${\it microsamplingDesign}$

Finding optimal microsampling designs for non-compartmental pharmacokinetic analysis

Adriaan Blommaert; Open Analytics 2020-04-05

Contents

1	Inti	roduction	2
2	Mo	del details	2
	2.1	Parametrization	3
	2.2	Log-normal parameters	3
3	mic	rosamplingDesign shiny application	4
	3.1	Construct a pharmacokinetic model	4
	3.2	Generate possible time points	5
	3.3	Rank time points	5
	3.4	Generate possible schemes	7
	3.5	Rank schemes	7
4	Fin	ding optimal designs using code	9
	4.1	Settings	9
	4.2	Construct a pharmacokinetic model	10
	4.3	Generate time points	10
	4.4	Rank time points	11
	4.5	Generate possible schemes	12
	4.6	Rank schemes	13
5	Adv	vanced options	14
	5.1	Parallelization	14
	5.2	Working with ranges	15
6	Me	mo of main functions	15
	6.1	Data generation	15
	6.2	Generate and rank time points	15
	6.3	Generate and rank schemes	15

1 Introduction

Microsampling, a novel blood sampling technique allows multiple blood samples to be taken per animal, reducing the number of animals required for pharmacokinetic-pharmacodynamic studies (Chapman et al. (2014)). Using sparce designs can in addition, avoid unnecessary sampling of these animals, provided an appropriate choice of sample times per animals is made. The microsamplingDesign package implements a general simulation methodology to find optimal sparse microsampling schemes aimed at non-compartmental pharmacokinetic analysis (algorithm III in Barnett et al. (2017)). This methodology consist of (1) specifying a pharmacokinetic model including variability among animals; (2) generating possible sampling times; (3) evaluating performance of each time point choice on simulated data; (4) generating possible schemes given a time point choice and additional constraints and finally (5) evaluating scheme performance on simulated data. The default settings differ from (Barnett et al. (2017))) in the default pharmacokinetic model used and the parameterization of variability among animals (see next section). A shiny web application is included, which guides users from model parametrization to optimal microsampling scheme.

2 Model details

A two compartmental oral dosing pharamcokinetic model (Gabrielsson and Weiner (2001)) is assumed:

$$\frac{dD_g}{dt} = -k_a D_g$$
$$V_c \frac{dC}{dt} = F \cdot k_a D_g - Cl \cdot C - Cl_d \cdot C + Cl_d \cdot C_t$$
$$V_t \frac{dC_t}{dt} = Cl_d \cdot C - Cl_d \cdot C_t$$

A dose of a substance (D_g) is administered to the gut, than graduadely absorbed into a central compartment leading to a increased concentration in the plasma (C). Where it can either be excreted or exchanged with a second peripheral compartment, the peripheral tissues, where the compound has a distinct concentration (C_t) in time, depending on the rate of exchange with the central compartment. We do not assume any excretion from the peripheral compartment.

Substance absorption and clearance are by default assumed to be capacity dependent (Michaelis-Menten kinetics):

$$k_a = \frac{V_{a,max}}{\kappa_{a,m} + D_g}$$
$$Cl = \frac{V_{e,max}}{\kappa_{e,m} + C}$$

We also leave the option open for one or both of these parameters to be constant.

For details see (Gabrielsson and Weiner (2001)).

2.1 Parametrization

- k_a is the absorption rate per unit of dose.
- V_c is the volume of the central compartment (plasma)
- V_t is the volume of the peripheral compartment (tissue)
- F bioavailability, the fraction of the dose that reaches the systemic circulation intact (dimensionless)
- Cl is the elimination rate from the central compartment (assumed the only spot where elimination occurs); in volume per time, related to the elimination rate in dose: $\left(k_e = \frac{Cl}{V_c}\right)$
- Cl_d is the distribution parameter between central and peripheral compartment; expressed in volume per time unit. It related to rates: $Cl_d = \frac{k_{ct}}{V_c} = \frac{k_{tc}}{V_t}$; with k_{ct} the rate from central to tissue (dose per time unit), and k_{tc} the rate from tissue to central compartment.
- $V_{a,max}$ is the maximum absorption rate (absolute rate is rate per dose x dose)
- $\kappa_{a,m}$ is the Michaelis-Menten constant for absorption
- $V_{e,max}$ is the maximum clearance rate (absolute rate is rate per concentration x concentration)
- $\kappa_{e,m}$ is the Michaelis-Menten constant for clearance.

