1 Deliverable 3

ACRO: a tool for automating disclosure checks on research output

Abstract

This paper discusses the development of an automatic tool for the statistical disclosure control (SDC) of research outputs. The purpose of the tool (ACRO, for Automatic Checking of Research Outputs) is to distinguish between research output that is safe to publish, output that requires further analysis and output that cannot be published because of substantial disclosure risk. This work was commissioned by Eurostat as a proof-of-concept.

In this document we review the problem and design goals, and describe the Stata implementation of the proof-of-concept. We reflect on the lessons learned from the specific implementation.

We also reflect on the broader lessons learned about the desirability of an automatic tool; in particular, the need for acceptance of the “good-enough-ness” quality of a tool. That is, that the tool reduces the burden to the researcher and/or to the data holder who has to certify that the output is safe to publish. There are clear dividing lines between the statistical problems where the cost-benefit argument supports automation, and where it does not.

1.Introduction

When researchers produce statistics from confidential data, a residual risk exists that the output breaches confidentiality. For example, a table may disclose that an individual with unusual characteristics has a rare illness, or that a group of survey respondents all claim to have taken illegal drugs. Avoiding such a 'disclosure' is a concern of the statistical organisations (SOs) which provide the data.

The risk of such a disclosure increases with the sensitivity and identifiability of the data. The highest-risk data is typically made accessible to researchers through 'research data centres' (RDCs). These allow the researchers full access to view, manipulate and model the data in an environment controlled by the SOs. The secure RDC environment typically blocks researchers from accessing the internet, uploading or downloading data, sharing with others, or otherwise removing or re-identifying the data. In addition, researchers using RDCs usually undergo some form of data governance training (ADSS, 2016, appendix) to ensure that they follow procedure

Nevertheless, despite the user training and the secure environment, there remains the possibility that an analyst may accidentally produce a disclosive output from this very sensitive data. Therefore, most RDCs operate some process to check the output produced by the researcher. Statistical disclosure control (SDC) is the process of identifying and countering disclosure risk in quantitative data and analyses. We refer to the techniques to cover the full range of analytical outputs 'output SDC' (OSDC), to distinguish it from 'input SDC', the removal of identifying information from microdata before giving to researchers.

There are different operating models for applying OSDC, depending on context. SOs producing similar tables regularly (eg quarterly employment data) can make use of automated tools such as tauArgus or SDCtable, which are designed for this process. For more flexibility, SOs are investigating 'confidentiality on the fly' tools (Eurostat, 2018), such as the TableBuilder developed by the Australian Bureau of Statistics. This uses the 'cell-key' perturbation method to provide a high degree of both flexibility and security.

These are less appropriate for research environments, which are characterised by extensive data manipulation, selection of novel subsets, a great variety of statistical techniques, and idiosyncratic presentation of results. As a result, SOs that check outputs rely upon manual inspection by trained checkers, a potentially slow and costly process. Figure 1 gives a schematic representation of the process.

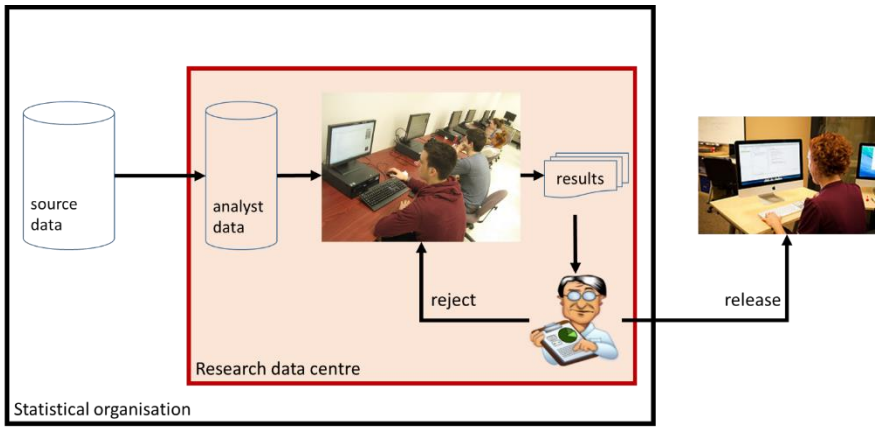

Figure 1 Schematic representation of workflow in a research data center

In December 2019 the authors were commissioned by Eurostat to investigate the possibility of an automated tool to speed up their RDC output checking procedures. The team were required to produce, if possible, a 'proof-of-concept' that could be tested in Eurostat and other SOs. The aim was not to produce a tool that could address all SDC problems; rather, the ambition was to design a

tool that would substantially reduce the need for manual checking at Eurostat, as well as gaining acceptance from both output checkers and RDC users (researchers).

This paper describes the result of that process. Section 2 covers the concept of OSDC and output checking. Section 3 describes the design and development of the resulting Stata tool, ACRO (Automated Checking of Research Outputs). Section 4 reflects on the lessons learned and the future possibilities, including how ACRO could be further developed using non-proprietary software. Section 5 concludes.

In this document we use 'OSDC' to refer to the collection of statistical techniques to identify the disclosure risk and to mitigate the risk, and 'output checking' to refer to the workflow/operational procedures followed by the SO.

2. Background: the output-checking problem

Research output presents a very low base level of risk: data are typically sampled, manipulated and transformed in ways that make the disclosure risk negligible in realistic environments. Researchers also tend to be interested in low-risk outputs such as estimation; high-risk outputs such as tables are used to describe the analytical dataset and so are likely to have large numbers of observations.

Nevertheless, OSDC is used to reduce that risk still further by imposing restrictions which guard against errors and provide an extra margin of safety. This is acceptable to researchers because good practice for ensuring statistical validity (such as many observations, or lack of extreme values or outliers) is consistent with good confidentiality practice.

The following sub-section considers the underlying statistical issues. Sub-section 2.2 considers operational aspects. Sub-section 2.3 brings these together to draw the implications for automatic SDC tools.

2.1 Statistical issues

1. Identification and association

Disclosure may take the form of (a) identity disclosure – that is, identification of an individual, or group of individuals, and/or (b) attribution disclosure – the association of some protected information with that individual. As the latter is generally not feasible without the former, attention tends to focus on the identification of individuals by, for example, discovering that a combination of categories can be used to identify one individual.

Identification is normally required to be exact for a disclosure to take place; that is, a person or group is unambiguously identified. In contrast, the association does not need to be exact to be disclosive. For example, an exact disclosure would demonstrate that the richest person in a named village earned €765,432 in a year. An approximate disclosure would show that the richest person in the village earned between €740,000 and €780,000 in the year; whether this counts as a disclosure in practice would depend on the tolerance of uncertainty at the SO.

In this paper we only consider actual confidentiality breaches. However, we note that SOs are also concerned with potential confidentiality breaches: for example, a single person listed in a category might be perceived as identifying a data subject, even if it does not. As a result, most SOs aim to avoid statistics that give the impression of disclosure, irrespective of whether this is the case.

2. Primary disclosure

Primary disclosure is where a statistic directly allows inferences to be made about individual unit (be that human or business) from the information in that statistic.

Frequencies allow individuals or groups to be identified from the categories used for the statistic (e.g., a single company making electric wheelchairs in Belgium; all individuals in a survey identified

as diabetic also have the fragile-X gene).

