Paper RE06

# Automating Real-World Data Analysis using R Packages and Shiny: A Three-Stage Workflow

Alexandrina Lambova, IQVIA, Sofia, Bulgaria
Nikolay Trankov, IQVIA, Sofia, Bulgaria
Elena Chaparova, IQVIA, Sofia, Bulgaria
Stavros Oikonomou, IQVIA, Sofia, Bulgaria
Louise Raiteri, IQVIA, London, United Kingdom

## ABSTRACT

Real-world data (RWD) plays a crucial role in healthcare and research, yet its accessibility is hindered by legal and privacy constraints. Researchers often develop programming code on synthetic data, with only summary results available when applied to real data. To address this, the IQVIA Biostatistics analytics hub have devised an automated, three-stage approach for conducting RWD studies. The first stage involves a data-source specific R package for standardised, validated transformation and cleaning of RWD. The second stage employs a data-source agnostic R package for cohort selection, summary output creation, and visualisations based on individual patient data, incorporating statistical disclosure control. The final stage utilises a Shiny[1] application to provide visualisations based on summarised data created by the R package. We will present the successfully implemented approach using the Cancer Analysis System, UK's national cancer registry, and its synthetic data Simulacrum, demonstrating its effectiveness in navigating the complexities of RWD research.

## INTRODUCTION

The development of drugs is done under strict control, with routine data collection during each clinical trial phase. Once approval has been obtained and the drug has been released to market, additional data is only regularly collected to monitor safety for the purposes of pharmacovigilance. In recent years, the use of data, collected for other purposes (called secondary data), is increasingly utilised to monitor drug efficacy. Since this data was not generated for the specific purpose, it avoids skew and reflects the actual real-world usage of drugs. Therefore, the data collected from the real world is called Real-world data (RWD) and the evidence stemming from it is called Real-world evidence (RWE).

### REAL-WORLD DATA CHALLENGES

In its essence, RWD is collected for specific purposes, often for claims or reimbursement. Therefore, it typically contains very detailed and patient sensitive information. Data related to patient health and containing multiple sensitive data points is under strict protection from government and data holder policies which can make the access, analysis and reporting of the results challenging. Accessing data and regulatory compliance and disclosure control rules can also vary between databases. Additionally, since RWD is not collected for the purpose of RWE analysis it is often messy, heterogeneous and susceptible to various measurements errors and missingness, resulting in lower quality compared to data from controlled trials. Often proxies and algorithms are required to derive important study variables. One further challenge of RWD is the lack of common data structure and format between databases which leads to the need to spend time to understand and bring the data in the structure needed to perform analysis. Despite these challenges, RWD remains valuable for evidence generation and decision-making and is playing an increasing role in healthcare [2].

### OUR GOALS

Our overarching goal is to improve the programming and analysis of RWE studies. Historically the delivery of RWE studies has followed a bespoke approach despite commonalities between studies which poses increased risk for errors and time inefficiencies. Our aims are to increase the quality of the study outputs, minimise errors and reduce the time and resources for analysis and training required via providing an automated and standardised pipeline for RWD analysis.

### TYPICAL RWE STUDY PROGRAMMING WORKFLOW

The programming of RWE studies is typically a complicated process that involves multiple stages and aspects of the analysis. It could be broken down into the following steps:

***Data loading*** - whether the study data comes from a national registry with millions of records or from a small subset of electronic health records, it always needs to be loaded into a time- and memory- efficient way, in order to be analysed.after that.

***Data formatting*** - working with RWD always entails some degree of messiness in the data. This can range from minimal formatting to complex transformations needing to be applied.

***Data cleaning*** - since RWD is not rigorously collected for the purpose of data analysis, sometimes it can contain invalid or missing values. That is why some cleaning procedures should always be applied.

***Cohort definition*** – each study defines the analysis cohorts to be used. These definitions depend on multiple factors and are study specific.

***Variable definition*** – each study defines the variables that should be used in further analyses. Proxy algorithms are often required for the derivation of some of the needed variables.

***Analysis*** – the most important and complicated part of the study is the actual analysis of the data. It can range from simple summary statistics to sophisticated models.

***Reporting*** - the outputs from the analyses, applied to the data, should be recorded in analysis tables and figures, which need to be exported to a wide range of formats.

