Paper AD02

An EnrollmentModeling R Package

Stephen Gormley, Amgen Ltd., Uxbridge, United Kingdom

ABSTRACT

The Data Sciences team within the Centre for Design and Analysis ("**CfDA**") at Amgen Ltd. have developed an *EnrollmentModeling* R Package that, for a multi-centre Clinical Trial, predicts the probability of achieving an enrollment target time, the predicted stopping time with confidence bounds and the optimal site allocation given a variety of country, site and study specific constraints.

This paper explains the business use case for the underlying mathematical models and provides details of a few of the key design choices encountered during R Package development, testing, deployment and documentation including:

- A few of the key R packages used in the development of the software, including:
  - Infrastructure help with **devtools**[1]
  - Documentation using **roxygen2**[2].
  - Extensive testing using **testthat**[3].
  - Appropriate code coverage using **covr**[4].
  - Improved efficiency in the global optimisation with **Rcpp**[5].
  - Further enhancements to the global optimisation by a differential evolution algorithm **DEoptim::DEoptim**[6].
- **S3**[7], one of three Object Oriented ("**OO**") approaches in R, as the primary design choice.
- Functional layering of the code for ease of use and maintenance perspectives.

Note: The *EnrollmentModeling* R Package was built from the enrollment process models developed in *Anisimov & Fedorov (2007)*[9], *Anisimov (2011)* [10] with a number of accompanying complex R scripts. This presentation will not detail the underlying mathematical methodology that forms the basis of the package.

INTRODUCTION

DATA SCIENCES

The Amgen (Research & Development) Data Sciences team use site, country and investigator level data from a variety of internal and external data repositories to answer a number of business use cases. These repositories are consumed with automated data pipelines (programmed in R using Apache Airflow[11] as a scheduler) into a clinical data lake ("**CDL**") hosted on Amazon Web Services ("**AWS**")[12]. The data in the CDL is then consumed by a variety of R Packages, Python Modules, R Markdown ("**RMD**") Files and RShiny visualisations which are each developed, tested and deployed to answer one or more of these business use cases.

Note: The Data Sciences team is responsible for the development, testing, documentation and deployment of the automated data pipelines, the R Packages, RMD files, the Python modules and the visualisations.

This paper describes one of the Data Sciences solution's put in place to *Improve Clinical Trial Enrollment* and includes a high-level overview of the overall R solution and takes a more detailed look at the *EnrollmentModeling* R Package that was produced as an individual element of the R solution.

DATA SCIENCES TEAM

To give additional context, to this paper, the Data Sciences group is made up of a number of highly experienced full-time employees and as of **4 October 2019** is formed of the following team members:

- Eight Software Engineers (e.g. Java Programmers, Database Developers, SAS programmers, Platform Architects).
- Two Machine Learning ("**ML**") Experts.
- Two Statisticians.
- Two Source Data and/or Business Experts (with little to no knowledge of Statistics, Software Engineering nor ML).

KEY TAKEAWAY ITEMS

This paper has four main aims:

- First, to give readers an understanding of one of the **Amgen** business use cases and the use of R to solve the business use cases.
- Secondly, to give readers an understanding, at a high-level, of the complete R solution.
- Thirdly, **and primarily**, to give readers a greater understanding, with a more detailed look, of the *EnrollmentModeling* R Package which formed part of the overall R solution.
- Fourthly, to give an overview and understanding of three optimisation techniques used to speed up the solution for the third of the four requirements (See **FOUR REQUIREMENTS** section).

A FEW KEY DEFINITIONS

Clinical Trial: A Clinical Trial compares the effects of one treatment with another, it may involve subjects/patients, healthy people, or both. People volunteer to test new treatments, interventions or tests as a means to prevent, detect, treat or manage various diseases or medical conditions. A Clinical Trial may also be referred to as a Study in this paper.

Enrollment: Before a subject enrols in a Clinical Trial, they must be recruited, screened, and give their informed consent. Further, enrollment is strictly regulated and requires significant time and resource commitments.

A Product/A Drug: A product refers to the drug that is being assessed by the Clinical Trial to show that it is safe and works. One to many Clinical Trials may be required to prove that a product is efficacious and safe for patients before being approved by regulatory authorities.

Enrollment: US spelling of Enrolment (Amgen is a US Company).

Modeling: US spelling of Modelling.

BUSINESS USE CASE

IMPROVE CLINICAL TRIAL ENROLLMENT

Amgen runs a large number of Clinical Trials in a number of countries and as enrollment takes a long time (See **A FEW KEY DEFINITIONS** section), the quicker Amgen can enrol subjects into a Clinical Trial then the faster medicines can get to patients. Therefore, when planning a Clinical Trial and also at a mid-way point during a Clinical Trial (e.g. interim, safety review), key members of the study team need to make informed decisions regarding enrollment.

The main stakeholder in this process at Amgen is the **Global Clinical Study Manager** ("**GCSM**"), who is the person responsible for delivering all Clinical Trials for a treatment/product. This key stakeholder (and a few other key members), given a number of site, country and study level constraints (determined using historical data from a variety of internal and external data repositories), for each Clinical Trial at Amgen must consider a number of things, including:

- Which countries should be selected for a trial?

- How many sites should be selected in each country?
- In which countries should we recruit for the study that will:
    - enrol the fastest
    - cost the least
    - obtain a desired Probability of Success ("**PoS**"), given a target enrollment time and number of subjects in the trial.

