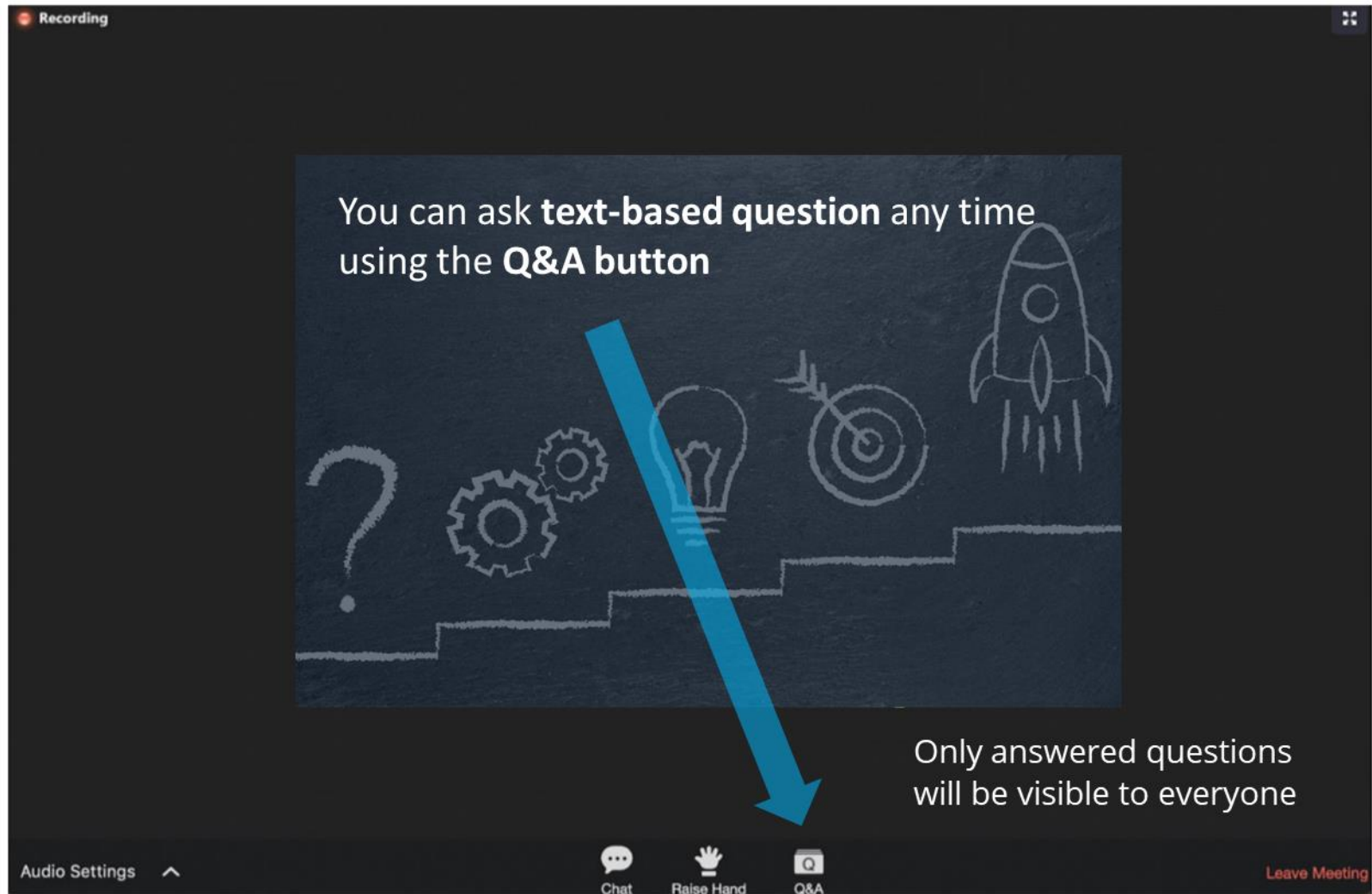


- The webinar starts at **2:00 pm Basel Time**  
**8:00 am East Coast Time**
- Everyone is placed on mute during the webinar
- Webinar material (including presentation) can be downloaded here:  
<https://training.intiquan.com/MIDDmodules/M2.2.zip>
- The webinar will be recorded
  - Recording will be made available on the following link:  
<https://training.intiquan.com/MIDDmodules/M2.2.mp4>
  - Recording available ~1 day after the webinar



# Q&A during Webinar



Recording

You can ask **text-based question** any time using the **Q&A button**

Only answered questions will be visible to everyone

Audio Settings ^

Chat Raise Hand Q&A

Leave Meeting

The screenshot shows a webinar interface with a dark background. At the top left, there is a 'Recording' indicator. The main content area features a chalkboard-style illustration with a sequence of icons: a question mark, two interlocking gears, a lightbulb, a target with an arrow, and a rocket. A large blue arrow points from the text 'Q&A button' to the 'Q&A' button in the bottom toolbar. The bottom toolbar includes 'Audio Settings', 'Chat', 'Raise Hand', 'Q&A', and a 'Leave Meeting' button.

# Module 2.2

## Basic NLME modeling

IntiQuan Webinar Series on efficient support of  
Model Informed Drug Development (MIDD)

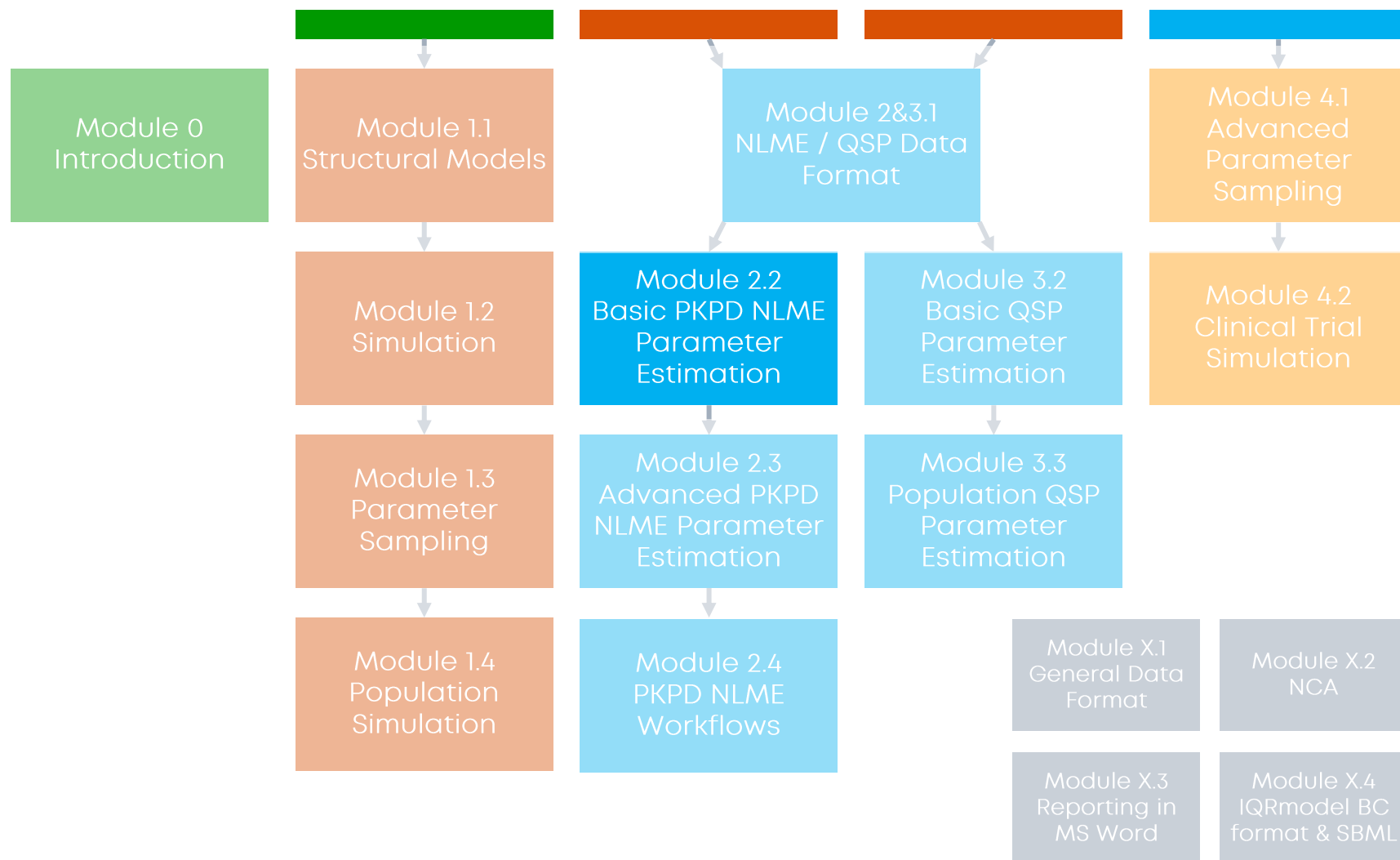
# Outline

1. Background
2. NLME Project definition explained
3. Examples
4. Special elements
5. Conclusions
6. Outlook webinar modules
7. Q&A

# Background

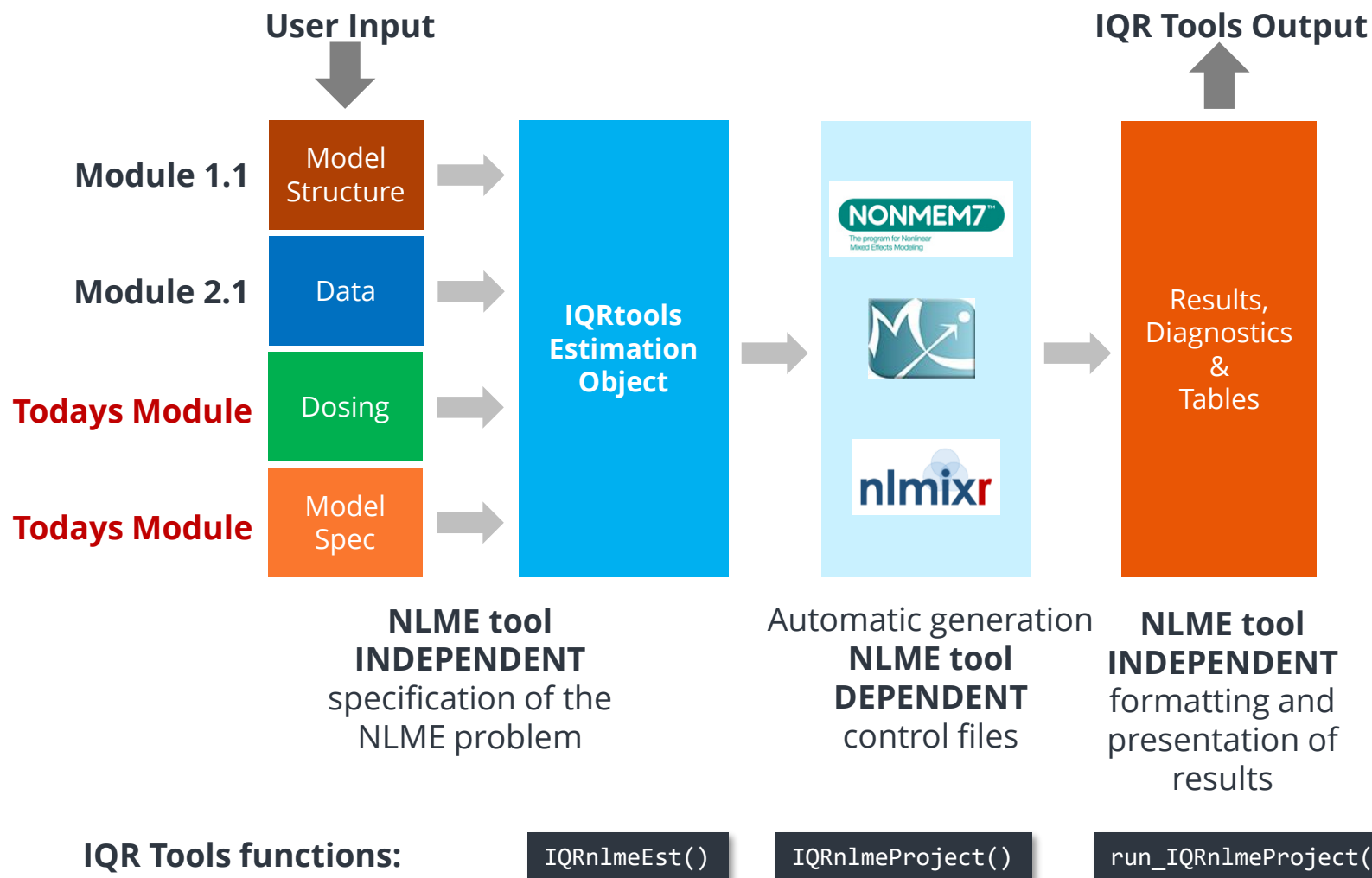
# Overview of Webinar Series by IntiQuan

