Paper DH08

Data Managers and R programming, the unlikely pair

Anaëlle (Bonetta) Perretti, Idorsia Pharmaceuticals, Allschwil, Switzerland

ABSTRACT

Let's imagine stone age, Data Managers did not have any tool available to perform data review on extracted SAS datasets. We were laboriously using excel files and playing with the vlookup function. Data filters were manually created at each review round and precious time was lost.
At Idorsia, we have sought help from our Data Science team. They agreed to train us on the R programming language and introduced us to the tidyverse. Under their supervision, we are replacing old-fashioned excel work with R scripts to program monthly metrics, data comparisons, data cleaning cross-checks and to perform our oversight activities.
In this paper, we will discuss the experience of learning R for people with little or no programming background and share examples of how we are applying our new skills to transform the way we work with data.

INTRODUCTION

Data cleaning is the most important task of a Data Manager, working in house or outsourcing these activities to a CRO. Cleaning is done through several means; programmed edit checks, protocol deviations, external data reconciliation, or manual review of the eCRF (electronic case report form) pages to ensure compliance with the protocol. Programming is not always natural for a Data Manager and understanding of the programming languages comes with experience and participation to the development of edit checks specifications or external reconciliation checks.
For identification of data issues outside of database programming, Data Managers do rely a lot on cross-functional work with statisticians, Data Management programmers and Statistical programmers.
In the past years we were using Microsoft excel as tool to review and compare datasets with each other, perform cleaning cross-checks, provide a cleaning status of the clinical trial or to oversee the cleaning done by external vendors. However, excel is not designed to perform cross-checking of datasets, mainly because of issues in the formats of the data, and limitation of the vlookup function. These cross-checks were therefore time consuming and had to be repeated each time a new refresh of data was needed.
We have been asking for a tool to be able to perform our Data Management checks - something that could be handled by ourselves, easily customizable, reproducible, and that could be shared with our Data Management colleagues to achieve a standardized approach.
In our company, the Data Scientist team has been able to provide us with this required tool, they have trained us on RStudio programming. For several weeks we had two hours of in person trainings from the very basic approach until a quite elaborate approach which will be developed later in this paper.
Let me guide you through our Data Management journey into R programming applied to Data Management activities, and how this innovative approach has changed our daily work.

WHAT DO DATA MANAGERS DO ?

In order for you to understand our R for Data Management project development and application, let's first start by having a closer look at Data Managers tasks and activities in a clinical trial.
In a clinical trial everything starts with collection of the patient's data into the eCRF or clinical database. Electronic data entry of these patient information is done by the investigator or delegate thanks to the source documents and patient folders available on investigational site. Upon data entry investigators/sites receive some queries or data clarification forms.
These queries can be automatically generated by the electronic data capture system thanks to dedicated programming in the background, details of the programming are maintained in the edit checks specifications document developed by the Data Manager in collaboration with the programming team. Some queries can be manually generated by the Data Manager following manual review of the data or cross-checks done while conducting the data review on a subset of data.
All data entered in the eCRF are compiled into SAS datasets and exported on a regular basis and/or on study specific milestones in a raw data transfer. Depending on the setup of the trial some external data, such as PK, PD, biomarkers,

ECGs or labs, can be incorporated in the SAS datasets or at the level of the raw data transfer before being provided to the SDTM team for SDTM programming before delivery to the Statistical programming team.

Data Management team is in the centre of all the clinical data flow and have several tasks to manage in order to deliver a clean and compliant set of data to the Statistical team for analysis. Several tasks are to be completed such as managing discrepancies and issuing queries. Reponses provided by the sites will be reviewed and requeries will be done if data entered are still discrepant. Data Managers are also implicated in reviewing the protocol deviation (PD) outcomes, mainly the programmed protocol deviations being triggered in our case some of them are fully programmed, and no additional manual review is necessary. Cross functional review is usually done on a monthly basis or whenever necessary with the Protocol Deviation review team to discuss both programmed PDs and manual PD (i.e., PDs created by CRAs).