2.2 Log-normal parameters

Individual animals are assumed to have the same underlying model, with different parameters simulated from an underlying log-normal distribution parametrized in terms of the mean and the coefficient of variation.

we assume a random variable X to be log-normally distributed with parameters μ and σ :

$$X = \exp\left(\mu + \sigma Z\right)$$

with Z a standard normal variable.

Now, we want to extract μ and σ from and coefficient of variation (CV = sd(X)/E(X)) of the original scale. we can use the relation for the mean:

$$E(X) = \exp\left(\mu + \frac{\sigma^2}{2}\right)$$

and the relation for the coefficient of variation:

$$CV(X) = \sqrt{\exp \sigma^2 - 1}$$

Therefore:

$$\sigma = \sqrt{\ln(CV^2 + 1)}$$

and

$$\mu = \ln\left(E(X)\right) - \frac{\sigma^2}{2}$$

For the multivariate log-normal distribution, we use a the same approach per variable and can simulate a random vector:

$$\boldsymbol{X} = \exp\left(\boldsymbol{\mu} + \boldsymbol{Z}\boldsymbol{\sigma}^{T}\right)$$

with $Z \sim \mathcal{N}(\mathbf{0}, \Sigma)$ and Σ a specified correlation matrix. More information see (Halliwell (2015))

Optimize	e microsampling scher Change Parameters	1
	Click to change	
Pk model	parameter value coeffVariation explanation	
	Data input 1 F 1.00 0.00 bioavallability	
	2 volumePlasma 10.00 0.20 volume of the central compartment (plasma)	
	Model parameters 3 Cld 15.00 0.20 distribution parameter between central and peripheral Compariment expressed in volu-	
Time points	Show 10 volumeTissue 15.00 0.20 volume of the peripheral compariment (fissue)	
	5 VmaxAbsorption 5.00 0.20 maximum absorption rate (absolute rate is rate per dose x dose)	
Generate possible time points	parameter 8 kappa/fMAbsorpton 0 0.20 Michaells-Menten constant for absorption	Y.
	7 KaConstant 0.20 constant absorption rate per unit of dose (overrides Michaelis Mantan kinetics)	te D - New
Ronk time pointa	8 VmaxClearance 30.00 0.20 maximum desrance rste (absolute rate is rate per concentration x	Vare in
-	2 volumePlasma 9 kapp4/MClearance 0.25 0.20 concentration)	North Anna San
Schemes	10 Ciconstant 0.20 constant clearance rate (overndes rutchealis-Menten kinetics.)	
	3 Cid	4 4
Senerate possible schemes	4 volume linux	
Rank schomos	Cless	
	5 VmaxAbsorption	
	6 Janual Millionman 25 02	
		Natura Natura
	7 KaConstant 0.2 🖁 🔐	= V _{ep} /2
	a maccante a a a a	
	9 xappoMMCkiannoa 0.25 0.2 concertator(C)	
	10 CiConstant: 02	
	Simulate example data	
	serving to to environments Previous 1 Next Prelias of individual samples him lead parameters	
	prive for survive later to Huggston model blow correctable	
	Citizen Gerages warmin cover	
	Dosing into	
	time dose	
	1 0.00 2.50	
	2 0.00 0.00	
	S 0.00 0.00	
	4 0.00 0.00	
	5 0.00 0.00	
	6 0.00 0.00	
	7 0.00 0.00	
	6 Generatik example ourves	
	O Graphical settings	

Figure 1: Construct a PK model

3 microsamplingDesign shiny application

Before diving into the R code of the microsamplingDesign package, we give a more intuitive introduction to the methodology using the included shiny application. In a local R session we can start the application:

```
library( microsamplingDesign )
runMicrosamplingDesignApp( installDependencies = TRUE )
```

The first time you want to run the application, use *installDependencies* = TRUE to automatically install the additional R-package required for this shiny application in addition to the microsamplingDesign package dependencies.