IntiQuan Webinar Series on efficient support of Model Informed Drug Development (MIDD)





# General IQR Tools approach at NLME modeling



# Goals of this module

- **You will have seen examples on how to**
  - ✓ Easily setup NLME parameter estimations based on
    - ✓ A dataset (see Module 2.1)
    - ✓ A structural model (see Module 1.1)
    - ✓ A flexible and powerful description of the error model, statistical model, covariate model, etc.
  - ✓ Automatic conversion to NONMEM, MONOLIX, and NL MIXR projects
  - ✓ Execution of the NLME parameter estimation
  - ✓ Automatic post-processing of results
  - ✓ Comparison of any desired models
  - ✓ Generation of model development tables
  - ✓ Easy access to all generated output tables
- **Giving you the possibility to define own models and perform own parameter estimations**



# Download of webinar material

- The Webinar material is available as a convenient download
- After installation of IQR Tools, simply type the following

```
library(IQRtools)  
install_MIDDmodule("2.2")
```

- Or download directly from:

```
https://training.intiquan.com/MIDDmodules/M2.2.zip
```

Modules 2.2, 2.3, & 2.4 require presence of NONMEM and/or MONOLIX on the system to fully run the examples

Convenient setup of IQR Tools options, including interfacing with NONMEM and MONOLIX on the system through the IQR Tools options setup:

```
library(IQRtools)
setup_IQRtools()
```

```
49 # UNIX setup
50 # -----
51 # Name of the NONMEM executable or shell script. Required calling syntax:
52 # "command controlfile outputfile"
53 # NONMEM Version 7.2/7.3/7.4 have been tested with IQR Tools.
54 .PATH_SYSTEM_NONMEM <- list(
55   # First entry is used as default version
56   NM743 = 'nmfe74'
57 )
58
59 # MacOS setup
60 # -----
61 # Name of the NONMEM executable or shell script. Required calling syntax:
62 # "command controlfile outputfile"
63 # NONMEM Version 7.2/7.3/7.4 have been tested with IQR Tools.
64 .PATH_SYSTEM_NONMEM <- list(
65   NM74 = "nmfe74"
66 )
67
68 # WINDOWS setup
69 # -----
70 # Path to NONMEM (Version 7.2/7.3/7.4 have been tested) batch files. The info
71 # is provided as a list. By default the first entry is used but the user can
72 # switch when calling the IQRnlmeProject function.
73 .PATH_SYSTEM_NONMEM <- list(
74   NM74 = "C:/nm74g64/run/nmfe74.bat"
75 )
```

**NONMEM7™**

The program for Nonlinear  
Mixed Effects Modeling

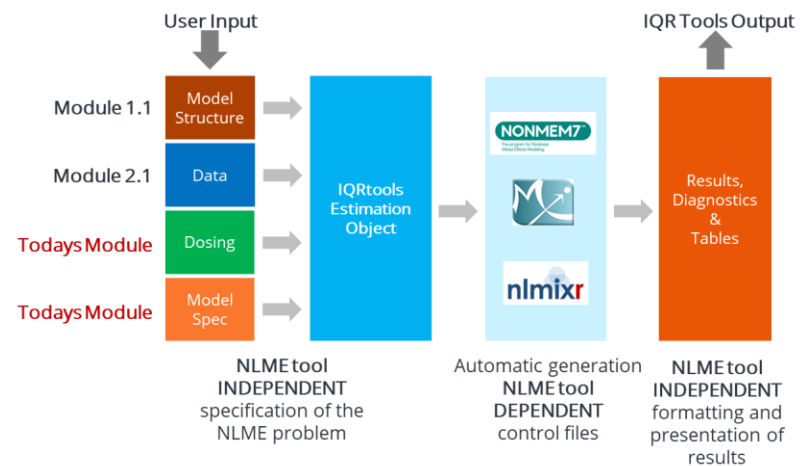


Parallel NONMEM runs handled by IQR Tools as well – can be set up in options as well.  
Requires a shell/batch script available on systems command line with the following calling syntax:

command NRCORES controlfile outputfile

- **Web based demo server:** <http://iqdesktop.intiquan.com:5000>
  - Can be used freely – on a first come first serve basis
  - Fully functional (bring your own license for NONMEM and/or MONOLIX)
- **Local installation easy**
  - Windows installation video:  
[https://iqdesktop.intiquan.com/doc/IQDesktop\\_Installation\\_Windows.mp4](https://iqdesktop.intiquan.com/doc/IQDesktop_Installation_Windows.mp4)
  - Installation guide
    - [Windows](#)
    - [macOS](#)
    - [Linux](#)

# NLME Project definition explained



# Structural model definition

## Example 1

- An IQRmodel in the syntax defined in Module 1.1 defines the structural model to be used for NLME parameter estimation
  - INPUT1-INPUT $n$  identifiers link model to dosing records in the dataset
  - OUTPUT1-OUTPUT $n$  identifiers link model to observation records in the dataset

```
# Load structural model based on IQRmodel syntax
model <- IQRmodel("example_1_model.txt")
```

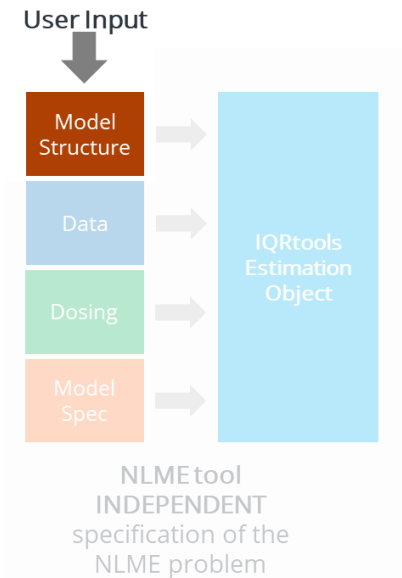
example\_1.R

```
***** MODEL STATES
d/dt(Ad) = -ka*Ad + INPUT1
d/dt(Ac) = ka*Ad - CL/Vc*Ac

***** MODEL PARAMETERS
ka = 1    # Absorption rate parameter (1/hours)
CL = 3    # Apparent clearance (L/hours)
Vc = 60   # Apparent central volume (L)

***** MODEL VARIABLES
# Calculate plasma concentration in ug/mL
Cc = Ac/Vc
# Assign output variable for matching with dataset
OUTPUT1 = Cc # Plasma concentration
```

example\_1\_model.txt



# Data definition

## Example 1

- Data is defined by:
  - **datafile**: Path to an NLME dataset or a corresponding data.frame (see Module 2.1) - required
  - **covNames** / **catNames**: Vectors of column names in dataset to consider as candidate continuous / categorical covariates - optional (default: NULL)
  - **regressorNames**: Vector of model parameter names used as regressors and defined in the dataset (default: NULL)

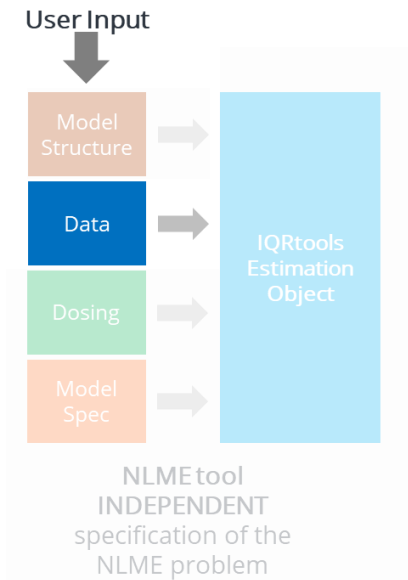
```
# OPTION 1: Define dataset by path
data <- data_IQRest(
  datafile = "example_1_data.csv",
  covNames = c("WT0"),
  catNames = c("SEX"),
  regressorNames = NULL
)
```

Uses data as define in previously generated modeling dataset (suggested for regulatory modeling)

```
# OPTION 2: Define dataset by data.frame
dataset <- IQRloadCSVdata("example_1_data.csv")

data <- data_IQRest(
  datafile = dataset,
  covNames = c("WT0"),
  catNames = c("SEX"),
  regressorNames = NULL
)
```

Allows ad-hoc modifications of the data (good for exploratory use)



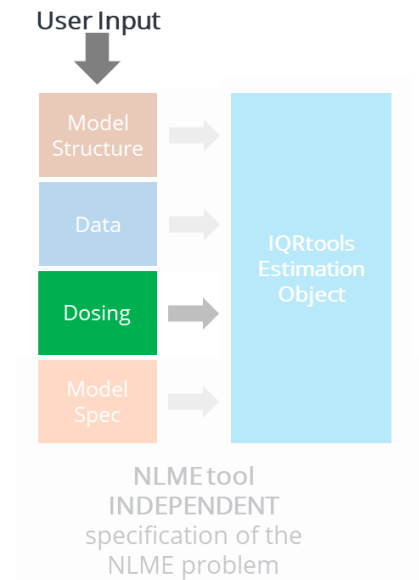
# Dosing definition

## Example 1

- Some information related to the dosing needs to be provided for each input present in the model.

```
# Define dosing
dosing <- dosing_IQRest(
  INPUT1 = c(type="BOLUS",Tlag="Tlag1")
)
```

example\_1.R

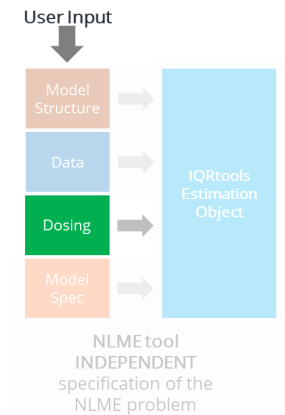




# Dosing definition - Example with all general possibilities

```
# Define dosing
dosing <- dosing_IQRest(
  INPUT1 = c(type="BOLUS"),
  INPUT2 = c(type="BOLUS", Tlag="Tlag1"),
  INPUT3 = c(type="INFUSION"),
  INPUT4 = c(type="INFUSION", Tlag="Tlag1"),
  INPUT5 = c(type="ABSORPTION0", Tk0="TK05"),
  INPUT6 = c(type="ABSORPTION0", Tk0="TK06", Tlag="Tlag6"),
  ...
)
```