Coding review is also an important part of Data Managers' work, in Idorsia coding is performed either by the coder outsourced CRO or internally. In both cases, our expert coders are reviewing the coding listings provided by the CRO or produced internally and provide their comments on how the coding should be done.

Reconciliation of Serious Adverse events is done manually if data is not electronically collected in electronic data capture system, to make sure that the safety database information is matching 100% eCRF data.

The last reconciliation is done on external third-party vendor data such as ECGs, PK, PDs, biomarkers, these data are reconciled between external source and eCRF data. Programmed listings are being reviewed and queries are raised if discrepancies are found.

Last but not least, if Data Management is outsourced to a CRO, Sponsors are doing vendor oversight, defined in a vendor oversight plan and thus quality control is done on a monthly basis on a sample of queries performed by the CRO, or on a set of patients to verify that data review is done accurately by the CRO. This oversight can include any of the previous tasks mentioned as well as spot checks done when the Data Managers wants to conduct some accuracy checks on the data.

These activities are performed on an ongoing basis no matter the phase of a clinical study, setup, conduct, but of course with a threshold of 100% cleaning for close-out for interim or final database lock.

ONCE UPON A TIME IN DATA MANAGEMENT WORLD

As explained, cleaning activities implicate a lot of cross-checks on recurrent data. In this section of the paper we will focus on how things were done previously, and a couple of examples of manual tasks performed by Data Managers will be presented.

PREPARATION OF WEEKLY DATA MANAGEMENT STATUS

Cleaning activities performed by Data Management can somehow be measured by metrics extracted from the clinical database. For example, the number of queries currently open requiring an answer from site, number of missing pages or visits pending data entry, query age above a certain threshold, percentage of source data verification (SDV) to be done. These metrics are presented to the Clinical Trial Team or shared with other stakeholders upon request no matter the clinical trial status (conduct, deliverables, or database lock).

Preparation of these metrics requires generation of an excel listing called Clean Patient Tracker (CPT). This listing has been specified in conjunction with the sponsor and CRO and based on the needs of the clinical team. The CPT contains information about patients and sites. In order to summarise the relevant information a new summary table is then prepared and compiled in a joined Clinical Trial Team document, the executive summary, which contains cross-functional status in addition to the Data Management status from all the team members.

Compilation of the weekly Data Management status requires manual filtering on the relevant columns (Number of missing visits over 21 days, Number of missing eCRF pages for more than 21 days, Number of outstanding queries, Percentage of SDV completed) and copy pasting into the new joined table.

The exact same exercise needs be done with a different excel listing, Query overview, to produce the total number of queries on the trial. On our study we have two types of closed queries, both need to be summed to get the total number. The result is then also pasted into the joined executive summary table.

These different steps must be reproduced manually every week and cannot be automatically generated.

RESPONDING TO SPECIFIC QUESTIONS – MULTIPLE INFORMED CONSENTS ICFS EXAMPLE

While performing data review directly into the clinical database, some data inconsistencies could be flagged by Data Managers, programming, clinical or medical teams. In this case further investigation will be necessary to evaluate the impact on all patients.

In this example, our trial expectation is to have only one Informed Consent signed by patient, and we have identified a patient with two lines in the informed consent dataset, meaning that this patient has two informed consents reported in the electronic data capture system. This is not allowed by our clinical study protocol.

Informed consent SAS dataset is in this case extracted in excel format, and manual filtering is done via a pivot table to be able to assess how many patients have more than one informed consent reported. As an outcome of the investigation, we have identified four patients impacted by the multiple informed consent reporting. These patients will be queried accordingly in the clinical database. This type of check will either need to be done manually on a monthly basis or a new edit check will have to be programmed to automate the identification of the discrepancy.