Magnitudes allow values to be ascribed to individuals or groups (maximum earnings in a village; minimum research and development spent in an industry). Magnitudes are also susceptible to dominance. For example, turnover and employment in a small town may be dominated by a large international company headquartered there, so that statistics are largely approximations of the values for that one big organisation. Dominance is almost impossible to assess simply by looking at an output, but requires detailed knowledge of specific values used to construct the dataset.

Primary disclosure is managed by simple rules. Minimum thresholds ensure that there are sufficient observations to be uncertain about any individual respondent. Even with minimum thresholds, if the contribution to a statistic is dominated by a few individuals, this may allow large respondent values to be approximated. Dominance is managed by the 'p%-rule': ordering observations 1..N with the largest first, there is no dominance if the sum of observations 3..N is at least p% of largest observation. Dominance can also be managed by an 'N, K' rule, where the largest N observations must contribute less than K% of the total. These are easy for the researcher to calculate, but are often ignored, and difficult for an output-checker to assess from the final output.

One implicit problem of primary disclosure is the case of non-complete categories: for example, only displaying the percentage of male respondents and omitting the proportion of non-males. The same simple rules clearly apply, but the problem lies in noting that the set of categories is not complete. As Hundepool et al (2012) and Ritchie (2019) note, the solution is to require researchers to produce a complete set of categories, but this is not necessarily enforced.

In the case of primary disclosure, automated output checking may have an advantage over manual checks. Checking for dominance is a deterministic procedure, as is ensuring that the full set of categories is accounted for. An automated solution may be able to overcome the lack of interest in these checks shown by humans.

3. Class disclosure

Class disclosure, also known as group disclosure, occurs when all or none of the observations fit into a particular category (some SOs define class disclosure as 'almost every/almost no observation' but this is a risk management definition rather than a statistical one). Rather than a specific individual being identified, it is enough to know that an individual fits into a group about which something is known. For example, if a study reports that all students surveyed have tried cannabis, then this is a disclosure for any student who can be identified as a survey respondent, without any other information.

Class disclosure breaches the general rule that good statistics and disclosure protection go hand-in-hand. The most valuable class disclosures are also likely to be the most sensitive; for example, a disclosure that no-one living in a poor ex-mining community earns over €500m per year is technically but not practically disclosive, and of little value. In contrast, a disclosure that no-one in the area earns over €20,000 per year is both an important statistical finding, and very informative about the people in that area.

Class disclosure can be dealt with by disallowing the publication of figures where 0% or 100% of the population can be put into one cell. However, this is complicated by the existence of 'structural zeroes'. These are cases where the expectation is that the cell is zero. For example, when tabulating qualifications of different age groups, it would not be disclosive to show that none of the 16-17 year olds have acquired a university degree. A blanket rule of "no 0% or 100%" could also create perverse incentives. For example, a statement "0% of this group had a college education" could be replaced by "37% left school with no qualifications, and 63% achieved school qualifications but nothing higher"; this is much harder to pick up. For these reasons, class disclosure is the hardest primary disclosure risk to assess automatically.

4. Sample or population?

When the data is a sample rather than the whole population, the question of whether small or zero values are problematic becomes more complex. Consider the statement in the previous subsection

that “no-one in the area earns over €20,000 per year is…very informative”. If this only relates to the sample, then it is not disclosive about the population. On the other hand, if those in the sample can be easily identified (for example, some of those sampled give interviews to the local newspaper), then it can be disclosive about identified individuals.

Researchers often select subsamples of the data to explore particular hypotheses. This reduces the likelihood of any information about the population being disclosed through class disclosure; however, it may increase the likelihood of individual disclosure by listing selection criteria (location ethnicity etc). The selection of a subsample of the data may not be clear to the output checker.

5. Secondary disclosure

Secondary disclosure is where the statistic can be combined with other information to reveal identities or values. Examples of secondary disclosure include suppressed cells being reconstructed using totals or other statistics. Secondary disclosure within a single output is straightforward to deal with: reconstruction of suppressed values in tables can be dealt with by secondary suppression (keeping totals but removing other values so reconstruction is impossible), or by recalculating totals after suppression so that only the display values are totalled.

The major problem for secondary disclosure is differencing; that is comparing two similar statistics that only differ in a known way (one more observation, say, or the data being broken down into different overlapping categories). The difference between the two statistics can be used to compute both frequencies (differences in the number of observations in categories) and exact values (for example, by reverse engineering two sample means which differ only by a single observation.

Secondary disclosure is an unsolvable problem from a theoretical perspective (Ritchie, 2019) as all possible past and future outputs are in scope for differencing. Even if secondary disclosure checks are restricted to an archive of N previous outputs, there remains a major computation problem. Taking a pragmatic approach to developing automated testing, it would be possible to check a new output against the archive items individually in linear time – that is to say that N pairwise checks would be needed. If it was felt that the new output would be compared against all possible differenced pairs of existing outputs, there would be $O(N^2)$ of these; comparing against triples of outputs would scale as N^3, etc. By contrast, Moore’s Law suggests that at best computing power doubles every 18 months, so simply “throwing computational power” at the problem can never be a solution for anything but small archives (low N).

Hence, all SOs develop rules as to what constitutes a ‘reasonable’ set of outputs to check (for example, recent outputs by the same researcher, or outputs produced by the SO from the same dataset). This is a context- and technology-specific issue which is likely to change over time (Ritchie and Smith, 2018): a decision needs to be made about the likely growth rate of outputs to be checked (and in the archive), the risk of disclosure by differencing from 2 or more existing outputs, and the cost/availability of compute resource. Any automated output checking system therefore needs to allow for the subjective preferences of the SO as to how far differencing is considered.

6. Automated reasoning based on semantic metadata

Automated checking is difficult because of the need to understand context. In theory, this could be addressed by formally describing the semantic (i.e. ‘meaningful’) characteristics of the data. Consider the properties of a variable called ‘age’: as well as being transitive and ordinal, it embodies a temporal relation (in a fixed population the people age over time).

The semantic characteristics of the data can be used directly to give context to the differencing risk: “the maximum age of individuals in the dataset is non-disclosive”. The transitive property (x>y, y>z implies x>z) allows for automated reasoning about set inclusion. For example, imagine two outputs taken from the same population survey showed that (i) the maximum salary for people between 18 and 23 was €25,000, and (ii) the maximum salary for people between 18 and over, but less than 25 was €35,000. Knowing age is transitive, it can be *automatically* inferred that someone in the group of 24 year-olds earned €35,000.

However, knowing age is temporal, means that if the two outputs came from different releases of the

data (say 6 months apart) then the automated reasoning could only make probabilistic inferences. These would have to take into account the rates at which people aged, and the distributions of age and birth-months within a population.

7. ‘Safe’ and ‘unsafe’ statistics

Ritchie (2008) introduced the concept of ‘safe’ and ‘unsafe’ output types (originally referred to as ‘safe and unsafe’ outputs, but the term was changed to avoid confusion). These are sometimes now referred to as ‘low review’ and ‘high review’ outputs in user training such as ONS (2019).

A ‘safe output type’ is one where there is no meaningful disclosure risk in the statistic because of its mathematical form - an example is a linear regression coefficient (Ritchie, 2019a). Safe output types may have some associated rules. For example, in the case of linear regression coefficients the explanatory variables must not consist solely of categorical variables with all interactions included as regressors. These rules need to be capable of being easily, quickly and reliably checked.

An ‘unsafe output type’ is one where the mathematical form allows for primary or secondary disclosure without exceptional difficulty. For example, the mean is an ‘unsafe output type’ as it is plausibly susceptible to primary disclosure (singe observation, dominance) and secondary disclosure (by differencing). For an ‘unsafe output type’ to be released, it needs to be demonstrated that the output, in the specific instance being considered, has no significant disclosure risk.

