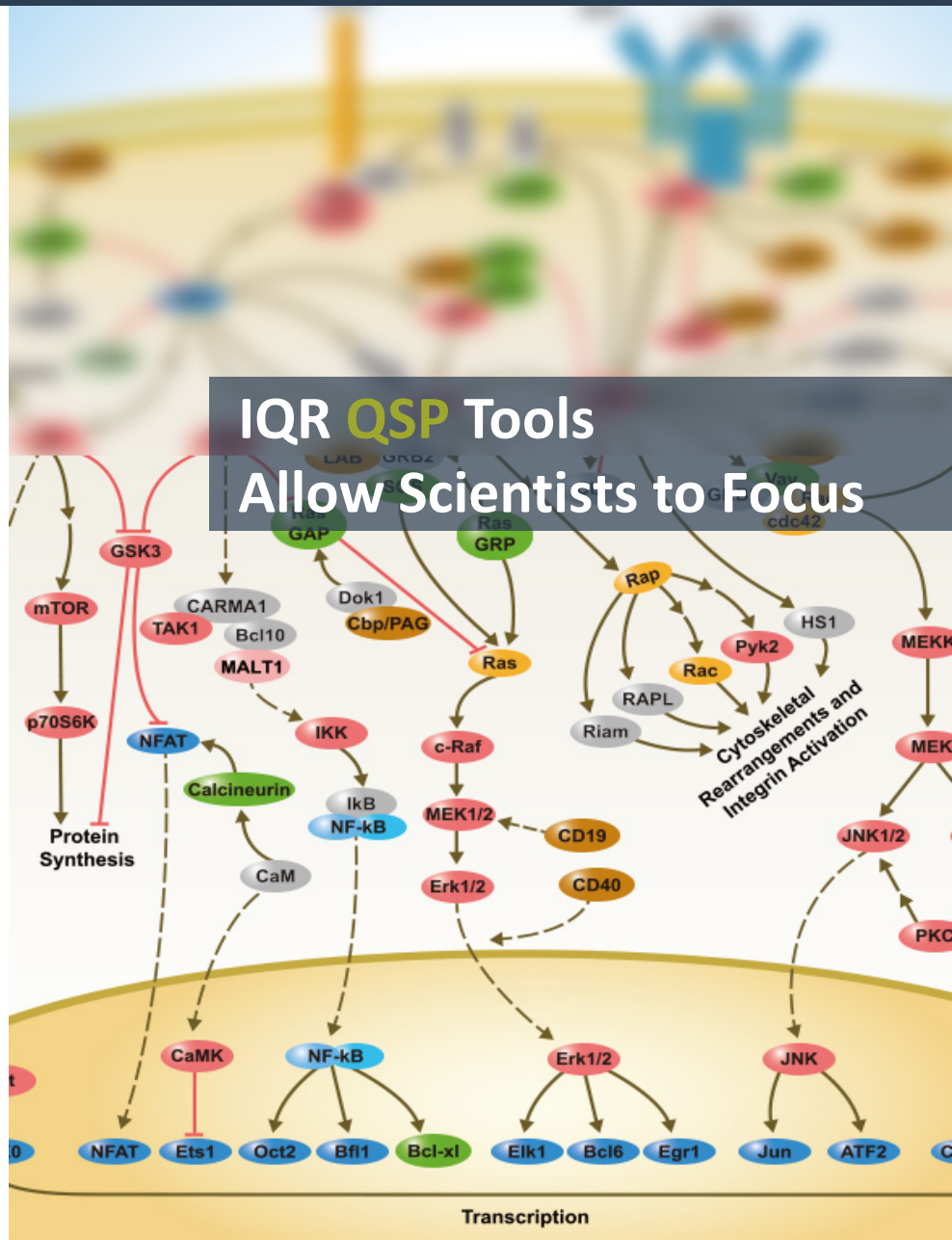
WORKSHOP

Introduction to Advanced QSP Modeling in R

Henning Schmidt, IntiQuan

ACoP10, October 24, 2019



ABOUT INTIQUAN

- **Swiss Modeling & Simulation consulting company**
- **Founded in 2015**

- **Main working areas**
  - **Pharmacometric modeling & simulation**
    - Preclinical to submission
    - Oncology, infectious diseases, respiratory, immunology
    - Pediatric analyses
    - QSP

  - **R based tools and workflow approaches supporting Modeling & Simulation**
    - Data standards & preparation of datasets
    - Data analysis (exploration, NCA)
    - Efficient modeling workflows (PK, PKPD, QSP)
    - Seamless Word reporting

IntiQuan

THE INTIQUAN TEAM

Henning Schmidt

Founder and Managing Director

- Dr. Automatic Control, KTH Stockholm
- Dipl.-Ing. Electrical Eng., TH Darmstadt
- Ingénieure SUPELEC, Paris
- Robert Bosch (1996-1999)
- Fraunhofer-Chalmers (2004-2007)
- Novartis (2008-2015)

Daniel Kaschek

Consultant

- Dr. rer. Nat, Physics, Uni Freiburg
- Dipl., Physics, Uni Freiburg
- Uni Freiburg (2014-2017)

Daniel Lill

Junior Consultant, PhD Student

- M.Sc., Physics, Uni Freiburg

Andrijana Radivojevic

External Consultant - Business Development

- Dr. phil. Systems Biology, EPFL Lausanne
- Dipl.-Ing. Electrical Engineering, Serbia
- Novartis (20012-2018)

Anne Kümmel

Director

- Dr. sc. Process Eng., ETH Zürich
- Dipl.-Ing. Mechanical Eng., RWTH Aachen
- Novartis (2008-2014)
- Actelion (2014-2017)

Benjamin Guiastrennec

Consultant

- PhD, Uppsala University, Sweden
- PharmD, Montpellier, France
- Certara
- Pharmetheus
- Novartis

Willi Weber

Senior Medical Advisor

- Dr. Chemistry
- PD Dr. med.
- Former head of M&S Sanofi, Frankfurt

IntiQuan

WORKSHOP OUTLINE

- Introduction
- Models and their Simulation (ODE / BC syntax)
- Linking Models and Data
- Parameter Estimation

Coffee Break

- Can I trust my model? Informing needed experiments
- Hands-On Session
- Special Workflow Topics
  - Reuse of models from model databases – import of SBML models
  - QSP going NLME
  - Full NLME
  - Evaluation of hybrid models (NLME+QSP)
  - Generalized dataset format

End of Workshop

- Available for Q&A after Workshop

WORKSHOP GOALS

You will have learned to

- Write your own mechanistic QSP models in ODE or biochemical format
- Simulate models
- Setup data used for estimation and its exploration
- Performing powerful parameter estimation with and without consideration of parametric variability
- Use model-based information to better inform the design of new experiments, increasing mechanistic understanding and trust in your model

INTRODUCTION

MODELS AND RELATED CHALLENGES

The Models

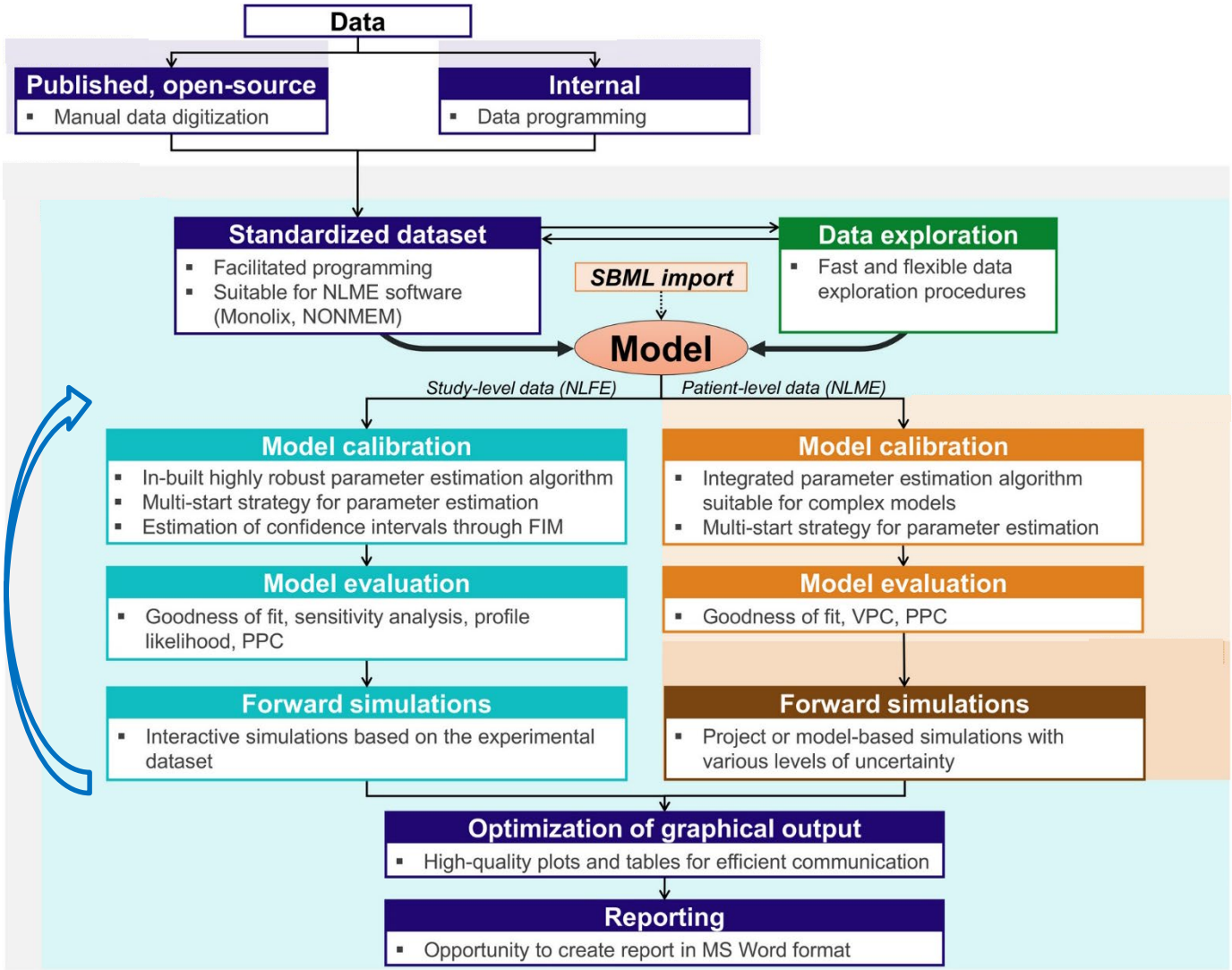
- Typically non-linear ODE based models of medium to large complexity

The Challenges

- Finding user-friendly and powerful tools supporting the user in efficient modeling and simulation
  - Numerically accurate and stable parameter estimation methods for complex systems
  - Fast numerical solution, allowing seamless and cheap parallelization
  - Ease of handling of large amounts of heterogenous experimental data
  - Integration of clinical data (with variability) and mechanistic modeling
  - Same platform, allowing for fully reproducible and workflow-based approaches

Profile likelihood?
Fix one parameter and estimate the posterior.. => computational demand high

AN EFFICIENT QSP WORKFLOW AND USER-FRIENDLY TOOLS ARE CRITICAL FOR MODEL DEVELOPMENT AND SUPPORT OF DRUG DEVELOPMENT QUESTIONS



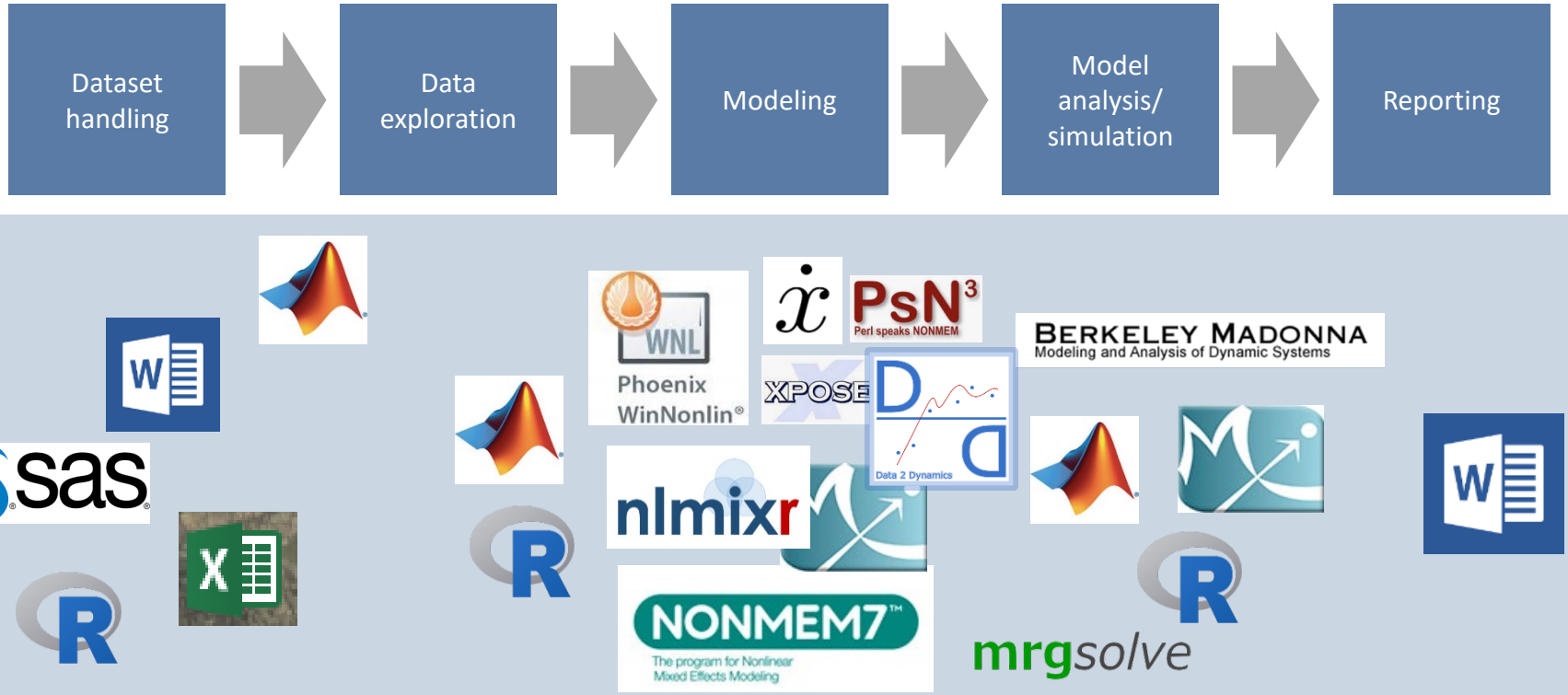
Handling of

- Heterogenous data
- Summary data (no variability)
- Individual data (variability)

https://ascpt.onlinelibrary.wiley.com/doi/full/10.1002/psp4.12426

IntiQuan

TYPICALLY TOOL LANDSCAPE RESULTS IN A FRAGMENTED MODELING WORKFLOW



Various formats
Manual documentation

Re-coding on project basis

Switching between tools

Re-coding of models

Cumbersome simulation & analysis

Copy/paste of results to report

Limited reproducibility

IntiQuan

WHAT ARE YOUR EXPERIENCES & EXPECTATIONS?

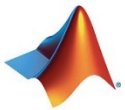
IntiQuan

2005-2015: MATLAB BASED SBTOOLBOX(2)

Systems Biology Toolbox for MATLAB: a computational platform for research in systems biology FREE

Henning Schmidt ✉, Mats Jirstrand

Bioinformatics, Volume 22, Issue 4, 15 February 2006, Pages 514–515,
https://doi.org/10.1093/bioinformatics/bti799
Published: 29 November 2005 **Article history** ▾

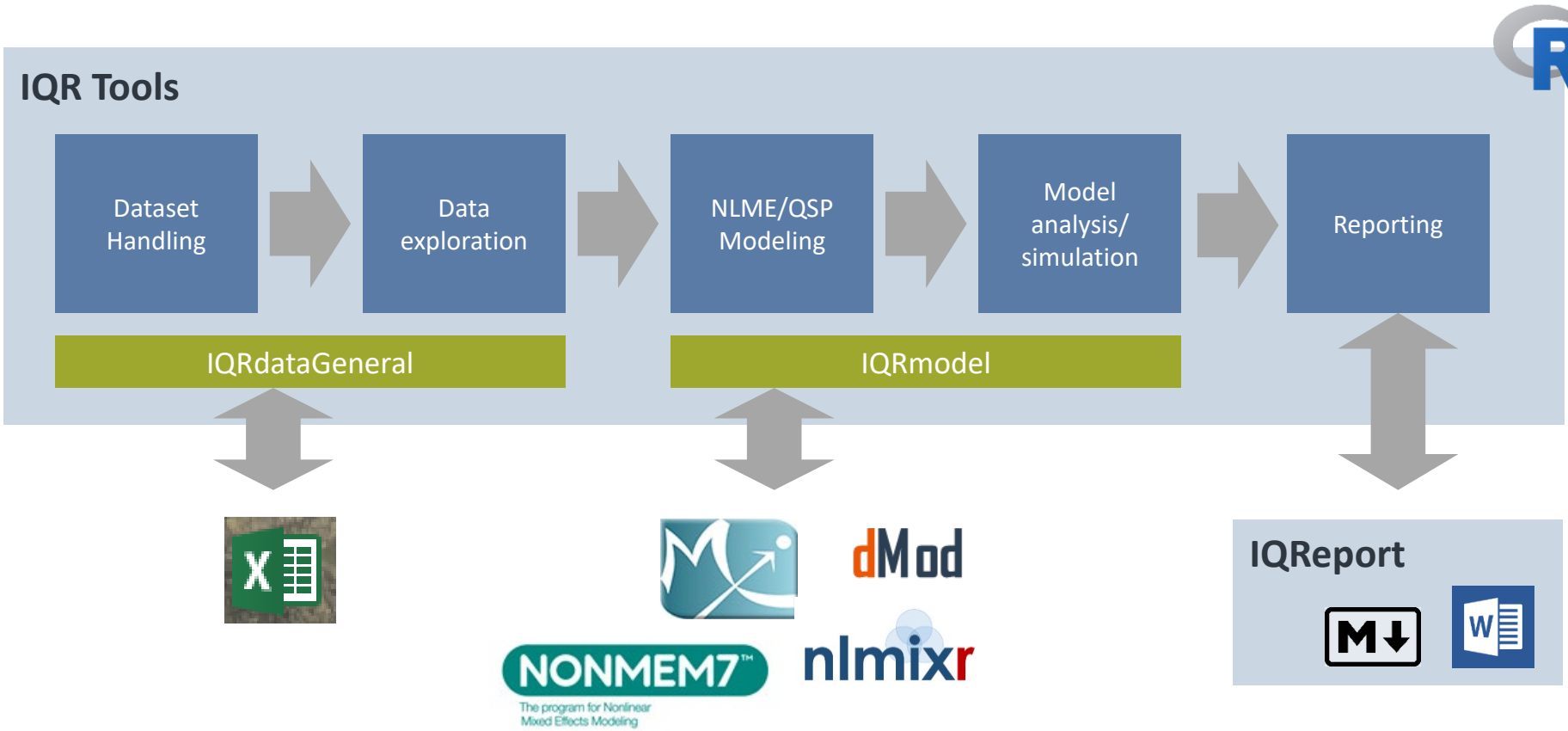


http://www.sbtoolbox2.org

- Already in 2006
  - SBTOOLBOX provided 100-1000x faster simulation than SimBiology (Nowadays speed difference should be smaller)
  - Allowed to consider multi experiment parameter estimation considerably facilitating SysBio and QSP type of modeling
- 2009-2015
  - Seamless interfaces to NLME tools were added, making SBTOOLBOX2 the first tool covering the full range from early mechanistic to full NLME modeling
  - Used also for regulatory submissions
- 2016
  - Transition to R
- 2017-2019
  - Working over the QSP modeling part

IntiQuan

WE INTEGRATED ALL STEPS IN AN R-BASED WORKFLOW => IQR TOOLS



Script and function-based workflow based on tool-independent data format and model definition - covering Systems Biology, Systems Pharmacology, and Pharmacometrics

IntiQuan

IQR TOOLS COMES WITH A DETAILED DOCUMENTATION, PROVIDING PLENTY OF EXECUTABLE EXAMPLES

https://iqrtools.intiquan.com/doc/book/

Modeling & Simulation in R

Supporting Efficient Model Informed Drug Development with IQR Tools

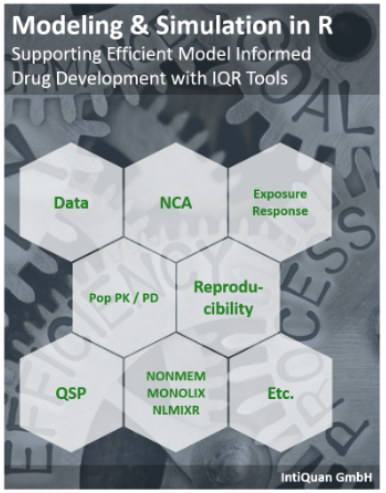
IntiQuan GmbH

27 November 2018

Preface

Welcome to the official documentation of the IQR Tools R package!

A lot of material is contained already and we are heavily working on extending the documentation. With the fantastic bookdown R package we now have finally found the right documentation approach.



Who this book is for

This book is for people who want to efficiently perform Modeling & Simulation tasks related to Pharmacometrics, Quantitative Systems Pharmacology, and/or Systems Biology in the R environment.

- Modeling & Simulation experts in the pharma industry
- Academic researchers and students
- Scientists who want to use the right tool for the available task

To get a quick overview of the IQR Tools capabilities, have a quick look at the provided Case Studies.

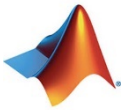
How to read this book

IntiQuan

QUICK OVERVIEW OF QSP SPECIFIC COMPONENTS IN IQR TOOLS

Structural Model

SimBiology



BERKELEY MADONNA
Modeling and Analysis of Dynamic Systems

Input processed by IQR Tools or other tools

Experimental Data

- In-vitro
- In-vivo, pre-clinical
- Clinical data

Sys-Model

- Simulation
- Data Exploration
- Experimental Conditions
- IIV

Sys-Est
Sys-Project

Estimation Tool

dMod

- Parameter **Estimation**
- **Identifiability** Analysis

If desired, NONMEM, MONOLIX, or NLMIXR can be used as well

Outputs for further processing with IQR Tools

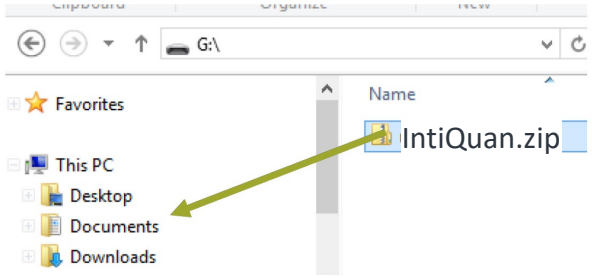
IntiQuan

SETTING UP THE WORKSHOP MATERIAL

STEP1 – Workshop material provided on USB stick

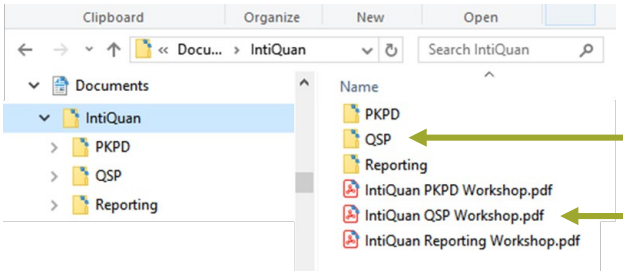
USB:

- Copy IntiQuan.zip to your home folder



Windows: The "home" folder is the "Documents" folder.

STEP2 – Extract the IntiQuan.zip file "here"



QSP workshop material, examples, exercises

This presentation

IntiQuan

LET'S START

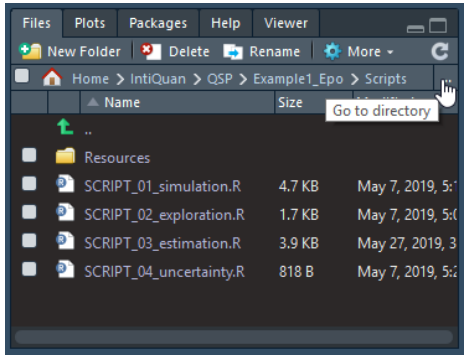
- The example code is available in

```
~/IntiQuan/QSP/Example1_Epo/Scripts
```

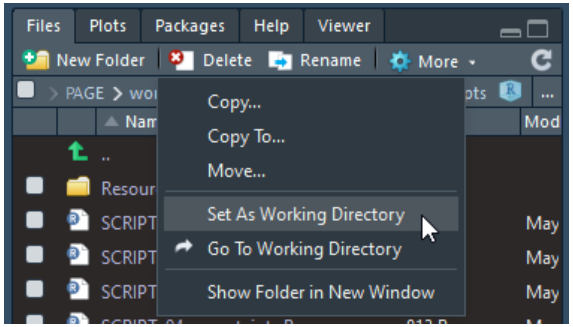
home = This PC → Documents (on Windows)

- In RStudio (recommended), go to above directory and set as working directory

(1)



(2)



- It is recommended to execute the code in parallel to the presentation on the slides

IntiQuan

TEST LOADING OF IQR TOOLS

```
> library(IQRtools)
IQRtools: The solution for Modeling & Simulation across
          Systems Pharmacology, Systems Biology, and Pharmacometrics.

          Version: 1.0.8

          Licensed to: henning (henning@intiquan.com)
          Expiration date: 27-February-2027

          NEW: Support Forum on https://support.intiquan.com/
```

If you see such a message with version number >= 1.0.8 all is fine!