REVOLUTION

In this section we will focus on the development of automated and reproductible R scripts to facilitate our daily work. We will start by focusing on the previous cases presented in previous section and we will see how these have been adapted into R scripts.
Considering all the different activities that Data Managers handle in a clinical trial cleaning workflow, in which cases R scripts could be developed? The answer is: to all previously introduced tasks and by facilitating the data review process. Once the programs have been designed, developed, and validated, we can use them as required and reproduce them as much as possible. Some examples are provided in the next sections of this paper.

BACKGROUND

Our innovative approach has been initiated by the need of a dedicated tool for Data Managers to grant us live access to clinical trial data. The prerequisites for this tool are an ability to perform a variety of checks and manipulate data for all cleaning activities in an autonomous way, to be able to cross-check many datasets and finally to be repeated to eliminate all manual parts of the work.
At Idorsia we have received an internal R training from our team of Data Scientists, consisting of two hours training per week for 10 weeks. The first part of the training focused on the basics of R programming and RStudio workbench, then moved on to inspecting a dataframe, joining, summarising, grouping, counting {dyplr}, then learning how to work with strings {stringr}, transposing data {tidyr}, and transposing and working with dates and datetimes {lubridate}. In the second part of the training we focused on base R, tibbles, vectors, data types, we also worked on how to handle missing data in a dataframe and different date formats. If interested we could also participate to advanced training sessions focusing on {gt}, {gtExtras}, {ggplot2} and quarto outputs.
In our department the strategy was to train all Data Managers, however use of the approach on each trial was not set as mandatory and in case of questions or need to further develop our R scripts we could reach out to our Data Science team to support us and help with the development of our ideas. A specific shared folder has been created in each trial to allow us to share the developed codes with each other, no matter if trials are internal or outsourced, and independently from the type of electronic data capture system used.
In the next paragraphs further details will be provided on how we adapted the previous examples via R programming.

WEEKLY DATA MANAGEMENT STATUS – EXAMPLE

In order to automatise the generation of the weekly Data Management status we mainly used {dpylr} and {gt} packages. CPT excel listing is received through daily data transfer from the CRO and uploaded in RStudio as a data frame. A new dataframe `Total_numbers_p` has been created with selection of the relevant data (Number of missing visits/records over 21 days, Number of missing eCRF pages for more than 21 days, Number of outstanding queries, Percentage of SDV completed) and filtered on the final total numbers.
Variables are then renamed, and table is displayed as a {gt} table.

```
#Preparing table with total numbers in metrics
Total_numbers_p <-
  CPT_p2 |>
  select(
    number_of_screened_patients,
    number_of_randomized_patients,
    number_of_clinical_event,
    number_of_self_injection_performed,
    number_of_missing_visits_records_over_21_days,
    number_of_missing_e_crf_pages_for_more_than_21_days,
    percentage_of_sdv_completed,
    number_of_open_queries_for_more_than_14_days,
    number_of_outsdanding_queries
  ) |> tail(1) |>
  rename(
```

```
    "Nbr screened"=number_of_screened_patients,
    "Nbr randomized"=number_of_randomized_patients,
    "Nbr Clinical Events"=number_of_clinical_event,
    "Nbr Injections"=number_of_self_injection_performed,
    "Missing visits > 21 days"=number_of_missing_visits_records_over_21_days,
    "Missing pages > 21 days"=number_of_missing_e_crf_pages_for_more_than_21_days,
    "% SDV"=percentage_of_sdv_completed,
    "Nbr Open queries > 14 days"=number_of_open_queries_for_more_than_14_days,
    "Nbr Outstanding queries"=number_of_outsdanding_queries
  ) |>
  mutate(Date = today()) |>
  select(Date,everything())

#Creating GT table
Total_numbers_p |> gt() |>
  tab_header(title = md("Data Management Metrics and Cleaning Status")) |>
  tab_source_note(source_note = paste("Refreshed on", Sys.Date())) |>
  tab_spanner(label = "Missing data", columns = starts_with("Missing")) |>
  tab_spanner(label = "Queries", columns = 9:10)
```