Questions of primary and secondary disclosure only apply to ‘unsafe output types’:

Output type		Action		Decision
Safe	→	Check rules	→	Release if rules are met
Unsafe	→	Check primary and secondary disclosure	→	Release if non-disclosive

Safe output types can therefore be checked automatically. Bond et al (2015) provide the most recent list of safe and unsafe output types.

8. Actual versus potential disclosure

The above discussion identifies potential disclosure based on implicit worst-case assumption that the identification of a sample unique is a breach of confidentiality. As noted in Hafner et al (2015), this is a sensible criterion in the context of methodological advance. However, in genuine analytical environments the worst-case assumption is difficult to justify; there is little support for the claim that a single observation or an empty cell definitely exposes confidential information about a known individual or groups.

Unfortunately, it is hard to develop rules as to what turns a potentially disclosive output into a non-disclosive one. Converting values into deviations from means, for example, might be an appropriate transformation in some cases, but not others.

SOs persist with the worst-case assumption for two reasons. First, as noted above, OSDC rules can generally be accommodated in genuine research; experience of research centres over the last fifteen years or so shows that a cautious approach does not normally impact significantly on the usefulness of outputs. Second, SOs are concerned about perceptions of confidentiality breaches: a single observation may be non-disclosive, but it could be misinterpreted as disclosive and used to attack the reputation of the SO. For both these reasons, SOs are expected to be more cautious than the strictest statistical case justifies.

It has been argued (e.g. Ritchie and Smith, 2018) that growth in computing power, accessible data sources such as social media, and machine learning will increase re-identification possibilities in the future, and therefore SOs should take a position which is stricter than currently justified to prevent future breaches of confidentiality. It remains to be seen whether this will affect research outputs.

2.2 Operational aspects

Alves and Ritchie (2019) note that output checking is not a statistical problem, but an operational problem. That is to say: the statistical issues discussed above come into play when faced with an output to evaluate, but the operational decisions that lead up to whether an output is reviewed by a human, a machine, or no-one, are much more important for both resource use and confidentiality protection.

Alves and Ritchie (2019) use a customer-segmentation model from management literature (Runners, Repeaters, Strangers and Aliens, or RRSA) to separate outputs into four types:

- Runners: the bulk of outputs which can be dealt with simply automatic rules (for example, a simple table with a threshold rule, or regression output)
- Repeaters: a small but significant part of outputs which require some human intervention to interpret rules and guidelines (for example a scatter plot of residuals)
- Strangers: rare events which require statistical knowledge and should lead to the development of new rules and new types of runners/repeaters
- Aliens: output requests which are outside the scope of the service, such as asking for microdata to be released

Automated output checking should be expected to deal with runners, and may be able to deal with repeaters under certain conditions (perhaps via adaptive limits for tolerance, or, in the longer term, some form of machine learning). Automated checking should not be expected to deal with strangers or aliens, other than to highlight that something unexpected has been presented.

This establishes that automated checking cannot be a universal solution. Rather, its goal should be to reduce human intervention to the minimum necessary. This is not necessarily a bad thing. Knowing that human judgment can be exercised in exceptional cases is important to researchers, and helps to build the bridge of trust between researchers and the support team. The key to automated output checking is to ensure that the time of the output checkers is used most effectively, doing things that only a human can do, or that a human does best.

The choice of organisational process, be that rules-based or principles-based, also affects the feasibility of output checking.

Rules-based OSDC (RBOSDC) processes takes a strict position on the clearance – either something is cleared in accordance with a pre-defined rule, or it is not. If there is no rule, a rule needs to be defined. There are no exceptions. In the RRSA model this means that there are only runners (yes/no outputs) and strangers (reasons to develop new rules). For example, tables get a pass/fail based on a simple threshold rule; regression models are approved on a simple degrees-of-freedom threshold; maxima and minima are not allowed.

This system works well for official statistics but is poorly designed for research outputs because (1) it is hard to define sufficient and sufficiently accurate rules (Ritchie, 2008); (2) each rule has to do two jobs, confidentiality protection and releasing useful results; and (3) it takes no account of variable transformations, subsampling, or subjective categorisation.

Principles-based OSDC (PBOSDC) process uses rules-of-thumb rather than hard rules. All outputs are potentially allowed if the researchers can demonstrate that the output is non-disclosive and important, and that the request for exception is just that – an exception. For example, a table with cells below the threshold may be released if the researcher can convince the output checker that (1) this is non-disclosive (2) this is required for publication, and (3) the researcher is not misusing the privilege of asking for an exception. Without these three conditions, the rule-of-thumb is treated as a hard rule. The advantages of the PBOSDC (rule of thumb) regime are that: (1) it can be simple as they do not need to cover all cases; (2) it can focus on confidentiality, as usefulness is addressed through the exception process; and (3) output checkers can reject any analysis on the grounds that the analyst is abusing the system, irrespective of whether the output is non-disclosive or not. This is

an extremely efficient process, as it embeds the researcher as part of the safe-output production process (Alves and Ritchie, 2019). The disadvantage is that this system requires active buy-in from the researcher, which in turn requires specialist training.

In the context of the RRSA model, runners, repeaters and strangers are allowed in PBOSDC. Repeaters can include runners where an exception is being sought (e.g., tables with cell counts below the threshold). Strangers are acceptable, in their own right and as a reason to develop new runner/repeater guidelines.

2.3 Implications for output checking

The above discussion highlights areas of possibility and limitations for automatic output checking.

- 'Primary disclosure' and 'safe output types' are areas where simple rules can give unambiguous and consistent clearance decisions
- Automated checking may be better placed to solve some of the process mistakes made by individuals
- Secondary disclosure is likely to involve, at minimum, a number of arbitrary rules about what to check
- Understanding the 'meaning' of variables may affect their perceived disclosiveness
- Assessing the actual disclosure risk of the outputs is probably beyond the scope of an automated procedure at present, except in very simple cases
- An automated tool is unlikely to be able to deal with all cases, except by severely restricting the type of output

In summary, from a statistical perspective, automatic output checking will require subjective decisions reflecting the SO's perception of risk to be incorporated into the model.

3. Development of the proof-of-concept tool

3.1 Design

For ease of reference, we refer to the tool as ACRO (Automatic Checking of Research Outputs). The initial design considered the operational and statistical elements which could feasibly be implemented, using the MoSCoW design framework (must-have, should-have, could-have, won't have; Clegg and Barker, 1994); see Table 1.

Table 1 ACRO Functionality

Must have (required functionality)	Statistics: ability to deal with • Tabulation of descriptives (frequency, percentiles, moments, maxima and minima) • Linear and non-linear estimation • Weights Operations • No loss of information except where suppressed • Primary suppression and recalculation of totals • Exceptions • User choice which statistics to submit for checking • User ability to overwrite previous outputs • Easier and faster checking for output checker
Should have (absence will notably diminish functionality)	Operations • Automatic suppression at the user's request • Minimal training for users • Minimal need for users to adjust their code • Dataset-specific SDC criteria
Could have (presence will enhance functionality)	Statistics: • Graphs Operations • Minimal training for output checker • Minimal setup to be installed on the system used by the researcher • Output which has added value for the user • Disclosure by differencing
Will not have this time (not necessary, or too complex/time-consuming for this implementation)	Statistics: • Other Operations • Secondary suppression

The choices of elements to be included or excluded were agreed with Eurostat in advance.