Otherwise execute:

```
source("https://iqrtools.intiquan.com/install.R")
installVersion.IQRtools()
```


IntiQuan

TESTING THE INSTALLATION

- Execute the following:

```
library(IQRtools)
test_IQRtools()
```

- If you see the following - then all is fine

```
> library(IQRtools)
> test_IQRtools()
  TIME Cyclin     YT      PYT   PYTP   MPF Cdc25P Wee1P   IEP APCstar     k2   kwee   k25   R1       R2      R3       R4       R5
1    1 0.0172 0.0116 0.000900 0.0198 0.073   0.95  0.95 0.242   0.313 0.0817 0.0595 0.162 0.01 0.00141 0.0086 0.000292 0.000690
2    2 0.0172 0.0116 0.000882 0.0198 0.073   0.95  0.95 0.242   0.313 0.0817 0.0595 0.162 0.01 0.00141 0.0086 0.000292 0.000689
       R6       R7       R8       R9      R10      R11     R12     R13     R14     R15       R16       R17      R18      R19
1 0.00742 0.000948 0.000146 0.000576 7.36e-05 7.92e-05 0.00434 0.00321 0.00162 0.00597 -5.13e-05 -5.13e-05 5.13e-05 3.33e-06
2 0.00741 0.000947 0.000143 0.000565 7.21e-05 7.92e-05 0.00435 0.00321 0.00162 0.00597 -1.39e-05 -1.39e-05 1.01e-04 1.30e-05

IQRsimres object>
```

MODELS AND THEIR SIMULATION (ODE / BC SYNTAX)

`Example0_Models/Scripts/SCRIPT_01_models.R`

Getting started: A first ODE model

```
R> new_IQRmodel()
```

```
********** MODEL NAME
```
Model identifier

```
********** MODEL NOTES
```
Provide detailed description of the model

```
********** MODEL STATES
```
ODEs and initial values

```
d/dt(X1) = -R1, X1(0)=1
d/dt(Y1) = R1, Y1(0) = 0
```

```
********** MODEL PARAMETERS
```
Parameter values

```
k = 0.1
```

```
********** MODEL VARIABLES
```
Auxiliary variables

```
Total = X1 + Y1
```

```
********** MODEL REACTIONS
```
Reactions (fluxes)

```
R1 = k*X1
```

```
********** MODEL FUNCTIONS
```
Auxiliary functions

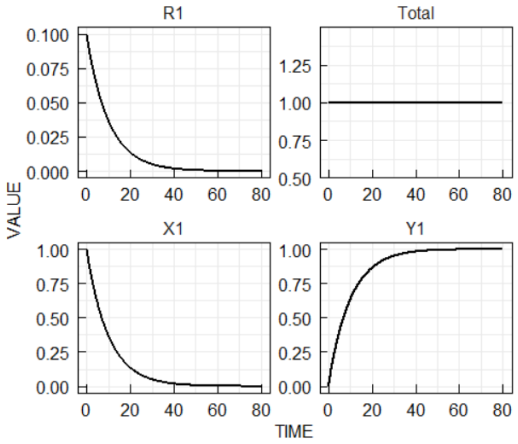
```
********** MODEL EVENTS
```
Discrete state events

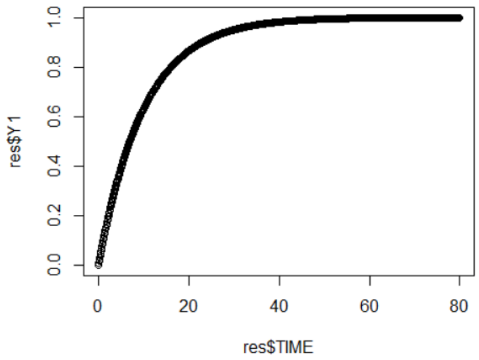
https://iqrtools.intiquan.com/doc/book/modelImplementation.html

LOADING AN ODE MODEL INTO R, SIMPLE SIMULATION, AND PLOTTING

Example0_Models/Scripts/SCRIPT_01_models.R

```
##############################
# Basic model simulation
##############################

# Look at the model file
file.edit("firstmodel.txt")

# Load the model from text file into R as an object
model <- IQRmodel("firstmodel.txt")

# Simulate the model (parameters and initial conditions
# as defined in the model file)
res <- sim_IQRmodel(model,simtime=80)

# Plot results
plot(res)

# Plot specific output
plot(res$TIME,res$Y1)
```





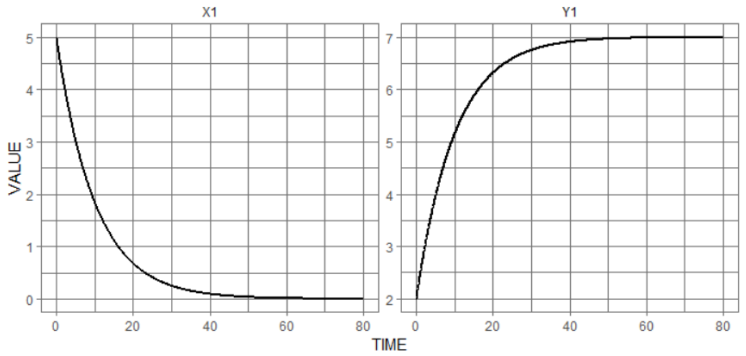
firstmodel.txt

```
1  ********** MODEL NAME
2
3  Provide Model Name
4
5  ********** MODEL NOTES
6
7  Provide Model Information
8
9  ********** MODEL STATES
10
11 d/dt(X1) = -R1
12 d/dt(Y1) = R1
13
14 X1(0) = 1
15 Y1(0) = 0
16
17 ********** MODEL PARAMETERS
18
19 k = 0.1
20
21 ********** MODEL VARIABLES
22
23 Total = X1 + Y1
24
25 ********** MODEL REACTIONS
26
27 R1 = k*X1
28
29 ********** MODEL FUNCTIONS
30
31
32 ********** MODEL EVENTS
```

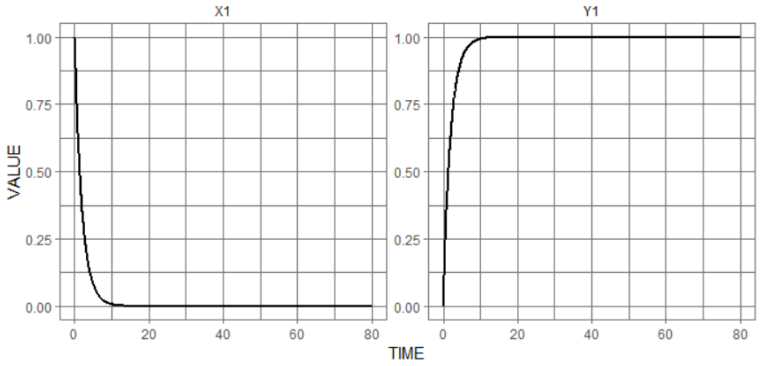

IntiQuan

CHANGING PARAMETERS AND INITIAL CONDITIONS FOR THE SIMULATION

```
##############################
# Simulation with changed initial conditions
##############################

# IC for X1=5, IC for Y1=2
res <- sim_IQRmodel(model,simtime=80,IC = c(X1=5,Y1=2))

# Plotting only subset and not all elements
plot(res[,c("TIME","X1","Y1")])
```




```
##############################
# Simulation with changed parameters
##############################

# Setting k to 0.5
res <- sim_IQRmodel(model,simtime=80,parameters = c(k=0.5))

# Plotting only subset and not all elements
plot(res[,c("TIME","X1","Y1")])
```



IntiQuan

EXPORTING A MODEL AND BIOCHEMICAL EQUATION NOTATION

```
# Exporting as ODE model
export_IQRmodel(model,filename = "modelexported.txt")

# Exporting as BC model
export_IQRmodel(model,filename = "modelexportedBC.txt",FLAGbc = TRUE)
```

ODE Notation

```
1  ********** MODEL NAME
2
3  Provide Model Name
4
5  ********** MODEL NOTES
6
7  Provide Model Information
8
9  ********** MODEL STATES
10
11 d/dt(X1) = -R1
12 d/dt(Y1) = R1
13
14 X1(0) = 1
15 Y1(0) = 0
16
17 ********** MODEL PARAMETERS
18
19 k = 0.1
20
21 ********** MODEL VARIABLES
22
23 Total = X1 + Y1
24
25 ********** MODEL REACTIONS
26
27 R1 = k*X1
28
29 ********** MODEL FUNCTIONS
30
31
32 ********** MODEL EVENTS
```

modelexported.txt

BC Notation

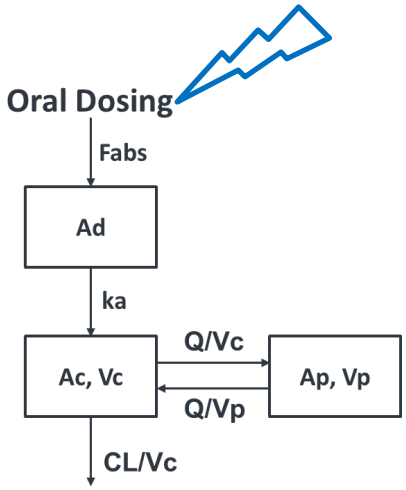
```
1  ********** MODEL NAME
2
3  Provide Model Name
4
5  ********** MODEL NOTES
6
7  Provide Model Information
8
9  ********** MODEL STATE INFORMATION
10
11
12 X1(0) = 1
13 Y1(0) = 0
14
15 ********** MODEL PARAMETERS
16
17 k = 0.1
18
19 ********** MODEL VARIABLES
20
21 Total = X1 + Y1
22
23 ********** MODEL REACTIONS
24
25 X1 => Y1 : R1
26         vf = k*X1
27
28 ********** MODEL FUNCTIONS
29
30
31 ********** MODEL EVENTS
```

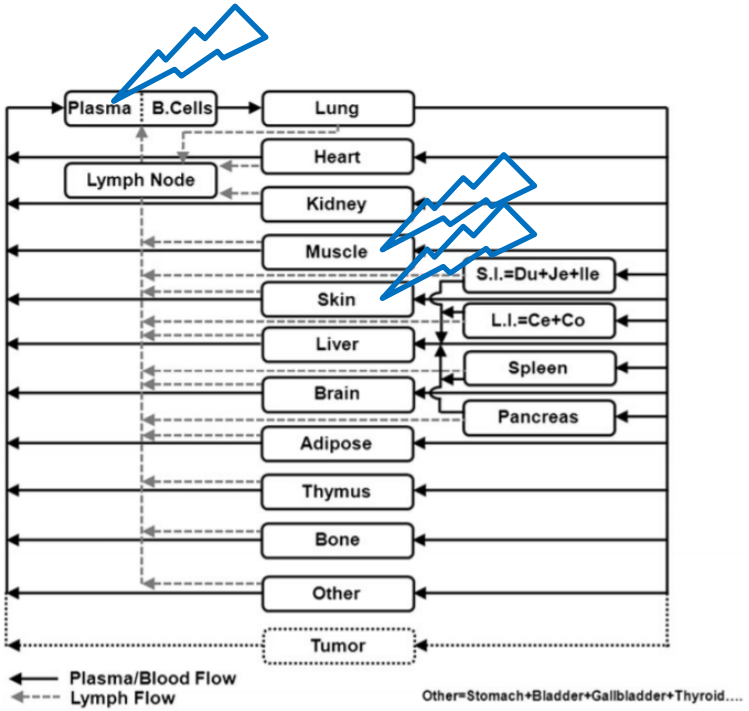
A + B => C : R1
Vf = k1*A1*B1

modelexportedBC.txt

IntiQuan

DOSING INPUTS TO MODELS





X1 => Y1

Dosing information consists of:

- Dosing time(s)
- Amount(s)
- Location(s)
- Duration(s)
- Potential repetition(s) (number and interval)

IntiQuan

DOSING INPUTS TO MODELS

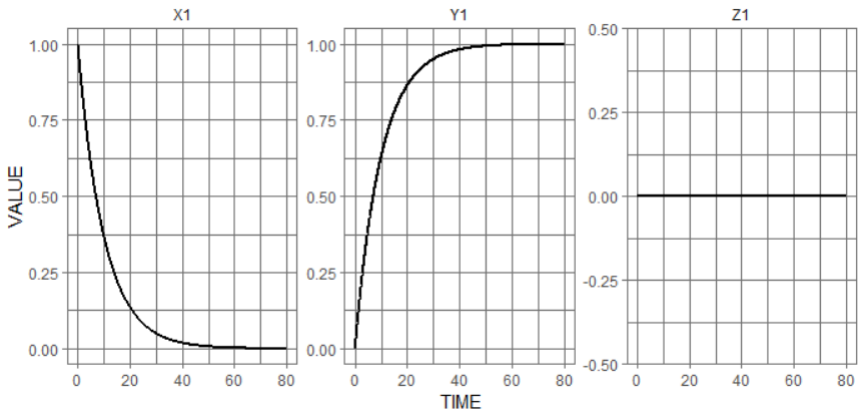
firstmodel_dosing.txt

```
1  ********** MODEL NAME
2
3  Simple example with dosing input
4
5  ********** MODEL NOTES
6
7  Provide Model Information
8
9  ********** MODEL STATES
10
11 d/dt(X1) = -R1 + INPUT1
12 d/dt(Y1) = R1
13
14 d/dt(Z1) = -k*Z1 + factor*INPUT2
15
16 X1(0)    = 1
17 Y1(0)    = 0
18 Z1(0)    = 0
19
20 ********** MODEL PARAMETERS
21
22 factor   = 0.5
23 k        = 0.1
24
25 ********** MODEL VARIABLES
26
27 Total    = X1 + Y1
28
29 ********** MODEL REACTIONS
30
31 R1       = k*X1
32
33 ********** MODEL FUNCTIONS
34
35
36 ********** MODEL EVENTS
37
```

A dosing input to a model is defined by

- The keyword "INPUT*" where "*" is 1, 2, 3, 4, …
- The term is added to a differential equation
- By default INPUT* is assumed to be 0

```
# Load the model from text file into R as an object
model <- IQRmodel("firstmodel_dosing.txt")

# Simulate the model (parameters and initial conditions
# as defined in the model file)
res <- sim_IQRmodel(model,simtime=80)

# Plot results
plot(res[,c("TIME","X1","Y1","Z1")])
```



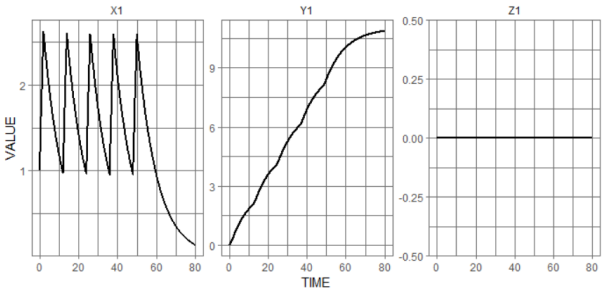
IntiQuan

DEFINITION OF A DOSING OBJECT AND USE IN SIMULATION OF MODEL

```
############################
# Definition of a dosing object
############################

dosing <- IQRdosing(
  TIME = 0, # Time of dosing (required)
  AMT  = 2, # Amount of dosing (required)
  ADM  = 1, # Location of administration 1=>INPUT1, 2=>INPUT2, ... (required)
  TINF = 2, # Duration of administration (optional. default: 0)
  ADDL = 4, # Number of additional repeated doses (optional. default: 0)
  II   = 12 # Dosing interval (required if ADDL is defined)
)

dosing
```

```
> dosing
  TIME ADM AMT TINF
1    0   1   2    2
2   12   1   2    2
3   24   1   2    2
4   36   1   2    2
5   48   1   2    2

IQRdosing object
  Number of inputs: 1
```

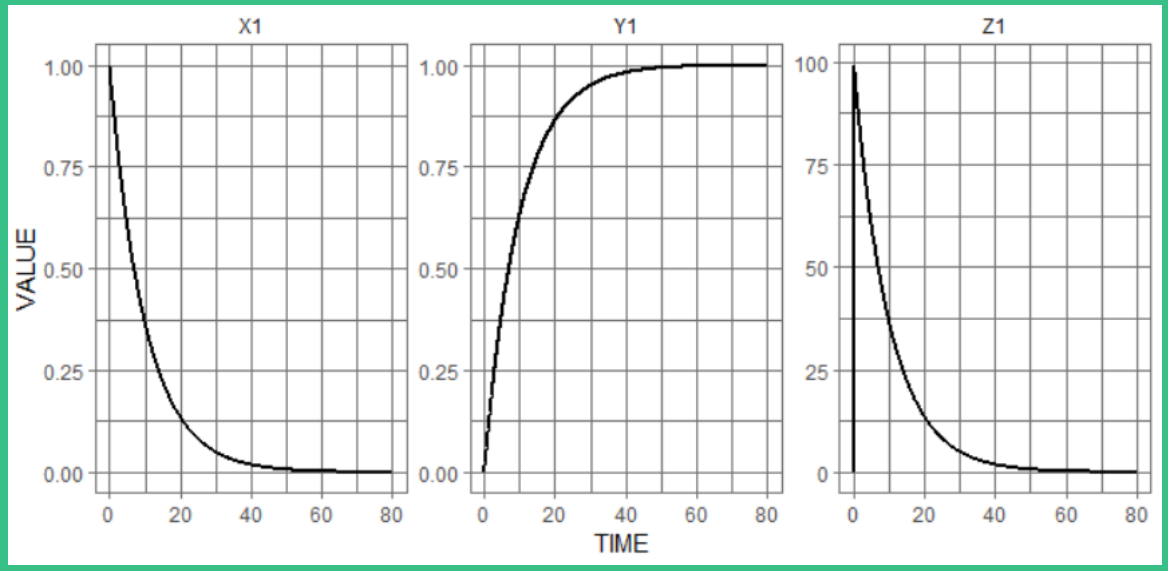
```
##############################
# Simulation of model with dosing object
##############################

# Load the model from text file into R as an object
model <- IQRmodel("firstmodel_dosing.txt")

# Simulate the model with the dosing object
res <- sim_IQRmodel(model,simtime=80,dosingTable = dosing)

# Plot results
plot(res[,c("TIME","X1","Y1","Z1")])
```



IntiQuan

HANDS-ON EXCERCISE

- Simulate a single dose (bolus: TINF=0) of 200 into INPUT2
- Plot the results




```
> dosing
  TIME ADM AMT  TINF
1    0   2 200 1e-04

IQRdosing object
  Number of inputs: 1
```

TINF=0 implemented as 1e-4 as Bolus as a Dirac pulse with infinite height numerically not feasible

IntiQuan

ESSENTIALLY NO LIMIT IN MODEL COMPLEXITY - AS LONG AS IT IS BASED ON ODES

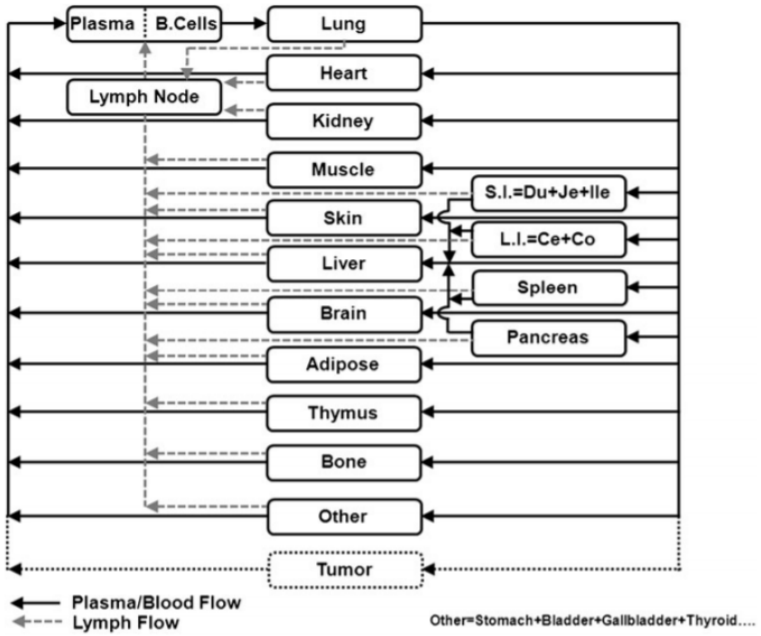
Example

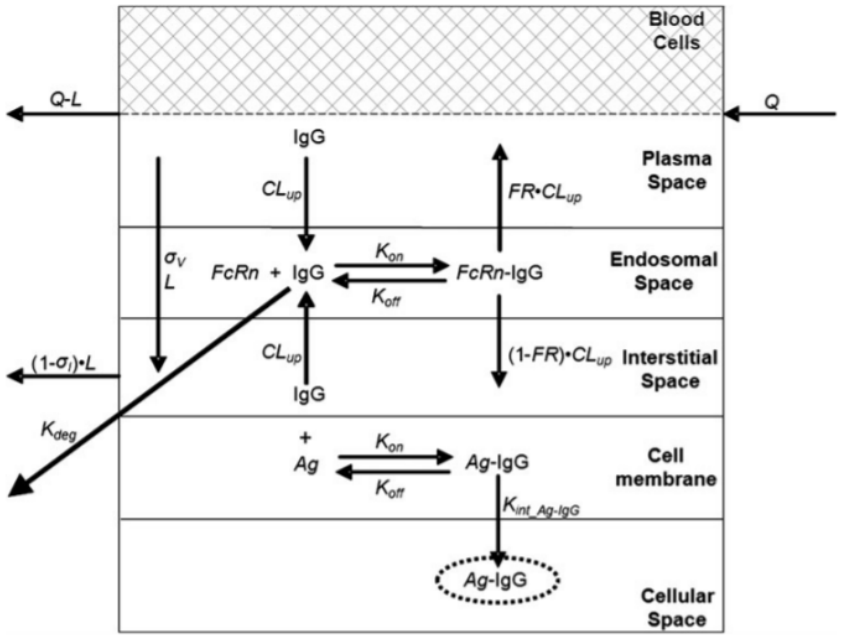
Journal of Pharmacokinetics and Pharmacodynamics
February 2012, Volume 39, Issue 1, pp 67–86 | Cite as

Towards a platform PBPK model to characterize the plasma and tissue disposition of monoclonal antibodies in preclinical species and human

Authors Authors and affiliations

Dhaval K. Shah, Alison M. Betts





IntiQuan

COMPARISON OF MODEL IMPLEMENTATION BASED ON ODES AND BIOCHEMICAL REACTION NOTATION

ODE based model code

Hard to edit and expand the model

Prone to mistakes

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

# PLASMA
# ------
# IV dosing assumed into plasma compartment. Scaling done from mg to M
# mg/1000 -> g * MW -> mol / volume -> molar

d/dt(CPL_Exo1M) = (((TUMOR_PLQ-TUMOR_LF)*CTUMOR_V_Exo1M)/PL_V)+(((HE_PLQ-HE_LF)*CHE_V_Exo1M)/PL_V)+(((KI_PLQ-KI_LF
)*CKI_V_Exo1M)/PL_V)+(((MU_PLQ-MU_LF)*CMU_V_Exo1M)/PL_V)+(((SK_PLQ-SK_LF)*CSK_V_Exo1M)/PL_V)+(((BR_PLQ-BR_LF
)*CBR_V_Exo1M)/PL_V)+(((AD_PLQ-AD_LF)*CAD_V_Exo1M)/PL_V)+(((TH_PLQ-TH_LF)*CTH_V_Exo1M)/PL_V)+((((LIV_PLQ-LIV_LF
)+(SI_PLQ-SI_LF)+(LI_PLQ-LI_LF)+(SP_PLQ-SP_LF)+(PA_PLQ-PA_LF))*CLIV_V_Exo1M)/PL_V)+(((BO_PLQ-BO_LF)*CBO_V_Exo1M
)/PL_V)+(((OTLN_PLQ-OT_LF)*COT_V_Exo1M)/PL_V)-(((LU_PLQ+LU_LF)*CPL_Exo1M)/PL_V)+(LN_LF*CLN_Exo1M)/PL_V + 1/(1000*MW
*PL_V)*INPUT1


# BLOOD CELL
# ----------
d/dt(CBC_Exo1M) = ((TUMOR_BCQ*CTUMOR_BC_Exo1M)/BC_V)+((HE_BCQ*CHE_BC_Exo1M)/BC_V)+((KI_BCQ*CKI_BC_Exo1M)/BC_V
)+((MU_BCQ*CMU_BC_Exo1M)/BC_V)+((SK_BCQ*CSK_BC_Exo1M)/BC_V)+((BR_BCQ*CBR_BC_Exo1M)/BC_V)+((AD_BCQ*CAD_BC_Exo1M)/BC_V
)+((TH_BCQ*CTH_BC_Exo1M)/BC_V)+((BO_BCQ*CBO_BC_Exo1M)/BC_V)+((OTLN_BCQ*COT_BC_Exo1M)/BC_V)+(((LIV_BCQ+SI_BCQ+LI_BCQ
+SP_BCQ+PA_BCQ)*CLIV_BC_Exo1M)/BC_V)-((LU_BCQ*CBC_Exo1M)/BC_V)


# LUNG
# ----

# Vascular
d/dt(CLU_V_Exo1M) = (((LU_PLQ+LU_LF)*CPL_Exo1M)/LU_VV)-(((LU_PLQ)*CLU_V_Exo1M)/LU_VV)-(((1-LU_VRC)*LU_LF*CLU_V_Exo1M
)/LU_VV)-((LU_CL_UP*CLU_V_Exo1M)/LU_VV)+((LU_CL_UP*FR*CLU_E_B_Exo1M)/LU_VV)

```

BC equation-based model code

Easy to understand and edit

Easy to add new reactions / mechanisms as desired

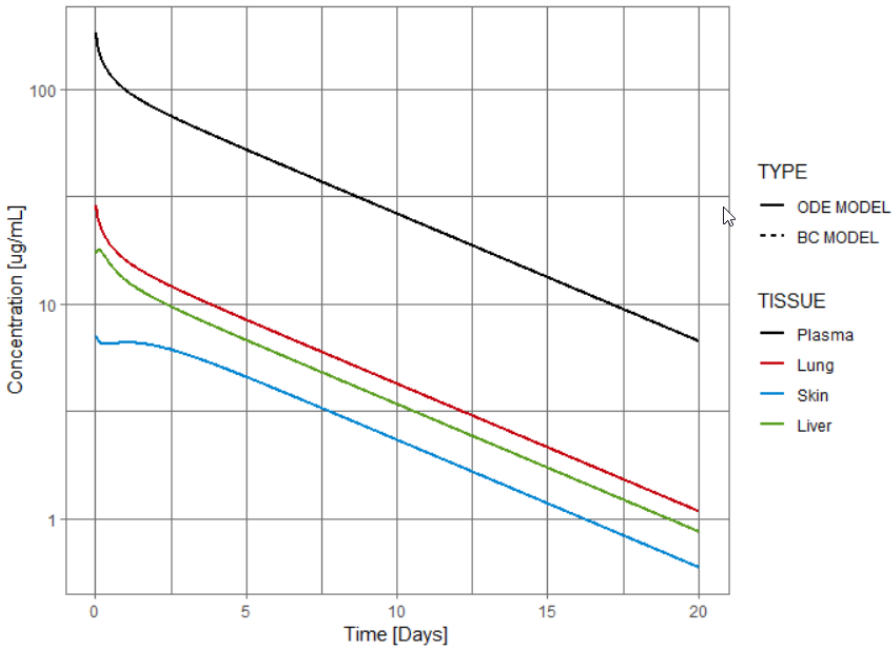
```
# ------------------------------------------
# HEART
# ------------------------------------------

# Connecting flows Lung / Plasma
# ------------------------------

A_LU_V => A_HE_V : vA_LU_HE
  vf = HE_PLQ*C_LU_V

A_HE_V => A_PL : vA_HE_PL
  vf = (HE_PLQ-HE_LF)*C_HE_V


# Connecting flows blood cells
# ----------------------------

A_HE_BC => A_BC : vA_HE_BC_BC
  vf = HE_BCQ*C_HE_BC

A_BC => A_HE_BC : vA_BC_HE_BC
  vf = HE_BCQ*C_LU_BC


# Vascular (same for all tissues)
# -------------------------------

A_HE_V => A_HE_I : vA_HE_V_I
  vf = (1-HE_VRC)*HE_LF*C_HE_V

A_HE_V => A_HE_E_U : vA_HE_V_E_U
  vf = HE_CL_UP*C_HE_V


# Interstitial (same for all tissues)
# -----------------------------------

A_HE_I => A_HE_E_U : vA_HE_I_E_U
  vf = HE_CL_UP*C_HE_I

A_HE_I => A_LN : vA_HE_I_LN
  vf = (1-HE_ISRC)*HE_LF*C_HE_I


# Endosomal (same for all tissues)
# --------------------------------

A_HE_E_B => A_HE_V + A_HE_FcRn : vA_HE_E_B_V
  vf = HE_CL_UP*FR*C_HE_E_B

A_HE_E_B => A_HE_I + A_HE_FcRn: vA_HE_E_B_I
  vf = HE_CL_UP*(1-FR)*C_HE_E_B

```

SIMULATION OF BC BASED MODELS WORKS IDENTICALLY TO SIMULATION OF ODE BASED MODELS