Further, more specific questions are also considered for each individual study (based on senior leadership priorities), including:

- Do we need a reasonably high or low confidence of meeting enrollment target time (e.g. 12 months, 2 years)?
- Do we desire lower cost alternatives given we can achieve the target timeline with reasonable confidence?
- Should we consider a range of probability scenarios (e.g. 50%, 60%, 95%)?

PREVIOUS STATE

Before the overall R solution (See **OVERALL R SOLUTION** section) was designed and implemented the key stakeholders did have information at their disposal to answer a few of these questions. However, the key stakeholders:

- Had to source data from various internal and external data sources (which involved a number of manual steps and required multiple licences).
- Had to manually pull data from the individual providers website.
- Visualised their findings in PowerPoint.
- Did not have a solution that would return a PoS.
- Did not have a solution that would optimise site allocation across countries in the study.

SOFTWARE SYSTEM REQUIREMENTS

FOUR KEY REQUIREMENTS

Based on the key business use case the Data Sciences team documented four key high-level requirements that, if delivered, would answer the questions detailed in the previous section, as follows.

The Data Sciences team shall develop a solution that allows key stakeholders during clinical study planning to:

1. Determine the predicted stopping times and the probability of successfully achieving enrollment for a study given a target enrollment duration and the planned number of subjects, along with other key site and country level constraints.
2. Determine the predicted stopping time in **a specific country** given a target enrollment duration and number of subjects planned in the country.
3. Determine the optimal allocation of sites in each country (across the whole study) to minimise cost and/or speed of recruitment (given a number of study, site and country level constraints) for a number of probabilities to enrol by the target enrollment time.
4. Allow enrollment to be re-projected using historical and/or actual data mid-way through the Clinical Trial.

THREE FURTHER REQUIREMENTS

In order to improve the reliability of the data and the inference drawn from any statistical analysis three further key high-level requirements were documented, as follows.

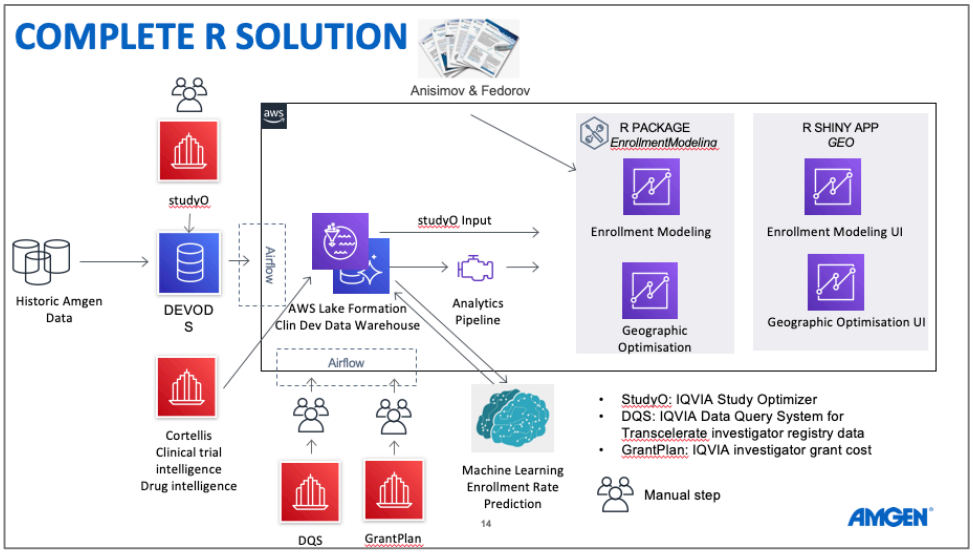
The Data Sciences team shall:

1. Identify the more appropriate external data repositories and create an automated pipeline to consume this data.
2. Store the data in an internal repository to ensure appropriate access controls and give easy access to all stakeholders.
3. Visualise all findings such that the solution can be deployed centrally and can easily be accessed by all stakeholders.

OVERALL R SOLUTION

From the seven higher level requirements the Data Sciences team designed, developed, tested, documented and deployed a complete R solution from the various data sources (the source) to the visualisations (the final target).

ARCHITECTURAL DESIGN



R SOLUTION DETAILS

This paper shall not go into detail regarding the complete R solution, though a number of key decisions and design choices were made, including:

- Data is consumed from a variety of internal and external repositories that have been reviewed and chosen by our Data Sciences Data Experts:
  - **StudyO**[13]: IQVIA Study Optimizer.
  - **DQS**[14]: IQVIA Data Query System (DrugDev).
  - **GrantPlan**[15]: IQVIA Investigator Grant Cost.
  - **Cortellis**[16]: Clinical Trial Intelligence.
  - **DEVODS**: Amgen internal repository.
- The CDL (and also the Data Sciences R server) is hosted on AWS.
- Credentials are secured using a Hashicorp Vault[17].
- All database interactions are run using R (e.g. DDL execution, data insert, update and/or delete).
- Data in the CDL is refreshed on a schedule using Apache Airflow (and a few Python scripts).
- Extraction of the data is made with R code using the external repositories API (or manual extraction where necessary).
- The *EnrollmentModeling* R Package consumes the automated pipeline of data and is also a standalone R Package deployed on R Studio Server Pro[18].
- Machine learning algorithms are trained, validated and tested for a specific use case around the creation of an enrollment rate (used to help select countries and sites) using the R *tidymodels*[19] framework.
- An RShiny application is deployed using R Studio Connect[20] and consumes the *EnrollmentModeling* R Package and visualises the results.