Statistically, tabulation and common estimators were seen as essential to the functionality of ACRO. Dealing with weights was important. Graphs were considered ‘nice to have’ but the difficulty of assessing graphs meant these were a lower priority. No other statistics were programmed.

Operationally, the tool ideally would allow for automatic primary suppression, with totals recalculated as necessary. The alternative would have been to fail all unacceptable outputs, requiring users to apply suppression before submitting tables. This would have been difficult for all concerned, particularly as ACRO would struggle to distinguish between genuinely empty cells and cells set to null by the user. ACRO also needed to have some mechanism for requesting ‘exceptions’, in line with PBOSDC. ACRO needed to allow users to select the outputs to be included, and to overwrite output generated (but not submitted for approval), otherwise it would be overwhelmed by unnecessary or redundant requests. Finally, ACRO clearly needed to be easier and faster for the output checker, otherwise the whole exercise had no point.

Minimal set-up and training for the user was thought to be important in ensuring the acceptance of the tool by the user. Finally, if SDC criteria could not be dataset-specific, then the proof-of-concept would have limited application in genuine environments.

Ease of set-up for the system manager and limited training for the output checker were identified as nice-to-have elements. Both involve one-off costs, so even if these were substantial, they would not necessarily affect the ongoing functionality of the tool. Adding value to the user was important to encourage adoption, but not vital. Users would already be incentivised to use the tool by shorter clearance times, but better output might strengthen this.

Checks against secondary disclosure by differencing between outputs was categorised as a ‘nice-to-have’ as the team felt it extremely unlikely that this could be achieved, but they had some ideas which they considered experimenting with. By adding this in the ‘could’ section, they allowed for the possibility of developing solutions.

Secondary (within-table) suppression was not considered, for two reasons. First, this is difficult to achieve even in dedicated table-suppression programs such as tauArgus and sdcTable, and usually requires expert input if anything other than the simplest minimum-cells rules are to be applied. Second, the team took the view that secondary suppression is a poor SDC solution for research outputs. The argument is that it creates more risk of errors and more risk of disclosure by differencing (the underlying assumption is that research outputs are highly idiosyncratic and so only primary suppression is needed). For both these reasons, only primary suppression was applied, and totals recalculated post-suppression where relevant. If the user chooses not to have primary suppression applied automatically, then outputs failing primary disclosure checks are rejected, and the user has to redesign the table so remove primary confidential cells. In this way secondary suppression is avoided.

3.2 Implementation

ACRO was developed in Stata. This was largely determined by the skill-set of the team, but also because Stata was felt to be more readable than R, its main competitor in research environments. Python was considered as a more future-proof alternative. However, the team was less familiar with Python, and it was unclear how it could be integrated seamlessly into R or Stata.

ACRO requires a set of Stata ado-files to be uploaded in a folder (ideally read-only) accessible to the researcher, along with an Excel macro-enabled template. To run ACRO, the researcher initially calls two lines of set-up code. Before each output to be reviewed, he or she puts ‘safe’ in front of the command to be run, with some optional parameters; otherwise, the command is unchanged. The commands are sent to an Excel file; the researchers receives messages about the SDC check, as well as being able to check the Excel file.

Finally, the researcher calls the command ‘finalise’ which prepares an Excel file for review by the output checker. The output checker can then see all the requested outputs: which have failed, which have passed, which have passed after suppression, and which would have failed but have had an

'exception' requested. A macro allows the output checker to review the 'exceptions' and then quickly delete those which are not acceptable.

ACRO's functionality was plotted in flow diagrams to understand the necessary decision points. Appendix 1 illustrates the flow charts underlying the mid-program functionality; these are the final descriptions, not the ones originally developed which were to be substantially adapted.

The code delivered on all of the 'must, should, could' noted above with three exceptions:

- No satisfactory solution was found for disclosure by differencing, although this did raise several useful avenues to explore, as discussed below
- Graphs were included so that all the requested output could be in one file for review. Because of the difficulty of automatically checking, these were all set to 'review'. It has been pointed out to the team that some graphs, such as histograms, could have been assessed using the same code as for frequency tables
- Percentiles, other than medians, were not implemented; these were felt to be a lower priority, and were omitted due to time constraints.

For maxima and minima a simple ban was applied – that is, requests to report the max/minimum were rejected or suppressed (if automatic suppression was chosen by the user). The alternatives were to allow the user to signal whether maxima and minima were structural (eg 0-100%), or to apply threshold rules for the number of observations at the maximum/minimum. Neither of these were felt to be better solutions in terms of usability and protecting the data.

After implementation, the code and user/manager guides were sent to several SOs. Amongst those who were able to test ACRO (as opposed to reviewing the documentation) the common responses were:

- It had good potential to reduce manual output checking
- It wasn't quite as intuitive as the team hoped, even with the guides, but was easy to get to grips with after a little time
- A wider range of functions would substantially improve its functionality
- A wider range of languages (especially R and SAS) would be necessary to make it usefully throughout the RDCs

3.3 Summary: does ACRO address the statistical problems?

Section 2.1 listed eight statistical issues to be considered by an automated tool. The table below considers how well ACRO addresses these issues.

Issue	Addressed?
Identification and association	Fully addressed The organisation has full freedom to set the appropriate SDC limits consistent with its risk preferences. Exceptions can be granted, subject to manual review
Primary disclosure (small numbers and dominance)	Fully addressed. This is a deterministic operation, given the SDC rules. ACRO is likely to check for primary disclosure better than a human as checks which are difficult for humans (such as dominance) can be easily automated
Class disclosure	Not addressed Empty and full cells are allowed in tables: as data are usually

	(sub)samples, absence of information is usually uninformative. The alternative, removing tables with empty cells, was felt to be too restrictive
Sample or population?	Not addressed. ACRO only understands counts in cells, not their true representation in the world. Therefore, it over-protects.
Secondary disclosure	Not addressed As noted above, there is no general solution to this problem.
Automated (semantic) reasoning	Not addressed ACRO uses no semantic information to improve its decision-making
'Safe' and 'unsafe' statistics	Fully addressed This is directly implemented (it greatly simplifies the checking problem), albeit currently for a small range of statistics
Actual versus potential disclosure	Not addressed ACRO is a blunt instrument, assuming every potential risk is an actual one; as such it over-protects.

Overall, ACRO deals well with simple rules, and the 'exception' procedure allows some flexibility. However, the bluntness of an automated tool means that

- it over-protects by treating every possible risk as an actual risk
- it may over- or under-protect by being unable to determine context

In a research environment, the first of these is likely to dominate; that is, ACRO is more strict than it needs to be when balancing utility against security. However, this is also the approach taken by SOs in manual output checking: experience shows that simple crude rules work well, as long as there is the opportunity to allow for exceptions (Alves and Ritchie, 2020).

4. Lessons learned

4.1 Programme design

A key development was the realisation that one Stata command ('table') could reproduce all the element of other descriptive generators (eg "tabulate", "sum"). Coding therefore focused on 'table' with other commands being handled as calls to 'table'. This saved considerable duplication.

This then affected the development of ACRO for regression outputs, where a similar approach of looking for a 'universal function' was applied and found. In this case, Stata's internal representation of regression results was consistent across all linear and non-linear regression models, and could be used to provide consistent outputs for checking (as 'safe output types', only degrees of freedom are checked for regressions).