```
125 ##############################
126 # Simulate more complex model
127 ##############################
128
129 file.edit("Shah mAb PBPK.txt")
130 file.edit("Shah mAb PBPK BC.txt")
131
132 modelODE <- IQRmodel("Shah mAb PBPK.txt",FLAGsym=FALSE)
133 modelBC <- IQRmodel("Shah mAb PBPK BC.txt",FLAGsym=FALSE)
134
135 # Simulate IV Single Dose 1 mg/kg ----
136
137 dosing <- IQRdosing(TIME=0,AMT=0.028*1,ADM=1) # (mg) 1 mg/kg
138
139 resmodelODE <- sim_IQRmodel(model = modelODE,
140                             dosingTable = dosing,
141                             simtime = 20*24,
142                             opt_abstol = 1e-15,
143                             opt_reltol = 1e-15,
144                             opt_maxstep = 0.05)
145
146 resmodelBC <- sim_IQRmodel(model = modelBC,
147                            dosingTable = dosing,
148                            simtime = 20*24,
149                            opt_abstol = 1e-15,
150                            opt_reltol = 1e-15,
151                            opt_maxstep = 0.05)
152
```

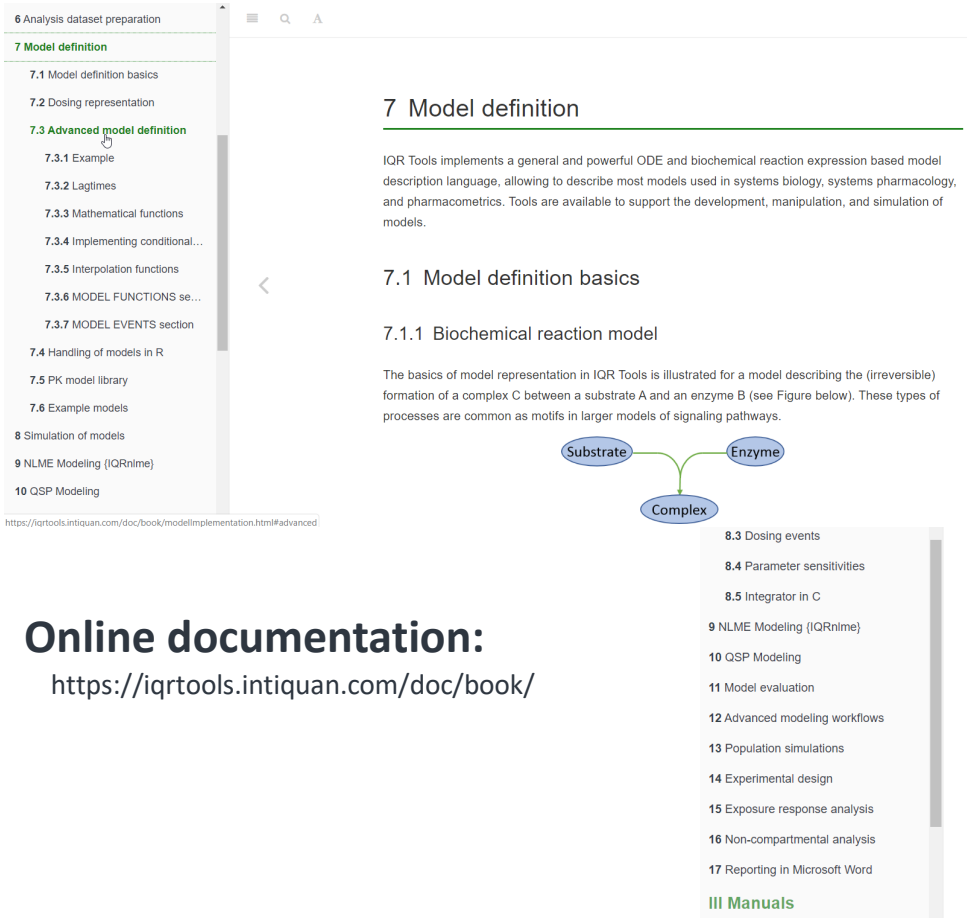
Mouse: 0.028 kg assumed weight

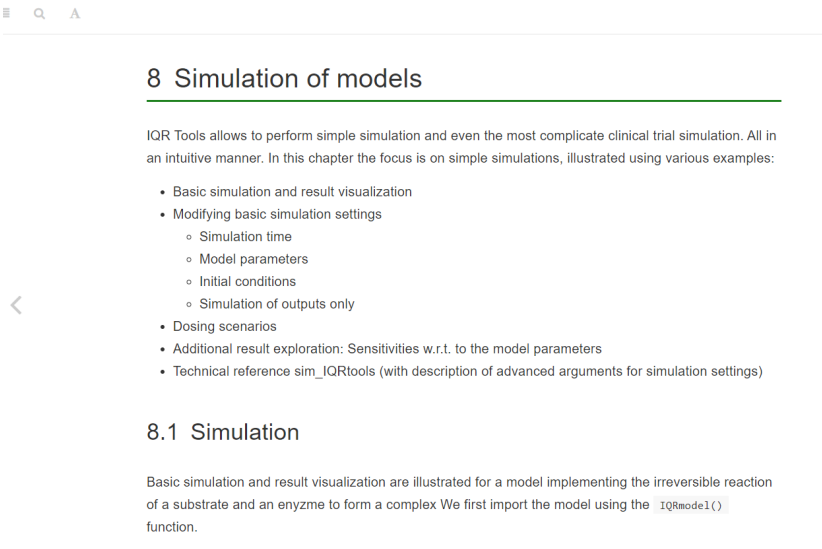
Example0_Models/Scripts/SCRIPT_01_models.R



IntiQuan

MORE INFORMATION ON THE MODEL DEFINITION SYNTAX AND SIMULATION AVAILABLE AS ONLINE AND OFFLINE DOCUMENTATION



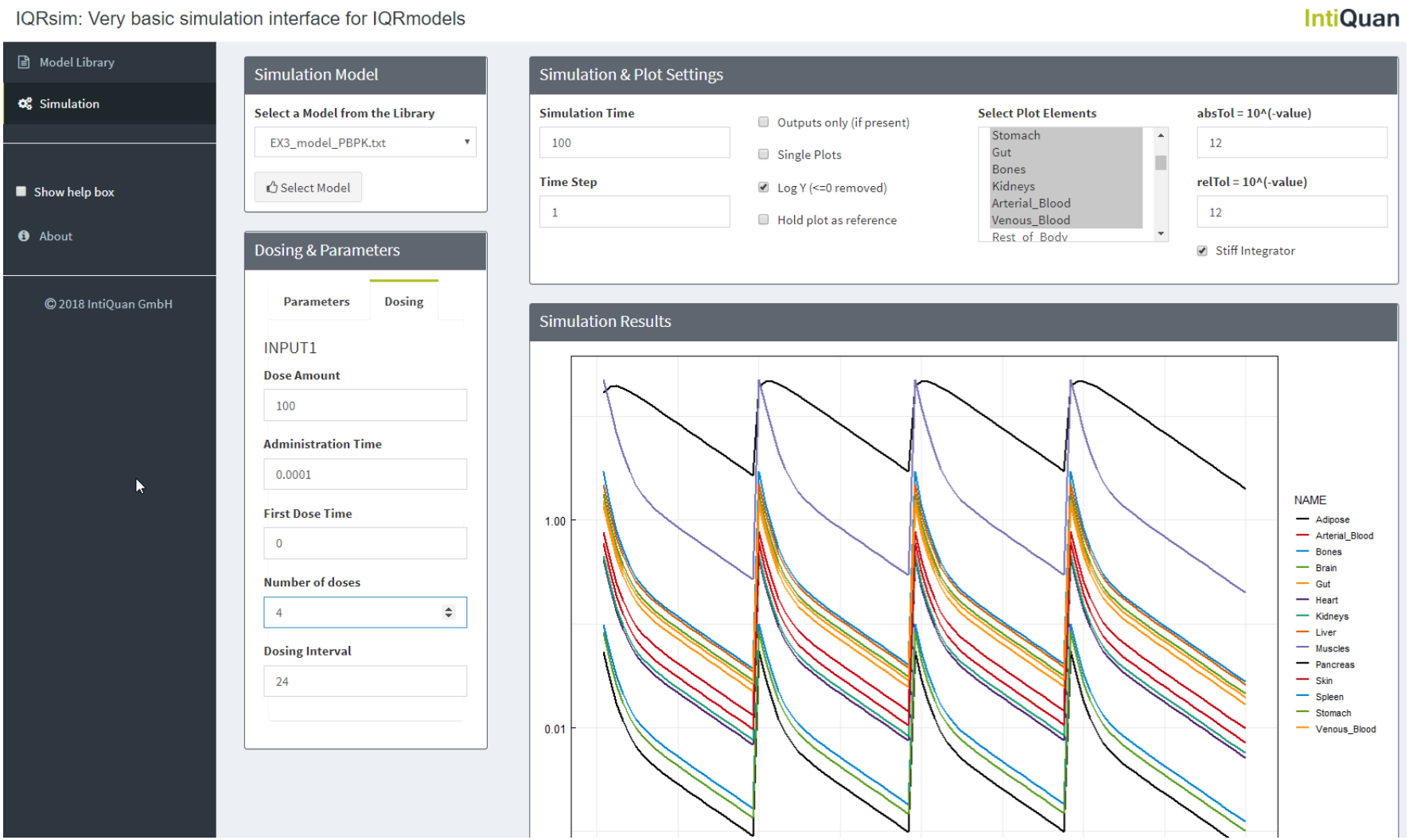


Online documentation:
https://iqrtools.intiquan.com/doc/book/

Offline documentation: `doc_IQRtools()`

IntiQuan

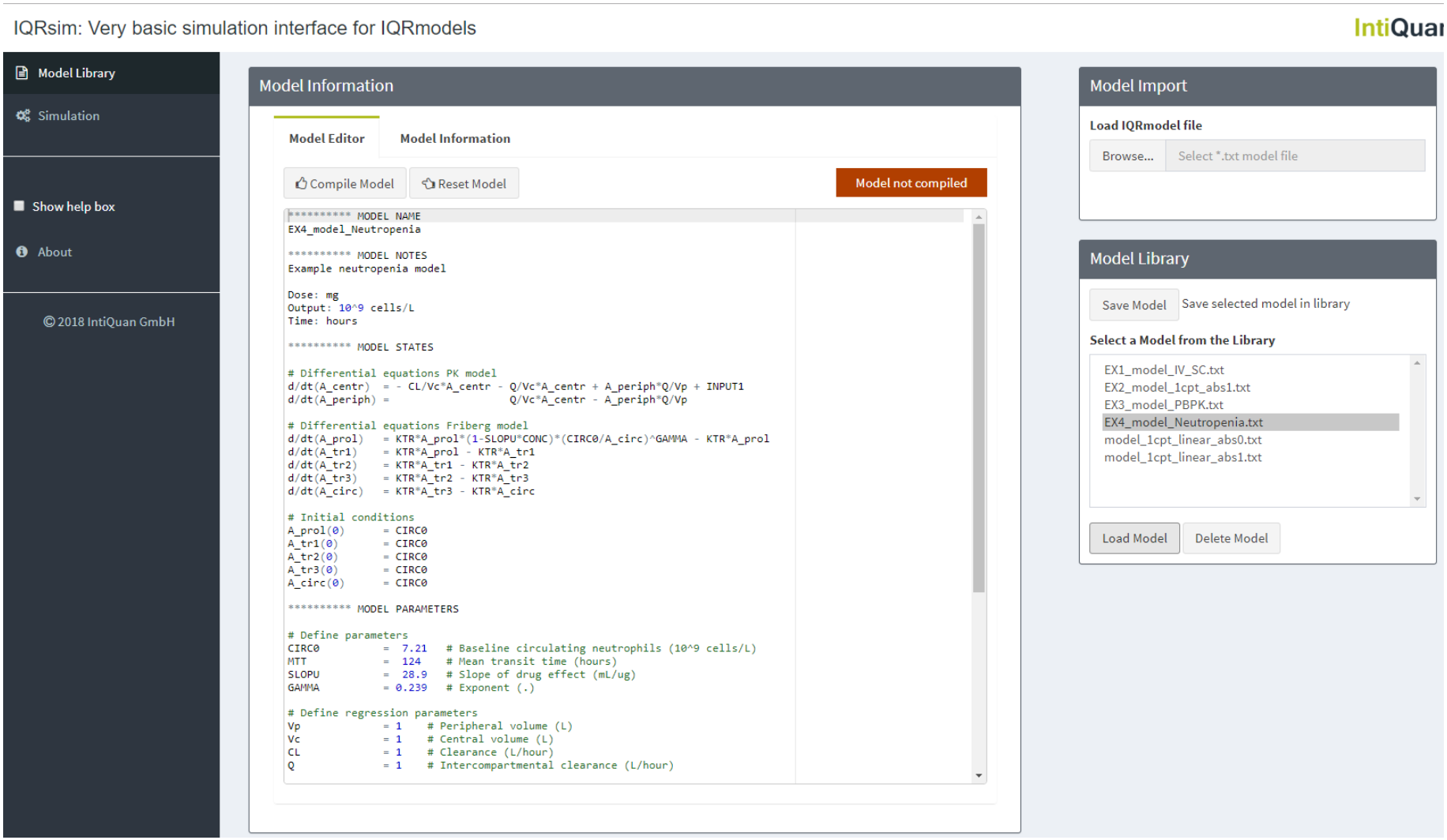
SIMULATION SHINY APP



Start by executing **install_IQRsim()**
run_IQRsim()

IntiQuan

IQRsim ALSO CONTAINS A MODEL EDITOR AND MODEL LIBRARY



run_IQRsim(model) adds an IQRmodel to the library and starts the Shiny App

IntiQuan

HANDS-ON EXERCISES

- Opening the IQR Tools documentation page

```
R> doc_IQRtools()
```

- Execution of an example in the documentation

```
R> examples_IQRtools()
```

- Write your own model
  - With a dosing input
  - Or get a model from SimBiology into IQR Tools

- Simulate that model
  - With a defined dosing scheme
  - For different parameters

THE WORKSHOP EXAMPLE

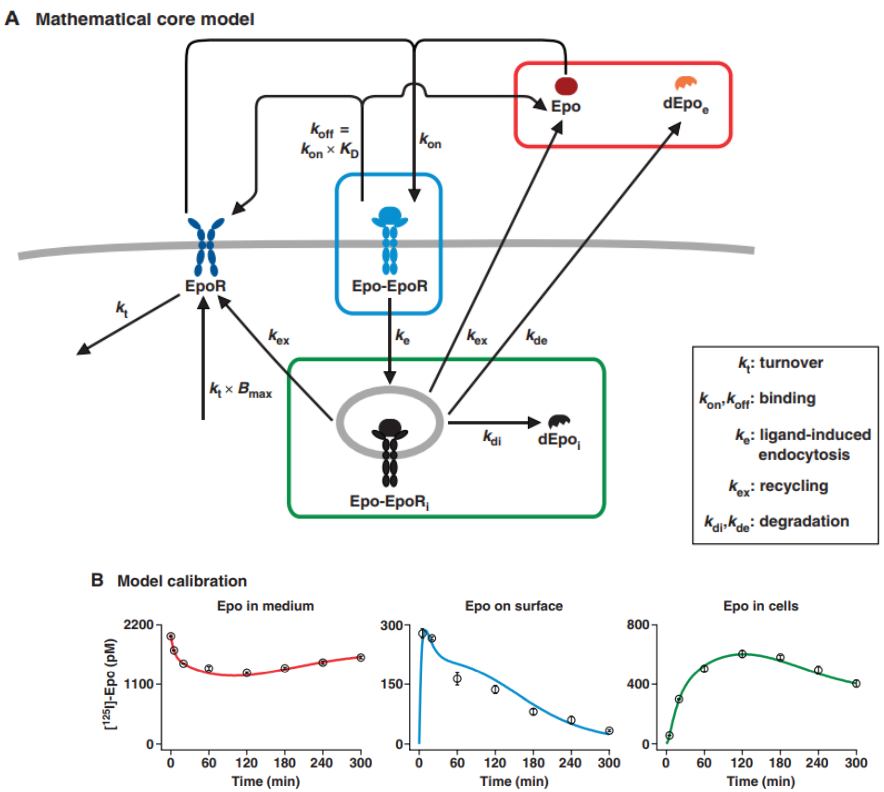
EPO DEGRADES UPON RECEPTOR BINDING AND INTERNALIZATION

REPORTS

Covering a Broad Dynamic Range: Information Processing at the Erythropoietin Receptor

Verena Becker,[1,2]* Marcel Schilling,[1]* Julie Bachmann,[1] Ute Baumann,[1] Andreas Raue,[3] Thomas Maiwald,[3]† Jens Timmer,[3,4,5] Ursula Klingmüller[1,2]‡

www.sciencemag.org SCIENCE VOL 328 11 JUNE 2010



- EpoR is newly synthesized (k_t B_{max})
- EpoR is internalized (k_t)

- Epo binds to EpoR (k_{on})
- Epo-EpoR dissociate ($k_{off} = k_{on}$ K_D)
- Epo-EpoR internalizes (k_e)

- Epo-EpoR$_i$ recycles into EpoR + Epo (k_{ex})
- Epo degrades and stays in intracellular compartment (k_{di})
- Epo degrades and is released to extracellular compartment (k_{de})

IntiQuan

WE WILL LEARN HOW TO …

Based on the publication

- Implement the model in IQR Tools
- Import the published data
- Estimate parameters
- Analyze identifiability of estimated parameters via the profile likelihood

Based on simulated data

- Introduce the concept of "conditions"
  (here: different Epo preparations with different receptor binding affinity)
- Jointly estimate parameters from multi-condition data

MODELING AND SIMULATION

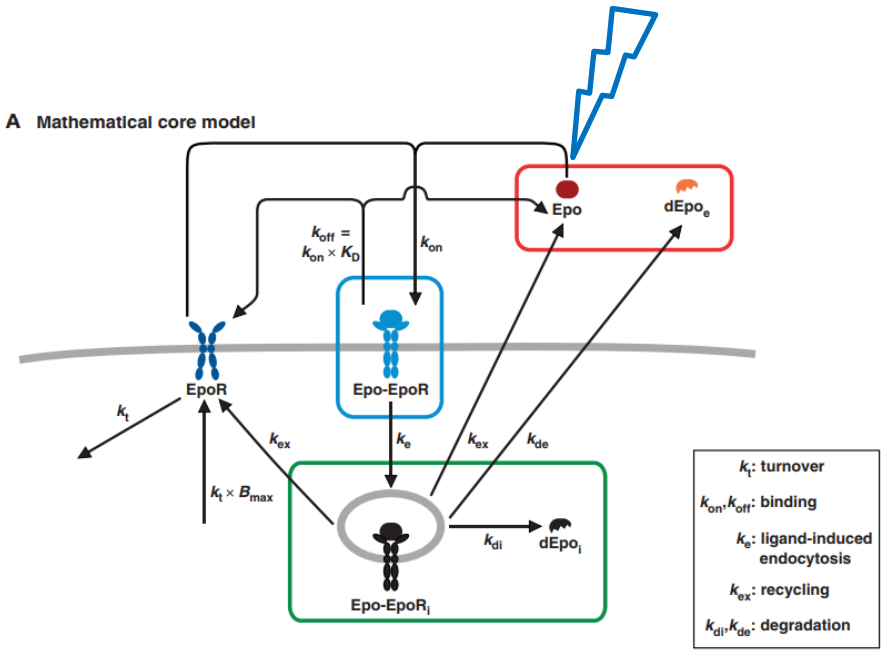
Example1_Epo/Scripts/SCRIPT_01_simulation.R

SETTING UP THE MODEL DESCRIPTION (ODES)

See: Example1_Epo/Scripts/Resources/model.txt

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

d/dt(Epo)       = initEpo*INPUT1 + EpoEpoRi*kex + EpoEpoR*koff - Epo*EpoR*kon
d/dt(EpoR)      = EpoEpoRi*kex + EpoEpoR*koff - EpoR*kt + initEpo*initEpoRrel*kt - Epo*EpoR*kon
d/dt(EpoEpoR)   = Epo*EpoR*kon - EpoEpoR*koff - EpoEpoR*ke
d/dt(EpoEpoRi)  = EpoEpoR*ke - EpoEpoRi*kdi - EpoEpoRi*kde - EpoEpoRi*kex
d/dt(dEpoi)     = EpoEpoRi*kdi
d/dt(dEpoe)     = EpoEpoRi*kde

Epo(0)          = 0
EpoR(0)         = initEpo*initEpoRrel
EpoEpoR(0)      = 0
EpoEpoRi(0)     = 0
dEpoi(0)        = 0
dEpoe(0)        = 0


********** MODEL VARIABLES

Epoext = Epo + dEpoe
Epoint = EpoEpoRi + dEpoi

Epoextcpm = log10(offset + (Epoext*scale)/initEpo)
Epomemcpm = log10(offset + (EpoEpoR*scale)/initEpo)
Epointcpm = log10(offset + (Epoint*scale)/initEpo)

OUTPUT1 = Epoextcpm
OUTPUT2 = Epomemcpm
OUTPUT3 = Epointcpm
```

Epo can be administered ("dosing input")



Keyword for outputs/observations - which will be used to link the model to the data for estimation and comparison

IntiQuan

IQRsysModel() TRANSLATES THE MODEL FILE INTO AN R OBJECT FOR QSP MODELING

```
R> # Create sysModel object
R> sysobj <- IQRsysModel("Resources/model.txt")

R> # Get basic information
R> print(sysobj)
R> sysobj
```

```
--- IQRsysModel object - basic information: -----------

Model:       model.txt
Dosing:      No dosing.
Data:        No data
Specs:       No model specification

Conditions:  BASE

--- Additional information stored: --------------------

Multistart estimation results:  no
Profile likelihood analysis:  no
```

`IQRsysModel(model, ...)`

- Contains at least the **structural model** and the default **parameter values**
- More elements for a modeling project can be added:
  - Dosing
  - Data
  - Model specifications

- Elements of the object can be accessed, manipulated and added:
  - Parameters, simulations, optimization results, etc.

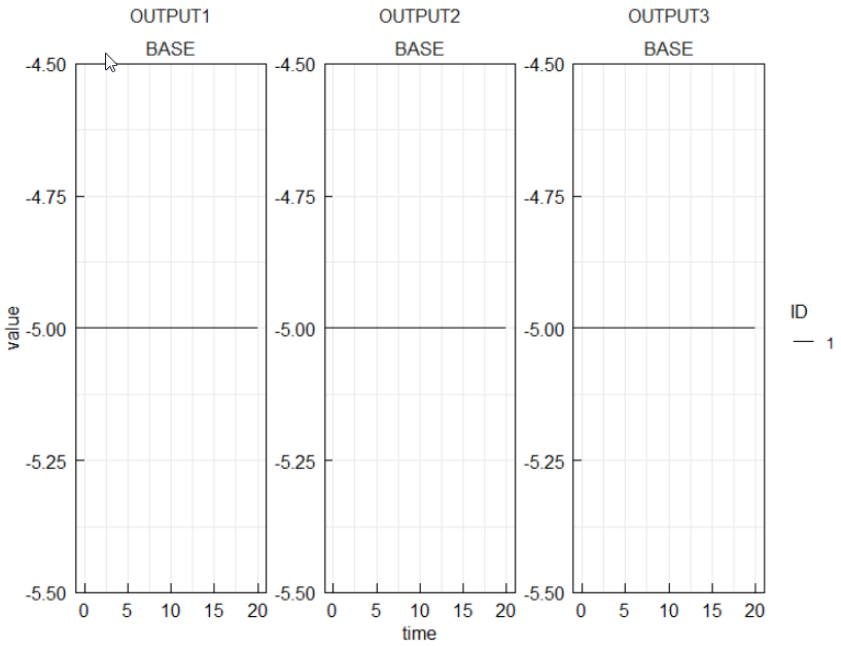
sim_IQRsysModel() LETS YOU PERFORM MODEL SIMULATION

```
R> # Add simulation to the sys object and plot
R> sysobj <- sim_IQRsysModel(sysobj)

R> plot_IQRsysModel(sysobj)
```

Explore further options with
?sim_IQRsysModel()

Examples:

- simtime = seq(1, 20, .1)
- FLAGoutputsOnly = TRUE

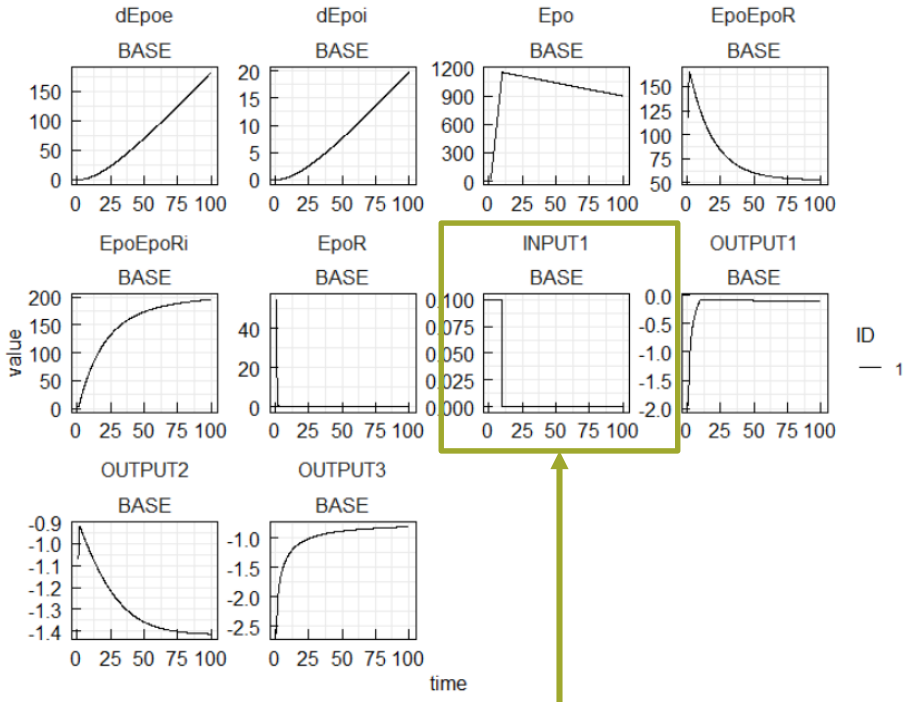


Pretty trivial simulation result …

USE THE dosing ARGUMENT OF IQRsysModel() TO DEFINE MODEL INPUTS/DOSING/STIMULUS

Basic dosing is defined via IQRdosing()

ADM	which INPUT
TIME	begin of dosing
TINF	duration of dosing
AMT	amount




```
R> # Define dosing
R> mydosing <- IQRdosing(TIME = 0,
                         ADM = 1,
                         TINF = 10,
                         AMT = 1)

R> # Provide dosing information to the sysModel
R> sysobj <- IQRsysModel("Resources/model.txt",
                         dosing = mydosing)

R> # Create output
R> sysobj <- sim_IQRsysModel(sysobj, simtime = seq(1, 100, .1), FLAGoutputsOnly = FALSE)
R> plot_IQRsysModel(sysobj)
```

Note: this adds the dosing into the sysobj!

IntiQuan

getPars_IQRsysModel() LETS YOU PRINT AND ACCESS MODEL PARAMETERS

```
R> # Get parameter information of the model
R> getPars_IQRsysModel(sysobj)
```

```
--- Global parameters ---------------------------------

   Parameter          | Value    | Estimate
   ------------------------------------------
                      |          |
   **Error model**    |          |
                      |          |
   error_ADD1         | 1        | +L
   error_ADD2         | 1        | +L
   error_ADD3         | 1        | +L
                      |          |
   **Fixed effects**  |          |
                      |          |
   initEpo            | 1350     | -
   initEpoRrel        | 0.129    | -
   kde                | 0.012    | -
   kdi                | 0.0013   | -
   ke                 | 0.055    | -
   kex                | 0.00058  | -
   koff               | 0.081    | -
   kon                | 0.15     | -
   kt                 | 0.016    | -
   offset             | 1e-05    | -
   scale              | 0.98     | -

--- No local parameters defined -----------------------

Values rounded to 4 significant digits.
Estimated/fixed parameters (+/-)
Estimation on (N) natural / (L) log / (G) logit scale.
```

```
R> # Store parameter values in vector
R> mypars <- getPars_IQRsysModel(sysobj)
R> mypars
```

```
R> # Selectively get parameters
R> mypars <- getPars_IQRsysModel(sysobj,
                                 "kde", "ke", "kon")
R> mypars
```


IntiQuan

setPars_IQRsysModel() LETS YOU CHANGE MODEL PARAMETERS

```
R> # Selectively set parameters
R> sysobj <- setPars_IQRsysModel(sysobj,
                kde = 0.05,
                ke  = 0.1,
                kon = 0.21)
```

```
--- Global parameters ---------------------------------

   Parameter          | Value    | Estimate
   ------------------------------------------
                      |          |
   **Error model**    |          |
                      |          |
   error_ADD1         | 1        | +L
   error_ADD2         | 1        | +L
   error_ADD3         | 1        | +L
                      |          |
   **Fixed effects**  |          |
                      |          |
   initEpo            | 1350     | -
   initEpoRrel        | 0.129    | -
   kde                | 0.05     | -
   kdi                | 0.0013   | -
   ke                 | 0.1      | -
   kex                | 0.00058  | -
   koff               | 0.081    | -
   kon                | 0.21     | -
   kt                 | 0.016    | -
   offset             | 1e-05    | -
   scale              | 0.98     | -

--- No local parameters defined -----------------------

Values rounded to 4 significant digits.
Estimated/fixed parameters (+/-)
Estimation on (N) natural / (L) log / (G) logit scale.
```

CONDITION-SPECIFIC PARAMETERS ARE TYPICAL FOR QSP APPLICATIONS

Example 1: Inhibitor

- inhibited reactions have smaller rate constant

```
CONDITIONS = (base, inhibitor)
```

Example 2: Knock-out

- affected targets have zero/small initial values

```
CONDITIONS = (base, target1, target2, …)
```

Here: Different Epo preparation

- receptor-binding rate can be affected

```
CONDITIONS = (Epo_Human, Epo_Recombinant)
```

HANDS-ON EXERCISES - CONDITIONS

- Can you think of other examples where condition specific (local) parameters are important to consider?
- Have you experienced this need in your own projects?

