Transforming Survey Paradata

Laura Yoder, Andrew Piskorowski, and Mark Simonson
University of Michigan, Survey Research Center

1. Abstract

Paradata that are captured during the survey process are a valuable source of information in helping us understand and improve the data collection process.

Paradata which are linked directly to the administration of a survey instrument are collected automatically through the Blaise software (i.e., audit trail). The ADT file from Blaise 4 has been valuable in understanding interviewer behavior. With Blaise 5, we have been able to widen the collection of paradata to include behavior on web-SAQ (self-administered questionnaires) and/or mixed mode projects (i.e., interviewer and web-SAQ combined).

The main focus of this paper will be to share the results of a utility to automatically parse these sources of paradata from Blaise 4 and Blaise 5 into usable tables for analysis, reporting and quality control. The data from each version can be stored together and used in conjunction with other systems like time keeping, expense, survey data, and sample management. This paper will identify and demonstrate:

- Parsing the Blaise audit data (from 4.8 or 5.x versions) into relation tables (or CSV files)
- Examples of how to use these data for quality control, reporting purposes and as a backup copy of survey results
- Calculating useful paradata measures from these data (e.g., time on/time between page, last question seen/answered)
- Aggregating the data at various levels (e.g., page-level, session-level, respondent-level)

In addition, reporting tools such as SSRS (SQL Server Reporting Services), Excel, and Power BI are used to distribute the data to various user groups (e.g., PIs, production managers, statisticians, etc.). The resulting output is also available in a SQL database and can be accessed using other reporting or analysis tools. The transformation techniques and standard paradata, reporting can be implemented by any user of Blaise 5 paradata to enhance the use of these data.

2. Introduction

The UM Survey Research Center and other groups have previously written a number of papers and presentations about Blaise audit information and survey paradata more broadly. The focus of this paper will be a new tool developed by the University of Michigan Survey Research Center (UM-SRO Audit Parser +) that can parse both Blaise 4 and Blaise 5 formats and output the information into a standard relational database format that can be integrated into the broader survey system environment.

Although the format and information contained in Blaise 4 and 5 audit trail files is quite distinct, there is enough similarity that we have have been able to design a set of tables that can capture information from both systems. In general (but not always), Blaise 4 offers a subset of information available in Blaise 5. Therefore, some fields (columns) in the paradata tables may be empty for a Blaise 4 project. One major difference is the introduction of “Page” events to Blaise 5 - a page can have one or more fields displayed and has its own set of events. For Blaise 4 projects, there will be no Page level information.

Based on this information, the basic structure of our paradata information are as follows, with one to many relationships (parent to child) moving left to right:

Blaise 5 structure
Case level (CaseSummary table) → Audit Session(s) (ADTSession) → Audit Pages (ADTPage) → Audit Fields (ADTField)

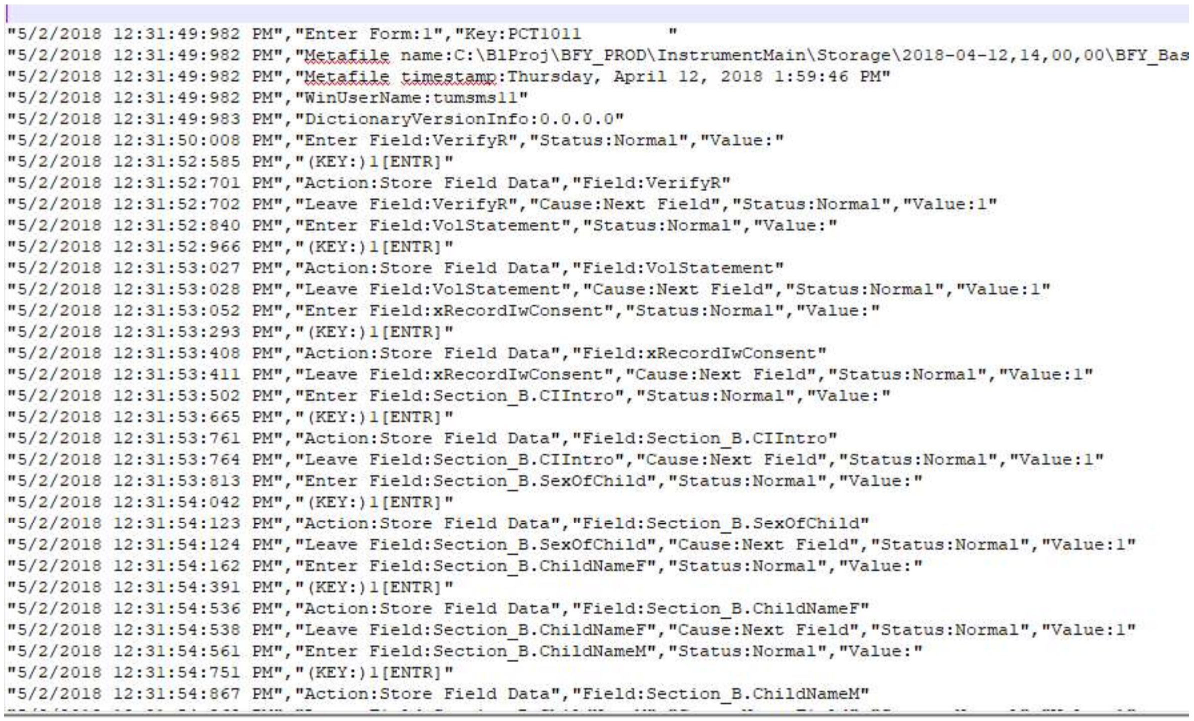
Blaise 4 structure
Case level (CaseSummary table) → Audit Session(s) (ADTSession) → Audit Fields (ADTField)

3. Blaise Paradata Native Formats - a brief overview

Blaise 4 stores audit information in a delimited text file and Blaise 5 uses SQLite or SQL Server format. Due to this difference, the parser program needs to be flexible enough to read and write to a variety of formats. In addition, while there is a structure to the information in the Blaise audit files, effective parsing requires code that can easily manipulate and extract string information.

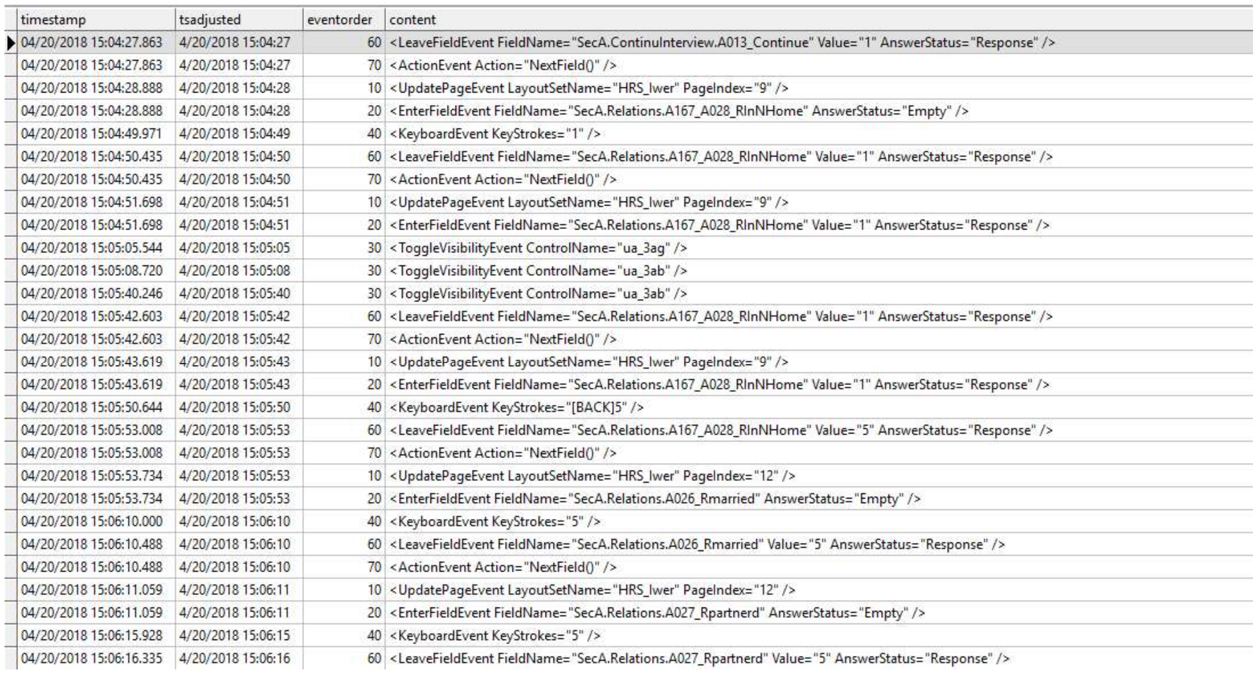
Below are small snippets of raw audit data used to provide context make apparent why a parser application is needed.