3.1 Construct a pharmacokinetic model

Start the application by constructing a pharacokinetic model.

Example parameters are shown on start up. To modify these parameters click on **Modify parameters** and a spreadsheet is displayed allowing modifying parameter values and their coefficient of variation (see Figure 1).

Next include dosing information by filling out one or several lines, click on **Generate example curves** to check simulated time-concentration curves (see Figure 2).

One can adapt the scale of the graphs by clicking on **Graphical settings**.

Note that the pharmacokinetic model in the application does not contain any measurement error.



Figure 2: Check model by generating example curves

3.2 Generate possible time points

Time point options are generated from a time constraints table specifying the number of time points per time zone and minimum sampling interval in each row. Note that the endTime is not included in the zone itself but is the startTime of the next zone.

Finally click on the button **Generate time points**, to recieve all possible combinations in table form (see Figure 3).

3.3 Rank time points

Time points options are ranked by measuring the difference between approximating the average timeconcentration curve based on a limited number of time points on sample data and the actual average curve at the maximal number of time points you want to consider. This is a measure of bias caused by choosing a certain time point option rather then sampling at the maximum number of time points.

In the application ranking time points takes 2 steps:

3.3.1 Generate sample data

Specify the approximate number of animals you would like to use in you scheme and the number of simulated datasets to generate. Then press **Generate data to rank time points**. A selection of simulated data will be displayed (see Figure 4).



Optimize microsampling schemes for non-compartmental analysis

Construct Pk-model	Tim 9 Tim	e cons e constraints	traints						
Time points		startTime	endTime	nPointsPerZone	timeInterval				
	1	0.00	3.00	3.00	0.50				
Canarate nossible time colote	2	3.00	5.00	2.00	0.50				
crementate possible unite points	3	0.00	0.00	0.00	0.50				
tank time points	4	0.00	0.00	0.00	0.50				
	5	0.00	0.00	0.00	0.50				
Schemes	6	0.00	0.00	0.00	0.50				
oononioe	7	0.00	0.00	0.00	0.50				
Generate possible schemes	clear	time constr	aints						
Rank schemes	Ger	erate t	ime po	oints					
	Generate time points								

Possible time points

{0:05:1:15:2:25:3:35:4:45:5}

Time point combinations meeting constraints Number of combinations

60							
Time	point c	ombina	tions				
Show	10 ~	ontries			Search:		
	TimePo	ainti 4	TimePoint2	TimePoint3	TimePoint4	TimePoint5	TimePoint6
1		0.5	1	1.5	3	3.5	5
2		0.5	1	2	З	3.5	5
3		0.5	1	2.5	3	3.5	5
4		0.5	1.5	2	з	3.5	5
5		0.5	1.5	2.5	3	3.5	5
6		0.5	2	2.5	3	-3.5	5
7		1	1.5	2	3	3.5	5
8		1	1.5	2.5	3	3.5	5
9		1	2	2.5	3	3.5	5
10		1.5	z	2.5	3	3.5	5
Showin	id 1 to 10	of 60 ent	les	Previous	1 2 3	4 5	6 Next

Time In hours

Figure 3: Generate time points

Optimize microsampling schemes for non-compartmental analysis Pk model 3.5 timePointOption6 0.5 2 2.5 3 Data generation ImePointOption7 Ξt 1.5 2 3 3.5 Construct Pk-model Number of animals per sche 1,5 2.5 3.5 timePointOption8 3 Time points 3 2 2.5 3 3.5 timePointOption9 1 Generate por Number of simulation samples sible time points TimePointOpTion10 1.5 2.5 3.5 2 3 1000 Rank lime po Showing 1 to 10 of 60 entries Previous 1 2 4 5 Next 3 6 Greater number of samples will increase simulation time Schemes 06 Generale data to rank time points Simulated data Generate possible schemes Rank time points 0.04-Bank schomes 0° Rank time points 0.03 Choice of time points DIASTNA Number of top time point options to disp 60 10 centration in 0.02 50 0.0