- **Conditions are related to categorical covariates**

USE THE modelSpec ARGUMENT OF IQRsysModel() TO DEFINE MULTI-CONDITIONAL MODELS

```
R> # Create sysModel with local parameters
R> sysobj <- IQRsysModel(
     model = "Resources/model.txt",
     dosing = IQRdosing(TIME = 0, ADM = 1, AMT = 1),
     modelSpec = list(
       LOCmodel = list(
         kon = c("Epo_Human", "Epo_Recombinant")
       ),
       LOCvalues = list(
         kon = c(Epo_Human = 1.5, Epo_Recombinant = 0.15)
       )
     )
   )
```

define which parameters have specific values for specific conditions

predefine values for local parameters

```
R> # Basic print-out shows conditions
R> sysobj
```

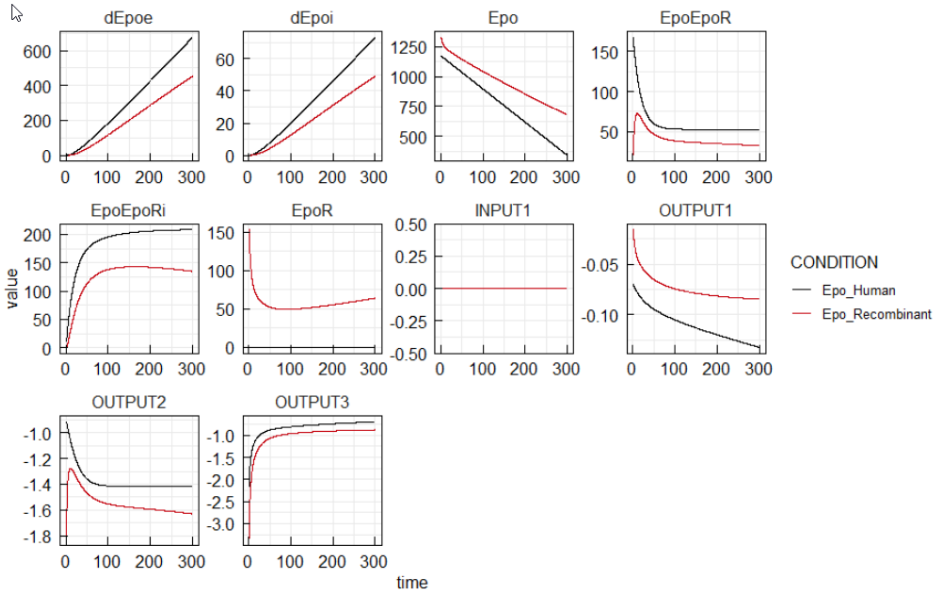
```
> sysobj
--- IQRsysModel object - basic information: ----------

Model:        Resources/model.txt
Dosing:       IQRdosing object
Data:         No data
Specs:        Model was specified.

Conditions:   Epo_Human
              Epo_Recombinant

--- Additional information stored: --------------------

Multistart estimation results:  no
Profile likelihood analysis:  no
```


IntiQuan

setPars_IQRsysModel() AND sim_IQRsysModel() FOR MULTI-CONDITIONAL MODELS

```
R> # Set condition-specific parameter values (2 equivalent ways)
R> sysobj <- setPars_IQRsysModel(sysobj, kon_Epo_Recombinant = 1e-5)
R> sysobj <- setPars_IQRsysModel(sysobj, kon = c(Epo_Recombinant = 1e-4))

R> # Create output, same dosing applied for all conditions
R> sysobj <- sim_IQRsysModel(sysobj,
                             simtime = seq(1, 300, .1),
                             FLAGoutputsOnly = FALSE)
```

```
R> plot_IQRsysModel(sysobj,
                    group = "CONDITION")
```



IntiQuan

USE THE dosing ARGUMENT OF IQRsysModel() TO IMPLEMENT CONDITION-SPECIFIC DOSING

```
R> # Create sysModel with local dosing and parameters
R> sysobj <- IQRsysModel(
     model = "Resources/model.txt",

     dosing = list(
       Epo_Human = IQRdosing(TIME = 0, ADM = 1, AMT = 1, TINF = 0),
       Epo_Recombinant = IQRdosing(TIME = 0, ADM = 1, AMT = 2, TINF = 10)
     ),

     modelSpec = list(
       LOCmodel = list(
         kon = c("Epo_Human", "Epo_Recombinant")
       ),
       LOCvalues = list(
         kon = c(Epo_Human = 1.5, Epo_Recombinant = 1e-4)
       )
     )
   )
```

- List of IQRdosing objects
- Refer to condition by name

Only single dose shown here - multiple dose used as example in script

IntiQuan

HANDS-ON EXERCISES

- Perform a simulation and plot the result
- Change the dosing and re-simulate and plot

LINKING MODELS AND DATA

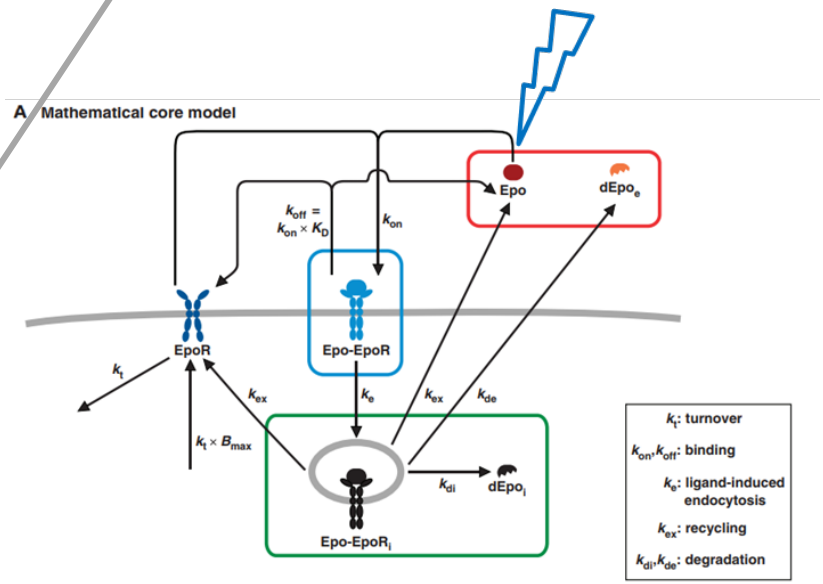
Example1_Epo/Scripts/SCRIPT_02_exploration.R

WORKSHOP EXAMPLE DATA

	A	B	C	D	E	F	G	H	I
1	ID	CONDITION	NAME	TIME	DV	YTYPE	ADM	AMT	TINF
2	1	Experiment_1	Epo_Input	0		0	1	1	0
3	1	Experiment_1	Epo_ext_cpm	0.82	-0.020931791	1			
4	1	Experiment_1	Epo_ext_cpm	0.82	-0.026385465	1			
5	1	Experiment_1	Epo_ext_cpm	0.82	-0.024223047	1			
6	1	Experiment_1	Epo_ext_cpm	5.82	-0.084467631	1			
7	1	Experiment_1	Epo_ext_cpm	5.82	-0.08214924	1			
8	1	Experiment_1	Epo_ext_cpm	5.82	-0.088621051	1			
9	1	Experiment_1	Epo_ext_cpm	20.82	-0.150362274	1			
10	1	Experiment_1	Epo_ext_cpm	20.82	-0.152233487	1			
11	1	Experiment_1	Epo_ext_cpm	20.82	-0.153436851	1			
12	1	Experiment_1	Epo_ext_cpm	60.82	-0.170195393	1			
27	1	Experiment_1	Epo_mem_cpm	0.82	-1.429135175	2			
28	1	Experiment_1	Epo_mem_cpm	0.82	-1.316224534	2			
29	1	Experiment_1	Epo_mem_cpm	0.82	-1.358778455	2			
30	1	Experiment_1	Epo_mem_cpm	5.82	-0.882141109	2			
31	1	Experiment_1	Epo_mem_cpm	5.82	-0.894373495	2			
32	1	Experiment_1	Epo_mem_cpm	5.82	-0.857377813	2			
33	1	Experiment_1	Epo_mem_cpm	20.82	-0.905144715	2			
51	1	Experiment_1	Epo_int_cpm	0.82	-3.185133265	3			
52	1	Experiment_1	Epo_int_cpm	0.82	-2.877880297	3			
53	1	Experiment_1	Epo_int_cpm	0.82	-3.087418117	3			
54	1	Experiment_1	Epo_int_cpm	5.82	-1.560285508	3			
55	1	Experiment_1	Epo_int_cpm	5.82	-1.577874104	3			
56	1	Experiment_1	Epo_int_cpm	5.82	-1.56562894	3			
57	1	Experiment_1	Epo_int_cpm	20.82	-0.846271858	3			
58	1	Experiment_1	Epo_int_cpm	20.82	-0.847746989	3			

Single Epo dose

Measurements



OUTPUT1 = Epoextcpm
OUTPUT2 = Epomemcpm
OUTPUT3 = Epointcpm

IntiQuan

HOW TO ORGANIZE DATA FOR MODELING? APPROACH IN IQR TOOLS

Use same data format across QSP and NLME modeling

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	CONDITION	ID	TIME	DV	YTYPE	AMT	ADM	TINF	NAME	IXGDF	USUBJID	MDV	EVID	CENS	RATE	TIMEPOS	TAD
2	Experiment_1	1	0	.	.	1	1	0	Epo_Input	1	SJ1	1	1	0	0	0	0
3	Experiment_1	1	0.82	-0.020931791	1	.	.	.	Epo_ext_cpm	2	SJ1	0	0	0	0	0.82	0.82
4	Experiment_1	1	0.82	-0.026385465	1	.	.	.	Epo_ext_cpm	3	SJ1	0	0	0	0	0.82	0.82
5	Experiment_1	1	0.82	-0.024223047	1	.	.	.	Epo_ext_cpm	4	SJ1	0	0	0	0	0.82	0.82
6	Experiment_1	1	0.82	-1.429135175	2	.	.	.	Epo_mem_cpm	26	SJ1	0	0	0	0	0.82	0.82
7	Experiment_1	1	0.82	-1.316224534	2	.	.	.	Epo_mem_cpm	27	SJ1	0	0	0	0	0.82	0.82
8	Experiment_1	1	0.82	-1.358778455	2	.	.	.	Epo_mem_cpm	28	SJ1	0	0	0	0	0.82	0.82
9	Experiment_1	1	0.82	-3.185133265	3	.	.	.	Epo_int_cpm	50	SJ1	0	0	0	0	0.82	0.82
10	Experiment_1	1	0.82	-2.877880297	3	.	.	.	Epo_int_cpm	51	SJ1	0	0	0	0	0.82	0.82
11	Experiment_1	1	0.82	-3.087418117	3	.	.	.	Epo_int_cpm	52	SJ1	0	0	0	0	0.82	0.82
12	Experiment_1	1	5.82	-0.084467631	1	.	.	.	Epo_ext_cpm	5	SJ1	0	0	0	0	5.82	5.82
13	Experiment_1	1	5.82	-0.08214924	1	.	.	.	Epo_ext_cpm	6	SJ1	0	0	0	0	5.82	5.82
14	Experiment_1	1	5.82	-0.088621051	1	.	.	.	Epo_ext_cpm	7	SJ1	0	0	0	0	5.82	5.82
15	Experiment_1	1	5.82	-0.882141109	2	.	.	.	Epo_mem_cpm	29	SJ1	0	0	0	0	5.82	5.82
16	Experiment_1	1	5.82	-0.894373495	2	.	.	.	Epo_mem_cpm	30	SJ1	0	0	0	0	5.82	5.82
17	Experiment_1	1	5.82	-0.857377813	2	.	.	.	Epo_mem_cpm	31	SJ1	0	0	0	0	5.82	5.82
18	Experiment_1	1	5.82	-1.560285508	3				Epo_int_cpm	53	SJ1	0	0	0	0	5.82	5.82

- Row based format with a single "event" in each row
- An event can be a dosing or an observation
- Additional columns can be added as covariates and/or for documentation and annotation of the data
- The "ID" column allows differentiation between subjects / experiments

This format is most flexible and covers all type of data that typically is considered in Pharmacometric and QSP modeling

IQRsysData() READS DATA IN THE NATURAL FORMAT USED BY IQRTOOLS FOR MODELING

```
R> dataSys <- IQRsysData("../Data/dataSYS.csv")
R> dataSys
```

	CONDITION	ID	TIME	DV	YTYPE	AMT	ADM	TINF	NAME	IXGDF	USUBJID	MDV	EVID	CENS	RATE	TIMEPOS	TAD
1	Experiment_1	1	0.00	NA	NA	1	1	0	Epo_Input	1	SJ1	0	1	0	0	0.00	0.00
2	Experiment_1	1	0.82	-0.02093179	1	NA	NA	NA	Epo_ext_cpm	2	SJ1	0	0	0	0	0.82	0.82
3	Experiment_1	1	0.82	-0.02638547	1	NA	NA	NA	Epo_ext_cpm	3	SJ1	0	0	0	0	0.82	0.82
4	Experiment_1	1	0.82	-0.02422305	1	NA	NA	NA	Epo_ext_cpm	4	SJ1	0	0	0	0	0.82	0.82
5	Experiment_1	1	0.82	-1.42913517	2	NA	NA	NA	Epo_mem_cpm	26	SJ1	0	0	0	0	0.82	0.82
6	Experiment_1	1	0.82	-1.31622453	2	NA	NA	NA	Epo_mem_cpm	27	SJ1	0	0	0	0	0.82	0.82

- **CONDITION** condition identifier (character)
- **ID** subject identifier (integer) (**here: 1-to-1 relation between CONDITION and ID**)
- **TIME** time since first dose (numeric)
- **DV** observed value (numeric), NA or 0 for doses, LLOQ value if CENS = 1
- **YTYPE** number of the output (integer, 1 for OUTPUT1, 2 for OUTPUT2, etc.)
- **AMT** dose amount (numeric)
- **ADM** number of input (integer, 1 for INPUT1, 2 for INPUT2, etc.)
- **TINF** infusion time (numeric)
- **NAME** description of the record (character)
- **IXGDF** index number of the record (integer)
- **USUBJID** unique subject ID (character)
- **MDV** missing data flag (1-records will be removed for parameter estimation)
- **EVID** Event ID (0 for observations, 1 for dosing records)
- **CENS** censoring flag (0 for uncensored, 1 for censored (below lower limit of quantification))
- **RATE** only used by NONMEM
- **TIMEPOS** time from first event per subject (numeric)
- **TAD** time after previous dose

DEFINING DOSING IN THE DATA

CONDITION	ID	TIME	DV	YTYPE	AMT	ADM	TINF	NAME	IXGDF	USUBJID	MDV	EVID	CENS	RATE	TIMEPOS	TAD
		10	NA	NA	1	1	0				1	1	0	0	0	0

Dosing columns:

- TIME time of dosing (first dose in subject has TIME=0)
- AMT dose amount
- ADM number of input (1 for INPUT1, 2 for INPUT2, etc.)
- TINF duration of dosing / infusion time

Derived columns:

- set NA for DV (data value) and YTYPE (output identifier)
- MDV = 1 (treated like missing data value)
- EVID = 1 (means dosing event)
- CENS = 0 (no censoring)
- RATE = 0 (not used by IQRsysModel)
- TIMEPOS = 0 (time since first event in ID, depends on other events)
- TAD = 0 (time since last dose, which for a dose always is 0)

Other columns:

- Set according to CONDITION, ID, USUBJID of the subject
- NAME: informative name for dosing record, e.g. "Epo Dose"

DEFINING OBSERVATIONS IN THE DATA

CONDITION	ID	TIME	DV	YTYPE	AMT	ADM	TINF	NAME	IXGDF	USUBJID	MDV	EVID	CENS	RATE	TIMEPOS	TAD
		0.82	2.1	1	NA	NA	NA				0	0	0	0	0.82	0.82

Data/Observation columns:

- TIME time of observation relative to time of first dose in subject (0)
- DV numeric observation value. If value below the LLOQ then set DV=LLOQ and CENS=1
- YTYPE number of output (1 for OUTPUT1, 2 for OUTPUT2, etc.)

Derived columns:

- set NA for AMT, ADM and TINF (dosing columns)
- MDV = 0 data value 0 if not missing, 1 if missing or not to be considered
- EVID = 0 (observation event)
- CENS = 0 (0 if observation above LLOQ. Set to 1 if observation below LLOQ)
- RATE = 0 (not used by IQRsysModel)
- TIMEPOS time since first event in subject
- TAD time since previous dose

Other columns:

- Set according to CONDITION, ID, USUBJID of the subject
- NAME: informative name for dosing record, e.g. "Epo Dose"

DEFINING BLOQ OBSERVATIONS IN THE DATA

BLOQ: BELOW LIMIT OF QUANTIFICATION

CONDITION	ID	TIME	DV	YTYPE	AMT	ADM	TINF	NAME	IXGDF	USUBJID	MDV	EVID	CENS	RATE	TIMEPOS	TAD
		9.82	0.1	1	NA	NA	NA				0	0	1	0	0.82	0.82

Data/Observation columns:

- TIME time of observation relative to time of first dose in subject (0)
- DV set data value to the **lower limit of quantification (LLOQ)**
- YTYPE number of output (1 for OUTPUT1, 2 for OUTPUT2, etc.)
- CENS = 1 (indicate that data point is censored -> will use BLOQ estimation method

Derived columns:

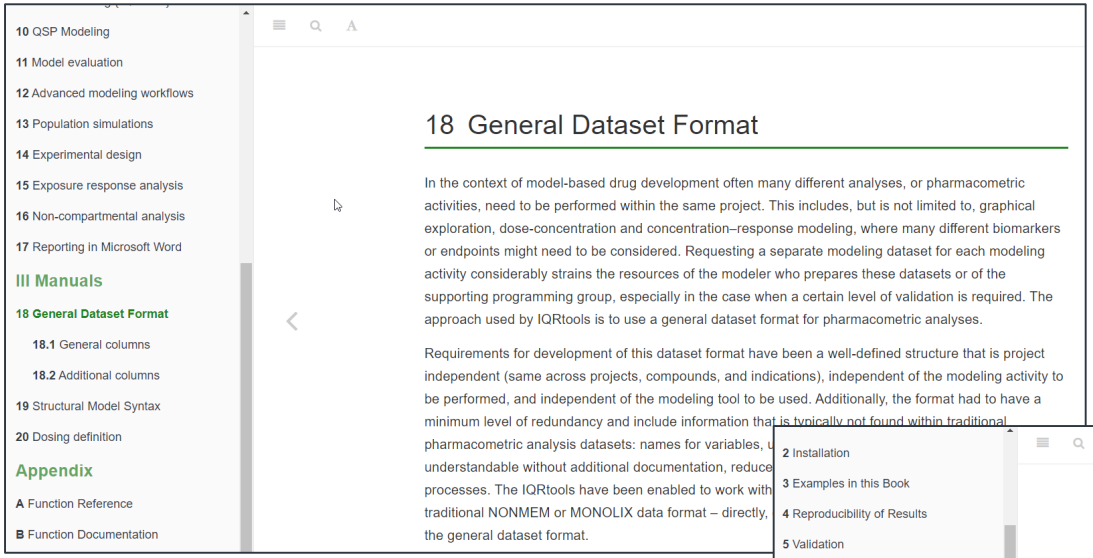
- set NA for AMT, ADM and TINF (dosing columns)
- MDV = 0 (data value not missing)
- EVID = 0 (means **no** dosing event)
- RATE = 0 (not used by IQRsysModel)
- TIMEPOS time since first event in subject
- TAD time since previous dose

Other columns:

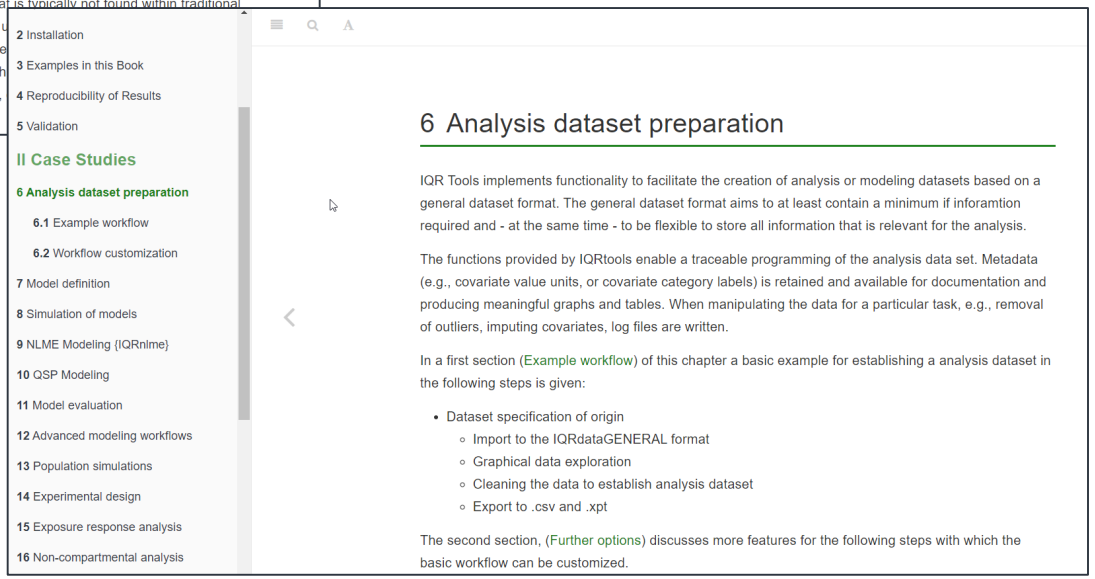
- Set according to CONDITION, ID, USUBJID of the subject
- NAME: informative name for dosing record, e.g. "Epo Dose"

STANDARDIZED GENERAL DATA FORMAT APPLICABLE TO BOTH QSP AND NLME MODELING FACILITATES DATA PROGRAMMING AND GENERATION OF QSP AND NLME MODELING DATASETS

https://iqrtools.intiquan.com/doc/book/GenDatFormat.html



More about it later



https://iqrtools.intiquan.com/doc/book/analysis-dataset-preparation.html

IntiQuan

IQRdataGENERAL() OR import_IQRsysData() HELP TO SIMPLIFY DATA PREPARATION

1. Embedding in IQR Tools

- IQR QSP tools are embedded in IQR tools
- IQR tools provided **general dataset format** via `IQRdataGENERAL()`
- `exportSYS_IQRdataGENERAL()` exports general data set to modeling data set

2. Perform standard imputation

- Many columns can be imputed automatically (MDV, EVID, etc.)
- Only define absolutely necessary columns
- Use `import_IQRsysData()` to import as `IQRsysData` object
- Clinical data often contains repeated doses - they can be efficiently coded in the dataset in a single row - import_IQRsysData() will expand these for further processing

	ID	CONDITION	NAME	TIME	DV	YTYPE	ADM	AMT	TINF
1	1	Experiment_1	Epo_Input	0.00	NA	0	1	1	0
2	1	Experiment_1	Epo_ext_cpm	0.82	-0.02093179	1	NA	NA	NA
3	1	Experiment_1	Epo_ext_cpm	0.82	-0.02638547	1	NA	NA	NA
4	1	Experiment_1	Epo_ext_cpm	0.82	-0.02422305	1	NA	NA	NA
5	1	Experiment_1	Epo_ext_cpm	5.82	-0.08446763	1	NA	NA	NA
6	1	Experiment_1	Epo_ext_cpm	5.82	-0.08214924	1	NA	NA	NA

HANDS-ON EXERCISES

Example1_Epo/Scripts/SCRIPT_02_exploration.R

- Exercise: Have a look at "../Data/dataRAW.csv", import with import_IQRsysData() and compare output to raw dataset.

	A	B	C	D	E	F	G	H	I	J
1	ID	CONDITION	NAME	TIME	DV	YTYPE	ADM	AMT	TINF	
2	1	Experiment_1	Epo_Input	0		0	1	1	0	
3	1	Experiment_1	Epo_ext_cpm	0.82	-0.020931791	1				
4	1	Experiment_1	Epo_ext_cpm	0.82	-0.026385465	1				
5	1	Experiment_1	Epo_ext_cpm	0.82	-0.024223047	1				
6	1	Experiment_1	Epo_ext_cpm	5.82	-0.084467631	1				
7	1	Experiment_1	Epo_ext_cpm	5.82	-0.08214924	1				
8	1	Experiment_1	Epo_ext_cpm	5.82	-0.088621051	1				
9	1	Experiment_1	Epo_ext_cpm	20.82	-0.150362274	1				
10	1	Experiment_1	Epo_ext_cpm	20.82	-0.152233487	1				
11	1	Experiment_1	Epo_ext_cpm	20.82	-0.153436851	1				
12	1	Experiment_1	Epo_ext_cpm	60.82	-0.170195393	1				
13	1	Experiment_1	Epo_ext_cpm	60.82	-0.191715048	1				
14	1	Experiment_1	Epo_ext_cpm	60.82	-0.174728994	1				
15	1	Experiment_1	Epo_ext_cpm	120.82	-0.199590556	1				
16	1	Experiment_1	Epo_ext_cpm	120.82	-0.203916678	1				
17	1	Experiment_1	Epo_ext_cpm	120.82	-0.208689945	1				
18	1	Experiment_1	Epo_ext_cpm	180.82	-0.178569866	1				

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	CONDITION	ID	TIME	DV	YTYPE	AMT	ADM	TINF	NAME	IXGDF	USUBJID	MDV	EVID	CENS	RATE	TIMEPOS	TAD
2	Experiment_1	1	0	.	.	1	1	0	Epo_Input	1	SJ1	1	1	0	0	0	0
3	Experiment_1	1	0.82	-0.020931791	1	.	.	.	Epo_ext_cpm	2	SJ1	0	0	0	0	0.82	0.82
4	Experiment_1	1	0.82	-0.026385465	1	.	.	.	Epo_ext_cpm	3	SJ1	0	0	0	0	0.82	0.82
5	Experiment_1	1	0.82	-0.024223047	1	.	.	.	Epo_ext_cpm	4	SJ1	0	0	0	0	0.82	0.82
6	Experiment_1	1	0.82	-1.429135175	2	.	.	.	Epo_mem_cpm	26	SJ1	0	0	0	0	0.82	0.82
7	Experiment_1	1	0.82	-1.316224534	2	.	.	.	Epo_mem_cpm	27	SJ1	0	0	0	0	0.82	0.82
8	Experiment_1	1	0.82	-1.358778455	2	.	.	.	Epo_mem_cpm	28	SJ1	0	0	0	0	0.82	0.82
9	Experiment_1	1	0.82	-3.185133265	3	.	.	.	Epo_int_cpm	50	SJ1	0	0	0	0	0.82	0.82
10	Experiment_1	1	0.82	-2.877880297	3	.	.	.	Epo_int_cpm	51	SJ1	0	0	0	0	0.82	0.82
11	Experiment_1	1	0.82	-3.087418117	3	.	.	.	Epo_int_cpm	52	SJ1	0	0	0	0	0.82	0.82
12	Experiment_1	1	5.82	-0.084467631	1	.	.	.	Epo_ext_cpm	5	SJ1	0	0	0	0	5.82	5.82
13	Experiment_1	1	5.82	-0.08214924	1	.	.	.	Epo_ext_cpm	6	SJ1	0	0	0	0	5.82	5.82
14	Experiment_1	1	5.82	-0.088621051	1	.	.	.	Epo_ext_cpm	7	SJ1	0	0	0	0	5.82	5.82
15	Experiment_1	1	5.82	-0.882141109	2	.	.	.	Epo_mem_cpm	29	SJ1	0	0	0	0	5.82	5.82
16	Experiment_1	1	5.82	-0.894373495	2	.	.	.	Epo_mem_cpm	30	SJ1	0	0	0	0	5.82	5.82
17	Experiment_1	1	5.82	-0.857377813	2	.	.	.	Epo_mem_cpm	31	SJ1	0	0	0	0	5.82	5.82
18	Experiment_1	1	5.82	-1.560285508	3				Epo_int_cpm	53	SJ1	0	0	0	0	5.82	5.82

