by the trapezoidal rule). A simulation-based power analysis (see Appendix, section Parallel and Cross-over Trials Power Calculations (Workflow A)) indicated that 130 subjects would be adequate given all the prior information we had on PP kinetics with the reference formulation. To simulate a parallel BE trial with 130 subjects per arm, we fixed the population means and variances to the central values of their prior distributions (MLE value in [31]). Virtual subjects were sampled from their population distribution. We set residual error σ^2 to its MLE [31]. The value of δ_2 was set to its mean estimate after calibration with the abbreviated trial data. The other δ values were set at 1. The dosing schedule and sampling times were the same as in the above trials.

Workflow B uses Monte Carlo simulations to obtain an estimate of the joint posterior predictive distribution of the ratios δ_{Cmax} and δ_{AUC} between the population geometric means of C_{max} (and AUC, respectively) for the test and the reference formulations. To be fully model-integrated, we estimated C_{max} by computing PP plasma concentration at 100 time points during the last dosing period (the system is at steady-state, and we checked that 100 points was largely enough to get a stable estimate of C_{max}); AUC was computed by numerical integration (adding one ODE to the system of ODEs to solve) over the same period. We formed the δ_{Cmax} and δ_{AUC} ratios for 1000 simulated trials with 1000 subjects per arm each. Each trial was simulated exactly as the large trial described above, except for the number of subjects.

Large cross-over virtual trial simulation, BE assessment, type I and II errors analyses

We also simulated a realistically large cross-over BE trial and analyzed it using standard tests. In this case, treatment, sequence and period effects are estimated by regression and a TOST test is used for the mean ratio on the data-based estimates of C_{max} and partial AUC over the last PP3Mr or PP3Mt dosing periods (AUC was estimated by the trapezoidal rule). A simulation-based power analysis (see Appendix, section Parallel and Cross-over Trials Power Calculations (Workflow A)) indicated that 130 subjects would be more than adequate given all the prior information we had on PP kinetics with the reference formulation. To simulate a parallel BE trial with 130 subjects per arm, we fixed the population means and variances to the central values of their prior distributions (MLE value in [31]). Virtual subjects were sampled from their population distribution, as above. We set residual error σ^2 to its MLE [31]. The value of δ_2 was set to its mean estimate after calibration with the abbreviated trial data. The other δ values were set at 1. The dosing schedule and sampling times were the same as in the above trials.

Safe-space calculations

For workflow A, 1000 virtual trials with 130 subjects per arm were run as above, except for sampling each element of δ from a uniform [0.5,2] distribution. This generated a large number of bioequivalent and non-bioequivalent test formulations. We computed the geometric mean ratios test over reference for C_{max} and AUC for those simulated trials. BE for C_{max} and AUC of each trial was assessed with a TOST test. BE was declared if it passed for both C_{max} and AUC. Color-coded TOST BE passes and fails were plotted against the values sampled for the different components of vector δ.

For workflow B, safe-space calculations required computing the posterior predictive distribution of δ_{Cmax} and δ_{AUC} over the whole drug-release parameters' space. However, we know that the safe-space limits should be crisp (because they are not blurred by uncertainty induced by limited trial size), and the workflow A safe-space calculations gave us a rough estimate of safe-space shape. We therefore focused on determining the boundaries of the safe-space for the two most critical absorption parameters (f_3 and $k_{as3,max}$) of the PK model. For each trial point (near the boundary) of the f_3 and $k_{as3,max}$ space, we based our decision for BE on the δ_{Cmax} and δ_{AUC} ratios from 1000 simulated trials with 1000 subjects per arm each. Each trial was simulated exactly as the large trial described above, except for the number of subjects.

Software

We coded the structural PK model as a C-language routine callable from workflow *R* [33] scripts using the Nimble *R* package [33–35] to perform Monte Carlo simulations, Bayesian inference and posterior analyses. The corresponding codes are given in Appendix, section Computer codes).

Results

Prior model checking

To check our PM model implementation we compared the simulations obtained with it to the measured paliperidone concentrations reported in [31]. Figure 3 presents simulated paliperidone plasma concentration measurement percentiles overlaid with the reported data summaries for several PP1M and PP3Mr dosages. One hundred clinical trials with 130 subjects were simulated to integrate uncertainty in population parameters values, inter-individual variability, and residual error. The median, 5th and 95th percentiles of the plasma concentrations for the 130 subjects were computed in each trial. The blue bands in Figure 3 are bounded by the 5th and 95th percentiles of the distributions of those three quantiles over the 100 trials. Despite missing information about the subjects' covariates, our implementation of the model captures well the reported PP1M and PP3Mr kinetics, including inter-individual variability and residual error. Summary ratios of predicted over observed PP3Mr median concentrations values do not exceed 1.25. The code used for that plot (Population PK model implementation in Nimble R v8_pop) and further details are given in Appendix, section Prior model checking.

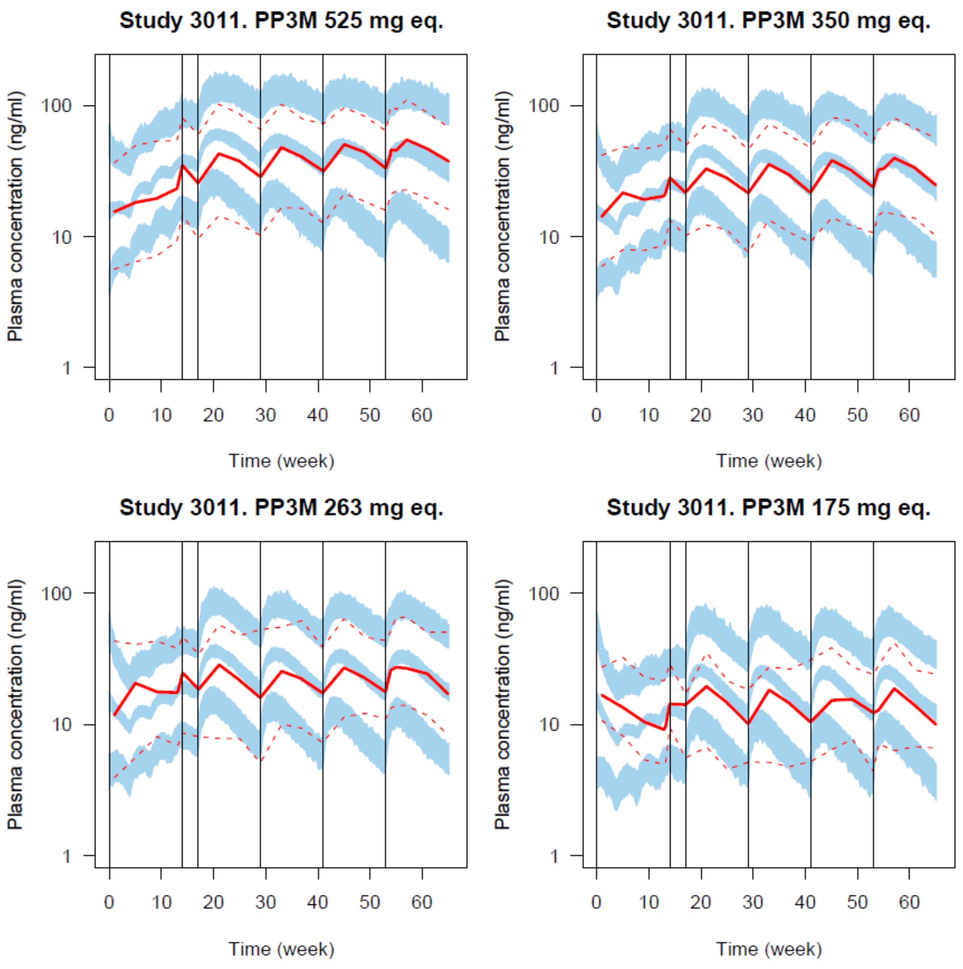

Figure 3: Simulated plasma PP concentrations with the PP1M and PP3Mr model for all validation plots in the original paper[29] for various dosages. Four injections of PP1M (dose range 50 to 150 mg eq.) are followed by four injections of PP3Mr (doses indicated in each panel). A total of 100 of clinical trials with 130 subjects were simulated. The solid and dashed red lines represent the median, 5th and 95th percentiles of the observations as reported in the original publication; the shaded blue areas represent the 90 % confidence interval of the median, 5th and 95th percentiles predicted by our implementation of the model.

Abbreviated clinical trial simulation

Figure 4 shows the simulated concentration data, C_{max}, and AUC / Δ_t for the simulated abbreviated clinical trial. C_{max} and AUC were computed for the last PP3Mr or PP3Mt dosing period in which plasma concentration were sampled at weeks 54, 55, 57, 61 and 65, and Δ_t is corresponding time span (11 weeks). AUC / Δ_t is an average concentration and can be plotted on the same scale (see also in Appendix, section Abbreviated clinical trial simulation summary plot).

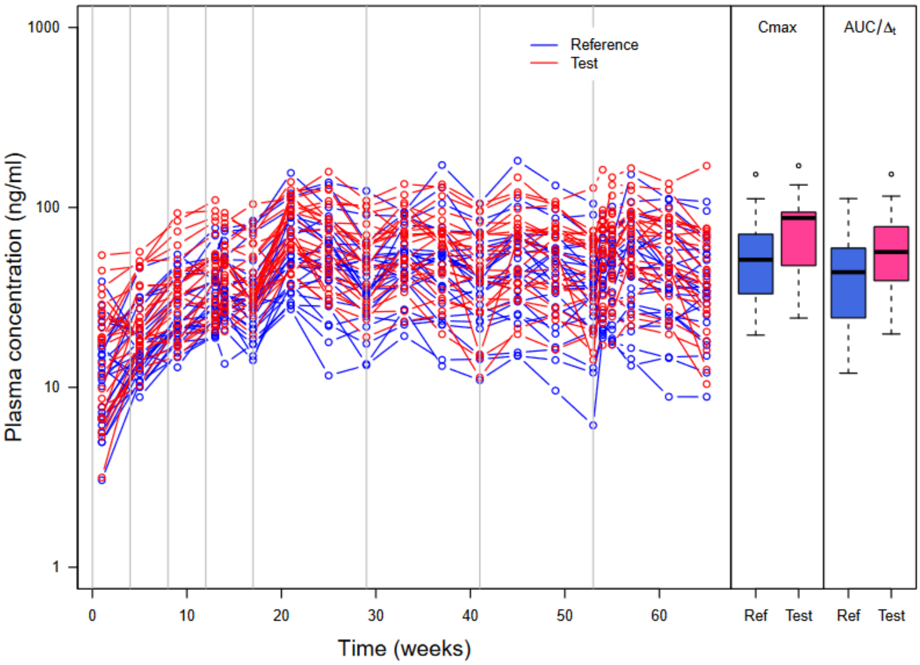

Figure 4: Simulated plasma PP concentrations, C_{max}, and AUC / Δ_t for 25 subjects per arm in a parallel abbreviated virtual trial when parameter $k_{as3,max}$ was increased by 5% from the value of the reference formulation. The subjects received four injections of PP1M (150 mg eq.) prior to four injections (525 mg eq.) of PP3Mr (blue) or PP3Mt (red). The grey lines indicate the injection times.

C_{max} and AUC are increased on average in the test formulation (geometric mean ratio test/reference at 1.38 for both). Such an increase is large compared to the couple of percents expected with a 5% increase in $k_{as3,max}$ (see in Appendix, section Sensitivity analysis of impact of drug-release parameters on *Cmax* and *AUC*). A standard two-one-sided *t*-test (TOST test) on either C_{max} or AUC did *not* allow to conclude to bioequivalence, due to the size of the difference between formulations, a low number of subjects and large inter-individual variability in C_{max} and AUC (51% and 54%, respectively).

Recalibration of the model with the simulated abbreviated clinical trial data

MCMC sampling was used to estimate the joint posterior distribution of δ_2 and of the subject-specific parameters $f_{1,i}$, $f_{3,i}$, $k_{a,1,i}$, $k_{ar3,50,i}$, $k_{as3,max,i}$, $k_{as3,50,i}$, CL_i, V_i, and $Q_{central,i}(0)$ for the two groups of the abbreviated clinical trial (451 parameters altogether). Subject-specific parameters can be considered as nuisance parameters that were integrated over. Sufficient convergence was achieved for all parameters (see Appendix for a convergence plot and a histogram of all $\hat{R}$ values in section Convergence of the model recalibration by MCMC sampling). The posterior distribution of δ_2 had a geometric mean of 1.42, a geometric SD of 1.19, and a 95 % credibility interval of 1.0 to 1.97 (see Appendix, section Posterior distribution summary for parameter $\boldsymbol{\delta_2}$). Figure 5 plots its empirical posterior distribution (well approximated by a normal distribution), together with its prior. The posterior mean of δ_2 is quite high compared to the value (1.05) used for simulating the clinical data because we used a random abbreviated trial for recalibration in which test subjects happen to behave quite differently from the reference subjects. Only a much larger trial would be likely to yield more accurate estimates of differences between test and reference. Note that this is conservative from a consumer safety point of view.

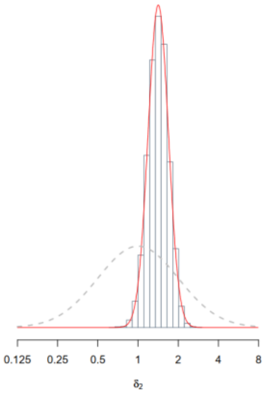

Figure 5: Posterior distribution of δ_2 (histogram and smooth density curve). The dotted line shows its prior distribution. The posterior is much more precise.

The posterior fit of the recalibrated model to the abbreviated trial data is very good, as shown in Figure 6 (see also in Appendix, section Observations *vs*. predictions plot for the recalibration step).

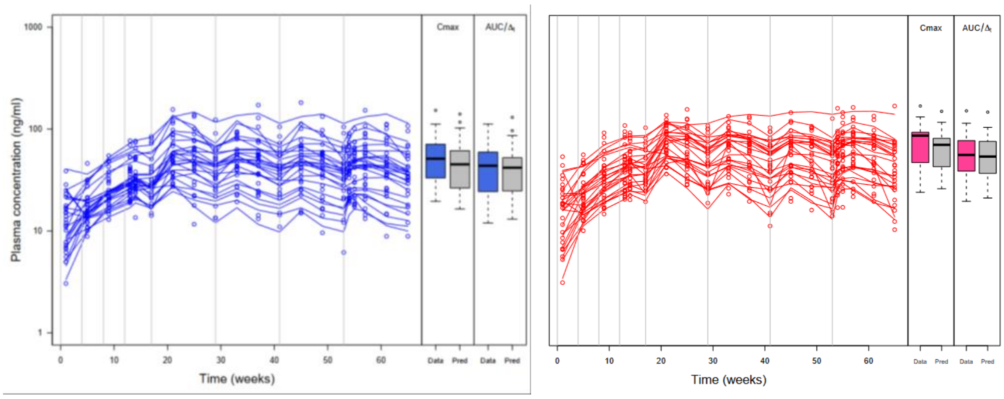

Figure 6: Abbreviated trial data (circles) and posterior model predicted profiles obtained with the maximum posterior population PK parameter values for the reference formulation (left panel) and the test formulation (right panel). Boxplots of C_{max}, and AUC / Δ_t are shown for the data (blue or red) and for the model predictions (grey). C_{max}, and AUC / Δ_t are noticeably higher for the test formulation.

Large virtual trial simulation, BE assessment, type I and II errors analyses

Partly Bayesian data-based workflow

A plot of a simulated large trial plasma concentration data with C_{max}, and AUC / Δ_t is shown on Figure 7 (see also in Appendix, section Large virtual trial trial simulation summary plots). In this trial, C_{max} and AUC are increased on average in the test formulation (geometric mean ratio test/reference at 1.08 and 1.06, respectively). The difference between $k_{as3,max}$ population means in test and reference formulations, δ_2, was set to 1.42. A standard two-one-sided *t*-test (TOST test) on either C_{max} or AUC concludes to bioequivalence, despite the large inter-individual variability in C_{max} and AUC (57% and 54%, respectively). This is due to the expected randomness of the virtual trial. That randomness impacts type I and type II errors (and therefore power) of the analysis (see in Appendix, sections Parallel and Cross-over Trials Power Calculations (Workflow A) and Type I Error Analysis (Data-based Workflow A)). Power can be good for a data-based approach in the case of perfect BE (also with good prior information and study designs more sophisticated than just parallel) but it degrades rapidly if sizeable differences exist between formulations (even though they are bioequivalent). Type I error was very low and below the expected 2.5% at each side of the BE interval. This is a side-effect of the low power of parallel BE trials; close to the BE boundaries no trial will conclude to BE due to unmeasured inter-subject variability.

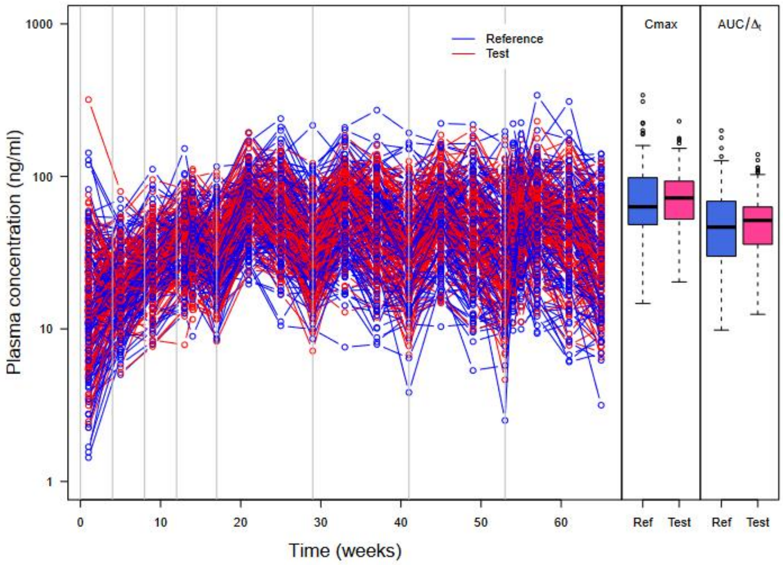

Figure 7: Simulated plasma PP concentrations, C_{max}, and AUC / Δ_t for 130 subjects per arm in a parallel virtual trial. The subjects received four injections of PP1M (150 mg eq.) prior to four injections (525 mg eq.) of PP3Mr (blue) or PP3Mt (red).

Fully Bayesian model-integrated workflow

Workflow B uses Monte Carlo simulations to approximate δ_{Cmax} and δ_{AUC} ratios posterior predictive distribution (see Figure 8). The decision about BE is immediate: the C_{max} ratio exceeds 1.25 with probability 0.354, and the AUC ratio exceeds it with probability 0.378, so BE should not be declared. The decision about BE differs from the one reached in the data-based workflow, because the latter relied on only one (albeit large) VBE trial, while here we “average” decision over 1000 trials.

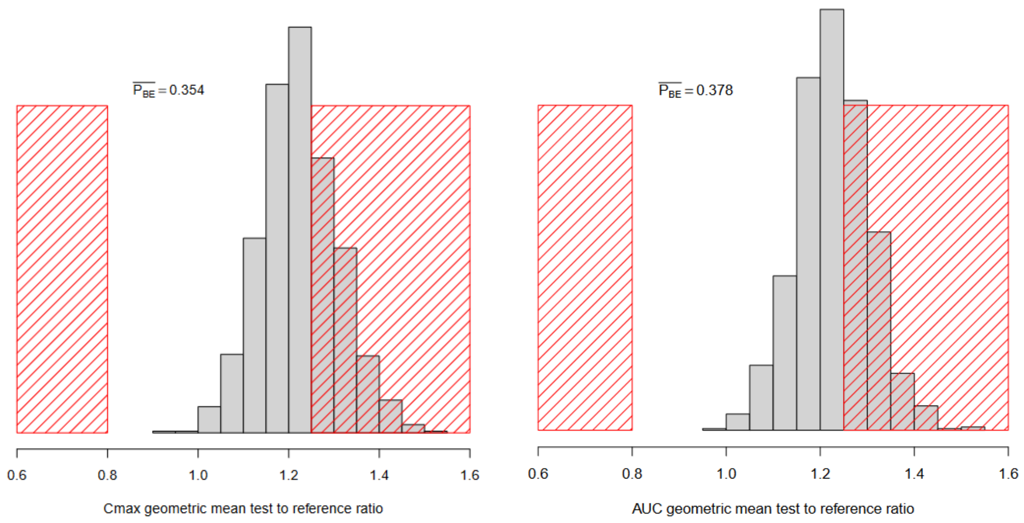

Figure 8: Histograms of the Bayesian marginal posterior predictive distributions of δ_{Cmax} and δ_{AUC} ratios. $\overline{P_{BE}}$ is the probability of non-bioequivalence. The red-shaded areas mark the standard non- bioequivalence regions (0.8 – 1.25).

Safe space Analysis

For workflow A, Figure 9 (left panel) shows the safe-space of model parameters f_3 and $k_{as3,max}$ (proxies for CQA's), as assessed by large virtual parallel trials. Parameters f_3 and $k_{as3,max}$ were the most influential parameters on safe-space definition and they interact, which is why we show the results in two dimensions (results for all six drug-release model parameters are given in Appendix, section Full safe-space calculations for the data-based parallel trial workflow). Given the low power of the TOST test near the BE limits, the safe-space region limits are fuzzy, and the "safest" space is quite reduced. The safe region is banded due to the structure of the PP kinetic model. The location of the full parallel trial we simulated for BE assessment is given by a blue cross. It falls in that region were decisions can be inconsistent. Safe-space calculations with a large cross-over trial lead to similar results, with a marked uncertainty, even if reduced compared to the above results (see Appendix, section Safe-space calculations for the data-based cross-over trial workflow).

The safe-space identified by workflow B is shown on Figure 9 right panel. Since power is no more a problem in that framework, the region is much better defined and about twice as large, but still coherent with the previous estimate (actually intermediate between the optimistic and the pessimistic estimates marked by green and red lines, respectively, in the left panel). A (δ_1, δ_2) pairs with value (1, 1.42) is a pass in this framework because on average it will not lead to exceedance of the BE limits for C_{max} nor AUC.

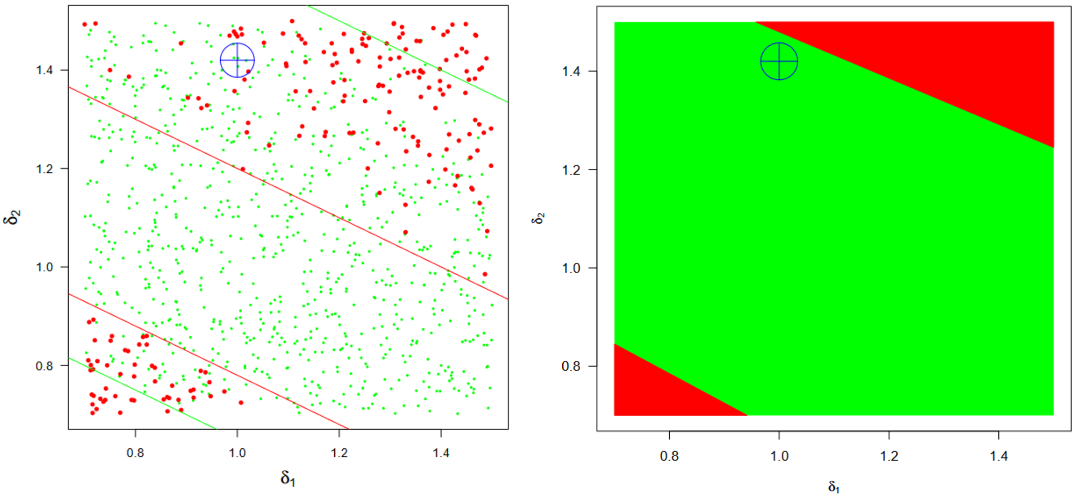

Figure 9: BE safe-space regions for the absorption parameters f_3 and $k_{as3,max}$ of the PP population PK model. Left: data-based estimate; the green dots indicate parallel PP trials (1000 trials, 500 subjects per arm) for which BE was declared using the TOST test; the red dots indicate failing trials; the red lines mark "sure" safe-space; the green lines mark the limits of surely non-BE space. The blue cross marks the location of the simulated full parallel trial we simulated for BE assessment. The intermediate areas stems from imperfect statistical power. Right: the fully Bayesian, model-based regions are much crisper.

Discussion and conclusion

We have presented two Bayesian virtual comparative clinical trial workflows. We demonstrated them with a realistic case study using an empirical population pharmacokinetic model of paliperidone palmitate LAI formulations. This work is not intended to be a VBE assessment for a particular product but a discussion of overarching issues in VBE.

PP long-acting injectable formulations are difficult to compare: between subject variability is high, and actual comparative trials seem unfeasible at reasonable costs and in reasonable time. However, the pharmacokinetics of the innovator formulation are well documented and a population PK model validated with clinical data on that formulation is available; the model describes the data well and was accepted by the US FDA [36]. Our implementation of it had to make a few approximations which should not affect our conclusions: we could not account for unreported covariate measurements; we had no information about subject-level parameters' covariance; we have uncertainty on the exact PP dose per subject and on previous treatments at start of the validation trials. That explains why our predicted population variability is a bit higher that the published one. Inter-occasion variability and modeling error are folded into residual error, but that should have minimal impact on our results because we simulated parallel clinical trials. Overall, predictions were within a factor 2 of the summary observations and the median estimates were within 25% of their observed counterparts reported in [31]. A refined model could assume that different formulations have different variabilities in release and absorption. They might be estimable from prior clinical data and abbreviated trial with a Bayesian population PBPK approach. An alternative would be to assume the possibility of different variances and assess its impact by sensitivity analysis.

We did not use *in vitro* evidence about test and reference differences, because this has already been demonstrated [2], and the model we used has no parameter measurable *in vitro*. Mechanistic PBPK models can better integrate prior information and data from *in vitro* experiments. However, we do not have such a model for PP LAI suspensions and a simpler model allows us to focus on the actual differences between workflows A and B.

Both workflows start with a definition of prior distributions and their Bayesian recalibration with available data, here from an abbreviated trial. In fact, the recalibration step is not needed, if the prior data already inform the model sufficiently to make it valid for prediction purposes. The data-based workflow A then assesses a simulated VBE trial with a frequentist test as if it were real. The particular abbreviated trial used impacts both workflows; the particular large trial simulated impacts only workflow A. Despites using an abbreviated trial which by chance over-estimates formulation differences, workflow A declares BE, but again by chance. Safe-space analysis shows that the simulated large trial falls in the area were BE decisions are quite random because of low power. The problem is posed of how to define “large” (and still “realistic”) for a virtual trial. Performing a standard statistical analysis of only *one* large trial is also problematic, because the decision hinges entirely on one trial realization. Changing deign of the large trial from parallel to cross-over increases the power of Workflow A, but it remains largely affected by uncertainty and is not optimal. Averaging over many very large trials could be done, but overall, assessing VBe on the basis of a large simulated trial just blurs the information already obtains up to the abbreviated trial stage. That is because a large simulated trial does not bring any new information and the subsequent statistical test just add unjustified randomness to the decision process.

The model-integrated workflow B is more coherent and bases decisions on expected future rather than on a particular virtual trial simulation. The posterior distribution of formulation differences is used to calculate posterior predictive distributions of PK measures of drug absorption. Those give us directly the probability that formulation differences will lead to unacceptable differences in drug absorption (see Figure 8). VBE assessment then simply estimates the probability that C_{max} or *AUC* differences exceed predefined limits. This is essentially equivalent to the current decision rule, with a probability estimated more accurately. Decision depends on the uncertainty we have on the size of the formulation differences, and that uncertainty is itself affected by the between-subject variability of measures of rate and extent of drug absorption (in particular in a parallel trial design). We used model-based estimates of C_{max} and *AUC* because it would not make sense to re-introduce measurement error in the process when it has already been accounted for during model calibration. Overall, workflow B controls consumer risk strictly while minimizing producer risk. The Bayesian decision rule also rewards data gathering in the first steps of the workflow. Note that the abbreviated trial could have had any design (the more informative, the better, so a cross-over design could be used, if possible). The design of the abbreviated trial should be closely examined, and the use of other metrics of rate and extent (*e.g.*, C_{min} and various forms of partial AUC) could be investigated [37,38].

Concerns about making the right decision with an acceptable error rate do not disappear in workflow B; However, standard statistical test performances (*e.g.*, type I and II error assessment) do not apply anymore because there is no need for a large trial and associated statistical testing. In our case study, if we declare bioequivalence and let the drug go to market, there is a 35% chance that we release a non-bioequivalent product; if we do not declare bioequivalence and block the product, there is a 65% chance that the product is in fact bioequivalent in the population. So, it is a judgement call, but if we adhere to the strict practice of controlling direct consumer risk at 5% we would reject bioequivalence, with a relatively high direct producer risk. A deeper problem is that a VBE framework, either data-based or model-integrated, has very little specific clinical evidence (only an abbreviated trial) available. However, it benefits from using a validated (*i.e.*, as good as possible) structural model, *in vitro* data and published prior information (which can be massive in the case of PBPK models). Therefore, model structure and correct parameterization are very important for both workflows, and model verification is of paramount importance in VBE. Modeling error can be introduced and could have more impact than in a BE assessment [1]. Standard BE trial analyses also make assumptions (like when using drug plasma concentrations for assessing the local bioequivalence of a drug targeting the gastro-intestinal-tract), but the issue is more glaring in VBE assessment. A further complication is that we simulate the abbreviated trial and the “ground truth” of our case study is laid bare for everyone to compare to the results of workflows A and B. Readers can immediately see the incoherences between “truth” and “decision”: the data-based workflow leads to a correct decision if we know the truth, but an incorrect decision given the information from the abbreviated trial; On the contrary, the model-integrated workflow decision is correct, given the abbreviated trial, but incorrect given the ground truth. In a “real life VBE assessment”, we would only have a model, its prior parameter distributions, *in vitro* data and data from one abbreviated clinical trial. Ground truth would not be accessible to us and workflow A would always be at the mercy of incoherent abbreviated trial and large trial simulations. However, we show that workflow B is more coherent and safer for everyone (producers *and* consumers). In a way, in a data-based VBE framework, type I and type II calculations on the virtual large trial can be a smoke screen giving a false sense of security, as if they were dispelling the only source of potential error, while they mask the real crux of VBE: having a correct model. So, we should not conduct VBE assessments like we do for BE assessments and should not judge a VBE assessment like a BE one.

Safe space analyses average over many simulations and are not affected by a specific trial simulation. However, they still differ between the two workflows and this may be viewed as our most important contribution. Safe-space calculations are more precise with workflow B because producer risk is minimized. Those calculations for workflow B took longer (12 hours on an 8-core laptop machine) than for workflow A. Preliminary calculations with workflow A could orient the search for precise safe-space boundaries in a fully Bayesian framework, as we have shown here.

Overall, we have shown that a Bayesian framework is well-suited for VBE assessment. We think that virtual comparative trials would benefit in general from the transparency and improved accuracy it provides. We need to gain more experience with it, in particular on real case studies with mechanistic, *e.g.* PBPK models.

References

1. Tsakalozou E, Babiskin A, Zhao L. Physiologically-based pharmacokinetic modeling to support bioequivalence and approval of generic products: a case for diclofenac sodium topical gel, 1%. CPT: Pharmacometrics and Systems Pharmacology. 2021;10:399-411 (PMC8129718).

2. Hsieh N-H, Bois FY, Tsakalozou E, Ni Z, Yoon M, Sun W, et al. A Bayesian population physiologically based pharmacokinetic absorption modeling approach to support generic drug development: application to bupropion hydrochloride oral dosage forms. Journal of Pharmacokinetics and Pharmacodynamics. 2021;48:893-908 (PMC8604781).

3. Sharan S, Fang L, Lukacova V, Chen X, Hooker AC, Karlsson MO. Model-informed drug development for long-acting injectable products: summary of American College of Clinical Pharmacology symposium. Clinical Pharmacology in Drug Development. 2021;10:220–8.

4. Gong Y, Zhang P, Yoon M, Zhu H, Kohojkar A, Hooker AC, et al. Establishing the suitability of model-integrated evidence to demonstrate bioequivalence for long-acting injectable and implantable drug products: summary of workshop. CPT: Pharmacometrics & Systems Pharmacology. 2023;12:624–30.

5. Tozer TN, Bois FY, Hauck WH, Chen M-L, Williams R. Absorption rate vs. exposure: which is more useful for bioequivalence testing. Pharmaceutical Research. 1996;13:453-456 (PMID8692741).

6. Zhang F, Jia R, Gao H, Wu X, Liu B, Wang H. In silico modeling and simulation to guide bioequivalence testing for oral drugs in a virtual population. Clinical Pharmacokinetics. 2021;60:1373–85.

7. Jamei M. Recent advances in development and application of physiologically-based pharmacokinetic (PBPK) models: a transition from academic curiosity to regulatory acceptance. Current Pharmacology Reports. 2016;2:161-169 (PMC4856711).