A large amount of programming time was initially devoted to trying to handle and reproduce Stata's 'by' command, which split the data into exclusive subsets before running the command on each sub-group – which could in turn have multiple sub-groups. After some weeks, it became clear that letting Stata split the initial sub groups and simply presenting them as separate outputs provided a much cleaner, as well as simpler, set of outputs and code. This is in line with practice in other possible implementation languages, such as the dataframe "groupby" command in python and R.

4.2 Creation of an open-source version

The Stata/Excel development of ACRO was driven by practical considerations to develop a proof-of-concept in the short term. In the longer term, Eurostat requires a non-proprietary solution which could work with the full range of languages used by researchers.

The team reviewed the structure of the Stata code and identified that there were four structural elements to the code:

1. Set-up and completion of the outputs
2. Interpreting the Stata command to create the output for review
3. Carrying out the review
4. Sending the results of the review to the store (in this case, an Excel file)

In each case except Task 2, it was possible to conceive how the ACRO Stata command could be replaced by a call to an external program, for example, some Python code. However, interpreting the Stata code prior to the external call would still either need to be done in Stata, or have an interpreter in the external language. In addition, that external program would need the ability to interrogate the data in memory at the time the command was called, not read a file from disk. This suggests that a third-party (non-proprietary) solution would still need to have substantial modules in the analytical language to carry out Task 2.

However, the ACRO implementation may be over-complicated because it seeks to work seamlessly within Stata; it therefore tries to understand and interpret the output requests as if it were a Stata command so that all of the user's output is checked. A simpler version could reduce the set of options (for example, only allowing frequencies), reducing the work to be done to integrate with Stata. As ever, there is a compromise between complexity and functionality.

There is the problem of communication between part 1 and the others (where to store and update results). Either the source code would need to store some identification, or some predetermined set of file names is used, or possibly some sort of wrap-around environment like Jupyter Notebooks could maintain state information outside of the source code. This could be surmounted by using programming concepts such as "session-ids", but would require an external "clean-up" process to remove datafiles left by incompleted sessions.

Finally, there may be performance implications of an external open-source module carrying out the work. This would require the proprietary software to create a copy of the data to send to the external code, along with the requisite breakdowns. This is not necessarily slower. For example, in ACRO there is much storing temporary files so that the researcher does not find his or her data corrupted by the SDC checks; this could be avoided if the external program does not need to maintain the integrity of the files sent to it for checking.

In summary, converting ACRO to a separate (non-proprietary product) is doable but will involve at least some language dependent component and some way for the components to talk to each other. This may involve some compromise on functionality. It may also involve some drop in performance, but conceivably it could also boost performance of the checking process by having an external program work on the checking.

4.3 Costs, benefits, and being 'good enough'

ACRO was not designed to deal with all outputs. Conceptually, this is not feasible; not all outputs have rules defined for them. Practically, this is not desirable. First, not all outputs are easily assessed by automatic rules – consider scatter plots, for example. Second, not all outputs, even unsafe output types, pose a disclosure risk, and so the researcher can legitimately claim an 'exception'.

Against this there needs to be set the cost of developing increasing complex code, to deal with more types of statistics, or to provide more nuanced interpretations. Moreover, the more complex the code,

the more likely it is that the code becomes difficult to set up and/or use, reducing its appeal to all parties. This is why tau-Argus, for example, is not suitable for research environments. It may be comprehensive in its coverage of standard statistical tables, but the time-cost of defining the metadata and extracting the data subset, and the need for researchers to understand how to use it, means that any benefits of having a verified solution for the RDC are greatly outweighed by the costs.

ACRO was designed, in collaboration with Eurostat, to provide a 'good-enough' solution to output-checking: good enough to meet most demands, substantially reduce output-checker workloads, and be simple to set up and use, but without being an over-engineered behemoth. Reviewers suggested additional commands that could be included to substantially increase the value of the tool. As most of the suggestions are for estimation commands of the 'safe output type', it should be feasible to include these in future versions. However, ACRO, or any successor, is unlikely to provide a universal solution: the marginal benefits of adding commands fall, while the marginal costs increase.

4.4 Statistical issues

Two major statistical issues remain unresolved: semantic understanding and differencing.

The first is the problem of a semantic understanding of the data. ACRO assumes that any data being tabulated or graphed is confidential, and therefore applies its rules. This is why ACRO allows for exceptions: so that users can explain to the output-checker why, for example, tabulations of Herfindahl indexes are not confidential. This is not a particularly satisfactory solution.

In the design phase of the project, the team considered the issue of semantic knowledge; that is, having some metadata about confidentiality risk attached to variables. The tool could then decide: "This variable being tabulated has no confidentiality risk, therefore I need not apply a threshold rule to the outputs derived from it." None of the current statistical packages allow for such qualitative information to be lodged with the data.

The team was also unable to find a useful solution to this. The problem does not come from the initial definition of a variable as 'confidential' – we assume the SO could do this – but from what happens when the researcher begins manipulating and combining variables. Do confidential variables stay confidential through all transformations? If not, does the researcher get to decide if a transformed variable is confidential? This is open to abuse if the gains to producing statistics from self-defined 'non-confidential' data are much greater than from submitting 'confidential' data statistics for review.

In summary, the team's experience of developing ACRO has increased their conviction that more semantic metadata might significantly improve some functionality but cannot fully address the SDC problem.

The second issue is that of differencing. One of the hopes of the team was that, by collecting information on tables being produced, it should be possible within one set of outputs to carry out checks for differencing. Hence, this was included as a 'nice-to-have'.

Once this was considered in practice, a set of questions arose. Should differencing be done sequentially, as tables are generated? This would run into problems if a researcher generated some tables and then decided later not to use some. How about checking for differencing only when the tables are submitted for review? But this would involve re-running all the tables because there is the possibility that some tables used different subsets of the same variables. What if researchers create new variables from old ones (eg age transformed to age band)? How can these be tracked and checked unless ACRO also has information on (and can understand) how the transformed variables were created. Finally, should ACRO just test on pairwise tables? Ideally, it should be tested on all combinations, but this rapidly becomes complicated.

While these are not conceptually new problems, the development of ACRO has thrown these problems into sharp relief. However, one way of providing at least some supporting information has been considered. As ACRO collects metadata, it should be possible to store that information so that tabulations of variables can be tracked over time - repeated tabulations, slight variations, cross-tabs

with other variables. Collecting information about the outputs does not directly solve the differencing problem. However, it might build into a database which could be used both the improve the quality of data ("how do researchers usually categorise age?") and to see where lots of similar tables arise ("the most common tabulation is age in five-year bands against ethnicity"), so that checks on differencing risk can be targeted effectively.

4.5 Institutional factors

One issue that the project highlighted was that the effectiveness and acceptability of such an automatic checking tool is linked to the institutional structures of the SO.

For example, the project team is embedded in the culture of the UK safe RDCs, which use principles-based OSDC, and require all users (and staff) to go through the same training. Researcher training emphasises the impact on clearance times of providing good output, and of building a positive relationship with the support team. Users are made aware that low quality output requests may be rejected because of the resource cost to the checking team, irrespective of statistical risk. ACRO slips easily into this context – something to offer to the researchers as a way of making outputs even faster and easier. This is why the team emphasised getting the user experience right – to make engagement voluntary. However, even if not taken up voluntarily, it would be feasible to require it with a relatively small amount of fuss as long as there was an overall benefit.

The effective use of ACRO therefore requires some investment by the SO in researcher management. However, we would argue that this is a genuine investment – organisations which do train researchers in good output checking behaviour tend to have much higher throughput for lower resource costs than organisations which do not engage the researcher in the output process, even if checking is still manual.