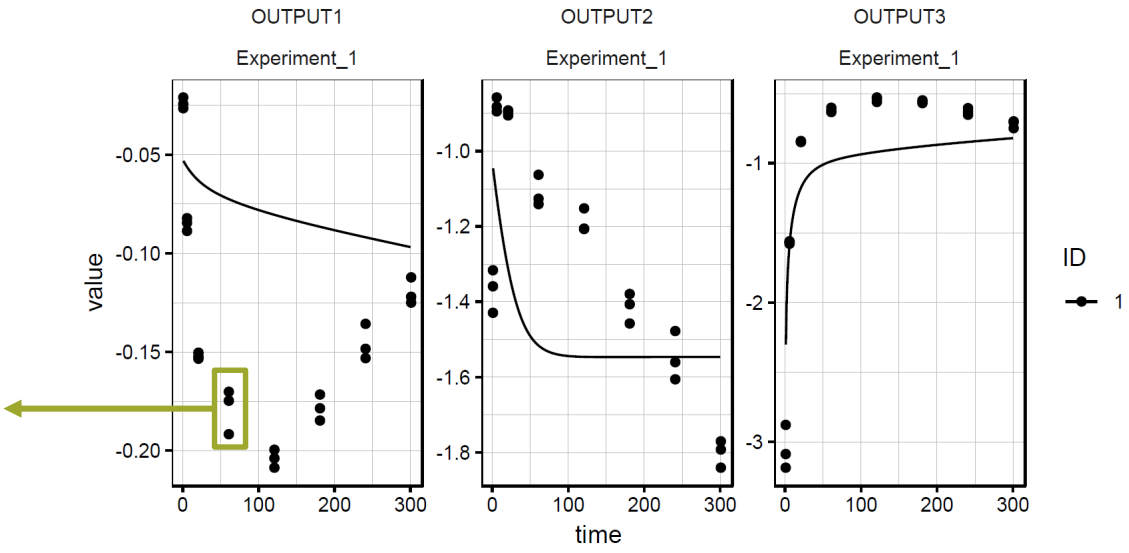
IntiQuan

USE THE data ARGUMENT OF IQRsysModel() TO LINK MODEL TO A DATA SET

```
R> # Load the model + data
R> sysobj <- IQRsysModel("Resources/model.txt",
                         data = list(datafile = "../Data/dataSYS.csv"))

R> # Simulate with user-defined time points
R> sysobj <- sim_IQRsysModel(sysobj, simtime = 1:300)

R> # Plot model + data
R> plot_IQRsysModel(sysobj)
```

data:

- measured triplicates per time point



IntiQuan

GOING TOWARDS ESTIMATION WITH as_IQRsysEst()

```
R> est <- as_IQRsysEst(sysobj,
  modelSpec = list(
    POPestimate = c(
      initEpo     = 0,
      initEpoRrel = 1,
      kde         = 1,
      kdi         = 1,
      ke          = 1,
      kex         = 1,
      koff        = 1,
      kon         = 1,
      kt          = 1
    )
  )
)
```

until now contained information about

- model, error model
- parameter values (= initial guesses)
- data

did not contain information about

- which parameters to consider for estimation
- on which scale (log assumed if not explicitly states)

```
R> # Print information collected in the estimation object
R> print(est)
```

ROBUST PARAMETER ESTIMATION

Example1_Epo/Scripts/SCRIPT_03_estimation.R

modelSpec REVISITED (SEE ?IQRnlmeEst())

QSP essential (same for NLME) | QSP extended (same for NLME) | QSP (no NLME)

```
modelSpec = list(

  # Population level
  POPvalues0       = Parameter values / initial guesses
  POPestimate      = Estimate or not
  # Individual level
  IIVdistribution  = Parameterize on log/normal/logit scale
  IIVvalues0       = SD of individual parameter values
  IIVestimate      = Estimate SD?
  # Residual error level
  errorModel       = Error model for each model output
  # Covariate level
  covariateModel   = Which parameter is affected by which covariate
  covariateModelValues = Effect size parameter
  COVestimate      = Estimate covariate effect or not
  COVcentering     = Reference values for covariates
  # Condition-Specific level
  LOCmodel         = Which parameter is condition-specific
  LOCvalues0       = Values per condition for specific parameters
  LOCestimate      = Estimate or not
```

) → Provide to IQRsysModel() or as_IQRsysEst()

IntiQuan

USE IQRsysProject() AND run_IQRsysProject() TO CREATE AND RUN ESTIMATION PROJECTS

```
R> # Load the model
R> sysobj <- IQRsysModel(
     model = "Resources/model.txt",
     data = list(datafile = "../Data/dataSYS.csv"),
     modelSpec = list(
       POPvalues0 = c(
         initEpo = 1350,          initEpoRrel = 0.01,        kde = 0.01,
         kdi     = 0.01,          ke          = 0.01,        kex = 0.01,
         koff    = 0.01,          kon         = 0.01,        kt  = 1
       ) ,
       errorModel = list(
         OUTPUT1 = c("abs", 0.1),    OUTPUT2 = c("abs", 0.1),     OUTPUT3 = c("abs", 0.1)
       )
     )
   )
```

- estimation object
- path information
- integrator options
- optimizer options
- optimization settings (bounds, spread)

```
R> # Create estimation object
R> est <- as_IQRsysEst(sysobj)
```

```
R> # Create project on hard disk
R> proj <- IQRsysProject(est,
                         projectPath = "../Models/RUN1",
                         opt.nfits = 1)
R> # Run the project
R> optsys <- run_IQRsysProject(proj, ncores = 1)
```

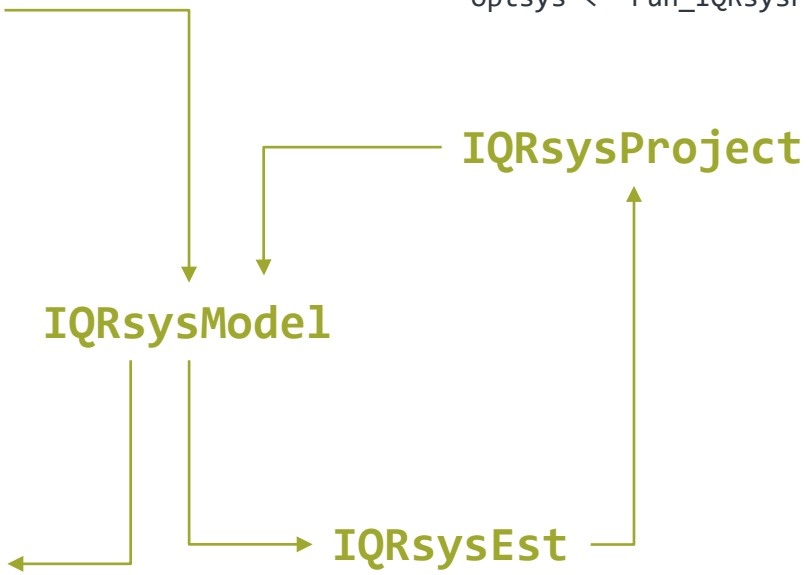
THE BIG PICTURE OF THE IQRsysModel WORKFLOW

- Load model, data and parameter information into IQRsysModel object

1

- **Manipulate** the model object:

```
sysobj <- setPars_IQRsysModel(sysobj, …)
sysobj <- sim_IQRsysModel(sysobj, …)
```



- Access information about estimated parameters and plot results
- **Manipulate** the model object:

4

```
optsys <- profile_IQRsysModel(optsys, …)
optsys <- switchOpt_IQRsysModel(optsys, …)
```

2

- Add missing estimation information via modelSpec

```
estobj <- as_IQRsysEst(sysobj, …)
```

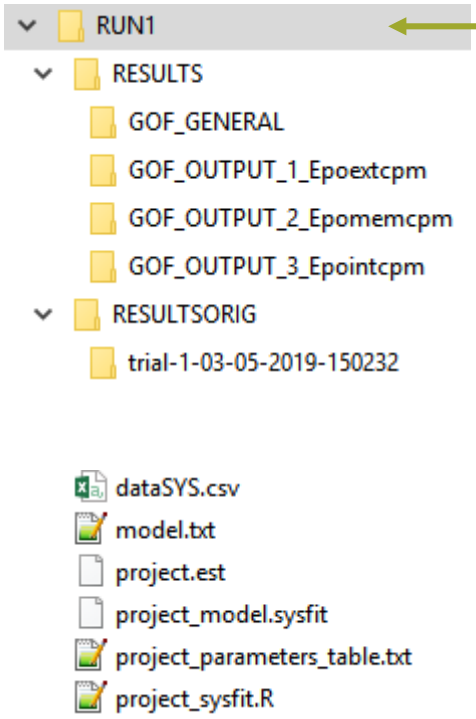
3

- Write to disk and run all computations. Return as **optimized IQRsysModel**.

```
proj <- IQRsysProject(estobj, …)
optsys <- run_IQRsysProject(proj, …)
```


IntiQuan

THE IQRsysProject FOLDER STRUCTURE



Project folder

- Results (output of IQRtools)
  - Goodness of Fit (GOF) plots
  - Output-specific GOF plots
- Original Results (output of estimation tool)

- Dataset
- Model file
- Estimation object
- Estimation tool generated object
- Estimation outcome (parameter values)
- Execution script to run the estimation

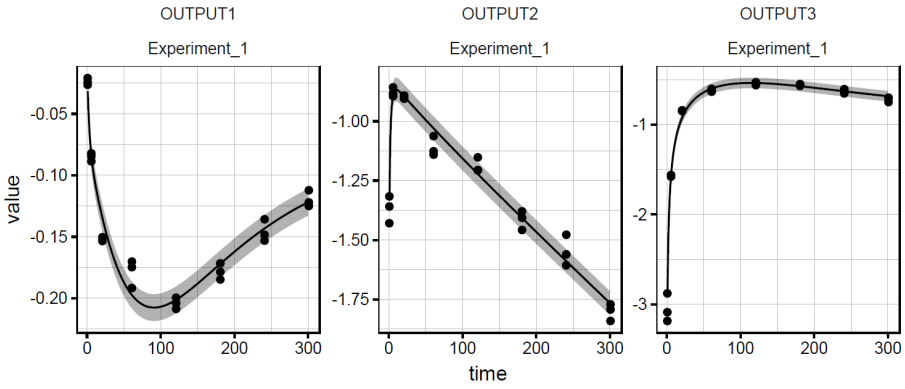
IntiQuan

run_IQRsysProject() RETURNS AN OPTIMIZED IQRsysModel

```
R> # Inspect estimated parameters and fit
R> getPars_IQRsysModel(optsys)
```

```
--- Global parameters ---------------------------------

   Parameter          | Value     | Estimate | RSE (%)
   ------------------------------------------------------
                      |           |          |
   **Error model**    |           |          |
                      |           |          |
   error_ADD1         | 0.01088   | +L       | 14.56
   error_ADD2         | 0.04966   | +L       | 14.59
   error_ADD3         | 0.0583    | +L       | 14.7
                      |           |          |
   **Fixed effects**  |           |          |
                      |           |          |
   initEpo            | 466.7     | +L       | 578.2
   initEpoRrel        | 0.03863   | +L       | 564.5
   kde                | 0.01233   | +L       | 4.641
   kdi                | 0.002077  | +L       | 16.42
   ke                 | 0.05406   | +L       | 3.536
   kex                | 0.001498  | +L       | 37.34
   koff               | 0.3053    | +L       | 27.61
   kon                | 0.003457  | +L       | 578.2
   kt                 | 50.88     | +L       | 686.4
   offset             | 1e-05     | -        | --
   scale              | 0.98      | -        | --

--- No local parameters defined -----------------------

Values rounded to 4 significant digits.
Estimated/fixed parameters (+/-)
Estimation on (N) natural / (L) log / (G) logit scale.
```

```
R> # Simulate model excluding t = 0
R> optsys <- sim_IQRsysModel(optsys, simtime = 1:300)
R> plot_IQRsysModel(optsys)
```



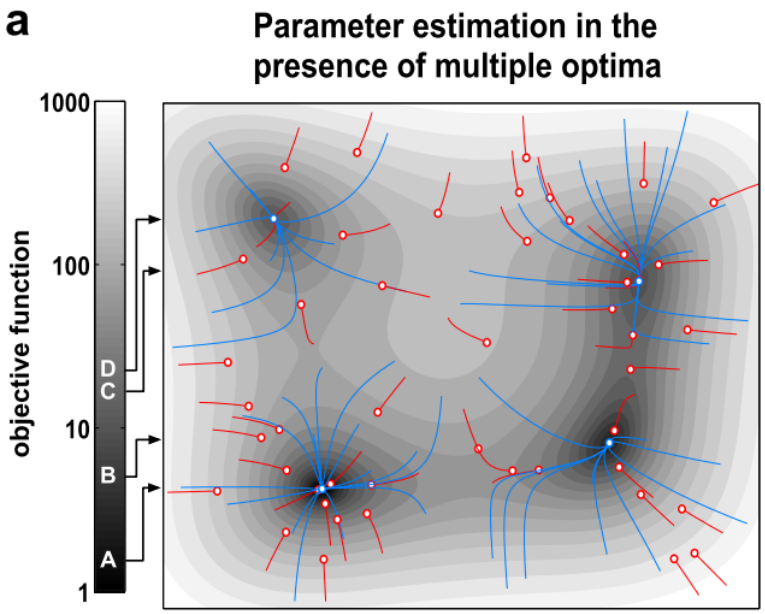
Is this the best we can get?

IntiQuan

OPTIMIZER IN IQR TOOLS DETECTS LOCAL OPTIMA

Maximum-likelihood estimation:

objective function $$l(\theta) = -2 \log L(\theta) = \sum_i \left[\left(\frac{y_i(\theta) - y_i^D}{\sigma_i} \right)^2 + \log(2\pi\sigma_i^2) \right]$$



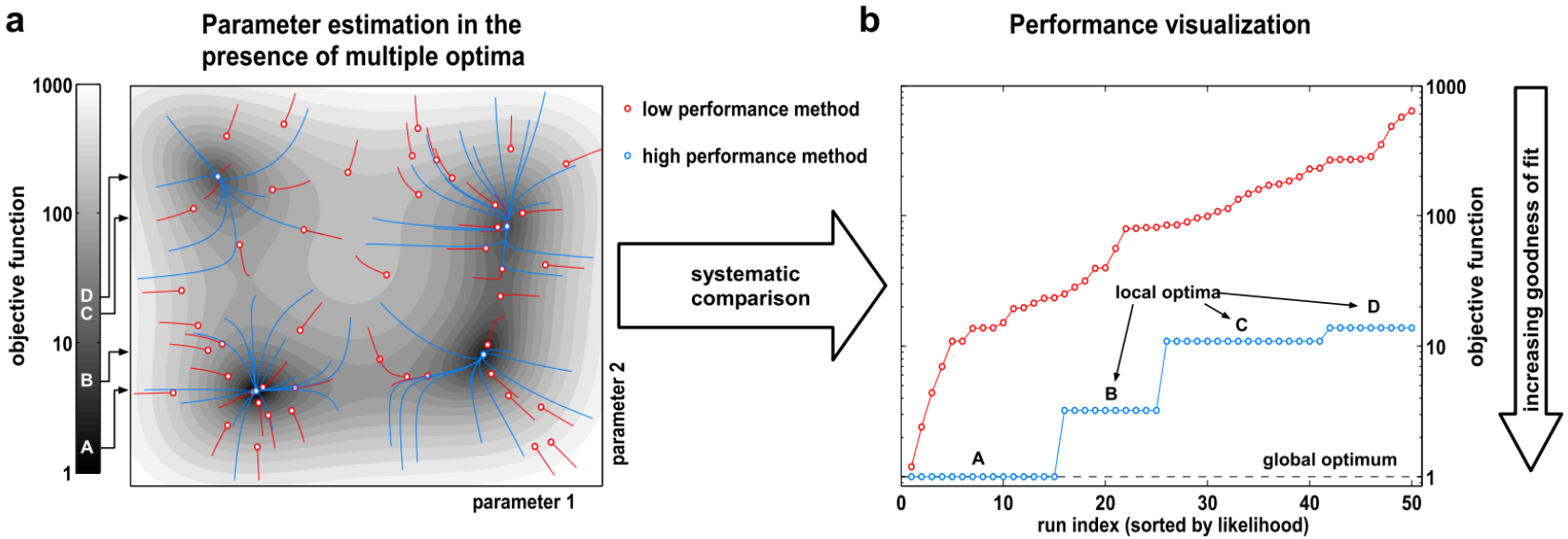
Raue, Andreas, et al. "Lessons learned from quantitative dynamical modeling in systems biology." *PloS one* 8.9 (2013): e74335.

Example with two parameters:

- Contour plot shows 4 minima
- **blue:** optimizer starts from different positions and converges to one of the local optima
  → **good optimizer**
- **red:** optimizer stops prematurely
  → **bad optimizer**

IntiQuan

KNOW WHEN YOU CAN TRUST YOUR MODEL



Raue, Andreas, et al. "Lessons learned from quantitative dynamical modeling in systems biology." *PloS one* 8.9 (2013): e74335.

Benefits of thorough optimization:

- Allows assessment if data could be explained in multiple different ways (#optima)
- Trust your model

IntiQuan

CREATE MULTI-START PROJECT WITH as_IQRsysProject(…, opt.nfits, opt.sd)

```
R> proj <- IQRsysProject(est, "../Models/RUN2",
                         opt.nfits = 36,
                         opt.sd = 3)
```

Schedule 36 fits from random positions in parameter space

Sample random parameters from normal distribution:

$N(t(\texttt{POPvalues0}), \texttt{opt.sd}^2)$

t(): transformation (no, log, logit)

More optimizer options:

```
opt.iterlim   maximal number of iterations
opt.parlower  lower parameter bound (e.g. error parameters)
opt.parupper  upper parameter bound
```

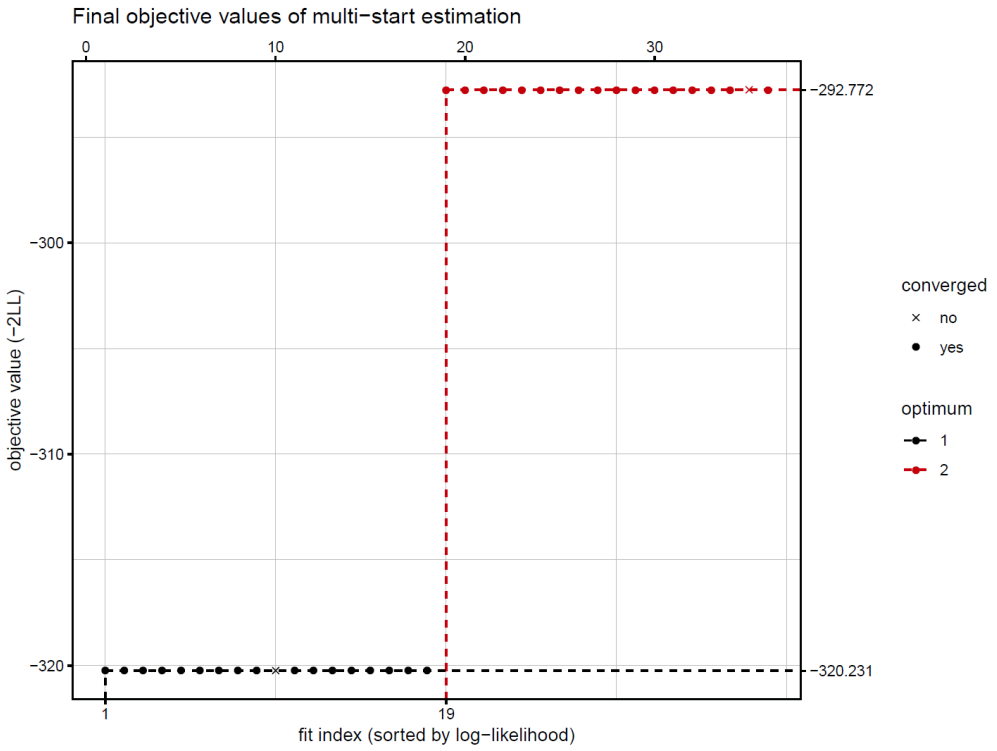
```
R> optsys <- run_IQRsysProject(proj,
                               ncores = 8)
```

Use 8 cores in parallel (one fit runs on one core)

`plotWaterfall_IQRsysModel(optsys)` SHOWS SORTED OBJECTIVE VALUES OF ALL FITS

- Shows the obj. value of each fit
- Sorts fits by obj. value
- Detects local optima by jumps in obj. value

- Highlights local optima by color
- Highlights fit index of local optima
- Highlights obj. values of local optima

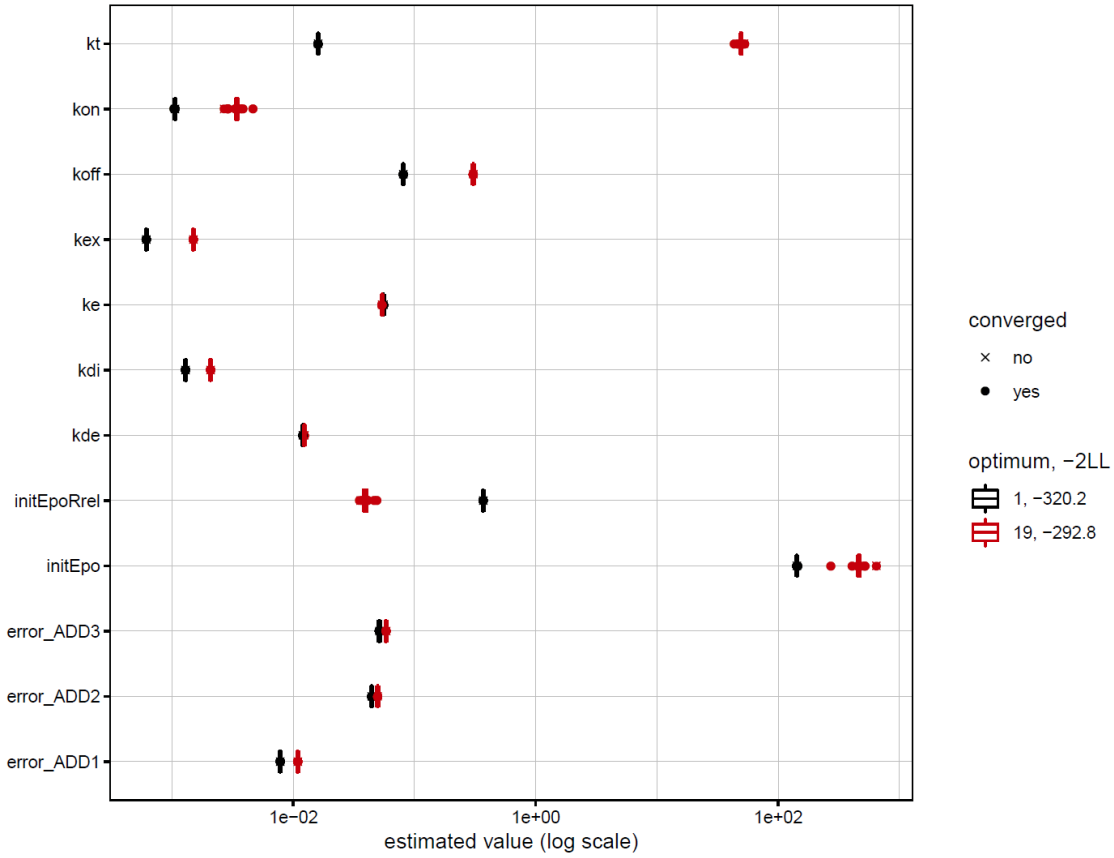


IntiQuan

`plotPars_IQRsysModel(optsys)` SHOWS PARAMETER ESTIMATES FOR ALL FITS

- Shows the estimated parameters of each fit as **dots and boxplot**
- Detects local optima by jumps in obj. value

- Highlights local optima by color
- Highlights fit index of local optima
- Highlights obj. values of local optima

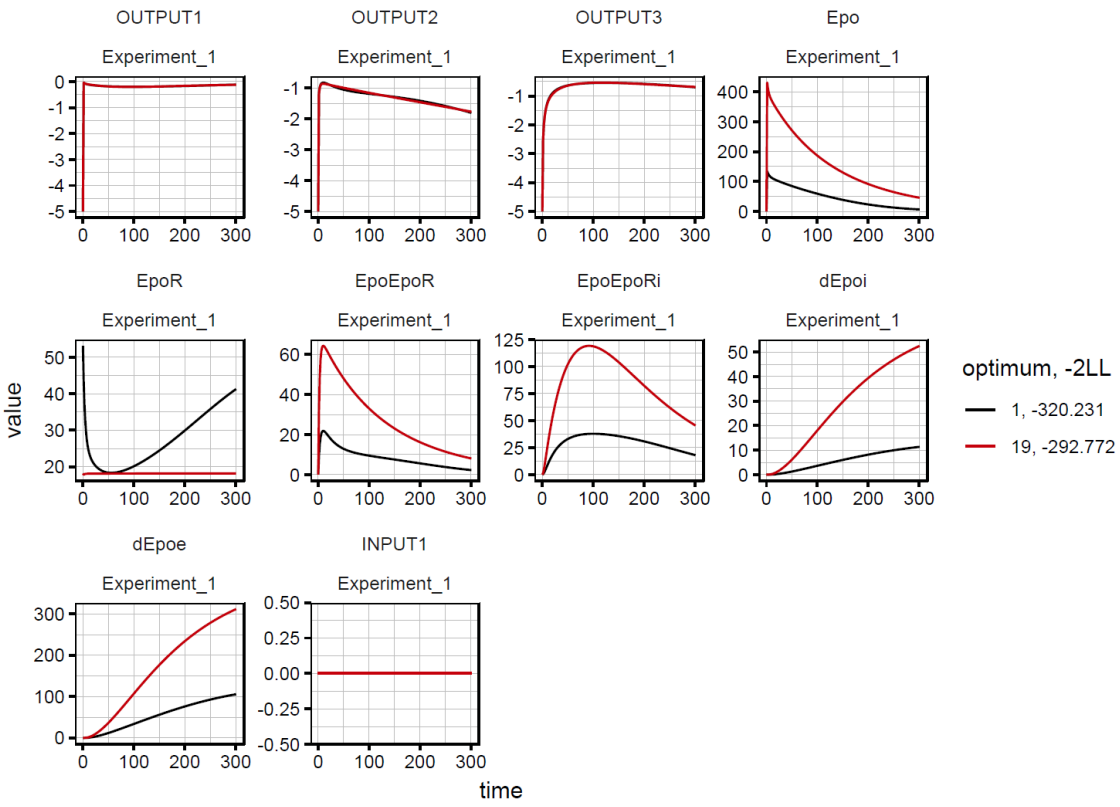


IntiQuan

plotPred_IQRsysModel(optsys) SHOWS MODEL TRAJECTORIES FOR ALL (DISTINCT) FITS

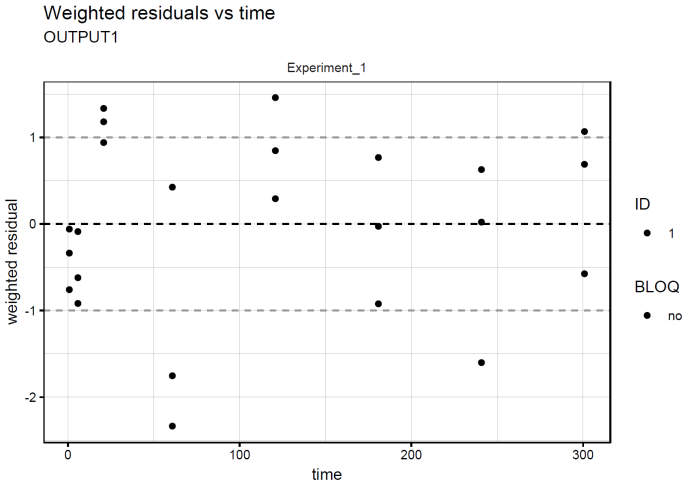
- Shows the estimated prediction of each unique fit
- Detects local optima by jumps in obj. value

