ReStoRunT: Simple Recording, Storing, Running and Tracing changes in Spreadsheets[1]

Wolfgang Müller[2], Lukrécia Mertová[2]

Abstract: In addition to the ubiquitous *big data*, one key challenge in data processing and management in the life sciences is the *diversity* of *small data*. Diverse pieces of small data have to be transformed into standards-compliant data. Here, the challenge lies not in the difficulty of single steps that need to be performed, but rather in the fact that many transformation tasks are to be performed once or only a few times. This limits the time that can be put into automated approaches, which in turn severely limits the verifiability of such transformations. As much of the data to be processed is stored in spreadsheets, within this paper we justify and propose a lightweight recording-based solution that works on a wide variety of spreadsheet programs, from Microsoft Excel to Google Docs.

Keywords: Provenance; Harmonisation; Spreadsheets

1 Introduction

One of the challenges of real-life data harmonisation in the life sciences is the implementation of standards in everyday work. The challenge lies in the fact that research needs to be flexible and fast, while in the end, one needs reliable data with known semantics. This is the gist of the FAIR principles [Wi16] - Findability, Accessibility, Interoperability and Reusability, which depend mostly on the known semantics of the data.

The semantics of the data is typically conveyed in one of three ways

1. Annotation to ontologies (for example, using web standards like the Resource Description Framework [CK04])
2. Description via markup languages (using SBML [Hu03], for example)
3. Via location in a spreadsheet (as done by many MIBBI [FA22] standards that provide mandatory sets of attributes and sometimes even precise file formats to be filled)

In the latter two cases, the semantics is not conveyed via an ontology but rather via the documentation of the respective formats.

[1] Supported by the Heidelberg Institute for Theoretical Studies and the Klaus Tschira Foundation, as well as MESI-STRAT. MESI-STRAT has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 754688.

[2] Heidelberg Institute for Theoretical Studies — HITS gGmbH, 69118 Heidelberg, Germany wolfgang.mueller@h-its.org

doi:10.18420/BTW2023-57

CSV files and spreadsheets play an essential role here. They are used as lightweight databases with bespoke data models and are transformed towards standard formats to convey semantics by adhering to a pre-defined structure.

As a matter of efficiency, scientists typically use formats compatible with the machines and the software they are using. They are usually different to standard formats such as MIBBI formats. Scientists design their everyday formats to be simple to use with their machines and their software. Often the formats are organized around one cut-and-paste operation from a key proprietary software. This way of action reduces errors and minimizes time for an outcome.

Standards like MIBBI provide mandatory sets of attributes or concrete file formats for data files. However, scientists are left alone with the question of how to bring their day-to-day files into a common format (*e.g.* for one partner or the local lab) or a mandatory format (*e.g.* a standard format for a data publication).

Both transformation tasks are indeed very similar. They mainly differ in the number of files concerned. Long-term experience with tools like RightField [Wo11] or openRefine [te22] showed that a given internal file format is typically used only a couple of times, a common format is used a couple of tens of times, and finally, the mandatory format is used thousands of times.

The difficulty in this setup does not lie in the transformations themselves. They are mostly based on simple operations on single values (such as moving a value to another cell) or tables (such as the transposition of a matrix or a permutation of columns). The difficulty lies in the combination of functional and non-functional requirements that are hard to fulfil via the typical approach, *i.e.* writing and deploying complex software.

As stated above, most of the transformations to be written pertain to comparatively few files. As a consequence, one has the following alternatives.

- Perform the transformation fully manually. However, a manual transformation is hard to check. Errors that distort the meaning of the measurement may go undetected and are hard to verify after the fact. Tracking changes needs additional software.
- Write a transformation via a program, be it Python [VD09], R [R 22], or workflow systems like KNIME [Be09]. This has the following potential drawbacks:
  - There is more work needed for testing the code than the work needed to do the transformation itself. This is frustrating, but too little testing may lead to later undetected errors.
  - For a multitude of formats there will be a multitude of pieces of transformation software. Thereby it becomes a challenge to keep track of which input lead to which output using which transformation software.

These problems call for the following:

- A recording-based solution. Creating the transformation software should be as simple as doing the transformation by hand.
- There must be a trace of both source and destination of the transformation. In addition, the software used for transformation should be recorded within the workbook that contains source and transformed data.
- Ideally, the methods used should be platform-independent and should work with as many spreadsheet systems as possible.