However, it is also worth noting that ACRO has only been tested with SO staff for feasibility, and has not been distributed to users yet for full beta testing. It may well be that assumptions of user acceptability in response to training are overstated. This needs to be tested empirically.

5. Conclusion and next steps

The ACRO project has demonstrated the feasibility of creating an automatic disclosure control tool. The ACRO code is being uploaded to GitHub and made available to users under EU Public Licence. Eurostat is the owner of the code.

This is not the first automatic SDC checking tool for research, but it is the first which explicitly incorporates, in its design principles, the best-practice researcher management principles of the 'EDRU' framework (ADSS, 2016). These principles argue that effective management of the research process requires, amongst other things, recognition of

- the behavioural and institutional factors that determine the success of processes
- risk management rather than risk avoidance as the decision-making criterion
- an evidence-based approach to identifying and assessing risk

The resulting tool meets the proof-of-concept design goals, and has generally received a positive welcome by expert users in trials. An open-source version of the tool is certainly feasible, although it is always likely to need some language-dependent components to provide the interface to the open-source checker. The project has also usefully clarified a number of issues: the balance between complexity and functionality, the value and limitations of semantic metadata, and the importance of the institution's approach to SDC.

At present the ACRO pilot is 'alpha': a working proof of concept. Moving to 'beta' (an approximation to a production tool, albeit still in Stata) requires (1) expansion of the programmed commands to cover all of the most common commands, where possible (2) development of a user engagement

program to support uptake (3) an understanding of how the tool can integrate with organisational processes. As one of the design goals was to make something that researchers would use by choice, a ‘beta’ version will require extensive testing on user acceptability.

Output checker acceptance is also important. The development team experienced a reluctance to engage with the tool at ‘alpha’ stage, partly due to the tool being developed in STATA.One future development is to adapt the developed pseudocode and formulate versions of the tool which can be run across a number of programs (for example R and MatLab).

References

ADSS (2016) Data Access Project: Final Report. Australian Department of Social Services. June. http://eprints.uwe.ac.uk/31874/

Alves K. and Ritchie F. (2019) "Runners, repeaters, strangers and aliens: operationalising efficient output disclosure control". Working papers in Economics, Bristol Centre for Economics and Finance no 20120904 https://ideas.repec.org/p/uwe/wpaper/20191904.html

Bond S., Brandt M., de Wolf P-P (2015) Guidelines for Output Checking. Eurostat. https://ec.europa.eu/eurostat/cros/system/files/dwb_standalone-document_output-checking-guidelines.pdf

Clegg D., and Barker R. (1994). Case Method Fast-Track: A RAD Approach. Addison-Wesley. ISBN 978-0-201-62432-8

Eurostat (2018) Confidentiality on the fly. Paper prepared for the 3rd Working Group on Methodology. Luxembourg. May. https://ec.europa.eu/eurostat/cros/system/files/item_4.3_confidentiality_on_the_fly_2018_04_27.docx

Hundepool A., Domingo-Ferrer J., Franconi L., Giessing S., Schulte Nordholt E., Spicer K. and de Wolf P-P (2012) Statistical Disclosure Control. Wiley

ONS (2019) Safe Researcher Training. Office for National Statistics.

Ritchie F. (2007) Statistical disclosure control in a research environment, mimeo, Office for National Statistics. Edited and reprinted as WISERD Data and Methods Working Paper no. 6 (2011). https://uwe-repository.worktribe.com/output/957311/statistical-disclosure-control-in-a-research-environment

Ritchie F. (2008) “Disclosure detection in research environments in practice”, in Work session on statistical data confidentiality 2007; Eurostat; pp399-406 https://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/2007/12/confidentiality/wp.37.e.pdf

Ritchie F. (2019) “10 is the safest number that there’s ever been”, Work session on statistical data confidentiality 2019; Eurostat. http://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.46/2019/mtg1/SDC2019_S3_UK_Ritchie_AD.pdf

Ritchie, F., & Smith, J (2018) Confidentiality and linked data. In G. Roarson (Ed.), Privacy and Data Confidentiality Methods – a National Statistician’s Quality Review. , (1-34). Newport: Office for National Statistics. https://uwe-repository.worktribe.com/output/856040

Appendix: ACRO Design Documentation

Overview of ACRO tool and workflow

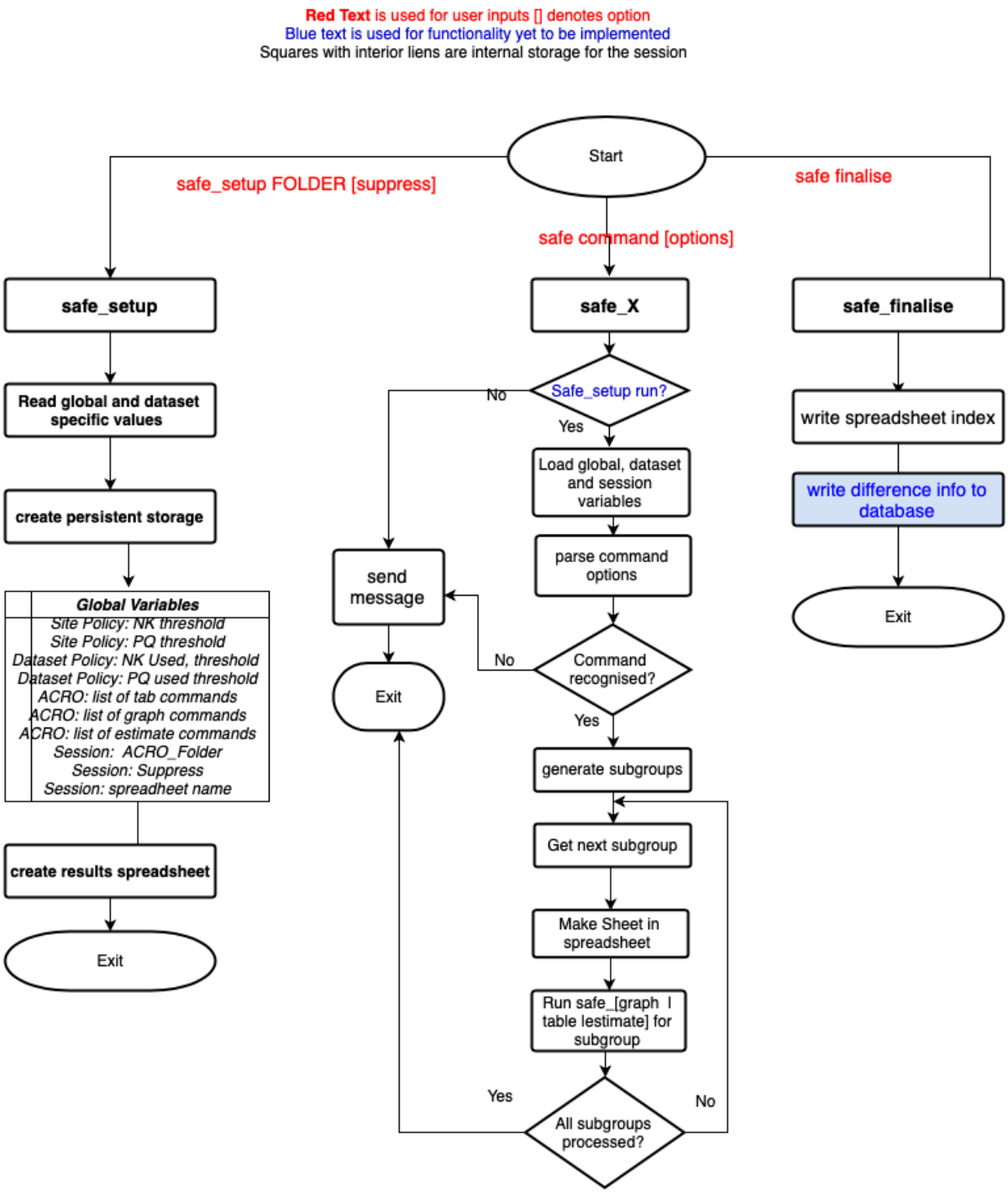

Figure 1 illustrates the current ACRO workflow. Boxes with blue backgrounds represent links to differencing system which would be the result of further work. For the 'safe_X' process the command X is a valid command (currently Stata) that is a member of one of three groups of commands listed in safe_globals. The box (process) labelled 'Run safe[graph|table|estimate] for

subgroup' then calls the appropriate process, passing the actual command requested as a parameter.