To finalize the last component of the status we used the same approach to sum the total number of closed queries. Query overview excel listing was used to create `Closed_sum_p2` and `Resolved_sum_p2` variables by selecting the maximum number of queries. These two were then summed to create `Total_p2`.
Table was then displayed in a {gt} table.

```
#Selecting maximum number of closed by DM queries
Closed_sum_p2 <-
  Closed_table_p2 |>
  select(closed_by_dm) |> max()
#Selecting maximum number of resolved by site queries
Resolved_sum_p2 <-
  Closed_table_p2 |>
  select(resolved_by_site) |> max()

#Calculating total number of closed queries
Total_p2 = Closed_sum_p2 + Resolved_sum_p2

#Creating total queries table with all variables
Total_queries_p2 <-
  tibble::tribble(~"Total  closed  by  DM",~"Total  resolved  by  site",~"Total  closed
queries",
                  Closed_sum_p2, Resolved_sum_p2,    Total_p2)
#Creating GT table
Total_queries_p2 |> gt() |>
  tab_header(title = md("Cumulative number of closed queries")) |>
  tab_source_note(source_note = paste("Refreshed on", Sys.Date())) |>
  tab_spanner(label = "Query category", columns = 1:2)
```

Refreshing the outputs of these two R scripts data can then be done on demand in a quick and efficient manner and are shared among all the Data Managers working on the trial.

MULTIPLE INFORMED CONSENTS – EXAMPLE

As mentioned previously it is important for Data Managers to be able to quickly identify data inconsistencies. In this example SAS datasets are read in R via a direct link between RStudio and internal shared folder and data is grouped and summarised to count the number of Informed consent lines for each patient.
Like all the other scripts, this one can be refreshed as needed and queries can then be raised for the correction of the discrepancies. If this issue is not an isolated case a new edit check will have to be programmed to automate the identification of the discrepancy into the clinical database.

```
#Grouping data by patient number, counting n consents
Count_ICF_p2 <- ICF_p2 |>
  group_by(subjid) |>
  summarise("Count of ICF entered" = n()) |>
  filter("Count of ICF entered">1) |>
  rename("Subject number"=subjid)
```

APPLICATIONS AND FUTURE PROJECTS

In summary, the use of R in daily Data Management activities has made a huge improvement. It has not only enabled us to gain time and accelerate our cleaning approach but is has also helped create data cross-checks required to review new datasets in a complete autonomous way.

In Idorsia, we currently use R to perform quality control of outsourced programming such as reprogramming a complex protocol deviation, flagging inconsistencies between Interactive Response Technology (IRT) system dates and time and clinical database screening dates and time. R is also a support during our monthly vendor oversight review, such as to identify a subset of queries to be reviewed for quality control checks.

As explained previously, we also use R on a weekly basis for metrics preparation and to generate summary metrics on demand, to be shared for instance with the sites.

On few Idorsia's trial we have recently implemented a risk-based quality monitoring approach to identify and track potential issues. The variables and calculation needed repeating each month and were complex to calculate manually. All the calculation steps have now been performed in R, and outputs are being provided to the clinical team in a quarto html format published on our Idorsia workbench RStudio aera.

So far one of the most complex tasks we have done is the preparation of an adjudication dashboard for an event driven trial. In order to be able to follow-up on a daily basis on the number of cases fully adjudicated and identify those with outstanding steps to be completed, we have developed a quarto output that is shared with the clinical team (see detailed code below).

Complexity of the programming comes from the data not being available in the same dataset, and from the need to create complex dataframes to prepare the data for the join function. Programming of the final table and preparation of all the intermediate steps require articulated data manipulations to create a complete dataset containing all the relevant information with the required level of detail.