- => List with all INPUTs in the model
- Value of list entries is a vector with 3 possible named arguments
  - **type:** String with "BOLUS", "INFUSION", "ABSORPTION0"
    - "BOLUS" and "INFUSION" can be used interchangeably as administration time driven by the TINF column in the dataset (INFUSION with TINF=0 => BOLUS)
  - **Tlag:** String with name of Tlag parameter in the model
    - Can be left undefined if no lag time to be considered (by default lag time=0)
    - If lag time should be considered for estimation, then it has to be defined here
  - **Tk0:** String with 0-order absorption time parameter in the model
    - Only to be defined if type="ABSORPTION0"
    - In case of type="ABSORPTION0" the absorption time is estimated and not taken from the dataset
    - Parameter for 0 order absorption time needs to be coded in the MODEL PARAMETERS section in the model



# Additional model specification

## Example 1

- Define the statistical and residual error model

```
# Define model specification
modelSpec <- modelSpec_IQRest(
  # Define initial guesses for fixed effects
  # And names of parameters to be considered for estimation
  POPvalues0      = c(ka=1,  CL=1,  Vc=10),

  # Define if fixed effect is estimated or not
  POPestimate     = c(ka=1,  CL=1,  Vc=1),

  # Define the distribution of the individual parameters
  IIVdistribution = c(ka="L", CL="L", Vc="L"),

  # Define initial guesses for the random effects
  IIVvalues0      = c(ka=0.5, CL=0.5, Vc=0.5),

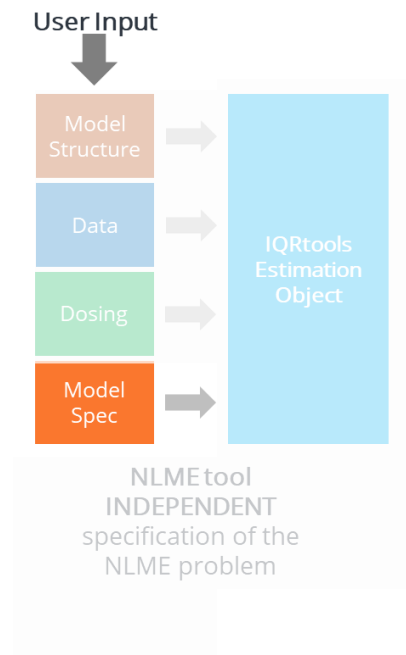
  # Define if random effect is estimated or not
  IIVestimate     = c(ka=1,  CL=1,  Vc=1),

  # Define error model
  errorModel = list(
    OUTPUT1 = c("abs",0.3)
  )
)
```

} Required

} Optional

example\_1.R



There are many more optional arguments – which we will see in Module 2.

# Additional model specification - more in detail

## Example 1

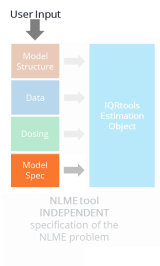
- **POPvalues0** entry required

- Defining the model parameters to be considered for estimation
- Other parameters that are present in the model will be kept on the values defined in the model

```
# Define model specification
modelSpec <- modelSpec_IQRest(
  # Define initial guesses for fixed effects
  # And names of parameters to be considered for estimation
  POPvalues0 = c(ka=1, CL=1, Vc=10)
)
```

example\_1.R

- Vector with named entries - names corresponding to parameter names
- Values represent initial guesses for the **fixed effects**



# Additional model specification - more in detail

## Example 1

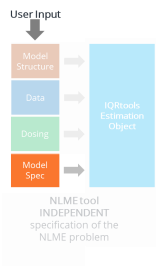
- **POPestimate** entry optional

- Defining the fixed effects that are estimated or kept fixed

```
# Define model specification
modelSpec <- modelSpec_IQRest(
  ...
  # Define if fixed effect is estimated or not
  # 1: estimate it,
  # 0: keep it fixed on initial guess
  POPestimate = c(ka=0, CL=1, Vc=1),
  ...
)
```

example\_1.R

- Vector with named entries - names corresponding to parameter names
- Values:
  - 0: keep parameter fixed on initial guess
  - 1: estimate parameter
  - Default if not provided: 1 (estimated)



# Additional model specification - more in detail

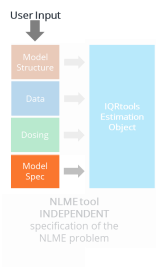
## Example 1

- **IIVdistribution** entry optional
  - Defining the individual parameter distributions

```
# Define model specification
modelSpec <- modelSpec_IQRest(
  ...
  # Define the distribution of the individual parameters
  # N: normally distributed (positive and negative values possible)
  # L: log-normally distributed (positive values only)
  # G: logit-normally distributed (between 0 and 1 only)
  IIVdistribution = c(ka="L", CL="N", Vc="G"),
  ...
)
```

example\_1.R

- Vector with named entries - names corresponding to parameter names
- Values:
  - "N": Normal distribution
  - "L": Log-Normal distribution
  - "G": Logit-Normal distribution
  - Default if not provided: "L"



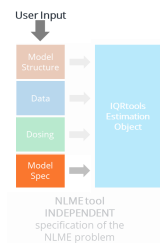
# Additional model specification - more in detail

## Example 1

- **IIVvalues0** entry optional
  - Defining the initial conditions for the **random effects**

```
# Define model specification
modelSpec <- modelSpec_IQRest(
  ...
  # Define initial guesses for the random effects (standard deviation)
  IIVvalues0      = c(ka=0.5, CL=0.5, Vc=0.5),
  ...
)
```

example\_1.R



- Vector with named entries - names corresponding to parameter names
- Values:
  - Initial guesses for the **standard deviation** of the **random effects**
  - Default if not provided: 0.5

# Additional model specification - more in detail

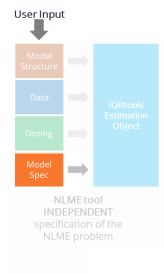
## Example 1

- **IIVestimate** entry optional
  - Defining the random effects that are estimated or kept fixed

```
# Define model specification
modelSpec <- modelSpec_IQRest(
  ...
  # Define if random effect is estimated or not
  # 1: estimate it,
  # 2: keep it fixed on initial guess,
  # 0: no random effect (independent of IIVvalues0 setting)
  IIVestimate = c(ka=0, CL=1, Vc=2),
  ...
)
```

example\_1.R

- Vector with named entries - names corresponding to parameter names
- Values:
  - 0: no random effect on parameter (independent on value set in IIVvalues0)
  - 1: estimate parameter
  - 2: keep parameter fixed on initial guess
  - Default if not provided: 1 (estimated)





# Additional model specification - more in detail

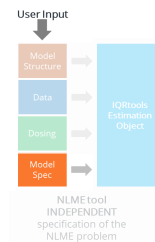
## Example 1

- **errorModel** entry optional

- Defining the residual error model to be used on each OUTPUT

```
# Define model specification
modelSpec <- modelSpec_IQRest(
  ...
  # Define error model - 3 options available
  # OUTPUT1 = c("abs",0.3) - additive with initial guess of sd
  # OUTPUT1 = c("rel",1) - proportional with initial guess of sd
  # OUTPUT1 = c("absrel",0.3,1) - additive-proportional with add/prop guess
  errorModel = list(
    OUTPUT1 = c("abs",0.3)
  )
  ...
)
```

example\_1.R



- List with named entries - names corresponding to OUTPUT names
- Values: vectors with 2-3 elements
  - `c("abs",0.3)` - additive error with initial guess of standard deviation
  - `c("rel",0.5)` - proportional error with initial guess of standard deviation
  - `c("absrel",0.3,1)` - additive/proportional error with initial guesses of standard deviation for add/prop part
  - Default: absolute error model

# Additional model specification

- More optional settings are available
- These will be covered later
- Detailed documentation available in the R help functionality

modelSpec\_IQRest {IQRtools}

R Documentation

Defines a modelSpec list for use in setting up estimation objects.

## Description

Can be used for syspharm and NLME type of estimation objects. This is a convenience function to allow better documentation of the elements that can be in the modelSpec argument for IQRnlmeEst and IQRsysEst. No consistency checks are done at this point. This is conducted in the IQRnlmeEst and IQRsysEst functions.

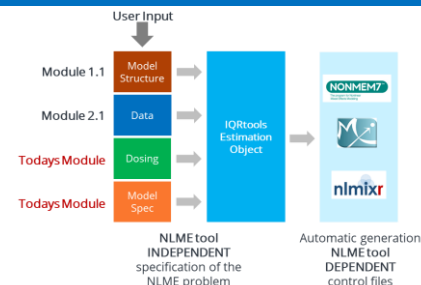
## Usage

```
modelSpec_IQRest(  
  POPvalues0,  
  POPestimate = NULL,  
  IIVdistribution = NULL,  
  IIVvalues0 = NULL,  
  IIVestimate = NULL,  
  IOVvalues0 = NULL,  
  IOVestimate = NULL,  
  errorModel = NULL,  
  covarianceModel = NULL,  
  covariateModel = NULL,  
  covariateModelValues = NULL,  
  COVestimate = NULL,  
  COVcentering = NULL,  
  PriorVarPOP = NULL,  
  PriorVarCovariateModelValues = NULL,  
  PriorDFerrorModel = NULL,  
  PriorIIV = NULL,  
  PriorDFIIV = NULL  
)
```

# Generation of an NLME Project

## Example 1

- Two step approach
  - **Step 1:** Combine model, data, dosing, and model specification into an "estimation object" - which still is **estimation tool independent**
  - **Step 2:** Convert the estimation object in an "NLME Project" that is **tool dependent**. Currently supported tools: NONMEM, MONOLIX, NLMIXR (V1)



```
# Generate estimation object (tool independent)
est <- IQRnlmeEst(model,dosing,data,modelSpec)

# Generate models in NLMIXR, NONMEM, MONOLIX (tool dependent)
IQRnlmeProject(
  est,
  projectPath = "Models/MODEL_01_NONMEM",
  tool = "NONMEM",
  comment="NONMEM - no lag time, add error, no covariates"
)
```

example\_1.R

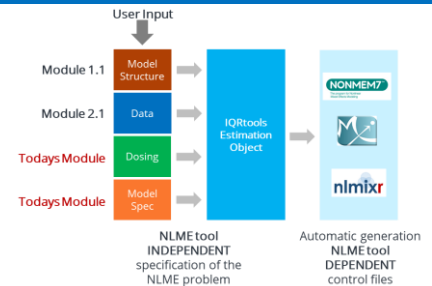
Binding information together in an estimation object. **This step also serves to check consistency of provided information**

Converts an estimation object into a tool dependent project. Allows control over tool dependent settings, algorithms

# Generation of an NLME Project

## Example 1

- Different parameter estimation tools (NONMEM, MONOLIX, NL MIXR) handled by defining the name of the tools
  - And potentially (as needed) tool dependent options



```
# NONMEM
IQRnlmeProject(
  est,
  projectPath = "Models/MODEL_01_NONMEM",
  tool = "NONMEM",
)

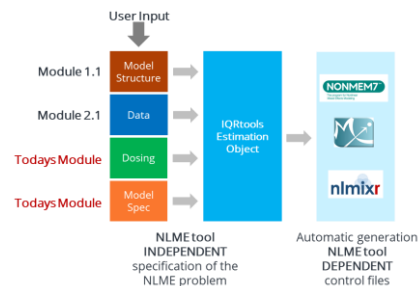
# MONOLIX
IQRnlmeProject(
  est,
  projectPath = "Models/MODEL_01_MONOLIX",
  tool = "MONOLIX",
)

# NL MIXR
IQRnlmeProject(
  est,
  projectPath = "Models/MODEL_01_NL MIXR",
  tool = "NL MIXR",
)
```

example\_1.R

# IQRnlmeProject

- Detailed documentation available ...
- We will use several of these different arguments during the examples



## Creating an IQRnlmeProject object

### Description

This function takes an IQRnlmeEst object and other optional input arguments and exports an NMLE project to a folder. It is this folder that we call an "IQRnlmeProject" object and it is tool specific (e.g. NONMEM, MONOLIX, or NLMIXR). The original IQRmodel used here will be exported to the project folder as well as "model.txt". The project folder will contain in addition the IQRnlmeEst object as a project.est file.