SPECIFIC R PACKAGE SOLUTION

In order to deliver on the four key requirements (See **FOUR KEY REQUIREMENTS** section) an *EnrollmentModeling* R Package was built from the enrollment process models and a number of complex R scripts developed by *Anisimov & Fedorov (2007)* [9], *Anisimov (2011)* [10].

WHY AN R PACKAGE?

R was chosen over other languages, and an R Package was developed over standalone R scripts, for a number of reasons, including:

- An easy to use and to setup testing framework.
- Easy to use code coverage tools.
- Easy to create in-line documentation
- Automatic code completion.
- A large number of Comprehensive R Archive Network ("**CRAN**") packages.
- Open source with a wide R community (that can be contacted and respond quickly to any questions and/or comments raised).
- Seamless GitLab[21] (and other source control) integration.
- Allows for an OO approach (i.e. **S3**, **S4** and **R6**), thus bringing the OO benefits of inheritance, polymorphism and encapsulation[22]
- Fundamentally, it is straightforward to develop, test, document and deploy an R Package for key stakeholders to consume.

Finally, R, R Packages and R's OO approaches help with adherence to good Software Engineering principles as described in the next section.

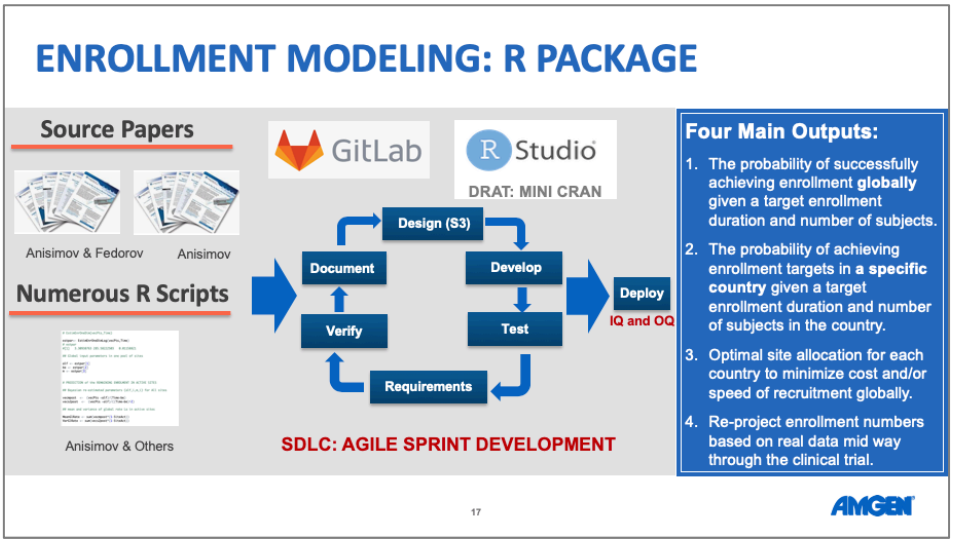
GOOD SOFTWARE ENGINEERING PRINCIPLES

The Data Sciences team has a core group of experienced Software Engineers (See **DATA SCIENCES TEAM** section) and aim to produce code that is at a minimum:

- reliable
- easy to use
- efficient
- well tested, with tests traceable to requirements
- well documented
- well commented
- (importantly) easy to maintain.

The Data Sciences team do this for two reasons: First, developing code that confirms to these principles is simply good Software Engineering practice; and secondly, Amgen is highly regulated and the Data Sciences team need to document a number of SDLC tasks in accordance with departmental, company and regulated policies.

R PACKAGE SDLC AND PLATFORM ARCHITECTURE



Before the design, development and/or testing of any code was initiated, an appropriate SDLC was chosen along with a few other key platform architectural decisions (decisions that have been made for all code developed from within the Data Sciences team), including:

- A (**W**)Agile scrum and sprint[23] methodology for software development is used, ensuring that requirements, design and testing are all documented in accordance with Amgen's standard templates/documentation for such software systems. (**W**)Agile: both **W**aterfall and Agile.

- GitLab is used for source control, Continuous Integration ("**CI**"), documentation, vignettes, readMe files and also as part of the full deployment process of dev, test and prod to a production R Server.

- R Studio Server Pro is used for development and testing of code, a Docker Image with a physical R server on AWS.

- An internal MINI CRAN repository has been created and used to resolve problems caused with numerous package version updates on CRAN, i.e. a gateway repository between CRAN and the Data Sciences server.

- In accordance with FDA guidelines a formal Installation Qualification ("**IQ**") and Operational Qualification ("**OQ**") process is documented and followed before full deployment of any software system.

Note: Although this paper does not go into detail regarding these architectural design decisions' I am more than happy to answer any questions on this subject (See **CONTACT INFORMATION** section).

R PACKAGE DEVELOPMENT

THREE OO APPROACHS IN R

With the business use case revolving around enrollment (e.g. Fitting Enrollment, Optimising Enrollment, Country Level Summaries of Enrollment, Reprojection of Enrollment) the solution leant itself to an OO solution and the benefits that an OO approach brings (particularly inheritance and polymorphism) and R has three OO approaches:

1. **R6:** a full on OO paradigm, with private and public members, static methods, instantiation of a class, encapsulated methods (with methods associated to an instantiated object), type checking, and, importantly, objects are mutable.

2. **S3:** a more lightweight OO approach, which is simple to use, ubiquitously used by R contributors and, importantly, objects are immutable. In its simplest form, S3 works by wrapping an R object in metadata and this metadata is then used to dispatch an object to its own object defined function.