***Visualisation and interpretation –*** some RWD sources don't allow direct access to their data. In such cases only summarised results are available. Further visualisation of the summarised results is often needed for easier interpretation and reporting.

This pipeline of steps represents the standard RWE study programming workflow. The complexity of each step could vary depending on the used data source and the performed analysis. Some parts of these analyses can be reused between different studies. This could reduce the needed resources but also increase the possibility of errors if not implemented in a proper way. Best programming practices and concepts should be used to create a validated but flexible framework in order to automate as many steps from the study pipeline as possible.

## OUR 3-STAGE APPROACH

To achieve our goals for high-quality RWE studies and reduction of the needed resources we developed a three-stage approach that automates the study analysis workflow and allows generation of standard high quality study outputs. For its implementation we used the R programming language that is specialised for statistical analysis but at the same time is a very flexible and powerful tool for development and automation. There are various validated open-source R packages that allow easy application of different statistical analyses using best statistical practices. All these advantages come with a great suite of tools for development of R-based dashboards and applications such as the Shiny packages that make R well-suited for implementation of the whole analysis workflow automation.

The study analysis workflow can be split in four parts:

- Data source specific programming – includes the data loading, formatting, and cleaning steps. Each data source has its specifics that should be considered. Specific automation is needed for each data source.
- Project specific programming – includes definitions of study cohorts and variables. They can vary significantly between different studies depending on their indication and objectives. They are very difficult to automate.
- Analysis and reporting – the statistical analyses and outputs can vary between the studies depending on their objectives and specifics. However, there is a discrete number of standard analyses and objectives used in the RWE studies. Therefore, automation that allows flexibility, easy extensibility and customisation is needed. It is challenging but possible to implement.
- Visualisation and interpretation – can be automated relying on standard formatting and structure of the input data.

We created a family of R packages and tools that automate three of the four parts of the study analysis workflow. Because of its modular, puzzle-like architecture, separated in three independent stages, it allows great flexibility that fits in most of the RWE studies. Each of the stages can be easily extended without affecting any of the other parts.

#### STAGE 1: DATA LOADING AND WRANGLING

The first stage of our approach covers the data source specific parts of the study programming. It includes loading and wrangling of the initial analysis data. The real-world data is often very large and consist of multiple linked data sets. Its structure and formatting can vary significantly between the different data sources. Multiple cleaning and formatting steps are needed to transform the data into a usable format.

To automate this stage of the programming pipeline we developed data source specific R packages. Each package uses generic data source agnostic loading and formatting functions but defines its own loading, formatting, and

cleaning rules. These rules are provided in structures called specifications. Each package contains pre-defined default specifications for the data source. However, they can easily be extended or changed for each study if needed.

**Loading specification** - the loading specification is an R list containing one item for each dataset that should be loaded. Each item contains information about the path to the data set and a list of the columns to be loaded and their specific format.

The basic format of the loading specification is as follows:

```
loading_spec <- list(
  <dataset1> = list(
    input = <path_with_filename1>,
    select = c(
      <column1> = <class to be loaded as>,
      <column2> = <class to be loaded as>
    )
  ),
    <dataset2> = list(
    input = <path_with_filename2>,
    select = c(
      <column1> = <class to be loaded as>,
      <column2> = <class to be loaded as>,
      <column3> = <class to be l oaded as>
    )
  )
)
```

Below is a simple example of a loading specification:

```
loading_spec <- list(
  patient = list(
    input = file.path(path, "patient.csv"),
    select = c(
      PATIENTID = "character",
      VITALSTATUS = "character",
      ETHNICITY = "character")
),
    tumour = list(
    input = file.path(path, "tumour.csv"),
    select = c(
      TUMOURID = "character",
      AGE = "character",
      DIAGNOSISDATE = "character",
      TUMOUR_STAGE = "character")))
```

**Formatting specification** - the formatting specification is an R list that contains one item for each dataset to be formatted. Each item contains lists of formatting functions to be applied on the columns in the dataset. Each data

source specific package contains a predefined set of formatting functions consistent with the specifics of the data. However, these functions can be easily extended for the specific study if needed.