8. Loisios-Konstantinidis I, Hens B, Mitra A, Kim S, Chiann C, Cristofoletti R. Using physiologically based pharmacokinetic modeling to assess the risks of failing bioequivalence criteria: a tale of two ibuprofen products. The AAPS Journal. 2020;22:113 (PMID32830289).

9. Lin L, Wong H. Predicting oral drug absorption: mini review on physiologically-based pharmacokinetic models. Pharmaceutics. 2017;9:41.

10. Upton RN, Mould DR. Basic concepts in population modeling, simulation, and model-based drug development: part 3 - Introduction to pharmacodynamic modeling methods. CPT Pharmacometrics and Systems Pharmacology. 2014;3:e88.

11. Goutelle S, Woillard J, Buclin T, Bourguignon L, Yamada W, Csajka C, et al. Parametric and nonparametric methods in population pharmacokinetics: experts' discussion on use, strengths, and limitations. The Journal of Clinical Pharmacology. 2022;62:158–70.

12. Cristofoletti R, Patel N, Dressman JB. Assessment of bioequivalence of weak base formulations under various dosing conditions using physiologically based pharmacokinetic simulations in virtual populations. Case examples: ketoconazole and posaconazole. Journal of Pharmaceutical Sciences. 2017;106:560-569 (PMID27865610).

13. Dubois A, Gsteiger S, Pigeolet E, Mentré F. Bioequivalence tests based on individual estimates using non-compartmental or model-based analyses: evaluation of estimates of sample means and type I error for different designs. Pharmaceutical Research. 2010;27:92–104.

14. Loingeville F, Bertrand J, Nguyen TT, Sharan S, Feng K, Sun W, et al. New model–based bioequivalence statistical approaches for pharmacokinetic studies with sparse sampling. The AAPS Journal. 2020;22:141 (PMID33125589).

15. Möllenhoff K, Loingeville F, Bertrand J, Nguyen TT, Sharan S, Zhao L, et al. Efficient model-based bioequivalence testing. Biostatistics. 2022;23:314-327 (PMID32696053).

16. Cristofoletti R, Rowland M, Lesko LJ, Blume H, Rostami-Hodjegan A, Dressman JB. Past, present, and future of bioequivalence: improving assessment and extrapolation of therapeutic equivalence for oral drug products. Journal of Pharmaceutical Sciences. 2018;107:2519–30.

17. Bois FY, Hsieh N-H, Gao W, Chiu WA, Reisfeld B. Well-tempered MCMC simulations for population pharmacokinetic models. Journal of Pharmacokinetics and Pharmacodynamics. 2020;47:543–59.

18. Wedagedera JR, Afuape A, Chirumamilla SK, Momiji H, Leary R, Dunlavey M, et al. Population PBPK modeling using parametric and nonparametric methods of the Simcyp Simulator, and Bayesian samplers. CPT: Pharmacometrics and Systems Pharmacology. 2022;11:755-765 (PMC9197540).

19. Gelman A, Bois FY, Jiang J. Physiological pharmacokinetic analysis using population modeling and informative prior distributions. Journal of the American Statistical Association. 1996;91:1400–12.

20. Breslow N. Biostatistics and Bayes. Statistical Science [Internet]. 1990;5. Available from: https://projecteuclid.org/journals/statistical-science/volume-5/issue-3/Biostatistics-and-Bayes/10.1214/ss/1177012092.full

21. Fluehler H, Grieve AP, Mandallaz D, Mau J, Moser HA. Bayesian approach to bioequivalence assessment: an example. Journal of Pharmaceutical Sciences. 1982;72:1178–81.

22. Racine-Poon A, Grieve AP, Flühler H, Smith AFM. A two-stage procedure for bioequivalence studies. Biometrics. 1987;43:847–56.

23. Peck CC, Campbell G. Bayesian approach to establish bioequivalence: why and how? Clinical Pharmacology and Therapeutics. 2019;105:301–3.

24. Selwyn MR, Hall NR. On Bayesian methods for bioequivalence. Biometrics. 1984;40:1103.

25. Ghosh P, Rosner GL. A semi-parametric Bayesian approach to average bioequivalence. Statistics in Medicine. 2007;26:1224–36.

26. Ghosh P, Gönen M. Bayesian modeling of multivariate average bioequivalence. Statistics in Medicine. 2008;27:2402–19.

27. Schuirmann DJ. A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. Journal of Pharmacokinetics and Biopharmaceutics. 1987;15:657–80.

28. Ghosh P, Khattree R. Bayesian approach to average bioequivalence using Bayes’ factor. Journal of Biopharmaceutical Statistics. 2003;13:719–34.

29. Valsecchi P, Barlati S, Garozzo A, Deste G, Nibbio G, Turrina C, et al. Paliperidone palmitate in short- and long-term treatment of schizophrenia. Rivista di Psichiatria. 2019;54:235–48.

30. Samtani MN, Vermeulen A, Stuyckens K. Population pharmacokinetics of intramuscular paliperidone palmitate in patients with schizophrenia: a novel once-monthly, long-acting

formulation of an atypical antipsychotic. Clinical Pharmacokinetics. 2009;48:585-600 (PMID19725593).

31. Magnusson MO, Samtani MN, Plan EL, Jonsson EN, Rossenu S, Vermeulen A, et al. Population pharmacokinetics of a novel once-every 3 months intramuscular formulation of paliperidone palmitate in patients with schizophrenia. Clinical Pharmacokinetics. 2017;56:421-433 (PMID27743205).

32. Gelman A, Rubin DB. Inference from iterative simulation using multiple sequences (with discussion). Statistical Science. 1992;7:457–511.

33. R Development Core Team. R: A Language and Environment for Statistical Computing (R Foundation for Statistical Computing, Vienna, Austria, http://www.R-project.org) [Internet]. Vienna, Austria: R Foundation for Statistical Computing; 2013. Available from: http://www.R-project.org

34. de Valpine P, Turek D, Paciorek CJ, Anderson-Bergman C, Lang DT, Bodik R. Programming with models: writing statistical algorithms for general model structures with NIMBLE. Journal of Computational and Graphical Statistics. 2017;26:403–13.

35. NIMBLE Development Team. NIMBLE: MCMC, Particle Filtering, and Programmable Hierarchical Modeling, https://cran.r-project.org, https://r-nimble.org. doi: 10.5281/zenodo.1211190; 2022.

36. US FDA C for DE and R. Clinical Pharmacology and Biopharmaceutics Review - Application Number 207946orig1s000 [Internet]. US FDA; 2014 [cited 2023 Jan 21]. Available from: https://www.accessdata.fda.gov/drugsatfda_docs/nda/2015/207946Orig1s000ClinPharmr.pdf

37. Gajjar P, Dickinson J, Dickinson H, Ruston L, Mistry HB, Patterson C, et al. Determining bioequivalence possibilities of long acting injectables through population PK modelling. European Journal of Pharmaceutical Sciences. 2022;179:106296 (PMID36184958).

38. Lionberger RA, Raw AS, Kim SH, Zhang X, Yu LX. Use of partial AUCto demonstrate bioequivalence of zolpidem tartrate extended release formulations. Pharmaceutical Research. 2012;29:1110–20.

Acknowledgements

We thank Pauline Bogdanovich for helping with the coding and simulations of the cross-over design trial analyses. We also thank the Simcyp Library team for helping with literature access. All funding was internally provided by Certara Inc. Both authors are employees and potential shareholders of Certara.

Appendix

Contents

Paliperidone palmitate long-acting injectable PK models 24
Hierarchical population model 25
Prior model checking 26
Sensitivity analysis of impact of drug-release parameters on C_{max} and AUC 30
Abbreviated clinical trial simulation summary plot 31
Convergence of the model recalibration by MCMC sampling 32
Posterior distribution summary for parameter $\boldsymbol{\delta 2}$ 33
Observations *vs*. predictions plot for the recalibration step 33
Large virtual trial trial simulation summary plots 34
  Parallel trial case 34
  Cross-over trial case 34
Parallel and Cross-over Trials Power Calculations (Workflow A) 35
  Methods 35
  Results 36
Type I Error Analysis (Data-based Workflow A) 38
  Methods 38
  Results 39
Full safe-space calculations for the data-based parallel trial workflow 41
Safe-space calculations for the data-based cross-over trial workflow 42
Computer codes 42
  Structural model C code (v4) for parallel trials 42
  Structural model C code (v5) for cross-over trials 43
  Population PK model implementation in Nimble R (v8_pop) 45
  Statistical model in R (v13) for parallel trials 51
  Data-based R workflow for parallel trials (v4) 55
  Statistical model in R (v14) for cross-over trials 66
  Data-based R workflow for cross-over trials (v5) 70
  TOST code in R 77
  C_{max} and AUC data-based calculations code in R 79
  Structural model C code (v6) 79
  Statistical model in R (v16) 80
  Fully Bayesian workflow in R (v3) 84
  Fully Bayesian safe-space calculations in R 86
References 90

Paliperidone palmitate long-acting injectable PK models

The PP1M model published by Samtani *et al*. [1], also used by Magnusson *et al*. [2], is a two-compartment model with a depot and a central compartment. The structure of the PP3Mr model [2] is similar, but with two saturable release processes (described by Hills equations) from the depot compartment.

The two models were jointly used to model trials in which the starting dose is PP1M (for equilibration of the patients) followed by PP3Mr injections [2]. The equations are solved concurrently because PP1M depot may still release drug after the first PP3Mr injection. This is the approach taken by Magnusson *et al*. [2]. The model considers the fact that some subjects had already been treated with PP before entering the trial and had an unknown quantity, $Q_{central}(0)$, of PP in the central compartment. This quantity is therefore an additional model parameter. Note that this model assumes that all injections go to the same injection site, replenishing the previous depot.

For the PP1M model, after an intra-muscular injection of a $Dose_1$ of paliperidone palmitate in the depot compartment at the j-th injection time, t_{ij}, a fraction f_1 of $Dose_1$ is available for release from the depot through a zero-order process up to time t_{l1}, at which $f_1 \times Dose_1$ has been released. After t_{l1}, the remaining of $Dose_1$ is released through a first order process with rate constant $k_{a,1}$. The corresponding ordinary differential equations are:

$$\frac{\partial Q_{depot,1}}{\partial t} = -\frac{f_1 \times Dose_1}{t_{l1}}, \text{ with } Q_{depot,1}(t_{ij}) \mathrel{+}= f_1 \times Dose_1, \text{if } t < t_{ij} + t_{l1} \quad (2)$$

$$\frac{\partial Q_{depot,1}}{\partial t} = -(k_{a,1} \times Q_{depot,1}), \text{ with } Q_{depot,1}(t_{ij} + t_{l1}) \mathrel{+}= (1 - f_1) \times Dose_1, \text{if } t \geq t_{ij} + t_{l1} \quad (3)$$

$$\frac{\partial Q_{central}}{\partial t} = -\frac{\partial Q_{depot,1}}{\partial t} - CL \times \frac{Q_{central}}{V} \quad (4)$$

where $Q_{depot,1}$ and $Q_{central}$ are the amounts of drug in the depot and central compartments, CL is the drug clearance from the central compartment and V is the volume of that compartment.

The structure of the PP3Mr model [2] is similar, but with two saturable release processes (rapid and slow, described by Hills equations) from the depot compartment.

$$\frac{\partial Q_{depot,r,3}}{\partial t} = -\frac{k_{ar3,max} \times Q_{depot,r,3}}{k_{ar3,50} + Q_{depot,r,3}}, \text{ with } Q_{depot,r,3}(t_{ij}) \mathrel{+}= f_3 \times Dose_3 \quad (5)$$

$$\frac{\partial Q_{depot,s,3}}{\partial t} = -\frac{k_{as3,max} \times {Q_{depot,s,3}}^{\gamma}}{{k_{as3,50}}^{\gamma} + {Q_{depot,s,3}}^{\gamma}}, \text{ with } Q_{depot,s,3}(t_{ij}) \mathrel{+}= (1 - f_3) \times Dose_3 \quad (6)$$

$$\frac{\partial Q_{central}}{\partial t} = -\frac{\partial Q_{depot,r,3}}{\partial t} - \frac{\partial Q_{depot,s,3}}{\partial t} - CL \times \frac{Q_{central}}{V} \tag{7}$$

where $Q_{depot,r,3}$, $Q_{depot,s,3}$ and $Q_{central}$ are the respective amounts of drug in the rapid-release depot, slow-release depot, and central compartments; $k_{ar3,max}$, $k_{ar3,50}$, $k_{as3,max}$, $k_{as3,50}$, and γ are Hills drug-release and absorption parameters. $Dose_3$ is the dose of paliperidone palmitate at the j-th injection time; f_3 is the fraction of $Dose_3$ going to the fast release depot.

Hierarchical population model

The above structural model was developed, calibrated and checked in a population framework with large clinical datasets of the innovator's drug [2]. We use the same framework.

At the subject level, plasma concentration measurements were assumed to be lognormally distributed with a geometric mean given by the model-predicted subject-specific central compartment concentration profile and a variance σ^2 in log-space. Predicted plasma concentration values at times $t_{i,j}$ were obtained using the structural model, f, described above:

$$C_{i,j} \sim LN\left(f\left(\theta_i, t_{i,j}\right), \sigma^2\right) \tag{8}$$

For parameters $k_{a,1}$, $k_{as3,max}$, $k_{as3,50}$, $k_{ar3,50}$, CL, and V, subject-specific parameter values $\boldsymbol{\theta}_i$ were assumed to be lognormally distributed around population geometric means $\boldsymbol{\mu}$ with variances Σ^2 in log-space:

$$\theta_i \sim LN(\mu, \Sigma^2) \tag{9}$$

Parameters $k_{ar3,max}$ and γ were assumed to be the same for all subjects. In the analyses of Samtani *et al*. and Magnusson *et al*., a multivariate normal distribution was used, but they did not report the covariances' values. We assumed that they were negligible and use only the variances they provided. This does not seem to affect the ability of the model to reproduce the results of Magnusson *et al*. (see main text).

For parameters f_1 and f_3, a logit transformation was used and the corresponding logit, κ, was assumed to be lognormally distributed:

$$\kappa_i \sim LN\left(\frac{\mu}{1-\mu}, \Sigma^2\right) \tag{10}$$

$$\theta_i = \frac{1}{1+\exp{(-\kappa)}} \tag{11}$$

The initial quantity of PP in the central compartment, $Q_{central}(0)$, was not reported for the subjects of the Magnusson *et al.* trials. We assumed that subject-level values $Q_{central,i}(0)$ were lognormally distributed around a population geometric mean equal to 30 mg eq. of PP, with geometric SD 1.5. Those values were adjusted manually by us to match the starting PP plasma concentration levels shown in Figure 3 of the main text. They have a very small impact on the concentrations during the last dosing period, about one year later. We also have uncertainty on the exact dose of PP1M for each subject (some unreported dose adjustment was applied to the last three doses of PP1M to reach the therapeutic window for each subject), but we left that to be part of residual error (and it is unclear whether this reduced subject variability or not).

To model differences between the reference (PP3Mr) and test (PP3Mt) formulations, we introduced a vector of relative changes, δ, affecting the geometric means of the six drug-release and absorption parameters of the model, f_3 (the fraction of PP rapidly released), $k_{as3,max}$ (maximum release rate from the slow depot), $k_{ar3,max}$ (maximum release rate from the rapid depot), $k_{as3,50}$ (Hills coefficient for the slow-release depot), $k_{ar3,50}$ (Hills coefficient for the rapid depot), and γ (Hills power), in that order. Each mean (termed $\mu_{i,test}$ in the following equation) for the test formulation, given the reference formulation value $\mu_{i,ref}$ and the relative change δ_i, was computed as:

$$\mu_{i,test} = \delta_i \times \mu_{i,ref}, \text{with } i \in \{1, \dots, 6\} \qquad (12)$$

Those drug-release parameters should be related to product formulation CQA’s such as drug dissolution, injection medium composition, *etc*. Magnusson *et al.* [2] gave estimates for the parameters’ population geometric means and geometric variances (the latter transformed to coefficients of variation, CV, in natural space), together with precisions (as CVs) of those estimates. We used those numbers, appropriately transformed, to define prior distributions for the model’s parameters (for details, see Structural model C code (v4)). Magnusson *et al.* also introduced covariate measurements made on their subjects, but individual covariates values were not reported in the original model [2]. Therefore, their covariate model was not implemented here.

Prior model checking

Model checking consisted in comparing the simulations obtained with our model to the measured paliperidone concentrations of reported in the Magnusson *et al.* [2]. In the latter, individual data were not reported, but the median, 5th and 95th percentiles of the observations are given in their Figure 6. We digitized those plots with the Engauge Digitizer®. The model was run to ensure that the implementation of the code and of the dosing schedule were adequate.

The dose regimen simulated was as follows: all subjects received an injection of PP1M (dose range 50–150 mg eq.) every 4 weeks for 4 months; they were then switched to PP3M (dose range 175–525 mg eq.) with an injection every 12 weeks for one year. Because the subjects were exposed to paliperidone before the trial, we estimated visually the quantity in plasma at the start of the trial. For each subject, this quantity (in mg) was sampled from a lognormal distribution with geometric mean 30 mg and variance 1.5 in log scale.

The model predicts "true" or rather average plasma concentrations of paliperidone in virtual subjects. It can also simulate measured concentrations (simulated data) by adding a random error to the predicted average plasma concentration (Figure 10). The distribution of this random error is given by the published residual error model [2]; it accounts for measurement error, inter-occasion variability and modeling error. Figure 11 shows simulation results for a small cohort of 20 subjects. The code used for those plots is given below (Population PK model implementation in Nimble R (v8_pop)).

Plots of the predicted *vs*. observed (by Magnusson *et al.* [2]) median, fifth percentile and 95th percentile PP3Mr plasma concentrations at the various measurement times are shown in Figure 12. Practically all predictions are within a factor 2 away from the observation. Ratios averaged over time are given in Table 1. The average ratios for the median concentrations do not exceed 1.25.

Table 1: Average predicted-over-observed median, fifth percentile and 95th percentile PP3Mr plasma concentration ratios for the different trial doses presented in Magnusson *et al.* [2].

Dose	Median ratio	5 %tile ratio	95 %tile roadio
525	1.05	0.97	1.26
350	1.08	0.99	1.22
263	1.12	1.00	1.16
175	1.24	1.18	1.54

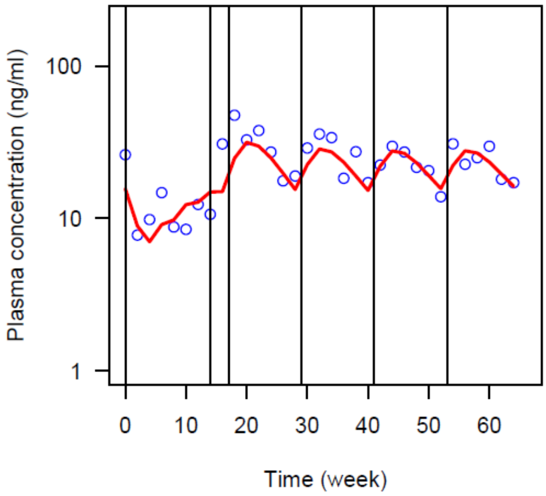

Figure 10: Simulation of plasma PP concentrations for a virtual subject with the joint PP1M (INVEGA) and PP3M (TRINZA) models [2]. Four injections of PP1M (75 mg eq.) are followed by four injections of PP3M (263 mg eq.). Parameter values were set to the best estimates given in the original publication [2]. Red line: average (predicted) plasma concentration of paliperidone; Blue points: simulated concentration measurements, with random error.

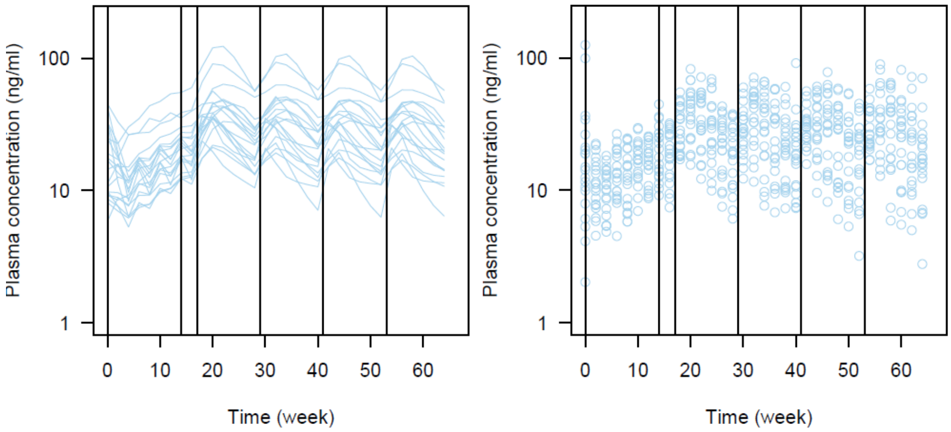

Figure 11: Simulation plasma PP concentrations for 20 virtual subjects with the PP1M (INVEGA) and PP3M (TRINZA) models [2]. Four injections of PP1M (75 mg eq.) are followed by four injections of PP3M (263 mg eq.). Parameter values were set to the best estimates given in the original publication [2]. Left panel: average (predicted) plasma concentrations of paliperidone; Right panel: simulated concentration measurements, with random error.

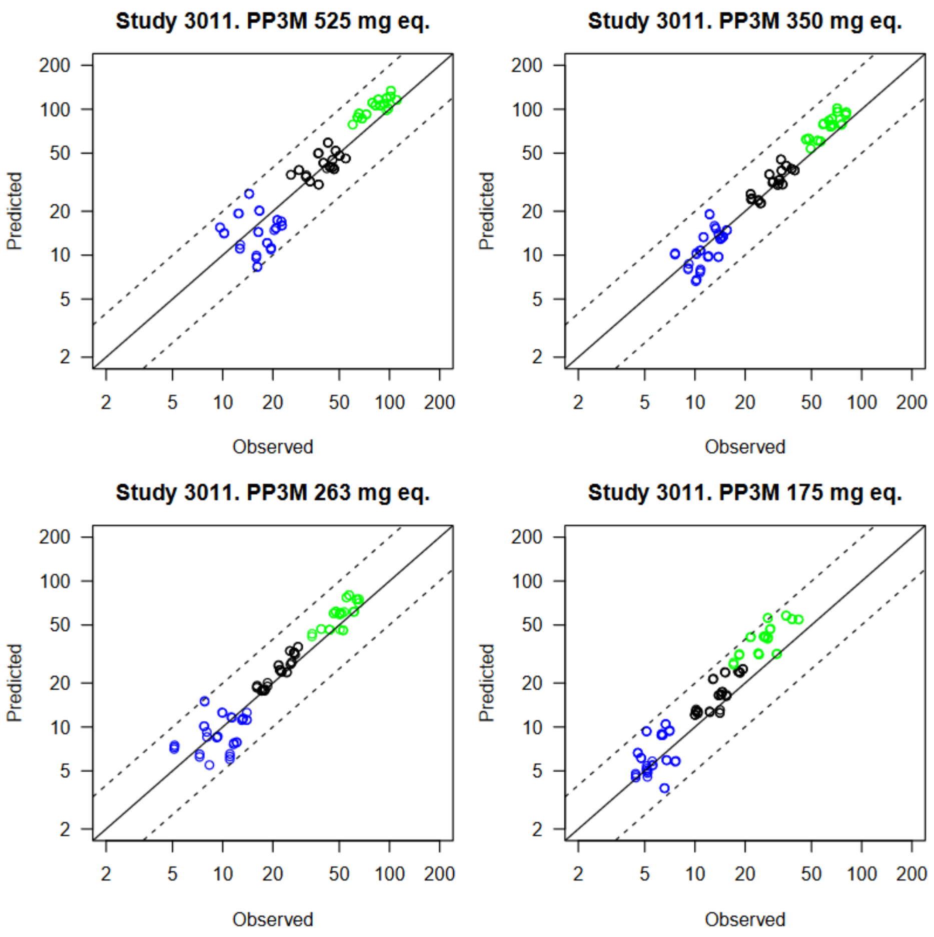

Figure 12: Predicted *vs*. observed (by Magnusson *et al.* [2]) medians (black), fifth percentiles (blue) and 95th percentiles (green) PP plasma concentrations at the various measurement times over the four PP3Mr (TRINZA) dosing periods. Parameter values were set to the best estimates given in the original publication.

We also implemented a version of the model able to simulate cross-over clinical trials with sequences PP1M, PP3Mr, PP3Mt or PP1M, PP3Mt, PP3Mr (see sections Structural model C code (v5) for cross-over trials and Statistical model in R (v14) for cross-over trials, below).

Figure 13 shows the simulation of a subject PP plasma levels in a PP1M, PP3Mr, PP3Mt sequence with this model.

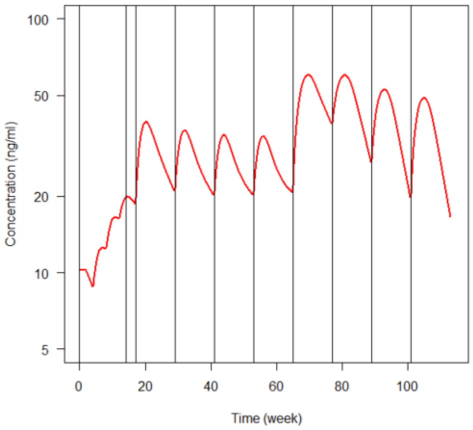

Figure 13: Simulation of plasma PP concentrations for a virtual subject with the cross-over trial model. Four injections of PP1M (75 mg eq.) are followed by four injections of PP3M reference formulation (TRINZA®) (263 mg eq.) and four injections of the same dose of a test PP3M formulation with f_3 (the fraction of PP rapidly released), and $k_{as3,max}$ twice as high as in the reference formulation. Red line: predicted plasma concentration of paliperidone for a random subject.

Sensitivity analysis of impact of drug-release parameters on C_{max} and AUC

We investigated the effect of changing the drug-release and absorption parameters of the PP3Mt model on the maximum PP plasma concentration (C_{max}) and the area under the curve (AUC) in the last PP3Mt dosing period. Those drug-release parameters of the model are $k_{ar3,max}$, $k_{ar3,50}$, $k_{as3,max}$, $k_{as3,50}$, γ, and f_3. The population mean of each drug-release parameter was changed one-at-a-time by ±5%, *i.e.*, by setting the relative change δ_i to 0.95 and 1.05. The same dosing regimen and sampling scheme as in the Magnusson et al.'s trial were applied. Only the highest doses for PP1M (150 mg eq.) and PP3Mt (525 mg eq.) were tested.

An analysis was run for one subject to quantify the impact on C_{max} and AUC of modifying by 5% the drug-release parameters. Figure 14 (left two panels) shows that in that case f_3, $k_{as3,max}$ and $k_{ar3,max}$ can impact C_{max} up to 2.2%, and that the AUC can be modified up to 3% by f_3 and $k_{as3,max}$. These influential parameters are positively correlated with the two PK parameters. C_{max} and AUC are most sensitive to $k_{as3,max}$ which controls PP release from the slow-release depot. For LAI products, absorption conditions the concentration decay phase (flip-flop) and we are using partial AUC at steady-state. So, the release rates logically condition trough concentrations, and therefore C_{max} and AUC. Figure 14, right panel, also shows how these small changes in the parameters' values affect PP plasma kinetic profile during the last dosing period (between 53 and 65 weeks, with blood sampling at weeks 54, 55, 57, 61 and 65).

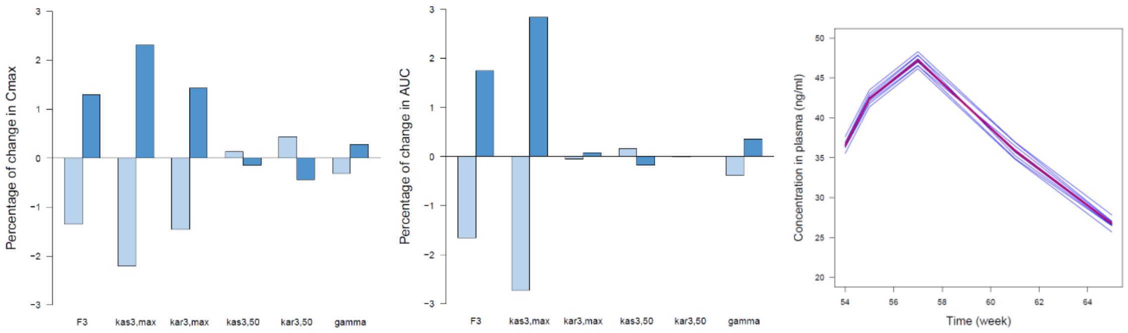

Figure 14: Percentages of change in C_{max} (left panel) and AUC (middle panel) when the model drug-release parameters are decreased (light blue) or increased (dark blue) by 5%. Right panel: predicted PP plasma concentration when the values of the drug-release parameters are set to the reference values (red curve) or modified by 5% (blue curves). Four injections of PP1M (150 mg eq.) were followed by four injections of either PP3Mr or PP3Mt (525 mg eq.). The plot is for the last dosing period.

Abbreviated clinical trial simulation summary plot

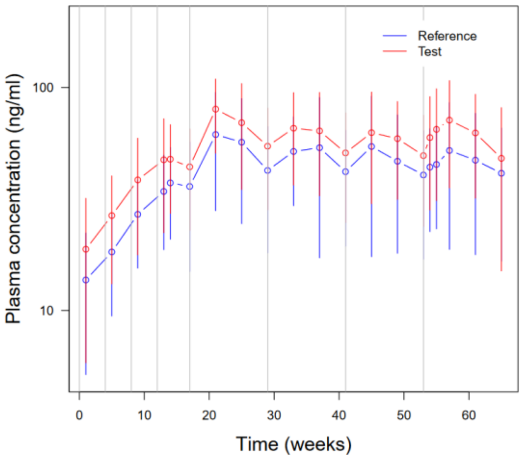

Figure 15: Simulated plasma PP concentrations averages of subjects (25 per arm) for the two arms of the simulated parallel abbreviated virtual trial when parameter $k_{as3,max}$ was increased by 5% from the value of the reference formulation. The subjects received four injections of PP1M (150 mg eq.) prior to four injections (525 mg eq.) of PP3Mr (blue) or PP3Mt (red). The vertical bars span ± 1 SD around the averages. The grey lines mark injection times.

Convergence of the model recalibration by MCMC sampling

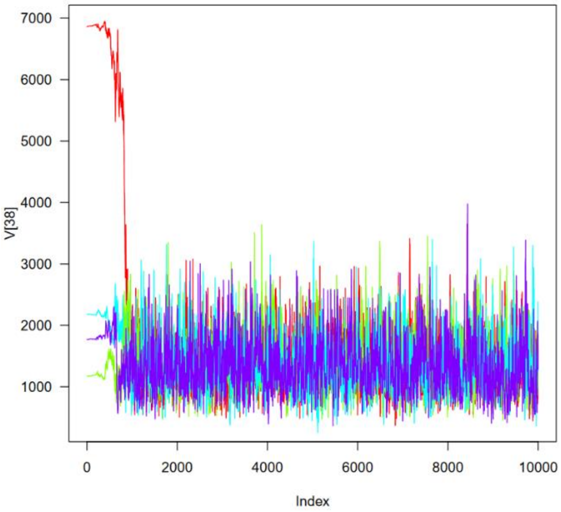

Figure 16: Illustration of MCMC sampling convergence. The values of parameter V (for virtual subject 38) sampled by four simulated Markov chains (different colors) are plotted against the number of iterations. The chains started from different random values and converged in probability to the target distribution only after about 1000 iterations. For all parameters, the first 2500 iterations were discarded to make sure that only values at convergence were kept.

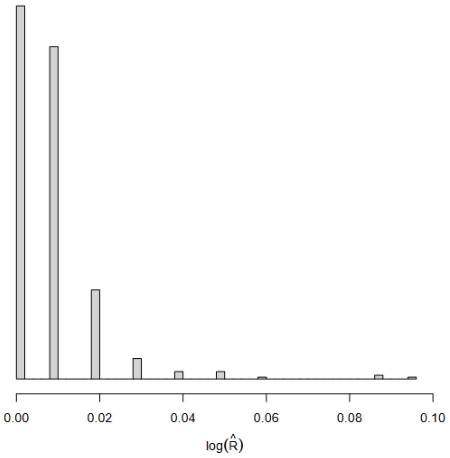

Figure 17: Histogram (over 451 model parameters) of the logarithms of the MCMC convergence diagnostic $\hat{R}$.

Posterior distribution summary for parameter $\boldsymbol{\delta_2}$

Table 2: Summary statistics and convergence diagnostic $\hat{R}$ for the marginal posteriors of $\log(\delta_2)$, actually sampled, and δ_2.

