Rosetta Stones in Population Pharmacokinetics and Pharmacodynamics
Rich Haney, Cellular Statistics, LLC

Introduction
1. In this work we look at the practical and also theoretical aspects of Rosetta Stones and their use in Population PK/PD. By a "Rosetta Stone" problem in this case we mean a problem of translating among data in different forms and models in different languages.
2. We make use of a "key" among languages that is not a language, but is algebraic in nature. In this context, we convert models to systems of equations that primarily involve the use of polynomials.
3. We hope to add to the public domain models originally in other languages (e.g., NONMEM) translated to Monolix.
4. We hope to be of assistance in deploying new standards or new platforms by companies such as the Pharsight Corporation.

Rosetta Stones - some practical aspects
1. There are many examples of these for financial and costing systems. For example, one program (also named "Rosetta Stone") in the past has been very good at converting back and forth between different representations of software cost models in different formats. (Those have been cost models obtained using nonlinear mixed effects analyses.)
2. Many Computer Aided Software Engineering (CASE) tools since the 1970s have used this idea. These features include automated construction of a user interface from an algebraic forms such as database schema and data dictionary. However, these are typically not formally defined.

Rosetta Stones - some mathematical aspects
1. As implied in the last note, taking a mathematical approach to Rosetta Stones is not crucial, but we think it has some benefits.
2. One perspective is that "key" or "ingua fancia (common language), for our Rosetta Stone is actually not really a language at all. Instead, it is their common algebraic structure or algebraic specification.
3. In accord with theorems from algebraic logic, it is helpful for our StStml to make use of various additional versions of our files containing equations. The different versions "relate" to one other. Therefore, instead of having datasets, we have "database algebra". Instead of having models, we have "algebras of models".
4. In accord with #3, we create various other versions of StStml files of equations. In fact, we find that each version has its own practical value. We see theory and practice very much aligned here.

Our own Rosetta Stone – some details
From the algebraic form, we then generate models in different languages. Here, we give here some details on our translator from BUGS to NONMEM. (Continuing our use of algebraic notation, we name our translator BUGS --> NONMEM, or BUGS2NONMEM.)

1. A NONMEM ADVAN I or ADVAN 2 type of model with single dosing yields for (i in 1:n.ind) {
   K[i] * Subject[i,1]…
   K[i] * Subject[i,1]…
   \[ K[i] \cdot Subject[i,1]\ldots \]
}
2. One perspective is that "key" or "ingua fancia (common language), for our Rosetta Stone is actually not really a language at all. Instead, it is their common algebraic structure or algebraic specification.
3. In accord with theorems from algebraic logic, it is helpful for our StStml to make use of various additional versions of our files containing equations. The different versions "relate" to one other. Therefore, instead of having datasets, we have "database algebra". Instead of having models, we have "algebras of models".
4. In accord with #3, we create various other versions of StStml files of equations. In fact, we find that each version has its own practical value. We see theory and practice very much aligned here.

StatML - Files of equations for data and models
1. We use StStml, a very simple form of XML in which we store vectors and equations in the same file. We use StStml for problems that involve POP PK/PD and proteomics applications together. The advantage is simplicity, flexibility, and modifiability. For Pop PK/PD, as the diagram below shows, we convert from NONMEM to StStml, and then other representations as follows:

   StStml - Sample Lines of Models & Data
In order to represent sets of models and data together (as so-called "algebraic varieties", and, more generally, "equational varieties"), we use about 12 types of equations overall. The following includes some different equations and examples of other equations as well:

   AMT(1): AMT <INPUTS> <[60000 60000 60000 60000 60000]>
   AMT(1): AMT <INPUTS> <[60000 60000 60000 60000 60000]>
   AMT: AMT <INPUTS> <[60000 60000 60000 60000 60000]>

   DV(1): DV <OBSS> <[198.5 100.7 248.6 1459.8 655.2]>
   DV(1): DV <OBSS> <[198.5 100.7 248.6 1459.8 655.2]>
   DV: DV <OBSS> <[198.5 100.7 248.6 1459.8 655.2]>

   In (DV + OBSS) D
   In (DV + OBSS) D
   In (DV + OBSS) D

   InitValues A
   InitValues A
   InitValues A

   DifferentialEq A
   DifferentialEq A
   DifferentialEq A

   \[ (\partial A[i] / \partial t) \cdot K[i] \cdot Subject[i,1]\ldots \]
   \[ (\partial A[i] / \partial t) \cdot K[i] \cdot Subject[i,1]\ldots \]
   \[ (\partial A[i] / \partial t) \cdot K[i] \cdot Subject[i,1]\ldots \]

   Integrals A
   Integrals A
   Integrals A

   iPred
   iPred
   iPred

In this approach, we translate NONMEM, BUGS and other models to StStml, viewed as equations of different types, and for which equation types and other vocabulary are given separately. We then convert back from StStml to practical languages such as NONMEM. We also convert StStml to formal languages, primarily typed lambda calculus.

Partial Summary of Results
1. We have translated a large number of models back and forth among languages including NONMEM, BUGS, Maple, and Monolix. We have probably done about 600 individual translations from some given source to some given target. However, to be really complete, we would need to do 5-10 times that number.
2. We were able to do about 80 round trips properly.
3. Our use of BUGS originally made use of the final estimates from NONMEM runs. In fact, this is how we would tend to use BUGS in general. On the other hand, we need to test all our BUGS models using the same initial estimates originally given to NONMEM.
4. Maple was extremely effective as a tool. However, use of Maple of coding up translation to polynomial form of categorical data models was not helpful as we had hoped. (We have not had true full categorical data models to use, though.)

Conclusion
We easily generated Monolix "project" files in MATLAB, stored separately from Monolix structural models. With respect to generation of Monolix structural models, we had some difficulty with generating multiple dosing models & new types of ODEs.

References

Acknowledgements
The concept of Open Statistical Services and StStml was partly in response to a meeting with the NONMEM consortium in September 2004 in which the author was involved. Our earlier work on StStml was with Patrick McConnell. For that we are also thankful for funding from California.

Several Michaud and Brad Sparks of Presens Technologies did our database & model comparison tool. This allows us to compare hundreds of models to one another.