0.00

Only a selection of data is shown

Figure 4: Generate data to rank time points

Pk model	Data generation Number of animals per scheme	Tim	ne point ranki	ng		54	arch:	
ime points	a Number of simulation samples	<u> </u>	name 🕴	criterion #	rank 🕴	TimePoint1	TimePoint2	TimePoint3
ani: Sma points	1600	(3)	ImePointOption6	0.001848	:t- 69	0,5	2	2.5
un ane ponto	Greater number of samples will increase simulation time	2	tmePointOpton5	0 001852	2	0.5	1.5	2.5
chemes	© Generate data to rank time points	3	timePointOption4	0.001856	3	0.5	1.5	2
norale possible schemes		4	timePoIntOption2	0.001864	- 4	0.5	1	2
Rank schemes	Rank time points	5	timePointOption3	0.001894	5	0.5	1	2.5
	0° Rank lime points	6	tmePointOption1	0.001909	6	0.5	1	1.5
	Choice of time points	7	tmePointOpton56	0.002575	7	0.5	2	2.5
	Number of top time point options to display	8	tmePoIntOption55	0.002579	8	0.5	1.5	2.5
	10	(0)	timePointOption53	0.002621	9	0.5	1	2.5
		10	tmePointOption16	0.002641	10	0.5	2	2.5
		Show	ing 1 to 10 of 10 entries	É.			Previou	is t No
		Che	osen samplin	g points				
		1	TimePoint1 TimeP	oint2 Time	Point3	TimePoint4 Tir	mePoint5 Time	2Point6
			0.50	1.50	2.50	3.00	3.50	5.00

Figure 5: Rank time points and select one

3.3.2 Rank time points

After checking the generatated data, click on **Rank time points** to estimate the bias of each time point option. Calculations might take a few minutes, depending on the the number of simulation samples and time point options. When calculations are finished, time point options are tabulated from small to large deviation from the best accuracy. You can select a time point option by clicking on a row in the time point ranking table (see Figure 5).

3.4 Generate possible schemes

Given the time points, we will construct schemes specifying which subjects are sampled at which time points.

To generate these schemes, fill out the scheme's dimensions and the maximum number of repetitions of individual schemes. You can already assess the possible number of schemes by clicking on **Check number** of schemes before constraints wich is much faster then generating the schemes first. Reconsider scheme dimensions when the number of schemes is too large. The possible number of schemes can also be cut down by imposing *scheme constraints*. Finally click on **Generate schemes** to receive all schemes meeting constraints. This might take a few minutes (see Figure 6).

3.5 Rank schemes

Schemes are ranked by their precision of estimating the area under the curve (AUC) and maximum concentration (Cmax) on simulated data.

Again we work in 2 steps:

Optimize microsampling schemes for non-compartmental analysis

Pk model	Scheme dimensions	Tir	ne point	S						
Construct Fil-model	Number of subjects	(0,5 ; 1,5 ; 2,5 ;	3;3.5;5)						
Time points	Number of subjects									
Generate possible time points		S Nu	imber of	scheme	s before a	pplying co	onstraints			
Rarik time points	Number of observations per subject min	1	,001							
Schemes	*	(i) If too	large a numbe	er (100,000), i	reconsider 'schem	e dimensions'				
Generale possible schemes	max	Nu	imber of	scheme	s meetina	constrain	ts			
Rank schemes	5	(B)			0					
	Repetition of individual schemes	0	47.							
	Warning: repeating individual schemes will greatly increase the possible number of schemes	Sc	hemes r	neetina (constraints					
	Automation of repetitions individual schemes	8		neeting .	constraint		Sugreb:		-	
		Sho	w 10 ° e	ntries			search:		_	
	Scheme constraints		scheme	subject	timePoint1	timePoint2	timePoint3	timePoInt4	timePoint5	timePoint
	Scheme constraints	1	schemet	subject1	tue	rue	true	true	talse	true
	1 minObsPerTimePoint 2.00	2	schemet	sub(ect2	3'UB	3ne	false	3'UB	true	true
	2 maxConsecSamples 3.00	3	schemet	subject3	true	ta/se	talse	Tue	true	true
	4 Choese a constraint 1.00	4	schemet	subject4	TUR	710	true	TUB	talse	true
	5 Choose a constraint 1.00	6	scheme2	subject1	tue	тие	talse	true	true	true
	Heeet scheme constrailing	6	scheme2	sub(ect2	true	false	true	fa/se	true	true
	Generate schemes	7	scheme2	subject3	tue	tuć	true	truo	true	talse
	ct Check number of schemes before constraints	8	scheme2	subject4	true	tue	false	faise	true	Ince
	Too large a number will slow down computation, reconsider scheme dimensions	9	scheme3	subject1	tue	talse	talse	telse	true	true

Figure 6: Generate schemes

Optimize microsampling schemes for non-compartmental analysis

1

Pk model			4	scheme1	subject4	true	true	true	true	false
	Data generation		5	scheme2	subject1	true	true	talse	tue	true
Construct Pk-model	Number of simulation samples		6	scheme2	subject2	true	faise	true	false	true
Time points	1000		7	scheme2	subject3	true	true	true	true	true
Generate possible time points	Take a small number for testing, and a large number e.g. 1000 for the final run		8	scheme2	subject4	true	true	faise	faise	true
Rank time points	oge Generate data to rank schemes		9	scheme3	subject1	true	faise	false	false	true
O-h-man	Bank Schemes		10	scheme3	sub/ect2	true	false	true	tue	true
Scremes	Objective function		Show	ing 1 to 10 of	2,188 entries	Previous	1 2	3 4 3	5 21	9 Next
Generate possible schemes	Relative importance area under the curve (AUC)									
Flank schemes	50	(c)	Sin	nulated	data					
	Relative Importance maximum concentration (Cmax)									
	50	(6)		0.06-				N		
	o g Rark schemes				N			1		
	Obviou of Outcome		(asme	D.04 ·						
	Choice of Scheme		u u		1	1				CUIVE .
	Number of lop schemes to display	(6)	tratio		1	1	- 1		-	· sangle curve
	10	۲	oncer	0.02		~	s 11			
			0		1					
					1					
				0.00			11			
				.o	ĵ.	2	à	4	5	
						- D	me in hours			

Figure 7: Generate data to rank schemes

Cinto a entrellina of data in phone

Optimize microsampling schemes for non-compartmental analysis	Optimize microsampling sch	emes for non-comp	partmental analysis
---	----------------------------	-------------------	---------------------

Pk model Canatrust Pk model	Data generation Number of simulation samples	8	Only a	selection of data is sh eme ranking	own.				
Time points		(9)	Show	10 🗸 entries			Search:		
Generate possible time points	Take a small number for testing, and a large number e.g. 1000 for the tinal run			name 0	var_auc 1	var_cMax		criterion	rank
Rank time points	OS Generate data to rank schemes		(1 7)	scheme39	0.000098	0.000017	0.6114	156023930336	1
Schemes	Rank Schemes		2	scheme89	860000.0	0.000017	0.6128	328511928707	2
	Objective function		3	scheme287	0.000098	0.000017	0.6130	58145990248	3
Generate possible schemes	Relative Importance area under the curve (AUC)		4	scheme90	0.000099	0.000017	0.6142	293625211224	4
Park schemes	50	(0)	5	scheme87	0.000099	0.000017	0.6144	182496676645	5
	Relative Importance maximum concentration (Cmax)	0	Showin	ng 1 to 5 of 5 entries				Provinus	Next
	© Rank schemes		Cho Show	sen scheme			Search:		
	Number of top schemes to display			timePoint1	timePoint2	timePoint3	timePointd	timePoints	timePoint6
	5.	(0)	subje	ct1 true	tue	true	false	true	false
			subje	ot2 true	tue	false	true	true	faise
			subje	ct3 true	false	true	faise	true	true
	Summary report		subje	ct4 true	false	false	true	tue	true
	& Generate report		Showin	ng 1 to 4 of 4 entries				Previous 1	Next

Figure 8: Rank schemes

3.5.1 Generate sample data

Generate data by specifying the number of simulation samples and click **Generate data to rank schemes** (see Figure 7).