The basic format of the formatting specification is as follows:

```
formatting_spec <- list(
  <dataset1> = list(
    list(<column1>, <function1>, <optional function1 arg1>),
    list(<column2>, <function2>, <optional function2 arg1>)
  ),
  <dataset2> = list(
    list(<column1>, <function3>, <optional function3 arg1>),
    list(<column2>, <function4>, <optional function4 arg1>),
    list(<column3>, <function5>, <optional function5 arg1>)))
```

The following example shows the formatting of the factor columns VITALSTATUS and ETHNICITY from the patient dataset. Pre-specified rules *format_vitalstatus* and *format_ethnicity* are used. Additional arguments can be provided to the functions when needed. The columns AGE, DIAGNOSISDATE and TUMOUR_STAGE from the tumour dataset are also formatted and converted to the desired types:

```
formatting_spec <- list(
 patient = list(
    list(VITALSTATUS, format_vitalstatus),
    list(ETHNICITY, format_ethnicity, granular = FALSE)
  ),
 tumour = list(
    list(AGE, as.numeric),
    list(DIAGNOSISDATE, format_date),
    list(TUMOUR_STAGE, format_stage)
  )
)
```

The usage of data source specific packages for data loading and wrangling significantly reduces the amount of time and efforts needed for the programming of this stage of the analysis, especially for data scientists that are not very familiar with the specifics of the data source. The usage of validated functions and specifications also significantly reduces the time needed for quality check of the data wrangling, guarantees better quality of the produced data and easier subsequent analysis.

#### STAGE 2: ANALYSIS

The second stage of the workflow combines analysis of the study population and reporting of the summarised results. Our R package called “IQVIA R Analysis” implements a set of flexible and extensible functionalities that allow easy generation of various statistical analyses and outputs with a standard structure and formatting, including attrition tables and flowcharts, analysis tables, treatment sequencing tables and Sankey diagrams, time to event analysis outputs. Small number suppression is embedded in all provided outputs. Additionally the package allows generation of table shells in the standard format during the design of the study without the need to access the analysis data.

The architecture of the package is based on the main principles of object oriented programming in combination with powerful R packages and concepts such as S4 classes*, data.tree* [3] and *data.table* [4]. The hierarchy of classes and interfaces implemented behind allows for very easy extending of the package functionalities both during the package development or outside the package during programming of study analysis. The usage of generic functions allows customisation of the analysis included in the produced outputs. Each user can define their own analysis and include the results from them in the output tables if they follow the predefined format of the generic functions. Tree-based structures are used to allow flexible hierarchies of variables with nested levels.

**Attrition functions**

The first step in each RWD analysis is the application of the study inclusion and exclusion criteria and the generation of an attrition table and/or attrition flowchart diagram to show the counts and percentages of patients meeting each of the study criteria. The hierarchy of the attrition criteria and the derived analysis cohorts and sub-cohorts could very easily and naturally be represented by a tree-based structure. We use this abstraction in the IQVIA R Analysis package and provide a set of attrition functions that allow step-by-step building of an attrition tree during the application of the study inclusion and exclusion criteria on the initial data. This approach allows filtering of the data to be applied directly at each attrition step/criterion without the need for storage of additional intermediate counts or binary variables. The constructed attrition tree can be converted directly to a flowchart diagram or to an attrition table. Preorder traversal of the tree (root – left – right) is used for the generation of the table by default. These are the main attrition functions included in the package:

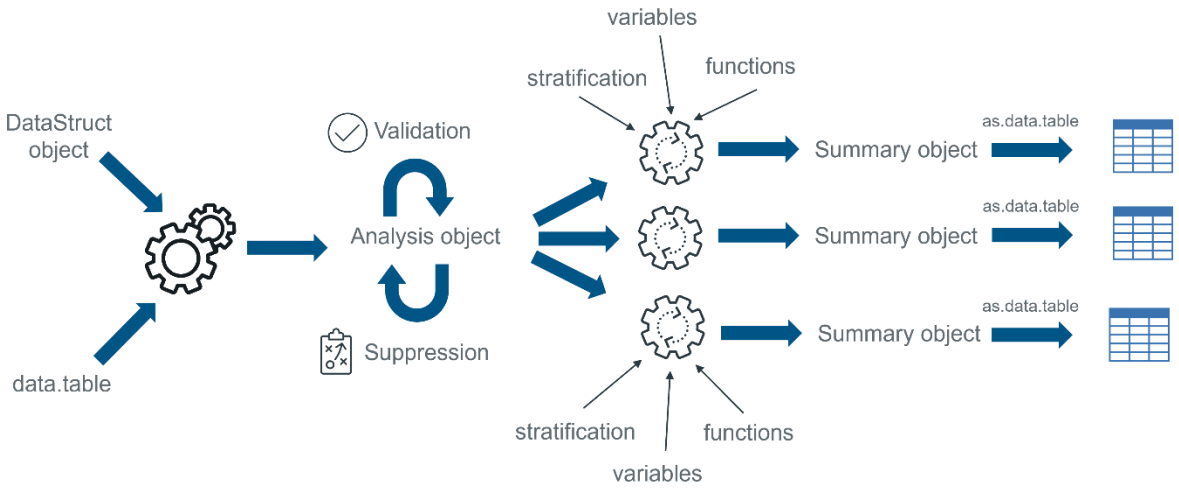
- start_attrition(…) – function that creates an attrition tree based on an initial cohort of patients. Stratification based on one or more categorical variables can be specified.
- add_attrition_step(…) – function that adds a new attrition step/criterion as a node in the attrition tree. It can be created from the filtered at this step data or from a binary variable in the data set. Each attrition node contains information about the entity (patient) ids included, accompanying label to be presented in the attrition table/diagram, type of the criterion (inclusion/exclusion).
- add_nested_steps(…) – function that allows the addition of a whole sub-tree/hierarchy to the attrition tree. The hierarchy is created automatically from one or more categorical variables provided to the function. This functionality is very useful when splitting of the patients by categorical variables should be reported in the attrition table.
- create_attrition_table(…) – function that gets an attrition tree and returns an attrition table with a standard structure and formatting. It allows great flexibility in the percentage calculations. Various modes of small number suppression are included.
- create_flowchart_diagram (…) – function that gets an attrition tree and returns a flowchart diagram with a standard structure and formatting.

**Analysis tables**

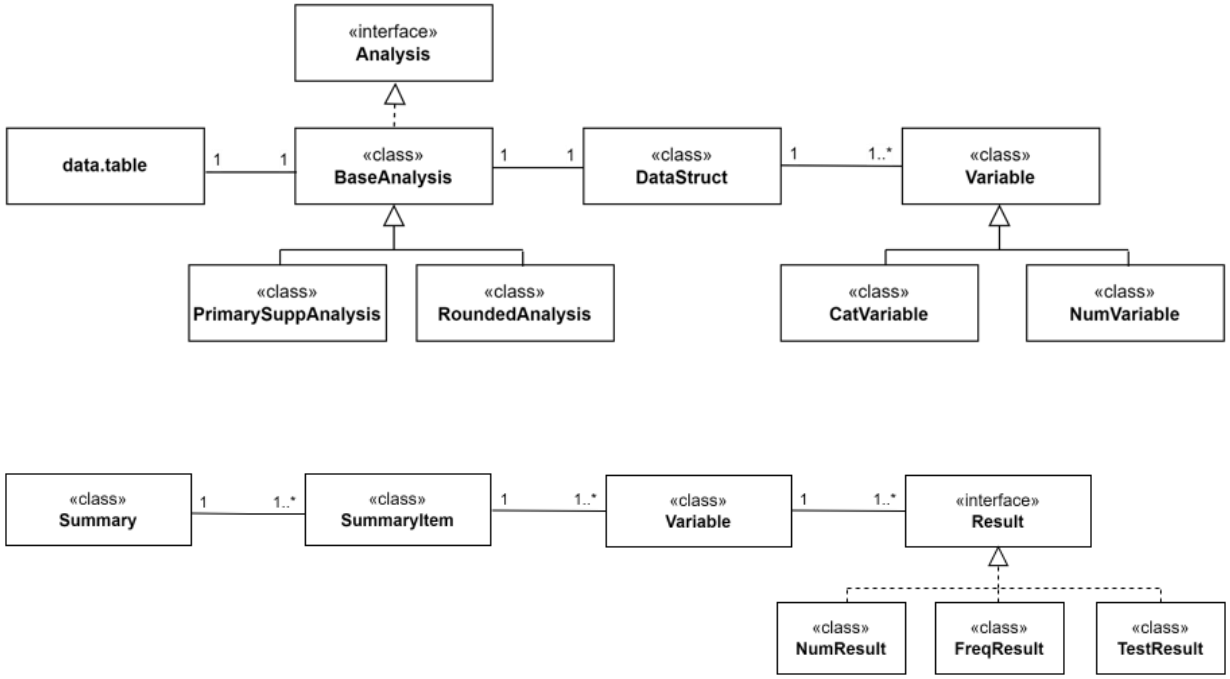
The development of a functionality that allows flexible and customizable generation of analysis tables is a challenging task. It should allow easy creation of outputs with standardised formatting but at the same time flexibility and extensibility of the statistics and analysis included in the tables (e.g. summarised statistics such as mean, median, percentiles; frequency counts and percentages; rates; results from comparison tests; coefficients and p-values from regression models, etc.). An additional difficulty is the need for implementation of small number suppression in the analysis tables. The suppression rules can vary between the different data sources. Low-resolution suppression includes rounding of counts, percentages, rates and other values. High-resolution suppression combines primary suppression of small patient counts and secondary suppression that protects the primary suppressed values from back calculation. The applied suppression should be consistent between all study outputs.