   For example:

```
> x <- c(1,2,3)
> class(x)
> "numeric"
> class(x) <- "objectTypeNew"
> x
[1] 1 2 4
attr(,"class")
[1] "objectTypeNew"
```

   With the above code a function could now be written (along with a few other simple steps) as `print.objectTypeNew <- function() {…}` then when `print(anObjectOfTypeNew)` is run then the `print.objectTypeNew()` function is run. The user just types `print()` and under the hood R is dispatching to the appropriate function (i.e. signature override, method dispatch, polymorphism)

   Note: To see the S3 methods in R for any function, simply type `methods("functionName")` at the command line, e.g. `methods("print"), methods("plot")`

- **S4**: provides a more formal OO approach over S3, but is very similar to S3 other than that there are specialised functions for creating classes `setClass()` and methods `setMethod()`. In my limited experience S4 is rarely used in the R community.

For full details on S3, S4 and R6 please refer to the numerous online tutorials and guides on the topics and also see links in the **RECOMMENDED READING** section.

WHY S3?

S3 was chosen as the most appropriate OO solution for the *EnrollmentModeling* R Package (over the other two approaches) for four main reasons:

- First, S3's simple to use OO benefit of polymorphism (aka in R as method dispatch).
- Secondly, S3 still gives the OO benefit of inheritance via (…) and alternative ways (though it is a bit tricky).
- Thirdly, S3 is ubiquitously used by R contributors, easy to use and for others to comment.
- Fourthly, S3 is in accordance with the functional programming paradigm: when an object is passed into an S3 function it is not going to change.

Further, deferring to **Hadley Wickham** an R guru on this topic and many other R topics:

> *"Overall, when picking an OO system, I recommend that you default to S3. S3 is simple, and widely used throughout base R and CRAN. While it's far from perfect, its idiosyncrasies are well understood and there are known approaches to overcome most shortcomings. If you have an existing background in programming you are likely to lean towards R6, because it will feel familiar. I think you should resist this tendency for two reasons. Firstly, if you use R6 it's very easy to create a non-idiomatic API that will feel very odd to native R users, and will have surprising pain points because of the reference semantics. Secondly, if you stick to R6, you'll lose out on learning a new way of thinking about OOP that gives you a new set of tools for solving problems."* [24]

The Data Sciences team do use R6 quite extensively, the Java programmers in the team like it and for bigger projects it does seem to be preferable. Personally, R6 does seem to be verbose and is hard to navigate (a disadvantage from a maintenance perspective).

FURTHER PACKAGE DESIGN: FUNCTIONAL LAYERS

With a large number of complex R scripts and source papers another design choice (for a number of reasons, but primarily in order to make the code easier to use and easier to maintain) was grouping the code into four layers using R's S3 OO approach.

The four programming layers are as follows:

Layer One: Highest Level: Main Exposed APIS

- This level is exposed to the user, code is well documented, well commented, loggers incorporated, and careful thought was given to efficiencies in the code.
- All of these functions are individually tested and include:
  - S3 Classes (functions) that allow instantiation of the objects that contain the input data required and configuration.
  - Four S3 functions to answer the four high level requirements.
  - Getter functions to easily return key information from the complex list objects that are returned from the function calls.

Layer Two: Second Level Functions

- This level is not exposed to the user, though the code is still well documented and commented for other internal programmers, loggers incorporated, and careful thought was given to efficiencies in the code.
- All of these functions are individually tested and are simply used to dispatch to the third level functions based on the S3 configuration objects instantiated in Layer One.
- Rather than wrap S3 metadata around all objects, configuration is also used to dispatch (with IF ELSE) to differing algorithms dependent on site, study or country level constraints.

Layer Three: Third Level Functions

- This level is not exposed to the user, though the code is still well documented and commented for other internal programmers, loggers incorporated, and even more

careful thought was given to efficiencies in the code (as this does the main bulk of the work).
- These functions are not individually tested, as they are called from the second level functions via the configuration passed and are covered by tests of this layer. However, this code is reviewed by a QC Analyst.
- This layer contains the main set of controller code and does all of the hard work of the package.

Layer Four: Lowest Level Functions

- This level is not exposed to the user, code has skeleton documentation, minimal comments and loggers are not incorporated.
- This layer contains a large number of complex R functions supplied by the SME.
- No individual tests of these functions form part of package.
- These complex functions have been developed using a variety of input data with expert review (by the SME) of the extensive output. Further, the SME has over 40 years' experience in this field, a former Professor of Statistics with 200+ papers published, so we have a very high confidence that these functions work as expected.

KEY PACKAGES USED IN DEVELOPMENT AND TESTING

As described previously (See **WHY AN R PACKAGE?** section) R has an easy to use development, testing and documentation framework and a number of open source R Packages were used to help with the creation of the *EnrollmentModeling* R Package, including:

- **devtools**: Made developing the *EnrollmentModeling* R Package much easier, including, bootstrapping the infrastructure, creating documentation from roxygen2 tags, automatically creating the DESCIPTION/NAMESPACE files.

- **roxygen2**: Made the creation of the in-line documentation much easier. Using inline header tags in the R functions, **roxygen2** automatically rendered the tags into the required markup language that were needed in the package root `./MAN/` folder (i.e. no need to write markup language directly into the `./MAN/` folder).

- **testthat**: The unit testing framework for R which was used extensively in the *EnrollmentModeling* R package. The **testthat** package is easy to use and is ubiquitously used in the R community for testing of functions.

- **covr**: Used to track and report the code coverage for the *EnrollmentModeling* R Package throughout development and testing. The **covr** package is an easy to use tool that shows clearly the code coverage achieved for any package.