Table 1. Blaise 4 audit data



```
"5/2/2018 12:31:49:982 PM","Enter Form:1","Key:PCT1011        "
"5/2/2018 12:31:49:982 PM","Metafile name:C:\BlProj\BFY_PROD\InstrumentMain\Storage\2018-04-12,14,00,00\BFY_Bas
"5/2/2018 12:31:49:982 PM","Metafile timestamp:Thursday, April 12, 2018 1:59:46 PM"
"5/2/2018 12:31:49:982 PM","WinUserName:tumsmsl1"
"5/2/2018 12:31:49:983 PM","DictionaryVersionInfo:0.0.0.0"
"5/2/2018 12:31:50:008 PM","Enter Field:VerifyR","Status:Normal","Value:"
"5/2/2018 12:31:52:585 PM","(KEY:)1[ENTR]"
"5/2/2018 12:31:52:701 PM","Action:Store Field Data","Field:VerifyR"
"5/2/2018 12:31:52:702 PM","Leave Field:VerifyR","Cause:Next Field","Status:Normal","Value:1"
"5/2/2018 12:31:52:840 PM","Enter Field:VolStatement","Status:Normal","Value:"
"5/2/2018 12:31:52:966 PM","(KEY:)1[ENTR]"
"5/2/2018 12:31:53:027 PM","Action:Store Field Data","Field:VolStatement"
"5/2/2018 12:31:53:028 PM","Leave Field:VolStatement","Cause:Next Field","Status:Normal","Value:1"
"5/2/2018 12:31:53:052 PM","Enter Field:xRecordIwConsent","Status:Normal","Value:"
"5/2/2018 12:31:53:293 PM","(KEY:)1[ENTR]"
"5/2/2018 12:31:53:408 PM","Action:Store Field Data","Field:xRecordIwConsent"
"5/2/2018 12:31:53:411 PM","Leave Field:xRecordIwConsent","Cause:Next Field","Status:Normal","Value:1"
"5/2/2018 12:31:53:502 PM","Enter Field:Section_B.CIIntro","Status:Normal","Value:"
"5/2/2018 12:31:53:665 PM","(KEY:)1[ENTR]"
"5/2/2018 12:31:53:761 PM","Action:Store Field Data","Field:Section_B.CIIntro"
"5/2/2018 12:31:53:764 PM","Leave Field:Section_B.CIIntro","Cause:Next Field","Status:Normal","Value:1"
"5/2/2018 12:31:53:813 PM","Enter Field:Section_B.SexOfChild","Status:Normal","Value:"
"5/2/2018 12:31:54:042 PM","(KEY:)1[ENTR]"
"5/2/2018 12:31:54:123 PM","Action:Store Field Data","Field:Section_B.SexOfChild"
"5/2/2018 12:31:54:124 PM","Leave Field:Section_B.SexOfChild","Cause:Next Field","Status:Normal","Value:1"
"5/2/2018 12:31:54:162 PM","Enter Field:Section_B.ChildNameF","Status:Normal","Value:"
"5/2/2018 12:31:54:391 PM","(KEY:)1[ENTR]"
"5/2/2018 12:31:54:536 PM","Action:Store Field Data","Field:Section_B.ChildNameF"
"5/2/2018 12:31:54:538 PM","Leave Field:Section_B.ChildNameF","Cause:Next Field","Status:Normal","Value:1"
"5/2/2018 12:31:54:561 PM","Enter Field:Section_B.ChildNameM","Status:Normal","Value:"
"5/2/2018 12:31:54:751 PM","(KEY:)1[ENTR]"
"5/2/2018 12:31:54:867 PM","Action:Store Field Data","Field:Section_B.ChildNameM"
```

Table 2. Blaise 5 audit data

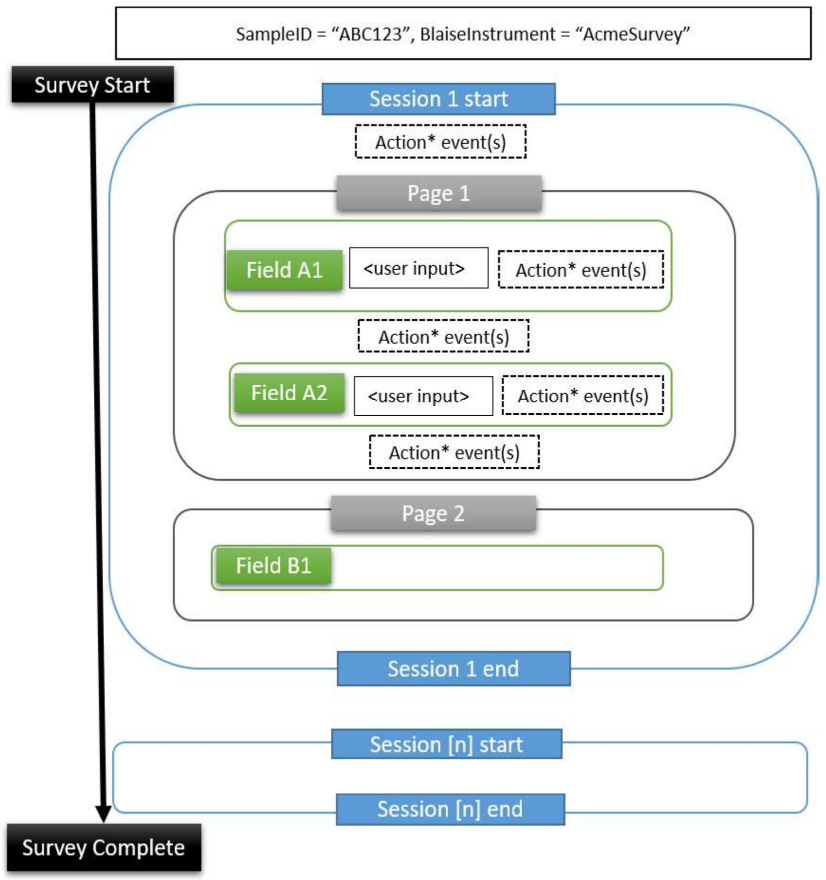


timestamp	tsadjusted	eventorder	content
04/20/2018 15:04:27.863	4/20/2018 15:04:27	60	<LeaveFieldEvent FieldName="SecA.ContinuInterview.A013_Continue" Value="1" AnswerStatus="Response" />
04/20/2018 15:04:27.863	4/20/2018 15:04:27	70	<ActionEvent Action="NextField()" />
04/20/2018 15:04:28.888	4/20/2018 15:04:28	10	<UpdatePageEvent LayoutSetName="HRS_Iwer" PageIndex="9" />
04/20/2018 15:04:28.888	4/20/2018 15:04:28	20	<EnterFieldEvent FieldName="SecA.Relations.A167_A028_RInNHome" AnswerStatus="Empty" />
04/20/2018 15:04:49.971	4/20/2018 15:04:49	40	<KeyboardEvent KeyStrokes="1" />
04/20/2018 15:04:50.435	4/20/2018 15:04:50	60	<LeaveFieldEvent FieldName="SecA.Relations.A167_A028_RInNHome" Value="1" AnswerStatus="Response" />
04/20/2018 15:04:50.435	4/20/2018 15:04:50	70	<ActionEvent Action="NextField()" />
04/20/2018 15:04:51.698	4/20/2018 15:04:51	10	<UpdatePageEvent LayoutSetName="HRS_Iwer" PageIndex="9" />
04/20/2018 15:04:51.698	4/20/2018 15:04:51	20	<EnterFieldEvent FieldName="SecA.Relations.A167_A028_RInNHome" Value="1" AnswerStatus="Response" />
04/20/2018 15:05:05.544	4/20/2018 15:05:05	30	<ToggleVisibilityEvent ControlName="ua_3ag" />
04/20/2018 15:05:08.720	4/20/2018 15:05:08	30	<ToggleVisibilityEvent ControlName="ua_3ab" />
04/20/2018 15:05:40.246	4/20/2018 15:05:40	30	<ToggleVisibilityEvent ControlName="ua_3ab" />
04/20/2018 15:05:42.603	4/20/2018 15:05:42	60	<LeaveFieldEvent FieldName="SecA.Relations.A167_A028_RInNHome" Value="1" AnswerStatus="Response" />
04/20/2018 15:05:42.603	4/20/2018 15:05:42	70	<ActionEvent Action="NextField()" />
04/20/2018 15:05:43.619	4/20/2018 15:05:43	10	<UpdatePageEvent LayoutSetName="HRS_Iwer" PageIndex="9" />
04/20/2018 15:05:43.619	4/20/2018 15:05:43	20	<EnterFieldEvent FieldName="SecA.Relations.A167_A028_RInNHome" Value="1" AnswerStatus="Response" />
04/20/2018 15:05:50.644	4/20/2018 15:05:50	40	<KeyboardEvent KeyStrokes="[BACK]5" />
04/20/2018 15:05:53.008	4/20/2018 15:05:53	60	<LeaveFieldEvent FieldName="SecA.Relations.A167_A028_RInNHome" Value="5" AnswerStatus="Response" />
04/20/2018 15:05:53.008	4/20/2018 15:05:53	70	<ActionEvent Action="NextField()" />
04/20/2018 15:05:53.734	4/20/2018 15:05:53	10	<UpdatePageEvent LayoutSetName="HRS_Iwer" PageIndex="12" />
04/20/2018 15:05:53.734	4/20/2018 15:05:53	20	<EnterFieldEvent FieldName="SecA.Relations.A026_Rmarried" AnswerStatus="Empty" />
04/20/2018 15:06:10.000	4/20/2018 15:06:10	40	<KeyboardEvent KeyStrokes="5" />
04/20/2018 15:06:10.488	4/20/2018 15:06:10	60	<LeaveFieldEvent FieldName="SecA.Relations.A026_Rmarried" Value="5" AnswerStatus="Response" />
04/20/2018 15:06:10.488	4/20/2018 15:06:10	70	<ActionEvent Action="NextField()" />
04/20/2018 15:06:11.059	4/20/2018 15:06:11	10	<UpdatePageEvent LayoutSetName="HRS_Iwer" PageIndex="12" />
04/20/2018 15:06:11.059	4/20/2018 15:06:11	20	<EnterFieldEvent FieldName="SecA.Relations.A027_Rpartnerd" AnswerStatus="Empty" />
04/20/2018 15:06:15.928	4/20/2018 15:06:15	40	<KeyboardEvent KeyStrokes="5" />
04/20/2018 15:06:16.335	4/20/2018 15:06:16	60	<LeaveFieldEvent FieldName="SecA.Relations.A027_Rpartnerd" Value="5" AnswerStatus="Response" />

While the audit data contains a wealth of detail, it looks a bit messy and is not easily be used "as is". If only the information were structured in a manner we could use for reporting and other purposes!

So, rather than go into detail about the native formats (that is what the Parser Application is for, after all), we will outline the structure and key components that are being parsed and how these relate to the output data that is stored in relational tables.

Audit data structure conceptualized



One key takeaway from the figure above is that there are **one or more sessions for each respondent survey**. The audit data provides information about the session itself at the start and end of the session, such as timestamps, device used, browser type and whether the session was suspended or the survey was completed (except when the session is closed unexpectedly).

Once a session starts (assuming it is not shut down immediately), typically a page is displayed and then a field or set of fields within the page are displayed (in Blaise 4, there is no "page" event, but the field concept is similar). The audit information captures the page-level information and then the first field that is ready for input (where the input cursor resides). If the user enters some data (via keystrokes or selecting from a control like a radio button or drop-down), the input information is stored. Once a user moves to the next field or page, additional information is captured.

In the diagram, we have a reference to "**Action* events**". Note that Blaise 5 captures most everything as some type of "event", and within this event you have various key-value pairs. For example: KeyboardEvent KeyStrokes="1" is a keyboard event with a key-value pair of keystrokes="1". There also something called an "ActionEvent" - for example, ActionEvent Action="NextField()". The Action events are quite common and our output database has columns to generically capture these events. But there are many other possible events (for example, GoToUriEvent, ToggleVisibilityEvent, SuppressSignalEvent, etc.). For simplicity, the diagram uses "Action*" as a way to refer to both Action and all these other type of events that only occur in certain circumstances.