Parameter	Geometric Mean	Geometric SD	2.5 %tile	Median	97.5 %tile	$\hat{R}$
$\log(\delta_2)$	0.349	0.175	0.00	0.352	0.680	1.01
δ_2	1.42	1.19	1.00	1.42	1.97	1.01

Observations *vs*. predictions plot for the recalibration step

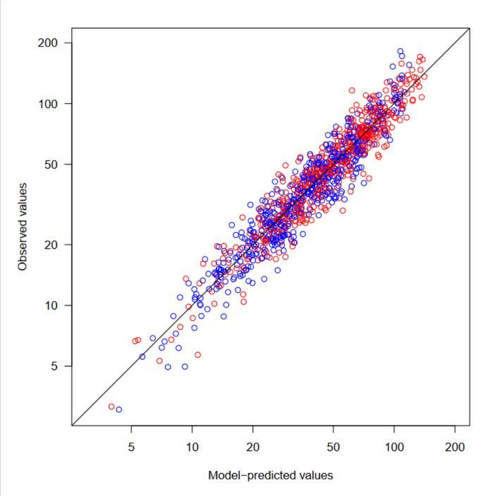

Figure 18: Observed PP plasma concentrations *vs*. corresponding posterior predictions with the maximum posterior (most likely) population PK parameter values. In blue: reference formulation group; in red: test group.

Large virtual trial trial simulation summary plots

Parallel trial case

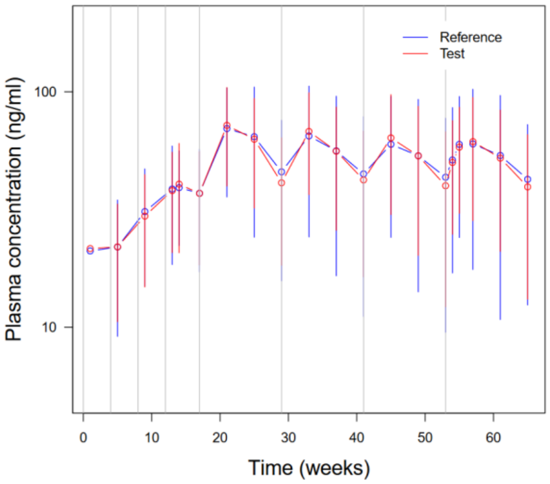

Figure 19: Simulated plasma PP concentrations averages of subjects (130 per arm) for the two arms of the simulated *parallel* virtual trial. The difference between $k_{as3,max}$ population means in test and reference formulations, δ_2, was set to 1.42 (geometric mean of its posterior distribution). By chance, this difference does *not* translate into large differences between test and reference PP plasma concentrations. The subjects received four injections of PP1M (150 mg eq.) prior to four injections (525 mg eq.) of PP3Mr (blue) or PP3Mt (red). The vertical bars span ± 1 SD around the averages. The grey lines mark injection times.

Cross-over trial case

A plot of the simulated large cross-over trial plasma concentration data with C_{max}, and AUC / Δ_t is shown on Figure 20. In this trial, C_{max} and AUC are increased on average in the test formulation, as the box plots on the Figure show. The difference between $k_{as3,max}$ population means in test and reference formulations, δ_2, was set to 1.42. A standard two-one-sided *t*-test (TOST test for cross-over trials) on either C_{max} or AUC concludes to bioequivalence, despite the large inter-individual variability in C_{max} and AUC.

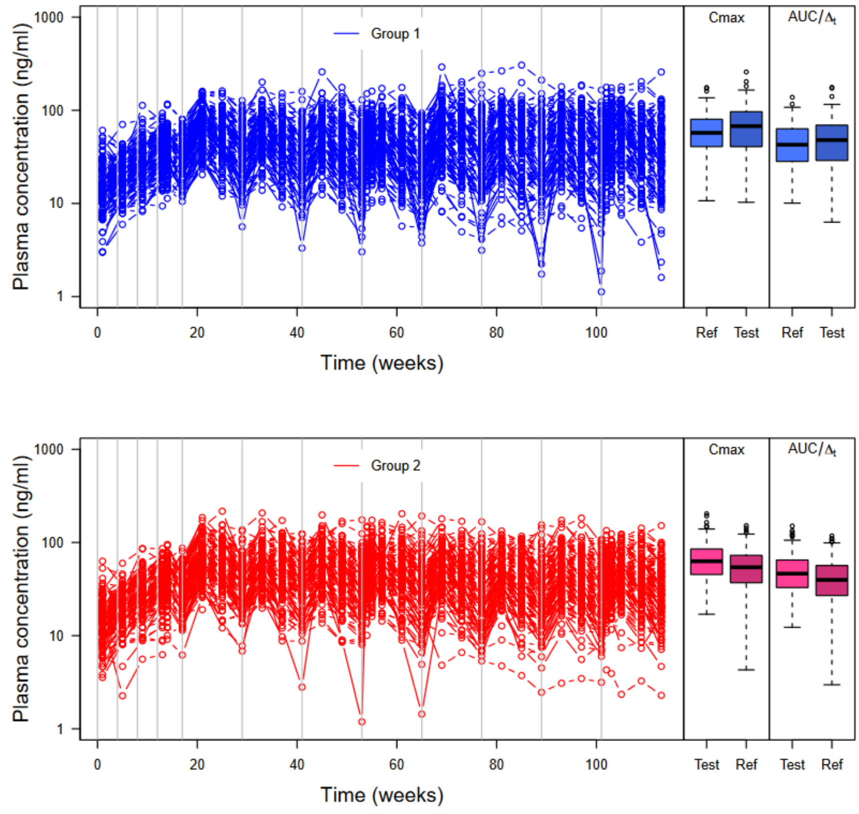

Figure 20: Simulated plasma PP concentrations averages of subjects (130 per arm) for the two arms of the simulated *cross-over* virtual trial. The difference between $k_{as3,max}$ population means in test and reference formulations, δ_2, was set to 1.42 (geometric mean of its posterior distribution). The subjects received four injections of PP1M (150 mg eq.) prior to four injections (525 mg eq.) of PP3Mr and then four injections PP3Mt (blue), or the reverse sequence (PP3Mt then PP3Mr, in red). The grey lines mark injection times.

Parallel and Cross-over Trials Power Calculations (Workflow A)

Methods

Parallel case

Power is the probability P of declaring BE when the test formulation is actually bioequivalent to the reference formulation. It is equal to 1 minus type II error, which measures producer risk (the probability of declaring non-BE when the products are in fact bioequivalent. In a simulation context, if all model parameters distributions for the test formulation model are exactly the same as for the reference formulation model (in the case of the population PK model we use, all components of the vector δ at value 1) then we are certain that test and reference are bioequivalent, and we can compute power.

If the test and reference formulations are not strictly identical (some components of $\delta \neq 1$), we can still estimate the probability P of declaring BE, but we should not call it "power", even though the formulations might still be bioequivalent in terms of C_{max} and AUC (if the structural parameters' relative differences δ are small enough).

In any case, the probability of declaring BE for a given value of δ can be calculated with the following pseudo-algorithm:

1. Simulate 1000 virtual trials of arm size N_{max} with δ set at the chosen value (we took N_{max} = 500 subjects per arm). Only N virtual subjects among N_{max} will be used to compute power at arm size N (lower or equal to N_{max}); For that:
2. Set arm size N to 2.
3. For each one of the 1000 virtual trials, draw randomly (without replacement) N subjects, assess and record BE for C_{max} and AUC using a TOST test; global BE is declared if BE is declared for both C_{max} and AUC.
4. Compute probability P at arm size N as the number of trials that declared BE over the total number of simulated trials (1000).
5. Increment arm size by 1.
6. Go to step 3 if arm size $\leq N_{max}$, otherwise go to step 7.
7. End; report probability P for each arm size.

See workflow code v4 below.

Cross-over case

Similar calculations were performed for the case of a large cross-over trial (up to 500 subjects). Here also one thousand (1000) trials were simulated. See code in section Data-based R workflow for cross-over trials (v5), below.

Results

Parallel case

The results of the above calculations are shown on Figure 21 for several values of δ_2 (the test over reference ratio of the drug-release parameter $k_{as3,max}$). All the other components of δ were set to 1. In the case of perfect bioequivalence (δ_2 = 1), 80% power or better is expected with 120 virtual subjects per arm or more. Power decreases when difference δ_2 increases; for example, with δ_2 = 1.2, more than 200 virtual subjects would probably be needed to reach 80% power.

Table 3 gives a numerical summary of Figure 21. With 130 subjects per arm, power (the probability of declaring bioequivalence when δ_2 = 1.0) is just slightly above 80%.

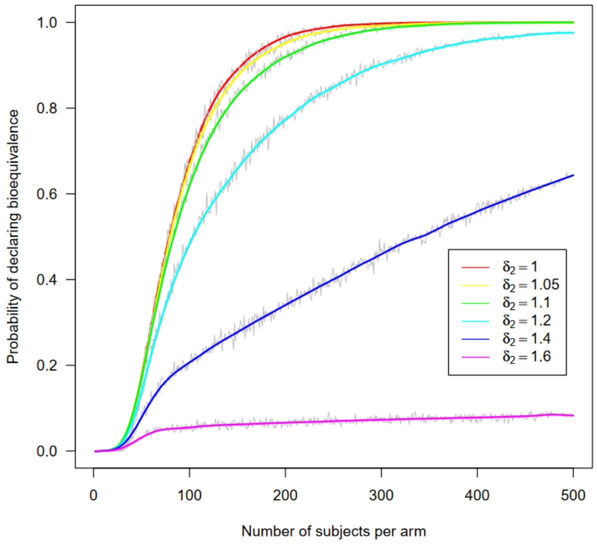

Figure 21: Estimated probability *P* of declaring bioequivalence as a function of the number of subjects per arm in a PP *parallel* virtual clinical BE trial. The PP population PK model was used to simulate 1000 virtual trials with different values of δ_2 (the test over reference ratio of the drug-release parameter $k_{as3,max}$). BE was assessed for *C*$_{max}$ and *AUC* with a TOST test.

Table 3: Probability of declaring bioequivalence as a function of the number of subjects per arm in a PP *parallel* virtual clinical BE trial and the underlying true value of δ_2. BE was assessed for C_{max} and *AUC* with a TOST test.

δ_2	Number of subjects per arm					
	100	130	200	300	400	500
1.00	0.68	**0.83**	0.97	1.00	1.00	1.00
1.05	0.66	0.81	0.95	0.99	1.00	1.00
1.10	0.62	0.77	0.92	0.99	1.00	1.00
1.20	0.48	0.60	0.77	0.90	0.96	0.98
1.40	0.20	0.25	0.34	0.46	0.56	0.64
1.60	0.06	0.06	0.06	0.07	0.08	0.08

Cross-over case

With the same number of subjects per arm (130), a cross-over trial, as expected, is consistently more powerful, as shown in Figure 22.

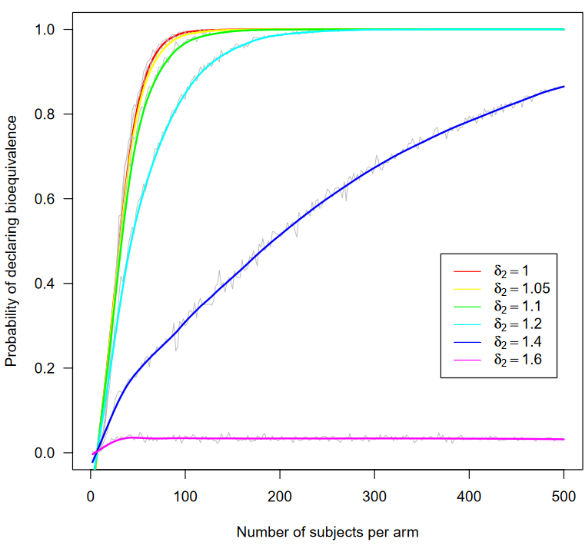

Figure 22: Estimated probability P of declaring bioequivalence as a function of the number of subjects per arm in a PP *cross-over* virtual clinical BE trial. The PP population PK model was used to simulate 1000 virtual trials with different values of δ_2 (the test over reference ratio of the drug-release parameter $k_{as3,max}$). BE was assessed for C_{max} and AUC with a TOST test.

Type I Error Analysis (Data-based Workflow A)

Methods

Type I error is the probability of declaring BE when the test formulation is actually *not* bioequivalent to the reference formulation. It measures a consumer risk. In a simulation context, it is necessary to generate trials with differences between test and reference. This can be done by sampling vector δ values in a large range. The trials for which the simulated data-based ratio of test over reference for C_{max} or AUC is outside the range [0.8, 1.25] can be considered as truly non-bioequivalent, if the trial size is large. We can then use the TOST to get its opinion on BE and check the fraction of trials for which TOST declares BE when the trial was in fact non-BE. This fraction is an estimate of type 1 error.

Type I error was therefore calculated with the following pseudo algorithm:

1. Simulate 1000 virtual trials of arm size N_{max} (we took N_{max} = 500 subjects per arm).

2. For each one of the 1000 virtual trials, sample each element of δ from a uniform [0.5,2] distribution. Compute geometric mean ratios test over reference for C_{max} and *AUC* for the simulated data. Assess and record BE for C_{max} and *AUC* using a TOST test; global BE is declared if BE is declared for both C_{max} and *AUC*.
3. Probability of declaring BE is the number of trials that declared BE over the total number of simulated trials (1000). Type I error is the probability of declaring BE when the C_{max} and *AUC* ratios are outside interval [0.8, 1.25].

Similar calculations were performed in the case of a 2-by-2 cross-over clinical trial (code in section Data-based R workflow for cross-over trials (v5), below).

Results

Figure 23 shows the probability of declaring BE with TOST as a function of the actual differences in C_{max} and *AUC* ratios for 1000 virtual trials. We can see that in no case BE was declared when the raw data showed non-compliant differences in C_{max} or *AUC*. Actually, outside the [0.9, 1.1] interval, approximately, BE was never declared. This is because TOST judges BE on the basis of confidence limits, and because there is also a producer risk (power is not always 100% with a size 1000 per arm trial). More precise type I error calculations would require computing the fraction of false positive TOST BE in different bins of difference values, but in any case, near the borders 0.8 and 1.25 those fractions would clearly be null. Overall, we can conclude that consumer risk is very small.

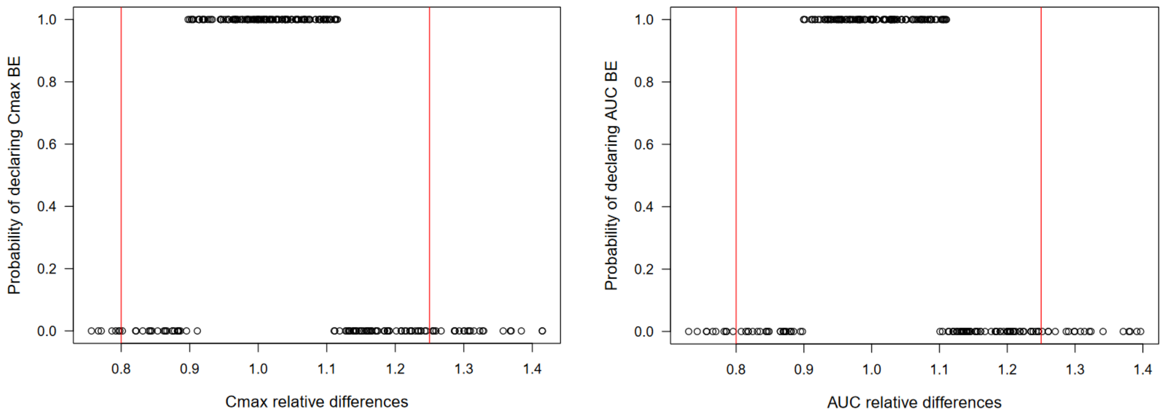

Figure 23: Estimated probability *P* of declaring bioequivalence as a function geometric of the data-based mean ratio test over reference for C_{max} (left panel) or *AUC* (right panel) in 1000 PP *parallel* virtual clinical BE trials with 500 subjects per arm. The PP population PK model was used to simulate each trial with different values of vector δ (the test over reference ratios for the population PK model absorption parameters). BE was assessed for C_{max} and AUC with a TOST test.

Figure 24 shows similar results for C_{max} in cross-over clinical trials. The increased power of cross-over trials compared to parallel trials translate in wider BE regions, but no trial declares bioequivalence beyond the 0.8 and 1.25 mean ratio limits.

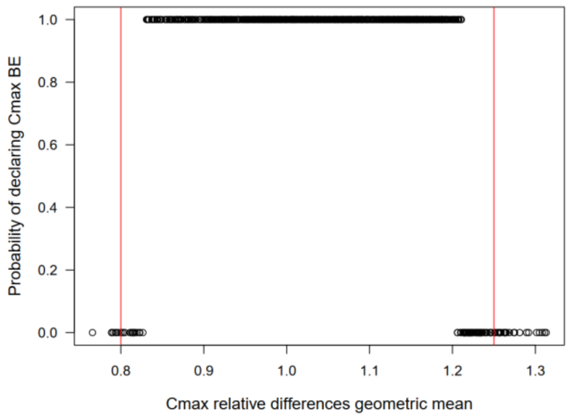

Figure 24: Estimated probability *P* of declaring bioequivalence as a function geometric of the data-based mean ratio test over reference for C_{max} in 1000 PP *cross-over* virtual clinical BE trials with 500 subjects per arm. The PP population PK model was used to simulate each trial with different values of vector δ (the test over reference ratios for the population PK model absorption parameters). BE was assessed for C_{max} with a TOST test.

Full safe-space calculations for the data-based parallel trial workflow

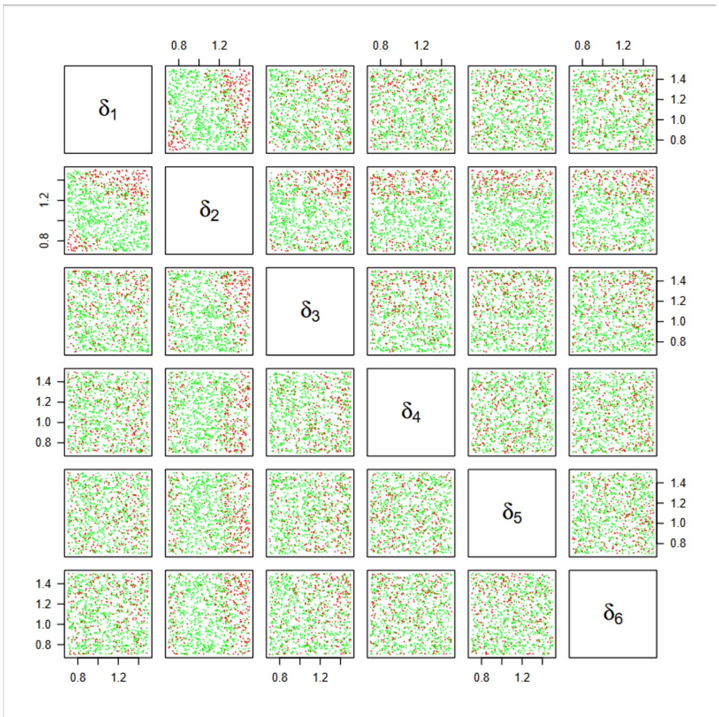

Figure 25: Data-based BE safe-space regions for the six drug-release parameters of the PP population PK model. The green dots mark the PP trials (1000 trials, 500 subjects per arm) for which BE was declared using the TOST test; the red dots indicate failing trials.

Safe-space calculations for the data-based cross-over trial workflow

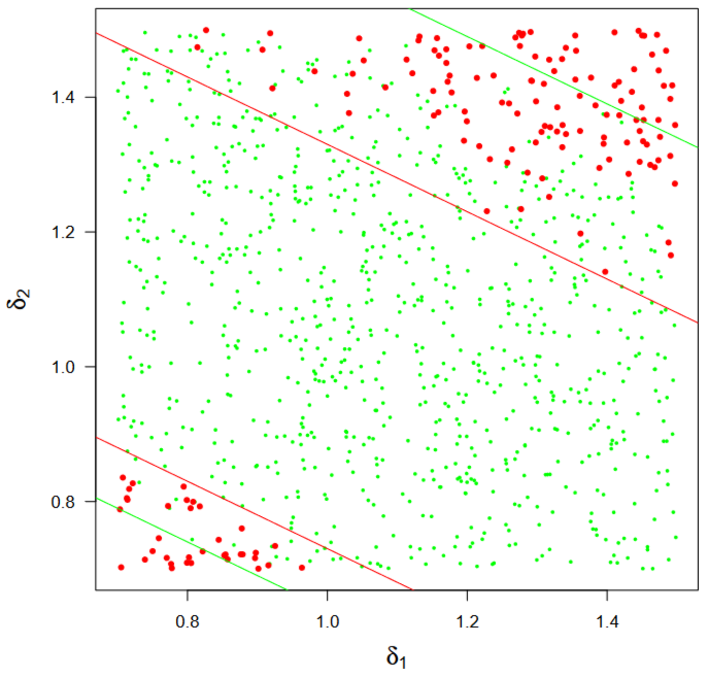

Figure 26: Data-based BE safe-space regions for the absorption parameters f_3 and $k_{as3,max}$ of the PP population PK model in a cross-over trial. Left: data-based estimate; the green dots indicate PP trials (1000 trials, 500 subjects per arm) for which BE was declared using the TOST test; the red dots indicate failing trials; the red lines mark “sure” safe-space; the green lines mark the limits of surely non-BE space. The blue cross marks the location of the simulated full parallel trial we simulated for BE assessment. The intermediate areas stems from imperfect statistical power.

Computer codes

Structural model C code (v4) for parallel trials

```
/* compile within R with system("R CMD SHLIB PP3M_model.c")
   V04
*/
#include <R.h>

#define Nparms 12

static double parms[Nparms];

/* A trick to keep up with the parameters */
#define Dose_PP1M     parms[0]
#define F2            parms[1]
#define Duration_2    parms[2]
#define ka_PP1M       parms[3]
#define ka1_max       parms[4]
```

```
#define ka3_max      parms[5]
#define kamt1_50     parms[6]
#define kamt3_50     parms[7]
#define gamma        parms[8]
#define CL           parms[9]
#define V            parms[10]
#define PP3M_start   parms[11]

/* initializer: same name as the dll (without extension) */
void PP13M_model_v04(void (* odeparms)(int *, double *))
{
    int N = Nparms;
    odeparms(&N, parms);
}

/* Derivatives */
void derivs(int *neq, double *t, double *y, double *ydot, double *yout, int*ip)
{
  double Ke;

  // State variables
  // Q_depot_s1 = y[0] # quantity (mg) in PP1M slow absorption depot
  // Q_depot_s3 = y[1] # quantity (mg) in PP3M slow absorption depot
  // Q_depot_r3 = y[2] # quantity (mg) in PP3M fast absorption depot
  // Q_central  = y[3] # quantity (mg) in central compartment

  // ODEs
  // Quantity in PP1M depot slow absorption
  ydot[0] = -ka_PP1M * y[0];

  // Quantities in PP3M depots
  if (*t < PP3M_start) { // use PP1M model, PP3M model differentials are null
    // Quantity in PP3M depot slow absorption
    ydot[1] = 0;
    // Quantity in PP3M depot rapid absorption
    ydot[2] = 0;
  }
  else { // use PP1M and PP3M models concurrently
    // Quantity in PP3M depot slow absorption
    ydot[1] = -ka1_max * pow(y[1], gamma) /
               (pow(kamt1_50, gamma) + pow(y[1], gamma));
    // Quantity in PP3M depot rapid absorption
    ydot[2] = -ka3_max * y[2] / (kamt3_50 + y[2]);
  }

  // Quantity in central compartment
  // clearance from central
  Ke = CL / V;
  // hard-code the zero-order inputs after PP1M injections
  if (((0 <= *t)    && (*t < 319))  || ((672 <= *t)  && (*t < 991)) ||
      ((1344 <= *t) && (*t < 1663)) || ((2016 <= *t) && (*t < 2335))) {
    ydot[3] = F2 * Dose_PP1M / Duration_2
              -(ydot[0] + ydot[1] + ydot[2]) - Ke * y[3];
  }
  else {
    ydot[3] = -(ydot[0] + ydot[1] + ydot[2]) - Ke * y[3];
  }
}
/* End */
```

Structural model C code (v5) for cross-over trials

```
/* compile within R with system("R CMD SHLIB PP3M_model.c")
   V05: three formulations dosing.
*/
#include <R.h>

#define Nparms 13
```

```
static double parms[Nparms];

/* Keep up with the parameters */
#define Dose_PP1M    parms[0]
#define F2           parms[1]
#define Duration_2   parms[2]
#define ka_PP1M      parms[3]
#define ka1_max      parms[4]
#define ka3_max      parms[5]
#define kamt1_50     parms[6]
#define kamt3_50     parms[7]
#define gamma        parms[8]
#define CL           parms[9]
#define V            parms[10]
#define PP3M_start   parms[11]
#define ka1_max_T    parms[12]

/* initializer: same name as the dll (without extension) */
void PP13M_model_v05(void (* odeparms)(int *, double *))
{
  int N = Nparms;
  odeparms(&N, parms);
}

/* Derivatives */
void derivs(int *neq, double *t, double *y, double *ydot, double *yout, int*ip)
{
  double Ke;

  // State variables
  // Q_depot_s1 = y[0] # quantity (mg) in PP1M slow absorption depot
  // Q_depot_s3 = y[1] # quantity (mg) in PP3M slow absorption depot
  // Q_depot_r3 = y[2] # quantity (mg) in PP3M fast absorption depot
  // Q_central  = y[3] # quantity (mg) in central compartment

  // ODEs
  // Quantity in PP1M depot slow absorption
  ydot[0] = -ka_PP1M * y[0];

  // Quantities in PP3M depots
  if (*t < PP3M_start) { // use PP1M model, PP3M model differentials are null
    // Quantity in PP3M depot slow absorption
    ydot[1] = 0;
    // Quantity in PP3M depot rapid absorption
    ydot[2] = 0;
  }
  else { // use PP1M and PP3M models concurrently
    // Quantity in PP3M depot slow absorption
    if (*t < 10920) {  // 10920 = 65*24*7
      ydot[1] = -ka1_max * pow(y[1], gamma) /
                (pow(kamt1_50, gamma) + pow(y[1], gamma));
    } else {
      ydot[1] = -ka1_max_T * pow(y[1], gamma) /
                (pow(kamt1_50, gamma) + pow(y[1], gamma));
    }
    // Quantity in PP3M depot rapid absorption
    ydot[2] = -ka3_max * y[2] / (kamt3_50 + y[2]);
  }

  // Quantity in central compartment
  // clearance from central
  Ke = CL / V;
  // hard-code the zero-order inputs after PP1M injections
  if (((0 <= *t)    && (*t < 319))  || ((672 <= *t)  && (*t < 991)) ||
      ((1344 <= *t) && (*t < 1663)) || ((2016 <= *t) && (*t < 2335))) {
    ydot[3] = F2 * Dose_PP1M / Duration_2
              -(ydot[0] + ydot[1] + ydot[2]) - Ke * y[3];
  }
  else {
    ydot[3] = -(ydot[0] + ydot[1] + ydot[2]) - Ke * y[3];
  }
}

/* End */
```