- **sinew**: Automated the roxygen2 comments. Another level of abstraction from creating markup files in the `./MAN/` folder, this used the function signature to automatically generate the **roxygen2** tags. This was extremely useful in bootstrapping the documentation in the large number of complex R scripts.

- **S3**: One of R's three OO approaches (See **WHY S3?** section).

- **Rcpp**: Used to integrate R and C++ and used specifically in the *EnrollmentModeling* R Package to reduce the optimisation time.

For further details on each of these packages refer to the CRAN repository which contains all of these package's and corresponding pdf Reference Manuals and inline help (e.g. https://cran.r-project.org/web/packages/devtools/index.html). Furthermore, full tutorials and guides can easily be found on line, including R package development best practices and also see links in the **RECOMMENDED READING** section.

R PACKAGE OUTPUT

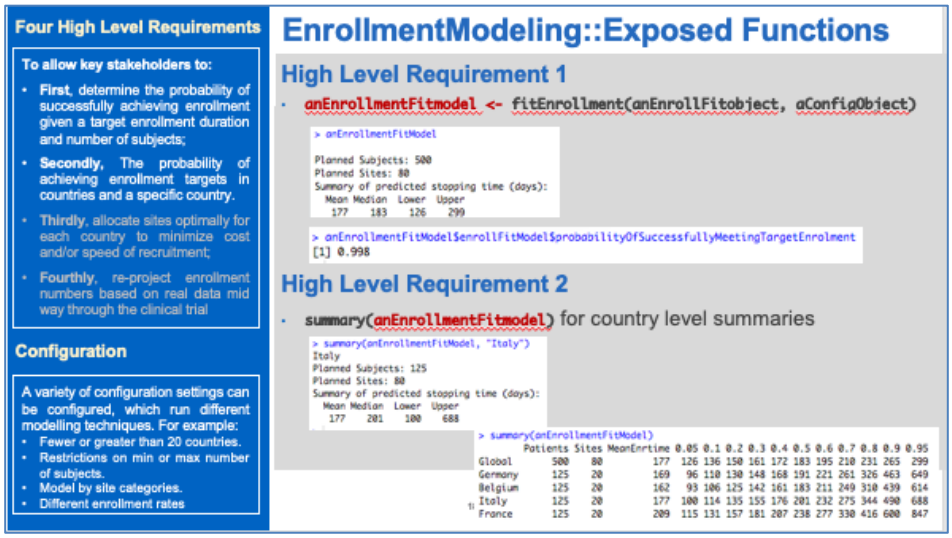
With reference to the four requirements: (See **FOUR REQUIREMENTS** section):

1. *Determine the probability of successfully achieving enrollment for a study given a target enrollment duration and the number of planned subjects (determined by the sample size).*
2. *Determine the probability of achieving enrollment targets in **a specific country** given a target enrollment duration and number of subjects planned in the country.*
3. *Determine the optimal allocation of sites in each country (across the whole study) to minimise cost and/or speed of recruitment (given a number of study, site and country level constraints) given a number of probabilities to enrol by the target enrollment time.*
4. *Allow enrollment to be re-projected using historical and/or actual data mid-way through the Clinical Trial.*

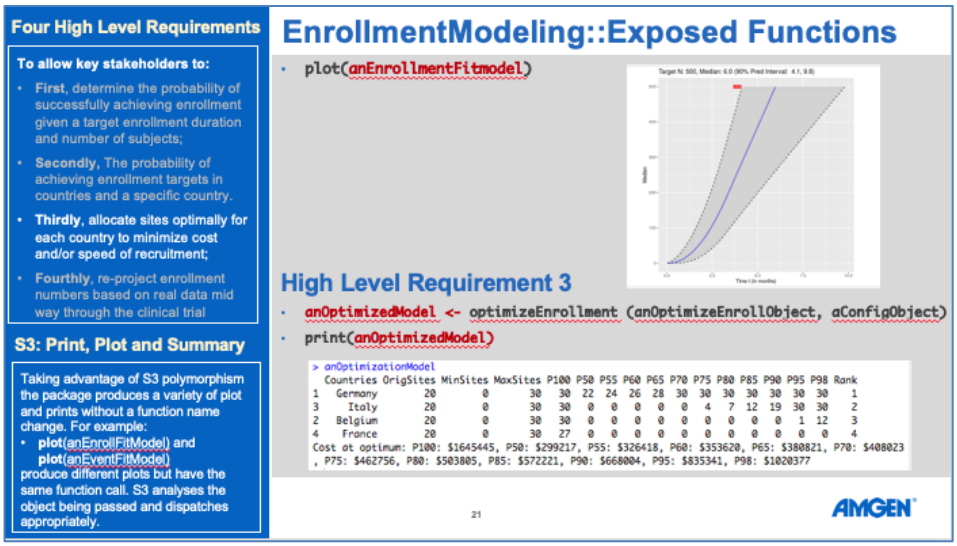
In order to deliver each of these requirements, the *EnrollmentModeling* R package was developed with a large number of outputs, complex list objects returned from each function call, extensions to existing S3 methods (i.e. `plot()`, `print()` and `summary()`) and also a set of completely new S3 functions.

High Level Requirement One: `fitEnrollment()` - A completely new S3 function that fits an enrollment model by accepting two S3 objects and returns the PoS and a summary of the predicted stopping time with confidence bounds.