4. The Audit Parser Application and Output Database Structure

To combat the proliferation of parsing tools and the resulting difficulties in consistently using the paradata information for reports and applications, UM SRC decided to build a generic parser application that can process all the different types of Blaise audit formats and output to a standard set of relational database tables. The result is that instead of focusing resources on the different parsing routines and understanding the unique set of information created by these routines, the focus would shift to understanding and querying a standard relational database created by the parser application.

4.1 How it works

The UM-SRO Audit Parser application was designed to do the following:

- Understand both Blaise 4.8 and Blaise 5 audit trail formats.
- Be able to read and write to and from text based delimited files, SQL Server tables and SQL Anywhere tables.
- Allow a "manual" mode, where a user selects a case to parse and can view the results.
- Allow a "batch" mode, where the program runs on a scheduled basis and processes multiple cases.
- Do all this with a minimum number of application dependencies, so the program can run on a user's PC or on a Server.

In addition, the application supports two basic approaches to finding and parsing audit information. One approach is to leverage Survey Management System (SMS) information about case status and use this information to find and process the Blaise audit trail data. The other approach is "self discovery" - based on an instrument id and file/database location, process all audit data that are found. In both cases, we need a mechanism to track what has already been processed and what still needs to be processed.

While the "parsing" part of the application is somewhat complex, the key to this application is really the output data model. There are trade-offs between capturing all the detail provided by the audit data and having a database that is easy to use, efficient and can be queried by various stakeholder groups, applications and reporting systems.

The resulting database model (implemented on SQL Server but generic enough for most any relational database system) mirrors the basic structure of the audit data itself. There are the
four tables make up the core of the data model.

Study Paradata Database tables (output from Parser application)

- **CaseSummary** - sample line level information with **one record for each case** for a given Blaise survey instrument (within the audit data the case id is called “key” or “KeyValue”, in CaseSummary it is “SampleId”).

- **ADTSession** - Blaise audit data session level data for each case. There is at least one session for each survey. The session has a unique number for each case which stored in the “BlaiseSession” column.

- **ADTPage** - Blaise audit data information at the “Page” level (Blaise 5 only). For Blaise 4, there will not be a record in this table.

- **ADTField** - Blaise audit data at the field level for each case. This is the most complex set of information. While some information (e.g., navigation action) falls outside the field event itself, in our model we decided to collapse everything between the “Enter Field” event and the next field’s “Enter Field” event into **one record**.

The tables are “relational”, that is, there is a **parent-child relationship** between them that allows you to use SQL code to join information into a single result set. The tables are ordered above to reflect the one-to-many relationship, from CaseSummary to ADTSession (for each case, one or more sessions), to ADTPage (for each session, one or more pages) to ADTField (for each page, one or more fields).

Audit data structure relationship to output database tables

In this section, we'll discuss how raw audit information is parsed and transformed into data that are part of a relational database. The relational database is the creation of UM SRC. It is modeled on the Blaise audit data and some naming conventions are similar but, to be clear, the tables and field names used are UM SRC conventions.

The structure of the current set of tables comes from common elements found in various other audit data tables created previously at UM SRC, an analysis of the Blaise 5 audit structure, and the common reporting and use case needs we have identified.

We also note that because Blaise uses "FieldName" to reference the unique question item name, this may cause a little confusion at times. Therefore, we will reference **database** "**fields**" as "**columns**" or "**column names**". These table names and column names that are part of the output tables are usually referenced in the form of **tableName.columnName.**

So let's take our most potentially confusing concept, the Blaise "FieldName", and try to sort out how we store it in the database as an example. The Blaise "fieldname" value from the audit trail (for our example, let's say we have a field named "Gender") is parsed and stored in a table called ADTField. The column name where it is stored is also called "fieldname". The resulting table reference is **ADTField.FieldName** and for this specific value, "Gender", we can make the notation **ADTField.FieldName = "Gender"**.

4.2 General Information About the Survey

The most general information about a given case and survey are stored at the Case-level and Session-level. Here we summarize some of the key information that is available.

4.2.1 Case Level

The **CaseSummary** table contains essentially **three types of information** in addition to the SampleId for the respondent:

1. Audit data processing and summary information.
2. Sample Management System (SMS) information.
3. Interviewer Quality Control (QC) information.

The parser can work **without** any SMS or Interviewer information, but we have added these for additional integration capabilities and QC reporting.

Case-level identity information

CaseSummary.SampleId: The respondent identifier, usually named "Key" or "KeyValue" in the Audit data.
CaseSummary.InstrumentName: The Blaise instrument name for the survey.
CaseSummary.InstrumentId: The Blaise InstrumentId GUID key (Blaise 5).

Case-level Audit data information

CaseSummary.ADTisComplete: True if the survey is complete based on the audit data information.
CaseSummary.ADTSessions: Number of sessions in the Audit data.
CaseSummary.ADTCompleteDT: Timestamp of when survey was completed based on the audit data.
CaseSummary.ADTLanguage: Primary language of the survey based on Audit data information.

SMS source data

CaseSummary.SMSisComplete: True if the survey is complete based on the SMS information.
CaseSummary.SMSSessions: Number of sessions based on the SMS.
CaseSummary.SMSCompleteDT: Timestamp of when survey was completed based on the SMS.
CaseSummary.SMSLanguage: Primary language of the survey based on the SMS.

Additional audit summary information (which comes from Sessions or Fields) has also been added to the CaseSummary table to allow a quick query of only this table (no joins needed) to get often requested information.

Audit summary information

CaseSummary.ADTSurveyTime: Total time (in minutes) for a survey based on the audit data (will equal the sum of all the session duration times).
CaseSummary.SurveyQVisited: Number of **unique** questions (fields) visited by the respondent (each field name is only counted once).
CaseSummary.SurveyQAnswered: Number of questions with an answer value provided (based on the last visit to the field).

Interviewer quality control information is also available for some projects. At UM, this information feeds into the Interviewer QC application.

Interviewer QC information

CaseSummary.SMSIwer / CaseSummary.ADTIwer: Interviewer id from the SMS system and/or audit data (when available).
CaseSummary.InterviewOrder: For a given interviewer, what is the order of the interview (based on completed date and time).
CaseSummary.MediaFiles: Number of media files found for this interview (e.g., video recordings or audio recordings).

4.2.2 Session Level

The **ADTSession** table contains session level information for each case based on the Blaise audit data session records. At minimum, a survey must have at least one session. If a survey is restarted, there may be more than one session. The session has an indicator (“completed”) to indicate if the survey has been completed in the given session.

Because Blaise 5 has multimode capability, the session information also helps us determine the mode of the interview, in addition to the type of device being used, the device height and width and other information.

Key Session level columns

ADTSession.BlaiseSession: Session number (unique for each SampleId).
ADTSession.Completed: True if the survey is completed in this session (last session).
ADTSession.DeviceType: Type of device derived from information in the session (PC, Tablet, SmartPhone).
ADTSession.Mode: Mode (CATI, Web, CAPI) used for session data. For a mixed mode project, sessions may have different modes. Currently, the parser derives the mode from various other fields.
ADTSession.LayoutSetName: Name of the page LayoutSetName (from the first page of the session).
ADTSession.OSName: Operating System Name (e.g., Windows, Mac, Android, etc. which is derived from the audit OS string).
ADTSession.Browser: Name of browser, if applicable.
ADTSession.DisplayWidth: User device display width .
ADTSession.DisplayHeight: User device display height.

ADTSession.LastField: Name of the last field entered in the session (parser stores last fieldname found for the session in this column).
ADTSession.LastFieldAnswered: Name of the last field with an answer value in the session (parser finds this information and stores it in this session column).

4.3 Timings

One of the most important pieces of information the audit data provides are timestamps at various levels of detail which allow us to calculate timings (durations). This timestamp information and duration information (usually in minutes) is parsed, calculated, and stored in the various tables.

At the highest level, we have Session timing information. Each session has a begin and end timestamp and we calculate the Session duration as the difference between these. If a survey only has one session, the Session duration equals the total survey time. For multiple sessions, however, the total survey time is the sum of the Session durations.

Here is a summary of **Session and Page level timing** information. There are timestamp fields available for Session and Page, but the calculated Duration time (done by the Parser application based on the timestamp fields) is usually what is needed for reporting.

Case, Session and Page level duration fields

CaseSummary.ADTSurveyTime: Total survey time in minutes (calculated by Parser from Session durations).

ADTSession.SessionDuration: Time (in minutes) spent in a given session.
ADTSession.CumulativeDuration: Cumulative time (in minutes) for a given session and sum of previous sessions.

ADTPage.PageDuration: Time (in minutes) spent on a given page. The parser calculates this using the audit timestamp associated with a given page UpdatePageEvent and then uses the **next** page UpdatePageEvent timestamp as the end time.

Note that the final session ADTSession.CumulativeDuration will equal CaseSummary.ADTSurveyTime.

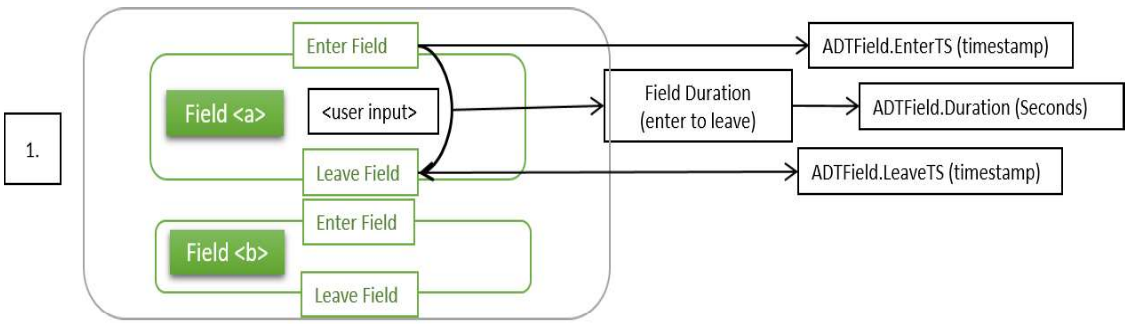
Field level timings can get a little more complex. This section describes some different types of timing information available at the field level.