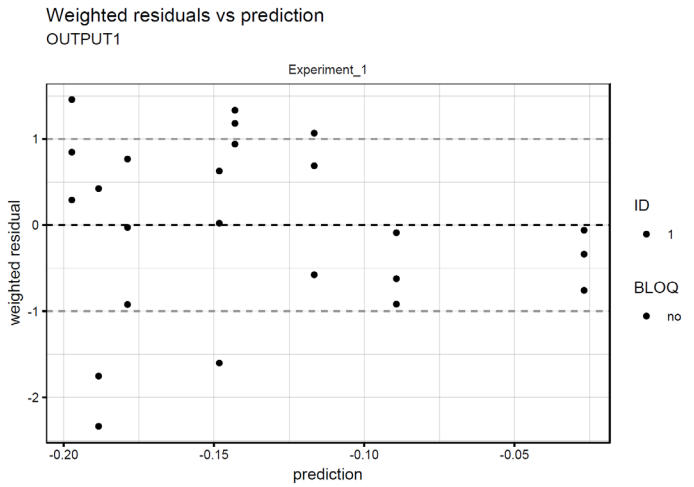
- Highlights local optima by color
- Highlights fit index of local optima
- Highlights obj. values of local optima

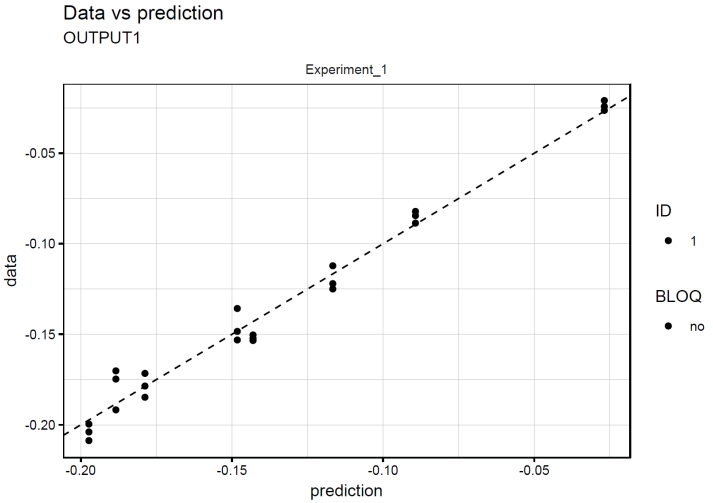


IntiQuan

OTHER DIAGNOSTIC PLOTS: `plotWRES_IQRsysModel(optsys)` AND `plotDVPRED_IQRsysModel(optsys)`

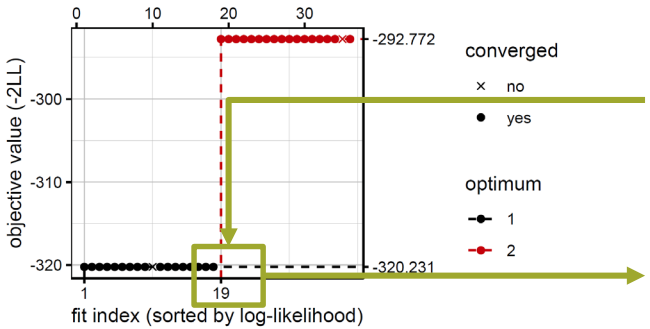






IntiQuan

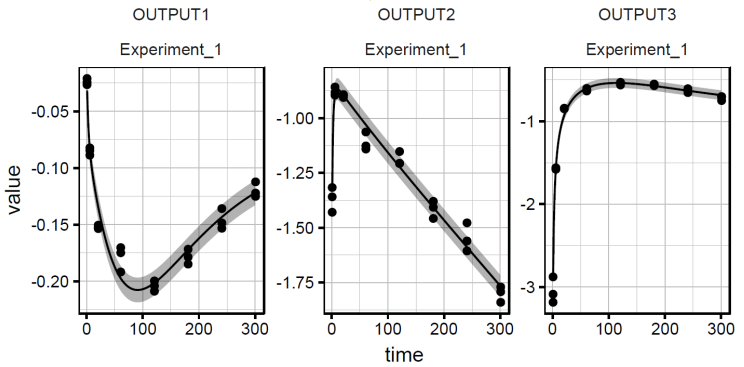
switchOpt_IQRsysModel() LETS YOU TAKE A CLOSE LOOK AT LOCAL OPTIMA




```
R> # Produce waterfall plot
R> plotWaterfall_IQRsysModel(optsys)

R> # Switch to another local optimum
R> optsys <- switchOpt_IQRsysModel(optsys, optimum = 19)

R> # Print parameters of the selected optimum
R> getPars_IQRsysModel(optsys)
```

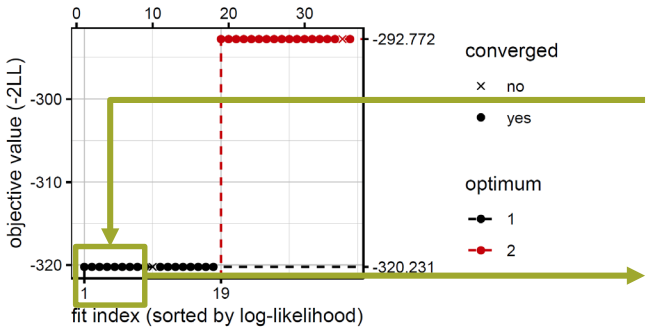
```
--- Selected optimum: 19 ----------------------------
--- Global parameters -------------------------------

   Parameter          | Value    | Estimate | RSE (%)
   ------------------------------------------------
                      |          |          |
   **Error model**    |          |          |
                      |          |          |
   error_ADD1         | 0.01088  | +L       | 14.55
   error_ADD2         | 0.04967  | +L       | 14.6
   error_ADD3         | 0.0583   | +L       | 14.7
                      |          |          |
   **Fixed effects**  |          |          |
                      |          |          |
   initEpo            | 463.5    | +L       | 578.1
   initEpoRrel        | 0.0392   | +L       | 564
   kde                | 0.01233  | +L       | 4.646
   kdi                | 0.002077 | +L       | 16.42
   ke                 | 0.05406  | +L       | 3.536
   kex                | 0.001498 | +L       | 37.35
   koff               | 0.3055   | +L       | 28
   kon                | 0.003433 | +L       | 578.2
   kt                 | 49.23    | +L       | 685.1
   offset             | 1e-05    | -        | --
   scale              | 0.98     | -        | --

--- No local parameters defined ----------------------

Values rounded to 4 significant digits.
Estimated/fixed parameters (+/-)
Estimation on (N) natural / (L) log / (G) logit scale.
```

```
R> # Simulate for the selected optimum and plot
R> optsys <- sim_IQRsysModel(optsys, simtime = 1:300)
R> plot_IQRsysModel(optsys)
```



IntiQuan

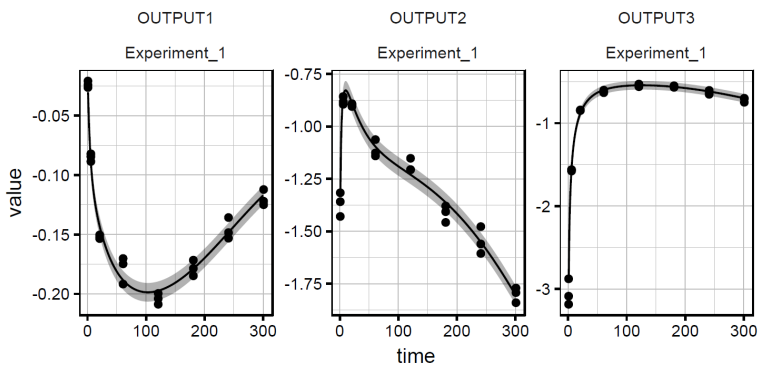
switchOpt_IQRsysModel() LETS YOU TAKE A CLOSE LOOK AT LOCAL OPTIMA




```
R> # Produce waterfall plot
R> plotWaterfall_IQRsysModel(optsys)

R> # Switch to another local optimum
R> optsys <- switchOpt_IQRsysModel(optsys, optimum = 1)

R> # Print parameters of the selected optimum
R> getPars_IQRsysModel(optsys)
```

```
--- Selected optimum: 1 ------------------------------

--- Global parameters --------------------------------

   Parameter          | Value    | Estimate | RSE (%)
   ------------------------------------------------
                      |          |          |
   **Error model**    |          |          |
                      |          |          |
   error_ADD1         | 0.007795 | +L       | 14.68
   error_ADD2         | 0.04428  | +L       | 15.01
   error_ADD3         | 0.0513   | +L       | 14.62
                      |          |          |
   **Fixed effects**  |          |          |
                      |          |          |
   initEpo            | 143.9    | +L       | 500
   initEpoRrel        | 0.3691   | +L       | 7.846
   kde                | 0.01201  | +L       | 4.636
   kdi                | 0.00129  | +L       | 25.69
   ke                 | 0.05556  | +L       | 3.684
   kex                | 0.000613 | +L       | 95.88
   koff               | 0.08085  | +L       | 20.38
   kon                | 0.001048 | +L       | 500.2
   kt                 | 0.01604  | +L       | 7.623
   offset             | 1e-05    | -        | --
   scale              | 0.98     | -        | --

--- No local parameters defined ----------------------

Values rounded to 4 significant digits.
Estimated/fixed parameters (+/-)
Estimation on (N) natural / (L) log / (G) logit scale.
```

```
R> # Simulate for the selected optimum and plot
R> optsys <- sim_IQRsysModel(optsys, simtime = 1:300)
R> plot_IQRsysModel(optsys)
```

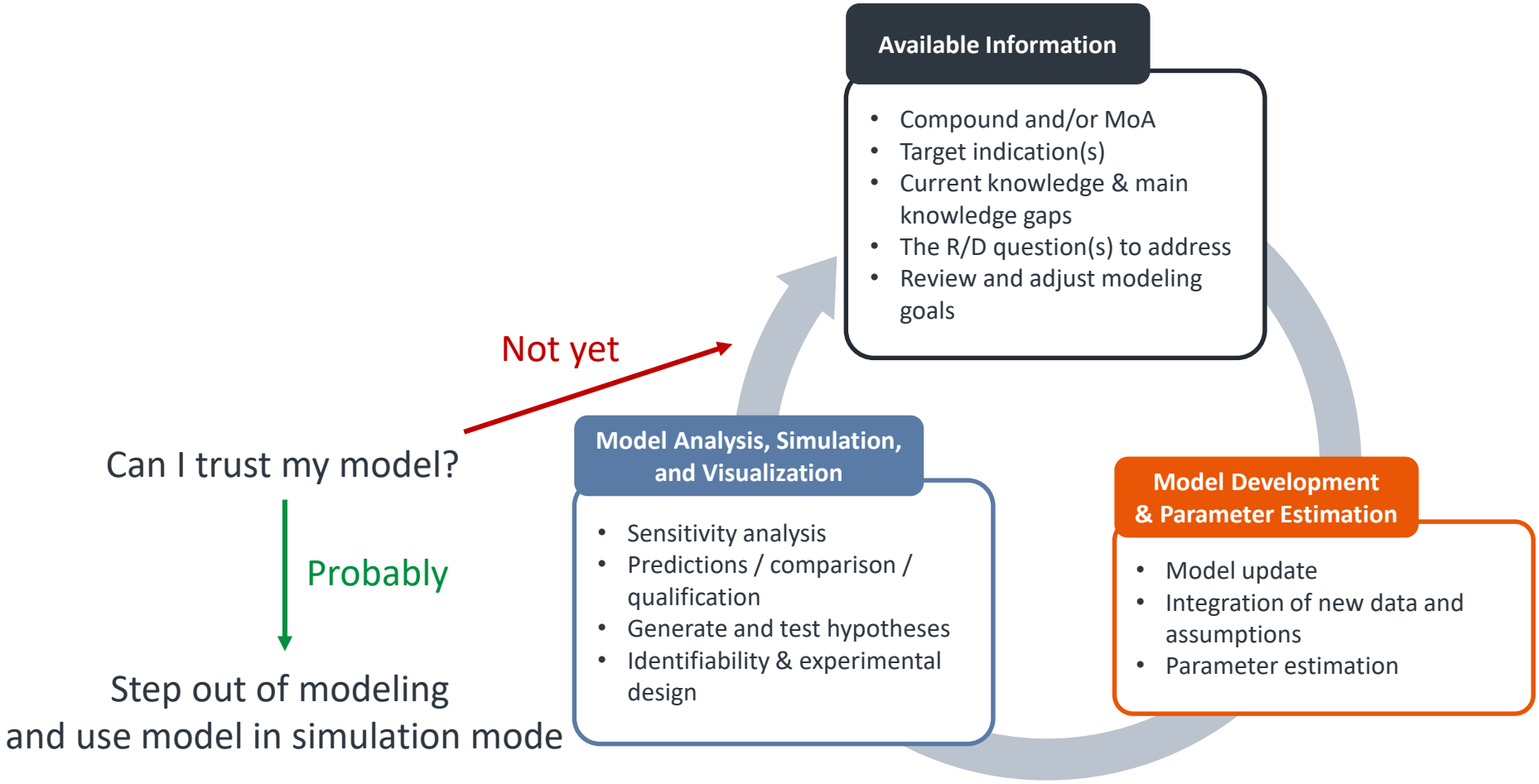


IntiQuan

Coffee Break

CAN I TRUST MY MODEL?
INFORMING NEEDED EXPERIMENTS

ITERATIVE PROCESS WITH THE AIM TO OBTAIN A FIT-FOR-PURPOSE MODEL

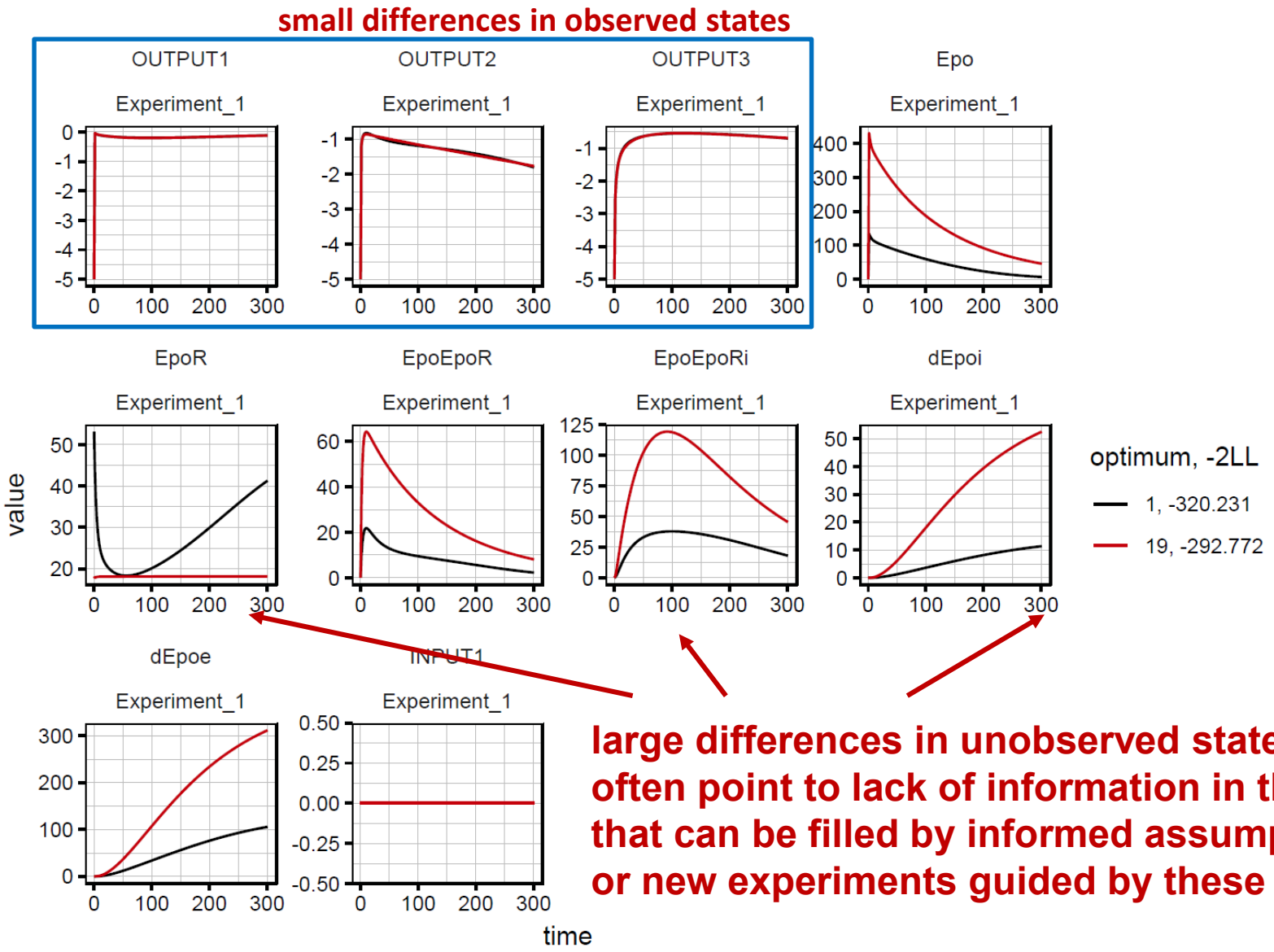


How would you define "Trust" in a model?

IntiQuan

INFORMATION IN LOCAL MINIMA

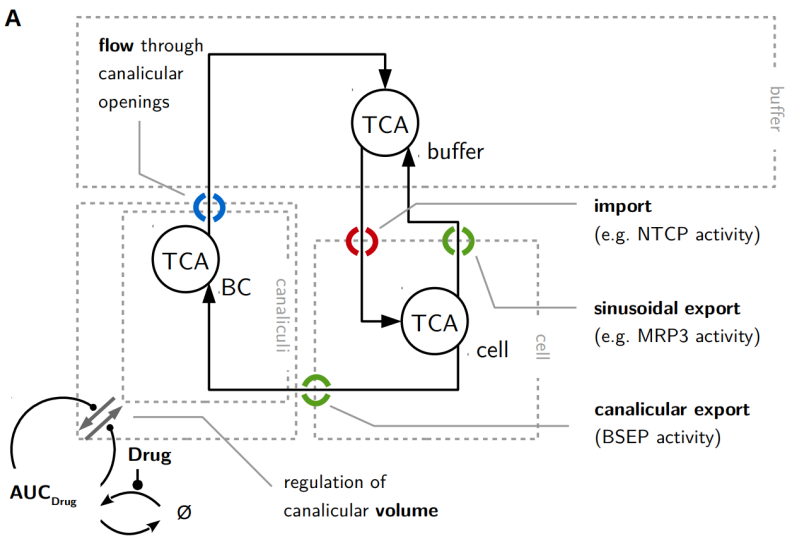
DIFFERENT LOCAL MINIMA CAN REPRESENT DIFFERENT MECHANISTIC HYPOTHESES BASED ON OBJECTIVE FUNCTION A DISCRIMINATION MIGHT BE IMPOSSIBLE

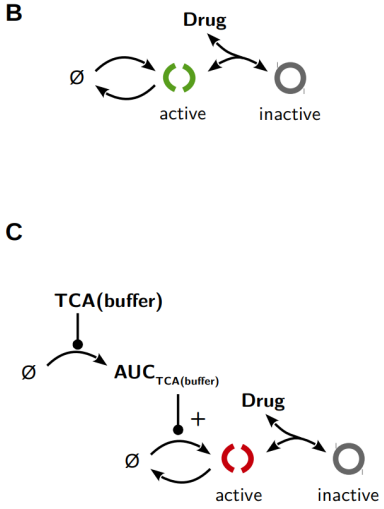


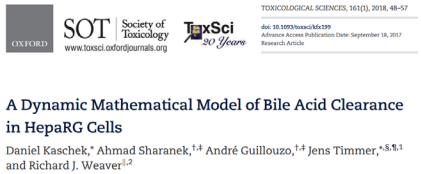
large differences in unobserved states often point to lack of information in the model that can be filled by informed assumptions or new experiments guided by these results

IntiQuan

ANOTHER EXAMPLE MODEL



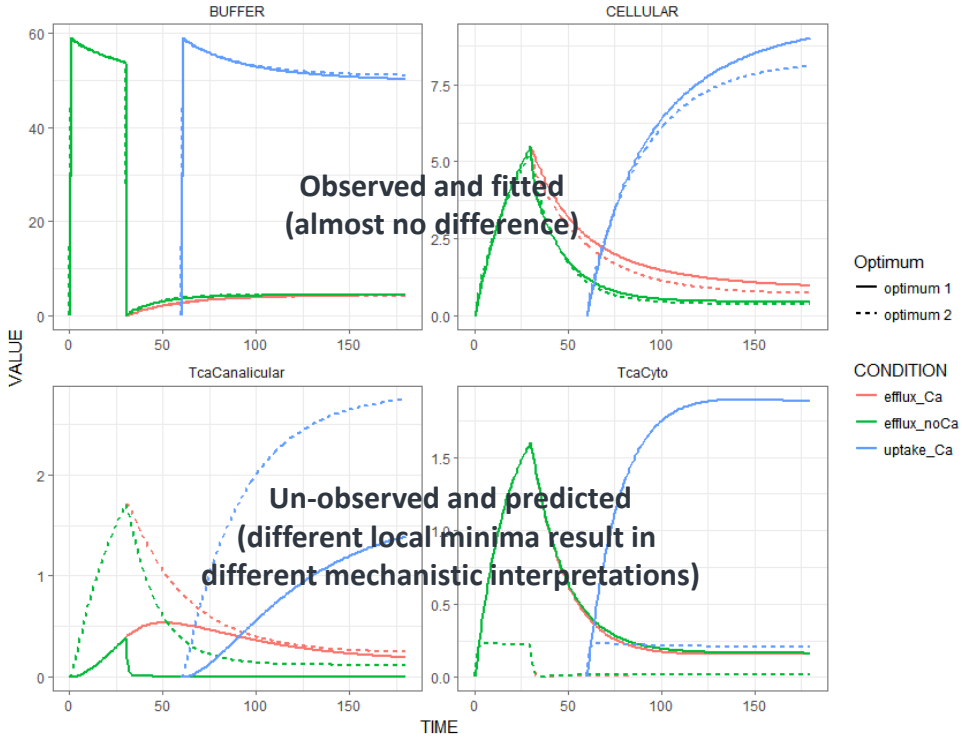




- **A: Bile acid (here TCA)** is transported between buffer, cytoplasm of liver cells and canaliculi
- **B, C:** Interaction between drug and transporter modeled by reaction network

IntiQuan

MAIN LOCAL MINIMA RESULT IN DIFFERENT INTERPRETATION, REQUIRING ADDITIONAL EXPERIMENTS TO ENSURE MECHANISM IDENTIFICATION



Observables

- BUFFER and CELLULAR very similar

Optimum 1:

- TCA mainly taken up in cytoplasm

Optimum 2:

- TCA mainly taken up in canaliculi

Previously available experiments: uptake of TCA

New defined experiments: efflux of TCA through buffer replacement

Result: Single optimum remaining, suggesting uptake in cytoplasm

IntiQuan

UNCERTAINTY ANALYSIS

Example1_Epo/Scripts/SCRIPT_04_uncertainty.R

IQR TOOLS PROVIDES CONFIDENCE INTERVALS

Parameter	Value	Estimate	RSE (%)
Error model			
error_ADD1	0.007799	+L	14.69
error_ADD2	0.0443	+L	15.02
error_ADD3	0.05132	+L	14.63

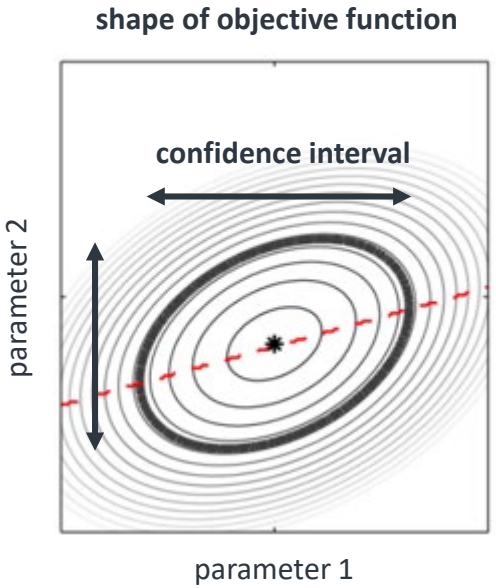
Why to look at confidence intervals:

- precision of your parameter estimates
- model can be trusted

Source of parameter uncertainty:

- data is inherently variable/noisy

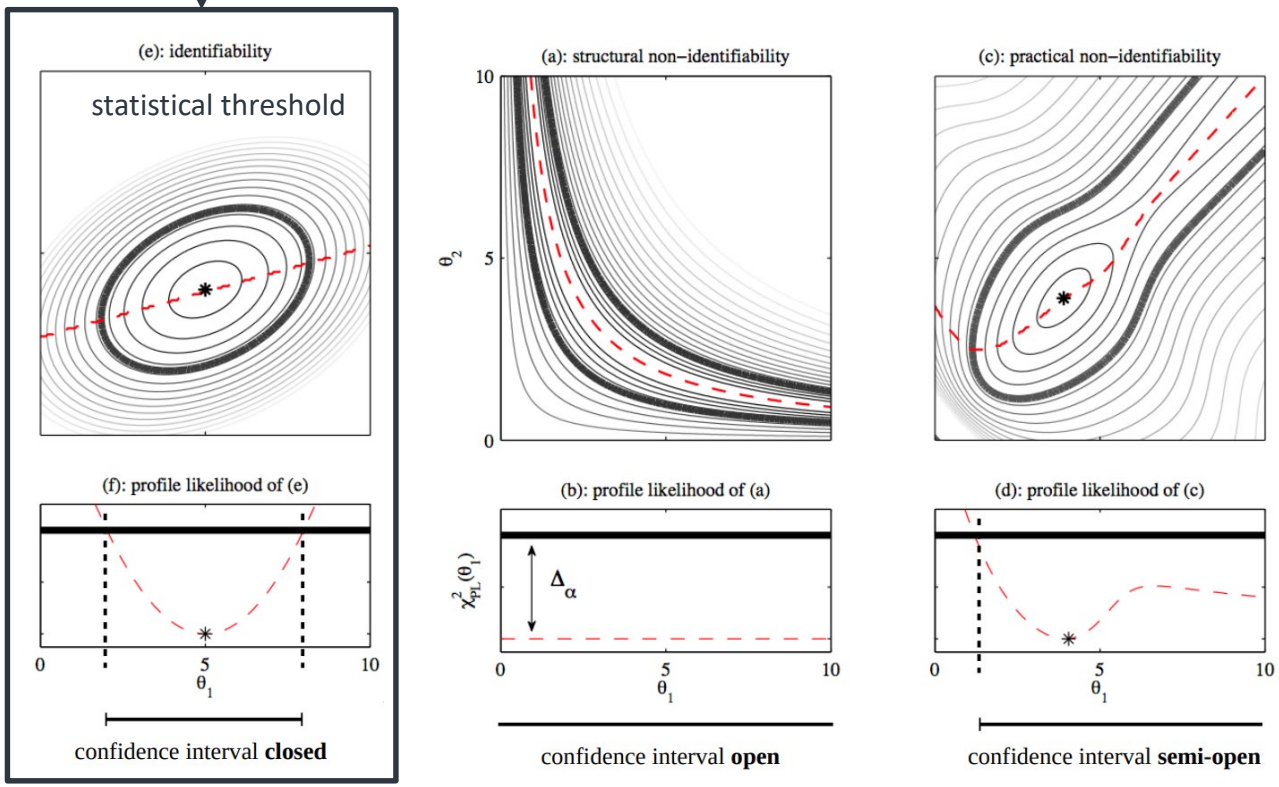
Shape of the objective function contains the information about the confidence intervals



IntiQuan

SHAPE OF THE LOG-LIKELIHOOD DETERMINES CONFIDENCE INTERVALS

- **Asymptotic**: quadratic around optimum
- **Not asymptotic:** non-trivial shape

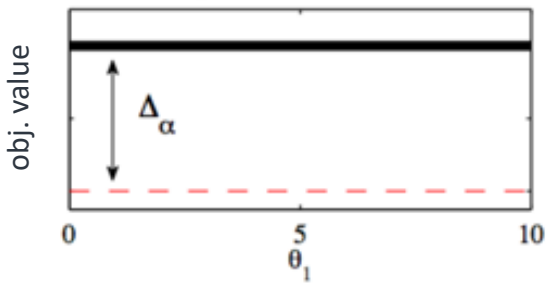


Raue, Andreas, et al. "Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood." *Bioinformatics* 25.15 (2009): 1923-1929.