3.5.2 Rank schemes

After data generation, specify the objective function by attaching a relative importance to different noncompartmental statistics and click on **Rank schemes** (see Figure 8). This might take some time.

Finally select a scheme by clicking on the **Scheme ranking** table.

When a final scheme is chosen, first click on **Generate report** and next on **Download report** to recieve a word document summarizing the main results.

4 Finding optimal designs using code

4.1 Settings

settings <- list()
settings\$nSamples <- 100 # increase for real life example
set.seed(124)</pre>

4.2 Construct a pharmacokinetic model

```
library( microsamplingDesign )
pkModel <- getExamplePkModel()</pre>
```

some useful functions:

```
modelParameters <- getParameters( pkModel )
knitr::kable( modelParameters[ , c(1:2) ] )</pre>
```

parameter	value
F	1.00
volumePlasma	10.00
Cld	15.00
volumeTissue	15.00
VmaxAbsorption	5.00
kappaMMAbsorption	2.50
KaConstant	NA
VmaxClearance	30.00
kappaMMClearance	0.25
ClConstant	NA

To generate your own pharmacokinitic model see:

?construct2CompModel

4.3 Generate time points

Possible time points are generated from a full set of time points:

```
fullTimePointsEx <- seq(0, 16, 0.5)
print(fullTimePointsEx)
#> [1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5
#> [15] 7.0 7.5 8.0 8.5 9.0 9.5 10.0 10.5 11.0 11.5 12.0 12.5 13.0 13.5
#> [29] 14.0 14.5 15.0 15.5 16.0
```

With the choice of options constraints by *timeZones*:

```
#timeZonesEx <- getExampleTimeZones()
timeZonesEx <- data.frame(startTime = c(0, 2, 3),
endTime = c(2, 3, 16),
nPointsPerZone = c(2, 1, 2))
knitr::kable(timeZonesEx)</pre>
```

startTime	endTime	nPointsPerZone
0	2	2
2	3	1
3	16	2

timeZones concept is defined such that : time zero is never included, last timePoint is always included.

Correct names should be used!

Now we can generate all time point options from a vector of possible time points under constraints defined in timeZones:

```
setOfTimePoints <- getAllTimeOptions( timeZones = timeZonesEx ,
  fullTimePoints = fullTimePointsEx )
# ?SetOfTimePoints # class definition
#str( setOfTimePoints ) # to see all slots in the example
slotNames( setOfTimePoints )
#> [1] ".Data" "fullTimePoints" "nFullTimePoints"
#> [4] "nTimePointsSelect" "nTimePointOptions" "ranking"
```

knitr::kable(head(getData(setOfTimePoints)))

	TimePoint1	TimePoint2	TimePoint3	TimePoint4	TimePoint5	TimePoint6
timePointOption1	0.5	1.0	2.0	3	3.5	16
timePointOption2	0.5	1.5	2.0	3	3.5	16
timePointOption3	1.0	1.5	2.0	3	3.5	16
timePointOption4	0.5	1.0	2.5	3	3.5	16
timePointOption5	0.5	1.5	2.5	3	3.5	16
timePointOption6	1.0	1.5	2.5	3	3.5	16

knitr::kable(tail(getData(setOfTimePoints)))