Key timing columns at the Blaise field level

ADTField.EnterTS: Timestamp associated with the Enter Field event (when the cursor moves into the field).
ADTField.LeaveTS: Timestamp associated with the Leave Field event (when the cursor moves out of the field).
ADTField.KeystrokeTS: Timestamp at beginning of a keystroke entry for the field.
ADTField.NextFieldBegTS: The EnterTS of the **next** field (if it exists).

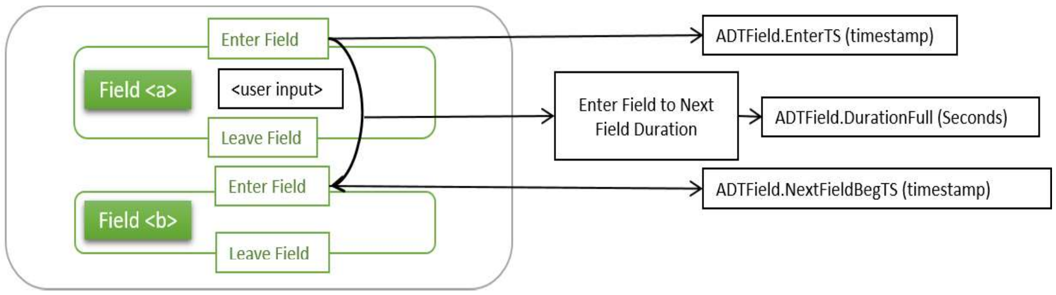
ADTField.Duration: Duration in seconds between Enter and Leave.
ADTField.DurationFull: Duration in seconds between Enter and the next Field Enter.

1. The duration between Enter and Leave field timestamps is a measure that only calculates the time within the field. Once the cursor leaves the field, the timing stops. Note that the sum of all these timings will likely be **less** than the total time for the page.


1.
Enter Field
Field <a>
<user input>
Leave Field
Enter Field
Field <b>
Leave Field
ADTField.EnterTS (timestamp)
Field Duration (enter to leave)
ADTField.Duration (Seconds)
ADTField.LeaveTS (timestamp)

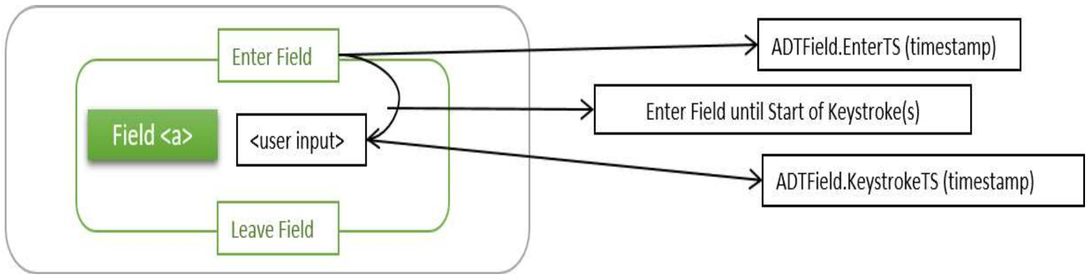
2. The duration between Enter event for a given field and the Enter for the **next** field captures the entire time on the field plus any events that may occur **after** the leave field event. For example, some complex conditional logic may have to execute or there can be screen displays that require a little extra time to render. We have found this is the **most consistent measure of "field" time** (as some versions of Blaise or modes do not always have accurate Leave Field times currently). Note that measure 1 should always be **less than or equal to** this measure.

2.


Enter Field
Field <a>
<user input>
Leave Field
Enter Field
Field <b>
Leave Field
ADTField.EnterTS (timestamp)
Enter Field to Next Field Duration
ADTField.DurationFull (Seconds)
ADTField.NextFieldBegTS (timestamp)

3. The time between EnterTS and KeystrokeTS is typically the time it takes to **read the item before starting to answer**. This information could be combined with length of the question text (not in the audit data) to get an idea of interviewer speaking speed or respondent reading speed (web survey).

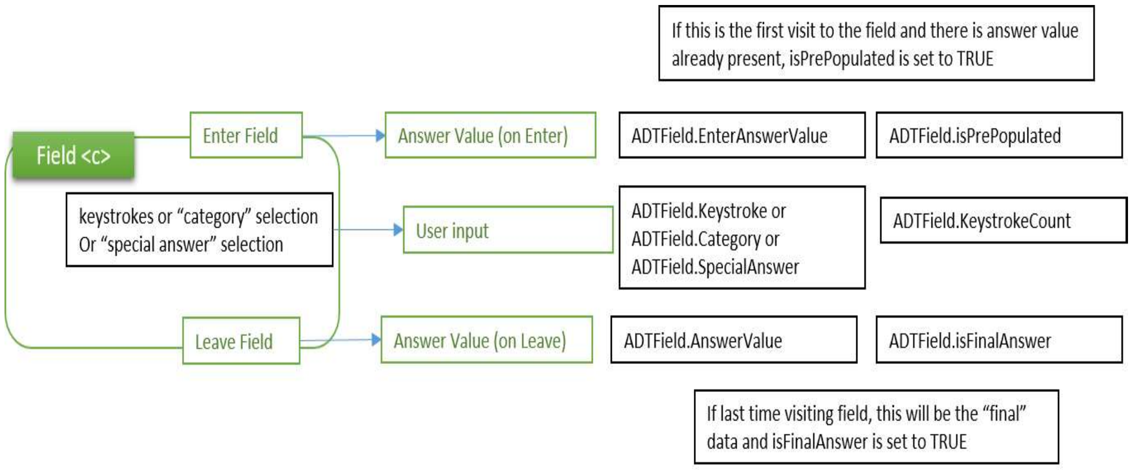
3.


Enter Field
Field <a>
<user input>
Leave Field
ADTField.EnterTS (timestamp)
Enter Field until Start of Keystroke(s)
ADTField.KeystrokeTS (timestamp)

4.4 Answering the Questions

Depending on the type of question, there is typically **keystroke** information available or what Blaise terms "**category**" information (selections from a drop-down list or radio button). There are also "**special answers**" entered via special "hot" keys or menu items, such as Ctrl-D for "Don't Know".

In addition to whatever action is taken to indicate an answer, the audit trail also stores any answer information that is already in the field before the user edits the data. The information is stored in "ADTField.**EnterAnswerValue**" and may be there because of a "preload" data or because the question is being revisited and was previously answered.


If this is the first visit to the field and there is answer value already present, isPrePopulated is set to TRUE
Field <c>
Enter Field
Answer Value (on Enter)
ADTField.EnterAnswerValue
ADTField.isPrePopulated
keystrokes or "category" selection Or "special answer" selection
User input
ADTField.Keystroke or ADTField.Category or ADTField.SpecialAnswer
ADTField.KeystrokeCount
Leave Field
Answer Value (on Leave)
ADTField.AnswerValue
ADTField.isFinalAnswer
If last time visiting field, this will be the "final" data and isFinalAnswer is set to TRUE

After the user is done editing the answer, the “final” data are stored in the column ADTField.AnswerValue, unless they fall into the category of a Special Answer, in which case they are stored in the column ADTField.SpecialAnswer (as in the example above, Ctrl-D will result in ADTField.SpecialAnswer = “Don’t Know”).

Because **a field can be visited more than once**, you need to know the last time a field was edited to get the final answer that will be in recorded in the main survey database. For this, we add a flag in the column ADTField.**isFinalAnswer**. By querying only the fields marked as ADTField.isFinalAnswer = True you can obtain the actual survey data that should match what is in the Blaise data file.

A summary of fields related to **answering a question item**

ADTField.EnterAnswerValue: Any data associated with the field **before** user input.
ADTField.**IsPrepopulated**: True if this is the first visit to the field and EnterAnswerValue is not empty.

ADTField.**Keystroke**: Keystroke values entered by the user (if applicable).
ADTField.**KeystrokeCount**: The number of keystroke values typed from ADTField.Keystroke as calculated by the Parser.
ADTField.Category: Category values selected (radio or drop-down type question), stored like “5” or “5-1-2”.

ADTField.AnswerValue: The actual answer value for the question when the user leaves the field.
ADTField.SpecialAnswer: For “Special Answer” type of responses (like using Ctrl-D to select “Don’t Know”), the answer value is stored here.
ADTField.isFinalAnswer: True if this is the **last** ADTField.AnswerValue for a given field (final data for a field). This is calculated by the Parser application.

4.4.1 Additional Information About “Keystroke” and “Category” Data

The **keystroke** field captures all the keystrokes recorded by the audit data. Keystroke information may be as simple as:

ADTField.Keystroke = “1”

Or look more like:

ADTField.Keystroke = “R says DK because of A[BACK]memoryt[BACK]”

The column “**KeyStrokeCount**” sums up the number of keystrokes for this field (counting special keys like “[BACK]” as one keystroke. For the second example above, it will be:

ADTField.KeyStrokeCount = 31

The column “**AnswerValue**” is what Blaise records as the answer for this question. For the second example above:

ADTField.AnswerValue = “R says DK because of memory”

For questions that already have data (either because the respondent returned to this question or it was “preloaded”), the information is stored in “**EnterAnswerValue**”. If there is an EnterAnswerValue and the corresponding AnswerValue is different, it indicates the value was changed.

The column “**Category**” is like Keystroke, except it stores the selections made from a control. Multiple selections are separated by a hyphen (-), so selecting answers 1 and 2 looks like this:

ADTField.Category: 1-2

If the question allows **multiple responses**, the AnswerValue would look like this:

ADTField.AnswerValue: 1-2

If only a **single selection** is allowed, the last selection is what will be the answer:

ADTField.AnswerValue: 2

4.5 User Navigation

Capturing how the user navigates through the survey is one of the unique things available in the audit trail information. It is important to understand that there is typically both **forward and reverse navigation** possible. So, a given page (identified with a Blaise PageIndex number) and a given field (identified with a Blaise FieldName) can be visited one or more time.

User navigation columns (all calculated or derived by the Parser application)

ADTPage.UserPageOrder: Order of pages as navigated by the user. Is unique across the entire survey.
ADTPage.PageVisitNumber: Visit number for same page. (If same page visited twice, 2nd time will have a value of 2).

ADTField.UserFieldOrder: Order of fields as navigated by the user. Is unique across the entire survey.
ADTField.FieldVisitNumber: Visit number for same field. (If same field visited twice, 2nd time will have a value of 2).
ADTField.LeaveFieldNav: A value of 1 indicates next field and 2 indicates next page, -1 indicates previous field and -2 indicates previous page.