Population PK model implementation in Nimble R (v8_pop)

```
## v8_pop: Parameters of the population distributions are distributed.
##         Modify the PK model (v04).

library(nimble)
## compile C ODE model for deSolve
Cmodel.name = "PP13M_model_v04"
dyn.load(paste(Cmodel.name, .Platform$dynlib.ext, sep = ""))

## -------------------------------------------------------------------------
## ODE solver: performs a simulation, given initial state values,
## output times and time-constant, scaled, parameters
R_ode = function(y, times, parms) {
  ## params = c( 1:Dose_PP1M, 2:F2,      3:Duration_2,  4:ka_PP1M,
  ##             5:ka1_max,   6:ka3_max, 7:kamt1_50,    8:kamt3_50, 9:gamma,
  ##            10:CL,       11:V,      12:PP3M_start, 13:F3,      14:Dose_PP3M,
  ##            15:Q_cen_0)
  ## The last two parameters should not be passed to the ODE solver, they are
  ## used here only.
  ## State variables (y) initial conditions
  y = c("Q_depot_s1" = 0,
        "Q_depot_s3" = 0,
        "Q_depot_r3" = 0,
        "Q_central"  = parms[15])
  ## The doses are specified as "events" affecting the state variables
  ## dosing times (hours)
  dose_times = c(c(0, 4, 8, 12) + parms[3] / (24 * 7),
                 17, 29, 41, 53) * 24 * 7
  N_doses = length(dose_times)
  ## changing state variables at dosing times
  vars = c("Q_depot_s1", "Q_depot_s3", "Q_depot_r3")
  v1 = (1 -  parms[2])  * parms[1]     # value applied to Q_depot_s1
  v2 = (1 -  parms[13]) * parms[14]    # value applied to Q_depot_s3
  v3 =       parms[13]   * parms[14]    # value applied to Q_depot_r3
  ## Form the events table
  eventdat = data.frame(var = c(rep("Q_depot_s1", 4), rep(vars[2:3], 4)),
                        time = c(dose_times[1:4], rep(dose_times[5:8], each=2)),
                        value = c(rep(v1, 4),
                                  rep(c(v2, v3), N_doses - 4)),
                        method = "add")
  ## Integrate numerically, with outputs at specified times
  result = deSolve::ode(y, times, func="derivs", parms=parms[1:12],
                        rtol=1e-6, atol=1e-6, dllname="PP13M_model_v04",
                        initfunc = "PP13M_model_v04",
                        events=list(data=eventdat))
  result = result[which(result[,1] %in% times),] # weed out extra times
  if (dim(result)[[1]] < length(times)) { ## integration failed
    return(rep(1E-30, length(times)))
  } else {
    ## compute central concentration, convert from mg/L to ng/ml, return
    return(result[,5] * 1E3 / parms[11])
  }
} # end of R_ode model solver

## -------------------------------------------------------------------------
## Nimble function with nimbleRcall. This is just a wrapper
nimble_ode = nimbleRcall(
  prototype = function(
    y     = double(1), # vector
    times = double(1), # vector
    parms = double(1)  # vector
  ) {},
  returnType = double(1), # outcome is a vector
  Rfun = 'R_ode')

## ---------------------------------------------------------------------------
## Hierarchical core Nimble (BUGS) code
myNimbleCode = nimbleCode({ ## BUGS (extended) code
  ## population mean (with prior if not fixed)
  F2t_m       <- logit(0.168) # From Samtani paper Table III.
  # from online resource 3 Magnusson, F2 = 0.153
  F3t_m       <- logit(0.209) # transformed F3
  ka1_pp1m_mv <- log(1 + 0.02^2)
  ka1_pp1m_m  ~  dlnorm(meanlog=log(4.88E-4), varlog=ka1_pp1m_mv) # (1/h)
  ka1_max_m   ~  dnorm(mean = 0.0904,  sd = 0.0696 * 0.0904)  # (mg/h)
  ##ka1_max_m   <-  0.0904  # (mg/h)
  ka3_max_m   ~  dnorm(mean = 0.164,   sd = 0.0465 * 0.164)   # (mg/h)
  kamt1_50_m  ~  dnorm(mean = 120,     sd = 0.0383 * 120)     # (mg)
  kamt3_50_m  ~  dnorm(mean = 21.4,    sd = 0.0952 * 21.4)    # (mg)
  gamma_m     ~  dnorm(mean = 1.44,    sd = 0.0165 * 1.44)    # unitless
  CL_m        ~  dnorm(mean = 3.84,    sd = 0.0216 * 3.84)    # (1/hr)
  V_m         <- 1960                                         # (L)
  ##
  ## pop (inter-individual) SDs (with prior if not fixed)
  if (SamEq3) { # SD or CV applies to F2, so: detransform
    ## F2_sd
    omega2    <- 0.064 / (0.168 * (1 - 0.168)) # Samtani eq. 3
    F2_sd     ~  dnorm(mean = omega2, sd = 0.02*omega2)
    F2_v      <- F2_sd^2
    ## F3_v
    F3_v      <- (0.854 * 0.209 / (0.209 * (1 - 0.209)))^2 # Samtani eq. 3
  } else { # SD or CV applies to F2 transformed
    F2_sd     ~  dnorm(mean = 0.064, sd = 0.02*0.064)
    F3_v      <- (abs(F3t_m) * 0.854)^2
  }
  ##
  ka1_pp1m_cv ~  dnorm(mean = 0.590, sd = 0.03*0.59)
  ka1_pp1m_v  <- log(1 + ka1_pp1m_cv^2)
  ##
  ka1_max_cv  ~  dnorm(mean = 0.827, sd = 0.0501 * 0.827)
  ka1_max_v   <- log(1 + ka1_max_cv^2)
  ##
  ka3_max_v   <- 0
  ##
  kamt1_50_cv ~  dnorm(mean = 0.500, sd = 0.101  * 0.500)
  kamt1_50_v  <- log(1 + kamt1_50_cv^2)
  ##
  kamt3_50_cv ~  dnorm(mean = 0.867, sd = 0.142  * 0.867)
  kamt3_50_v  <- log(1 + kamt3_50_cv^2)
  ##
  gamma_v     <- 0
  ##
  CL_cv_v     <- log(1 + 0.0317^2)
  CL_cv       ~  dlnorm(meanlog=log(0.357), varlog=CL_cv_v)
  CL_v        <- log(1 + CL_cv^2)
  ##
  V_v         <- log(1 + 0.628^2)
  ##
  ## measurement error variance in log for plasma concentration, ng/ml
  ## if we want uncertainty on the residual error we should use:
  res_cv_v    <- log(1 + 0.321^2)
  res_cv      ~  dlnorm(meanlog=log(0.306), varlog=res_cv_v)
  sigma2      <- log(1 + res_cv^2)
  ##
  ## for each subject
  for (i in 1:nsubjects) {
    tmp2[i]      ~  dnorm(mean = F2t_m, var = F2_v)
    F2[i]       <-  ilogit(tmp2[i])
    tmp3[i]      ~  dnorm(mean = F3t_m, var = F3_v)
    F3[i]       <-  ilogit(tmp3[i])
    ka_PP1M[i]  ~  dlnorm(meanlog=log(ka1_pp1m_m), varlog=ka1_pp1m_v) # (1/hr)
    ka1_max[i]  ~  dlnorm(meanlog=log(ka1_max_m),  varlog=ka1_max_v)  # (mg/h)
    ka3_max[i]  ~  dlnorm(meanlog=log(ka3_max_m),  varlog=ka3_max_v)  # (mg/h)
    kamt1_50[i] ~  dlnorm(meanlog=log(kamt1_50_m), varlog=kamt1_50_v) # (mg)
    kamt3_50[i] ~  dlnorm(meanlog=log(kamt3_50_m), varlog=kamt3_50_v) # (mg)
    gamma[i]    ~  dlnorm(meanlog=log(gamma_m),    varlog=gamma_v)    # no unit
    CL[i]       ~  dlnorm(meanlog=log(CL_m),       varlog=CL_v)       # (1/hr)
    V[i]        ~  dlnorm(meanlog=log(V_m),        varlog=V_v)        # (L)
    ##
    Q_cen_0[i]  ~  dlnorm(meanlog=log(Q_cen_0_mean), sdlog = log(Q_cen_0_sd))
    ##
    ## likelihood for concentration measurements
    ## Call the ODE solver to get predictions for each subject
    ## predictions are plasma concentrations, in ng/ml
    Conc[i, 1:ntimes] <- nimble_ode(y[1:nstates], times[1:ntimes],
                                    c(Dose_PP1M, F2[i], Duration_2,
                                      ka_PP1M[i], ka1_max[i], ka3_max[i],
                                      kamt1_50[i], kamt3_50[i], gamma[i],
                                      CL[i], V[i], PP3M_start, F3[i],
                                      Dose_PP3M, Q_cen_0[i]))
    ## data likelihood
    for (j in 1:ntimes) {
      C_plasma_obs[i,j] ~ dlnorm(meanlog=log(Conc[i,j]), varlog=sigma2)
    }
  }
}) # End myNimbleCode

## ------------------------------------------------------------------------
## Build and compile the model for predictions with various PP1M doses
## show the difference between data and mean
N = 1 # number of trials
nsubjects = 1 # number of subjects

times = seq(0, 65, 2) * 7 * 24 # up to 65 weeks, in hours

dose_pp1m = c(150, 100, 75,  50)
dose_pp3m = c(525, 350, 263, 175)
Q_cen_0_mean = 30 # geo mean
Q_cen_0_sd   = 1.5  # geo SD
data  = list()
inits = list(Q_cen_0_mean = Q_cen_0_mean, # (mg)
             Q_cen_0_sd   = Q_cen_0_sd,   # (mg)
             Dose_PP1M    = dose_pp1m[1], # (mg)
             Dose_PP3M    = dose_pp3m[1]) # (mg)

constants = list(nsubjects  = nsubjects,
                 SamEq3     = 1,             # Boolean
                 Duration_2 = 319,           # (h)
                 PP3M_start = 17*7*24,       # (h)
                 ntimes     = length(times),
                 times      = times,
                 nstates    = 4)

Rmodel = nimbleModel(myNimbleCode, constants, data, inits)
Cmodel = compileNimble(Rmodel, showCompilerOutput=F)

Node.names = Cmodel$getNodeNames(includeData=T)
Cmodel$simulate(nodes = Node.names)
res = values(Cmodel, Node.names)

par(mfrow=c(2,2), mar=c(4,4,3,1))
plot.times = times/(24*7)
for (j in 1:length(dose_pp3m)) { # for each dose of PP3M
  Cmodel$Q_cen_0_mean = Q_cen_0_mean
  Cmodel$Q_cen_0_sd   = Q_cen_0_sd
  Cmodel$Dose_PP1M = dose_pp1m[j]
  Cmodel$Dose_PP3M = dose_pp3m[j]

  Cmodel$simulate(nodes=Node.names)
  sub.mean_vector = values(Cmodel, "Conc")
  sub.data_vector = values(Cmodel, "C_plasma_obs")
  sub.mean = matrix(sub.mean_vector, nrow=nsubjects, byrow = F)
  sub.data = matrix(sub.data_vector, nrow=nsubjects, byrow = F)

  ## plot individual means
  plot (plot.times, times, type="n", col="red", lwd=2, log="y",
        las=1, xlab="Time (week)", ylab="Plasma concentration (ng/ml)",
        main=paste("Study 3011. PP3M", dose_pp3m[j] ,"mg eq."),
        xlim=c(0, 66), ylim=c(1, 200),
        yaxp=c(1, 100, 1)) # ylim should be adapted!!!!

  for (i in 1:nsubjects) {
    lines(plot.times, sub.data[i,], type ="p", col="blue",lwd=1)
    lines(plot.times, sub.mean[i,], type ="l", col="red", lwd=2)
  }
  abline(v=c(0, 14, 17, 29, 41, 53))
}

## ------------------------------------------------------------------------
## Build and compile the model for predictions with various PP1M doses
## small clinical study
N = 1 # number of trials
nsubjects = 20 # number of subjects

times = seq(0, 65, 2) * 7 * 24 # up to 65 weeks, in hours

dose_pp1m = c(150, 100, 75,  50)
dose_pp3m = c(525, 350, 263, 175)
Q_cen_0_mean = 30 # geo mean
Q_cen_0_sd   = 1.5  # geo SD
data  = list()
inits = list(Q_cen_0_mean = Q_cen_0_mean, # (mg)
             Q_cen_0_sd   = Q_cen_0_sd,   # (mg)
             Dose_PP1M    = dose_pp1m[1], # (mg)
             Dose_PP3M    = dose_pp3m[1]) # (mg)

constants = list(nsubjects  = nsubjects,
                 SamEq3     = 1,            # Boolean
                 Duration_2 = 319,          # (h)
                 PP3M_start = 17*7*24,      # (h)
                 ntimes     = length(times),
                 times      = times,
                 nstates    = 4)

Rmodel = nimbleModel(myNimbleCode, constants, data, inits)
Cmodel = compileNimble(Rmodel, showCompilerOutput=F)

Node.names = Cmodel$getNodeNames(includeData=T)
Cmodel$simulate(nodes = Node.names)
res = values(Cmodel, Node.names)

par(mfrow=c(2,2), mar=c(4,4,3,1))
plot.times = times/(24*7)
for (j in 1:length(dose_pp3m)) { # for each dose of PP3M
  Cmodel$Q_cen_0_mean = Q_cen_0_mean
  Cmodel$Q_cen_0_sd   = Q_cen_0_sd
  Cmodel$Dose_PP1M = dose_pp1m[j]
  Cmodel$Dose_PP3M = dose_pp3m[j]

  Cmodel$simulate(nodes=Node.names)
  sub.mean_vector = values(Cmodel, "Conc")
  sub.data_vector = values(Cmodel, "C_plasma_obs")
  sub.mean = matrix(sub.mean_vector, nrow=nsubjects, byrow = F)
  sub.data = matrix(sub.data_vector, nrow=nsubjects, byrow = F)

  plot (plot.times, times, type="n", col="red", lwd=2, log="y",
        las=1, xlab="Time (week)", ylab="Plasma concentration (ng/ml)",
        main=paste("Study 3011. PP3M", dose_pp3m[j] ,"mg eq."),
        xlim=c(0, 66), ylim=c(1, 200),
        yaxp=c(1, 100, 1)) # ylim should be adapted!!!!

  for (i in 1:nsubjects) {
    # plot individual mean
    #lines(plot.times, sub.mean[i,], type ="l", col="lightskyblue2",lwd=0.5)
    ## plot individual data
    lines(plot.times, sub.data[i,], type ="p", col="lightskyblue2",lwd=0.5)
  }
  abline(v=c(0, 14, 17, 29, 41, 53))
}

## ---------------------------------------------------------------------------
## Build and compile the model for predictions with various PP1M doses
## N =100 and nsbjects =130

nsubjects = 130 # number of subjects

## times = seq(0, 65, 1) * 7 * 24 # up to 65 weeks, in hours
times = c(0, seq(0, 65*7*24, 7*6)+12) ## (hours)
dose_pp1m = c(150, 100, 75,  50)
dose_pp3m = c(525, 350, 263, 175)
Q_cen_0_mean = 30 # geo mean
Q_cen_0_sd   = 1.5  # geo SD
data  = list()
inits = list(Q_cen_0_mean = Q_cen_0_mean, # (mg)
             Q_cen_0_sd   = Q_cen_0_sd,   # (mg)
             Dose_PP1M    = dose_pp1m[1], # (mg)
             Dose_PP3M    = dose_pp3m[1]) # (mg)

constants = list(nsubjects  = nsubjects,
                 SamEq3     = 1,            # Boolean
                 Duration_2 = 319,          # (h)
                 PP3M_start = 17*7*24,      # (h)
                 ntimes     = length(times),
                 times      = times,
                 nstates    = 4)

Rmodel = nimbleModel(myNimbleCode, constants, data, inits)
Cmodel = compileNimble(Rmodel, showCompilerOutput=F)

Node.names = Cmodel$getNodeNames(includeData=T)
Cmodel$simulate(nodes = Node.names)
res = values(Cmodel, Node.names)

Magnus1 = read.csv("Magnusson_Figure6_Panel1.csv")
Magnus2 = read.csv("Magnusson_Figure6_Panel2.csv")
Magnus3 = read.csv("Magnusson_Figure6_Panel3.csv")
Magnus4 = read.csv("Magnusson_Figure6_Panel4.csv")

N = 100 # number of trials

par(mfrow=c(2,2), mar=c(4,4,3,1))
plot.times = times/(24*7)
for (j in 1:length(dose_pp3m)) { # for each dose of PP3M
  Cmodel$Q_cen_0_mean = Q_cen_0_mean
  Cmodel$Q_cen_0_sd   = Q_cen_0_sd
  Cmodel$Dose_PP1M = dose_pp1m[j]
  Cmodel$Dose_PP3M = dose_pp3m[j]
  ## Monte Carlo simulations for mean and data predictions for random subjects
  quant.mean.p5.all  = matrix(0, nrow=N,
                              ncol=length(values(Cmodel, "Conc"))/nsubjects)
```

```
quant.mean.p50.all = matrix(0, nrow=N,
                            ncol=length(values(Cmodel, "Conc"))/nsubjects)
quant.mean.p95.all = matrix(0, nrow=N,
                            ncol=length(values(Cmodel, "Conc"))/nsubjects)
quant.data.p5.all  = matrix(0, nrow=N,
                            ncol=length(values(Cmodel,
                                               "C_plasma_obs"))/nsubjects)
quant.data.p50.all = matrix(0, nrow=N,
                            ncol=length(values(Cmodel,
                                               "C_plasma_obs"))/nsubjects)
quant.data.p95.all = matrix(0, nrow=N,
                            ncol=length(values(Cmodel,
                                               "C_plasma_obs"))/nsubjects)

for (i in 1:N) { # for each trial

  Cmodel$simulate(nodes=Node.names)
  sub.mean_vector = values(Cmodel, "Conc")
  sub.data_vector = values(Cmodel, "C_plasma_obs")

  sub.mean = matrix(sub.mean_vector, nrow=nsubjects, byrow = F)
  sub.data = matrix(sub.data_vector, nrow=nsubjects, byrow = F)

  quant.mean = apply(sub.mean, MAR=2, FUN=quantile, probs=c(0.05, 0.5, 0.95))
  quant.data = apply(sub.data, MAR=2, FUN=quantile, probs=c(0.05, 0.5, 0.95))

  quant.mean.p5.all[i,]  = quant.mean[1,]
  quant.mean.p50.all[i,] = quant.mean[2,]
  quant.mean.p95.all[i,] = quant.mean[3,]

  quant.data.p5.all[i,]  = quant.data[1,]
  quant.data.p50.all[i,] = quant.data[2,]
  quant.data.p95.all[i,] = quant.data[3,]
}

## plot individual means
plot (plot.times, times, type="n", col="red", lwd=2, log="y",
      las=1, xlab="Time (week)", ylab="Plasma concentration (ng/ml)",
      main=paste("Study 3011. PP3M", dose_pp3m[j] ,"mg eq."),
      xlim=c(0, 66), ylim=c(1, 200),
      yaxp=c(1, 100, 1)) # ylim should be adapted!!!!

quant.p5 = apply(quant.data.p5.all, MAR=2, FUN=quantile, probs=c(0.05, 0.95))
lines(plot.times,  quant.p5[1,],  col="lightskyblue2")
lines(plot.times,  quant.p5[2,],  col="lightskyblue2")
polygon(c(plot.times,rev(plot.times)),
        c(quant.p5[2,],rev(quant.p5[1,])),
        col="lightskyblue2", border = "lightskyblue2")

quant.p50 = apply(quant.data.p50.all, MAR=2, FUN=quantile,
                  probs=c(0.05, 0.95))
lines(plot.times,  quant.p50[1,],  col="lightskyblue2")
lines(plot.times,  quant.p50[2,],  col="lightskyblue2")
polygon(c(plot.times,rev(plot.times)),
        c(quant.p50[2,],rev(quant.p50[1,])),
        col="lightskyblue2", border = "lightskyblue2")

quant.p95 = apply(quant.data.p95.all, MAR=2, FUN=quantile,
                  probs=c(0.05, 0.95))
lines(plot.times,  quant.p95[1,],  col="lightskyblue2")
lines(plot.times,  quant.p95[2,],  col="lightskyblue2")
polygon(c(plot.times,rev(plot.times)),
        c(quant.p95[2,],rev(quant.p95[1,])),
        col="lightskyblue2", border = "lightskyblue2")

if (j == 1) {
  lines(Magnus1$Time, Magnus1$P50, col="red",  lwd=2)
  lines(Magnus1$Time, Magnus1$P5,  col="red",  lty=2)
```

```
    lines(Magnus1$Time, Magnus1$P95, col="red",  lty=2)
  }
  if (j == 2) {
    lines(Magnus2$Time, Magnus2$P50, col="red",  lwd=2)
    lines(Magnus2$Time, Magnus2$P5,  col="red",  lty=2)
    lines(Magnus2$Time, Magnus2$P95, col="red",  lty=2)
  }
  if (j == 3) {
    lines(Magnus3$Time, Magnus3$P50, col="red",  lwd=2)
    lines(Magnus3$Time, Magnus3$P5,  col="red",  lty=2)
    lines(Magnus3$Time, Magnus3$P95, col="red",  lty=2)
  }
  if (j == 4) {
    lines(Magnus4$Time, Magnus4$P50, col="red",  lwd=2)
    lines(Magnus4$Time, Magnus4$P5,  col="red",  lty=2)
    lines(Magnus4$Time, Magnus4$P95, col="red",  lty=2)
  }
  abline(v=c(0, 14, 17, 29, 41, 53))
}

## End.
```

Statistical model in R (v13) for parallel trials

```
## R/Nimble code for PP1M/PP3M population model, from Magnusson
## v13

library(nimble)

## compile C ODE model for deSolve
Cmodel.name = "PP13M_model_v04"
## system(paste0("R CMD SHLIB ", Cmodel.name, ".c"))
dyn.load(paste(Cmodel.name, .Platform$dynlib.ext, sep = ""))

## ==========================================================================
## Define the model

## --------------------------------------------------------------------------
## ODE solver: performs a simulation, given initial state values,
## output times and time-constant, scaled, parameters
## This is an R function
R_ode = function(y, times, parms) {
  ## parms: 1:Dose_PP1M, 2:F2,         3:Duration_2,  4:ka_PP1M,
  ##        5:ka1_max,   6:ka3_max,    7:kamt1_50,    8:kamt3_50,
  ##        9:gamma,    10:CL,        11:V,          12:PP3M_start,
  ##       13:F3,       14:Dose_PP3M, 15:Q_cen_0
  ## The last two parameters should not be passed to the ODE solver, they are
  ## used here only.
  ## State variables (y) initial conditions
  y = c("Q_depot_s1" = 0,
        "Q_depot_s3" = 0,
        "Q_depot_r3" = 0,
        "Q_central"  = parms[15])
  ## The doses are specified as "events" affecting the state variables
  ## dosing times (hours)
  dose_times = c(c(0, 4, 8, 12) + parms[3] / (24 * 7),
                 17, 29, 41, 53) * 24 * 7
  N_doses = length(dose_times)
  ## changing state variables at dosing times
  vars = c("Q_depot_s1", "Q_depot_s3", "Q_depot_r3")
  v1 = (1 -  parms[2])  * parms[1]     # value applied to Q_depot_s1
  v2 = (1 -  parms[13]) * parms[14]    # value applied to Q_depot_s3
  v3 =       parms[13]  * parms[14]    # value applied to Q_depot_r3
  ## Form the events table
  eventdat = data.frame(var = c(rep("Q_depot_s1", 4), rep(vars[2:3], 4)),
                        time = c(dose_times[1:4], rep(dose_times[5:8], each=2)),
                        value = c(rep(v1, 4),
                              rep(c(v2, v3), N_doses - 4)),
                        method = "add")
  ## Integrate numerically, with outputs at specified times
  result = deSolve::lsode(y, c(0,times), func="derivs", parms=parms[1:12],
                          rtol=1e-6, atol=1e-6, dllname="PP13M_model_v04",
                          initfunc = "PP13M_model_v04",
                          events=list(data=eventdat))
  result = result[which(result[,1] %in% times),] # weed out extra times
  if (dim(result)[[1]] < length(times)) { ## integration failed
    return(rep(1E-30, length(times)))
  } else {
    ## compute central concentration, convert from mg/L to ng/ml, return
    return(result[,5] * 1E3 / parms[11])
  }
} # end of R_ode model solver

## ---------------------------------------------------------------------------
## Nimble function with nimbleRcall. This is just a wrapper.
nimble_ode = nimbleRcall(
  prototype = function(
    y     = double(1), # vector
    times = double(1), # vector
    parms = double(1)  # vector
  ) {},
  returnType = double(1), # outcome is a vector
  Rfun = 'R_ode'
)

## ---------------------------------------------------------------------------
## Hierarchical core Nimble (BUGS) code
myNimbleCode = nimbleCode({ ## BUGS (extended) code
  ##
  ## REFERENCE group population mean (with prior if not fixed)
  F2t_m       <- logit(0.168) # From Samtani paper Table III.
  F3t_m       <- logit(0.209) # transformed F3
  ka1_pp1m_mv <- log(1 + 0.02^2)
  ka1_pp1m_m  ~  dlnorm(meanlog=log(4.88E-4), varlog=ka1_pp1m_mv) # (1/h)
  ka1_max_mv  <- log(1 + 0.0696^2)
  ka1_max_m   ~  dlnorm(meanlog=log(0.0904),  varlog=ka1_max_mv)  # (mg/h)
  ka3_max_mv  <- log(1 + 0.0465^2)
  ka3_max_m   ~  dlnorm(meanlog=log(0.164),   varlog=ka3_max_mv)  # (mg/h)
  kamt1_50_mv <- log(1 + 0.0383^2)
  kamt1_50_m  ~  dlnorm(meanlog=log(120),     varlog=kamt1_50_mv) # (mg)
  kamt3_50_mv <- log(1 + 0.0952^2)
  kamt3_50_m  ~  dlnorm(meanlog=log(21.4),    varlog=kamt3_50_mv) # (mg)
  gamma_mv    <- log(1 + 0.0165^2)
  gamma_m     ~  dlnorm(meanlog=log(1.44),    varlog=gamma_mv)    # unitless
  CL_mv       <- log(1 + 0.0216^2)
  CL_m        ~  dlnorm(meanlog=log(3.84),    varlog=CL_mv)       # (L/hr)
  V_m         <- 1960   # (L)
  ##
  ## REFERENCE pop (inter-individual) SDs (with prior if not fixed)
  ##
  ## F2_sd
  omega2    <- 0.064 / (0.168 * (1 - 0.168)) # Samtani eq. 4 inverted
  F2_sd     ~  dnorm(mean = omega2, sd = 0.02*omega2)
  F2_v      <- F2_sd^2
  ## F3_v, Samtani eq. 4 inverted
  F3_v      <- (0.854 / (1 - 0.209))^2
  ##
  ka1_pp1m_cv_v <- log(1 + 0.03^2)
  ka1_pp1m_cv   ~  dlnorm(meanlog=log(0.590), varlog=ka1_pp1m_cv_v)
  ka1_pp1m_v    <- log(1 + ka1_pp1m_cv^2)
  ##
  ka1_max_cv_v  <- log(1 + 0.0501^2)
  ka1_max_cv    ~  dlnorm(meanlog=log(0.827), varlog=ka1_max_cv_v)
  ka1_max_v     <- log(1 + ka1_max_cv^2)
  ##
```

```
ka3_max_v     <- 0
##
kamt1_50_cv_v <- log(1 + 0.101^2)
kamt1_50_cv   ~  dlnorm(meanlog=log(0.500), varlog=kamt1_50_cv_v)
kamt1_50_v    <- log(1 + kamt1_50_cv^2)
##
kamt3_50_cv_v <- log(1 + 0.142^2)
kamt3_50_cv   ~  dlnorm(meanlog=log(0.867), varlog=kamt3_50_cv_v)
kamt3_50_v    <- log(1 + kamt3_50_cv^2)
##
gamma_v       <- 0
##
CL_cv_v       <- log(1 + 0.0317^2)
CL_cv         ~  dlnorm(meanlog=log(0.357), varlog=CL_cv_v)
CL_v          <- log(1 + CL_cv^2)
##
V_v           <- log(1 + 0.628^2)
##
## TEST group population mean (with prior if not fixed)
## Delta is a vector containing the factor of difference between the
## ref and test compounds for the 6 drug-release  parameters
## 1 = no difference.
## Order in Delta: f_3, k_as3,max, k_ar3,max, k_as3,50, k_ar3,50, Gamma.
## If Do_fit > 0.5, sample the first 3 Delta.
if (Do_fit > 0.5) {
  logDelta2 ~  dnorm(mean=logmeanD2, var=logvarD2)
  Delta[2]  <- exp(logDelta2)
}
F3t_m_T      <- logit(0.209 * Delta[1]) # transformed F3 with delta
ka1_max_m_T  <- ka1_max_m   * Delta[2]  # (mg/h)
ka3_max_m_T  <- ka3_max_m   * Delta[3]  # (mg/h)
kamt1_50_m_T <- kamt1_50_m  * Delta[4]  # (mg)
kamt3_50_m_T <- kamt3_50_m  * Delta[5]  # (mg)
gamma_m_T    <- gamma_m     * Delta[6]  # unitless
##
## if F3_m is modified then F3_v is modified:
if (SamEq3) { # see above
   F3_v_T    <- (0.854 / (1 - 0.209 * Delta[1]))^2
} else {
   F3_v_T    <- (abs(F3t_m_T) * 0.854)^2
}
##
## measurement error variance in log for plasma concentration, ng/ml
## if we want uncertainty on the residual error we should use:
res_cv_v    <- log(1 + 0.321^2)
res_cv      ~  dlnorm(meanlog=log(0.306), varlog=res_cv_v)
sigma2      <- log(1 + res_cv^2)
## simpler is:
## sigma2 <- log(1 + 0.306^2)

## for each REFERENCE subject
for (i in 1:nsubjects_per_arm) {
  tmp2[i]      ~  dnorm(mean = F2t_m, var = F2_v)
  F2[i]        <- ilogit(tmp2[i])
  tmp3[i]      ~  dnorm(mean = F3t_m, var = F3_v)
  F3[i]        <- ilogit(tmp3[i])
  ka_PP1M[i]   ~  dlnorm(meanlog=log(ka1_pp1m_m), varlog=ka1_pp1m_v)  # (1/hr)
  ka1_max[i]   ~  dlnorm(meanlog=log(ka1_max_m),  varlog=ka1_max_v)   # (mg/h)
  ka3_max[i]   <- ka3_max_m                                           # (mg/h)
  kamt1_50[i] ~  dlnorm(meanlog=log(kamt1_50_m), varlog=kamt1_50_v)  # (mg)
  kamt3_50[i] ~  dlnorm(meanlog=log(kamt3_50_m), varlog=kamt3_50_v)  # (mg)
  gamma[i]     <- gamma_m
  CL[i]        ~  dlnorm(meanlog=log(CL_m),       varlog=CL_v)        # (L/hr)
  V[i]         ~  dlnorm(meanlog=log(V_m),        varlog=V_v)         # (L)
  ##
  ## quantity at t 0
  Q_cen_0[i]  ~  dlnorm(meanlog=log(Q_cen_0_mean), sdlog = log(Q_cen_0_sd))
  ##
```

```
    ## likelihood for concentration measurements
    ## Call the ODE solver to get predictions for each subject
    ## predictions are plasma concentrations, in ng/ml
    Conc[i, 1:ntimes] <- nimble_ode(y[1:nstates], times[1:ntimes],
                                    c(Dose_PP1M, F2[i], Duration_2,
                                      ka_PP1M[i], ka1_max[i], ka3_max[i],
                                      kamt1_50[i], kamt3_50[i], gamma[i],
                                      CL[i], V[i], PP3M_start, F3[i],
                                      Dose_PP3M, Q_cen_0[i]))
    ## data likelihood
    for (j in 1:ntimes) {
      C_plasma_obs[i,j] ~ dlnorm(meanlog=log(Conc[i,j]), varlog=sigma2)
    }
  }
  ## for each TEST subject
  for (i in (1+nsubjects_per_arm):(2*nsubjects_per_arm)) {
    tmp2[i]     ~  dnorm(mean = F2t_m, var = F2_v)
    F2[i]       <- ilogit(tmp2[i])
    tmp3[i]     ~  dnorm(mean = F3t_m_T, var = F3_v_T)
    F3[i]       <- ilogit(tmp3[i])
    ka_PP1M[i]  ~  dlnorm(meanlog=log(ka1_pp1m_m),   varlog=ka1_pp1m_v) # (1/hr)
    ka1_max[i]  ~  dlnorm(meanlog=log(ka1_max_m_T),  varlog=ka1_max_v)  # (mg/h)
    ka3_max[i]  <- ka3_max_m_T                                          # (mg/h)
    kamt1_50[i] ~  dlnorm(meanlog=log(kamt1_50_m_T), varlog=kamt1_50_v) # (mg)
    kamt3_50[i] ~  dlnorm(meanlog=log(kamt3_50_m_T), varlog=kamt3_50_v) # (mg)
    gamma[i]    <- gamma_m_T
    CL[i]       ~  dlnorm(meanlog=log(CL_m),         varlog=CL_v)       # (L/hr)
    V[i]        ~  dlnorm(meanlog=log(V_m),          varlog=V_v)        # (L)
    ##
    ## quantity at t 0
    Q_cen_0[i]  ~  dlnorm(meanlog=log(Q_cen_0_mean), sdlog = log(Q_cen_0_sd))
    ##
    ## likelihood for concentration measurements
    ## Call the ODE solver to get predictions for each subject
    ## predictions are plasma concentrations, in ng/ml
    Conc[i, 1:ntimes] <- nimble_ode(y[1:nstates], times[1:ntimes],
                                    c(Dose_PP1M, F2[i], Duration_2,
                                      ka_PP1M[i], ka1_max[i], ka3_max[i],
                                      kamt1_50[i], kamt3_50[i], gamma[i],
                                      CL[i], V[i], PP3M_start, F3[i],
                                      Dose_PP3M, Q_cen_0[i]))
    ## data likelihood
    for (j in 1:ntimes) {
      C_plasma_obs[i,j] ~ dlnorm(meanlog=log(Conc[i,j]),varlog=sigma2)
    }
  }
}) # End myNimbleCode
## End.
```