	TimePoint1	TimePoint2	TimePoint3	TimePoint4	TimePoint5	TimePoint6
timePointOption1945	0.5	1.0	2.0	15	15.5	16
timePointOption1946	0.5	1.5	2.0	15	15.5	16
timePointOption1947	1.0	1.5	2.0	15	15.5	16
timePointOption1948	0.5	1.0	2.5	15	15.5	16
timePointOption1949	0.5	1.5	2.5	15	15.5	16
timePointOption1950	1.0	1.5	2.5	15	15.5	16

note 0 never chosen , time 16 always included

4.4 Rank time points

To rank the timePoint options inside a SetOfTimePoints object, we first need to simulate PkData.

```
model <- getExamplePkModel()
fullTimePoints <- getTimePoints(setOfTimePoints)
pkDataForTimePoints <- getPkData(pkModel = model, timePoints = fullTimePoints,
    nSubjectsPerScheme = 5, nSamples = settings$nSamples )
plotObject(pkDataForTimePoints, nCurves = 5)</pre>
```



This is just small number of samples, in reality one would use a larger number such as 1000.

We can than use the rank function to find the optimal time points:

name	criterion	rank
timePointOption1306	0.0055133	1
timePointOption1216	0.0055254	2
timePointOption1300	0.0055293	3
timePointOption1307	0.0055303	4
timePointOption1217	0.0055424	5
timePointOption1301	0.0055463	6

```
#knitr::kable( tail( rankingTimePoints ) )
indTimeChoice <- getTopNRanking( rankingTimePoints , 1 )
bestTimeChoice <- setOfTimePoints[ indTimeChoice , ]
bestTimeChoice
#> TimePoint1 TimePoint2 TimePoint3 TimePoint4 TimePoint5 TimePoint6
#> 0.5 1.0 2.5 8.0 14.5 16.0
```

4.5 Generate possible schemes

timePointsChoice <- bestTimeChoice</pre>

To generate schemes we can define additional constraints:

```
constraintsExample <- getConstraintsExample()[c( 2 , 4 ) , ]
knitr::kable( constraintsExample )</pre>
```

	check	level	value
2	maxConsecSamples	subject	3
4	minObsPerTimePoint	scheme	2

Constraints are defined on 2 levels: *subject* or *scheme*.

```
setOfSchemes <- getSetOfSchemes( minNSubjects = 4 , maxNSubjects = 5 ,
minObsPerSubject = 4 , maxObsPerSubject = 5 ,
timePoints = timePointsChoice , constraints = constraintsExample ,
maxRepetitionIndSchemes = 1 , maxNumberOfSchemesBeforeChecks = 10<sup>8</sup> )
slotNames( setOfSchemes )
#> [1] ".Data" "timePoints" "nSchemes"
#> [4] "nSubjects" "designConstraints" "ranking"
```

The number of combinations can get get very large especially with maxRepetitionIndSchemes > 1.

4.6 Rank schemes

To rank schemes, we need matching Pkdata (number of animals and timePoints):

```
timePointsEx <- getTimePoints( setOfSchemes )
pkData <- getPkData( pkModel, timePoints = timePointsEx ,
    nSubjectsPerScheme = 5 , nSamples = settings$nSamples )
plotObject( pkData , nCurves = 7 , addZeroIsZero = TRUE )</pre>
```



To rank schemes, we have to define an objective function, based on the a scheme based statistic (AUC, ...) a weight representing its relative importance.

```
exampleObjective <- data.frame(
  criterion = c( "auc" , "cMax" , "tMax" ) ,
  weight = c( 9 , 1, 1 ) )
knitr::kable( exampleObjective )
```

criterionweightauc9cMax1tMax1

But be carefull cMax and tMax might be very variable when multiple doses are administered.

```
setOfSchemesRanked <- rankObject(setOfSchemes, pkData = pkData,
    objective = exampleObjective, varianceMeasure = "var", scaleWith = "max")
```