In the database, we capture the overall order of navigation through **pages** with **ADTPage.UserPageOrder** and through **fields** with **ADTField.UserFieldOrder**. UserPageOrder and UserFieldOrder continue across sessions and are unique for a respondent. By ordering results by UserPageOrder and UserFieldOrder, the exact sequence of events for a survey can be tracked.

Note that ADTPage.PageIndex and ADTField.FieldName, however, may **not be unique**. In the tables, we have added **ADTPage.PageVisitNumber** and **ADTField.FieldVisitNumber** to indicate the visit number. Every page/field in the audit data will have a VisitNumber = 1. If the page/field is visited again, the parser increments the VisitNumber and saves another record in the database tables.

The direction of navigation after leaving a field is captured in **ADTField.LeaveFieldNav**. A positive number indicates forward navigation and a negative number indicates backward navigation. A value of 1 indicates next field and 2 indicates next page, -1 indicates previous field and -2 indicates previous page.

Basic example of how the database tracks a user navigating through survey fields and answering questions

Using these concepts, here is a basic example. For the example, we will use two questions with Blaise FieldNames of “Gender” and “Age”. Let’s say Gender is selected from a pick list (Female = 1 and Male = 2) and Age is entered as a number.

The user will go to Gender and select the answer for Female (1), go the the Age question and enter “25”, and then return to the Gender item and change the answer to Male (2). How does this look in our Paradata field table (ADTField)?

The record in ADTField for the first visit to Gender looks something like this:

> ADTField.UserFieldOrder = 1
> ADTField.Fieldname = “Gender”
> ADTField.FieldVisitNumber = **1**,
> ADTField.Category=”1”
> ADTField.AnswerValue=”1”
> ADTField.isFinalAnswer = **False**
> ADTField.LeaveFieldNav = 1

The record in ADTField for the next field, Age, where the user enters and answer and then returns to the previous field (Gender) looks something like this:

> ADTField.UserFieldOrder = 2
> ADTField.Fieldname = “Age”
> ADTField.FieldVisitNumber = 1,
> ADTField.Keystroke=’26[Back]5”
> ADTField.AnswerValue=”25”
> ADTField.isFinalAnswer = True
> ADTField.LeaveFieldNav = **-1**

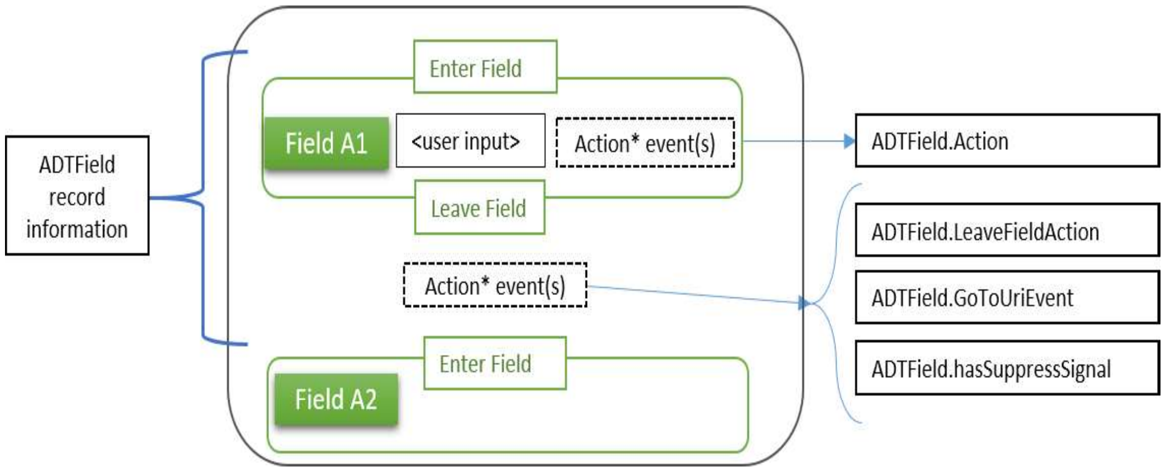
And, finally, returning to Gender for the final time, changing the answer and navigating to the next page:

> ADTField.UserFieldOrder = 3
> ADTField.Fieldname = “Gender”
> ADTField.FieldVisitNumber = **2**,
> ADTField.InitialAnswerValue=”1”
> ADTField.Category = “2”
> ADTField.AnswerValue = “2”
> ADTField.isFinalAnswer = **True**
> ADTField.LeaveFieldNav = 2

4.6 Action Events Within and Between Fields

There are many different types of “actions” that Blaise records and these may or may not be of interest to users of the data. To simplify things, there are two “action” columns for each record, one for actions while within the field and the other for actions after leaving the field. There are also some special columns to store other types of events.

As mentioned earlier, a single record for a Blaise Field includes all the audit information between the field’s Enter and Leave events, as well as any events that occur before the next field Enter event. Every field visit, therefore, has one record in the ADTField table and this record contains all the information from the time the user enters the field until the time the user enters the next field (or ends the survey).



“Action” and other Blaise events (within a field and after “Leave Field”)

ADTField.Action: Name of Blaise action or other events that occur before the “Leave Field” event (e.g., AssignField({Expression State.ActiveFieldName}# 'DontKnow').

ADTField.LeaveFieldAction: Name of Blaise action that occurs after leaving field (e.g., “NextField()”) or other types of events that occur before entering the next field.

ADTField.hasSuppressSignal: True if LeaveFieldAction has a “SuppressSignal” event (e.g., “SuppressSignal(SecI.check_1_)”)

ADTField.GoToUriEvent: Launching of other programs (e.g., “C:\Blaise\Manipula.exe” or “C:\TechSmith\Camtasia Studio 7\CamRecorder.exe”.

4.7 Special ADTField Records: REMARKS

Remarks are stored as records in the ADTField table and are distinguished from field data using the column ADTField.**isRemark = TRUE** to filter the records**.** This is different from the ADTField.**hasRemark** column, which indicates a field that has an associated remark (but is not the remark itself). The fieldname for a remark starts with the pound sign (**#**), so a remark for Age would be named #Age.

The remark itself is stored in the ADTfield.AnswerValue column.. Therefore, another use case for this data is to easily obtain all the user remarks entered into the instrument.

Remark related columns

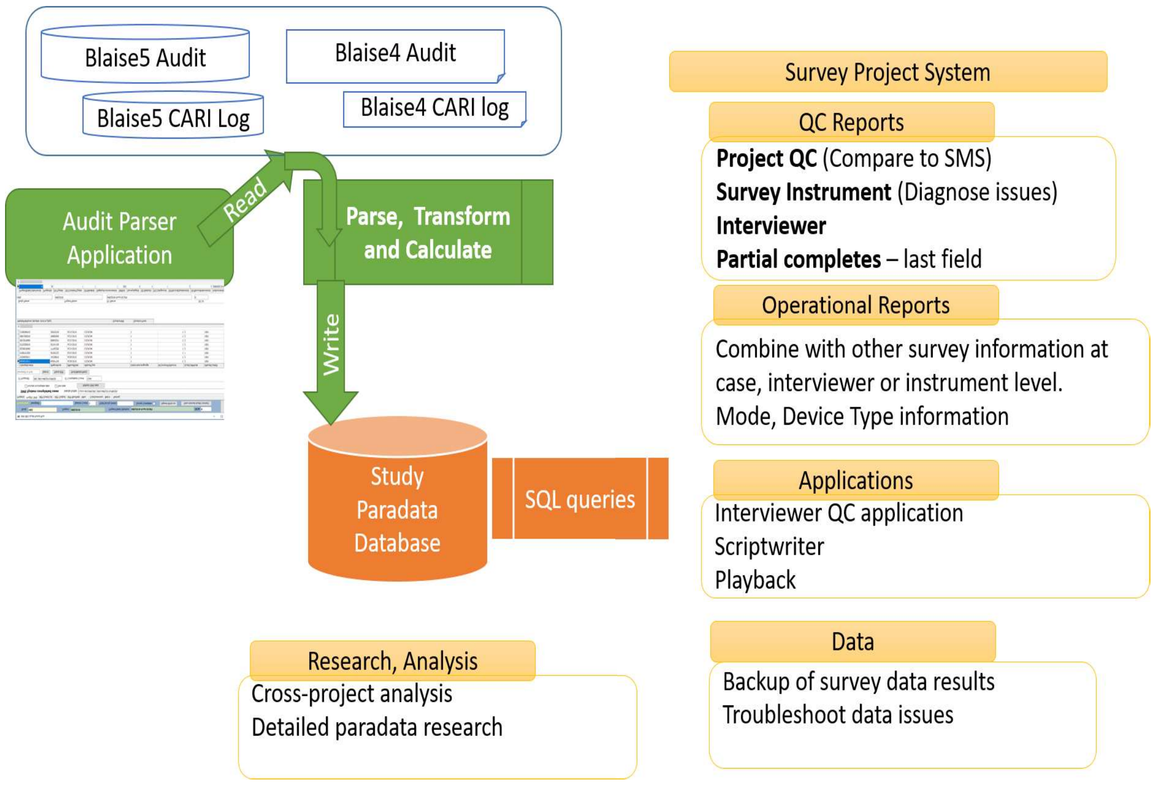
ADTField.FieldName: For remarks associated with a given field, the fieldname starts with # (e.g., #Age).
ADTField.isRemark: True indicates this record is a **remark** (not an answer field).

ADTField.hasRemark: True indicates the given field has a remark. In this example, the record for ADTField.FieldName = “Age” will also have ADTField.hasRemark = True.

5. Paradata As Part of a Survey Project System

With the Audit parser application converting raw audit information into tables, the Study Paradata database can be integrated as one part of the overall survey project system.

Study Paradata database as part of larger survey environment



Blaise5 Audit
Blaise4 Audit
Blaise5 CARI Log
Blaise4 CARI log
Audit Parser Application
Read
Parse, Transform and Calculate
Write
Study Paradata Database
SQL queries
Survey Project System
QC Reports
Project QC (Compare to SMS)
Survey Instrument (Diagnose issues)
Interviewer
Partial completes – last field
Operational Reports
Combine with other survey information at case, interviewer or instrument level.
Mode, Device Type information
Applications
Interviewer QC application
Scriptwriter
Playback
Data
Backup of survey data results
Troubleshoot data issues
Research, Analysis
Cross-project analysis
Detailed paradata research

The image above summarizes some of the ways UM SRC is integrating the audit data into the overall survey system via the Study Paradata database. In addition to reporting, which will touch on in the next section, note that the Paradata database is also used by various **applications**. In the past, these applications built in their own unique parsing routines, but now they can leverage the Paradata database instead of having to support parsing code (this issue was most apparent when moving from Blaise 4 to Blaise 5 as the parsing code for version 4 would no longer work for version 5). Here are some other possible use case examples for this data.