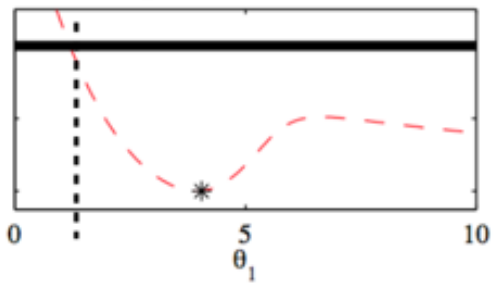
IntiQuan

PROFILE LIKELIHOOD ALLOWS DETECTION OF IDENTIFIABILITY ISSUES

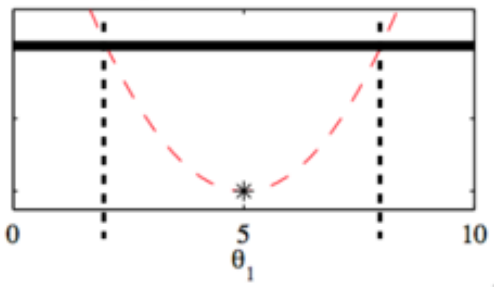
- Likelihood contour plots only for two parameters
- Profile likelihood available for unlimited number of parameters



Structural non-identifiability
(maximal uncertainty)



Practical non-identifiability
(finite confidence interval only for low conf. level)



Identifiable
(finite confidence interval)

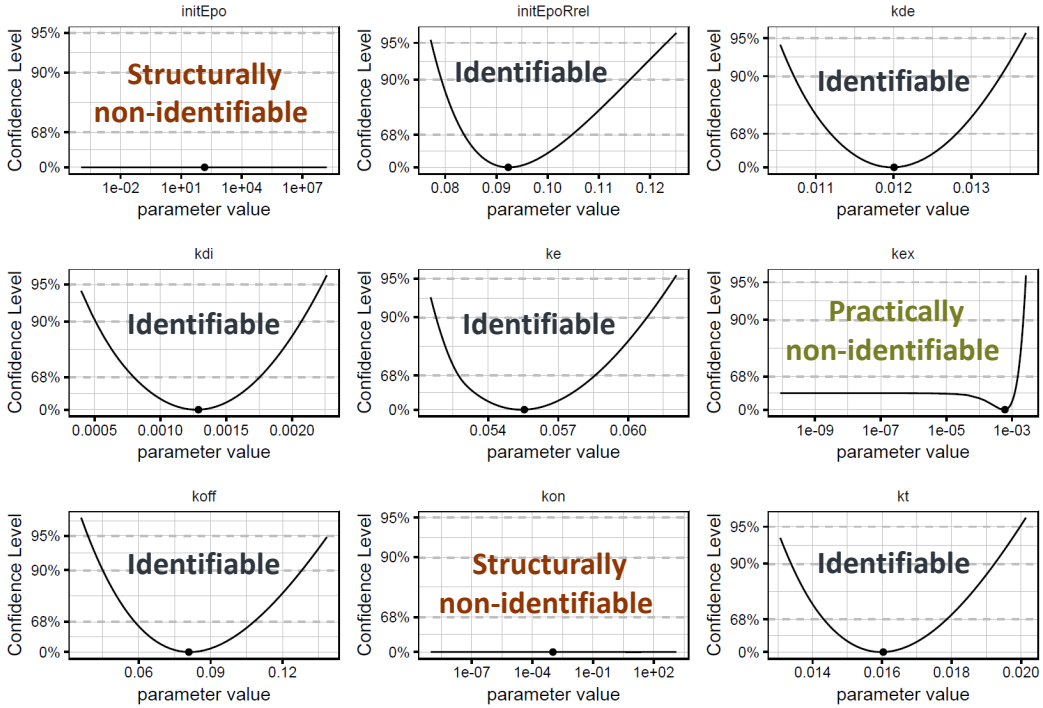
IntiQuan

profile_IQRsysModel() ADDS PROFILE LIKELIHOOD TO OPTIMIZED SYSMODEL

```
R> optsys <- profile_IQRsysModel(optsys,
                                 ncores = 8)
```

Use 8 cores in parallel (one parameter profile runs on one core)

```
R> plotProfile_IQRsysModel(optsys)
R> getPars(optsys)
```



IntiQuan

DESIGN OF NEW EXPERIMENTS OR INCORPORATION OF INFORMED ASSUMPTIONS BASED ON PROFILE LIKELIHOOD EVALUATION

`Example1_Epo/Scripts/SCRIPT_04_uncertainty.R`

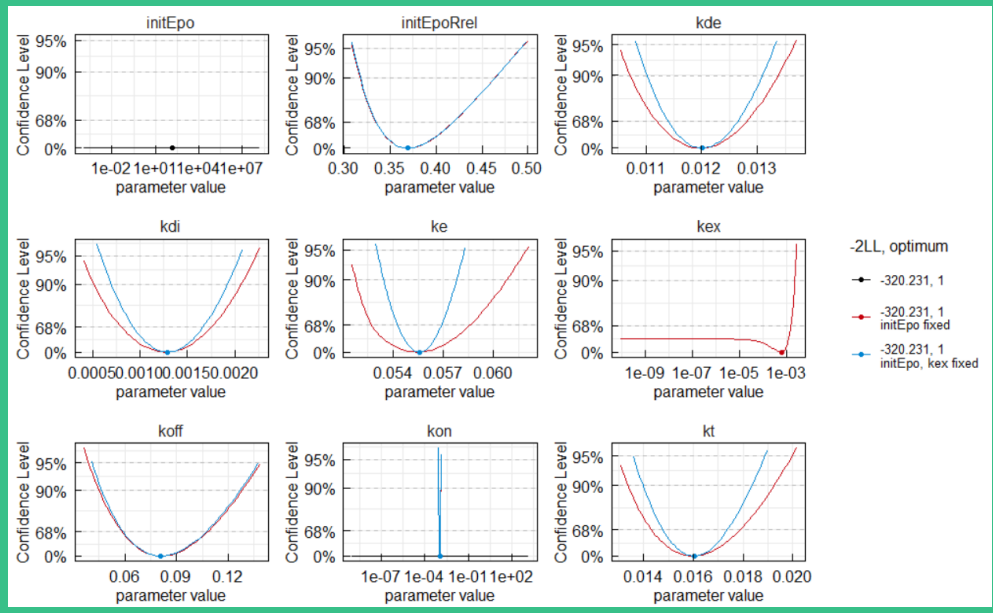
Exercise:

1. Assume initEpo (and kex) to be correct as estimated - how does it impact the parametric uncertainty?

```
R> optsys <- profile_IQRsysModel(optsys, ncores = 8,
                                 fixed = c("initEpo", "kex"))
R> plotProfile_IQRsysModel(optsys)
```

initEpo and kex fixed to last estimated values

=> all other parameters become identifiable



IntiQuan

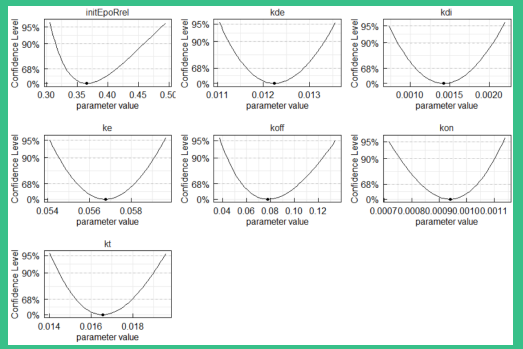
DESIGN OF NEW EXPERIMENTS OR INCORPORATION OF INFORMED ASSUMPTIONS BASED ON PROFILE LIKELIHOOD EVALUATION

Example1_Epo/Scripts/SCRIPT_04_uncertainty.R

Exercise:

2. Assume initEpo measured in experiment and kex assumed to be known (*agreed by all team members*)
   => New values need to be set in estimation object and fixed
   => New estimation needs to be performed
   => Profile LL calculated again

```
# ------------------------------------------------------------------------------#
# initEpo: lets assume it was subsequently measured to be 160
# kex:     lets assume the whole team agreed that kex=0.001 is the correct value
# ------------------------------------------------------------------------------#

# Convert previous model into an estimation object
# and set parameters according to new information - fix them and do not estimate them
newest <- as_IQRsysEst(
  sysModel = optsys,
  modelSpec = modelSpec_IQRest(
    POPvalues0  = c(kex = 0.001, initEpo=160),
    POPestimate = c(kex = 0,     initEpo=0)
  )
)

newest

# Generate and run the parameter estimation
proj <- IQRsysProject(newest, projectPath = "../Models/RUN3_kexfixed", opt.nfits = 24)
optsys2 <- run_IQRsysProject(proj)

# Generate new profiles
optsys2 <- profile_IQRsysModel(optsys2, ncores = 8)
plotProfile_IQRsysModel(optsys2)
```



IntiQuan

IT'S YOUR TURN - HANDS ON

`Excercise1_Epo/Scripts/SCRIPT_01.R`

Solution in: `Solution1_Epo/Scripts/SCRIPT_01.R`
(one of all possible solutions)

Alternatives:

1. **Exercise2_Phototransduction (solution provided as well)**
   Some slides with project background in the Exercise folder

2. **Or take one of your own models and implement it in IQR Tools**

EXERCISE 1 – EXPLORE DIFFERENT BINDING AFFINITIES

Problem:

New experimental data will be provided that contains Epo measurements for a different Epo preparation. Based on the current fitted model, how would Epo time courses look like for higher/lower Epo binding affinity?

Instructions:

- Simulate and plot the "model.txt" for the estimated parameter values, changing "kon"

or

- Define an `IQRsysModel` with conditions "Epo_Low", "Epo_Control", and "Epo_High" and set different "kon" values for each condition

EXERCISE 2 – TUNING BY HAND

Problem:

The new experimental data has arrived **("../Data/dataRAW_TwoPreps.csv")**. Can you guess the binding affinity from the some of the data?

Instructions:

- Import the new raw data as `IQRsysData` object and write to file "../Data/dataSYS_TwoPreps.csv"
- Create an `IQRsysModel` including the new data and condition-specific "kon"
- Plot only the data and try to guess the correct "kon" value for the new experiment
- Plot prediction and data for your guess

EXERCISE 3 – JOINT PARAMETER ESTIMATION

Problem:

Estimate all model parameters (except for "initEpo") and condition-specific "kon" values from the new data set. Choose a relative error model for OUTPUT1 and an absolute-proportional error model for OUTPUT2 and OUTPUT3.

Instructions:

- Create an IQRsysModel with the appropriate modelSpec and error model
- Run 36 fits and compute the profile likelihood
- Have a look at the goodness of fit plots
- Compare the profile likelihood between the old data (one condition) and new data (two conditions) side-by-side.

EXERCISE 4 – EXPLORATION OF LOCAL OPTIMA

Problem:

Parameter estimation reveals the existence of local optima. What is the difference between the best and second best optimum?

Instructions:

- Identify the parameters with the largest differences between first and second optimum based on `plotPars_IQRsysModel()`
- Compute the profile likelihood for these parameters for both optima (use `switchOpt_IQRsysModel()`) and see what happens when you use `plotProfile_IQRsysModel()`
- Create plots with `plot_IQRsysModel()` for both local optima

SPECIAL WORKFLOW TOPICS

REUSE OF MODELS FROM MODEL DATABASES IMPORT OF SBML MODELS

`Example3_SBML/Scripts/SCRIPT_01_SBML2R.R`

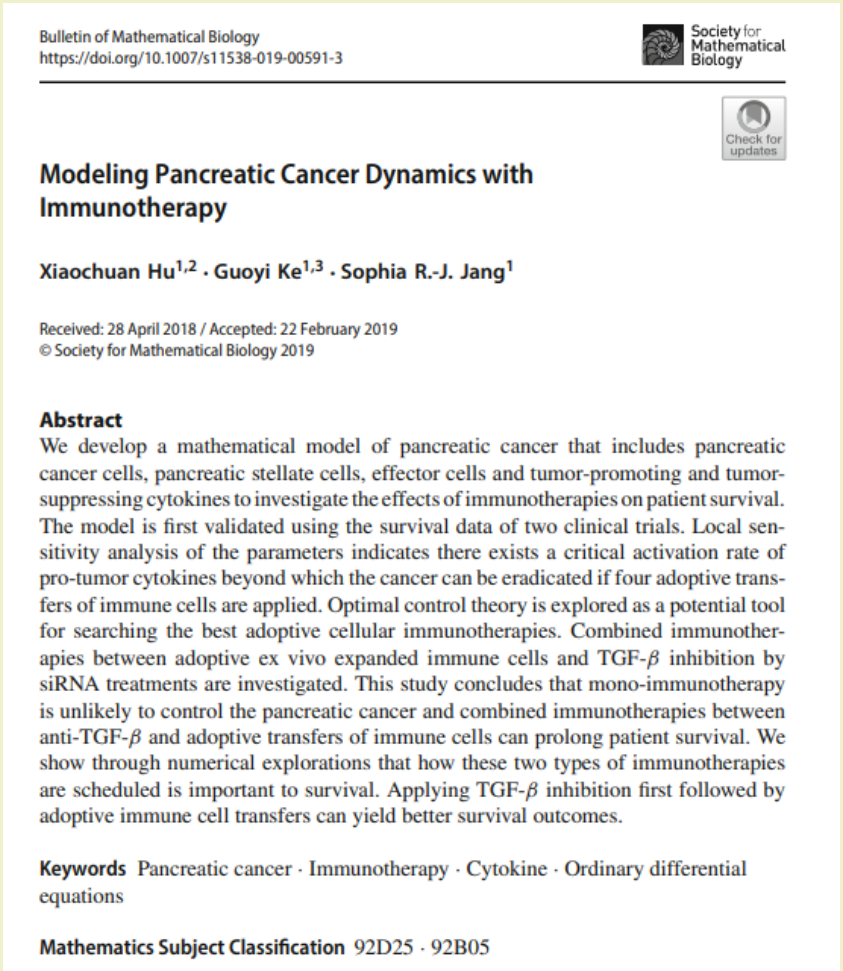
IMPORTING AN SBML MODEL TO IQR TOOLS

- **Limitation: supported on Windows only**
- **Requires installed IQRsbml** (https://www.intiquan.com/iqrsbml/)
- **Handles SBML Level 1 and 2 (Version 1-4)**

SELECTING 2 MODELS FROM THE BIOMODELS.NET DATABASE

https://link.springer.com/article/10.1007%2Fs11538-019-00591-3

https://www.ebi.ac.uk/biomodels/MODEL1112110004



Bulletin of Mathematical Biology
https://doi.org/10.1007/s11538-019-00591-3

Society for Mathematical Biology

Check for updates

Modeling Pancreatic Cancer Dynamics with Immunotherapy

Xiaochuan Hu[1,2] · Guoyi Ke[1,3] · Sophia R.-J. Jang[1]

Received: 28 April 2018 / Accepted: 22 February 2019

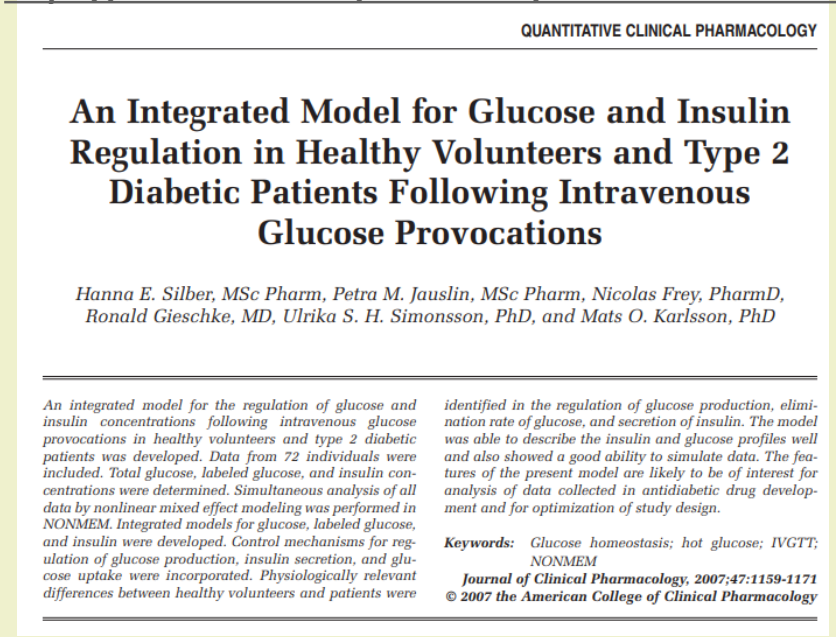
Abstract
We develop a mathematical model of pancreatic cancer that includes pancreatic cancer cells, pancreatic stellate cells, effector cells and tumor-promoting and tumor-suppressing cytokines to investigate the effects of immunotherapies on patient survival. The model is first validated using the survival data of two clinical trials. Local sensitivity analysis of the parameters indicates there exists a critical activation rate of pro-tumor cytokines beyond which the cancer can be eradicated if four adoptive transfers of immune cells are applied. Optimal control theory is explored as a potential tool for searching the best adoptive cellular immunotherapies. Combined immunotherapies between adoptive ex vivo expanded immune cells and TGF-β inhibition by siRNA treatments are investigated. This study concludes that mono-immunotherapy is unlikely to control the pancreatic cancer and combined immunotherapies between anti-TGF-β and adoptive transfers of immune cells can prolong patient survival. We show through numerical explorations that how these two types of immunotherapies are scheduled is important to survival. Applying TGF-β inhibition first followed by adoptive immune cell transfers can yield better survival outcomes.

Keywords Pancreatic cancer · Immunotherapy · Cytokine · Ordinary differential equations

Mathematics Subject Classification 92D25 · 92B05

https://accp1.onlinelibrary.wiley.com/doi/abs/10.1177/0091270007304457

https://www.ebi.ac.uk/biomodels/BIOMD0000000792



QUANTITATIVE CLINICAL PHARMACOLOGY

An Integrated Model for Glucose and Insulin Regulation in Healthy Volunteers and Type 2 Diabetic Patients Following Intravenous Glucose Provocations

Hanna E. Silber, MSc Pharm, Petra M. Jauslin, MSc Pharm, Nicolas Frey, PharmD, Ronald Gieschke, MD, Ulrika S. H. Simonsson, PhD, and Mats O. Karlsson, PhD

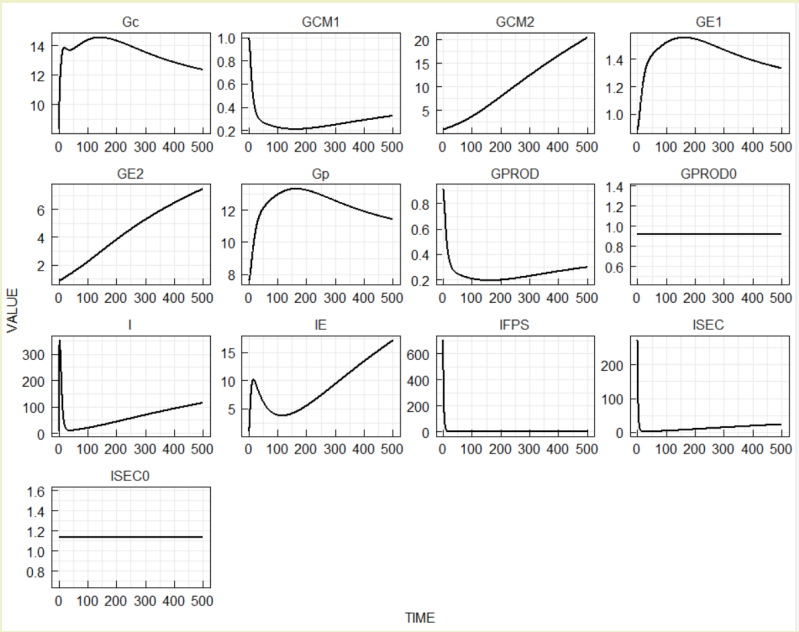
An integrated model for the regulation of glucose and insulin concentrations following intravenous glucose provocations in healthy volunteers and type 2 diabetic patients was developed. Data from 72 individuals were included. Total glucose, labeled glucose, and insulin concentrations were determined. Simultaneous analysis of all data by nonlinear mixed effect modeling was performed in NONMEM. Integrated models for glucose, labeled glucose, and insulin were developed. Control mechanisms for regulation of glucose production, insulin secretion, and glucose uptake were incorporated. Physiologically relevant differences between healthy volunteers and patients were identified in the regulation of glucose production, elimination rate of glucose, and secretion of insulin. The model was able to describe the insulin and glucose profiles well and also showed a good ability to simulate data. The features of the present model are likely to be of interest for analysis of data collected in antidiabetic drug development and for optimization of study design.

Keywords: *Glucose homeostasis; hot glucose; IVGTT; NONMEM*
Journal of Clinical Pharmacology, 2007;47:1159-1171

IntiQuan

EXAMPLE3_SBML/SCRIPTS/SCRIPT_01_SBML2R.R

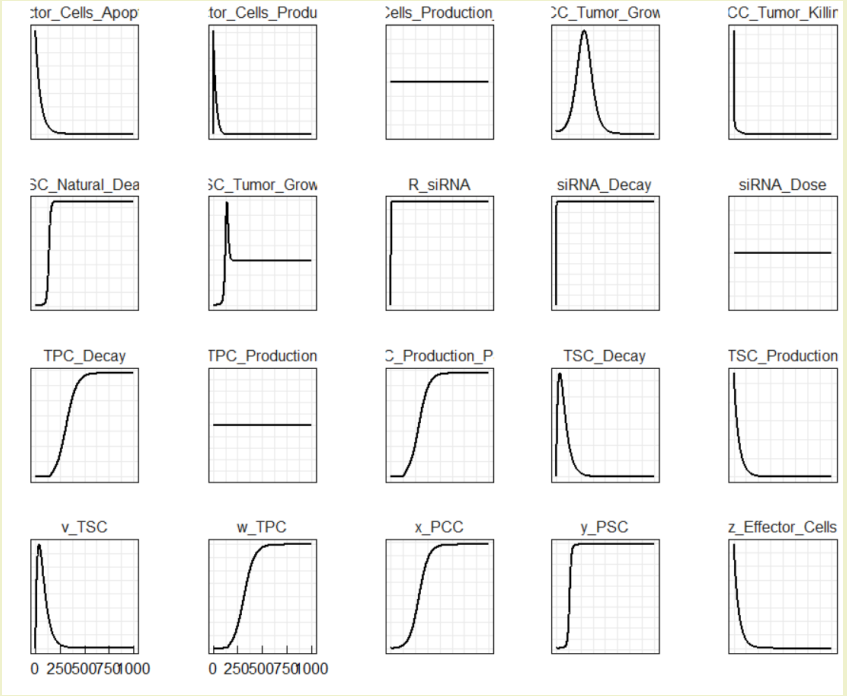
```
##############################################
# First model - Silber2007_IntravenousGlucose_IntegratedGlucoseInsulinModel
##############################################

# Download SBML model from biomodels.net
URL <-
"https://wwwdev.ebi.ac.uk/biomodels/model/download/MODEL1112110004.2?filename=MOD
EL1112110004_url.xml"
download.file(URL,"silber.xml")

# Looking at the SBML file
file.edit("silber.xml")

# Import as IQRmodel
model <- IQRmodel("silber.xml",FLAGsym=FALSE)

# Simulate and plot results
plot(sim_IQRmodel(model,simtime=500))

# Exporting IQRmodel as ODE based text file
export_IQRmodel(model,"silber.txt")

# Looking at the exported file
file.edit("silber.txt")
```



IntiQuan

EXAMPLE3_SBML/SCRIPTS/SCRIPT_01_SBML2R.R

```
##################################################
# Second model - Hu2019 - Modeling Pancreatic Cancer Dynamics with Immunotherapy
##################################################

# Download SBML model from biomodels.net
URL <-
"https://www.ebi.ac.uk/biomodels/model/download/BIOMD0000000792.2?filename=Hu2019
.xml"
download.file(URL,"HU2019.xml")

# Looking at the SBML file
file.edit("HU2019.xml")

# Import as IQRmodel
model <- IQRmodel("HU2019.xml",FLAGsym=FALSE)

# Simulate and plot results
plot(sim_IQRmodel(model,simtime=1000))

# Exporting IQRmodel as ODE based text file
export_IQRmodel(model,"HU2019.txt")

# Looking at the exported file
file.edit("HU2019.txt")
```



IntiQuan

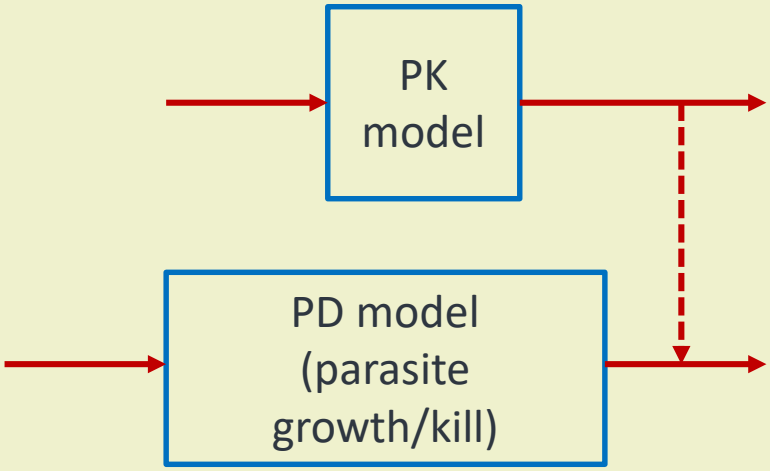
HANDS-ON EXCERCISE

- Select another SBML model and import it to R
- Simulate it
- Plot results

QSP GOING NLME

Example2_Malaria/Scripts/SCRIPT_01_estimation.R

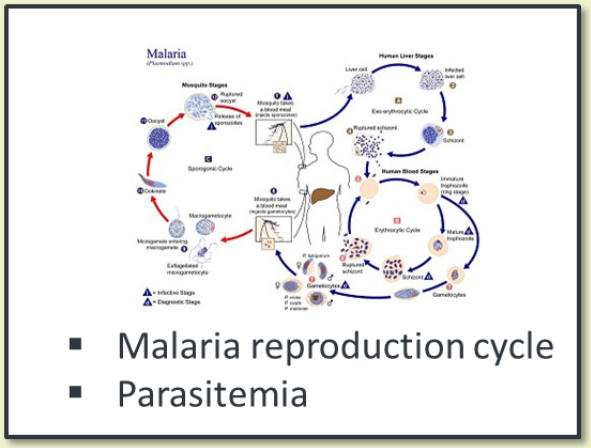
BACKGROUND



Step 1: Population PK modeling used to establish individual subject PK parameters

Step 2: Using individual PK parameters for simulation of PK, estimating the PD parameters

- Different tested PD models
- Not really suitable for modeling with NONMEM nor MONOLIX
- Still variability needs to be taken into account