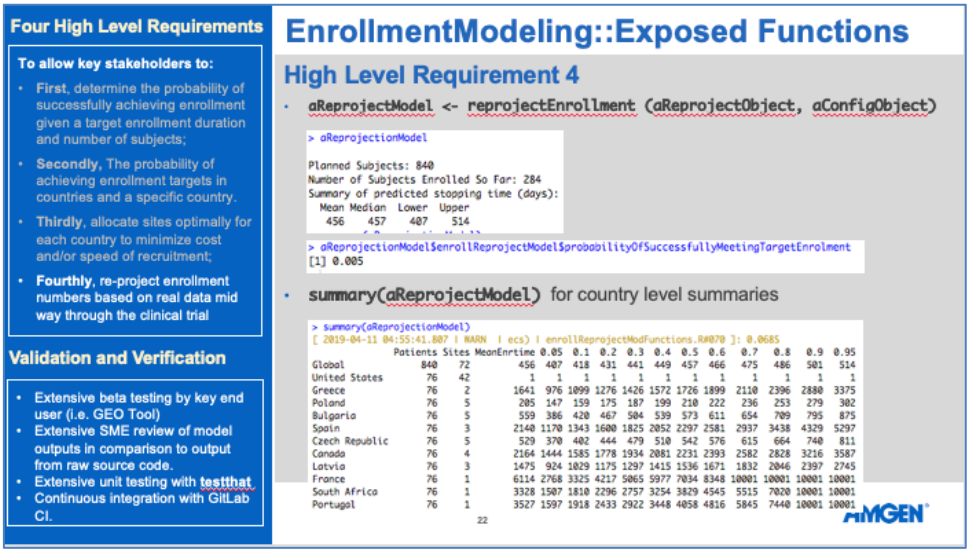
High Level Requirement Two: `summary(anEnrollmentFitModel)` – An extension to the existing S3 `summary()` function `summary.enrollfitmodel()` that uses method dispatch to run the function when an `enrollfitmodel` S3 object is passed. This function returns back a summary of the predicted stopping time by an individual country and a summary for all countries with various probabilities to enrol by the target enrollment time.



High Level Requierment Three: `optimizeEnrollment()` - A completely new S3 function that optimises enrollment by accepting two S3 objects and returns the optimal site allocation across all countries in the study that minimises cost given a variety of probabilities to enrol by the target enrollment time.



High Level Requirement Four: `reprojectEnrollment()` - A completely new S3 function that reprojects enrollment (at an interim timepoint) by accepting two S3 objects and returns the PoS and a summary of the predicted stopping time with bounds. Further, individual country level summaries are output using an extension to the existing S3 `summary()` function `summary.reprojectmodel()`



OPTIMISATION

One problem with the original SME code was the efficiency of the R scripts in delivering requirement three:

> *Determine the **optimal** allocation of sites in each country (across the whole study) to minimise cost and/or speed of recruitment (given a number of study, site and country level constraints) given a number of probabilities to enrol by the target enrollment time.*

This is a multidimensional optimisation problem with a number of countries with each country able to initiate 0 to j number of sites. So, the aim of this requirement is to find an allocation of sites across all countries that minimises:

- $\boldsymbol{TrialCost(N1 \dots NJ) = \sum_j (PtCost_j * MeanPts_j + SiteCost_j * N_j + CountryCost_j) * (N_j > 0)}$

Given that the probability to enrol, by the planned date, meets a minimum PoS threshold:

- $\boldsymbol{Pr(EnrollmentTime(N_1, \dots, N_j, Cap_j) \leq T_{plan}) \geq P_{plan}}$

Given that the number of sites in a country are restricted by country level minimums (e.g. in order to register a product in China, Amgen must have at least 5 sites and a minimum of 50 subjects as part of the filing) and maximums (i.e. only a finite number of sites in each country).

FIVE COUNTRY EXAMPLE

An example of a five country optimisation problem with each country having a maximum of 30 sites and various probabilities, is as follows:

```
  Countries OrigSites MinSites MaxSites P100 P50 P55 P60 P65 P70 P75 P80 P85 P90 P95 P98 Rank
1   Germany        20        0       30   30  29  30  30  30  30  30  30  30  30  30  30    1
3     Italy        20        0       30   29   0   0   2   6   7  12  14  18  23  27  26    2
2   Belgium        20        0       30    0   0   2   0   0   0   0   1   0   0   0   0    3
4    France        20        0       30   29   1   0   3   1   2   0   0   1   0   4  13    4
5       USA        20        0       20   15   1   0   0   0   1   0   0   0   1   1   1    5
Cost at optimum: P100: $1292136, P50: $415768, P55: $441666, P60: $471198, P65: $502058, P70: $537088,
P80: $616409, P85: $666257, P90: $732146, P95: $834624  P98: $928368
```

An example inference: with a site allocation of 30 in Germany, 26 in Italy, 0 in Belgium, 13 in France and 1 in USA would yield a 98% confidence of enrolling the target number of subjects in the target enrollment time, at a cost of **$928,368.** Alternatively**,** at a cost of **$732,146** with fewer overall sites would yield a 90% confidence. The GCSM can make a decision based on the priority of the study.

Note: this is obviously very computationally expensive, this specific five country scenario where each country can contribute up to 30 sites with an exhaustive search requires 31^5 possible combinations, which is equal to **28,629,151** (noting that we also calculate for a number of probabilities).

FOUR OPTIMISATION APPROACHES IN R

Four different approaches to solve this computationally expensive problem were investigated.