### Usage

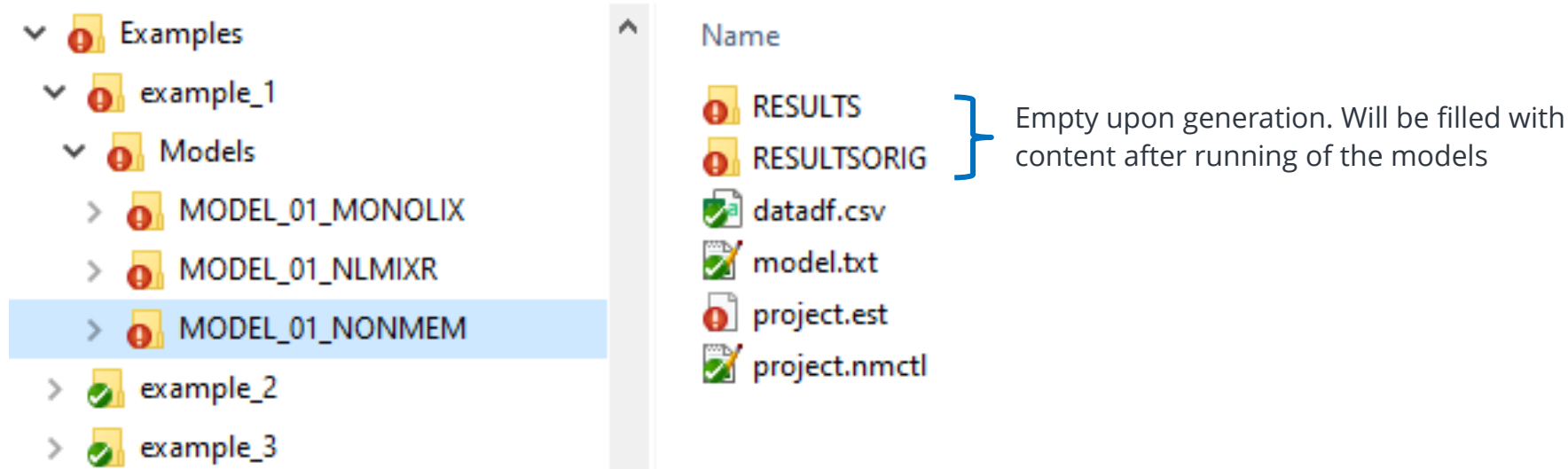
```

IQRnlmeProject(
  est,
  projectPath,
  comment = "",
  tool = "MONOLIX",
  toolVersion = NULL,
  multiTestN = 1,
  multiTestSD = 0.5,
  FLAGanalytic = TRUE,
  keepProjectFolder = FALSE,
  algOpt.SEED = 123456,
  algOpt.K1 = 500,
  algOpt.K2 = 200,
  algOpt.NRCHAINS = NA,
  algOpt.NONMEM.METHOD = "SAEM",
  algOpt.NONMEM.MAKEVAL = 9999,
  algOpt.NONMEM.SIGDIGITS = 3,
  algOpt.NONMEM.PRINT = 1,
  algOpt.NONMEM.COVSTEP_MATRIX = "s",
  algOpt.NONMEM.ADVAN7 = TRUE,
  algOpt.NONMEM.N1 = 1000,
  algOpt.NONMEM.TOL = 6,
  algOpt.NONMEM.SIGL = NULL,
  algOpt.NONMEM.M4 = FALSE,
  algOpt.NONMEM.FOCEIOFV = FALSE,
  algOpt.NONMEM.IMPORTANCESAMPLING = TRUE,
  algOpt.NONMEM.IMP_ITERATIONS = 10,
  algOpt.NONMEM.ITS = TRUE,
  algOpt.NONMEM.ITS_ITERATIONS = 10,
  algOpt.NONMEM.WRES = NULL,
  algOpt.NONMEM.PRED = NULL,
  algOpt.NONMEM.RES = NULL,
  algOpt.MONOLIX.individualParameters = "conditionalMode",
  algOpt.MONOLIX.indivMCMClength = 50,
  algOpt.MONOLIX.indivNsim = 10,
  algOpt.MONOLIX.indivRatio = 0.05,
  algOpt.MONOLIX.logLikelihood = "Linearization",
  algOpt.MONOLIX.fim = "Linearization",
  algOpt.MONOLIX.variability = "FirstStage",
  algOpt.MONOLIX.startTime = NULL,
  algOpt.MONOLIX.STIFF = TRUE,
  algOpt.NLMIXR.method = "SAEM",
  algOpt.NLMIXR.control = NULL,
  verbose = TRUE
)
  
```

# Looking at an IQRnlmeProject

## Example 1

- An IQRnlmeProject is a folder, containing
  - The generated NONMEM (or MONOLIX or NLMIXR) control file
  - The structural IQRmodel
  - And additional information for book-keeping



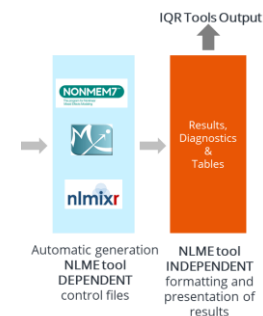
# Execution of an NLME Project

## Example 1

- IQRnlmeProjects can be executed

```
# Run the generated NONMEM model
run_IQRnlmeProject("Models/MODEL_01_NONMEM")
run_IQRnlmeProject("Models/MODEL_01_MONOLIX")
run_IQRnlmeProject("Models/MODEL_01_NLMIXR")
```

example\_1.R



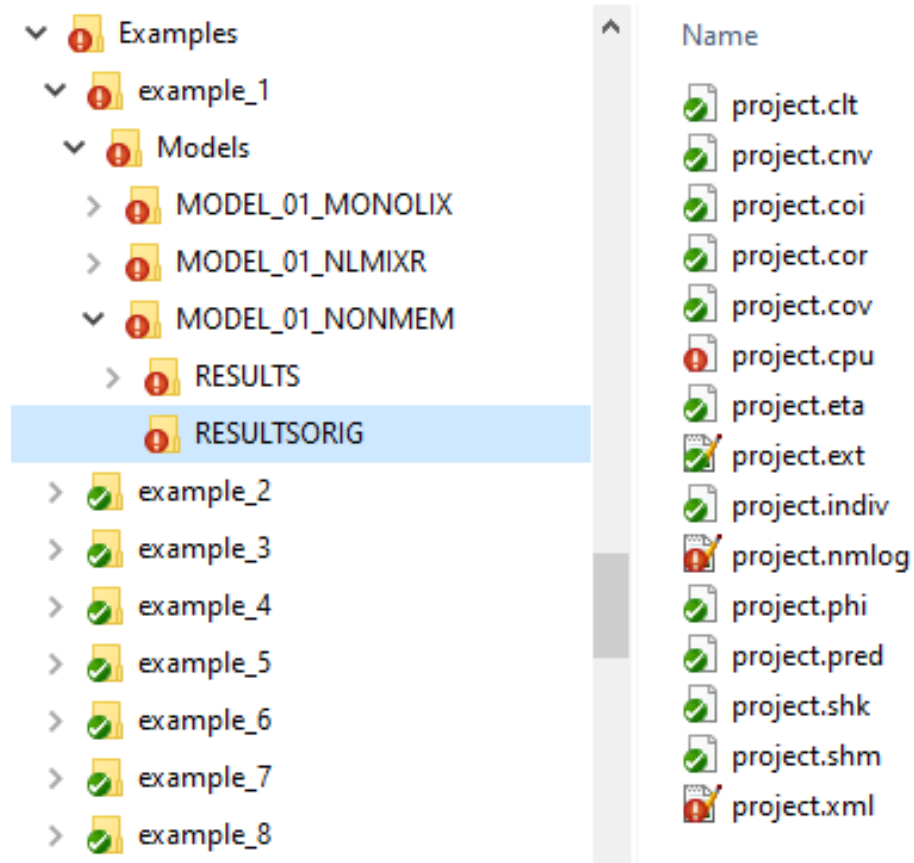
- run\_IQRnlmeProject
  - Executes the parameter estimation
  - Post-processes the results in a standard format



# Results of an IQRnlmeProject run

## Example 1

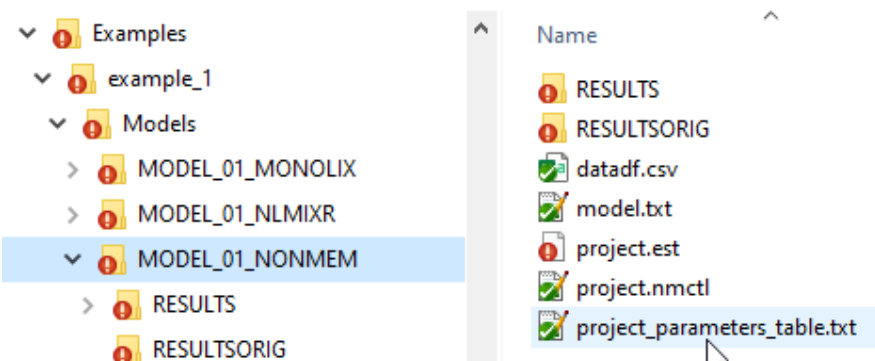
- Standard Tool dependent output (RESULTSORIG)



# Results of an IQRnlmeProject run

## Example 1

- Post-processed outputs
  - Parameter estimates table

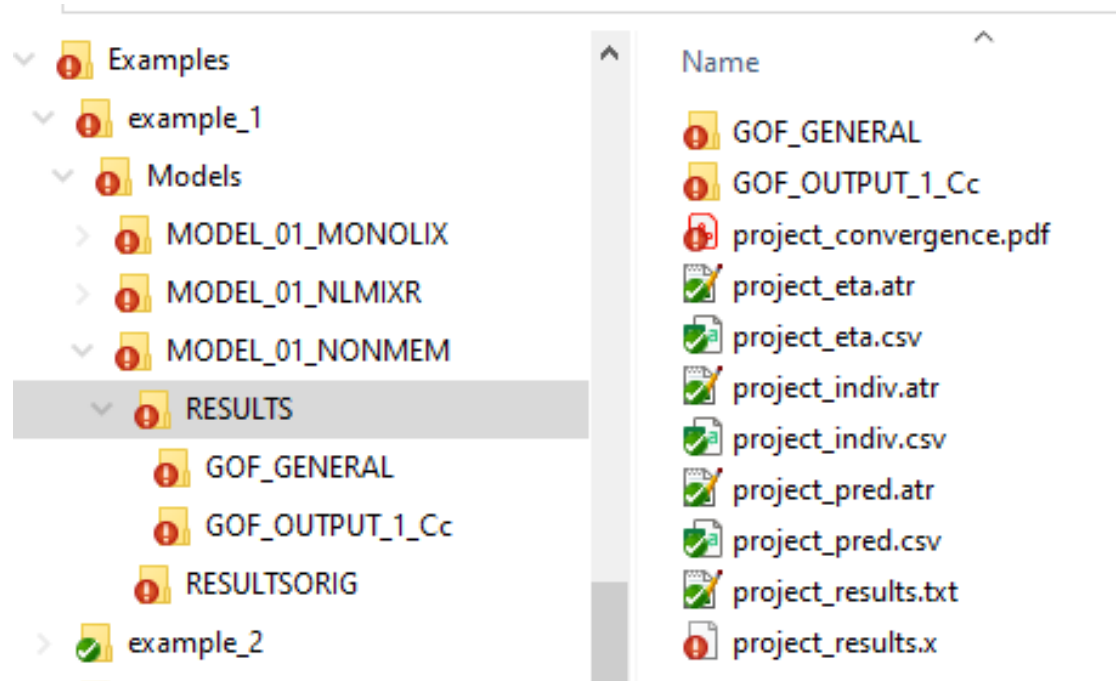


PARAMETER	VALUE	RSE	SHRINKAGE	COMMENT
**Typical parameters**				
ka	1.59	32.6%	-	Absorption rate parameter (1/hours)
CL	3.48	43.1%	-	Apparent clearance (L/hours)
Vc	40.2	24.3%	-	Apparent central volume (L)
**Inter-individual variability**				
omega (ka)	0.66	33.4%	6.7%	LogNormal
omega (CL)	0.284	66.2%	8.5%	LogNormal
omega (Vc)	0.156	73.8%	11.9%	LogNormal
**Residual Variability**				
error_ADD1	0.735	5.59%	13.2%*	Additive Error (ug/ml) - Plasma concentration
Objective function	127	-	-	-
AIC	141	-	-	-
BIC	161	-	-	-

# Results of an IQRnlmeProject run

## Example 1

- Post-processed outputs
  - Full set of standard diagnostics
    - General (folder GOF\_GENERAL)
    - One folder per observable (here only 1: GOF\_OUTPUT\_1\_Cc)
  - CSV files with standard output
    - Individual parameters
    - Random effects
    - Predictions, etc.