To solve these challenges, we created a pipeline of S4 classes and generic functions that cover the whole process of analysing and reporting results for a data set. These are the main steps included in the pipeline:

1) Create a DataStruct object that contains the formal definitions of all analysis variables (column name, label, variable type, validation rules).
2) Create an Analysis object from a given analysis data set and a DataStruct object. Validation of the data set against the variable definitions is applied. Data set level suppression is applied.
3) Pass the Analysis object to a function that creates a Summary object. Each Summary object represents one output table with its own subset of variables, stratification and analysis included.




The object-oriented approach used in the implementation of this pipeline allows easy extension with new suppression methods and new types of variables (both as part of the package or by a user for a specific study). Users can define their own validation or analysis functions that meet a predefined structure. Each analysis function can modify or add a new Result object to the output tables. The Summary object contains all information needed for the output in a structured format. It can be easily converted with the *as.data.table* function to a data table and exported to an Excel or Word file.




**STAGE 3: VISUALISATION AND REPORTING**

The final stage 3 of our workflow allows for easy creation of various visualisations and outputs based on summarised study results. The need for this functionality has risen due to some of the challenges related to RWD. As alluded to before, sometimes access to patient-level data is not granted to programmers, rather programming code is packaged and sent to data holders that run the analysis. The results delivered are large data tables containing the summarised

analysis results, often in Excel files. Once these results are reviewed, additional visualisations are needed for easier interpretation and presentation of the insights.

Stage 3 of the analysis pipeline is often conducted by statisticians and epidemiologist without programming experience. That is why a user interface is needed to automate their work and to reduce the needed time and effort. The Shiny package was used for the creation of an application that allows the generation of various visualisations and outputs based on summarised study results. It also allows for filtering and selection of the results. The final outputs can be easily exported to different formats and used directly in the study reports and presentations.

The widely used graphing library – *Plotly* [5] is used for the creation of plots and diagrams in the application. It is highly customisable, allowing for various types of visualisations. The generated plots are interactive, with the ability to zoom, select and scale them. More detailed information can be obtained by hovering on different elements within the plot. A uniform design language was used for all visualisations, to achieve greater consistency and reproducibility.

We have divided the supported graphs into several types:

- Overview diagram - once an Excel file, containing tables, is uploaded to the application and the appropriate sheet is selected, the user is presented with a treemap graph, showcasing the hierarchical structure of the tables, with the different stratifications and sub-stratifications. The size of the fields is proportional to the count, allowing for a quick overview of the size of each stratification level.
- Numeric graphs - a separate tab, containing graphs for numeric variables presents Box and Whisker plots, chosen as the most informative type of plot able to be produced with summary statistics only. It allows for the comparison of the Minimum, Maximum, Mean, Median and 1$^{st}$ and 3$^{rd}$ quartiles of each stratification for a selected variable.
- Categorical graphs – another tab contains various plots, allowing for the visualisation of categorical variables. A various selection of bar and pie charts allows for the most appropriate graph to be selected in order to best showcase the underlying data.

## PACKAGE DEVELOPMENT

Best programming practices are used during the development of the presented tools:

- Git and GitLab – to ensure easy version tracking and strict quality control.
- GitLab pipelines – allow automated quick package releases.
- Docker + ShinyProxy – pipeline for automated application deployment on an internal server.
- Jira – issue tracking, feature requests, resource tracking.