We start by creating a cut-off date variable to be able to filter only selected events.

```
#creating the cut-off date
cut_off <- "2023-01-01"
```

Episode dataset is tidied to contain only the relevant columns, but also needs a preparation step to account for the partial dates. A conservative rule is put in place to impute XX date as 01 for days and JAN for the months. Correct format is then applied to the newly created variable `episode_dt` that is renamed as previous variable `MISTDAT`.

```
#including partial dates to create conservative approach for dates
episode_idmc <-
  episode |>
  select(COUNTRY, SITENR, SUBJID, CASEID, CECID, QC_CECID, SITENR, SUBJID, MISTDAT,
EVENTID) |>
  mutate(partial_dt = str_detect(MISTDAT,"X")) |>
  separate(MISTDAT, c("day","month", "year"), sep = "/", remove = FALSE) |>
  mutate(MISTDAT_dt = as.Date(MISTDAT,format = "%d/%m/%Y"),
         convention_dt = case_when(
           partial_dt == TRUE & day == "XX" & month == "XX" & year == "XXXX" ~
"01/01/1900",
           partial_dt == TRUE & day == "XX" & month != "XX" & year != "XXXX" ~
paste0("01/",month,"/",year),
           partial_dt == TRUE & day == "XX" & month == "XX" & year != "XXXX" ~
paste0("01/01/",year))
  ) |>
  mutate(
    episode_dt = case_when(
      partial_dt == FALSE ~ MISTDAT_dt,
```

```
      partial_dt == TRUE ~ as.Date(convention_dt,format = "%d/%m/%Y")))

episode_idmc <- episode_idmc |>
  select(COUNTRY, SITENR, SUBJID, CASEID, CECID, QC_CECID, SITENR, SUBJID, episode_dt,
EVENTID) |>
  rename(MISTDAT=episode_dt)
```

Dataset `all_adjevent` containing the final adjudication results is also prepared and joined with previous dataset. `episode_idmc`.

```
#adjudication dataset
all_adjevent <-
  adjevent |>
  select(CASEID, CECID, CECID_STATUS, CECID_STATUS_TXT, RECNO, ASSIGNED_TO) |>
  filter(RECNO == 0)

#join episode and adjevent
events_pending_idmc <-
  episode_idmc |> full_join(all_adjevent)
```

The third dataset `adj_time_det` containing all the details of the episodes not yet adjudicated needs to be prepared and joined with the dataset from the previous step `events_pending_idmc` to create `merge_all` dataset.

```
# report with breakdown details
adj_time <-
  adj_time_det |> select(episode_number, subject_nr, event_onset_date_t0,
date_event_first_reported_in_e_socdat_t1a, episode_status, event_type,
primary_reason_why_package_cannot_be_considered_as_complete) |>
  rename(CECID=episode_number)

merge_all <-
  events_pending_idmc |> left_join(adj_time)
```

Newly created `merge_all` dataset is filtered by cut-off date to create `adj_sum` dataset.

```
#Dataset filtered on cut-off
#filter join dataset (episode adjevent on cut-off)
adj_sum <-
  merge_all |>
  filter(MISTDAT<= as.Date(cut_off))
```