#> start Ranking Schemes on cluster with 1 cores
schemeRanking <- getRanking(setOfSchemesRanked)
knitr::kable(head(schemeRanking))</pre>

name	var_auc	var_cMax	var_tMax	criterion	rank
scheme2174	0.0038693	8.35e-05	5.923712	0.3944018	1
scheme1117	0.0038757	7.02e-05	7.022727	0.3984378	2
scheme1083	0.0039049	7.02e-05	7.022727	0.4006205	3
scheme1520	0.0040224	7.02e-05	7.022727	0.4093937	4
scheme 1554	0.0040353	7.02e-05	7.022727	0.4103594	5
scheme1759	0.0041947	7.51e-05	6.038283	0.4151664	6

knitr::kable(tail(schemeRanking))

	name	var_auc	var_cMax	var_tMax	criterion	rank
2272	scheme92	0.0109541	0.0001181	6.096364	0.9445893	2272
2273	scheme295	0.0105663	0.0001506	7.516439	0.9478580	2273
2274	scheme367	0.0104885	0.0001597	7.754015	0.9494683	2274
2275	scheme218	0.0107443	0.0001266	8.381313	0.9563801	2275
2276	scheme219	0.0109218	0.0001298	7.503510	0.9626993	2276
2277	scheme365	0.0108802	0.0001506	7.516439	0.9713021	2277

indTopSchemes	<-	<pre>getTopNRanking(schemeRanking , nSelect = 1)</pre>
indBottomSchemes	<-	<pre>getTopNRanking(schemeRanking , nSelect = 1 , top = FALSE)</pre>
bestScheme	<-	<pre>setOfSchemesRanked[, , indTopSchemes]</pre>
<pre>knitr::kable(be</pre>	stScheme)

	timePoint1	timePoint2	timePoint3	timePoint4	timePoint5	timePoint6
subject1	TRUE	FALSE	TRUE	TRUE	TRUE	FALSE
subject2	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE
subject3	TRUE	FALSE	FALSE	TRUE	TRUE	TRUE
subject4	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE
subject5	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE

worstScheme <- setOfSchemesRanked[,, indBottomSchemes]
knitr::kable(worstScheme)</pre>

	timePoint1	timePoint2	timePoint3	timePoint4	timePoint5	timePoint6
subject1	TRUE	TRUE	FALSE	TRUE	FALSE	TRUE
subject2	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE
subject3	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE
subject4	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE
subject5	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

5 Advanced options

5.1 Parallelization

Parallelization by forking is supported on linux machines and can be used to seed up simulating pkData, generating or ranking timepoints or schemes. You need to specify the number of cores inside these functions (nCores):

```
setOfSchemesRanked <- rankObject(setOfSchemes, pkData = pkData,
objective = exampleObjective, varianceMeasure = "var", scaleWith = "max",
nCores = 2)
```

5.2 Working with ranges

Using ranges of parameters is also supported, see

?rankObjectWithRange

for details.

6 Memo of main functions

6.1 Data generation

- getExamplePkModel: Get an example of a PkModel
- construct2CompModel Construct your own 2 compartmental model
- *getPkData* to generate data from your a PkModel
- *plotObject* visualize model or data

6.2 Generate and rank time points

- getAllTimeOptions
- getPkData
- rankObject

6.3 Generate and rank schemes

- getSetOfSchemes
- \bullet getPkData
- $\bullet \ \ rankObject$

References

Barnett, Helen, Helena Geys, Tom Jacobs, and Thomas Jaki. 2017. "Optimal Designs for Non-Compartmental Analysis of Pharmacokinetic Studies."

Chapman, Kathryn, Simon Chivers, Dan Gliddon, David Mitchell, Sally Robinson, Tim Sangster, Susan Sparrow, Neil Spooner, and Amanda Wilson. 2014. "Overcoming the Barriers to the Uptake of Nonclinical Microsampling in Regulatory Safety Studies." *Drug Discovery Today* 19 (5). Elsevier: 528–32.

Gabrielsson, Johan, and Daniel Weiner. 2001. *Pharmacokinetic and Pharmacodynamic Data Analysis:* Concepts and Applications. Vol. 1. CRC Press.

Halliwell, Leigh J. 2015. "The Lognormal Random Multivariate." In Casualty Actuarial Society E-Forum, Spring 2015.