Safe Table

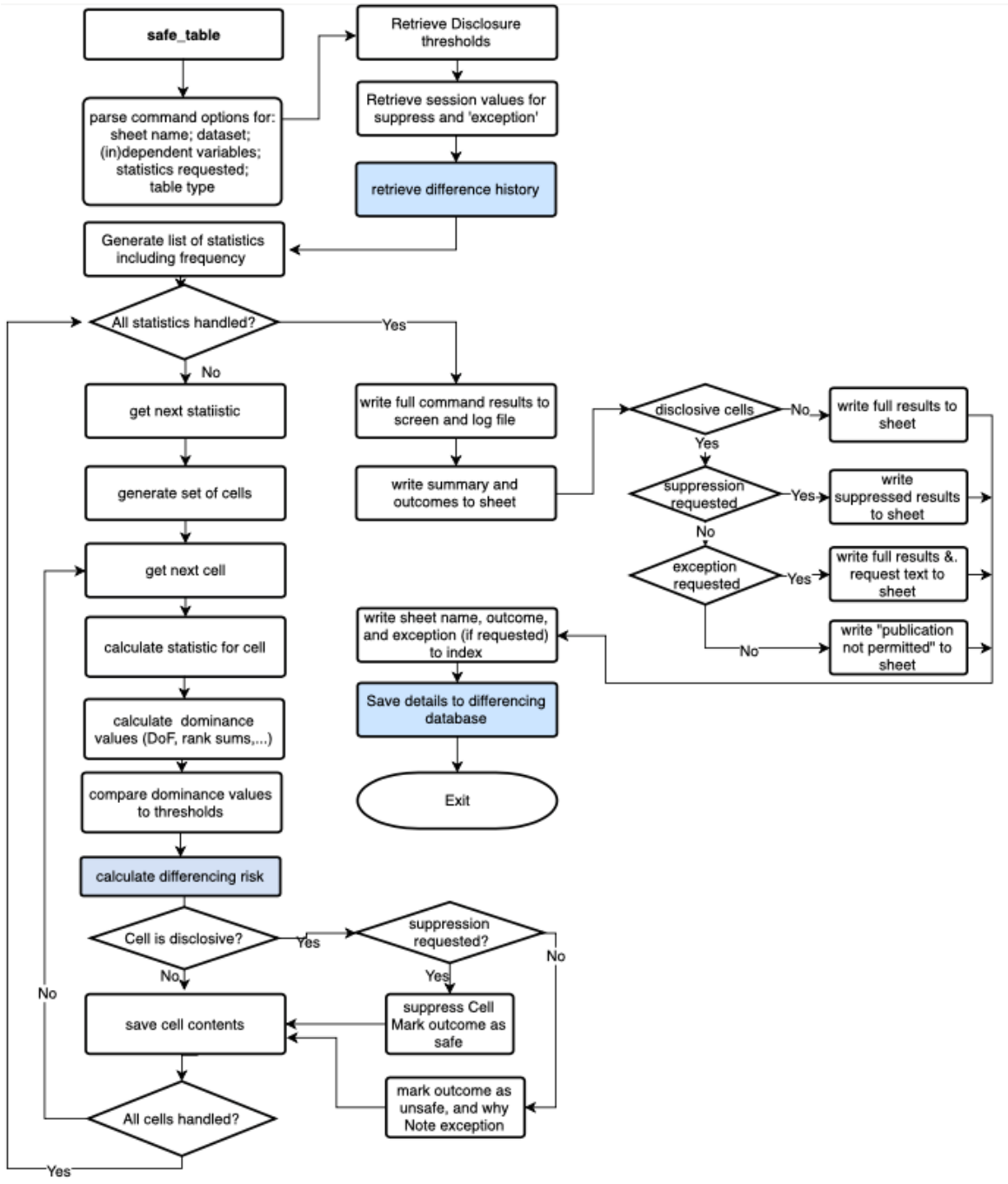

Figure 2: Flowchart for generic safe_table process

Figure 2 illustrates the generic workflow of the safe_table process. Figure 3 illustrates the slightly modified version as implemented in the current Stata prototype. The only difference is whether dominance checks are carried out in a separate loop or within the main code loop. This difference would depend on ease of implementation and integration with standard libraries in the language being used for a specific implementation.

The process "*Calculate Dominance Values"* calculate the number of observations in the cell, then ranks the observations by value and records values needed for disclosure checking such as the sum of the top N values and the remaining observations.