# Summary

- Easy setup and execution of NLME parameter estimation problems
- Automatic post-processing of results allows direct assessment of results to inform the subsequent modeling steps
- Encapsulation of NLME estimation problems in single folders (NLME Projects) allows structured storage
- Easy switching between tools (NONMEM, MONOLIX, NLMIXR) - no expert knowledge in setting up NONMEM control streams or MONOLIX MLXTRAN files required
- => Focus on expertise in NLME modeling, rather than in NONMEM or MONOLIX
- All modeling follows the same approach - the difference being only in the settings (structural model, data, dosing, model specification)

# Examples

# Example 1 – Basic

- Basic 1 compartment PK model with 1st order absorption
  - No lag time
  - Additive residual error model
  - No covariates
  - Consideration of NONMEM, MONOLIX, NL MIXR
- Code shown and discussed in previous section

# Example 2 - Lag time & different error model

- Basic 1 compartment PK model with 1st order absorption
  - Estimation of lag time
  - Additive-proportional residual error model
  - No covariates

```
# Define dosing
# If lag time desired to be estimated/fixed in the model
# then for each INPUTn with lag time the lag time parameter
# name needs to be defined
dosing <- dosing_IQRest(
  INPUT1 = c(type="BOLUS",Tlag="Tlag1")
)

# Define model specification
modelSpec <- modelSpec_IQRest(
  POPvalues0      = c(ka=1, CL=1, Vc=10, Tlag1=0.1),
  POPEstimate     = c(ka=1, CL=1, Vc=1, Tlag1=1),

  IIVdistribution = c(ka="L", CL="L", Vc="L", Tlag1="L"),

  IIVvalues0      = c(ka=0.5, CL=0.5, Vc=0.5, Tlag1=0.5),
  IIVestimate     = c(ka=1, CL=1, Vc=1, Tlag1=1),

  errorModel = list(
    OUTPUT1 = c("absrel",1,0.3)
  )
)
```

example\_2.R



# Example 3 - 0 order absorption & different error model

- Basic 1 compartment PK model with 0 order absorption into central compartment
  - Estimation of lag time
  - Estimation of 0-order absorption time
  - Proportional residual error model
  - No covariates

```
***** MODEL STATES
d/dt(Ac) = - CL/Vc*Ac + INPUT1

***** MODEL PARAMETERS
Tk0 = 1    # 0-order absorption time (hours)
CL  = 3    # Apparent clearance (L/hours)
Vc  = 60   # Apparent central volume (L)
```

example\_3\_model.txt

```
# Define dosing
# In the case of 0-order absorption through a selected INPUTn,
# the name of the model parameter representing the absorption time
# needs to be defined. The parameter has to be defined in the IQRmodel file.
dosing <- dosing_IQRest(
  INPUT1 = c(type="ABSORPTION0", Tlag="Tlag1", Tk0="Tk0")
)

# Define model specification
modelSpec <- modelSpec_IQRest(
  POPvalues0      = c(Tk0=1, CL=1, Vc=10, Tlag1=0.1),
  POPestimate     = c(Tk0=1, CL=1, Vc=1, Tlag1=1),
  IIVdistribution = c(Tk0="L", CL="L", Vc="L", Tlag1="L"),
  IIVvalues0      = c(Tk0=0.5, CL=0.5, Vc=0.5, Tlag1=0.5),
  IIVestimate     = c(Tk0=1, CL=1, Vc=1, Tlag1=1),
  errorModel = list(
    OUTPUT1 = c("rel", 0.3)
  )
)
```

example\_3.R

# Example 4 - IOV

- IOV only supported with IQR Tools through NONMEM
- Requires an OCC column in the dataset

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	
IXGDF	IGNORE	USUBJID	ID	TIME	TIMEPOS	TAD	TIMEUNIT	YTYPE	NAME	VALUE	DV	UNIT	LLOQ	CENS	MDV	EVID	AMT	ADM	II	ADDL	ROUTE	TINF	RATE	DOSE	DURATION	OCC	
1.	.	HS0815-01	1	0	0	0	DAYS	.	HS0815::C	80.1	0	mg	.	0	0	1	1	80.1	1	0	0	IV	0.082639	969.2773	80.1	0.082639	1
2.	.	HS0815-01	1	0.084722	0.084722	0.084722	DAYS	.	HS0815::C	38.1	38.1	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	1
3.	.	HS0815-01	1	0.166667	0.166667	0.166667	DAYS	.	HS0815::C	30.1	30.1	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	1
4.	.	HS0815-01	1	0.333333	0.333333	0.333333	DAYS	.	HS0815::C	29	29	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	1
5.	.	HS0815-01	1	1	1	1	DAYS	.	HS0815::C	25.8	25.8	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	1
6.	.	HS0815-01	1	2	2	2	DAYS	.	HS0815::C	21.6	21.6	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	1
7.	.	HS0815-01	1	7.009028	7.009028	7.009028	DAYS	.	HS0815::C	12.7	12.7	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	1
8.	.	HS0815-01	1	14.03264	14.03264	14.03264	DAYS	.	HS0815::C	11.3	11.3	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	1
9.	.	HS0815-01	1	21.01042	21.01042	21.01042	DAYS	.	HS0815::C	7.6	7.6	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	1
10.	.	HS0815-01	1	28.03264	28.03264	28.03264	DAYS	.	HS0815::C	6.9	6.9	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	1
11.	.	HS0815-01	1	42.02014	42.02014	42.02014	DAYS	.	HS0815::C	5.7	5.7	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	1
12.	.	HS0815-01	1	56.02083	56.02083	56.02083	DAYS	.	HS0815::C	4	4	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	1
13.	.	HS0815-01	1	87.01458	87.01458	87.01458	DAYS	.	HS0815::C	2.2	2.2	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	1
14.	.	HS0815-01	1	112.1917	112.1917	112.1917	DAYS	.	HS0815::C	1.2	1.2	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	1
15.	.	HS0815-01	1	500	500	0	DAYS	.	HS0815::C	80.1	0	mg	.	0	1	1	80.1	1	0	0	IV	0.082639	969.2773	80.1	0.082639	2	
16.	.	HS0815-01	1	500.0847	500.0847	0.084722	DAYS	.	HS0815::C	22.86	22.86	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	2
17.	.	HS0815-01	1	500.1667	500.1667	0.166667	DAYS	.	HS0815::C	18.06	18.06	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	2
18.	.	HS0815-01	1	500.3333	500.3333	0.333333	DAYS	.	HS0815::C	17.4	17.4	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	2
19.	.	HS0815-01	1	501	501	1	DAYS	.	HS0815::C	15.48	15.48	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	2
20.	.	HS0815-01	1	502	502	2	DAYS	.	HS0815::C	12.96	12.96	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	2
21.	.	HS0815-01	1	507.009	507.009	7.009028	DAYS	.	HS0815::C	7.62	7.62	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	2
22.	.	HS0815-01	1	514.0326	514.0326	14.03264	DAYS	.	HS0815::C	6.78	6.78	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	2
23.	.	HS0815-01	1	521.0104	521.0104	21.01042	DAYS	.	HS0815::C	4.56	4.56	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	2
24.	.	HS0815-01	1	528.0326	528.0326	28.03264	DAYS	.	HS0815::C	4.14	4.14	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	2
25.	.	HS0815-01	1	542.0201	542.0201	42.02014	DAYS	.	HS0815::C	3.42	3.42	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	2
26.	.	HS0815-01	1	556.0208	556.0208	56.02083	DAYS	.	HS0815::C	2.4	2.4	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	2
27.	.	HS0815-01	1	587.0146	587.0146	87.01458	DAYS	.	HS0815::C	1.32	1.32	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	2
28.	.	HS0815-01	1	612.1917	612.1917	112.1917	DAYS	.	HS0815::C	0.72	0.72	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	80.1	0	2
29.	.	HS0815-01	2	0	0	0	DAYS	.	HS0815::C	67.3	0	mg	.	0	1	1	67.3	1	0	0	IV	0.083333	807.6	67.3	0.083333	1	
30.	.	HS0815-01	2	0.084722	0.084722	0.084722	DAYS	.	HS0815::C	27.6	27.6	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	67.3	0	1
31.	.	HS0815-01	2	0.166667	0.166667	0.166667	DAYS	.	HS0815::C	27.6	27.6	ug/mL	0.2	0	0	0	0	0	.	0	0	0	0	0	67.3	0	1

example\_4/Data/dataNLME.csv

# Example 4 - IOV

- 2 compartmental distribution model with IV infusion
  - Consideration of IOV
  - Parallel execution of NONMEM run

```
# Define model specification - including IOV information
modelSpec <- modelSpec_IQRest(
  POPvalues0      = c(CL=0.5,   Vc=3,   Q1=0.5,   Vp1=3),
  POPEstimate     = c(CL=1,     Vc=1,   Q1=1,     Vp1=1),

  IIVdistribution = c(CL="L",   Vc="L", Q1="L",   Vp1="L"),

  IIVvalues0      = c(CL=0.3,   Vc=0.3, Q1=0.3,   Vp1=0.3),
  IIVestimate     = c(CL=1,     Vc=1,   Q1=2,     Vp1=1),

  # Define initial guesses for IOV standard deviation
  # Distribution of IOV same as for IIV
  IOVvalues0      = c(CL=0,     Vc=0.2, Q1=0,     Vp1=0.6),

  # Define if IOV is estimated
  # 1: estimate it,
  # 2: keep it fixed on initial guess,
  # 0: no IOV random effect (independent of IOVvalues0 setting)
  IOVestimate     = c(CL=1,     Vc=1,   Q1=1,     Vp1=1),

  errorModel = list(
    OUTPUT1 = c("rel",0.3)
  )
)
```

example\_4.R

```
# Run NONMEM model with IOV
run_IQRnlmeProject("Models/MODEL_04",Nparallel = 4)
```