In the next steps, `status` variable will be created in order to classify events based on completion or not of the final adjudication steps.

```
full <-
  adj_sum |>
  rename(primary=primary_reason_why_package_cannot_be_considered_as_complete) |>
  mutate(status = case_when(CECID_STATUS_TXT == "                                    ."
~ "Adjudicated by Algorithm",
                            CECID_STATUS_TXT == "Completed" ~ "Adjudication by the
CEC",
                            CECID_STATUS_TXT == "Assigned (to CEC meeting)" ~
"Adjudication ongoing (CEC meeting)",
                            CECID_STATUS_TXT == "Assigned (to CEC members)" ~
"Adjudication ongoing (CEC members)",
                            CECID_STATUS_TXT == "Non-reconciled" ~ "Adjudication
pending reconciliation",
                            episode_status=="Created" & primary == "T03: Upload of SD
(Site)" ~ "2a. Upload of SD (Site)",
```

```
                            episode_status=="Created" & primary == "T04: Review of SD
(CRA)" ~ "2b. Review of SD (CRA)",
                            episode_status=="Created" & primary == "T05: Send
translation (SOCAR)" ~ "3a. Send translation (DM)",
                            episode_status=="Created" & primary == "T06: Receive
translation (Alcolad)" ~ "3b. Receive translation (Vendor)",
                            episode_status=="Created" & primary == "T08: Reply to
query (Site)" ~ "4. Reply to query (Site)",
                            episode_status=="Created" & primary == "T09a: Complete and
sign event-related forms (Site)" ~ "5. Complete and sign event-related forms (Site)",
                            episode_status=="Created" & primary == "T09b: 30-day fup
not yet elapsed" ~ "6a. 30-day fup not yet elapsed",
                            episode_status=="Created" & primary == "T09c: Complete and
sign 30-day fup (Site)" ~ "6b. Complete and sign 30-day fup (Site)",
                            episode_status=="Created" & primary == "T09d: SDV not
performed (CRA)" ~ "7. SDV not performed (CRA)",
                            episode_status=="Created" & primary == "T10: Package not
prepared (SOCAR)" ~ "8. Package not prepared (DM)",
                            episode_status=="Created" & is.na(primary) ~ "9. Package
sent to Adjudicators",
                            is.na(CECID_STATUS_TXT) & is.na(primary) ~ "1. Episode ID
to be created")) |>
  group_by(status) |>
  summarise(Count =n())
```

`stub` variable will be created to group the different statuses in preparation for the final status table to be displayed.

```
# Creating categories for summary table by status of adjudication
Adj_table <-
  full |>
  mutate(stub = case_when(status %in% c("Adjudicated by Algorithm","Adjudication by
the CEC") ~ "Completed",
                          status %in% c("Adjudication ongoing (CEC
meeting)","Adjudication ongoing (CEC members)") ~ "Ongoing",
                          status %in% c("1. Episode ID to be created",
                                        "2a. Upload of SD (Site)",
                                        "2b. Review of SD (CRA)",
                                        "3a. Send translation (DM)",
                                        "3b. Receive translation (Vendor)",
                                        "4. Reply to query (Site)",
                                        "5. Complete and sign event-related forms
(Site)",
                                        "6a. 30-day fup not yet elapsed",
                                        "6b. Complete and sign 30-day fup (Site)",
                                        "7. SDV not performed (CRA)",
                                        "8. Package not prepared (DM)",
                                        "9. Package sent to Adjudicators") ~ "Not
ready for adjudication",
                          status %in% c("Adjudication pending reconciliation") ~
"Reconciliation"))
```

Counting and formatting of the table will be done to trigger final {gt} `merge_table`.

```
# Counting occurrences of the different status
Adj_table_final <-
  Adj_table |>
  group_by(stub) |>
  mutate(total=sum(Count)) |>
  rename("Current status"=status, "Number of episodes"=Count) |>
  ungroup()
```

```
# Creating stub_order variable for future display of the final table
stub_order <- c("Completed","Ongoing","Reconciliation" , "Not ready for adjudication")

#Creating merge variable to organize final table by categories
merge_table <-
  Adj_table_final |>
  mutate(
    merge = paste(stub, "(N=",total,")"),
    stub = factor(stub, levels = stub_order )
  ) |>
  arrange(stub)|>
  select("Current status", "Number of episodes", merge)

# Display final gt table
merge_table |>
  gt(rowname_col = "", groupname_col = "merge") |>
  tab_header(title = "Adjudication summary status", paste("Refreshed on", Sys.Date()))
|>
  tab_options(
    row_group.as_column = TRUE)
```