APPROACH 1: BRUTE FORCE EXACT LOOPING IN R

The original SME R scripts used a brute force exact approach to optimisation by looping through all of the possible scenarios. For each number of countries there is a separate function and for five countries is as follows:

```
FoptProbPG5 <- function(va,vb,vs2,vL,vU,nn,Pr){
opt.res <- c(sum(va*vU),vU)
for(i1 in vL[1]:vU[1]){
for(i2 in vL[2]:vU[2]){
for(i3 in vL[3]:vU[3]){
for(i4 in vL[4]:vU[4]){
for(i5 in vL[5]:vU[5]){
it1 <- c(i1,i2,i3,i4,i5,i6)
if(  PrTime(nn,vb,vs2,it1) < Pr  )
    opt.res <- opt.res
else{
if(sum(va*it1) >= opt.res[1])
    opt.res <- opt.res
else
    opt.res <- c(sum(va*it1),it1)
}}}}}
return(opt.res)
}
```

```
vL: Maximum Number of Sites
vU: Minimum Number of Sites
```

```
PrTime <- function(nn,vb,vs2,x){
i1 <- sum(x*vb)
i2 <- sum(x*vs2)
return(ifelse(i2<=0,
1-ppois(nn-1,i1),
1-pnbinom(nn-1,
size=i1^2/i2,
prob=i1/(i1+i2))))
}
```

This has the obvious advantage that it will always return the correct allocation of sites. However, it takes a long time to run (with **28,629,151** iterations) and for the five country example above takes **22mins** to return the optimal solution. Therefore, this really limits the number of countries and the maximum number of sites in a country that can realistically use this approach. Therefore, the *EnrollmentModeling* R Package restricted this to only work for up to **14 countries**.

APPROACH 2: BRUTE FORCE EXACT LOOPING IN RCPP

The first approach to improve the speed of the original code was to use C++ rather than R, R is mainly an abstraction of C++ so by programming in C++ the code is closer to the binaries. The Rcpp solution still uses the exact brute force looping but is one level of abstraction down. An example snippet of the C++ code:

```
NumericVector FoptProbPG5(NumericVector va,
                          NumericVector vb,
                          NumericVector vs2,
                          NumericVector vL,
                          NumericVector vU,
                          double nn,
                          double Pr) {
:
:
if (_sumIteratorAndVs2 <= 0) {
    _rpois=R::ppois(nn - 1, _sumIteratorAndVb,TRUE,FALSE);
    _1minusrpois= 1 - _rpois;
     returnVar=_1minusrpois;
} else {
    _multiplySumIteratorAndVb=_sumIteratorAndVb * _sumIteratorAndVb /
_sumIteratorAndVs2;
```

```
    _rpnbinom=R::pnbinom(nn - 1, _multiplySumIteratorAndVb, _sumIteratorAndVb /
(_sumIteratorAndVb +   _sumIteratorAndVs2),TRUE,FALSE);
    _1minusrpnbinom= 1 - _rpnbinom;
     returnVar=_1minusrpnbinom;
}
```

As the brute force R approach, this has the obvious advantage that it will always return the correct allocation of sites. However, the code is more verbose and was a little more difficult to understand with limited C++ experience in the Data Sciences team. Further, using `Rcpp` code requires a number of updates to the *EnrollmentModeling* R Package infrastructure, including:

- Code now has to be placed into the package root `./src/`
- `useDynLib(EnrollmentModeling)` in the NAMESPACE file
- `#include <Rcpp.h>`
- `using namespace Rcpp;`
- `// [[Rcpp::export]]`
- Need to *Install and Restart* rather than *CMD+SHIFT+L* (as C++ code needs to be compiled)

However, this code does run a lot quicker than the equivalent R code and for the five country example above takes **90 secs** to run. Microbenchmarking[25] of this function:

```
> microbenchmark::microbenchmark(optimizeEnrollment(…))
```

```
Unit: seconds
                                                                 expr      min       lq     mean   median       uq     max neval
 optimizeEnrollment(modelObj, additionalConfigObj, additionalCostObj) 86.06418 86.08524 86.13695 86.10583 86.11723 86.3662   100
```

APPROACH 3: USING DEOPTIM

The second approach to improve the speed of the original code was to use the optimisation package `DEoptim` and the specific function `DEoptim`, which performs an evolutionary global optimisation using a differential evolution optimisation algorithm.

```
DEoptim::DEoptim(deOptimFunc,
# specify lower and upper bounds on
each parameter to be optimised
                 lower=vL,
                 upper=vU,
# Other arguments needed for function
                 nn=nn,
                 vb=vb,
                 va=va,
                 vs2=vs2,
                 vec_cap=vecCap,
                 prob_criteria = i,
                 vAl = vAl,
                 vprob = vprob,
# a list of control parameters
                 control =
list(initialpop=tmp_population,
     itermax = 400,
     trace=FALSE)
:
:
)
```

Function to minimise:

```
deOptimFunc <- function(x,va,vb,vs2,nn,Pr) {
     if(PrTime(nn,vb,vs2,x) < Pr) {
        return(Inf)
      }
      return(sum(va*x))
  }
```

A few of the key arguments:

- `lower` – minimum number of sites vector.
- `upper` – maximum number of sites vector.
- `Other arguments needed for the function.`
- `initialpop` – initial solution from which to optimise
- `Itermax` – maximum number of iterations.
- `fnMap = round` – return an integer solution.