# Example 5 - Covariates

- 2 compartmental distribution model with first order absorption
- Definition of covariates (using default initial guesses and centering)

```
# Model specification
modelSpec1 <- modelSpec_IQRest(

# Typical subject parameters
POPvalues0 = c(kabs = 0.25, CL = 30, Vc = 30, Q1 = 25, Vp1 = 2300),
POPEstimate = c(kabs = 1, CL = 1, Vc = 1, Q1 = 1, Vp1 = 1),

# Between subject variability
IIVdistribution = c(kabs = "L", CL = "L", Vc = "L", Q1 = "L", Vp1 = "L"),
IIVvalues0 = c(kabs = 0.5, CL = 0.5, Vc = 0.5, Q1 = 0.5, Vp1 = 0.5),
IIVestimate = c(kabs = 1, CL = 1, Vc = 1, Q1 = 1, Vp1 = 1),

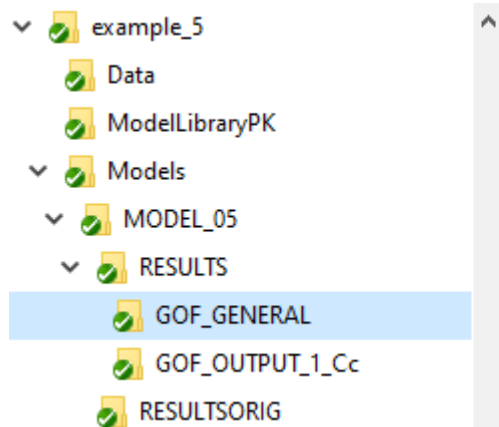
# Covariate model
# Defined by a list with named elements.
# - names are the parameters on which to add covariates
# - values are vectors with names of the candidate covariates
# - IQR Tools implements:
#   - Continuous covariates by standard power functions: *(cov/REF)^beta
#     - Default reference value REF: median in the dataset
#   - Categorical covariates: *exp(betaX*(cat=categoryX))
#     - Default reference category: smallest numerical category in cat
# - Use of MU referencing limits options for covariate implementation
# - It is always possible to code covariate relationships also directly
#   into the structural model - but with SAEM care should be taken - e.g.
#   a covariate coefficient then benefits from variability (fixed) based on
#   convergence considerations.
covariateModel = list(
  kabs = c("AGE", "SEX"),
  CL = c("SEX", "WT0"),
  Vc = c("SEX")
),

# Error model
errorModel = list(OUTPUT1 = c(type="rel", rel0 = 0.2))
)
```

example\_5.R

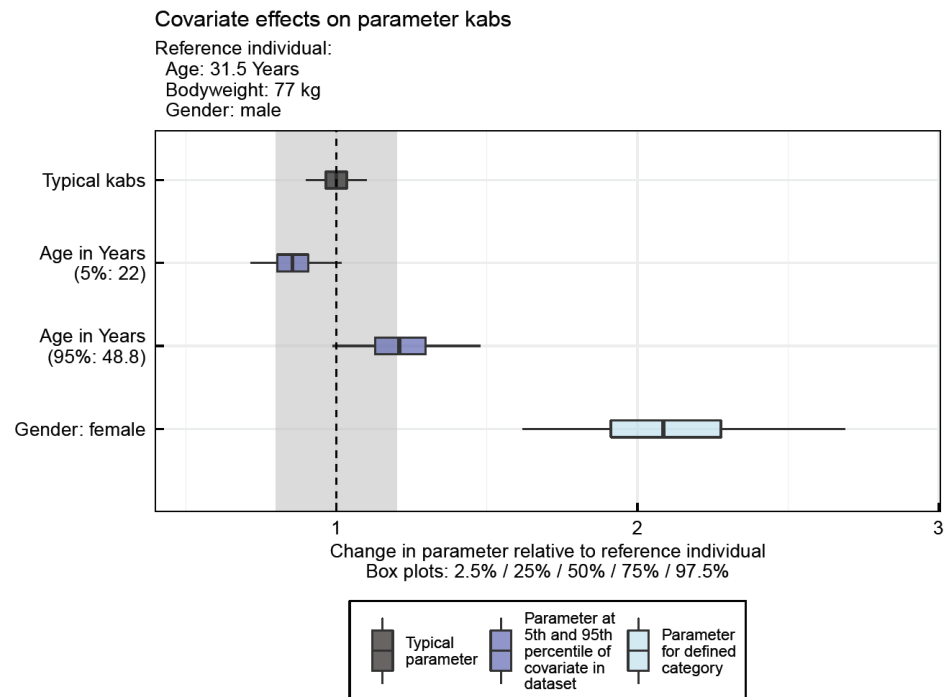
# Example 5 - Covariates

- Let's look at the output in the RESULTS folder



Name

- 01\_Random\_Effects.pdf
- 02\_ETAs\_vs\_COVs.pdf
- 03\_COV\_Impact\_Parameters.pdf
- 03\_COV\_Impact\_Parameters.txt



# Example 6 - Covariates with more control

- 2 compartmental distribution model with first order absorption
  - Definition of covariates
  - Initial guesses
  - Estimation/fixed
  - Centering of covariates

```
# Covariate model
covariateModel = list(
  kabs = c("AGE", "SEX"),
  CL   = c("SEX", "WT0"),
  Vc   = c("SEX")
),

# Covariate coefficient values
# -----
# Initial guesses for covariate coefficients can be provided in the
# following manner. In case of multi-level categorical covariates
# the same value will be used as starting guess for all levels different
# from reference.
covariateModelValues = list(
  kabs = c("AGE"=0.1, "SEX"=2),
  CL   = c("SEX"=2, "WT0"=0.75),
  Vc   = c("SEX"=0.3)
),

# Covariate coefficient estimation settings
# -----
# Value of 1 means the coefficient is estimated.
# Value of 0 means the coefficient is fixed.
COVestimate = list(
  kabs = c("AGE"=0, "SEX"=1),
  CL   = c("SEX"=1, "WT0"=0),
  Vc   = c("SEX"=1)
),

# Covariate centering
# -----
# Continuous covariates can be centered around a defined value
# Categorical covariates can have a defined reference value
# Continuous covariates that are not centered in this manner will
# be centered around the median in the dataset
# Categorical covariates that are not centered in this manner will
# obtain the smallest category value as reference value
COVcentering = c("WTKG"=70, SEX=1),
```

# Example 7 - Correlation of random effects

- 2 compartmental distribution model with first order absorption
  - Estimation of correlation of random effects
  - 3 examples included with different settings

```
# Model specification
modelSpec1 <- modelSpec_IQRest(

  ...

  # Covariance model
  # This example will estimate the correlation between random effects of
  # CL and Vc. Starting guesses cannot be provided - but also are not
  # really needed as the estimation algorithms can do a good job on their
  # estimation.
  covarianceModel = c("CL,Vc"),

  ...
)
```

```
# Model specification
modelSpec1 <- modelSpec_IQRest(
  ...
  covarianceModel = c("CL,Vc,Q1"),
  ...
)
```

```
# Model specification
modelSpec1 <- modelSpec_IQRest(
  ...
  covarianceModel = c("CL,Vc,Q1","Vp1,ka"),
  ...
)
```

example\_7.R

# Example 8 - Multi-INPUT & Multi-OUTPUT

- Dual PK models with 1st order, 0th order, and bolus/infusion administration
  - Totally constructed example with 6 INPUTs and 2 OUTPUTs
  - Translated to NONMEM ADVAN7 (optionally: ADVAN5)

```
***** MODEL STATES
```

```
d/dt(Ad) = - ka*Ad + FABS1*INPUT1
```

```
d/dt(Ac) = ka*Ad - Q1/Vc*Ac + Q1/Vp1*Ap1 - CL/Vc*Ac + FIV*INPUT2 + FABS0*INPUT3
```

```
d/dt(Ap1) = Q1/Vc*Ac - Q1/Vp1*Ap1
```

```
d/dt(A2d) = - ka2*A2d + F2ABS1*INPUT5
```

```
d/dt(A2c) = ka2*A2d - Q21/V2c*A2c + Q21/V2p1*A2p1 - CL2/V2c*A2c + F2IV*INPUT6 + F2ABS0*INPUT4
```

```
d/dt(A2p1) = Q21/V2c*A2c - Q21/V2p1*A2p1
```

```
***** MODEL VARIABLES
```

```
Cc = Ac/Vc
```

```
Cc2 = A2c/V2c
```

```
OUTPUT1 = Cc
```

```
OUTPUT2 = Cc2
```

example\_8\_model.txt

More than 1 INPUT on same compartment requires special setup of dataset for NONMEM  
For MONOLIX the standard dataset format can be used.



# Example 8 - Multi-INPUT & Multi-OUTPUT

- Define dosing

```
# Define dosing
# This is a purely constructed example ... have a look at the example_8_model.txt file.
# There are 6 INPUTn definitions. There are two cases of more than one INPUTn on the same
# compartment.
dosing      <- list(
  INPUT1 = c(type="BOLUS",Tlag="Tlag1"),
  INPUT2 = c(type="INFUSION"),
  INPUT3 = c(type="ABSORPTION0",Tk0="Tk0",Tlag="Tlag3"),
  INPUT4 = c(type="ABSORPTION0",Tk0="Tk02",Tlag="Tlag4"),
  INPUT5 = c(type="BOLUS",Tlag="Tlag5"),
  INPUT6 = c(type="INFUSION",Tlag="Tlag6")
)
```

example\_8.R

- Define model specification

```
# Define model specification
modelSpec   <- list(

  POPvalues0 = c(CL = 1, Vc = 10, Q1 = 1, Vp1 = 10),

  errorModel = list(
    OUTPUT1 = c("absrel", c(abs0=1,rel0=0.3)),
    OUTPUT2 = c("absrel", c(abs0=1,rel0=0.3))
  )
)
```

example\_8.R

# Example 8 - Multi-INPUT & Multi-OUTPUT

- Modified dataset for NONMEM
  - If more than one INPUT is defined on a compartment, then
    - CMT column instead of YTYPE column needs to be present
      - Number of output if observation, number of state to which to add dose if dosing event.
  - For MONOLIX this modification is not needed

```
# Define data
data      <- list(
  datafile = "dataNLME_CMT.csv"
)
```

example\_8.R

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	IXGDF	USUBJID	ID	TIME	TIMEPOS	TAD	TIMEUNIT	CMT	NAME	VALUE	DV	UNIT	LLOQ	CENS	MDV	EVID	AMT	ADM	TINF	RATE
2	25	X_X2101_0	1	0	0	0	DAYS	1	X:::Dose	300	0	mg	.	0	1	1	300	1	0	0
3	25	X_X2101_0	1	0	0	0	DAYS	2	X:::Dose	300	0	mg	.	0	1	1	300	2	0.035	8571.429
4	25	X_X2101_0	1	0	0	0	DAYS	2	X:::Dose	300	0	mg	.	0	1	1	300	3	0	0
5	26	X_X2101_0	1	0.035	0.035	0.035	DAYS	1	X:::Plasma	146.87	146.87	ug/mL	0.1	0	0	0	0	0	0	0
6	27	X_X2101_0	1	0.5	0.5	0.5	DAYS	1	X:::Plasma	121.98	121.98	ug/mL	0.1	0	0	0	0	0	0	0
7	28	X_X2101_0	1	1	1	1	DAYS	1	X:::Plasma	100.13	100.13	ug/mL	0.1	0	0	0	0	0	0	0
8	29	X_X2101_0	1	2.01	2.01	2.01	DAYS	1	X:::Plasma	74.1	74.1	ug/mL	0.1	0	0	0	0	0	0	0
9	30	X_X2101_0	1	3.997	3.997	3.997	DAYS	1	X:::Plasma	49.59	49.59	ug/mL	0.1	0	0	0	0	0	0	0
10	31	X_X2101_0	1	6.044	6.044	6.044	DAYS	1	X:::Plasma	43.89	43.89	ug/mL	0.1	0	0	0	0	0	0	0
11	32	X_X2101_0	1	9.002	9.002	9.002	DAYS	1	X:::Plasma	35.53	35.53	ug/mL	0.1	0	0	0	0	0	0	0
12	33	X_X2101_0	1	13.059	13.059	13.059	DAYS	1	X:::Plasma	38.19	38.19	ug/mL	0.1	0	0	0	0	0	0	0
13	34	X_X2101_0	1	20.032	20.032	20.032	DAYS	1	X:::Plasma	30.21	30.21	ug/mL	0.1	0	0	0	0	0	0	0
14	35	X_X2101_0	1	27.997	27.997	27.997	DAYS	1	X:::Plasma	20.52	20.52	ug/mL	0.1	0	0	0	0	0	0	0
15	89	X_X2101_0	2	0	0	0	DAYS	5	X:::Dose	444	0	mg	.	0	1	1	444	4	0.035	12685.71
16	89	X_X2101_0	2	0	0	0	DAYS	4	X:::Dose	444	0	mg	.	0	1	1	444	5	0	0
17	89	X_X2101_0	2	0	0	0	DAYS	5	X:::Dose	444	0	mg	.	0	1	1	444	6	0.035	12685.71
18	90	X_X2101_0	2	0.035	0.035	0.035	DAYS	1	X:::Plasma	140.98	140.98	ug/mL	0.1	0	0	0	0	0	0	0
19	91	X_X2101_0	2	0.5	0.5	0.5	DAYS	1	X:::Plasma	130.34	130.34	ug/mL	0.1	0	0	0	0	0	0	0
20	92	X_X2101_0	2	1.003	1.003	1.003	DAYS	1	X:::Plasma	120.06	120.06	ug/mL	0.1	0	0	0	0	0	0	0