## CASE EXAMPLE

We have successfully implemented the three-stage workflow for the Cancer Analysis System (CAS), a population-based database containing cancer registration and other linked health datasets collated and maintained by NHS England. We have extensive experience working with CAS, adding up to 5 years and several dozens of studies. This has allowed us to gain in-depth experience with the data source, understand the necessary steps for data cleaning and formatting, specific to CAS, automate and standardise them.

The strict patient confidentiality laws in the United Kingdom only allow for a small number of NHS England employees to access patient-level identifiable data [6]. Therefore, working with this data source requires the development of programming code without access to the real data. Due to the unavailability of data the programming code is instead developed using a simulated dataset – Simulacrum [7], which shares a similar structure and format to real data from CAS. Additionally, the running of programming code is performed only by NHS England staff in a tightly controlled environment. Any data released must not be able to be identified, therefore only summary statistics can be provided. These are further subject to primary and secondary suppression or rounding.

Our solution for the implementation of the automated workflow for CAS involves all three stages of our approach. First, we use IQVIA R CAS a CAS-specific package for Stage 1, which performs all functionalities related to data loading, formatting and cleaning. We then use the core IQVIA R Analysis package for Stage 2 of the workflow, which analyses the data and generates summary outputs. Finally, since we are unable to access the real data, we can use IQVIA R Visualisations for Stage 3, for any post-hoc figure creation, as required by the specific study.

## BENEFITS

Utilising a system of packages for programming studies based on RWD, compared with the prior bespoke approach offers several benefits:

- All functions need to be created and validated only once per package release. Until any updates are needed, they can be used without any changes. The programmer is guaranteed to receive the outputs as specified in the package and function documentation.
- The usage of validated functions and packages significantly reduces the possibility of programming errors during the analysis.
- Easier training for team members, new to programming or to our workflow. The extensive documentation provides explanations and examples of the most common use cases and difficulties encountered.
- The prior mentioned steps all amount to time savings. The amount of time saved based on our three-stage approach, compared to before its implementation varies between 40% for Stage 3 to 90% for Stage 1, with Stage 2 saving up to 80% of the time necessary for its programming.

## CONCLUSION

The growing need for RWE, stemming from RWD, has created a new set of challenges for statistical programmers. Compared to the structure and strict requirements that clinical trial data has, RWD contains a huge amount of variability. In order to create validated, reproducible results, we have devised a three-stage methodology, combining programming best practices with the knowledge we have gained in the pharmaceutical industry. This has allowed for a structured approach to the analysis of RWD, where each stage can be automated via pre-validated functions.

## REFERENCES

[1] Chang W, Cheng J, Allaire J, Sievert C, Schloerke B, Xie Y, Allen J, McPherson J, Dipert A, Borges B (2023). _shiny: Web Application.
Framework for R_. R package version 1.7.4.1, https://CRAN.R-project.org/package=shiny.
[2] Schad, F. and Thronicke, A. (2022) Real-world evidence-current developments and perspectives, International journal of environmental research and public health.
[3] Glur C (2023). _data.tree: General Purpose Hierarchical Data Structure_. R package version 1.1.0,
<https://CRAN.R-project.org/package=data.tree>.
[4] Dowle M, Srinivasan A (2023). _data.table: Extension of `data.frame`_. R package version 1.14.8,
<https://CRAN.R-project.org/package=data.table>.
[5] C. Sievert. Interactive Web-Based Data Visualization with R, plotly, and shiny. Chapman and Hall/CRC Florida, (2020).
[6] National Cancer Registration and Analysis Service (NCRAS). https://www.gov.uk/guidance/national-cancer-registration-and-analysis-service-ncras.
[7] HDI. Artificial patient-like cancer data to help researchers gain insights Simulacrum 2024.

https://simulacrum.healthdatainsight.org.uk/.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Author Name: Alexandrina Lambova
Company: IQVIA
Address: 103, "Aleksandar Stamboliyski" Blvd., Sofia Tower (Mall of Sofia), floor 7, 1164 Sofia, Bulgaria
Work Phone: N/A
Email: alexandrina.lambova2@iqvia.com
Website: http://www.iqvia.com/