This code runs a lot quicker than the `Rcpp` solution, and for the five country example above takes **~7 secs** to run. Microbenchmarking of this function:

```
> microbenchmark::microbenchmark(optimizeEnrollment(…))
```

```
Unit: seconds
                                                                 expr      min       lq     mean   median       uq      max neval
 optimizeEnrollment(modelObj, additionalConfigObj, additionalCostObj) 6.70907 6.873101 7.101971 7.001533 7.258835 8.329486   100
```

However, this did take a little deconstructing of the original function (Now: **`deOptimFunc`**) and further it might return a false minimum for larger country problems (particularly as this algorithm is not a full integer solution, `DEoptim` uses continuous values and `fnMap = round` has to be set to get an integer return). Nevertheless, in our testing this function is close to the brute force exact solution with up to ten countries.

APPROACH 4: USING GAISL

The next update to the package will be the use of `GA::GAISL` which is a full integer solution using maximisation (multiplied by **-**1) of a fitness function using islands genetic algorithms. However, unfortunately, this has not been incorporated into the *EnrollmentModelInig* R Pacakge as of **4 October 2019**. For further details, see the head of the Data Sciences team presentation given at the ***Applied Stochastic Modeling Conference 2019*** (http://www.asmda.es/asmda2019.html) and the paper: ***An analysis of Gray versus binary encoding in genetic search [Uday K. Chakraborty *, Cezary Z. Janikow, 2003].***

CONCLUSION

This paper has described one of the many Amgen business use cases that have been solved by the R&D Data Sciences team using R as the primary solution. The department, in which the GCSM is a member, have been using the complete R solution to help inform and improve Clinical Trial enrollment in a number of studies for the past 18 months. Although the subjective feedback has indicated the solution is working well, the Data Sciences team are currently creating an evaluation framework to test the predictions against real enrollment figures from Amgen Clinical Trials to ensure the predictions are as accurate as we expect (full deployment of the Evaluation Framework due **December 2019**).

This paper has also given readers an understanding, at a high-level, of the complete R solution and taken a more detailed look of the *EnrollmentModeling* R Package which formed part of the overall R solution (including a few optimisation techniques that can be applied to any multi-dimensional optimisation problem). However, a number of ideas in relation to R are at quite a high level, so if any reader would like to dive deeper into any of the details and/or has any questions related to anything in this paper then I am more than happy to be contacted via the details provided in the **CONTACT** section below.

REFERENCES

[9] *Anisimov V., Fedorov V.,* Modelling, prediction and adaptive adjustment of recruitment in multicentre trials, Statistics in Medicine, 26, 2007, 4958-4975.

[10] *Anisimov V., Statistical* Modelling of Clinical Trials (recruitment and randomization), Communications in Statistics - Theory and Methods, 40: iss. 19-20, 2011, 3684-3699.

[22] Advantages of using the object-oriented paradigm for designing and developing software Applied Computing, Mathematics and Statistics, Lincoln University, Canterbury, New Zealand [https://pdfs.semanticscholar.org/0c3b/98172de01f22f1694bd582dec6163f614a95.pdf]

[24] *Hadley Wickham*, Advanced R, https://adv-r.hadley.nz/oo-tradeoffs.html

---

ACKNOWLEDGMENTS

- **Matt Austin**, Executive Director, Data Sciences, Amgen Ltd, Thousand Oaks , California.
- **Vlad Anisimov**, Senior Manager, Data Sciences, Amgen Ltd, Cambridge, Cambridgeshire.
- **Marina James,** Manager, Data Sciences, Amgen Ltd, Cambridge, Cambridgeshire.

RECOMMENDED READING

[1] **devtools**: https://cran.r-project.org/web/packages/devtools/index.html
[2] **roxygen2**: https://cran.r-project.org/web/packages/roxygen2/index.html
[3] **testhat**: https://cran.r-project.org/web/packages/testthat/index.html
[4] **covr**: https://cran.r-project.org/web/packages/covr/index.html
[5] **Rcpp**: https://cran.r-project.org/web/packages/Rcpp/index.html
[6] **DEoptim**: https://cran.r-project.org/web/packages/DEoptim/index.html

[7] **S3:** http://adv-r.had.co.nz/S3.html
[8] **Docker Images**: https://support.rstudio.com/hc/en-us/articles/360021594513-Running-RStudio-with-Docker-containers
[9] *See References*
[10] *See References*
[11] **Apache Airflow**: https://airflow.apache.org
[12] **Amazon Web Services**: https://aws.amazon.com
[13] **IQVIA Study Optimizer**: https://www.iqvia.com/solutions/technologies/clinical-acceleration
[14] **IQVIA Data Query System (DrugDev)**: https://www.drugdev.com/solutions/site-selection/
[15] **GrantPlan, Investigator Grant Cost**: http://sp.grantplan.com/login.aspx
[16] **Cortellis, Clinical Trial Intelligence**: https://www.cortellis.com/intelligence/login.do
[17] **Hashicorp Vault**: https://www.vaultproject.io/
[18] **R Studio Server Pro**: https://rstudio.com/products/rstudio-server-pro/
[19] **R tidymodels framework**: https://rviews.rstudio.com/2019/06/19/a-gentle-intro-to-tidymodels/
[20] **R Studio Connect**: https://rstudio.com/products/connect/
[21] **GitLab**: https://about.gitlab.com
[22] *See References*
[23] **Agile Development with Sprint and Scrum**: https://www.dummies.com/careers/project-management/the-function-of-the-scrum-and-sprint-within-an-agile-project/
[24] *See References*
[25] **Microbenchmarking**: https://cran.r-project.org/web/packages/microbenchmark/index.html

---

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Stephen Gormley
Amgen Ltd.
1 Sanderson Road
Uxbridge
UB8 1DH
Tel: 01895 525 328