Data-based R workflow for parallel trials (v4)

```
## Partly Bayesian paliperidone palmitate VBE workflow
## With R:Nimble package
## v4

IDtag = "_4" # version number

## =========================================================================
## 0. Read the statistical model and TOST test
source("Statistical model v13.R")
source("Our_TOST.R")
source("Cmax_AUC.R")

## =========================================================================
## 1. Compile the model for prediction of an abbreviated BE trial

N.subjects.a = 25 # parallel trial, reference / test, 20 subjects per arm

## Magnusson dosings were at PP1M: 0, 4, 8, 12, PP3M: 17, 29, 41, 53 weeks.
## Magnusson sampling times were at:
##   1, 5, 9, 13, 14, 17(-),
##   21, 25, 29(-), 33, 37, 41(-), 45, 49, 53(-), 54, 55, 57, 61, 65
## (-) indicate "just before, say 1 hour before.
## We use the same times, but in hours.
Hr1 = 1 / (24 * 7) # one hour in weeks
times = c(1, 5, 9, 13, 14, 17-Hr1,
          21, 25, 29-Hr1, 33, 37, 41-Hr1, 45, 49, 53-Hr1, 54, 55, 57, 61, 65)
times = times * 24 * 7
plot.times = times / (24 * 7) # in weeks
N.times = length(plot.times)

dose_pp1m = 150
dose_pp3m = 525

Q_cen_0_mean = 30 # geo mean
Q_cen_0_sd   = 1.5  # geo SD

# order in Delta: f_3, k_as3,max, k_ar3,max, k_as3,50, k_ar3,50, Gamma
Delta = rep(1,6)

data  = list()

inits = list(Q_cen_0_mean = Q_cen_0_mean, # (mg)
             Q_cen_0_sd   = Q_cen_0_sd,   # (mg)
             Dose_PP1M    = dose_pp1m[1], # (mg)
             Dose_PP3M    = dose_pp3m[1], # (mg)
             Delta        = Delta)

constants = list(nsubjects_per_arm = N.subjects.a,
                 Do_fit            = 0,            # 0: no fit, > 0: index
                 SamEq3            = 1,            # Boolean
                 Duration_2        = 319,          # (h)
                 PP3M_start        = 17*7*24,      # (h)
                 ntimes            = N.times,
                 times             = times,
                 nstates           = 4)

Rmodel = nimbleModel(myNimbleCode, constants, data, inits, calculate=F)
Cmodel = compileNimble(Rmodel, showCompilerOutput=F)

## =========================================================================
## 2a. Simulate an abbreviated BE trial, assuming bioequivalence

## Simulate the whole trial at once
Node.names = Cmodel$getNodeNames(includeData=T)
Cmodel$Delta[2] = 1.05 # ka slow max
```

```
Cmodel$simulate(nodes = Node.names)
all.pred = values(Cmodel, "Conc")         # individual profiles no noise
all.data = values(Cmodel, "C_plasma_obs") # for just the simulated data

all.pred = matrix(all.pred, ncol=N.times, byrow = F) # subjects by row
all.data = matrix(all.data, ncol=N.times, byrow = F) # subjects by row

## Reference arm of the trial
ref.pred.mat = all.pred[1:N.subjects.a,]
ref.data.mat = all.data[1:N.subjects.a,]

## Test arm of the trial
test.pred.mat = all.pred[(N.subjects.a + 1):(2 * N.subjects.a),]
test.data.mat = all.data[(N.subjects.a + 1):(2 * N.subjects.a),]

# parameters
param.pop = values(Cmodel,
                   c("ka1_pp1m_m", "ka1_max_m", "ka3_max_m",
                     "kamt1_50_m", "kamt3_50_m", "gamma_m", "CL_m",
                     "F2_v", "ka1_pp1m_v", "ka1_max_v", "kamt1_50_v",
                     "kamt3_50_v", "CL_v", "sigma2"))
param.ind =  values(Cmodel,
                    c("F2", "F3", "ka_PP1M","ka1_max", "ka3_max",
                      "kamt1_50", "kamt3_50", "gamma", "CL", "V"))

param.ind.mat = matrix(param.ind, nrow=N.subjects.a, byrow = F)

colnames(param.ind.mat) = c("F2_Ref", "F2_Test", "F3_Ref", "F3_Test",
                            "ka_PP1M_Ref","ka_PP1M_Test",
                            "ka1_max_Ref", "ka1_max_Test",
                            "ka3_max_Ref","ka3_max_Test",
                            "kamt1_50_Ref", "kamt1_50_Test",
                            "kamt3_50_Ref", "kamt3_50_Test",
                            "gamma_Ref", "gamma_Test", "CL_Ref",
                            "CL_Test", "V_Ref", "V_Test")

rownames(param.ind.mat) = c(paste0("subject_", 1:N.subjects.a))

param.pop.mat = matrix(param.pop, nrow=1, byrow = F)

colnames(param.pop.mat) = c("ka1_pp1m_m", "ka1_max_m", "ka3_max_m",
                            "kamt1_50_m", "kamt3_50_m", "gamma_m", "CL_m",
                            "F2_v", "ka1_pp1m_v", "ka1_max_v",
                            "kamt1_50_v", "kamt3_50_v", "CL_v", "sigma2")

## Check
plot(as.numeric(ref.pred.mat), as.numeric(ref.data.mat),
     type ="p", col="blue",lwd=0.5, log="xy")
plot(plot.times, ref.pred.mat[1,],  type ="l", col="blue",lwd=0.5)
points(plot.times, ref.data.mat[1,], col="blue",lwd=0.5)

## Plot
plot(plot.times, plot.times, type="n", xlab="Time (weeks)",
     ylab="Plasma concentration (ng/ml)", yaxt ="n",
     ylim=c(1,1000), log="y",cex.lab=1.3)
axis(2, at = c(0.001,0.01, 0.1, 1, 10, 100, 1000),
     labels=c("0.001","0.01", "0.1","1", "10", "100", "1000"), las=1)
##
for (i in 1:N.subjects.a) {
  lines(plot.times, ref.data.mat[i,],  type="b", col="blue")
  lines(plot.times, test.data.mat[i,], type="b", col="red")
}
abline(v=c(0, 4, 8, 12, 17, 29, 41, 53))
legend(x=45, y=1000, leg=c("Reference", "Test"), lty=1,
       col=c("blue", "red"), bty="o", bg="white", box.lty=0)

bSave = FALSE
if (bSave) {
  ## Save simulated data
  names.col = c("time in h", paste0("subject_Ref_", 1:N.subjects.a),
                paste0("subject_Test_", 1:N.subjects.a))
  write.table(cbind(t(t(times)),t(ref.data.mat),t(test.data.mat)),
              file = paste0("AbbreviatedTrial_25subjectsPerArm_data", IDtag,
                            ".csv"),
              sep = ",", dec = ".", row.names = FALSE, col.names = names.col)

  ## Save mean individual profiles
  write.table(cbind(t(t(times)),t(ref.pred.mat),t(test.pred.mat)),
              file = paste0("AbbreviatedTrial_25subjectsPerArm_pred", IDtag,
                            ".csv"),
              sep = ",", dec = ".", row.names = FALSE, col.names = names.col)

  ## Save parameters
  write.table(param.ind.mat,
              file = paste0("AbbreviatedTrial_25subjectsPerArm_param_ind",
                            IDtag, ".csv"),
              sep=",", dec=".",
              row.names=c(paste0("subject_", 1:N.subjects.a)))
  write.table(param.pop.mat,
              file = paste0("AbbreviatedTrial_25subjectsPerArm_param_pop",
                            IDtag, ".csv"),
              sep=",", dec=".", row.names=FALSE)
}

## =========================================================================
## 2b. Alternative: read the abbreviated trial data

ab.data = read.csv("AbbreviatedTrial_25subjectsPerArm_data_4.csv")
dim(ab.data)

N.times      =  dim(ab.data)[1]
N.subjects.a = (dim(ab.data)[2] - 1) / 2 # parallel trial subjects per arm

plot.times = ab.data$time.in.h / (24 * 7) # in weeks

## Compute Cmax
istart = 16
iend   = N.times
itime  = istart:iend
iref   = 1:N.subjects.a + 1
itest  = (N.subjects.a + 1):(2 * N.subjects.a) + 1
Cmax.ref  = get.Cmax(ab.data[itime,iref])
Cmax.test = get.Cmax(ab.data[itime,itest])

## Compute AUC for each subject in the last dosing period (using the last 5
## time points for each subject)
AUC.ref  = get.AUC(plot.times[itime], ab.data[itime,iref])
AUC.test = get.AUC(plot.times[itime], ab.data[itime,itest])

## Plot concentrations, Cmax, AUC
pdf("Abbreviated trial plot.pdf")
layout(matrix(1:4,1,4), widths=c(0.8,0.1,0.1,0.01))
par(mar=c(5,5,15,0))
plot(plot.times, plot.times, type="n", xlab="Time (weeks)",
     ylab="Plasma concentration (ng/ml)", yaxt ="n",
     ylim=c(1,1000), log="y", cex.lab=1.5)
axis(2, at = c(0.001,0.01, 0.1, 1, 10, 100, 1000),
     labels=c("0.001","0.01", "0.1","1", "10", "100", "1000"), las=1)
##
for (i in 1:N.subjects.a) {
  lines(plot.times, ab.data[,i+1], type="b", col="blue") # ref
}
for (i in (N.subjects.a + 1):(2 * N.subjects.a)) {
  lines(plot.times, ab.data[,i+1], type="b", col="red")  # test
}
abline(v=c(0, 4, 8, 12, 17, 29, 41, 53), col="gray")
```

```
legend(x=45, y=1000, leg=c("Reference", "Test"), lty=1,
       col=c("blue", "red"), bty="o", bg="white", box.lty=0)
##
## Plot Cmax
par(mar=c(5,0,15,0))
boxplot(Cmax.ref, Cmax.test, xlim=c(0.5,2.5), ylim=c(1,1000), log="y",
        col=c("royalblue", "violetred1"),
        xaxt="s", yaxt="n", xlab="", ylab="", names=c("Ref", "Test"))
text(1.5, y=1000, lab="Cmax")
##
## Plot AUC divided by time difference (to be on a concentration scale)
diffT = plot.times[iend] - plot.times[istart]
boxplot(AUC.ref/diffT, AUC.test/diffT, xlim=c(0.5,2.5),
        ylim=c(1,1000), log="y", col=c("royalblue", "violetred1"),
        yaxt="n", xlab="", ylab="", names=c("Ref", "Test"))
text(1.5, y=1000, lab=expression(AUC / Delta[t]))
dev.off()

## Perform a standard BE test (two one-sided t-test) on trial results:
Cmax.yes = myTOST(Cmax.ref, Cmax.test)
AUC.yes  = myTOST(AUC.ref, AUC.test)
BE.yes   = Cmax.yes && AUC.yes
## all FALSE

## Cmax geometric means ratio
exp(mean(log(Cmax.test)) - mean(log(Cmax.ref)))

## Cmax CV
mean(c(sd(Cmax.ref) / mean(Cmax.ref), sd(Cmax.test) / mean(Cmax.test)))

## AUC geometric means ratio
exp(mean(log(AUC.test)) - mean(log(AUC.ref)))

## AUC CV
mean(c(sd(AUC.ref) / mean(AUC.ref), sd(AUC.test) / mean(AUC.test)))

## Remove columns names
colnames(ab.data) = NULL


## ========================================================================
## 3a. Bayesian calibration of the PP3M model given abbreviated trial data.

## Parallelize
library(parallel)
N.cores = detectCores() / 2
this_cluster <- makeCluster(N.cores)

## Create a function with all the needed code
run_MCMC_allcode <- function(seed) {
  ##
  library(nimble)
  source("Statistical model v13.R")
  ##
  ab.data = read.csv("AbbreviatedTrial_25subjectsPerArm_data_4.csv")
  N.subjects.a = (dim(ab.data)[2] - 1) / 2 # parallel trial subjects per arm
  ## Remove columns names
  colnames(ab.data) = NULL
  ##
  Hr1 = 1 / (24 * 7) # one hour in weeks
  times = c(1, 5, 9, 13, 14, 17-Hr1,
            21, 25, 29-Hr1, 33, 37, 41-Hr1, 45, 49, 53-Hr1, 54, 55, 57, 61, 65)
  times = times * 24 * 7
  plot.times = times / (24 * 7) # in weeks
  N.times = length(plot.times)
  ##
  dose_pp1m = 150
  dose_pp3m = 525
  ##
  Q_cen_0_mean = 30 # geo mean
  Q_cen_0_sd   = 1.5  # geo SD
  ##
  Delta = rep(1,6)
  ##
  data  = list(C_plasma_obs=t(ab.data[,-1]))
  ##
  inits = list(Q_cen_0_mean = Q_cen_0_mean, # (mg)
               Q_cen_0_sd   = Q_cen_0_sd,   # (mg)
               Dose_PP1M    = dose_pp1m[1], # (mg)
               Dose_PP3M    = dose_pp3m[1], # (mg)
               Delta        = Delta)
  ##
  constants = list(nsubjects_per_arm = N.subjects.a,
                   Do_fit = 1,    ## 0: no Delta fit, > 0: fit
                   logmeanD2  = 0,
                   logvarD2   = log(2)^2,
                   SamEq3     = 1,            # Boolean, leave at 1
                   Duration_2 = 319,          # (h)
                   PP3M_start = 17*7*24,      # (h)
                   ntimes     = length(times),
                   times      = times,
                   nstates    = 4)
  ##
  Rmodel = nimbleModel(myNimbleCode, constants, data, inits, calculate=F)
  ##
  conf = configureMCMC(Rmodel, thin=1, # useConjugacy = FALSE,
                       nodes=c("logDelta2",
                               "tmp2", "tmp3", "ka_PP1M", "ka1_max",
                               "kamt1_50", "kamt3_50", "CL", "V", "Q_cen_0"),
                       monitors=c("logDelta2",
                                  "F2", "F3", "ka_PP1M", "ka1_max",
                                  "kamt1_50", "kamt3_50", "CL", "V", "Q_cen_0",
                                  "Conc",
                                  "logProb_logDelta2",
                                  "logProb_tmp2", "logProb_tmp3",
                                  "logProb_ka_PP1M", "logProb_ka1_max",
                                  "logProb_kamt1_50", "logProb_kamt3_50",
                                  "logProb_CL", "logProb_V", "logProb_Q_cen_0",
                                  "logProb_C_plasma_obs"))
  ##
  Rmcmc  = buildMCMC(conf)
  Cmodel = compileNimble(Rmodel, showCompilerOutput=F)
  Cmcmc  = compileNimble(Rmcmc, project=Rmodel)
  ##
  Cmodel$ka1_pp1m_m  = 4.88E-4
  Cmodel$ka1_max_m   = 0.0904
  Cmodel$ka3_max_m   = 0.164
  Cmodel$kamt1_50_m  = 120
  Cmodel$kamt3_50_m  = 21.4
  Cmodel$gamma_m     = 1.44
  Cmodel$CL_m        = 3.84
  Cmodel$F2_v        = (0.064 / (0.168 * (1 - 0.168)))^2
  Cmodel$F3_v        = (0.854 / (1 - 0.209))^2
  Cmodel$ka1_pp1m_v  = log(1 + 0.590^2)
  Cmodel$ka1_max_v   = log(1 + 0.827^2)
  Cmodel$kamt1_50_v  = log(1 + 0.5^2)
  Cmodel$kamt3_50_v  = log(1 + 0.867^2)
  Cmodel$CL_v        = log(1 + 0.357^2)
  Cmodel$res_cv      = 0.306
  ##
  mysamples = runMCMC(Cmcmc, niter=10000, nburnin=2500, setSeed=seed)
  return(mysamples)
  ##
} ## End run_MCMC_allcode

chain_output <- parLapply(cl=this_cluster, X=1:N.cores, fun=run_MCMC_allcode)
```

```
save(chain_output,
     file=paste0("Parallel chains output.Delta2.fix pop",IDtag,".Rsave"))
## load(file=paste0("Parallel chains output.Delta2.fix pop",IDtag,".Rsave"))

pdf("Parallel chains trajectories.IWish.fix pop.pdf")
par(mfrow = c(1,1))
mycolors = rainbow(N.cores)
for (j in 1:dim(chain_output[[1]])[2]) {
  for (i in 1:N.cores) {
    this_output <- chain_output[[i]]
    if (i == 1) {
      plot(this_output[,j], type = "l", ylab = colnames(this_output)[j],
           col=mycolors[i], las=1)
    } else {
      lines(this_output[,j], col=mycolors[i])
    }
  }
}
dev.off()

## Close the cluster when you're done with it.
stopCluster(this_cluster)

## ============================================================================
## 3b. Check the model using the abbreviated BE trial
## There is no point of doing that with simulations of the prior model,
## unless we want to exercise the checking tools.
## For now, we just show fit plots.

## ============================================================================
## 4. Simulate a virtual parallel bioequivalence trials with many subjects

N.subjects.v = 130 # number of virtual subjects per arm

Hr1 = 1 / (24 * 7) # one hour in weeks
times = c(1, 5, 9, 13, 14, 17-Hr1,
          21, 25, 29-Hr1, 33, 37, 41-Hr1, 45, 49, 53-Hr1, 54, 55, 57, 61, 65)
times = times * 24 * 7
plot.times = times / (24 * 7) # in weeks
N.times = length(plot.times)

dose_pp1m = 150
dose_pp3m = 525

Q_cen_0_mean = 30 # geo mean
Q_cen_0_sd   = 1.5  # geo SD

Delta = rep(1,6)

data  = list()

inits = list(Q_cen_0_mean = Q_cen_0_mean, # (mg)
             Q_cen_0_sd   = Q_cen_0_sd,   # (mg)
             Dose_PP1M    = dose_pp1m[1], # (mg)
             Dose_PP3M    = dose_pp3m[1], # (mg)
             Delta        = Delta)

constants = list(nsubjects_per_arm = N.subjects.v,
                 Do_fit            = 0,           # 0: no fit, > 0: index
                 SamEq3            = 1,           # Boolean
                 Duration_2        = 319,         # (h)
                 PP3M_start        = 17*7*24,     # (h)
                 ntimes            = N.times,
                 times             = times,
                 nstates           = 4)

Rmodel = nimbleModel(myNimbleCode, constants, data, inits, calculate=F)
```

```
Cmodel = Rmodel
Cmodel = compileNimble(Rmodel, showCompilerOutput=F)

Cmodel$Delta[2]    = 1.42 # posterior mean
Cmodel$ka1_pp1m_m  = 4.88E-4
Cmodel$ka1_max_m   = 0.0904
Cmodel$ka3_max_m   = 0.164
Cmodel$kamt1_50_m  = 120
Cmodel$kamt3_50_m  = 21.4
Cmodel$gamma_m     = 1.44
Cmodel$CL_m        = 3.84
Cmodel$F2_v        = (0.064 / (0.168 * (1 - 0.168)))^2
Cmodel$F3_v        = (0.854 / (1 - 0.209))^2
Cmodel$ka1_pp1m_v  = log(1 + 0.590^2)
Cmodel$ka1_max_v   = log(1 + 0.827^2)
Cmodel$kamt1_50_v  = log(1 + 0.5^2)
Cmodel$kamt3_50_v  = log(1 + 0.867^2)
Cmodel$CL_v        = log(1 + 0.357^2)
Cmodel$res_cv      = 0.306

## Run a BE trial
Node.names = Cmodel$getNodeNames(includeData=T)
Cmodel$simulate(nodes = Node.names)
all.res  = values(Cmodel, Node.names)
all.data = values(Cmodel, "C_plasma_obs") # for just the simulated data
all.pred = values(Cmodel, "Conc") # for just the individual profile no noise

all.data = matrix(all.data, ncol=N.times, byrow = F) # subjects by row
all.pred = matrix(all.pred, ncol=N.times, byrow = F) # subjects by row

## Reference arm of the trial
ref.data.mat = all.data[1:N.subjects.v,]
ref.pred.mat = all.pred[1:N.subjects.v,]

## Test arm of the trial
test.data.mat = all.data[(N.subjects.v + 1):(2 * N.subjects.v),]
test.pred.mat = all.pred[(N.subjects.v + 1):(2 * N.subjects.v),]

## Compute Cmax
istart = 16
iend   = N.times
itime  = istart:iend
isub   = 1:N.subjects.v
Cmax.ref  = get.Cmax(t(ref.data.mat[ ,itime]))
Cmax.test = get.Cmax(t(test.data.mat[,itime]))

## Compute AUC for each subject in the last dosing period
AUC.ref  = get.AUC(plot.times[itime], t(ref.data.mat[ ,itime]))
AUC.test = get.AUC(plot.times[itime], t(test.data.mat[,itime]))

## Plot concentrations, Cmax, AUC
pdf("Virtual trial N 130 simulated profiles.pdf")
layout(matrix(1:4,1,4), widths=c(0.8,0.1,0.1,0.01))
par(mar=c(5,5,15,0))
plot(plot.times, plot.times, type="n", xlab="Time (weeks)",
     ylab="Plasma concentration (ng/ml)", yaxt ="n",
     ylim=c(1,1000), log="y", cex.lab=1.5)
axis(2, at = c(0.001,0.01, 0.1, 1, 10, 100, 1000),
     labels=c("0.001","0.01", "0.1","1", "10", "100", "1000"), las=1)
##
for (i in 1:N.subjects.v) {
  lines(plot.times, ref.data.mat[ i,], type="b", col="blue") # ref
  lines(plot.times, test.data.mat[i,], type="b", col="red")  # test
}
abline(v=c(0, 4, 8, 12, 17, 29, 41, 53), col="gray")
legend(x=45, y=1000, leg=c("Reference", "Test"), lty=1,
       col=c("blue", "red"), bty="o", bg="white", box.lty=0)
##
```

```
## Plot Cmax
par(mar=c(5,0,15,0))
boxplot(Cmax.ref, Cmax.test, xlim=c(0.5,2.5), ylim=c(1,1000), log="y",
        col=c("royalblue", "violetred1"),
        xaxt="s", yaxt="n", xlab="", ylab="", names=c("Ref", "Test"))
text(1.5, y=1000, lab="Cmax")
##
## Plot AUC divided by time difference (to be on a concentration scale)
diffT = plot.times[iend] - plot.times[istart]
boxplot(AUC.ref/diffT, AUC.test/diffT, xlim=c(0.5,2.5),
        ylim=c(1,1000), log="y", col=c("royalblue", "violetred1"),
        yaxt="n", xlab="", ylab="", names=c("Ref", "Test"))
text(1.5, y=1000, lab=expression(AUC / Delta[t]))
dev.off()

## =====================================================================
## 5. Perform a standard BE test (two one-sided t-test) on trial results,

Cmax.yes = myTOST(Cmax.ref, Cmax.test)
AUC.yes  = myTOST(AUC.ref, AUC.test)
BE.yes   = Cmax.yes && AUC.yes

## Cmax geometric means ratio
exp(mean(log(Cmax.test)) - mean(log(Cmax.ref)))

## Cmax CV
mean(c(sd(Cmax.ref) / mean(Cmax.ref), sd(Cmax.test) / mean(Cmax.test)))

## AUC geometric means ratio
exp(mean(log(AUC.test)) - mean(log(AUC.ref)))

## AUC CV
mean(c(sd(AUC.ref) / mean(AUC.ref), sd(AUC.test) / mean(AUC.test)))

## =====================================================================
## 6. Power: probability of declaring BE when it is true.
## This is equal to (1 - type II error), where type II error is the
## probability of rejecting BE when it is true
## Compute once for many subjects and then use only part of the data
## for smaller trial sizes.

N.subjects.v = 500 # number of virtual subjects per arm

Hr1 = 1 / (24 * 7) # one hour in weeks
times = c(1, 5, 9, 13, 14, 17-Hr1,
          21, 25, 29-Hr1, 33, 37, 41-Hr1, 45, 49, 53-Hr1, 54, 55, 57, 61, 65)
times = times * 24 * 7
plot.times = times / (24 * 7) # in weeks
N.times = length(plot.times)

dose_pp1m = 150
dose_pp3m = 525

Q_cen_0_mean = 30   # geo mean
Q_cen_0_sd   = 1.5  # geo SD

## We need to recompile, because nsubjects must be a constant
constants = list(nsubjects_per_arm = N.subjects.v,
                 Do_fit            = 0,           # 0: no fit, > 0: index
                 SamEq3            = 1,           # Boolean
                 Duration_2        = 319,         # (h)
                 PP3M_start        = 17*7*24,     # (h)
                 ntimes            = N.times,
                 times             = times,
                 nstates           = 4)

data  = list()
```

```
inits = list(Q_cen_0_mean = Q_cen_0_mean, # (mg)
             Q_cen_0_sd   = Q_cen_0_sd,   # (mg)
             Dose_PP1M    = dose_pp1m[1], # (mg)
             Dose_PP3M    = dose_pp3m[1], # (mg)
             Delta        = rep(1, 6))

Rmodel = nimbleModel(myNimbleCode, constants, data, inits, calculate=F)
Cmodel = Rmodel
Cmodel = compileNimble(Rmodel, showCompilerOutput=F)

Node.names = Cmodel$getNodeNames(includeData=T)
Cmodel$simulate(nodes = Node.names)
res = values(Cmodel, Node.names)

## Main simulation: bioequivalent case
Cmodel$Delta = c(1, 1, 1, 1, 1, 1) # true BE

N.mtc.v = 1000 # number of simulated virtual trials
BE.yes = rep(-1, N.mtc.v)
Cmax.ref = Cmax.test = matrix(0, nrow=N.mtc.v, ncol=N.subjects.v)
AUC.ref  = AUC.test  = matrix(0, nrow=N.mtc.v, ncol=N.subjects.v)
istart   = 16
iend     = N.times
itime    = istart:iend
for (i in 1:N.mtc.v) {
  ## Simulate the trial
  Cmodel$simulate(nodes = Node.names)
  all.data = values(Cmodel, "C_plasma_obs")
  all.data = matrix(all.data, ncol=N.times, byrow = F) # subjects by row
  ##
  ## Reference arm of the trial
  ref.data.mat = all.data[1:N.subjects.v,]
  ##
  ## Test arm of the trial
  test.data.mat = all.data[(N.subjects.v + 1):(2 * N.subjects.v),]
  ##
  ## find Cmax for each subject, record it
  Cmax.ref[i,]  = get.Cmax(t(ref.data.mat[ ,itime]))
  Cmax.test[i,] = get.Cmax(t(test.data.mat[,itime]))
  ##
  ## find AUC for each subject, record it
  AUC.ref[i,]  = get.AUC(plot.times[itime], t(ref.data.mat[, itime]))
  AUC.test[i,] = get.AUC(plot.times[itime], t(test.data.mat[,itime]))
  ##
  print(paste("Trial", i))
}

save(list=c("N.mtc.v", "N.subjects.v", "Cmax.ref", "Cmax.test", "AUC.ref",
            "AUC.test"), file=paste0("Cmax & AUC for power",IDtag,".Rsave"))
## load(file=paste0("Cmax & AUC for power",IDtag,".Rsave"))

## Now process increasing chunks of the results
power = rep(0, N.subjects.v)
BE.yes = rep(0, N.mtc.v)
for (k in 2:N.subjects.v) {
  N.subj.current = k
  my.index = sample.int(N.subjects.v, N.subj.current)
  for (i in 1:N.mtc.v) {
    Cmax.yes = myTOST(Cmax.ref[i,my.index], Cmax.test[i,my.index])
    AUC.yes  = myTOST(AUC.ref[i,my.index], AUC.test[i,my.index])
    #
    BE.yes[i] = Cmax.yes && AUC.yes
  }
  ## power:
  power[k] = sum(BE.yes) / N.mtc.v # probability of declaring BE
}

## Plot
```

```
pdf("Power plot after calibration.pdf")
plot(2:N.subjects.v, power[-1], type="l", xlab="Number of subjects per arm",
     ylab="Probability of declaring bioequivalence", xlim=c(1,N.subjects.v),
     ylim=c(0, 1), col="grey", lwd=0.5, las=1)
lines(2:N.subjects.v, supsmu(2:N.subjects.v, power[-1])$y, lwd=2, col="red")
dev.off()

## =========================================================================
## 7. Type I error: probability of declaring BE when it is not true.

## Compute once for many subjects and then use only part of the data
## for smaller trial sizes.

N.subjects.v = 500 # number of virtual subjects per arm

Hr1 = 1 / (24 * 7) # one hour in weeks
times = c(1, 5, 9, 13, 14, 17-Hr1,
          21, 25, 29-Hr1, 33, 37, 41-Hr1, 45, 49, 53-Hr1, 54, 55, 57, 61, 65)
times = times * 24 * 7
plot.times = times / (24 * 7) # in weeks
N.times = length(plot.times)

dose_pp1m = 150
dose_pp3m = 525

Q_cen_0_mean = 30   # geo mean
Q_cen_0_sd   = 1.5  # geo SD

## We need to recompile, because nsubjects must be a constant
constants = list(nsubjects_per_arm = N.subjects.v,
                 Do_fit            = 0,            # 0: no fit, > 0: index
                 SamEq3            = 1,            # Boolean
                 Duration_2        = 319,          # (h)
                 PP3M_start        = 17*7*24,      # (h)
                 ntimes            = N.times,
                 times             = times,
                 nstates           = 4)

data  = list()

inits = list(Q_cen_0_mean = Q_cen_0_mean, # (mg)
             Q_cen_0_sd   = Q_cen_0_sd,   # (mg)
             Dose_PP1M    = dose_pp1m[1], # (mg)
             Dose_PP3M    = dose_pp3m[1], # (mg)
             Delta        = rep(1, 6))

Rmodel = nimbleModel(myNimbleCode, constants, data, inits, calculate=F)
Cmodel = compileNimble(Rmodel, showCompilerOutput=F)

Node.names = Cmodel$getNodeNames(includeData=T)
Cmodel$simulate(nodes = Node.names)
res = values(Cmodel, Node.names)

## Main simulation
N.mtc.v = 1000 # number of simulated virtual trials
BE.yes = rep(-1, N.mtc.v)
Delta    = matrix(0, nrow=N.mtc.v, ncol=6)
Cmax.ref = Cmax.test = matrix(0, nrow=N.mtc.v, ncol=N.subjects.v)
AUC.ref  = AUC.test  = matrix(0, nrow=N.mtc.v, ncol=N.subjects.v)
istart   = 16
iend     = N.times
itime    = istart:iend
t.start = Sys.time()
for (i in 1:N.mtc.v) {
  ## Simulate the trial
  Delta[i,]    = runif(6, 0.7, 1.5)
  Cmodel$Delta = Delta[i,]
  Cmodel$simulate(nodes = Node.names)
  all.data = values(Cmodel, "C_plasma_obs")
  all.data = matrix(all.data, ncol=N.times, byrow = F) # subjects by row
  ##
  ## Reference arm of the trial
  ref.data.mat = all.data[1:N.subjects.v,]
  ##
  ## Test arm of the trial
  test.data.mat = all.data[(N.subjects.v + 1):(2 * N.subjects.v),]
  ##
  ## find Cmax for each subject, record it
  Cmax.ref[i,]  = get.Cmax(t(ref.data.mat[ ,16:N.times]))
  Cmax.test[i,] = get.Cmax(t(test.data.mat[,16:N.times]))
  ##
  ## find AUC for each subject, record it
  AUC.ref[i,]  = get.AUC(plot.times[itime], t(ref.data.mat[, itime]))
  AUC.test[i,] = get.AUC(plot.times[itime], t(test.data.mat[,itime]))
  ##
  print(paste("Trial", i))
}
t.end = Sys.time()
t.end - t.start
## 98% of the time is spend simulating trials

## Save
save(list=c("N.mtc.v", "N.subjects.v", "Delta",
            "Cmax.ref", "Cmax.test", "AUC.ref", "AUC.test"),
     file=paste0("Cmax & AUC for type 1 error", IDtag,".Rsave"))
## load(file=paste0("Cmax & AUC for type 1 error",IDtag,".Rsave"))

## Set number of subjects to use; we can use at most N.subjects.v subjects
N.subject.used = N.subjects.v
j = 1:N.subject.used

## Cmax difference per trial
Cmax.ref.means  = apply(log(Cmax.ref[,j]),  MARGIN=1, FUN=mean)
Cmax.test.means = apply(log(Cmax.test[,j]), MARGIN=1, FUN=mean)
## Test/Ref relative differences
Cmax.rel.diffs = exp(Cmax.test.means - Cmax.ref.means)
mean(Cmax.rel.diffs)

## AUC difference per trial
AUC.ref.means  = apply(log(AUC.ref[,j]),  MARGIN=1, FUN=mean)
AUC.test.means = apply(log(AUC.test[,j]), MARGIN=1, FUN=mean)
## Test/Ref relative differences
AUC.rel.diffs = exp(AUC.test.means - AUC.ref.means)
mean(AUC.rel.diffs)

## Compute BE for the various trials
Cmax.yes = AUC.yes = BE.yes = rep(0, N.mtc.v)
for (i in 1:N.mtc.v) {
  Cmax.yes[i] = myTOST(Cmax.ref[i,], Cmax.test[i,])
  AUC.yes[i]  = myTOST(AUC.ref[i,], AUC.test[i,])
  BE.yes[i]   = Cmax.yes[i] && AUC.yes[i]
}

## Plot passes
pdf("Type 1 error plot.pdf")
par(mar=c(8,5,7,1), las=1, cex.lab=1.2)
plot(Cmax.rel.diffs, Cmax.yes, type="p",
     xlab="Cmax relative differences geometric mean",
     ylab="Probability of declaring Cmax BE")
abline(v=c(0.8, 1.25), col="red")
plot(AUC.rel.diffs, AUC.yes, type="p",
     xlab="AUC relative differences geometric mean",
     ylab="Probability of declaring AUC BE")
abline(v=c(0.8, 1.25), col="red")
## lines(AUC.centers, AUC.error1, col="blue")
dev.off()
```

```
## ========================================================================
## 8. Safe space: probability of declaring BE when it is not true.
##    We have saved the Delta and BE decisions in the type 1 error
##    calculations; we can just reuse them.

## The saved simulation are the same as type 1 error but a copy has been made.
## load(file=paste0("Cmax & AUC for type 1 error",IDtag,".Rsave"))
load(file=paste0("Cmax & AUC for safe space",IDtag,".Rsave"))

## Set number of subjects to use; we can use at most N.subjects.v subjects
N.subject.used = N.subjects.v
j = 1:N.subject.used

## Cmax difference per trial
Cmax.ref.means  = apply(log(Cmax.ref[,j]),  MARGIN=1, FUN=mean)
Cmax.test.means = apply(log(Cmax.test[,j]), MARGIN=1, FUN=mean)
## Test/Ref relative differences
Cmax.rel.diffs = exp(Cmax.test.means - Cmax.ref.means)
mean(Cmax.rel.diffs)

## AUC difference per trial
AUC.ref.means  = apply(log(AUC.ref[,j]),  MARGIN=1, FUN=mean)
AUC.test.means = apply(log(AUC.test[,j]), MARGIN=1, FUN=mean)
## Test/Ref relative differences
AUC.rel.diffs = exp(AUC.test.means - AUC.ref.means)
mean(AUC.rel.diffs)

## Compute BE for the various trials
Cmax.yes = AUC.yes = BE.yes = rep(0, N.mtc.v)
for (i in 1:N.mtc.v) {
  Cmax.yes[i] = myTOST(Cmax.ref[i,], Cmax.test[i,])
  AUC.yes[i]  = myTOST(AUC.ref[i,], AUC.test[i,])
  BE.yes[i]   = Cmax.yes[i] && AUC.yes[i]
}

## Plot safe space
pdf("Safe space plot.pdf")
is.BE = which(BE.yes == 1) # we could also use Cmax.yes or AUC.yes
my.panel = function (x,y) {
  par(las=1, cex.lab=1.2)
  points(x[ is.BE], y[ is.BE], pch=16, cex=0.3, col="green")
  points(x[-is.BE], y[-is.BE], pch=16, cex=0.3, col="red")
}
my.labels = c(expression(delta[1]), expression(delta[2]), expression(delta[3]),
              expression(delta[4]), expression(delta[5]), expression(delta[6]))
pairs(Delta, panel=my.panel, labels=my.labels)
dev.off()

## End.
```