Use case examples for Study Paradata tables

- **Backup of survey data.** A query that filters on ADTField.isFinalAnswer = True and includes ADTField.AnswerValue and ADTField.Fieldname will produce a set of data results.
- **Backup of remarks.** Interviewer remarks can be exported with a simple query for one case or the entire instrument.
- **Quality control.** Compare SMS information about survey complete status and survey completed date and time with the Audit data.

- **Intervene on potential issues** proactively. For example, query for large numbers or outliers such as number of sessions, survey time, field times or other anomalies that may indicate a problem with the instrument, internet connectivity, user issues, etc.
- Talk about "paradata" at your next social event and impress your friends.

6. Queries and Reporting - answering key questions

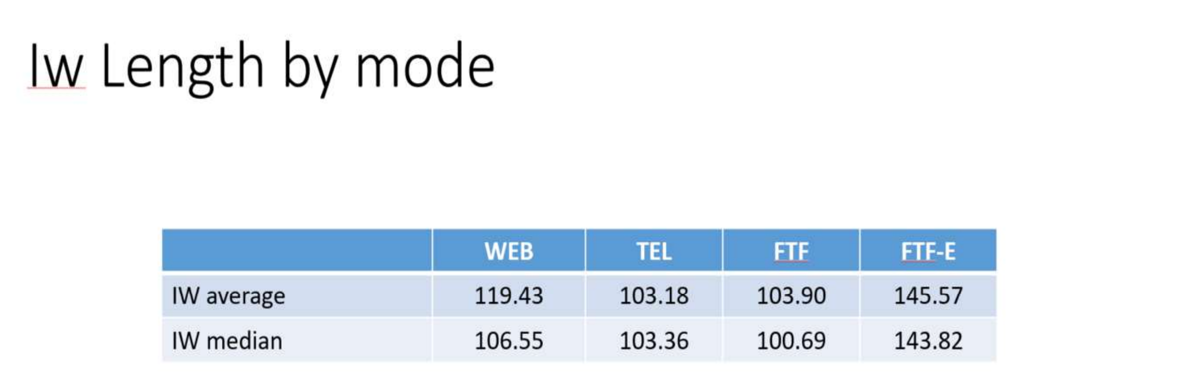
With the audit data parsed in a standard format, it allows for the development of many different types of reports. The storage of the data in relational tables (in SQL) facilitates easy access to the data from a variety of applications, including SAS, Excel, SPSS, R, and Python. For our data manager group, the advent of the parsed audit data saved us approximately 200 lines of SAS code used to create reports around survey timings.

Even without a reporting system or SQL queries, the Paradata information can be accessed using common tools like Excel Power Query and then displayed in data tables, charts, pivot tables, etc.

Note that as a **best practice**, at UM we recommend using database views and parameterized stored procedures for most end-user access to the data. This not only can further simplify access for the end user, but avoids user mistakes in creating custom query joins or filtering data, ensures that data is consistent and results are replicated no matter who uses the information.

Below are some examples of reports that use the Paradata tables. These are provided to help "bring to life" how the paradata information can used for survey projects.

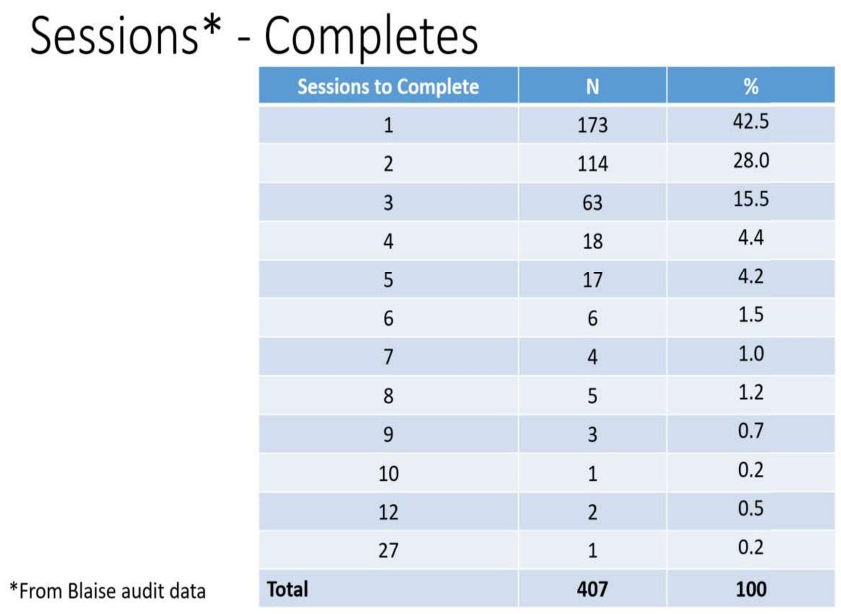
A simple, but effective report showing interview length by mode (for the same instrument).



Iw Length by mode

	WEB	TEL	FTF	FTF-E
IW average	119.43	103.18	103.90	145.57
IW median	106.55	103.36	100.69	143.82

The following report about number of sessions tells us how many attempts were needed to complete the survey. For those cases that required several sessions, the data was looked at in more detail to understand why so many sessions were required (which can be due to instrument issues, connectivity issues, etc.).



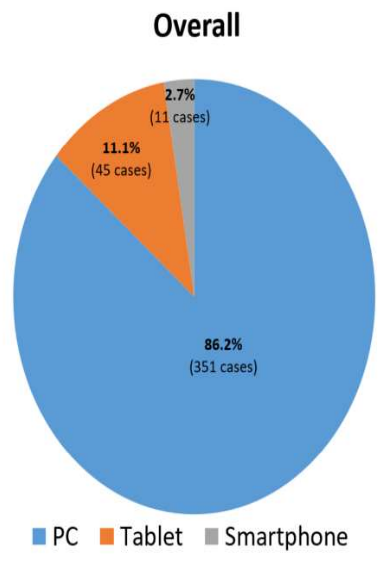
Sessions* - Completes

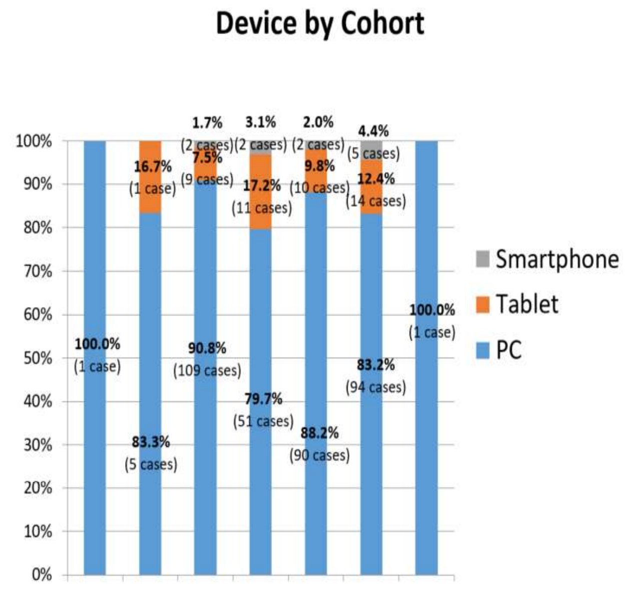
Sessions to Complete	N	%
1	173	42.5
2	114	28.0
3	63	15.5
4	18	4.4
5	17	4.2
6	6	1.5
7	4	1.0
8	5	1.2
9	3	0.7
10	1	0.2
12	2	0.5
27	1	0.2
Total	**407**	**100**

*From Blaise audit data

A report about the type of devices used for a web survey combined with survey Cohort information (although it could just as well be demographics like age, gender, or any other variable of interest). This report is based on the **last Session device** used. A variation of this report would be to look at those who have multiple sessions and switched devices or comparing devices used for incomplete surveys. In this manner, we may find if respondents are having more difficulty answering the survey with certain types of devices.

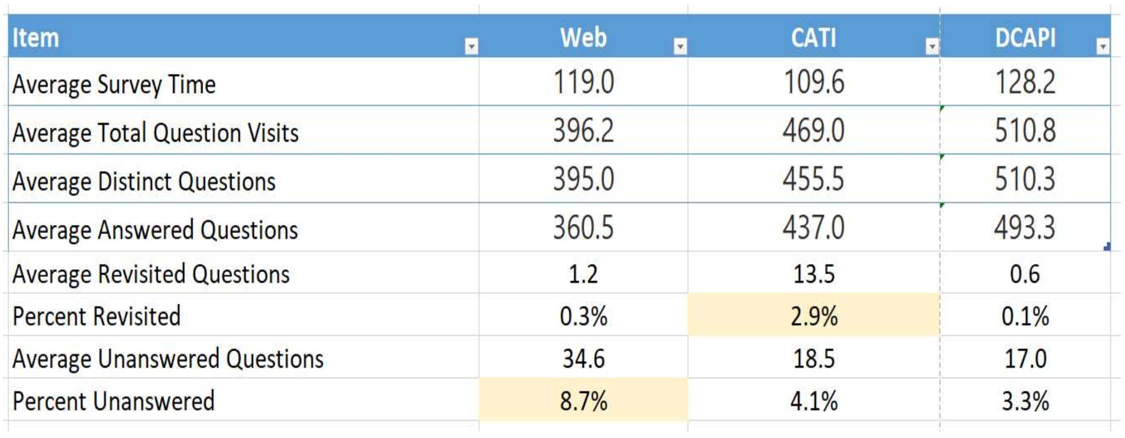
Devices Used* – Completes (N=407)