## Example 8 - Multi-INPUT & Multi-OUTPUT

- IQR Tools output on the console supports the user in the correct data specification

```
=====
IMPORTANT:
The dataset contains the "CMT" column. In order to correctly map dosing inputs
and observations with the CMT column, please make sure you used the following
entries in the CMT column (keep the ADM column):

Mapping of the INPUTn in the model and the CMT column in the dataset:
  INPUT1 => CMT column value: 1
  INPUT2 => CMT column value: 2
  INPUT3 => CMT column value: 2
  INPUT4 => CMT column value: 5
  INPUT5 => CMT column value: 4
  INPUT6 => CMT column value: 5

Mapping of the OUTPUTn in the model and the CMT column in the dataset:
  OUTPUT1 => CMT column value: 1
  OUTPUT2 => CMT column value: 2
=====
```

# Example 9 – Use of regressor from the dataset

## Application for custom covariate relationship in structural model

- Simple 1 compartment model

- Body weight (WT0) as covariate on CL
- Implemented directly in the structural model
- Individual values for WT0 obtained from WT0 column in the dataset (regressor)

```
***** MODEL STATES

d/dt(Ad) = -ka*Ad + INPUT1
d/dt(Ac) = ka*Ad - CLcov/Vc*Ac

***** MODEL PARAMETERS

ka = 1    # Absorption rate parameter (1/hours)
CL = 3    # Apparent clearance (L/hours)
Vc = 60   # Apparent central volume (L)

betaCLWT0 = 0.75 # Covariate body weight on CL (.)
WT0 = 70 # Body weight provided as regressor from the data (kg)

***** MODEL VARIABLES

# Define Covariate effect on CL
CLcov = CL * (WT0/70)^betaCLWT0
```

example\_9\_model.txt

- Multiple regressors can be used

- Order of regressor definition needs to be the same (in dataset and model)

# Example 9 – Use of regressor from the dataset

## Application for custom covariate relationship in structural model

- Definition of data and model specification to handle regressors

```
# Define dataset
data <- data_IQRest(
  datafile = "example_9_data.csv",
  catNames = c("SEX"), # Vector defining categorical candidate covariate columns in dataset
  regressorNames = c("WT0") # Implementation of covariate as regressor
)

# Define model specification
modelSpec <- modelSpec_IQRest(
  # Implementation of covariate WT0 on CL by regressor in the model
  # Requires definition of covariate coefficient name as normal parameter
  POPvalues0 = c(ka=1, CL=1, Vc=10, betaCLWT0=0.75),
  POPestimate = c(ka=1, CL=1, Vc=1, betaCLWT0=1),
  # If sign for betaCLWT0 is known it is beneficial to implement it as
  # positive parameter, choosing "L" as distribution
  IIVdistribution = c(ka="L", CL="L", Vc="L", betaCLWT0="L"),
  # Small IIV (here 5CV%) for betaCLWT0 - improves convergence
  IIVvalues0 = c(ka=0.5, CL=0.5, Vc=0.5, betaCLWT0=0.05),
  # Do not estimate the IIV for betaCLWT0
  IIVestimate = c(ka=1, CL=1, Vc=1, betaCLWT0=2),
  errorModel = list(
    OUTPUT1 = c("abs",0.3)
  )
)
```

- This approach can handle ...

- Time dependent regressors / covariates also with EM based methods
- Any covariate relationships
- Passing individual PK parameters via the dataset for sequential PK/PD modeling

# Summary

- Most model specification elements covered
- Typical range of longitudinal PK and PK/PD can be covered
- Remaining specification elements relate to Bayes estimation in NONMEM
  - Covered in later modules

modelSpec\_IQRest {IQRtools}

R Documentation

Defines a modelSpec list for use in setting up estimation objects.

## Description

Can be used for syspharm and NLME type of estimation objects. This is a convenience function to allow better documentation of the elements that can be in the modelSpec argument for IQRnlmeEst and IQRsysEst. No consistency checks are done at this point. This is conducted in the IQRnlmeEst and IQRsysEst functions.

## Usage

```
modelSpec_IQRest(  
  POPvalues0,  
  POPestimate = NULL,  
  IIVdistribution = NULL,  
  IIVvalues0 = NULL,  
  IIVestimate = NULL,  
  IOVvalues0 = NULL,  
  IOVestimate = NULL,  
  errorModel = NULL,  
  covarianceModel = NULL,  
  covariateModel = NULL,  
  covariateModelValues = NULL,  
  COVestimate = NULL,  
  COVcentering = NULL,  
  PriorVarPOP = NULL,  
  PriorVarCovariateModelValues = NULL,  
  PriorDFerrorModel = NULL,  
  PriorIIV = NULL,  
  PriorDFIIV = NULL  
)
```

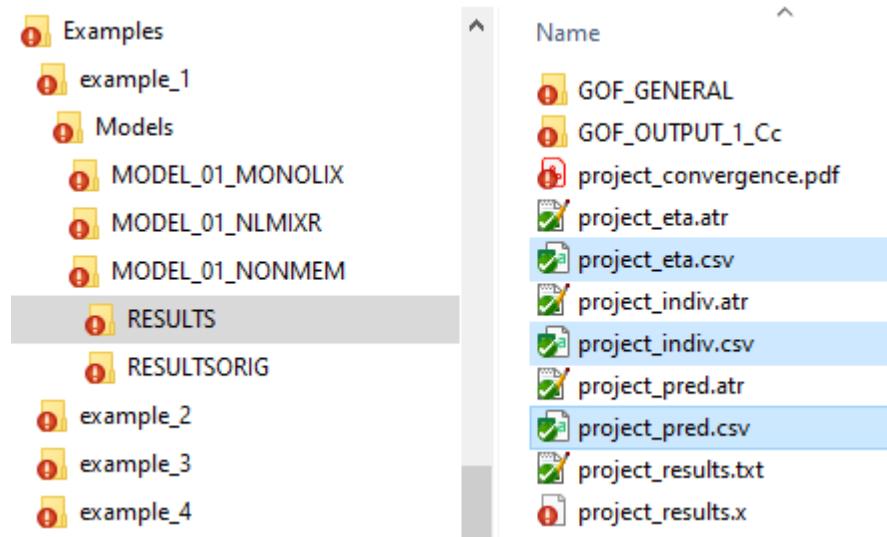
# Special elements

1. *How does IQR Tools handle MU referencing*
2. *How does IQR Tools handle sequential estimations (ITS -> SAEM -> IMP) when preparing a NONMEM model file?*
3. *How to use IQR Tools to define initial conditions for the compartments in model file?*
4. *Best practices on covariate selection*
5. *More complex examples on dataset preparation, e.g., urine data, metabolite data, dataset with initial conditions, etc.*
6. *How can IQR Tools help with model validation and diagnostics, (pc-)VPC, bootstraps*
7. *Auto-reporting of NONMEM / MONOLIX M&S results*



# Results CSV files

- Each IQRnlmeProject/RESULTS folder contains post-processed estimation results in CSV format
  - Same format across supported estimation software (NONMEM, MONOLIX, NL MIXR)



- Allowing to easily implement custom analyses, tables, graphics

# Functions allowing to access key information in an IQRnlmeProject

**project**: Path to an IQRnlmeProject folder

## Get structural model

```
model <- getModel_IQRnlmeProject(project)
```

## Get modeling dataset

```
data <- getData_IQRnlmeProject(project)
```

## Get data.frame with individual parameter estimates (includes covariates)

```
param <- getIndivParameters_IQRnlmeProject(project)
```

## Get data.frame with individual predictions

```
pred <- getIndivPredictions_IQRnlmeProject(project, FLAGcovariate = TRUE/FALSE)
```

## Get data.frame with individual ETAs

```
eta <- getETAs_IQRnlmeProject(project, FLAGcovariate = TRUE/FALSE)
```

## Sample parameters from an NLME project (same syntax as sample\_GPF() described in Module 1.3)

```
samp <- sample_IQRnlmeProject(project, ...)
```

# Summary tables and IQRnlmeProjectMulti object

- Any number of IQRnlmeProject folders can be combined in IQRnlmeProjectMulti objects

example\_1.R

```
# Generate a multi object with selected NLME projects
pM1 <- as_IQRnlmeProjectMulti(c("Models/MODEL_01_MONOLIX/", "Models/MODEL_01_NONMEM/"))

# Generate a multi object with all NLME projects in a given folder
pM2 <- as_IQRnlmeProjectMulti("Models")

# Generate a multi object with all NLME projects in a given folder and subfolders
pM3 <- as_IQRnlmeProjectMulti("../", FLAGrecursive = TRUE)
```

## Summary tables

- Function summary() can be used on IQRnlmeProjectMulti objects to generate 4 different summaries with different purposes (e.g., model development tables) - ideally exported to files

```
# Generate a summary
summary(pM2, pathname = "Summary/pM2")
```

example\_1.R

MODEL	OBJ	ka	CL	Vc	error_ADD1
**Fixed effects / Errors (RSE%**					
Models/MODEL_01_MONOLIX	346	1.59 (19%)	3.49 (8.6%)	40.2 (5%)	0.738 (7.7%)
Models/MODEL_01_NLMIXR	126	1.58 (19%)	3.48 (8.6%)	40.2 (5%)	0.735 (NA%)
Models/MODEL_01_NONMEM	127	1.59 (33%)	3.48 (43%)	40.2 (24%)	0.735 (5.6%)
**Random effects (RSE%**					
	**OBJ**	**omega(ka)**	**omega(CL)**	**omega(Vc)**	
Models/MODEL_01_MONOLIX	346	0.634 (23%)	0.271 (24%)	0.148 (27%)	
Models/MODEL_01_NLMIXR	126	0.623 (NA%)	0.27 (NA%)	0.151 (NA%)	
Models/MODEL_01_NONMEM	127	0.66 (33%)	0.284 (66%)	0.156 (74%)	

# Model comparison based on IQRnlmeProjectMulti objects

- Key models often need to be compared. The function `compareModels_IQRnlmeProjectMulti()` supports this efficiently

example\_1.R

```
# Generate comparison tables  
compareModels_IQRnlmeProjectMulti(pM2)
```