Statistical model in R (v14) for cross-over trials

```
## R/Nimble code for PP1M/PP3M population model
## v14.
## ========================================================================

library(nimble)

## compile C ODE model for deSolve
Cmodel.name = "PP13M_model_v05"
## system(paste0("R CMD SHLIB ", Cmodel.name, ".c"))
dyn.load(paste(Cmodel.name, .Platform$dynlib.ext, sep = ""))
## dyn.unload(paste(Cmodel.name, .Platform$dynlib.ext, sep = ""))

## ------------------------------------------------------------------------
## ODE solver: performs a simulation, given initial state values,
## output times and time-constant, scaled, parameters
## This is an R function
R_ode = function(y, times, parms) {
  ## parms: 1:Dose_PP1M, 2:F2,          3:Duration_2,  4:ka_PP1M,
  ##        5:ka1_max,    6:ka3_max,    7:kamt1_50,    8:kamt3_50,
  ##        9:gamma,     10:CL,        11:V,          12:PP3M_start,
  ##       13:F3,        14:Dose_PP3M, 15:Q_cen_0,    16:F3_2,
  ##       17:ka1_max_2
  ## Parameters 13-16 should not be passed to the ODE solver, they are
  ## used here only.
  ## State variables (y) initial conditions
  y = c("Q_depot_s1" = 0,
        "Q_depot_s3" = 0,
        "Q_depot_r3" = 0,
        "Q_central"  = parms[15])
  ## The doses are specified as "events" affecting the state variables
  ## dosing times (hours)
  dose_times = c(c(0, 4, 8, 12) + parms[3] / (24 * 7),
                 17, 29, 41, 53, 65, 77, 89, 101) * 24 * 7
  N_doses = length(dose_times)
  ## changing state variables at dosing times
  vars = c("Q_depot_s1", "Q_depot_s3", "Q_depot_r3")
  v1 = (1 -  parms[2])  * parms[1]  # applied to Q_depot_s1
  v2 = (1 -  parms[13]) * parms[14] # applied to Q_depot_s3
  v3 =       parms[13]  * parms[14] # applied to Q_depot_r3
  v4 = (1 -  parms[16]) * parms[14] # applied to Q_depot_s3 during test dosing
  v5 =       parms[16]  * parms[14] # applied to Q_depot_r3 during test dosing
  ## Form the events table
  eventdat = data.frame(var = c(rep("Q_depot_s1", 4), rep(vars[2:3], 8)),
                        time = c(dose_times[1:4],
                                 rep(dose_times[5:12], each=2)),
                        value = c(rep(v1, 4),
                                  rep(c(v2, v3), 4), rep(c(v4, v5), 4)),
                        method = "add")
  ## Integrate numerically, with outputs at specified times
  result = deSolve::lsode(y, c(0,times), func="derivs", parms=parms[c(1:12,17)],
                          rtol=1e-6, atol=1e-6, dllname="PP13M_model_v05",
                          initfunc = "PP13M_model_v05",
                          events=list(data=eventdat))
  result = result[which(result[,1] %in% times),] # weed out extra times
  if (dim(result)[[1]] < length(times)) { ## integration failed
    return(rep(1E-30, length(times)))
  } else {
    ## compute central concentration, convert from mg/L to ng/ml, return
    return(result[,5] * 1E3 / parms[11])
  }
} # end of R_ode model solver

## -------------------------------------------------------------------------
## Nimble function with nimbleRcall. This is just a wrapper
nimble_ode = nimbleRcall(
  prototype = function(
    y     = double(1), # vector
    times = double(1), # vector
    parms = double(1)  # vector
  ) {},
  returnType = double(1), # outcome is a vector
  Rfun = 'R_ode'
)

## -------------------------------------------------------------------------
## Hierarchical core Nimble (BUGS) code
myNimbleCode = nimbleCode({ ## BUGS (extended) code
  ##
  ## REFERENCE population mean (with prior if not fixed)
  F2t_m        <- logit(0.168) # From Samtani paper Table III.
  # from online resource 3 Magnusson, F2 = 0.153
  F3t_m        <- logit(0.209) # transformed F3
  ka1_pp1m_mv  <- log(1 + 0.02^2)
```

```
ka1_pp1m_m  ~  dlnorm(meanlog=log(4.88E-4), varlog=ka1_pp1m_mv) # (1/h)
ka1_max_mv  <- log(1 + 0.0696^2)
ka1_max_m   ~  dlnorm(meanlog=log(0.0904),  varlog=ka1_max_mv)  # (mg/h)
ka3_max_mv  <- log(1 + 0.0465^2)
ka3_max_m   ~  dlnorm(meanlog=log(0.164),   varlog=ka3_max_mv)  # (mg/h)
kamt1_50_mv <- log(1 + 0.0383^2)
kamt1_50_m  ~  dlnorm(meanlog=log(120),     varlog=kamt1_50_mv) # (mg)
kamt3_50_mv <- log(1 + 0.0952^2)
kamt3_50_m  ~  dlnorm(meanlog=log(21.4),    varlog=kamt3_50_mv) # (mg)
gamma_mv    <- log(1 + 0.0165^2)
gamma_m     ~  dlnorm(meanlog=log(1.44),    varlog=gamma_mv)    # unitless
CL_mv       <- log(1 + 0.0216^2)
CL_m        ~  dlnorm(meanlog=log(3.84),    varlog=CL_mv)       # (L/hr)
V_m         <- 1960   # (L)
##
## REFERENCE pop (inter-individual) SDs (with prior if not fixed)
##
## FB: I am not sure about F2_v and F3_v. See eqs. 3 and 4 of Samtani.
## Their reported SD (or CV in Magnusson) may apply to F2 or F2t, the
## logit transform. Same for F3.
## OK, it does not make much of a difference any way; stick to SamEq3 true
if (SamEq3) { # SD or CV applies to F2, so: detransform
  ## F2_sd
  omega2    <- 0.064 / (0.168 * (1 - 0.168)) # Samtani eq. 4 inverted
  F2_sd     ~  dnorm(mean = omega2, sd = 0.02*omega2)
  F2_v      <- F2_sd^2
  ## F3_v, Samtani eq. 4 inverted
  F3_v      <- (0.854 / (1 - 0.209))^2
} else { # SD or CV applies to F2 transformed
  F2_sd     ~  dnorm(mean = 0.064, sd = 0.02*0.064)
  F3_v      <- (abs(F3t_m) * 0.854)^2
}
##
ka1_pp1m_cv_v <- log(1 + 0.03^2)
ka1_pp1m_cv   ~  dlnorm(meanlog=log(0.590), varlog=ka1_pp1m_cv_v)
ka1_pp1m_v    <- log(1 + ka1_pp1m_cv^2)
##
ka1_max_cv_v  <- log(1 + 0.0501^2)
ka1_max_cv    ~  dlnorm(meanlog=log(0.827), varlog=ka1_max_cv_v)
ka1_max_v     <- log(1 + ka1_max_cv^2)
##
ka3_max_v     <- 0
##
kamt1_50_cv_v <- log(1 + 0.101^2)
kamt1_50_cv   ~  dlnorm(meanlog=log(0.500), varlog=kamt1_50_cv_v)
kamt1_50_v    <- log(1 + kamt1_50_cv^2)
##
kamt3_50_cv_v <- log(1 + 0.142^2)
kamt3_50_cv   ~  dlnorm(meanlog=log(0.867), varlog=kamt3_50_cv_v)
kamt3_50_v    <- log(1 + kamt3_50_cv^2)
##
gamma_v       <- 0
##
CL_cv_v       <- log(1 + 0.0317^2)
CL_cv         ~  dlnorm(meanlog=log(0.357), varlog=CL_cv_v)
CL_v          <- log(1 + CL_cv^2)
##
V_v           <- log(1 + 0.628^2)
##
## TEST population mean (with prior if not fixed)
## Delta is a vector containing the factor of difference between the
## ref and test compounds for the first 2 absorption parameters
## 1 = no difference.
## Order in Delta: f_3, k_as3,max.
## If Do_fit > 0.5, sample Delta2.
if (Do_fit > 0.5) {
  logDelta2 ~  dnorm(mean=logmeanD2, var=logvarD2)
  Delta[2]  <- exp(logDelta2)
```

```
}
## Note that we were changing the pop mean on AVERAGE, the realized
## pop mean values could differ by more of less than Delta because of noise:
## ka1_max_m_T ~  dlnorm(meanlog=log(0.0904*Delta),varlog=ka1_max_mv)# (mg/h)
## It is simpler and more robust to do:
F3t_m_T      <- logit(0.209 * Delta[1]) # transformed F3 with delta
ka1_max_m_T  <- ka1_max_m   * Delta[2]  # (mg/h)
##
## if F3_m is modified then F3_v is modified:
if (SamEq3) { # see above
   F3_v_T    <- (0.854 / (1 - 0.209 * Delta[1]))^2
} else {
   F3_v_T    <- (abs(F3t_m_T) * 0.854)^2
}
##
## measurement error variance in log for plasma concentration, ng/ml
## if we want uncertainty on the residual error we should use:
res_cv_v    <- log(1 + 0.321^2)
res_cv      ~  dlnorm(meanlog=log(0.306), varlog=res_cv_v)
sigma2      <- log(1 + res_cv^2)

## for each subject of group 1 (REFERENCE then TEST sequence)
for (i in 1:nsubjects_per_arm) {
  ## for REFERENCE drug
  tmp2[i]     ~  dnorm(mean = F2t_m, var = F2_v)
  F2[i]       <- ilogit(tmp2[i])
  tmp3[i]     ~  dnorm(mean = F3t_m, var = F3_v)
  F3[i]       <- ilogit(tmp3[i])
  ka_PP1M[i]  ~  dlnorm(meanlog=log(ka1_pp1m_m), varlog=ka1_pp1m_v)  # (1/hr)
  ka1_max[i]  ~  dlnorm(meanlog=log(ka1_max_m),  varlog=ka1_max_v)    # (mg/h)
  ka3_max[i]  <- ka3_max_m                                            # (mg/h)
  kamt1_50[i] ~  dlnorm(meanlog=log(kamt1_50_m), varlog=kamt1_50_v)  # (mg)
  kamt3_50[i] ~  dlnorm(meanlog=log(kamt3_50_m), varlog=kamt3_50_v)  # (mg)
  gamma[i]    <- gamma_m
  CL[i]       ~  dlnorm(meanlog=log(CL_m),       varlog=CL_v)        # (L/hr)
  V[i]        ~  dlnorm(meanlog=log(V_m),        varlog=V_v)         # (L)
  ##
  ## for TEST drug
  tmp3_2[i]    ~  dnorm(mean = F3t_m_T, var = F3_v_T)
  F3_2[i]      <- ilogit(tmp3_2[i])
  ka1_max_2[i] ~  dlnorm(meanlog=log(ka1_max_m_T),  varlog=ka1_max_v) # (mg/h)
  ##
  ## quantity at t 0
  Q_cen_0[i] ~  dlnorm(meanlog=log(Q_cen_0_mean), sdlog = log(Q_cen_0_sd))
  ##
  ## likelihood for concentration measurements
  ## Call the ODE solver to get predictions for each subject
  ## predictions are plasma concentrations, in ng/ml
  Conc[i, 1:ntimes] <- nimble_ode(y[1:nstates], times[1:ntimes],
                                  c(Dose_PP1M, F2[i], Duration_2,
                                    ka_PP1M[i], ka1_max[i], ka3_max[i],
                                    kamt1_50[i], kamt3_50[i], gamma[i],
                                    CL[i], V[i], PP3M_start, F3[i],
                                    Dose_PP3M, Q_cen_0[i],
                                    F3_2[i], ka1_max_2[i]))
  ## data likelihood
  for (j in 1:ntimes) {
    C_plasma_obs[i,j] ~ dlnorm(meanlog=log(Conc[i,j]), varlog=sigma2)
  }
}

## for each subject of group 2 (TEST then REFERENCE sequence)
for (i in (1+nsubjects_per_arm):(2*nsubjects_per_arm)) {
  ## for TEST drug
  tmp2[i]      ~  dnorm(mean = F2t_m, var = F2_v)
  F2[i]        <- ilogit(tmp2[i])
  tmp3_2[i]    ~  dnorm(mean = F3t_m_T, var = F3_v_T)
  F3_2[i]      <- ilogit(tmp3_2[i])
```

```
    ka_PP1M[i]   ~  dlnorm(meanlog=log(ka1_pp1m_m), varlog=ka1_pp1m_v)  # (1/hr)
    ka1_max_2[i] ~  dlnorm(meanlog=log(ka1_max_m_T),varlog=ka1_max_v)   # (mg/h)
    ka3_max[i]   <- ka3_max_m                                           # (mg/h)
    kamt1_50[i]  ~  dlnorm(meanlog=log(kamt1_50_m), varlog=kamt1_50_v)  # (mg)
    kamt3_50[i]  ~  dlnorm(meanlog=log(kamt3_50_m), varlog=kamt3_50_v)  # (mg)
    gamma[i]     <- gamma_m
    CL[i]        ~  dlnorm(meanlog=log(CL_m),       varlog=CL_v)        # (L/hr)
    V[i]         ~  dlnorm(meanlog=log(V_m),        varlog=V_v)         # (L)
    ##
    ## for REFERENCE drug
    tmp3[i]    ~  dnorm(mean = F3t_m, var = F3_v)
    F3[i]      <- ilogit(tmp3[i])
    ka1_max[i] ~  dlnorm(meanlog=log(ka1_max_m),    varlog=ka1_max_v)  # (mg/h)
    ##
    ## quantity at t 0
    Q_cen_0[i] ~  dlnorm(meanlog=log(Q_cen_0_mean), sdlog = log(Q_cen_0_sd))
    ##
    ## likelihood for concentration measurements
    ## Call the ODE solver to get predictions for each subject
    ## predictions are plasma concentrations, in ng/ml
    Conc[i, 1:ntimes] <- nimble_ode(y[1:nstates], times[1:ntimes],
                                    c(Dose_PP1M, F2[i], Duration_2,
                                      ka_PP1M[i], ka1_max_2[i], ka3_max[i],
                                      kamt1_50[i], kamt3_50[i], gamma[i],
                                      CL[i], V[i], PP3M_start, F3_2[i],
                                      Dose_PP3M, Q_cen_0[i],
                                      F3[i], ka1_max[i]))
    ## data likelihood
    for (j in 1:ntimes) {
      C_plasma_obs[i,j] ~ dlnorm(meanlog=log(Conc[i,j]), varlog=sigma2)
    }
  }
}) # End myNimbleCode
```

Data-based R workflow for cross-over trials (v5)

```
## Paliperidone palmitate VBE workflow
## v5: Switch to a simple cross-over study design for the large VBE trial.

IDtag = "_5" # version number

## =========================================================================
## 0. Read the statistical model and TOST test
source("Statistical model v14.R")
source("Our_TOST.R")
source("Cmax_AUC.R")

## =========================================================================
## 4. Simulate a virtual crossover bioequivalence trial with many subjects

N.subjects.v = 130 # number of virtual subjects per arm

Hr1 = 1 / (24 * 7) # one hour in weeks
times = c(1, 5, 9, 13, 14, 17-Hr1,
          21, 25, 29-Hr1, 33, 37, 41-Hr1, 45, 49, 53-Hr1, 54, 55, 57, 61,
          65-Hr1, 69, 73, 77-Hr1, 81, 85, 89-Hr1, 93, 97, 101-Hr1, 102, 103,
          105, 109, 113)
times = times * 24 * 7
plot.times = times / (24 * 7) # in weeks
N.times = length(plot.times)

dose_pp1m = 150
dose_pp3m = 525

Q_cen_0_mean = 30 # geo mean
Q_cen_0_sd   = 1.5  # geo SD

## Factor Delta applied to population mean reference parameters to get
## population mean test parameters. Order in Delta: f_3, k_as3,max
Delta = rep(1,2)

data  = list()

inits = list(Q_cen_0_mean = Q_cen_0_mean, # (mg)
             Q_cen_0_sd   = Q_cen_0_sd,   # (mg)
             Dose_PP1M    = dose_pp1m[1], # (mg)
             Dose_PP3M    = dose_pp3m[1], # (mg)
             Delta        = Delta)

constants = list(nsubjects_per_arm = N.subjects.v,
                 Do_fit            = 0,           # 0: no fit, > 0: index
                 SamEq3            = 1,           # Boolean
                 Duration_2        = 319,         # (h)
                 PP3M_start        = 17*7*24,     # (h)
                 ntimes            = N.times,
                 times             = times,
                 nstates           = 4)

Rmodel = nimbleModel(myNimbleCode, constants, data, inits, calculate=F)
Cmodel = Rmodel
Cmodel = compileNimble(Rmodel, showCompilerOutput=F)

## Reference population means and variances MLE
Cmodel$Delta[2]    = 1.42 # posterior mean after abbreviated trial calibration
Cmodel$ka1_pp1m_m  = 4.88E-4
Cmodel$ka1_max_m   = 0.0904
Cmodel$ka3_max_m   = 0.164
Cmodel$kamt1_50_m  = 120
Cmodel$kamt3_50_m  = 21.4
Cmodel$gamma_m     = 1.44
Cmodel$CL_m        = 3.84
Cmodel$F2_v        = (0.064 / (0.168 * (1 - 0.168)))^2
Cmodel$F3_v        = (0.854 / (1 - 0.209))^2
Cmodel$ka1_pp1m_v  = log(1 + 0.590^2)
Cmodel$ka1_max_v   = log(1 + 0.827^2)
Cmodel$kamt1_50_v  = log(1 + 0.5^2)
Cmodel$kamt3_50_v  = log(1 + 0.867^2)
Cmodel$CL_v        = log(1 + 0.357^2)
Cmodel$res_cv      = 0.306

## Run a BE trial
Node.names = Cmodel$getNodeNames(includeData=T)
Cmodel$simulate(nodes = Node.names)
all.res  = values(Cmodel, Node.names)
all.data = values(Cmodel, "C_plasma_obs") # for just the simulated data
all.pred = values(Cmodel, "Conc") # for just the individual profile no noise

all.data = matrix(all.data, ncol=N.times, byrow = F) # subjects by row
all.pred = matrix(all.pred, ncol=N.times, byrow = F) # subjects by row

## Group 1 (REFERENCE then TEST sequence) arm of trial
g1.pred.mat = all.pred[1:N.subjects.v,]
g1.data.mat = all.data[1:N.subjects.v,]

## Group 2 (TEST then REFERENCE sequence) arm of trial
g2.pred.mat = all.pred[(N.subjects.v + 1):(2 * N.subjects.v),]
g2.data.mat = all.data[(N.subjects.v + 1):(2 * N.subjects.v),]

## save the simulation
## save(list=c("Node.names", "all.res", "g1.data.mat", "g2.data.mat"),
##      file="Virtual trial N 130 simulated profiles_v6.Rsave")
load(file="Virtual trial N 130 simulated profiles_v6.Rsave")

## Compute Cmax for each subject in the last dosing period of PP3M and PP3Mt
itime  = 16:20  # sampling times in last dosing period of 1st PP3M formulation
jtime  = 30:34  # sampling times in last dosing period of 2nd PP3M formulation
```

```
Cmax.g1_R = get.Cmax(t(g1.data.mat[,itime]))
Cmax.g1_T = get.Cmax(t(g1.data.mat[,jtime]))
Cmax.g2_T = get.Cmax(t(g2.data.mat[,itime]))
Cmax.g2_R = get.Cmax(t(g2.data.mat[,jtime]))

## Compute AUC for each subject in the last dosing period of PP3M and PP3Mt
AUC.g1_R = get.AUC(plot.times[itime], t(g1.data.mat[,itime]))
AUC.g1_T = get.AUC(plot.times[jtime], t(g1.data.mat[,jtime]))
AUC.g2_T = get.AUC(plot.times[itime], t(g2.data.mat[,itime]))
AUC.g2_R = get.AUC(plot.times[jtime], t(g2.data.mat[,jtime]))

## Plot concentrations, Cmax, AUC
pdf("Virtual trial N 130 simulated profiles.pdf")
layout(matrix(c(1,4,2,5,3,6,0,0),2,4), widths=c(0.8,0.1,0.1,0.01))
##
## Group 1 plots
par(mar=c(5,5,3,0))
plot(plot.times, plot.times, type="n", xlab="Time (weeks)",
     ylab="Plasma concentration (ng/ml)", yaxt ="n",
     ylim=c(1,1000), log="y", cex.lab=1.5)
axis(2, at = c(0.001,0.01, 0.1, 1, 10, 100, 1000),
     labels=c("0.001","0.01", "0.1","1", "10", "100", "1000"), las=1)
##
for (i in 1:N.subjects.v) {
  lines(plot.times, g1.data.mat[ i,], type="b", col="blue") # group 1
}
abline(v=c(0, 4, 8, 12, 17, 29, 41, 53, 65, 77, 89, 101), col="gray")
legend(x=45, y=1000, leg="Group 1", lty=1,
       col="blue", bty="o", bg="white", box.lty=0)
##
## Plot Cmax
par(mar=c(5,0,3,0))
boxplot(Cmax.g1_R, Cmax.g1_T, xlim=c(0.5,2.5), ylim=c(1,1000), log="y",
        col=c("royalblue1", "royalblue3"),
        xaxt="s", yaxt="n", xlab="", ylab="", names=c("Ref", "Test"))
text(1.5, y=1000, lab="Cmax")
##
## Plot AUC divided by time difference (to be on a concentration scale)
diffT = plot.times[20] - plot.times[16]
diffT_T = plot.times[34] - plot.times[30]
boxplot(AUC.g1_R/diffT, AUC.g1_T/diffT_T, xlim=c(0.5,2.5),
        ylim=c(1,1000), log="y", col=c("royalblue1", "royalblue3"),
        yaxt="n", xlab="", ylab="", names=c("Ref", "Test"))
text(1.5, y=1000, lab=expression(AUC / Delta[t]))
##
## Group 2 plots
par(mar=c(5,5,3,0))
plot(plot.times, plot.times, type="n", xlab="Time (weeks)",
     ylab="Plasma concentration (ng/ml)", yaxt ="n",
     ylim=c(1,1000), log="y", cex.lab=1.5)
axis(2, at = c(0.001,0.01, 0.1, 1, 10, 100, 1000),
     labels=c("0.001","0.01", "0.1","1", "10", "100", "1000"), las=1)
##
for (i in 1:N.subjects.v) {
  lines(plot.times, g2.data.mat[i,], type="b", col="red")  # group 2
}
abline(v=c(0, 4, 8, 12, 17, 29, 41, 53, 65, 77, 89, 101), col="gray")
legend(x=45, y=1000, leg="Group 2", lty=1,
       col="red", bty="o", bg="white", box.lty=0)
##
## Plot Cmax
par(mar=c(5,0,3,0))
boxplot(Cmax.g2_T, Cmax.g2_R, xlim=c(0.5,2.5), ylim=c(1,1000), log="y",
        col=c("violetred1", "violetred3"),
        xaxt="s", yaxt="n", xlab="", ylab="", names=c("Test", "Ref"))
text(1.5, y=1000, lab="Cmax")
##
## Plot AUC divided by time difference (to be on a concentration scale)
```

```
diffT_T = plot.times[20] - plot.times[16]
diffT = plot.times[34] - plot.times[30]
boxplot(AUC.g2_T/diffT_T, AUC.g2_R/diffT, xlim=c(0.5,2.5),
        ylim=c(1,1000), log="y", col=c("violetred1", "violetred3"),
        yaxt="n", xlab="", ylab="", names=c("Test", "Ref"))
text(1.5, y=1000, lab=expression(AUC / Delta[t]))
dev.off()

## =========================================================================
## 5. Perform BE test (TOST) on 2x2 crossover trial results,

Cmax.yes = myTOST.2x2(Cmax.g1_R, Cmax.g1_T, Cmax.g2_T, Cmax.g2_R)
AUC.yes  = myTOST.2x2(AUC.g1_R,  AUC.g1_T,  AUC.g2_T,  AUC.g2_R)
BE.yes   = Cmax.yes && AUC.yes

## =========================================================================
## 6. Power: probability of declaring BE when it is true.
## This is equal to (1 - type II error), where type II error is the
## probability of rejecting BE when it is true

## Compute once for many subjects and then use only part of the data
## for smaller trial sizes.

N.subjects.v = 500 # number of virtual subjects per arm

Hr1 = 1 / (24 * 7) # one hour in weeks
times = c(1, 5, 9, 13, 14, 17-Hr1,
          21, 25, 29-Hr1, 33, 37, 41-Hr1, 45, 49, 53-Hr1, 54, 55, 57, 61,
          65-Hr1, 69, 73, 77-Hr1, 81, 85, 89-Hr1, 93, 97, 101-Hr1, 102, 103,
          105, 109, 113)
times = times * 24 * 7
plot.times = times / (24 * 7) # in weeks
N.times = length(plot.times)

dose_pp1m = 150
dose_pp3m = 525

Q_cen_0_mean = 30   # geo mean
Q_cen_0_sd   = 1.5  # geo SD

## We need to recompile, because nsubjects must be a constant
constants = list(nsubjects_per_arm = N.subjects.v,
                 Do_fit            = 0,           # 0: no fit, > 0: index
                 SamEq3            = 1,           # Boolean
                 Duration_2        = 319,         # (h)
                 PP3M_start        = 17*7*24,     # (h)
                 ntimes            = N.times,
                 times             = times,
                 nstates           = 4)

data  = list()

inits = list(Q_cen_0_mean = Q_cen_0_mean, # (mg)
             Q_cen_0_sd   = Q_cen_0_sd,   # (mg)
             Dose_PP1M    = dose_pp1m[1], # (mg)
             Dose_PP3M    = dose_pp3m[1], # (mg)
             Delta        = rep(1,2))

Rmodel = nimbleModel(myNimbleCode, constants, data, inits, calculate=F)
## Cmodel = Rmodel
Cmodel = compileNimble(Rmodel, showCompilerOutput=F)

Node.names = Cmodel$getNodeNames(includeData=T)
Cmodel$simulate(nodes = Node.names)
res = values(Cmodel, Node.names)

## Main simulation: bioequivalent case
Cmodel$Delta = c(1, 1) # true BE
```

```
N.mtc.v = 1000 # number of simulated virtual trials
Cmax.g1_R = Cmax.g1_T = Cmax.g2_T = Cmax.g2_R =
AUC.g1_R  = AUC.g1_T  = AUC.g2_T  = AUC.g2_R  =
  matrix(0, nrow=N.mtc.v, ncol=N.subjects.v)
BE.yes = rep(-1, N.mtc.v)
itime  = 16:20  # sampling times in last dosing period for 1st PP3M formulation
jtime  = 30:34  # sampling times in last dosing period for 2nd PP3M formulation
t.start = Sys.time()
for (i in 1:N.mtc.v) {
  ## Simulate the trial
  Cmodel$simulate(nodes = Node.names)
  all.data = values(Cmodel, "C_plasma_obs")
  all.data = matrix(all.data, ncol=N.times, byrow = F) # subjects by row
  ##
  ## Group 1 (REFERENCE then TEST sequence) arm of trial
  g1.data.mat = all.data[1:N.subjects.v,]
  ##
  ## Group 2 (TEST then REFERENCE sequence) arm of trial
  g2.data.mat = all.data[(N.subjects.v + 1):(2 * N.subjects.v),]
  ##
  ## Compute Cmax for each subject in the last dosing period of PP3M and PP3Mt
  Cmax.g1_R[i,] = get.Cmax(t(g1.data.mat[,itime]))
  Cmax.g1_T[i,] = get.Cmax(t(g1.data.mat[,jtime]))
  Cmax.g2_T[i,] = get.Cmax(t(g2.data.mat[,itime]))
  Cmax.g2_R[i,] = get.Cmax(t(g2.data.mat[,jtime]))
  ##
  ## Compute AUC for each subject in the last dosing period of PP3M and PP3Mt
  AUC.g1_R[i,] = get.AUC(plot.times[itime], t(g1.data.mat[,itime]))
  AUC.g1_T[i,] = get.AUC(plot.times[jtime], t(g1.data.mat[,jtime]))
  AUC.g2_T[i,] = get.AUC(plot.times[itime], t(g2.data.mat[,itime]))
  AUC.g2_R[i,] = get.AUC(plot.times[jtime], t(g2.data.mat[,jtime]))
  ##
  print(paste("Trial", i))
}
t.end = Sys.time()
t.end - t.start

save(list=c("N.mtc.v", "N.subjects.v",
            "Cmax.g1_R", "Cmax.g1_T", "Cmax.g2_T", "Cmax.g2_R",
            "AUC.g1_R",  "AUC.g1_T",  "AUC.g2_T",  "AUC.g2_R"),
     file=paste0("Cmax & AUC for power",IDtag,".Rsave"))
## load(file=paste0("Cmax & AUC for power",IDtag,".Rsave"))

## Now process increasing chunks of the results
power = rep(0, N.subjects.v/2)
BE.yes = rep(0, N.mtc.v)
for (k in seq(2, N.subjects.v, 2)) {
  N.subj.current = k
  my.index = sample.int(N.subjects.v, N.subj.current) # randomize
  for (i in 1:N.mtc.v) {
    Cmax.yes = myTOST.2x2(Cmax.g1_R[i,my.index], Cmax.g1_T[i,my.index],
                          Cmax.g2_T[i,my.index], Cmax.g2_R[i,my.index])
    AUC.yes  = myTOST.2x2(AUC.g1_R[i,my.index], AUC.g1_T[i,my.index],
                          AUC.g2_T[i,my.index], AUC.g2_R[i,my.index])
    #
    BE.yes[i] = Cmax.yes && AUC.yes
  }
  ## power:
  power[k/2] = sum(BE.yes) / N.mtc.v # probability of declaring BE
}

## Plot
pdf("Power plot 2x2 crossover after calibration.pdf")
plot(seq(2, N.subjects.v, 2), power, type="l",
     xlab="Number of subjects per arm",
     ylab="Probability of declaring bioequivalence",
     xlim=c(1,N.subjects.v), ylim=c(0, 1), col="grey", lwd=0.5, las=1)
```

```
## smooth:
lines(seq(2, N.subjects.v, 2),
      supsmu(seq(2,N.subjects.v,2), power)$y, lwd=2, col="red")
dev.off()

## ========================================================================
## 7. Type I error: probability of declaring BE when it is not true.

## Compute once for many subjects and then use only part of the data
## for smaller trial sizes.

N.subjects.v = 500 # number of virtual subjects per arm

Hr1 = 1 / (24 * 7) # one hour in weeks
times = c(1, 5, 9, 13, 14, 17-Hr1,
          21, 25, 29-Hr1, 33, 37, 41-Hr1, 45, 49, 53-Hr1, 54, 55, 57, 61,
          65-Hr1, 69, 73, 77-Hr1, 81, 85, 89-Hr1, 93, 97, 101-Hr1, 102, 103,
          105, 109, 113)
times = times * 24 * 7
plot.times = times / (24 * 7) # in weeks
N.times = length(plot.times)

dose_pp1m = 150
dose_pp3m = 525

Q_cen_0_mean = 30   # geo mean
Q_cen_0_sd   = 1.5  # geo SD

## We need to recompile, because nsubjects must be a constant
constants = list(nsubjects_per_arm = N.subjects.v,
                 Do_fit            = 0,            # 0: no fit, > 0: index
                 SamEq3            = 1,            # Boolean
                 Duration_2        = 319,          # (h)
                 PP3M_start        = 17*7*24,      # (h)
                 ntimes            = N.times,
                 times             = times,
                 nstates           = 4)

data  = list()

inits = list(Q_cen_0_mean = Q_cen_0_mean, # (mg)
             Q_cen_0_sd   = Q_cen_0_sd,   # (mg)
             Dose_PP1M    = dose_pp1m[1], # (mg)
             Dose_PP3M    = dose_pp3m[1], # (mg)
             Delta        = rep(1,2))

Rmodel = nimbleModel(myNimbleCode, constants, data, inits, calculate=F)
Cmodel = compileNimble(Rmodel, showCompilerOutput=F)

Node.names = Cmodel$getNodeNames(includeData=T)
Cmodel$simulate(nodes = Node.names)
res = values(Cmodel, Node.names)

## Main simulation
N.mtc.v = 1000 # number of simulated virtual trials
BE.yes = rep(-1, N.mtc.v)
Delta    = matrix(0, nrow=N.mtc.v, ncol=2)
Cmax.g1_R = Cmax.g1_T = Cmax.g2_T = Cmax.g2_R =
AUC.g1_R  = AUC.g1_T  = AUC.g2_T  = AUC.g2_R  =
  matrix(0, nrow=N.mtc.v, ncol=N.subjects.v)
itime  = 16:20  # sampling times in last dosing period for 1st PP3M formulation
jtime  = 30:34  # sampling times in last dosing period for 2nd PP3M formulation
t.start = Sys.time()
for (i in 1:N.mtc.v) {
  ## Simulate the trial
  Delta[i,]    = runif(2, 0.7, 1.5)
  Cmodel$Delta = Delta[i,]
  Cmodel$simulate(nodes = Node.names)
  all.data = values(Cmodel, "C_plasma_obs")
  all.data = matrix(all.data, ncol=N.times, byrow = F) # subjects by row
  ##
  ## Group 1 (REFERENCE then TEST sequence) arm of trial
  g1.data.mat = all.data[1:N.subjects.v,]
  ##
  ## Group 2 (TEST then REFERENCE sequence) arm of trial
  g2.data.mat = all.data[(N.subjects.v + 1):(2 * N.subjects.v),]
  ##
  ## Compute Cmax for each subject in the last dosing period of PP3M and PP3Mt
  Cmax.g1_R[i,] = get.Cmax(t(g1.data.mat[,itime]))
  Cmax.g1_T[i,] = get.Cmax(t(g1.data.mat[,jtime]))
  Cmax.g2_T[i,] = get.Cmax(t(g2.data.mat[,itime]))
  Cmax.g2_R[i,] = get.Cmax(t(g2.data.mat[,jtime]))
  ##
  ## Compute AUC for each subject in the last dosing period of PP3M and PP3Mt
  AUC.g1_R[i,] = get.AUC(plot.times[itime], t(g1.data.mat[,itime]))
  AUC.g1_T[i,] = get.AUC(plot.times[jtime], t(g1.data.mat[,jtime]))
  AUC.g2_T[i,] = get.AUC(plot.times[itime], t(g2.data.mat[,itime]))
  AUC.g2_R[i,] = get.AUC(plot.times[jtime], t(g2.data.mat[,jtime]))
  ##
  print(paste("Trial", i))
}
t.end = Sys.time()
t.end - t.start

## Save
save(list=c("N.mtc.v", "N.subjects.v", "Delta",
            "Cmax.g1_R", "Cmax.g1_T", "Cmax.g2_T", "Cmax.g2_R",
            "AUC.g1_R",  "AUC.g1_T",  "AUC.g2_T",  "AUC.g2_R"),
     file=paste0("Cmax & AUC for type 1 error", IDtag,".Rsave"))
## load(file=paste0("Cmax & AUC for type 1 error",IDtag,".Rsave"))

## Set number of subjects to use; we can use at most N.subjects.v subjects
N.subject.used = N.subjects.v
j = 1:N.subject.used

## Test/Ref relative Cmax differences
tmp1 = exp(apply(log(Cmax.g1_T) - log(Cmax.g1_R),  MARGIN=1, FUN=mean))
tmp2 = exp(apply(log(Cmax.g2_T) - log(Cmax.g2_R),  MARGIN=1, FUN=mean))
Cmax.rel.diffs = apply(cbind(tmp1, tmp2),  MARGIN=1, FUN=mean)
mean(Cmax.rel.diffs)

## Test/Ref relative AUC differences
tmp1 = exp(apply(log(AUC.g1_T) - log(AUC.g1_R),  MARGIN=1, FUN=mean))
tmp2 = exp(apply(log(AUC.g2_T) - log(AUC.g2_R),  MARGIN=1, FUN=mean))
AUC.rel.diffs = apply(cbind(tmp1, tmp2),  MARGIN=1, FUN=mean)
mean(AUC.rel.diffs)

## Compute BE for the various trials
Cmax.yes = AUC.yes = BE.yes = rep(0, N.mtc.v)
for (i in 1:N.mtc.v) {
  Cmax.yes[i] = myTOST.2x2(Cmax.g1_R[i,], Cmax.g1_T[i,],
                           Cmax.g2_T[i,], Cmax.g2_R[i,])
  AUC.yes[i]  = myTOST.2x2(AUC.g1_R[i,], AUC.g1_T[i,],
                           AUC.g2_T[i,], AUC.g2_R[i,])
  BE.yes[i]   = Cmax.yes[i] && AUC.yes[i]
}

## Plot passes
pdf("Type 1 error plot 2x2 crossover.pdf")
## par(mfrow=c(2,1), mar=c(4,5,1,1), las=1, cex.lab=0.8)
par(mar=c(8,5,7,1), las=1, cex.lab=1.2)
plot(Cmax.rel.diffs, Cmax.yes, type="p",
     xlab="Cmax relative differences geometric mean",
     ylab="Probability of declaring Cmax BE")
abline(v=c(0.8, 1.25), col="red")
## lines(Cmax.centers, Cmax.error1, col="blue")
## lines(Cmax.centers, supsmu(1:(N.bins-1), Cmax.error1,span=0.03)$y,
##       col="blue")
##
plot(AUC.rel.diffs, AUC.yes, type="p",
     xlab="AUC relative differences geometric mean",
     ylab="Probability of declaring AUC BE")
abline(v=c(0.8, 1.25), col="red")
## lines(AUC.centers, AUC.error1, col="blue")
dev.off()

## =========================================================================
## 8. Safe space: probability of declaring BE when it is not true.
##    We have saved the Delta and BE decisions in the type 1 error
##    calculations; we can just reuse them.

## The saved simulation are the same as type 1 error but a copy has been made.
## load(file=paste0("Cmax & AUC for type 1 error",IDtag,".Rsave"))
load(file=paste0("Cmax & AUC for safe space",IDtag,".Rsave"))

## Set number of subjects to use; we can use at most N.subjects.v subjects
N.subject.used = N.subjects.v
j = 1:N.subject.used

## Test/Ref relative Cmax differences
tmp1 = exp(apply(log(Cmax.g1_T) - log(Cmax.g1_R),  MARGIN=1, FUN=mean))
tmp2 = exp(apply(log(Cmax.g2_T) - log(Cmax.g2_R),  MARGIN=1, FUN=mean))
Cmax.rel.diffs = apply(cbind(tmp1, tmp2),  MARGIN=1, FUN=mean)
mean(Cmax.rel.diffs)

## Test/Ref relative AUC differences
tmp1 = exp(apply(log(AUC.g1_T) - log(AUC.g1_R),  MARGIN=1, FUN=mean))
tmp2 = exp(apply(log(AUC.g2_T) - log(AUC.g2_R),  MARGIN=1, FUN=mean))
AUC.rel.diffs = apply(cbind(tmp1, tmp2),  MARGIN=1, FUN=mean)
mean(AUC.rel.diffs)

## Compute BE for the various trials
## Compute BE for the various trials
Cmax.yes = AUC.yes = BE.yes = rep(0, N.mtc.v)
for (i in 1:N.mtc.v) {
  Cmax.yes[i] = myTOST.2x2(Cmax.g1_R[i,], Cmax.g1_T[i,],
                           Cmax.g2_T[i,], Cmax.g2_R[i,])
  AUC.yes[i]  = myTOST.2x2(AUC.g1_R[i,], AUC.g1_T[i,],
                           AUC.g2_T[i,], AUC.g2_R[i,])
  BE.yes[i]   = Cmax.yes[i] && AUC.yes[i]
}

pdf("Safe space delta[1,2] plot 2x2 crossover.pdf")
is.BE = which(BE.yes == 1) # we could also use Cmax.yes or AUC.yes
par(mar=c(5,5,2,1))
plot(Delta[,1], Delta[,2], type="n", cex.lab=1.4, las=1,
     xlab=expression(delta[1]),
     ylab=expression(delta[2]))
points(Delta[ is.BE,1], Delta[ is.BE,2], pch=16, cex=0.5, col="green")
points(Delta[-is.BE,1], Delta[-is.BE,2], pch=16, cex=0.8, col="red")
abline(a=1.23, b=-0.5, col="red")
abline(a=1.83, b=-0.5, col="red")
abline(a=1.14, b=-0.5, col="green")
abline(a=2.09, b=-0.5, col="green")
dev.off()

## End.
```