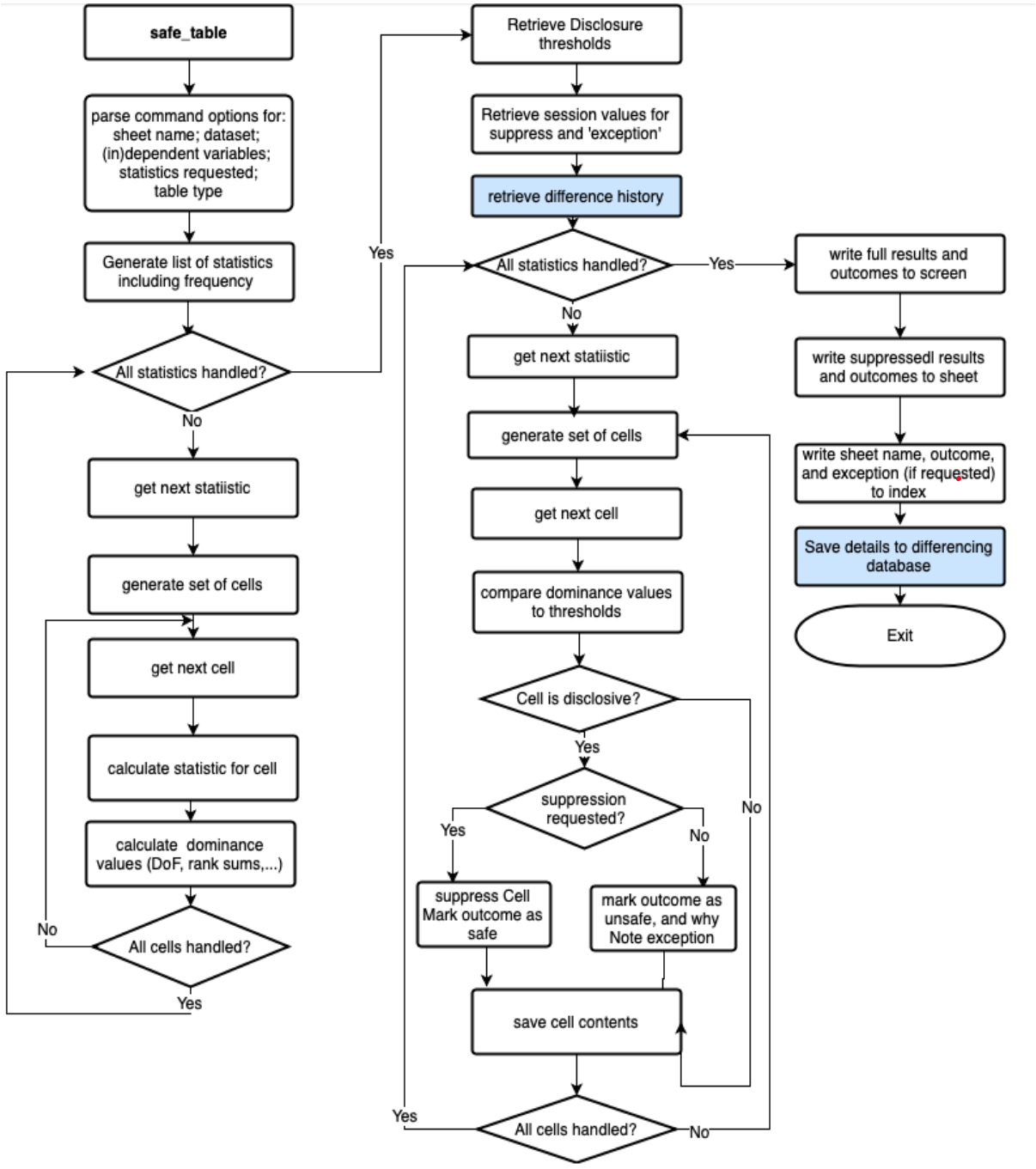

Figure 3: Safe_Table: Stata version as per ACRO prototype

Safe Estimates

Figure 4 illustrates the safe_estimates workflow. Note that all models (regression etc.) are considered 'safe' unless the Degrees of Freedom (DoF) falls below some threshold. The process in which the initial results are written to the data sheet, then overwritten if they are considered 'unsafe' is Stata specific and could be avoided in a generic implementation.

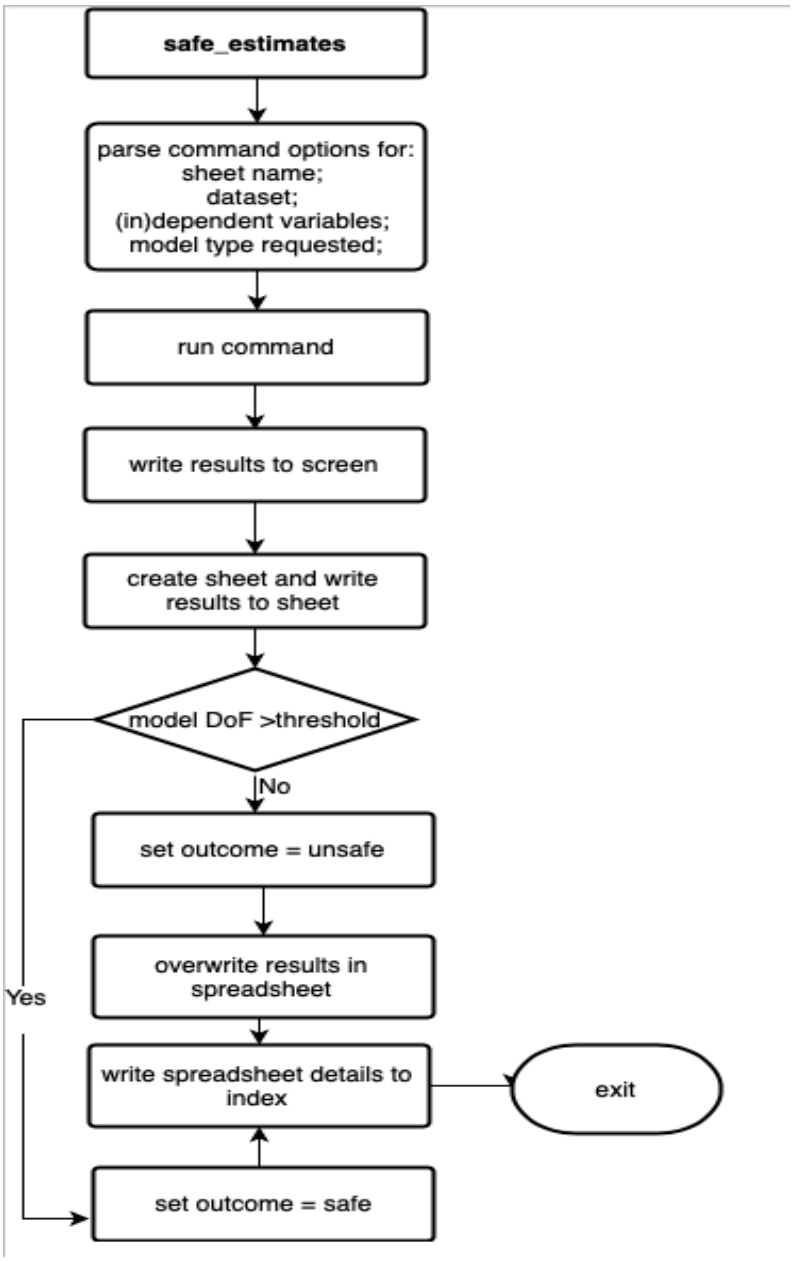

Figure 4: The safe_estimates workflow

Safe Graph

In the current version, all graphs are considered to need review. The information of some types of

graphs (notably histograms and bar charts) can be presented as a table and analysed as such (not yet implemented).

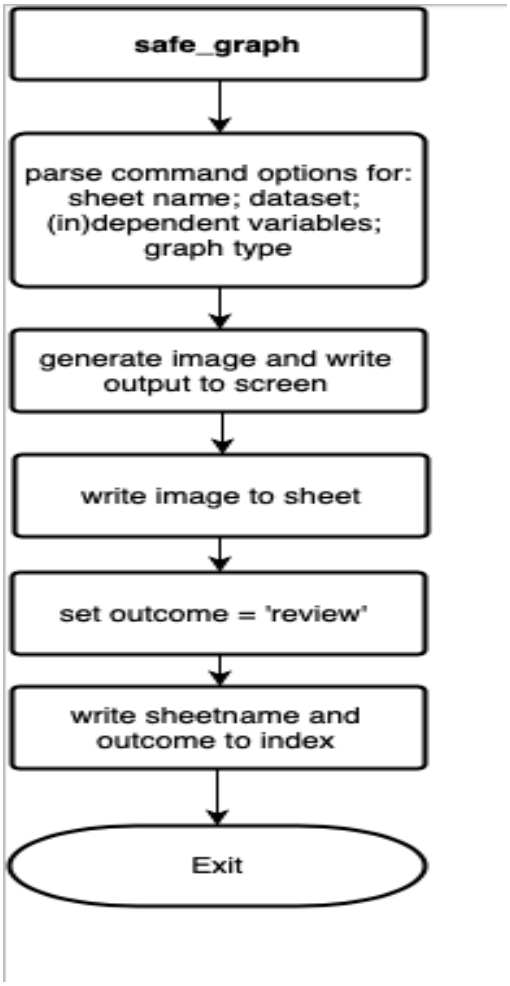

Figure 5: The safe_graph workflow