In the adjudication dashboard .html we also incorporated an interactive {gt} table for the clinical team to be able to search for the different pending statuses and to follow up with the sites.
Through the {gt} `opt_interactive()` function we activated sorting of the column values, filters in the columns to look for a specific value, interactive resizing to optimize the display, highlight of the selected line in the table, and other options to simplify display of the table in the .html output.

```
#Interactive table for adjudication dashboard

adj_sum |> select(COUNTRY, SITENR, SUBJID, CECID, EVENTID, event_onset_date_t0,
episode_status, primary_reason_why_package_cannot_be_considered_as_complete) |>
  filter(!is.na(SUBJID)) |>
  rename(Country = COUNTRY, Site = SITENR, Subject = SUBJID, "Episode number"= CECID,
         "Episode ID" = EVENTID, "Event date"= event_onset_date_t0,
         "Episode status" = episode_status, "Package preparation status" =
primary_reason_why_package_cannot_be_considered_as_complete) |>
  gt() |>
  opt_interactive(
    use_sorting = TRUE,
    use_filters = TRUE,
    use_resizers = TRUE,
    use_highlight = TRUE,
    use_compact_mode = TRUE,
    use_page_size_select = TRUE,
    page_size_default = 5,
    page_size_values = c(5, 10, 25, 50, 100),
    pagination_type = c("numbers", "jump", "simple")
  ) |>
  sub_missing(
    columns = everything(),
    rows = everything(),
    missing_text = "Completed"
  )
```

We also included a calculation on the percentage of events adjudicated for any deliverable or for the overall event adjudication in order to access this information easily and clearly.
All the generated outputs are kept to track progression of the cleaning and output generation is automized on a weekly basis.

CONCLUSION

To conclude, this new approach has completely changed Idorsia DM way of working, of thinking, and is introducing a new mindset for Data Managers. In a couple of months and thanks to the extensive training sessions we had with our Data Scientist team, R programming world has been made accessible to people with little or no programming background. We are now able to create complex cross checks, tables, dashboards, quarto outputs, interactive tables, and to customize them as needed.

Some of us continue their learning experience and develop more and more complicated tables and cross-checks. This new approach is helping us to solve a lot of issues in a timely and reproducible manner. Programming has become a daily task in any of the cleaning activities we perform. In our company we have now a pool of Data Managers being able to program in R, which is clearly an asset. We are also lucky to be supported by our Data Scientist team, in every step of our programming experience.

Being a Data Manager and being able to program in R are two completely compatible and complementary aspects even if these are not natural. By practicing, curiosity, and creativeness we can now improve our skills and develop our competences.

In collaboration with our Data Scientist team, we are currently working to create a Shiny app to set up a Data Management dashboard containing all the needed cleaning details, such as number of overall or detailed queries, follow-up of source data verification and casebook signatures, with interactive customizable tables for clinical team, adjudication status and providing a complete status of the study.

The door is now open for automation, simplification of our Data Management tasks and evolution of Data Managers' working methods.

ACKNOWLEDGMENTS

I would like to take the opportunity to thank our Idorsia Data Scientist team for their patience and dedication during all these weeks of training. They have been available and always willing to help, without them no improvements could have been made. Thank you also for your support and time in the development of our Shiny app. I would like to thank our Head of Biometry, who made this project possible by making sure that our Data Managers could allocate the time needed for their personal trainings.

I would also like to thank our Trial Statisticians for their support and improvements ideas, but also for their help in the development of several R scripts, in particular with respect to adjudication and dashboards.

Last but not the least I would like to thank my two Trial Data Managers for their creativity and support during the development of our scripts.

CONTACT INFORMATION

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

Anaëlle (Bonetta) Perretti
Senior Project Data Manager
Idorsia Pharmaceuticals
Hegenheimermattweg 91
Allschwil / 4123
Work Phone: +41 58 844 10 83
Email: anaelle.perretti@idorsia.com
Web: www.idorsia.com