TOST code in R

```
## v1: TOST for parallel trials
## v2: Add TOST for 2x2 (RT/TR) crossover trials

## TOST test function: takes ref and test PK parameters (Cmax...) and
## returns TRUE if they are bioequivalent
myTOST = function(X.ref, X.test) {
  N = length(X.ref)
  ##
  ## means in log space
  mu.ref  = mean(log(X.ref))
  mu.test = mean(log(X.test))
  ##
  ## variances in log space
  var.ref  = var(log(X.ref))
  var.test = var(log(X.test))
  ##
  logmeans.diff = mu.test - mu.ref
  logvars.mean  = (var.ref + var.test) / 2
  ##
  ## delta = t.variate * SE
  delta = qt(0.95, df = 2*N - 2) * sqrt(logvars.mean * 2 / N)
  CL_lo = logmeans.diff - delta
  CL_up = logmeans.diff + delta
  ##
  X.yes = (exp(CL_lo) > 0.8) && (exp(CL_up) < 1.25) # true or false...
  return(X.yes)
}

## TOST test function for cross-over trials
## returns TRUE if they are bioequivalent
myTOST.2x2 = function(X.g1_r, X.g1_t, X.g2_t, X.g2_r) {
  ## parameters are group 1 ref, then test then group 2 ref then test results
  ## assumes completely balanced design
  ## log-transform and contatenate the data
  y = log(c(X.g1_r, X.g1_t, X.g2_t, X.g2_r))
  ##
  Nsub.in_seq = length(X.g1_r)  # number of subjects per sequence
  Nsub.total  = Nsub.in_seq * 2 # total number of subjects
  Nobs.total  = Nsub.in_seq * 4 # number of observations
  ##
  ## Recode the design variables
  X = matrix(0, Nobs.total, 4)
  X[,1] = 1 # Intercept
  X[,2] = rep(c(0,1,1,0), each=Nsub.in_seq) # treatment
  X[,3] = rep(0:1, each=Nsub.total)         # sequence
  X[,4] = rep(1:0, each=Nsub.in_seq)
  ##
  b = solve(t(X)%*%X)%*%t(X)%*%y
  ##
  res = y - X%*%b # residuals
  ##
  ## Compute within subject variance
  my.i = 1:Nsub.in_seq
  res.delta = c(res[my.i +   Nsub.in_seq] - res[my.i],
                res[my.i + 3*Nsub.in_seq] - res[my.i + 2*Nsub.in_seq])
  var.within = (sum(res.delta * res.delta) / 2) / (Nsub.total - 2)
  ##
  SD.within = sqrt(var.within) # within subject standard deviation
  ## Compute total variance
  var.total = mean(res * res) * Nsub.total / (Nsub.total - 2)
  ## Compute between subject variance
  var.between = var.total - var.within
  ##
  SE.Form = SD.within * sqrt(1 / Nsub.in_seq)
  ##
  dof = (Nobs.total - 4) / 2
  ##
  tfactor = qt(0.95, dof)
  CL_up = b[2] + tfactor * SE.Form
  CL_lo = b[2] - tfactor * SE.Form
  ##
  X.yes = (exp(CL_lo) > 0.8) && (exp(CL_up) < 1.25) # true or false...
}

## End.
```

C_{max} and AUC data-based calculations code in R

```
## Compute Cmax and AUC given a vector of times and a matrix of concentration
## measurements at some times for different subjects (times in row, subjects
## in columns)

## Compute Cmax for different subjects
get.Cmax = function(conc) {
  return(apply(conc, MARGIN=2, FUN=max))
}

## Compute AUC for each subject
get.AUC = function(times, conc) {
  Ns  = dim(conc)[2]
  AUC = rep(0, Ns)
  Nt  = length(times)
  dx  = diff(times)
  for (i in 1:Ns) {
    AUC[i] = sum((conc[-Nt,i] + conc[-1,i]) * dx) / 2
  }
  return(AUC)
}
## End.
```

Structural model C code (v6)

```
/* compile within R with system("R CMD SHLIB PP3M_model.c")
   V05: Compute central concentration AUC by integration
*/
#include <R.h>

#define Nparms 12

static double parms[Nparms];

/* A trick to keep up with the parameters */
#define Dose_PP1M     parms[0]
#define F2            parms[1]
#define Duration_2    parms[2]
#define ka_PP1M       parms[3]
#define ka1_max       parms[4]
#define ka3_max       parms[5]
#define kamt1_50      parms[6]
#define kamt3_50      parms[7]
#define gamma         parms[8]
#define CL            parms[9]
#define V             parms[10]
#define PP3M_start    parms[11]

/* initializer: same name as the dll (without extension) */
void PP13M_model_v06(void (* odeparms)(int *, double *))
{
  int N = Nparms;
  odeparms(&N, parms);
}

/* Derivatives */
void derivs(int *neq, double *t, double *y, double *ydot, double *yout, int*ip)
{
  double Ke;

  // State variables
  // Q_depot_s1  = y[0] # quantity (mg) in PP1M slow absorption depot
  // Q_depot_s3  = y[1] # quantity (mg) in PP3M slow absorption depot
  // Q_depot_r3  = y[2] # quantity (mg) in PP3M fast absorption depot
  // Q_central   = y[3] # quantity (mg) in central compartment
  // AUC_central = y[4] # integral of y[3]

  // ODEs
  // Quantity in PP1M depot slow absorption
  ydot[0] = -ka_PP1M * y[0];

  // Quantities in PP3M depots
  if (*t < PP3M_start) { // use PP1M model, PP3M model differentials are null
    // Quantity in PP3M depot slow absorption
    ydot[1] = 0;
    // Quantity in PP3M depot rapid absorption
    ydot[2] = 0;
  }
  else { // use PP1M and PP3M models concurrently
    // Quantity in PP3M depot slow absorption
    ydot[1] = -ka1_max * pow(y[1], gamma) /
               (pow(kamt1_50, gamma) + pow(y[1], gamma));
    // Quantity in PP3M depot rapid absorption
    ydot[2] = -ka3_max * y[2] / (kamt3_50 + y[2]);
  }

  // Quantity in central compartment
  // clearance from central
  Ke = CL / V;
  // hard-code the zero-order inputs after PP1M injections
  if (((0 <= *t)    && (*t < 319))  || ((672 <= *t)  && (*t < 991)) ||
     ((1344 <= *t) && (*t < 1663)) || ((2016 <= *t) && (*t < 2335))) {
    ydot[3] = F2 * Dose_PP1M / Duration_2
              -(ydot[0] + ydot[1] + ydot[2]) - Ke * y[3];
  }
  else {
    ydot[3] = -(ydot[0] + ydot[1] + ydot[2]) - Ke * y[3];
  }

  // Q central AUC
  ydot[4] = y[3];
}

/* End */
```

Statistical model in R (v16)

```
## R/Nimble code for PP1M/PP3M population model, from Magnusson
## v14: Sample Delta from its posterior to abbreviated trial distribution.
##      Posterior of log(Delta[2]) is well approximated by a normal.
## v15: Uses C model v5 which calculates AUC by integration.
## v16: Reset AUC_central at last dosing time.

library(nimble)

## compile C ODE model for deSolve
Cmodel.name = "PP13M_model_v06"
## system(paste0("R CMD SHLIB ", Cmodel.name, ".c"))
dyn.load(paste(Cmodel.name, .Platform$dynlib.ext, sep = ""))
## dyn.unload(paste(Cmodel.name, .Platform$dynlib.ext, sep = ""))


## ============================================================================
## Define the model

## ----------------------------------------------------------------------------
## ODE solver: performs a simulation, given initial state values,
## output times and time-constant, scaled, parameters
## This is an R function
```

```
R_ode = function(y, times, parms) {
  ## parms: 1:Dose_PP1M, 2:F2,         3:Duration_2,  4:ka_PP1M,
  ##        5:ka1_max,   6:ka3_max,    7:kamt1_50,    8:kamt3_50,
  ##        9:gamma,    10:CL,        11:V,          12:PP3M_start,
  ##       13:F3,       14:Dose_PP3M, 15:Q_cen_0
  ## The last two parameters should not be passed to the ODE solver, they are
  ## used here only.
  ## State variables (y) initial conditions
  y = c("Q_depot_s1"  = 0,
        "Q_depot_s3"  = 0,
        "Q_depot_r3"  = 0,
        "Q_central"   = parms[15],
        "AUC_central" = 0)
  ## The doses are specified as "events" affecting the state variables
  ## dosing times (hours)
  dose_times = c(c(0, 4, 8, 12) + parms[3] / (24 * 7),
                 17, 29, 41, 53) * 24 * 7
  N_doses = length(dose_times)
  ## changing state variables at dosing times
  vars = c("Q_depot_s1", "Q_depot_s3", "Q_depot_r3")
  v1 = (1 -  parms[2])  * parms[1]     # value applied to Q_depot_s1
  v2 = (1 -  parms[13]) * parms[14]    # value applied to Q_depot_s3
  v3 =       parms[13]  * parms[14]    # value applied to Q_depot_r3
  ## Form the events table
  eventdat = data.frame(var = c(rep("Q_depot_s1", 4), rep(vars[2:3], 4),
                                "AUC_central"),
                        time = c(dose_times[1:4], rep(dose_times[5:8], each=2),
                                 dose_times[8]),
                        value = c(rep(v1, 4),
                                  rep(c(v2, v3), N_doses - 4), 0),
                        method = c(rep("add", 12), "replace"))
  ## Integrate numerically, with outputs at specified times
  results = deSolve::lsode(y, times, func="derivs", parms=parms[1:12],
                           rtol=1e-6, atol=1e-6, dllname="PP13M_model_v06",
                           initfunc = "PP13M_model_v06",
                           events=list(data=eventdat))
  ## results = result[which(results[,1] %in% times),] # weed out extra times
  nrow.res = dim(results)[[1]]
  if (results[nrow.res,1] < times[length(times)]) { ## integration failed
    return(rep(1E-30, 2))
  } else {
    ## central concentration AUC is computed over last dosing period,
    ## convert from hours * mg/L to week * ng/ml, return
    from = max(which(results[,1] == dose_times[N_doses]))
    Cmax = max(results[from:nrow.res,5])
    AUC = results[nrow.res,6] * 1E3 / parms[11] / 168
    return(as.numeric(c(Cmax, AUC)))
  }
} # end of R_ode model solver


## -------------------------------------------------------------------------
## Nimble function with nimbleRcall. This is just a wrapper
## to call the R-coded "R_ode" model solver from inside the "myNimbleCode" core
## function (defined next). It has to match the format of the "R_ode" function.
nimble_ode = nimbleRcall(
  prototype = function(
    y     = double(1), # vector
    times = double(1), # vector
    parms = double(1)  # vector
  ) {},
  returnType = double(1), # outcome is a vector
  Rfun = 'R_ode'
)

## -------------------------------------------------------------------------
## Hierarchical core Nimble (BUGS) code
myNimbleCode = nimbleCode({ ## BUGS (extended) code
  ##
  ## REFERENCE group population mean (with prior if not fixed)
  F2t_m       <- logit(0.168) # From Samtani paper Table III.
  # from online resource 3 Magnusson, F2 = 0.153
  F3t_m       <- logit(0.209) # transformed F3
  ka1_pp1m_mv <- log(1 + 0.02^2)
  ka1_pp1m_m  ~  dlnorm(meanlog=log(4.88E-4), varlog=ka1_pp1m_mv) # (1/h)
  ka1_max_mv  <- log(1 + 0.0696^2)
  ka1_max_m   ~  dlnorm(meanlog=log(0.0904),  varlog=ka1_max_mv)  # (mg/h)
  ka3_max_mv  <- log(1 + 0.0465^2)
  ka3_max_m   ~  dlnorm(meanlog=log(0.164),   varlog=ka3_max_mv)  # (mg/h)
  kamt1_50_mv <- log(1 + 0.0383^2)
  kamt1_50_m  ~  dlnorm(meanlog=log(120),     varlog=kamt1_50_mv) # (mg)
  kamt3_50_mv <- log(1 + 0.0952^2)
  kamt3_50_m  ~  dlnorm(meanlog=log(21.4),    varlog=kamt3_50_mv) # (mg)
  gamma_mv    <- log(1 + 0.0165^2)
  gamma_m     ~  dlnorm(meanlog=log(1.44),    varlog=gamma_mv)    # unitless
  CL_mv       <- log(1 + 0.0216^2)
  CL_m        ~  dlnorm(meanlog=log(3.84),    varlog=CL_mv)       # (L/hr)
  V_m         <- 1960   # (L)
  ##
  ## REFERENCE pop (inter-individual) SDs (with prior if not fixed)
  ##
  ## F2_sd
  omega2    <- 0.064 / (0.168 * (1 - 0.168)) # Samtani eq. 4 inverted
  F2_sd     ~  dnorm(mean = omega2, sd = 0.02*omega2)
  F2_v      <- F2_sd^2
  ## F3_v, Samtani eq. 4 inverted
  F3_v      <- (0.854 / (1 - 0.209))^2
  ##
  ka1_pp1m_cv_v <- log(1 + 0.03^2)
  ka1_pp1m_cv   ~  dlnorm(meanlog=log(0.590), varlog=ka1_pp1m_cv_v)
  ka1_pp1m_v    <- log(1 + ka1_pp1m_cv^2)
  ##
  ka1_max_cv_v  <- log(1 + 0.0501^2)
  ka1_max_cv    ~  dlnorm(meanlog=log(0.827), varlog=ka1_max_cv_v)
  ka1_max_v     <- log(1 + ka1_max_cv^2)
  ##
  ka3_max_v     <- 0
  ##
  kamt1_50_cv_v <- log(1 + 0.101^2)
  kamt1_50_cv   ~  dlnorm(meanlog=log(0.500), varlog=kamt1_50_cv_v)
  kamt1_50_v    <- log(1 + kamt1_50_cv^2)
  ##
  kamt3_50_cv_v <- log(1 + 0.142^2)
  kamt3_50_cv   ~  dlnorm(meanlog=log(0.867), varlog=kamt3_50_cv_v)
  kamt3_50_v    <- log(1 + kamt3_50_cv^2)
  ##
  gamma_v       <- 0
  ##
  CL_cv_v       <- log(1 + 0.0317^2)
  CL_cv         ~  dlnorm(meanlog=log(0.357), varlog=CL_cv_v)
  CL_v          <- log(1 + CL_cv^2)
  ##
  V_v           <- log(1 + 0.628^2)
  ##
  ## TEST group population mean (with prior if not fixed)
  ## Delta is a vector containing the factor of difference between the
  ## ref and test compounds for the 6 drug-release  parameters
  ## 1 = no difference.
  ## Order in Delta: f_3, k_as3,max, k_ar3,max, k_as3,50, k_ar3,50, Gamma.
  ## If Do_fit > 0.5, sample the first 3 Delta.
  if (Do_fit > 0.5) {
    logDelta2 ~  dnorm(mean=logmeanD2, var=logvarD2)
    Delta[2]  <- exp(logDelta2)
  }
  else {
    logDelta2 ~  dnorm(mean=0.34882, sd=0.17475) # abb. trial posterior
    Delta[2]  <- exp(logDelta2)
  }
  F3t_m_T      <- logit(0.209 * Delta[1]) # transformed F3 with delta
  ka1_max_m_T  <- ka1_max_m   * Delta[2]  # (mg/h)
  ka3_max_m_T  <- ka3_max_m   * Delta[3]  # (mg/h)
  kamt1_50_m_T <- kamt1_50_m  * Delta[4]  # (mg)
  kamt3_50_m_T <- kamt3_50_m  * Delta[5]  # (mg)
  gamma_m_T    <- gamma_m     * Delta[6]  # unitless
  ##
  ## if F3_m is modified then F3_v is modified:
  if (SamEq3) { # see above
     F3_v_T    <- (0.854 / (1 - 0.209 * Delta[1]))^2
  } else {
     F3_v_T    <- (abs(F3t_m_T) * 0.854)^2
  }
  ##
  ## measurement error variance in log for plasma concentration, ng/ml
  ## if we want uncertainty on the residual error we should use:
  res_cv_v    <- log(1 + 0.321^2)
  res_cv      ~  dlnorm(meanlog=log(0.306), varlog=res_cv_v)
  sigma2      <- log(1 + res_cv^2)
  ## simpler is:
  ## sigma2 <- log(1 + 0.306^2)

  ## for each REFERENCE subject
  for (i in 1:nsubjects_per_arm) {
    tmp2[i]     ~  dnorm(mean = F2t_m, var = F2_v)
    F2[i]       <- ilogit(tmp2[i])
    tmp3[i]     ~  dnorm(mean = F3t_m, var = F3_v)
    F3[i]       <- ilogit(tmp3[i])
    ka_PP1M[i]  ~  dlnorm(meanlog=log(ka1_pp1m_m), varlog=ka1_pp1m_v)  # (1/hr)
    ka1_max[i]  ~  dlnorm(meanlog=log(ka1_max_m),  varlog=ka1_max_v)   # (mg/h)
    ka3_max[i]  <- ka3_max_m                                           # (mg/h)
    kamt1_50[i] ~  dlnorm(meanlog=log(kamt1_50_m), varlog=kamt1_50_v)  # (mg)
    kamt3_50[i] ~  dlnorm(meanlog=log(kamt3_50_m), varlog=kamt3_50_v)  # (mg)
    gamma[i]    <- gamma_m
    CL[i]       ~  dlnorm(meanlog=log(CL_m),       varlog=CL_v)        # (L/hr)
    V[i]        ~  dlnorm(meanlog=log(V_m),        varlog=V_v)         # (L)
    ##
    ## quantity at t 0
    Q_cen_0[i]  ~  dlnorm(meanlog=log(Q_cen_0_mean), sdlog = log(Q_cen_0_sd))
    ##
    ## Call the ODE solver to get AUC_central for each subject
    ## prediction is plasma concentration AUC, in week * ng/ml
    Cmax_AUC[i,1:2] <- nimble_ode(y[1:nstates], times[1:ntimes],
                                  c(Dose_PP1M, F2[i], Duration_2,
                                    ka_PP1M[i], ka1_max[i], ka3_max[i],
                                    kamt1_50[i], kamt3_50[i], gamma[i],
                                    CL[i], V[i], PP3M_start, F3[i],
                                    Dose_PP3M, Q_cen_0[i]))
  }

  ## for each TEST subject
  for (i in (1+nsubjects_per_arm):(2*nsubjects_per_arm)) {
    tmp2[i]     ~  dnorm(mean = F2t_m, var = F2_v)
    F2[i]       <- ilogit(tmp2[i])
    tmp3[i]     ~  dnorm(mean = F3t_m_T, var = F3_v_T)
    F3[i]       <- ilogit(tmp3[i])
    ka_PP1M[i]  ~  dlnorm(meanlog=log(ka1_pp1m_m),   varlog=ka1_pp1m_v) # (1/hr)
    ka1_max[i]  ~  dlnorm(meanlog=log(ka1_max_m_T),  varlog=ka1_max_v)  # (mg/h)
    ka3_max[i]  <- ka3_max_m_T                                          # (mg/h)
    kamt1_50[i] ~  dlnorm(meanlog=log(kamt1_50_m_T), varlog=kamt1_50_v) # (mg)
    kamt3_50[i] ~  dlnorm(meanlog=log(kamt3_50_m_T), varlog=kamt3_50_v) # (mg)
    gamma[i]    <- gamma_m_T
    CL[i]       ~  dlnorm(meanlog=log(CL_m),         varlog=CL_v)       # (L/hr)
    V[i]        ~  dlnorm(meanlog=log(V_m),          varlog=V_v)        # (L)
    ##
    ## quantity at t=0
```

```
    Q_cen_0[i]  ~  dlnorm(meanlog=log(Q_cen_0_mean), sdlog = log(Q_cen_0_sd))
    ##
    ## Call the ODE solver to get AUC_central for each subject
    ## prediction is plasma concentration AUC, in week * ng/ml
    Cmax_AUC[i,1:2] <- nimble_ode(y[1:nstates], times[1:ntimes],
                                  c(Dose_PP1M, F2[i], Duration_2,
                                    ka_PP1M[i], ka1_max[i], ka3_max[i],
                                    kamt1_50[i], kamt3_50[i], gamma[i],
                                    CL[i], V[i], PP3M_start, F3[i],
                                    Dose_PP3M, Q_cen_0[i]))
  }

}) # End myNimbleCode

## End.
```

Fully Bayesian workflow in R (v3)

```
## Fully Bayesian model-based VBE
## v1: Based on paliperidone palmitate VBE workflow v4.
##     Uses statistical model v14, Cmax and AUC are data-based.
## v2: Uses statistical model v15, which calculates AUC over the last
##     dosing period by integration.
## v3: Uses statistical model v16, which calculates AUC over the last
##     dosing period by integration an calculates Cmax with many time points.

IDtag = "_3" # version number

## ==========================================================================
## Compile and run the model for many trial of many subjects,
## get Cmax and AUC.

## Parallelize
library(parallel)
N.cores = 5 # detectCores()
this_cluster = makeCluster(N.cores)

## Create a function with all the needed code
run_allcode <- function(seed) {
  ##
  library(nimble)
  source("Statistical model v16.R")
  ## source("Cmax_AUC.R")
  ##
  N.subjects.v = 1000 # number of virtual subjects per arm
  ##
  Hr1 = 1 / (24 * 7) # one hour in weeks
  times = c(0, seq(53, 65, (65-53)/100)) * 168 # (hours)
  N.times = length(times)
  ##
  dose_pp1m = 150
  dose_pp3m = 525
  ##
  Q_cen_0_mean = 30  # geo mean
  Q_cen_0_sd   = 1.5 # geo SD
  ##
  data  = list()
  ##
  inits = list(Q_cen_0_mean = Q_cen_0_mean, # (mg)
               Q_cen_0_sd   = Q_cen_0_sd,   # (mg)
               Dose_PP1M    = dose_pp1m[1], # (mg)
               Dose_PP3M    = dose_pp3m[1], # (mg)
               Delta        = rep(1, 6))
  ##
  constants = list(nsubjects_per_arm = N.subjects.v,
                   Do_fit     = 0,            # 0: no Delta fit, > 0: fit
                   SamEq3     = 1,            # Boolean, leave at 1
                   Duration_2 = 319,          # (h)
                   PP3M_start = 17*7*24,      # (h)
                   ntimes     = length(times),
                   times      = times,
                   nstates    = 5)
  ##
  Rmodel = nimbleModel(myNimbleCode, constants, data, inits, calculate=F)
  ## Cmodel = Rmodel
  Cmodel = compileNimble(Rmodel, showCompilerOutput=F)
  ##
  N.mtc.v  = 200 # number of simulated virtual trials
  Node.names = Cmodel$getNodeNames(includeData=T)
  Delta.Cmax = Delta.AUC = rep(0, N.mtc.v)
  ##
  for (i in 1:N.mtc.v) {
    ## Simulate the trial
    Cmodel$simulate(nodes = Node.names)
    ## all.data = values(Cmodel, "C_plasma_obs")
    ## all.data = matrix(all.data, ncol=N.times, byrow = F) # subjects by row
    all.res = values(Cmodel, "Cmax_AUC")
    ##
    ## Cmax values
    Cmaxs      = all.res[1:(2*N.subjects.v)]
    ref.Cmaxs  = Cmaxs[1:N.subjects.v] # reference arm of the trial
    test.Cmaxs = Cmaxs[(N.subjects.v + 1):(2 * N.subjects.v)] # test arm
    ##
    Cmax.ref.mean  = mean(log(ref.Cmaxs))
    Cmax.ref.sd    = sd  (log(ref.Cmaxs))
    Cmax.test.mean = mean(log(test.Cmaxs))
    Cmax.test.sd   = sd  (log(test.Cmaxs))
    Delta.Cmax[i]  = Cmax.test.mean - Cmax.ref.mean
    ##
    ## AUC values
    AUCs       = all.res[(2*N.subjects.v + 1):(4*N.subjects.v)]
    ref.AUCs   = AUCs[1:N.subjects.v] # reference arm of the trial
    test.AUCs  = AUCs[(N.subjects.v + 1):(2 * N.subjects.v)] # test arm
    ##
    AUC.ref.mean   = mean(log(ref.AUCs))
    AUC.ref.sd     = sd  (log(ref.AUCs))
    AUC.test.mean  = mean(log(test.AUCs))
    AUC.test.sd    = sd  (log(test.AUCs))
    Delta.AUC[i]   = AUC.test.mean - AUC.ref.mean
  }
  ##
  save(list=c("N.mtc.v", "N.subjects.v", "Delta.Cmax", "Delta.AUC",
              "Cmax.ref.mean", "Cmax.test.mean", "Cmax.ref.sd", "Cmax.test.sd",
              "AUC.ref.mean",  "AUC.test.mean",  "AUC.ref.sd",  "AUC.test.sd"),
       file=paste0("Delta Cmax & AUC model based_", seed,".Rsave"))
  return(1)
  ##
} ## End run_allcode

t.start = Sys.time()
chain_output <- parLapply(cl=this_cluster, X=1:N.cores, fun=run_allcode)
t.end = Sys.time()
t.end - t.start

## Close the cluster when you're done with it.
stopCluster(this_cluster)

## Post processing
## Compute probabilities of declaring BE for various trial sizes
for (j in 1:N.cores) { # for each Monte Carlo block
  ##
  load(file=paste0("Delta Cmax & AUC model based_", j,".Rsave"))
  ##
  if (j == 1) {
    Delta.Cmax.all = Delta.Cmax
    Delta.AUC.all  = Delta.AUC
  } else {
    Delta.Cmax.all  = c(Delta.Cmax.all, Delta.Cmax)
    Delta.AUC.all   = c(Delta.AUC.all,  Delta.AUC)
  }
}
length(Delta.Cmax.all)

Ratio.Cmax = exp(Delta.Cmax.all)
Ratio.AUC  = exp(Delta.AUC.all)

## Plot Delta Cmax distribution
xlims = c(0.6, 1.6)
pdf(paste0("Delta Cmax distribution", IDtag,".pdf"))
hist(Ratio.Cmax, breaks=10, axes=F, xlim=xlims,
     main="", xlab="Cmax geometric mean test to reference ratio")
rect(min(xlims), 0, 0.8,        N.mtc.v, dens=10, col="red")
rect(1.25,       0, max(xlims), N.mtc.v, dens=10, col="red")
axis(1)
P.BE = (length(which((Ratio.Cmax <= 0.8))) +
        length(which((Ratio.Cmax >= 1.25)))) / (N.cores * N.mtc.v)
legend(0.8, N.mtc.v * 1.1,
       substitute(bar(P[BE]) == list(x), list(x = P.BE)), bty="n")
dev.off()

mean(Ratio.Cmax)
sd(Ratio.Cmax)

## Plot Delta AUC distribution
xlims = c(0.6, 1.6)
pdf(paste0("Delta AUC distribution", IDtag,".pdf"))
hist(Ratio.AUC, breaks=10, axes=F, xlim=xlims,
     main="", xlab="AUC geometric mean test to reference ratio")
rect(min(xlims), 0, 0.8,        N.mtc.v, dens=10, col="red")
rect(1.25,       0, max(xlims), N.mtc.v, dens=10, col="red")
axis(1)
P.BE = (length(which((Ratio.AUC <= 0.8))) +
        length(which((Ratio.AUC >= 1.25)))) / (N.cores * N.mtc.v)
legend(0.8, N.mtc.v * 1.1,
       substitute(bar(P[BE]) == list(x), list(x = P.BE)), bty="n")
dev.off()

mean(Ratio.AUC)
sd(Ratio.AUC)

## Correlation plot
pdf("Correlation Ratio Cmax - Ratio AUC.pdf")
par(mar=c(5,5,2,2))
xlims = c(0.7, 1.6)
plot(Ratio.Cmax, Ratio.AUC, las=1, xlim=xlims, ylim=xlims, type="n",
     xlab="", ylab="")
rect(0,    0,    0.8, 2,   lty=0, col="lightpink")
rect(1.25, 0,    2,   2,   lty=0, col="lightpink")
rect(0,    0,    2,   0.8, lty=0, col="lightpink")
rect(0,    1.25, 2,   2,   lty=0, col="lightpink")
## rect(0,    0,    0.8, 2,   dens=10, col="red")
## rect(1.25, 0,    2,   2,   dens=10, col="red")
## rect(0,    0,    2,   0.8, dens=10, col="red")
## rect(0,    1.25, 2,   2,   dens=10, col="red")
par(new=T)
plot(Ratio.Cmax, Ratio.AUC, las=1, xlim=xlims, ylim=xlims,
     xlab=expression(delta[C[max]]), ylab=expression(delta[C[AUC]]))
dev.off()

## End.
```