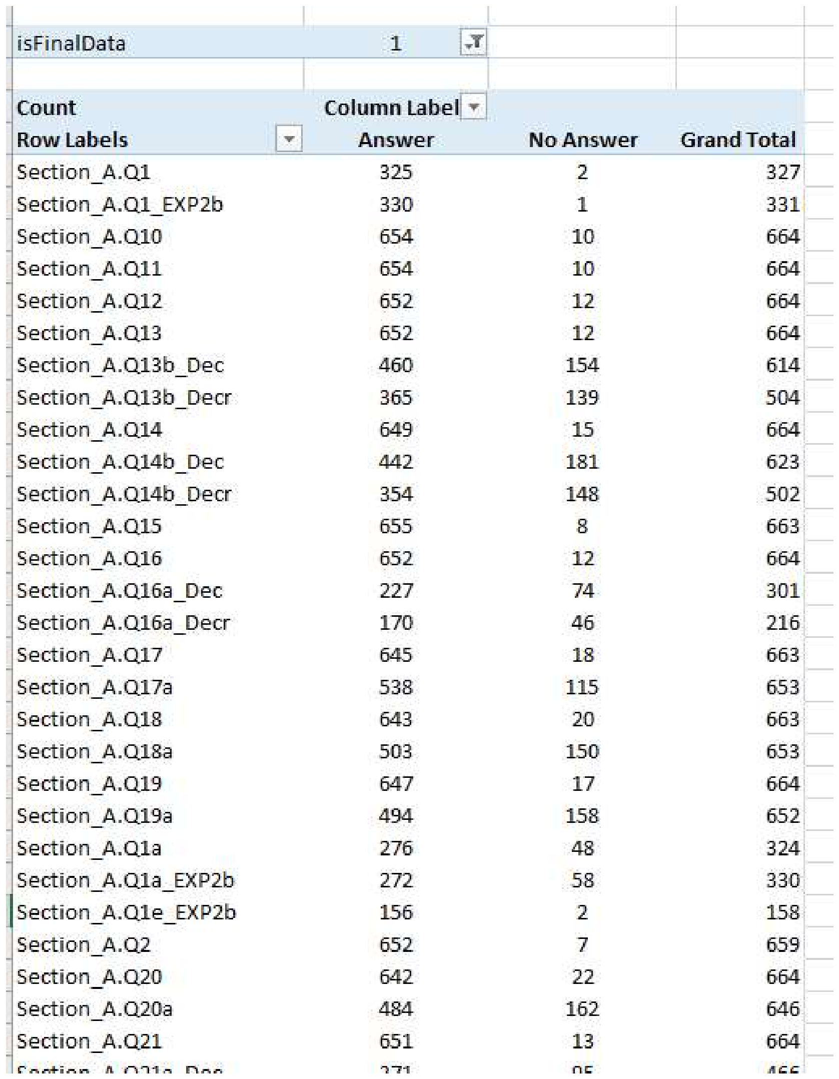
An Excel table created from the Paradata database. The difference between Total Question visits and Distinct Questions tells us how many questions were **revisited**. The difference between distinct questions and answered questions tells us how many of the question fields were left unanswered.

We can supplement a report like this with drill down information such as the question fields that had the highest number of revisits or highest percentage of being unanswered.



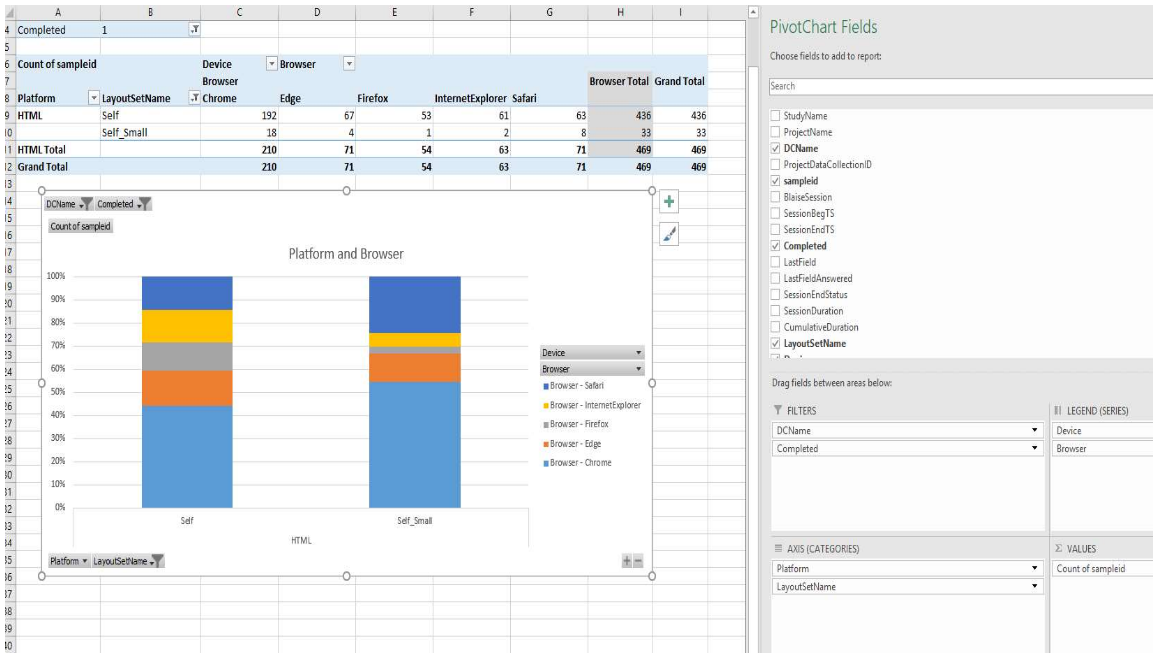
Item	Web	CATI	DCAPI
Average Survey Time	119.0	109.6	128.2
Average Total Question Visits	396.2	469.0	510.8
Average Distinct Questions	395.0	455.5	510.3
Average Answered Questions	360.5	437.0	493.3
Average Revisited Questions	1.2	13.5	0.6
Percent Revisited	0.3%	2.9%	0.1%
Average Unanswered Questions	34.6	18.5	17.0
Percent Unanswered	8.7%	4.1%	3.3%

Analyzing "**no answer**" distribution for each survey question in Excel. While this information can come directly from the survey data, using the audit data we can confirm how many answered or not based on only those who actually **visited** the question. And, with the Paradata database, it is operationally easy to make this part of a regular reporting process while the survey is in the field.



isFinalData	1		
Count	**Column Label**		
Row Labels	**Answer**	**No Answer**	**Grand Total**
Section_A.Q1	325	2	327
Section_A.Q1_EXP2b	330	1	331
Section_A.Q10	654	10	664
Section_A.Q11	654	10	664
Section_A.Q12	652	12	664
Section_A.Q13	652	12	664
Section_A.Q13b_Dec	460	154	614
Section_A.Q13b_Decr	365	139	504
Section_A.Q14	649	15	664
Section_A.Q14b_Dec	442	181	623
Section_A.Q14b_Decr	354	148	502
Section_A.Q15	655	8	663
Section_A.Q16	652	12	664
Section_A.Q16a_Dec	227	74	301
Section_A.Q16a_Decr	170	46	216
Section_A.Q17	645	18	663
Section_A.Q17a	538	115	653
Section_A.Q18	643	20	663
Section_A.Q18a	503	150	653
Section_A.Q19	647	17	664
Section_A.Q19a	494	158	652
Section_A.Q1a	276	48	324
Section_A.Q1a_EXP2b	272	58	330
Section_A.Q1e_EXP2b	156	2	158
Section_A.Q2	652	7	659
Section_A.Q20	642	22	664
Section_A.Q20a	484	162	646
Section_A.Q21	651	13	664
Section_A.Q21a_Dec	271	95	466

An example of an Excel pivot table and chart that can be used to look at the data in a number of different ways. Having Audit data in the database makes it easy to create pivot views of the information.



Completed	1							
Count of sampleid		Device Browser	Browser					
Platform	LayoutSetName	Chrome	Edge	Firefox	InternetExplorer	Safari	Browser Total	Grand Total
HTML	Self	192	67	53	61	63	436	436
	Self_Small	18	4	1	2	8	33	33
HTML Total		210	71	54	63	71	469	469
Grand Total		210	71	54	63	71	469	469

PivotChart Fields

Choose fields to add to report:

- [ ] StudyName
- [ ] ProjectName
- [✓] DCName
- [ ] ProjectDataCollectionID
- [✓] sampleid
- [ ] BlaiseSession
- [ ] SessionBegTS
- [ ] SessionEndTS
- [✓] Completed
- [ ] LastField
- [ ] LastFieldAnswered
- [ ] SessionEndStatus
- [ ] SessionDuration
- [ ] CumulativeDuration
- [✓] LayoutSetName

Drag fields between areas below:

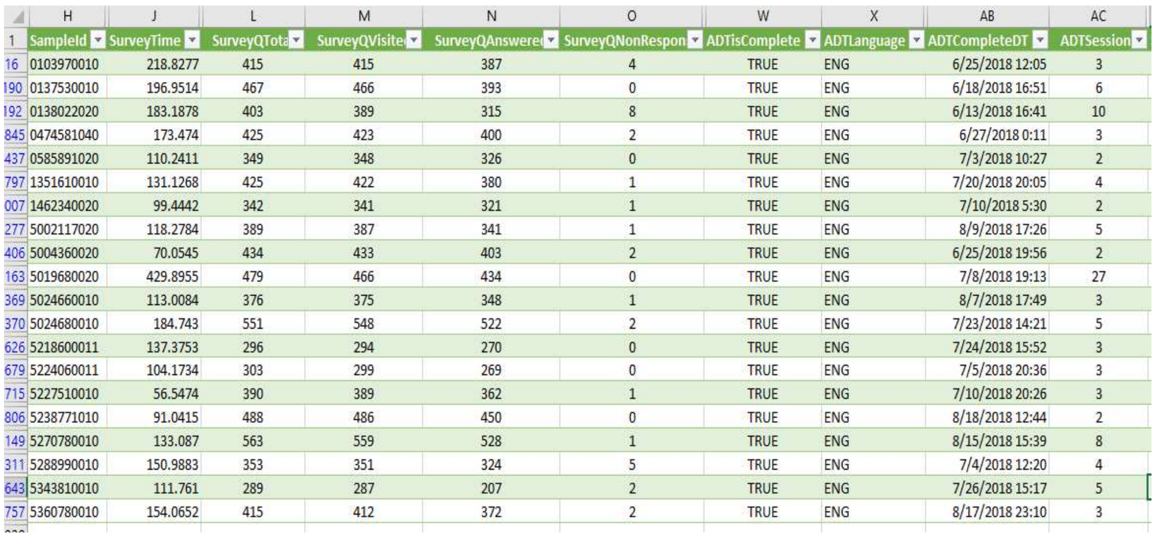
FILTERS: DCName, Completed

LEGEND (SERIES): Device, Browser

AXIS (CATEGORIES): Platform, LayoutSetName

Σ VALUES: Count of sampleid

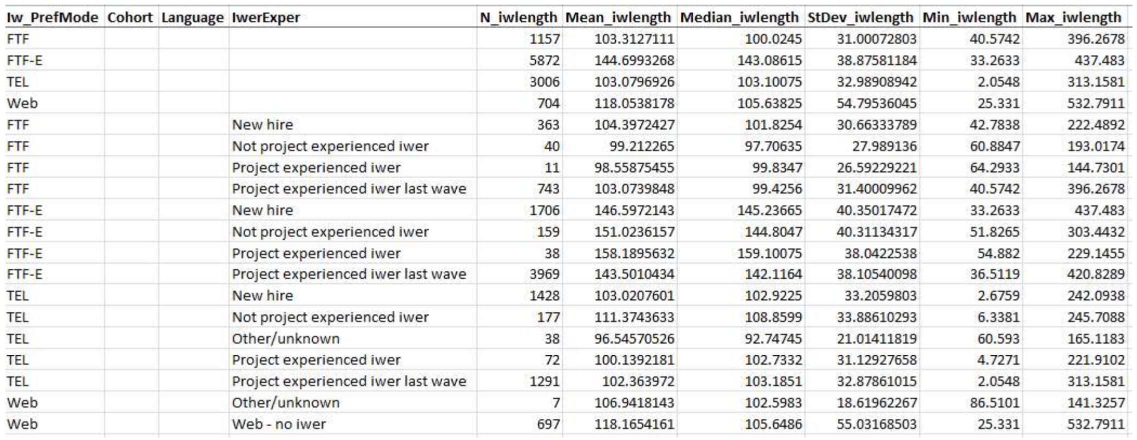
A simple Excel data table, using CaseSummary data, that can be filtered. For example, you can find the survey that has 27 sessions!



	SampleId	SurveyTime	SurveyQTota	SurveyQVisite	SurveyQAnswere	SurveyQNonRespon	ADTisComplete	ADTLanguage	ADTCompleteDT	ADTSession
16	0103970010	218.8277	415	415	387	4	TRUE	ENG	6/25/2018 12:05	3
190	0137530010	196.9514	467	466	393	0	TRUE	ENG	6/18/2018 16:51	6
192	0138022020	183.1878	403	389	315	8	TRUE	ENG	6/13/2018 16:41	10
845	0474581040	173.474	425	423	400	2	TRUE	ENG	6/27/2018 0:11	3
437	0585891020	110.2411	349	348	326	0	TRUE	ENG	7/3/2018 10:27	2
797	1351610010	131.1268	425	422	380	1	TRUE	ENG	7/20/2018 20:05	4
007	1462340020	99.4442	342	341	321	1	TRUE	ENG	7/10/2018 5:30	2
277	5002117020	118.2784	389	387	341	1	TRUE	ENG	8/9/2018 17:26	5
406	5004360020	70.0545	434	433	403	2	TRUE	ENG	6/25/2018 19:56	2
163	5019680020	429.8955	479	466	434	0	TRUE	ENG	7/8/2018 19:13	27
369	5024660010	113.0084	376	375	348	1	TRUE	ENG	8/7/2018 17:49	3
370	5024680010	184.743	551	548	522	2	TRUE	ENG	7/23/2018 14:21	5
626	5218600011	137.3753	296	294	270	0	TRUE	ENG	7/24/2018 15:52	3
679	5224060011	104.1734	303	299	269	0	TRUE	ENG	7/5/2018 20:36	3
715	5227510010	56.5474	390	389	362	1	TRUE	ENG	7/10/2018 20:26	3
806	5238771010	91.0415	488	486	450	0	TRUE	ENG	8/18/2018 12:44	2
149	5270780010	133.087	563	559	528	1	TRUE	ENG	8/15/2018 15:39	8
311	5288990010	150.9883	353	351	324	5	TRUE	ENG	7/4/2018 12:20	4
643	5343810010	111.761	289	287	207	2	TRUE	ENG	7/26/2018 15:17	5
757	5360780010	154.0652	415	412	372	2	TRUE	ENG	8/17/2018 23:10	3

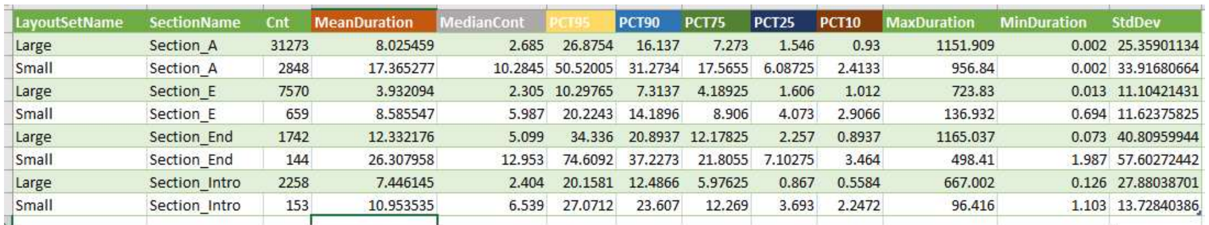
Using additional data from our SMS and combining on SampleId, we can look at interview length by interviewer experience and preferred mode of data collection.

Combined with data from sample management system

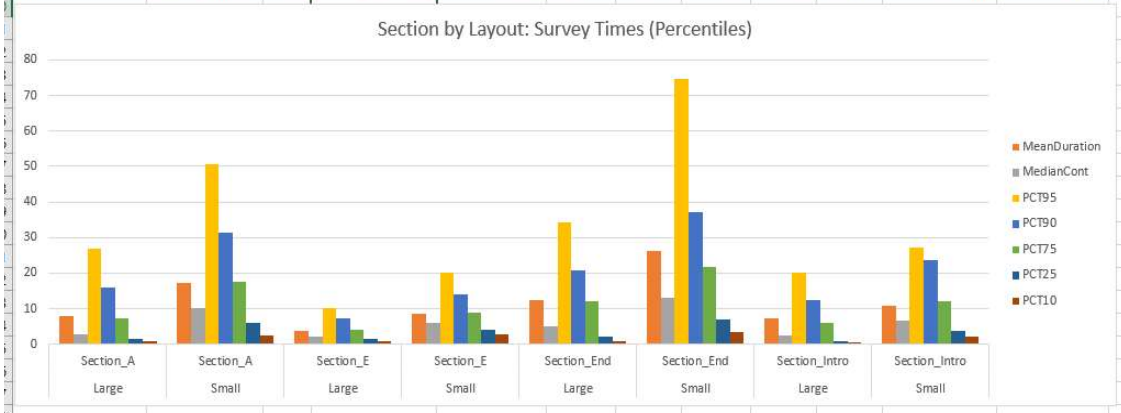


Iw_PrefMode	Cohort	Language	IwerExper	N_iwlength	Mean_iwlength	Median_iwlength	StDev_iwlength	Min_iwlength	Max_iwlength
FTF				1157	103.3127111	100.0245	31.00072803	40.5742	396.2678
FTF-E				5872	144.6993268	143.08615	38.87581184	33.2633	437.483
TEL				3006	103.0796926	103.10075	32.98908942	2.0548	313.1581
Web				704	118.0538178	105.63825	54.79536045	25.331	532.7911
FTF			New hire	363	104.3972427	101.8254	30.66333789	42.7838	222.4892
FTF			Not project experienced iwer	40	99.212265	97.70635	27.989136	60.8847	193.0174
FTF			Project experienced iwer	11	98.55875455	99.8347	26.59229221	64.2933	144.7301
FTF			Project experienced iwer last wave	743	103.0739848	99.4256	31.40009962	40.5742	396.2678
FTF-E			New hire	1706	146.5972143	145.23665	40.35017472	33.2633	437.483
FTF-E			Not project experienced iwer	159	151.0236157	144.8047	40.31134317	51.8265	303.4432
FTF-E			Project experienced iwer	38	158.1895632	159.10075	38.0422538	54.882	229.1455
FTF-E			Project experienced iwer last wave	3969	143.5010434	142.1164	38.10540098	36.5119	420.8289
TEL			New hire	1428	103.0207601	102.9225	33.2059803	2.6759	242.0938
TEL			Not project experienced iwer	177	111.3743633	108.8599	33.88610293	6.3381	245.7088
TEL			Other/unknown	38	96.54570526	92.74745	21.01411819	60.593	165.1183
TEL			Project experienced iwer	72	100.1392181	102.7332	31.12927658	4.7271	221.9102
TEL			Project experienced iwer last wave	1291	102.363972	103.1851	32.87861015	2.0548	313.1581
Web			Other/unknown	7	106.9418143	102.5983	18.61962267	86.5101	141.3257
Web			Web - no iwer	697	118.1654161	105.6486	55.03168503	25.331	532.7911

In Excel, we aggregated field level data to get timing information (including percentiles) by Section and LayoutSetName (small indicates it is for a mobile device, large is for a PC or tablet… Median is the 50th percentile).



LayoutSetName	SectionName	Cnt	MeanDuration	MedianCont	PCT95	PCT90	PCT75	PCT25	PCT10	MaxDuration	MinDuration	StdDev
Large	Section_A	31273	8.025459	2.685	26.8754	16.137	7.273	1.546	0.93	1151.909	0.002	25.35901134
Small	Section_A	2848	17.365277	10.2845	50.52005	31.2734	17.5655	6.08725	2.4133	956.84	0.002	33.91680664
Large	Section_E	7570	3.932094	2.305	10.29765	7.3137	4.18925	1.606	1.012	723.83	0.013	11.10421431
Small	Section_E	659	8.585547	5.987	20.2243	14.1896	8.906	4.073	2.9066	136.932	0.694	11.62375825
Large	Section_End	1742	12.332176	5.099	34.336	20.8937	12.17825	2.257	0.8937	1165.037	0.073	40.80959944
Small	Section_End	144	26.307958	12.953	74.6092	37.2273	21.8055	7.10275	3.464	498.41	1.987	57.60272442
Large	Section_Intro	2258	7.446145	2.404	20.1581	12.4866	5.97625	0.867	0.5584	667.002	0.126	27.88038701
Small	Section_Intro	153	10.953535	6.539	27.0712	23.607	12.269	3.693	2.2472	96.416	1.103	13.72840386


Section by Layout: Survey Times (Percentiles)
Section_A
Large
Section_A
Small
Section_E
Large
Section_E
Small
Section_End
Large
Section_End
Small
Section_Intro
Large
Section_Intro
Small
MeanDuration
MedianCont
PCT95
PCT90
PCT75
PCT25
PCT10