IntiQuan

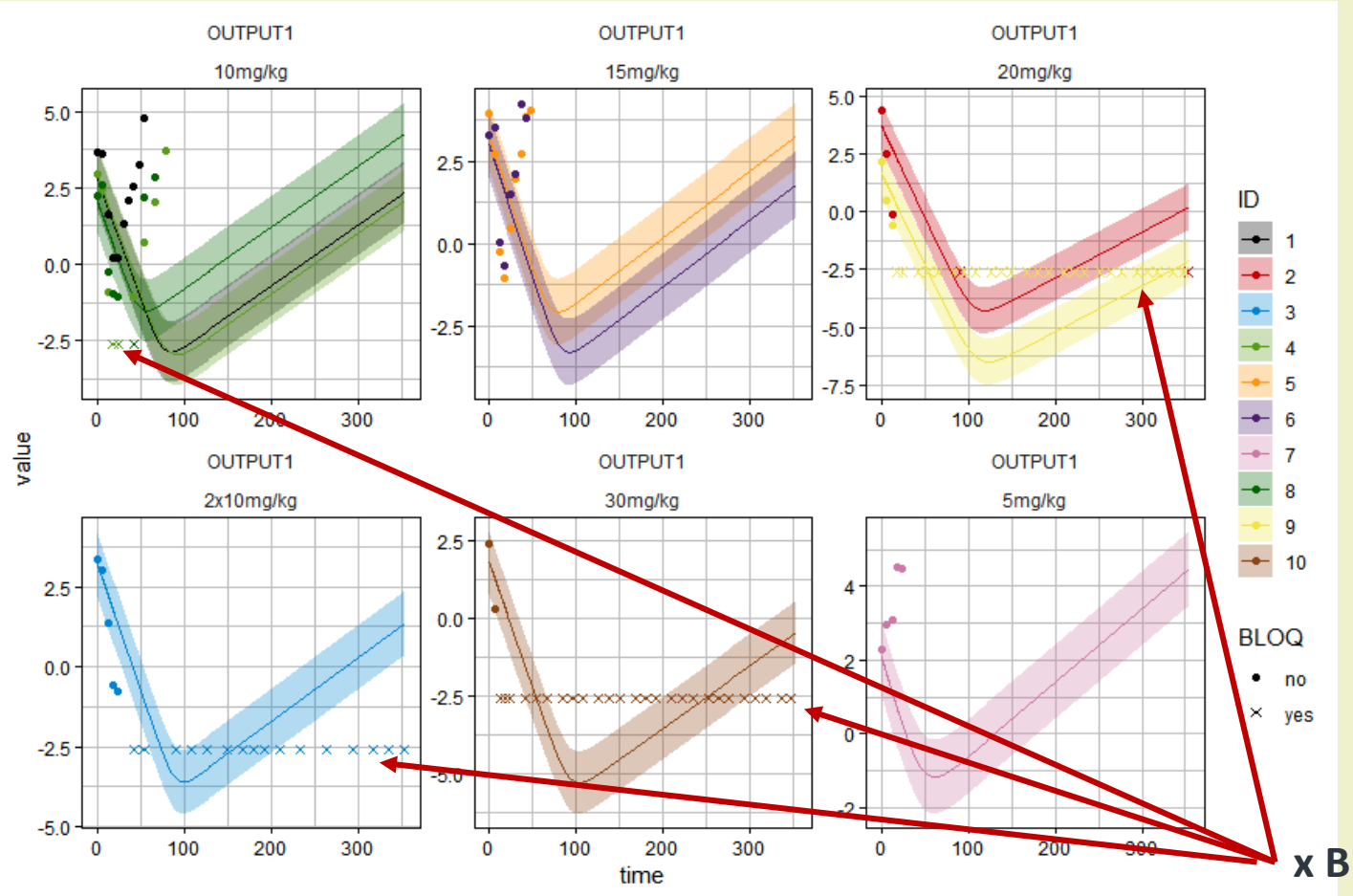
EXAMPLE: MALARIA PD MODEL ESTIMATION BASED ON MONOLIX POPULATION PK MODEL

- Malaria PD model
- Data definition with individual PK parameters as regressors obtained from NLME PK model
- Model specification

```
sysobj <- IQRsysModel(

  # Model
  model = "Resources/model.txt",

  # Data
  data  = list(
    datafile = "../Data/data.csv",
    regressorNames = c("Fabs1", "ka", "CL", "Vc",
                       "Q1", "Vp1", "Tlag1", "PLbase")
  ),

  # Specs
  modelSpec = list(
     POPvalues0      = c(GR = 0.02 , hill = 4,
                         EMAX = 0.1, EC50 = 0.01),
     POPestimate     = c(GR = 1    , hill = 1,
                         EMAX = 1  , EC50 = 1),
     errorModel = list(
       OUTPUT1 = c(type = "abs", guess = 1)
     )
  )
)
```

CHECKING THE INITIAL GUESS

```
plot(sysobj)
```

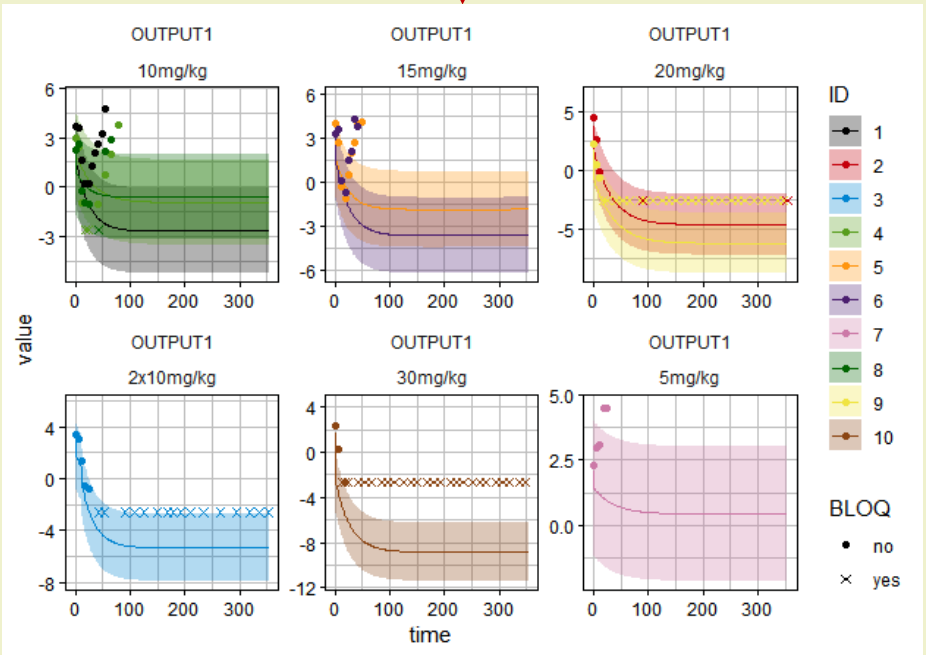


x BLOQ observations

IntiQuan

RUN POOLED ESTIMATION (GLOBAL PD PARAMETERS)

```
# Estimate population parameters
est <- as_IQRsysEst(sysobj)
proj <- IQRsysProject(est,
                      projectPath = "../Models/01_NoIIV",
                      opt.nfits = 24,
                      opt.sd = 2,
                      opt.parupper = c(hill = 10),
                      opt.parlower = c(hill = 1))

optsys <- run_IQRsysProject(proj, ncores = 8, FLAGgof = FALSE)

# Plot best fit
plot(optsys)
```

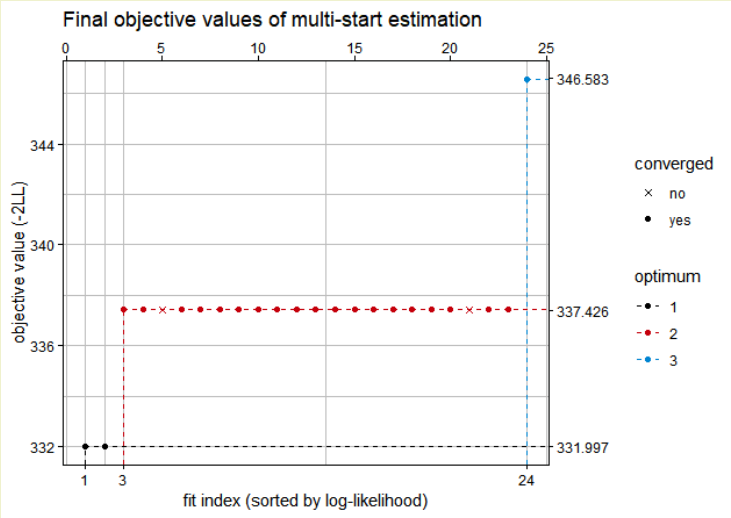


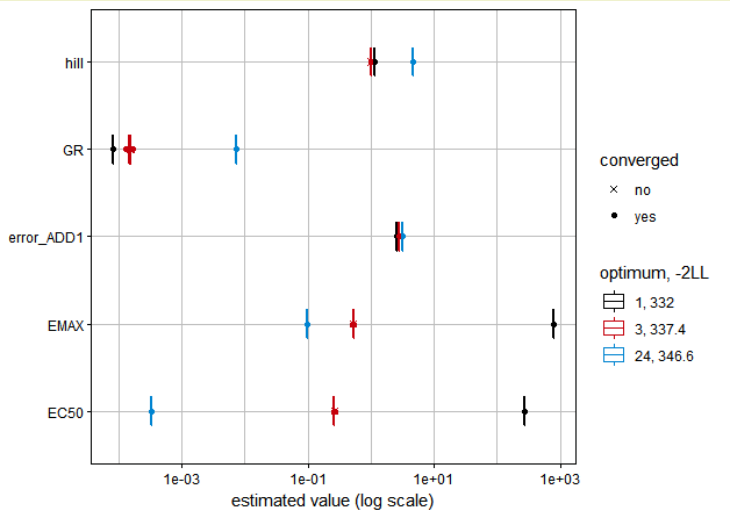
Bad fit? What about local optima?

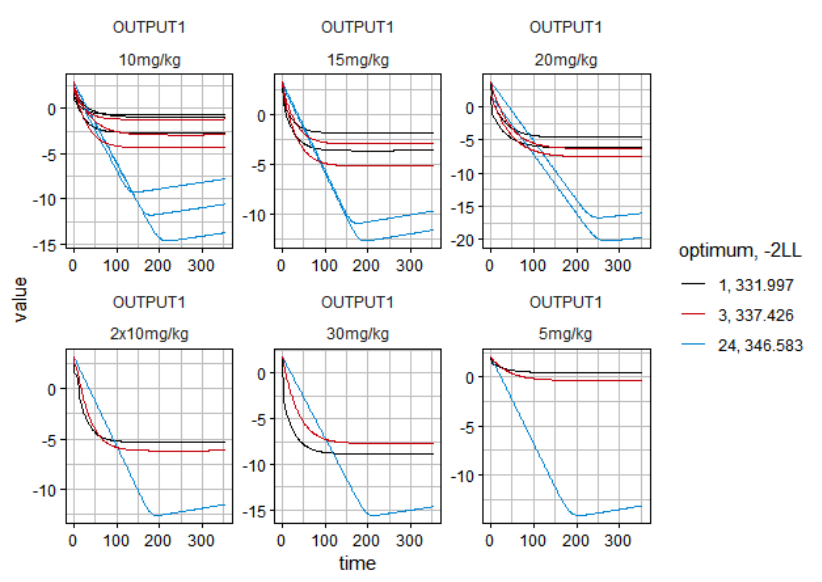
```
plotWaterfall_IQRsysModel(optsys)
plotPars_IQRsysModel(optsys)
plotPred_IQRsysModel(optsys,
                     states = "OUTPUT1")
```


IntiQuan

NONE OF THE LOCAL OPTIMA DESCRIBES THE DATA





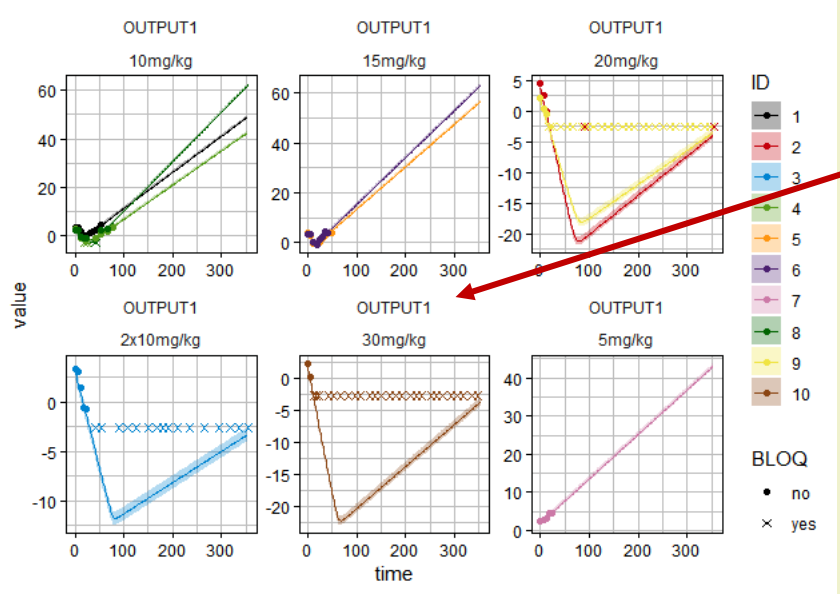


The data cannot be consistently described by the PD model with population parameters alone.

IntiQuan

ESTIMATE WITH INTER-INDIVIDUAL VARIABILITY

```
# Estimate population parameters + IIV
est <- as_IQRsysEst(sysobj,
                    modelSpec = list(
                      IIVvalues0      = c(GR = 0.5, EMAX = 0.5, EC50 = 0.5),
                      IIVestimate     = c(GR = 2  , EMAX = 2  , EC50 = 2  )
                    ))

proj <- IQRsysProject(est,
                      projectPath = "../Models/02_IIV",
                      opt.nfits = 24,
                      opt.sd = 2,
                      opt.parupper = c(hill = 10),
                      opt.parlower = c(hill = 1))

optsys <- run_IQRsysProject(proj, ncores = 8, FLAGgof = TRUE)

plot(optsys)
```



IQR Tools QSP parameter estimation allows to consider random effects / inter-individual variability on parameters

Leading to an adequate description of observed data

IntiQuan

WHY WOULD YOU PREFER IQR QSP TOOLS OVER MONOLIX/NONMEM

- **Models under development frequently have non-identifiable parameters**
  - NONMEM with FOCE(I) methods have difficulties to converge
  - SAEM method converge but final objective values will show stochastic fluctuations
  - SAEM method "requires" random effects on all parameters

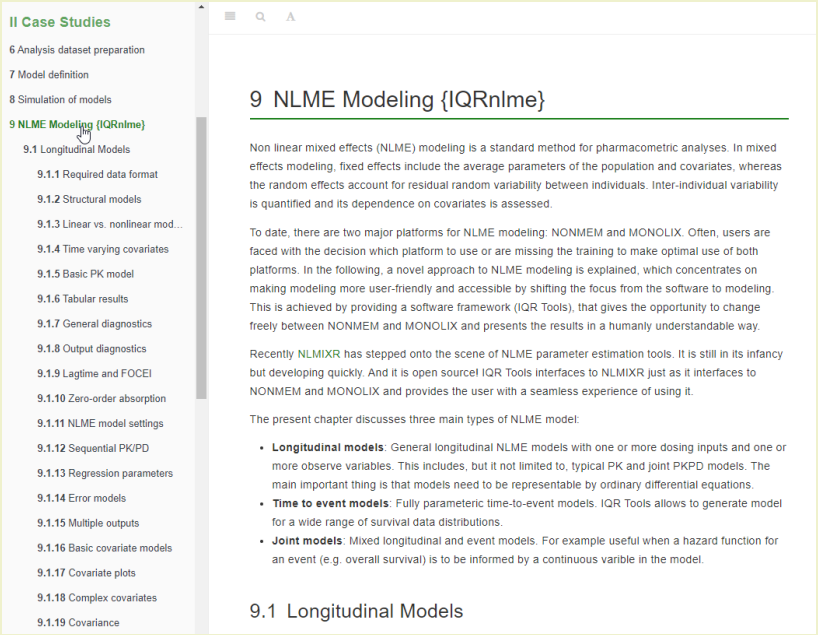
  - IQR Tools **robustly converges** and returns the **precise objective value**

- **IQR Tools provides structural insights via the profile likelihood method**
  - traditional profile likelihood: 1 point of the profile ~ 1 fit
    => Computationally badly tractable
  - our fast profile likelihood: 1 profile ~ 1 fit
    => Computationally reasonable

- **IQR Tools "runs 30 fits in the time Monolix runs 1 fit"**
  - faster execution time allows exploration of parameter space / mechanisms

FULL NLME

IQR TOOLS ALLOWS FULL NLME MODELING THROUGH SEAMLESS INTERFACES TO NONMEM AND MONOLIX



https://iqrtools.intiquan.com/doc/book/nlme-modeling-iqrnlme.html

- Separate installation of NONMEM or MONOLIX required
- Same dataset format as for QSP modeling
- Same model syntax, etc.

IntiQuan

EVALUATION OF HYBRID MODELS (NLME+QSP)

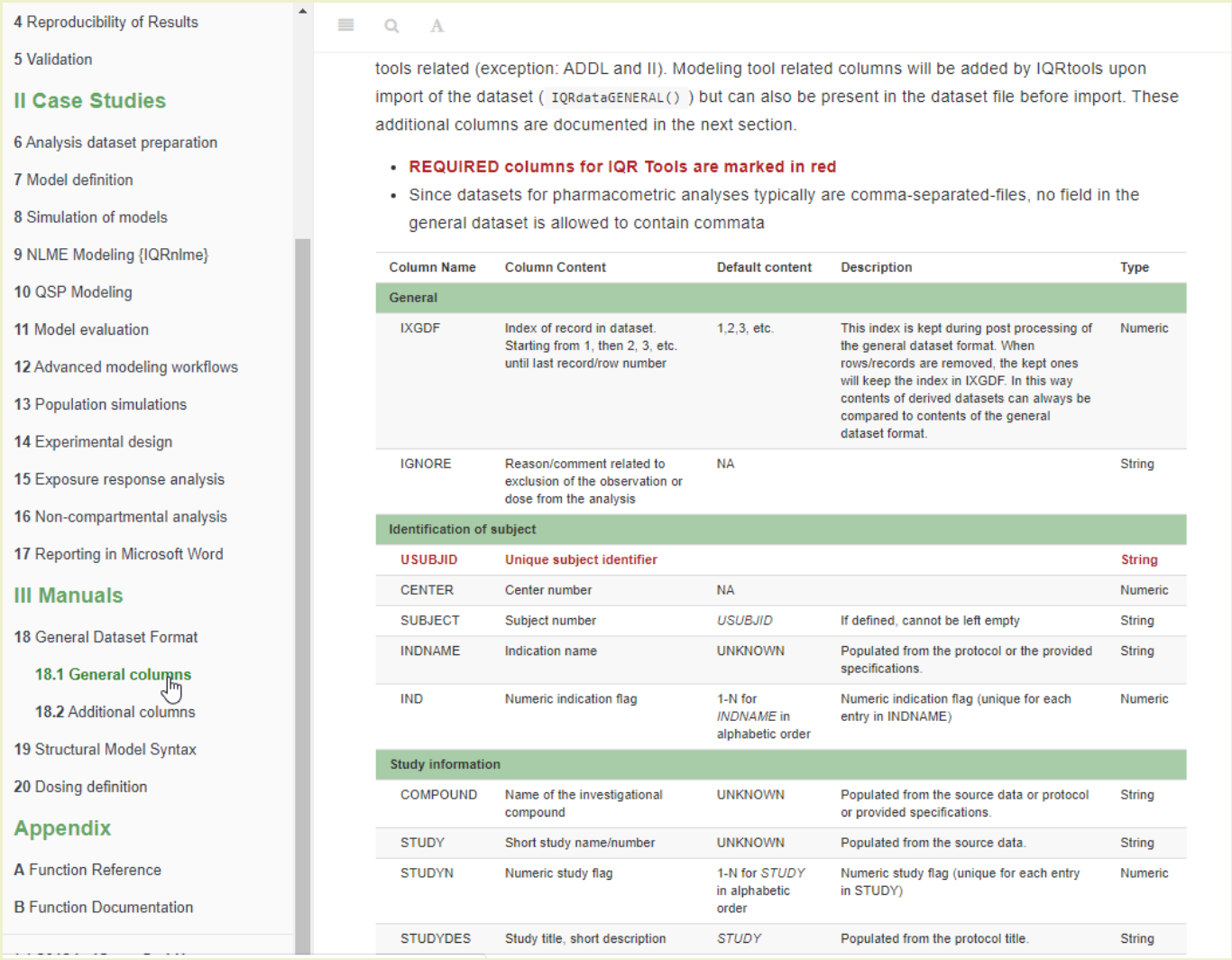
EVALUATION OF HYBRID MODELS (NLME+QSP) MANY COMBINATIONS AND FEW LIMITATIONS

- Example: Sequential NLME / QSP estimation for Malaria example

- Can be used in estimation or simulation

GENERALIZED DATASET FORMAT

GENERALIZED DATASET FORMAT



tools related (exception: ADDL and II). Modeling tool related columns will be added by IQRtools upon import of the dataset (`IQRdataGENERAL()`) but can also be present in the dataset file before import. These additional columns are documented in the next section.

- **REQUIRED columns for IQR Tools are marked in red**
- Since datasets for pharmacometric analyses typically are comma-separated-files, no field in the general dataset is allowed to contain commata

Column Name	Column Content	Default content	Description	Type
General				
IXGDF	Index of record in dataset. Starting from 1, then 2, 3, etc. until last record/row number	1,2,3, etc.	This index is kept during post processing of the general dataset format. When rows/records are removed, the kept ones will keep the index in IXGDF. In this way contents of derived datasets can always be compared to contents of the general dataset format.	Numeric
IGNORE	Reason/comment related to exclusion of the observation or dose from the analysis	NA		String
Identification of subject				
USUBJID	**Unique subject identifier**			**String**
CENTER	Center number	NA		Numeric
SUBJECT	Subject number	*USUBJID*	If defined, cannot be left empty	String
INDNAME	Indication name	UNKNOWN	Populated from the protocol or the provided specifications.	String
IND	Numeric indication flag	1-N for *INDNAME* in alphabetic order	Numeric indication flag (unique for each entry in INDNAME)	Numeric
Study information				
COMPOUND	Name of the investigational compound	UNKNOWN	Populated from the source data or protocol or provided specifications.	String
STUDY	Short study name/number	UNKNOWN	Populated from the source data.	String
STUDYN	Numeric study flag	1-N for *STUDY* in alphabetic order	Numeric study flag (unique for each entry in STUDY)	Numeric
STUDYDES	Study title, short description	*STUDY*	Populated from the protocol title.	String

https://iqrtools.intiquan.com/doc/book/GenDatFormat.html

IntiQuan

GENERALIZED DATASET FORMAT

- Representation of (pre-)clinically relevant information
- Humanly readable
- Easy to program from source data
- Able to capture all types of data typically informing QSP, Sysbio, and Pharmacometric modeling efforts
- Automatic conversion to NLME / QSP modeling datasets

CONCLUDING REMARKS

WHY WOULD YOU PREFER IQR QSP TOOLS OVER SIMBIOLOGY?

- **You would not …**
  - Both tools are highly synergistic
  - **SimBiology allows nice graphical representation**
    - Useful for discussion with biologists and the team
  - **IQR Tools allows powerful parameter estimation and fast simulation**
    - On a recent customer project the speed difference in simulation was **~45 fold** in favor of IQR Tools
    - Seamless NLME interface
    - …
  - **It is straight forward to convert a SimBiology model to an IQRmodel**

SUMMARY

IQR QSP Tools

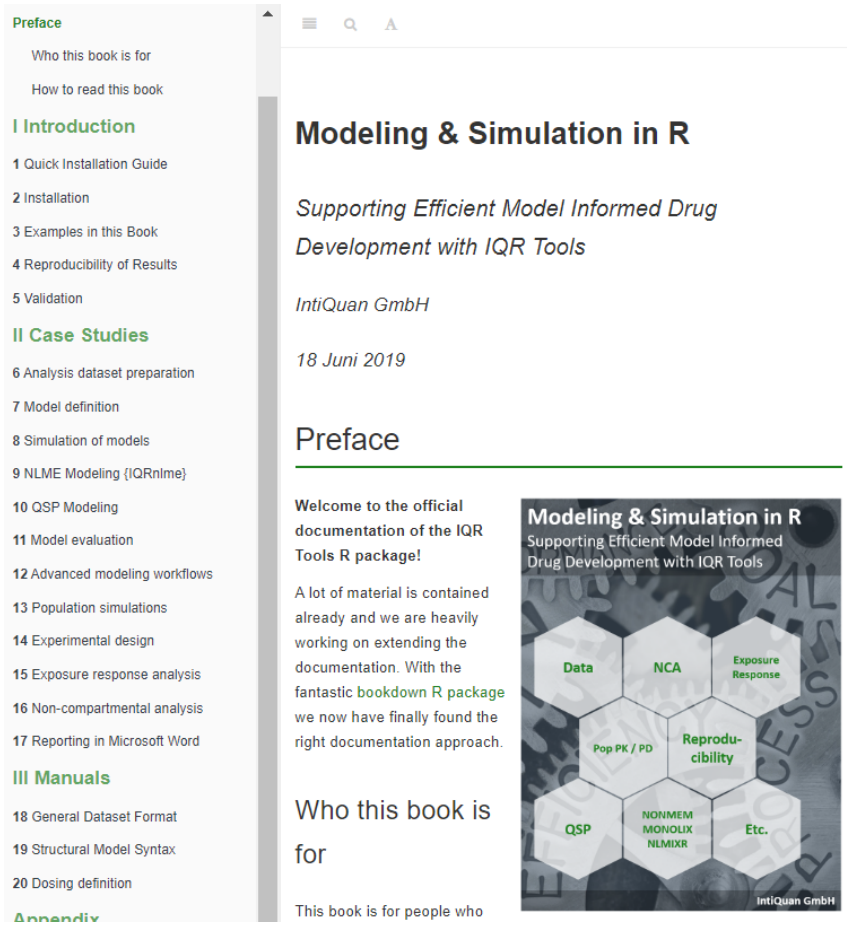
- **powerful**
  - fast simulation
  - accurate parameter estimation
  - user-friendly syntax
- **versatile**
  - mechanistic modeling
  - population modeling
  - combination of both
- **etc**
  - constantly improved
  - new features
  - fit for use in a regulatory context
  - script based approach leads to 100% reproducibility

WORKSHOP GOALS - CHECK

You should now be able to

- Write your own mechanistic QSP models in ODE or biochemical format
- Simulate models
- Setup data used for estimation and its exploration
- Performing powerful parameter estimation with and without consideration of parametric variability
- How to use model-based information to better inform the design of new experiments, increasing mechanistic understanding and trust in your model

(guided by the detailed documentation)



https://iqrtools.intiquan.com/doc/book/

IntiQuan

REFERENCES

Learn IQR Tools:

- https://iqrtools.intiquan.com/doc/book/index.html

The general QSP parameter estimation and uncertainty analysis methodology:

- https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0074335

The concrete QSP parameter estimation and uncertainty analysis methodology:

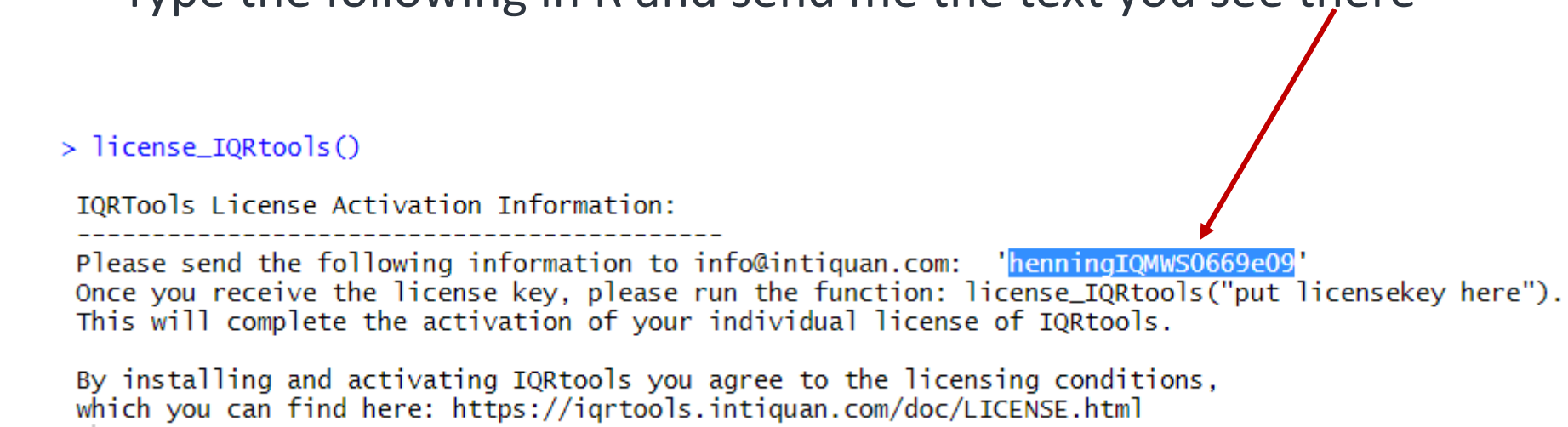
- https://www.jstatsoft.org/article/view/v088i10

The original Epo paper:

- https://science.sciencemag.org/content/328/5984/1404

IQR TOOLS LICENSES

- Every participant gets an own license valid >1 year
- Type the following in R and send me the text you see there



- I will send you a key that you can install as follows:

```
>
> license_IQRtools("PLACE KEY HERE")
```


IntiQuan

IntiQuan