Fully Bayesian safe-space calculations in R

```
## Fully Bayesian model-based VBE, safe-space calculations
## v1: based on workflow v4.
## v2: try to go faster by starting from non-BE

IDtag = "_2" # version number
IDrun = 6    # run number

## ==========================================================================
## Compile and run the model for many trial of many subjects,
## get Cmax and AUC.

## Parallelize
library(parallel)
N.cores = 8 # detectCores()
this_cluster = makeCluster(N.cores)

## Create a function with all the needed code
run_allcode <- function(seed, N.cores, IDrun) {
  ##
  Delta1.vals     = seq(1.5, 0.8, -0.2/7)[9:22]
  Delta2.vals     = seq(1.5, 1, -0.01)
  ##
  library(nimble)
  source("Statistical model v16.R")
  ##
  N.subjects.v = 1000 # number of virtual subjects per arm
  ##
  Hr1 = 1 / (24 * 7) # one hour in weeks
  times = c(0, seq(53, 65, (65-53)/100)) * 168 # (hours)
  N.times = length(times)
  ##
  Q_cen_0_mean = 30  # geo mean
  Q_cen_0_sd   = 1.5 # geo SD
  ##
  dose_pp1m = 150
  dose_pp3m = 525
  ##
  data  = list()
  ##
  inits = list(Q_cen_0_mean = Q_cen_0_mean, # (mg)
               Q_cen_0_sd   = Q_cen_0_sd,   # (mg)
               Dose_PP1M    = dose_pp1m[1], # (mg)
               Dose_PP3M    = dose_pp3m[1], # (mg)
               Delta        = rep(1,6))
  ##
  constants = list(nsubjects_per_arm = N.subjects.v,
                   Do_fit     = 0,            # 0: no Delta fit, > 0: fit
                   SamEq3     = 1,            # Boolean, leave at 1
                   Duration_2 = 319,          # (h)
                   PP3M_start = 17*7*24,      # (h)
                   ntimes     = length(times),
                   times      = times,
                   nstates    = 5)
  ##
  Rmodel = nimbleModel(myNimbleCode, constants, data, inits, calculate=F)
  ## Cmodel = Rmodel
  Cmodel = compileNimble(Rmodel, showCompilerOutput=F)
  ##
  Node.names = Cmodel$getNodeNames(includeData=T)
  ##
  Cmodel$Delta[1] = Delta1.vals[seed]
  index.D2        = which(Node.names == "Delta[2]")
  N.delta2        = length(Delta2.vals)
  Delta           = matrix(0, nrow=N.delta2, ncol=6)
  ##
  BE = rep(NA, N.delta2)
  N.mtc.v  = 1000 # number of simulated virtual trials
  stop_after_this_trial = FALSE
  ##
  for (j in 1:N.delta2) {
    ##
    Cmodel$Delta[2] = Delta2.vals[j]
    Delta[j,] = Cmodel$Delta
    N.fail = 0
    for (i in 1:N.mtc.v) {
      ## Simulate the trial
      Cmodel$simulate(nodes = Node.names[-index.D2]) # Delta[2] not sampled
      all.res = values(Cmodel, "Cmax_AUC")
      ##
      ## Cmax values
      Cmaxs      = all.res[1:(2*N.subjects.v)]
      ref.Cmaxs  = Cmaxs[1:N.subjects.v] # reference arm of the trial
      test.Cmaxs = Cmaxs[(N.subjects.v + 1):(2 * N.subjects.v)] # test arm
      ##
      Cmax.ref.mean  = mean(log(ref.Cmaxs))
      Cmax.ref.sd    = sd  (log(ref.Cmaxs))
      Cmax.test.mean = mean(log(test.Cmaxs))
      Cmax.test.sd   = sd  (log(test.Cmaxs))
      Delta.Cmax     = Cmax.test.mean - Cmax.ref.mean
      ##
      ## AUC values
      AUCs       = all.res[(2*N.subjects.v + 1):(4*N.subjects.v)]
      ref.AUCs   = AUCs[1:N.subjects.v] # reference arm of the trial
      test.AUCs  = AUCs[(N.subjects.v + 1):(2 * N.subjects.v)] # test arm
      ##
      AUC.ref.mean   = mean(log(ref.AUCs))
      AUC.ref.sd     = sd  (log(ref.AUCs))
      AUC.test.mean  = mean(log(test.AUCs))
      AUC.test.sd    = sd  (log(test.AUCs))
      Delta.AUC      = AUC.test.mean - AUC.ref.mean
      ##
      ## Compute BE for this trial
      Ratio.Cmax = exp(Delta.Cmax)
      Ratio.AUC  = exp(Delta.AUC)
      Cmax.yes   = ((Ratio.Cmax > 0.8) && (Ratio.Cmax < 1.25))
      AUC.yes    = ((Ratio.AUC  > 0.8) && (Ratio.AUC  < 1.25))
      BE.yes     = Cmax.yes && AUC.yes
      ##
      N.fail = N.fail + as.integer(!BE.yes) # cumulated number of failures
      print(paste("trial", i, ", fails:", N.fail))
      if (N.fail > N.mtc.v * 0.05) {
        ## this Delta vector will not pass, stop
        BE[j] = FALSE
        break
      } else {
        if (i == N.mtc.v) {
          ## this Delta vector lead to pass, do not look at further Delta's
          stop_after_this_trial = TRUE
        }
      }
    } # end for ith trial
    ##
    if (stop_after_this_trial) {
      BE[j] = TRUE
      break
    }
  } # end for jth Delta[2] value
  ##
  IDnum = seed + N.cores * (IDrun - 1)
  save(list=c("N.mtc.v", "N.subjects.v", "Delta", "BE"),
       file=paste0("Safe-space model based_", IDnum,".Rsave"))
  return(1)
  ##
} ## End run_allcode

t.start = Sys.time()
chain_output = parLapply(cl=this_cluster, X=1:N.cores, fun=run_allcode,
                         N.cores=N.cores, IDrun=IDrun)
t.end = Sys.time()
t.end - t.start

## Close the cluster when you're done with it.
stopCluster(this_cluster)

## ===============
## Post processing
n.start = 17 # real useful work starts at 17
n.end   = 52
for (j in n.start:n.end) { # for each Monte Carlo block
  ##
  if (!(j %in% 30:32)) { # weed out unneeded points
    myname = paste0("Safe-space model based_", j,".Rsave")
    load(file=myname)
    ##
    if (j == n.start) {
      Delta.all = Delta
      BE.all    = BE
    } else {
      Delta.all = rbind(Delta.all, Delta)
      BE.all    = c(BE.all, BE)
    }
  }
}

## Plot safe space
pdf("Safe space points plot v2.pdf")
mycols = rep("orange", length(BE.all))
mycols[which(BE.all == F)] = "red"
mycols[which(BE.all == TRUE)] = "green"
mylims = c(0.7,1.5)
plot(Delta.all[,1:2], las=1, xlim=mylims, ylim=mylims,
     xlab=expression(delta[1]), ylab=expression(delta[2]),
     col=mycols, pch=15, cex=0.55)
## lower limit
a = 1.265; b = -0.6 # y = a + b * x
abline(a, b, col="green")
## upper limit
a = 1.950; b = -0.47 # y = a + b * x
abline(a, b, col="green")
dev.off()

pdf("Safe space region plot v2.pdf")
mycols = rep("orange", length(BE.all))
mycols[which(BE.all == F)] = "red"
mycols[which(BE.all == TRUE)] = "green"
mylims = c(0.7,1.5)
plot(Delta.all[,1:2], las=1, xlim=mylims, ylim=mylims, type="n",
     xlab=expression(delta[1]), ylab=expression(delta[2]),
     col=mycols, pch=15, cex=0.55)
rect(0.7, 0.7, 1.5, 1.5, col="green", bord=NA)
## lower limit
a = 1.265; b = -0.6 # y = a + b * x
## abline(a, b, col="green")
polygon(c(0.7, 0.7, (0.7-a)/b), c(0.7, a+b*0.7, 0.7), col="red", bord="red")
## upper limit
a = 1.950; b = -0.47 # y = a + b * x
## abline(a, b, col="green")
polygon(c(1.5, 1.5, (1.5-a)/b), c(1.5, a+b*1.5, 1.5), col="red", bord="red")
dev.off()

## End.
```

References

1. Samtani MN, Vermeulen A, Stuyckens K. Population pharmacokinetics of intramuscular paliperidone palmitate in patients with schizophrenia: a novel once-monthly, long-acting formulation of an atypical antipsychotic. Clinical Pharmacokinetics. 2009;48:585-600 (PMID19725593).

2. Magnusson MO, Samtani MN, Plan EL, Jonsson EN, Rossenu S, Vermeulen A, et al. Population pharmacokinetics of a novel once-every 3 months intramuscular formulation of paliperidone palmitate in patients with schizophrenia. Clinical Pharmacokinetics. 2017;56:421-433 (PMID27743205).

Reviewers' Comments:

Reviewer #1:

1. Line 48-49. The authors make a statement about arbitrarily limit the size of virtual trials is suboptimal. In my experience and my understanding of the current practices, the size of a virtual trial should mimic the size of the actual trial taking into account the knowledge of the drug product as well as its estimated variability. The authors are requested to clarify what they mean by this statement.

Precisely, we challenge the current practice of mimicking an actual trial size and demonstrate the advantages of using a different approach where size is not a problem anymore. The usual requirements for an actual trial are reasonable but they do not need to apply to a virtual trial. We modified the text of the introduction to make this point clearer.

2. Line 59. The absorption parameter is a drug specific parameter. What should differ between the test and reference product are the release characteristics of the test and reference formulation.

You are right, but in the compartmental pharmacokinetic model used here, those parameters aggregate drug-release and absorption phenomena. We have modified the text and call them appropriately now.

3. Line 189: The authors state that they discarded the first 2500 iterations. The authors should justify the use of the remaining 7500 iterations only instead of using the whole data sets and whether the results would still be the same if using the whole data set or just the remaining 7500 iterations.

The iterations of the MCMC sampling procedure do not generate “data” but generate “samples” from the joint posterior parameter distribution. Practically all the MCMC sampling procedure, including the one we used, need a warm-up (also call burn-in) period before converging to that target distribution. The samples generated during that first period must be discarded. If kept, they would produce spurious results. We have added a new figure (Figure 5) in the Appendix to illustrate MCMC sampling convergence. The chains started from different random values and converged in probability to the target distribution only after about 2000 or less iterations. For all parameters, the first 2500 iterations were discarded to make sure that only values at convergence were kept.

4. For the safe space calculations, the authors are requested to explain why the number of subjects were not kept the same.

You are right, there is no particular reason to change the number of subjects for the partly Bayesian workflow A safe space calculations. We redid the calculations with 130 subjects per arms. They turn out to lead to the same results than with 500 subjects. The safe space calculations for the fully-Bayesian workflow B are posterior probability calculations and should use maximum precision; hence the higher number of subjects for workflow B. That comes at the cost of longer calculation time, as mentioned in the text.

5. Any use of a model assumes that there are well-defined acceptance criteria for the model. Did the authors use a prediction acceptance criterion above which the model is deemed unacceptable. It is my experience that regulatory agencies expect to see a well-defined acceptance criterion to decide on the validity of the model and its ability to be used for regulatory decision making.

Yes, indeed, we followed the usual criteria. Overall, predictions were within a factor 2 of the summary observations and the median estimates were within 25% of their observed counterparts reported in Magnusson *et al*. This is now explained in the text and detailed with a new Figure and Table in the Appendixs. We would like to point out that the paliperidone data and model we used here have been accepted by FDA for the NDA, *i.e.*, implicitly validated for PK predictions. Furthermore, the model has been fitted by Magnusson *et al*., so that usual purely predictive criteria do not apply very well, but we do not have the original data points to check the model more finely.

6. For ease of reading, it is suggested to include some table of the results. It is difficult to assess or review results presented in graphical form only.

Yes, we understand, but the space constraints are very hard to match in that case. We therefore added three Tables, but placed them in Appendix: First, for model checking and to answer the previous question, we added a goodness of fit summary Table; Second, a Table summarizing the power calculations has been added to the relevant section; Third, a Table of posterior summary values for log (δ_2) and δ_2 is now given in a new section of the Appendix.

Reviewer #2:

This manuscript by Brochot and Bois has been submitted for consideration in the AAPS Journal. In this work, the authors describe a fully Bayesian model-model based virtual bioequivalence (VBE) framework and compare the results to a data-based VBE workflow. The workflows are illustrated using a hypothetical virtual paliperidone palmitate generic formulation, comparing it to the reference using a prior published model.

Having taken time to carefully consider the manuscript, I believe that the work is certainly interesting and the approach outlined could add benefit to the community. However, the manuscript at the moment is not suitable for publication and further work is necessary.

I explain my concerns below:

1. Scientifically, some of the results do not appear to make sense to me. On page 13, in the large virtual trial simulation, the partly Bayesian data-based workflow declares the hypothetical formulation to be bioequivalent, and the fully Bayesian workflow declares it to be non-bioequivalent. However, the marker shown on Figure 9 (right panel) shows the hypothetical formulation to be within the safe space. This needs to be corrected as it presents a very confusing story to the reader.

The results are a bit counter-intuitive, but not incoherent. The fully Bayesian workflow *B*, indicates that on average, “bad luck” being excluded, bioequivalence should be obtained with a large virtual trial simulation of size 130 (or even 500, see answer to question 4 of reviewer 1). However, the data-based workflow *A*, judges BE only on the basis of *one* simulation and it falls in the “grey area” when decisions can be inconsistent just because of luck, that is between the red and green lines. So, in a way, the blue cross for workflow *B* marks just a combination of parameter values; for workflow *A* it marks the same combination, but *also* a particular virtual trial.

2. It is not clear to me why Paliperidone Palmitate was chosen to illustrate the concepts in this paper. It is not clear why a complex long acting injectable is necessary, when perhaps a simpler product may make it easier to demonstrate the concepts.

The point is well taken and we basically agree, but we found the paliperidone data and published model interesting, well documented and (importantly) accepted by FDA for the NDA, *i.e.*, implicitly validated for PK predictions. There are also no generics for the TRIZA product studied here and VBE seems to be the only feasible approach from an economic point of view. Finally, a recent paper by Gajjar *et al*., which you mention below, has used the example of paliperidone INVEGA formulation (once a month) to study bioequivalence metrics sensitivity to various formulation parameters. We had failed to mention it because of the drastic space limitation, but we do now in the revision. This seems therefore quite a topical issue. We are afraid that for a simpler drug most people would say that a simple clinical trial would be enough and not challenging and that going for VBE is not very interesting in that case...

3. The authors have used Cmax and AUC as metrics used to determine bioequivalence. However, C_{min} is also necessary in some products, for instance with the chosen example of Paliperidone Palmitate. In addition, partial AUC's are also necessary for some products based on therapeutic and pharmaceutical considerations (https://ascpt.onlinelibrary.wiley.com/doi/10.1002/cpt.2174). Indeed, partial AUC's were accepted as necessary by the FDA for Paliperidone Palmitate, the chosen case study in this work. It would be prudent to extend the analysis to include C_{min} and partial AUC's.

Yes, for example Gajjar *et al.* in 2023 have discussed various metrics and their impact on BE using sensitivity analysis. We did not use C_{min} but note that we are using partial AUC at steady state (we now mention it clearly in the paper). Gajjar *et al.* also studied PP1M, not PP3M, and the results might be different. It might be interesting in the case of PP3M to assess BE with C_{min} also. However, this falls out of the scope of our work (long and complex enough to describe already). We went for a simplification by using just C_{max} and *AUC*. Nevertheless, we thank you for your comment and we added a brief discussion of the possibility of using alternative metrics, and mention Gajjar *et al*. work at that occasion, together with Lionberger *et al*. 2012 (we were well aware of those publications, but space limitations again bit us).

4. Why was a cross-over study not considered or simulated?

Well, we stayed probably too close, for once, to reality, as a real cross-over BE trial for PP3M is probably out of question for time and resources reasons. We agree that for VBE, a cross-over design could be used (we now mention it explicitly in the discussion). We actually plan to investigate this question, but it is out of the scope of this paper.

5. Figures such as figure 4 and figure 7 are difficult to interpret with many lines on top of each other. The authors should consider alternative visualization of this data to help accurately show their message.

We suspected that many readers would like to "see all the data" to get a feel for it (we do too). We do not really like summarizing groups by averaging over subjects, because this can be misleading, but in this case, we get rather clear pictures and we added them in Appendix to answer your comment. You can now more clearly see the peril of relying only on a single virtual trial. Note also that Figures 5 and 6 help a little bit in visualizing the abbreviated trial data.

6. The authors should comment and explain in detail whether they think this approach would be accepted by regulatory bodies, and if not what the short comings in this method are.

We do not want to make assumptions about the acceptability of this approach from a regulatory point of view, partly because the method can already be used internally by industry to orient decisions (so regulatory bodies are not our sole target audience), and partly because we are not a regulatory body and want to avoid perceived conflicts of interest. We trust the agencies to review the literature and make their own judgments about proposed innovations. Indeed, we will watch with interest those developments. We have just been awarded an FDA grant where those issues will probably be discussed.

7. The authors should consider using covariates to adequately describe population variance, and if not, should explain in detail why covariates are not necessary. Also, typically different formulations have different variability in absorption, and as the absorption rate is the rate determining step for a long acting injectable, the authors should explain in detail how this can be accounted for.

We would like to take covariates into accounts, but the covariate data and full covariance matrix were not published by the innovator company. We could not do a good job at that and preferred to leave the covariate model out, at the expense of a slight likely overestimation of inter-subject variability. We already explain that in the paper. Note also that it would be a second-order refinement of the model and does not affect our overall results about the advantages of a fully Bayesian approach. We also agree that different formulations may have different variability in absorption. In a population PK context, the solution would be to estimate from the prior clinical data and abbreviated trial a covariance matrix per formulation. In a population PBPK context, the same approach could be used if a Bayesian approach is taken, even though we doubt that *in vitro* evidence would be available on that difference. A simpler conservative approach would be to assume the possibility of differing variances and assess their impact by sensitivity analysis, for example. We added a few sentences to the discussion to address this point.

8. In addition, the narrative is very difficult to understand and does not allow the results to be adequately explained. Further explanation of the terms, concepts and results is necessary to allow the interesting science to be available to the readership. The methods and results should also be more carefully explained to allow the reader to follow the science better.

We tried to address this comment by improving the text in many places. We agree that the concepts are unusual to many readers and we think that space is too short. We may have to write a book at some point!

9. The paper also does not adequately take into account prior literature, which is very important for understanding the context of this work and implications it can have for BE assessment. For instance, there was recently an FDA workshop on model based approaches for long acting injectables (https://www.complexgenerics.org/LAI2021/) with a summary provided in the following paper: https://ascpt.onlinelibrary.wiley.com/doi/10.1002/psp4.12931. There have also been previous workshop discussions summarized here: https://accp1.onlinelibrary.wiley.com/doi/10.1002/cpdd.928. A data-based BE method using population PK approaches based on Paliperidone Palmitate was recently published (https://www.sciencedirect.com/science/article/pii/S0928098722001816), with the data-based approach also previously applied to BE assessment of oral formulations (https://link.springer.com/article/10.1007/s11095-011-0662-8). It would be interesting to see how the results of this paper directly compare to the data-based approach published on Paliperidone Palmitate.

Yes, we now quote the useful papers of Gajjar *et al*. and Lionberger *et al*. However, the Gajjar paper does not directly compare because the products are different (they studied PP1M, not PP3M). We also agree that the recent workshop reports do deserve to be cited and we added them to the introduction.

10. The authors comment in the introduction that BE trials can be long an expensive – is this always the case? Oral product trials may not be that long with short washout periods. The authors may also benefit from gaining feedback from colleagues not directly involved in modelling or colleagues not familiar with Bayesian approaches, since a publication on BE is likely to be considered from a wide variety of readers in pharmaceutical sciences who may wish to use the ideas for their own product development.

We agree with this statement and were careful to say that BE trials *can* be long, not that they *are* necessarily long. We amended the introduction a bit to give an example and added a reference.

Reviewer #3:

The proposed approach, once published, can have a significant impact on a very important area: bioequivalence (BE) studies for long-acting injectables. We need to carefully scrutinize the proposal presented here for virtual bioequivalence (VBE) studies in this regard. There are some major flaws in the simulation plan for performance checks of the proposed approach and theoretical framework before its publication.

1. Both the data-based workflow and fully Bayesian workflow require evaluations for type 1-2 error controls. While the authors correctly pointed out that the data-based workflow can achieve this, they still need to test the likelihood of a Bayesian workflow leading to a difference in conclusion when the underlying product is either bioequivalent (BE) or non-bioequivalent (NBE). It is not true that the Bayesian workflow does not require such evaluation. It may be necessary to define BE or NBE products upfront and be open to applying some subjectivity here.

We agree and you are right, but the question is quite deep. We did not mean to say that concerns about making the right decision with an acceptable error rate disappear completely in workflow B; it’s just that standard statistical test performances, *stricto sensu*, do not apply anymore. We have added a sentence in the methods section presenting the workflow to make it clear. We have actually performed type-I error analyses for the data-based workflow but did not show them in the paper. We have now put them in Appendix and discuss them. For the model-integrated workflow there is no large trial and associated statistical test. Conditionally on the (posterior) distribution of formulation differences and associated posterior predictive distributions of PK measures of drug absorption, we can obtain the probability that the estimated formulation differences lead to unacceptable differences in drug absorption (Figure 8 of the paper). In our case study, if we declare bioequivalence and let the drug go to market, there is a 35% chance that we release a non-bioequivalent product; if we do not declare bioequivalence and block the product, there is a 65% chance that the product is in fact bioequivalent in the population. So, it is a judgement call or a regulatory choice, but if we adhere to the strict practice of controlling direct consumer risk (at 5%) we would reject bioequivalence, with a relatively high direct producer risk. A deeper problem is that in a VBE framework, either data-based (workflow A) or model-integrated (workflow B), there is very little specific clinical evidence (only an abbreviated trial) but we have the help of a validated (*i.e.,* as good as possible) structural model and *in vitro* data. Usual BE trial analyses also make assumptions and have some untold model lurching in the back (like when using drug plasma concentrations when we want to assess bioequivalence of a gastro-intestinal-tract-acting drug), but the issue is more glaring in VBE assessment. A further problem we, as authors of this paper, face is that we present a doubly virtual BE analysis. We simulate the abbreviated trial and the "ground truth" of our case study is laid bare for everyone to compare to the results of workflows A and B. Readers can immediately see the incoherences between "truth" and "decision": the data-based workflow leads to a correct decision if we know the truth, but an incorrect decision given the information from the abbreviated trial; On the contrary, the model-integrated workflow decision is correct, given the abbreviated trial, but incorrect given the ground truth. The fact is that in a "real life VBE assessment", we will only get the model, the prior, and the abbreviated trial data; ground truth will be inaccessible to us. So, we will always be at the mercy of incoherent abbreviated trial and large trial simulations. What we show, is that on average the Bayesian workflow is more coherent and safer for everyone (producers *and* consumers). In a way, in a data-based VBE framework, type I and type II calculations on the virtual large trial can be a smoke screen, giving a false sense of security, as if they were dispelling the only source of potential error and masking the real issues of VBE (model quality). So, we should not conduct a VBE assessment as a BE assessment and should not judge a VBE assessment like a BE one. The model structure and correct parameterization are very important for both data-base and model-integrated workflows. We had mentioned that already, together with more obscure and technical considerations on the use of added-value-of-information calculation to assess the impact of the size of the abbreviated trial. This was not making justice to the issue you raised and we have extended and re-organized the discussion to fully answer your point.

2. In the simulation, the authors used a parallel study design, but the most popular design used in practice is still the steady-state cross-over study design, even considering its long study duration, high cost, and dropout rates. Simulations need to be conducted with the cross-over study design.

They do not really need to be done, but can be done, and we now present corresponding results.

3. In theory, a correctly optimized data-based approach can provide reasonably comparative power. Please state the reasons why this would be the case under some scenarios.

Yes, a correctly optimized data-based approach with good prior knowledge of the difference between formulations and the best possible design would have reasonable power. We have added that point to the text of the results' section but note that power is often computed using the assumption that products are strictly bioequivalent (in fact, identical). If deviations of C_{max} or *AUC* by say 5% are allowed, power drops down already as shown on the power plot in Appendix, while actually a deviation by up to 20 to 25% should be possible without infringing the BE definition; however, our power plot shows that in such cases reasonable power would be reached only at very large trial sizes.

4. When generating virtual abbreviated BE data for methodology performance checks, the authors assumed no changes in pharmacokinetic (PK) parameters for the reference product. This is invalid since the field has been witnessing significant variations between studies, even for PK. The authors need to re-generate abbreviated virtual BE data with additional variations from the original PK model, informed by new drug development.

As we mentioned above, this is by no mean intended to be for a regulatory assessment of our model and even less whether a given generic product, which does not exist, is BE or not. We are just presenting new concepts and methodology. The model did simulate a totally different sample from the Janssen original studies and generating a new sample and redoing all calculations will not bring anything substantive to the work. Conclusions about safe-space differences between the two workflows, which is probably our most important contribution, will not be affected in any way by new simulations because we already average over many simulations. We would come up with a different abbreviated trial, a different posterior distribution of difference between test and reference and then a different large trial, and potentially different decisions about product bioequivalence. But even regulatory agencies do not require redoing clinical trials just because results might be different and the same should apply to virtual trials. There would be no end to that, what would be the point? Again, this is not a VBE assessment for a product; this is intended to be a discussion of overarching issues in VBE. Computational asymptotics could be tried on the abbreviated trials but this would amount to added value of information calculations which, as we indicated in the paper already, would be interesting but beyond its scope. We have added a discussion of the trial simulations in the discussion to make this point clearer for all readers.

Below are editorial recommendations:

5. Virtual trials: change to virtual studies

“Studies” or “trials” are used in the literature. “Studies” are more general and “trials” is more specific of a test of hypothesis (BE or not), so we do prefer trials. We made sure we are consistent throughout.

6. "Virtual Trial Comparisons" is confusing by meaning and should be revised.

Yes, we replaced that, including in the paper title, by “virtual comparative trial” which seems more standard.

7. Line 22: Please by specific on what circumstances "BE trials can be long and very expensive".

Yes, we amended the text with an example and added a reference.

8. Model-based BE: change to Model integrated BE, as currently defined for BE assessment.

Well, at least as defined by FDA, for example in “Establishing the Suitability of Model-Integrated Evidence (MIE) to Demonstrate Bioequivalence ...” by Gong *et al*., which we now cite. For the sake of uniformity, we made the requested change everywhere.

9. Line 47: It states: "At the limit, the standard statistical tests would need to operate with zero-length confidence intervals, and power analyses become moot." I do not think this is true. FDA would recommend a Modeling Analysis Plan ahead of time where the applicant need to specify the virtual study sample size.

Yes, FDA would probably do that, but that is precisely what we discuss and propose to improve upon, based on the fact that virtual trial sizes are practically unlimited and that asymptotics-based tests are not applicable at the (infinity) limit. That is just a coherence argument.

10. Line 51: "Limiting arbitrarily the size of virtual trials is sub-optimal in terms of decision making. It lowers power and affects both producers and consumers because a safe product, potentially less expensive, might not be approved when it could be" is an over-statement. The modeling & simulation plan always estimate type-I and type-II errors to make sure a reasonable risk control for both consumers and firms (producers).

Reasonable maybe, or at least borne with because losses from increased producer risks are recouped later by higher drug prices, but sub-optimal certainly, as we show. Those are not incompatible statements.

11. Line 79: "If the model needs to be improved, recalibration using the abbreviated trial data can be tried. " It is a must step. You cannot directly use a model from literature without any update for regulatory use.

Yes, we agree and that was obvious to us. Our point was technical though, and we meant *Bayesian* recalibration, not just simply updating, manually or otherwise. We have modified the sentence to make it clearer.

12. Line: 104: "Questions about type I and type II errors of the statistical tests for a simulated trial disappear from our concerns" is an overstatement. Type I and II errors will still need to be evaluated in a different form. It will not disappear when using a Bayesian approach.

Yes, this was also the gist of the first point you made and which we answered above.

13. Please define what is "safe space analysis" in the first place.

Yes, this is done at the first occurrence of the term, at the end of the introduction. We added a reference for more details.

14. Line 171: "Parameter 3, was determined to be the most influential on C_{max} and AUC". With a constant CL, you would expect the only parameter that impact on AUC would be bioavailability. Please clarify.

Usually yes, but we have a product where the rate of absorption conditions the decay phase (flip-flop) and we are using partial AUC at steady-state. So, the release rates condition the trough concentrations, and therefore C_{max} and *partial AUC*.

15. Line 175: "Because our population PK model has strong prior information on the reference formulation parameters, a Bayesian approach is well suited to estimate the value of the difference 2. Metropolis-Hastings (a.k.a., random walk) Markov-chain Monte Carlo (MCMC) sampling was used to obtain a sample of parameter values from their joint posterior distribution given the abbreviated trial data. We fixed the population means and variances to the central values (MLE values given by Magnusson et al.) of their prior distributions." It is a dangerous move to assume that there is no PK parameter change for the reference product across trials. I strongly recommend an evaluation scheme that there will be PK variabilities across studies.

This comment is similar to point 4 above and our answer would be the same. It would be a refinement of the approach and we discuss it.

Reviewer #4:

1. The rationale for this research stated in the first sentence of the abstract "The recent emergence of virtual trial comparisons, ..., begs for formalization of their analyses" makes no sense. Comparisons cannot beg - only humans can beg. It appears that the rationale or motivation for the research in this report is based upon the authors comments made in lines 45-55 in the Introduction.

Yes, but grammatically, in that sentence, it is the “recent emergence” that “begs”. Arguably, this figure of style may seem obscure, as emergences do not beg either. So, we have amended the sentence to make it clearer. What we write in lines 45-55 of the text is summarized just after the revised sentence in the abstract.

2. There are numerous terms expressed in this manuscript that are unconventionally employed, mostly undefined in the manuscript and confusing, including "safe space analyses", "data streams", "calibration", "abbreviated BE trial", "CQA safe-space analyses", "reference formulations' critical quality attributes (CQA)", "workflow", etc. Their undefined usage makes the entire manuscript hard to understand.

We are sorry for the jargon. We define “safe space analyses” at the first occurrence of the term, at the end of the introduction; we added a reference for more details. We replaced “data streams” by “kinds of data”. Bayesian calibration is approximately equivalent to model fitting; we precise it in the text at the first occurrence now. “Abbreviated trials” are used in the biosimilars literature (*e.g.,* Wu *et al.*, 2018, PLoS ONE 13: e0208354); we now precise their meaning (small-size studies) at the first occurrence. CQA is the abbreviation of the rather clear “critical quality attribute” defined by FDA since at least 2012 (www.fda.gov/media/83904/download), see also Pepin *et al.*, 2021, J. Pharm. Sci. 110:555, for example. Our use of workflow is not really unconventional.

3. The description of the case study analyses, manuscript figures, and Appendix together are poorly integrated, so that the core findings are obscure.

With limited space we cannot present all the useful results in the body of the paper (we even had to add some more to answer this review) and we agree that going back and forth between the text and the Appendix is unwieldy. But there is not much we can do. We tried to improve the text flow though. We hope it is better now.