Comparison of model results for multiple models

Parameter	Models/MODEL_01_MONOLIX	Models/MODEL_01_NLMIXR	Models/MODEL_01_NONMEM
ka	1.59 (19%)	1.58 (19%)	1.59 (33%)
CL	3.49 (8.6%)	3.48 (8.6%)	3.48 (43%)
Vc	40.2 (5%)	40.2 (5%)	40.2 (24%)
omega(ka)	0.634 (23%)	0.623 (NA%)	0.66 (33%)
omega(CL)	0.271 (24%)	0.27 (NA%)	0.284 (66%)
omega(Vc)	0.148 (27%)	0.151 (NA%)	0.156 (74%)
error_ADD1	0.738 (7.7%)	0.735 (NA%)	0.735 (5.6%)
OBJ	346	126	127
BIC	364	159	161
AIC	360	140	141

# Implementation of covariates

- The standard handling of covariates follows the MU referencing implementation

$$P_i = \text{invTrans}( \text{Trans}(P) + \underbrace{\text{betaCOV} * \log(\text{COV}_i / \text{REF})}_{\text{Continuous covariate implementation. Multiple additive terms for multiple continuous covariates}} + \underbrace{\text{betaCATx} * (\text{CAT}_i == X)}_{\text{Categorical covariate implementation. Multiple additive terms for multiple levels of a categorical covariate and/or multiple categorical covariates}} + \text{ETA}_i )$$

- **P**: population mean value
- **Trans()**: Transformation defined by the distribution of the IIV
- **invTrans()**: Inverse transformation of Trans()
- **COV<sub>i</sub>**: individual continuous covariate
- **REF**: reference / centering value for continuous covariate
- **CAT<sub>i</sub>**: individual categorical covariate
- **X**: categorical covariate value (different from reference group)
- **ETA<sub>i</sub>**: Individual random effect

# Implementation of covariates

- Intentional transformation of continuous covariates ( $\log(\text{COVi}/\text{REF})$ )

$$P_i = \text{invTrans}( \text{Trans}(P) + \text{betaCOV} * \log(\text{COVi}/\text{REF}) + \text{ETA}_i )$$

- Ensures that for log-normally distributed parameters ( $\text{Trans}() = \log()$ ) the following covariate relationship is implemented:

$$P_i = P * (\text{COVi}/\text{REF})^{\text{betaCOV}} * \exp(\text{ETA}_i)$$

- Important:**

- MU referencing based implementation of covariates is **very beneficial** for convergence
- EM based methods only allow handling of time-independent covariates**
  - NONMEM FO/CE/I will take into account time-dependency - but not SAEM or BAYES
  - Covariate relationships that cannot be implemented using this approach can always be coded in the structural model - passing the covariates as regressors
  - Covariate coefficients then are considered normal parameters to be defined in POPvalues0, etc. (see Example 9)**
  - With SAEM care needs to be taken to allow for fixed small IIV on resulting covariate coefficient parameters - when estimated

# BLOQ handling

- Different methods for BLOQ handling are handled via the dataset (see Module 2.1)
- If M3 or M4 methods was chosen during generation of the dataset format, then the CENS column in the dataset is populated in the same way
  - When generating an IQRnlmeProject
    - By default, the M3 method is used
    - The M4 method can be chosen through the optional argument `algOpt.NONMEM.M4`  
`IQRnlmeProject(algOpt.NONMEM.M4 = TRUE, ...)`
    - The choice between M3 and M4 only exists for NONMEM projects

# Optional control arguments of IQRnlmeProjects

## Showing default settings

- Tool independent

```
comment = "",  
tool = "MONOLIX",  
toolVersion = NULL,  
multiTestN = 1,  
multiTestSD = 0.5,  
FLAGanalytic = TRUE,  
keepProjectFolder = FALSE,  
algOpt.SEED = 123456,  
algOpt.K1 = 500,  
algOpt.K2 = 200,  
algOpt.NRCHAINS = NA,
```

- NONMEM

```
algOpt.NONMEM.METHOD = "SAEM",  
algOpt.NONMEM.MAXEVAL = 9999,  
algOpt.NONMEM.SIGDIGITS = 3,  
algOpt.NONMEM.PRINT = 1,  
algOpt.NONMEM.COVSTEP_MATRIX = "S",  
algOpt.NONMEM.ADVAN7 = TRUE,  
algOpt.NONMEM.N1 = 1000,  
algOpt.NONMEM.TOL = 6,  
algOpt.NONMEM.SIGL = NULL,  
algOpt.NONMEM.M4 = FALSE,  
algOpt.NONMEM.FOCEIOFV = FALSE,  
algOpt.NONMEM.IMPORTANCESAMPLING = TRUE,  
algOpt.NONMEM.IMP_ITERATIONS = 10,  
algOpt.NONMEM.ITS = TRUE,  
algOpt.NONMEM.ITS_ITERATIONS = 10,  
algOpt.NONMEM.WRES = NULL,  
algOpt.NONMEM.PRED = NULL,  
algOpt.NONMEM.RES = NULL,
```

- MONOLIX

```
algOpt.MONOLIX.individualParameters = "conditionalMode",  
algOpt.MONOLIX.indivMCMClength = 50,  
algOpt.MONOLIX.indivNsim = 10,  
algOpt.MONOLIX.indivRatio = 0.05,  
algOpt.MONOLIX.logLikelihood = "Linearization",  
algOpt.MONOLIX.fim = "Linearization",  
algOpt.MONOLIX.variability = "FirstStage",  
algOpt.MONOLIX.startTime = NULL,  
algOpt.MONOLIX.STIFF = TRUE,
```



# Running NLME models outside of IQR Tools

- Generated NONMEM or MONOLIX models can be executed outside IQR Tools
- **Interesting use case:**
  - Generate MONOLIX NLME project
  - Open MONOLIX GUI, load generated NLME project
  - Assess initial guesses graphically, perform a first quick estimation
  - Update initial guesses
  - Generate NONMEM NLME project
  - Estimate in NONMEM

# Modification of generated control files

- IQR Tools generated control files could be modified post generation
- Allowing to implement specific settings of interest that are not readily supported by IQR Tools NLME API

# NONMEM reproducibility

- Parallel NONMEM runs (using more than 1 core) with EM methods lead to different parameter estimates for each model run (same model, same initial guesses, same seed, same dataset, same computer, same compiler, etc.)
  - Differences can be substantial!
- Through special settings IQR Tools and IQdesktop are able to limit the differences to a bare minimum
- **Dedicated webinar of NONMEM reproducibility planned for later**

# Supported MONOLIX versions

- MONOLIX  $\geq$  2019R1 is supported
- Since MONOLIX 2019R2 a bug is present in MONOLIX which can lead to MONOLIX models exported from IQR Tools not being correctly read (even though correct MONOLIX syntax is used)
  - **Work around 1: Use MONOLIX 2019R1**
  - Work around 2: Set FLAGanalytic=FALSE in call to IQRnlmeProject
  - Work around 3: Wait for MONOLIX 2021R1

# Conclusions

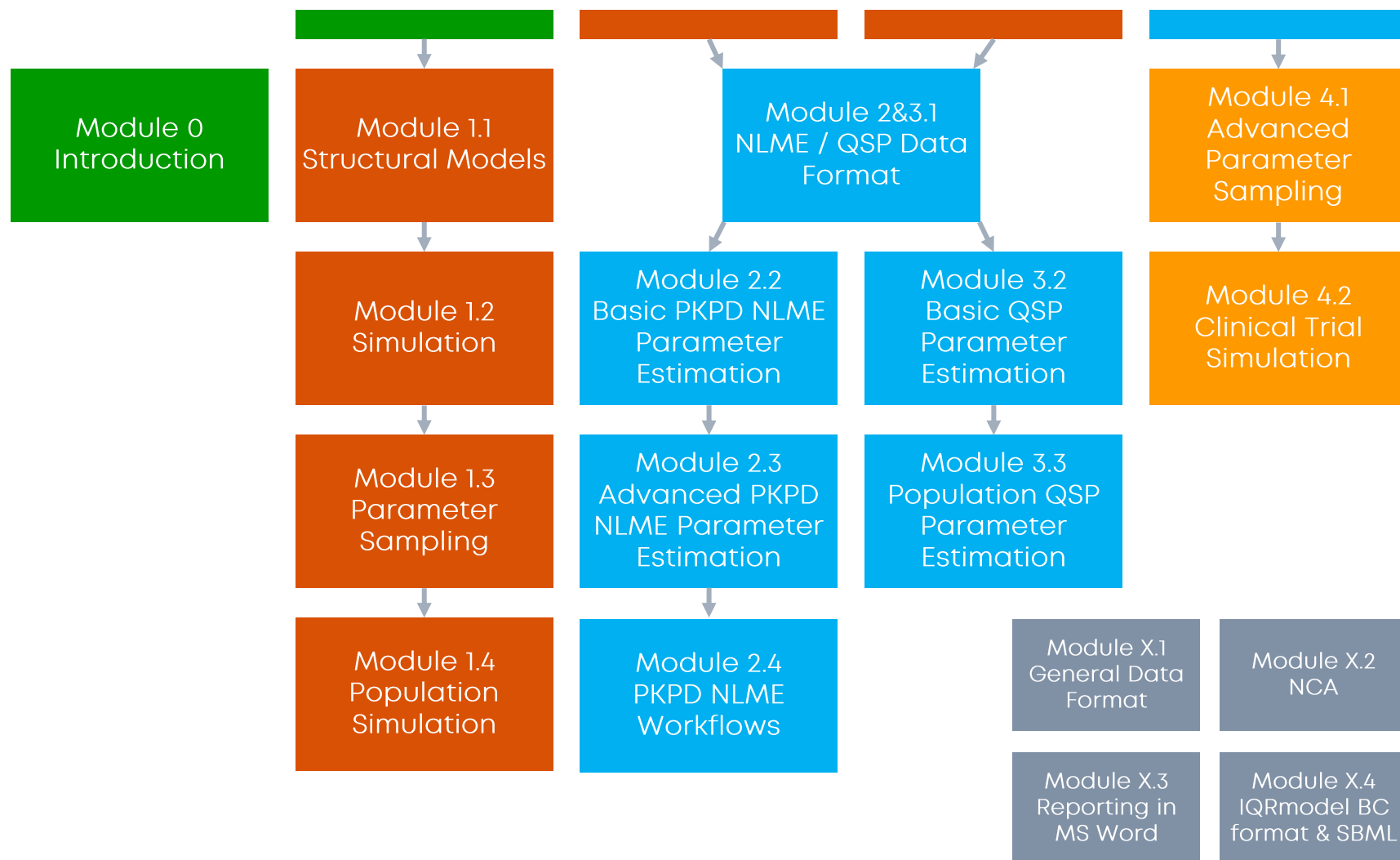
# Conclusions

- Easy setup and execution of NLME parameter estimation problems
- Automatic post-processing of results allows direct assessment of results to inform the subsequent modeling steps
- Encapsulation of NLME estimation problems in single folders (NLME Projects) allows structured storage
- Easy switching between tools (NONMEM, MONOLIX, NLMIXR) - no expert knowledge in setting up NONMEM control streams or MONOLIX MLXTRAN files required
  - => Focus on expertise in NLME modeling, rather than in NONMEM or MONOLIX

# Outlook webinar modules

# Overview of Webinar Series by IntiQuan

IntiQuan Webinar Series on efficient support of Model Informed Drug Development (MIDD)





# Q&A session

Thank  
You

# Contact information



IntiQuan GmbH  
Elisabethenstrasse 23  
4051 Basel  
Switzerland

Phone: +41 76 603 28 06  
E-mail: [info@intiquan.com](mailto:info@intiquan.com)  
Web: [www.intiquan.com](http://www.intiquan.com)