In building the solutions, it has to be kept in mind that there are two types of users in most realistic scenarios: (i) *the data steward*, *i.e.* an experienced user whose focus is on data quality. They are typically able to choose their toolchain for performing data transformation. (ii) *the end user*, *i.e.* the scientist who is generating the data and has to provide standards-compliant data. Typically they have the following challenges:

- They are restricted in the tools they can use due to security concerns. The machines are often managed by the institute, which makes it hard to install plugins and add-ons (*e.g.* Excel), install scripts based on languages not yet installed as well as new executables.
- They are restricted in the use of cloud services due to security concerns. Data paths are carefully monitored, and sending early experimental data to a cloud service is discouraged, for example.
- They are restricted in the time they can invest into tools that do not directly increase their chances of getting a paper accepted. It means tools should be easy to use and cannot *e.g.* expect dexterity or an advanced level of long-term concentration for their use.
- Much preliminary data exchange is still done via mail. This favours methods that easily pass antivirus software (*i.e.* no macros). This also favours methods that enable sending related data in one single file as opposed to having to send a collection of files.

ReStoRunT (Record, Store, Run, Trace transformations in Excel sheets) addresses the needs expressed above. It is a recording-based solution that comes in two flavours, (i) an operating protocol that can be performed manually by experimentalists and already captures most of the advantages of ReStoRunT. (ii) A collection of small Python scripts (to be extended) in case the use of Python is possible. We took Python, as it is a frequently used programming language, also in the biological context. In both cases, the original data and the transformed data stay together in one file, enabling easy sending.

Within this paper, we first describe the key properties of MS Excel that play a role later. We then describe a toy example that we solve via ReStoRunT. Afterwards, we describe some software tools that simplify the use of ReStoRunT and then compare the outcome to the state of the art. This is followed by a summary and an outlook on future work.

2 Key properties of Spreadsheets and Workbooks

According to Wikipedia [Wi23], the first product introducing the concept of spreadsheets that auto-update was VisiCalc in 1979. It enabled the interactive laying out of data in a table containing cells and combining these cells using formulas. A formula in a cell was able to aggregate information from other cells, such as summing them up. The key innovation was a simple, intuitively graspable way to update the cells when a dependent cell was changed.

However, Lotus 1-2-3, an early successor, advertised already in 1983 that it had functionality for using Lotus sheets as a simple database. Excel, starting in 1985, offered the same. So, it does not come as a surprise that scientists soon took up using Excel as a simple database. And —while database scientists reserve the term database for software that has other properties, such as a well-defined data model— the popularity of spreadsheets as makeshift databases are due to their ease of use and flexibility. The present tool tries to alleviate some of the drawbacks of use of Excel as a database.

2.1 Definitions

Within this paper, a spreadsheet S is viewed as an n-dimensional arbitrarily large matrix:

$$S = \begin{pmatrix} A1 & B1 & C1 & \dots & Z1 & AA1 & AB1 & \dots \\ A2 & B2 & C2 & \dots & Z2 & AA2 & AB2 & \dots \\ An & Bn & Cn & \dots & Zn & AAn & ABn & \dots \end{pmatrix} \tag{1}$$

Each element of a spreadsheet is called a cell, and it contains data of an arbitrary type. The format of a cell determines how it is interpreted. Numbers adhering to the locale are recognized as such (*e.g.* $1,2$ in Germany corresponds to 1.2 in the UK). Formulas are expressions that start with an '=' sign.

The cell is uniquely identified by a cell address (or location), which consists of a sheet identification, a column letter, and a row number. For example, cell c has a cell location $A6$ in the spreadsheet S. Formally written as $S!A6$, where ! is a delimiter.

Formulas can also reference cells, so $= A1 * B1$ will be the value obtained by multiplying the number in cell $A1$ by the number in cell $B1$.

Addresses in cells are implicitly relative. So if the formula $= A1 * B1$ is written into the cell $C1$, copying the data from $C1$ to $C2$ will change the formula to $= A2 * B2$. This is very useful for performing the same operation on numerous cells, *e.g.* multiplying the price by the number of items or similar. Sometimes this is unwanted, *e.g.* when applying the same tax rate to multiple items. For this purpose, it is possible to reference cells fixedly, $= \$A\$1 * B1$ would become $= \$A\$1 * B2$ upon copying it to $C2$.

In this paper, we propose a *ReStoRunT copy sheet* C of a source sheet S with n lines and m columns, which is is a matrix with cells $C!Address(l, c)$ $1 \leq l \leq n$, $1 \leq c \leq m$, where

$Address(l, c)$ denotes the string consisting of column letter and line number for line l and column c, and each cell $C!Address(l, c)$ is a reference to $S!Address(l, c)$.

The formula = $A$1 references the content of the cell $A1$. = α!$A$1 denotes the content of the cell $A1$ in the sheet α, = α!$A$2 the cell $A2$, and so forth. However if α!$A$2 is empty, = α!$A$2 is not shown and treated as an empty cell, but as 0! This needs to be filtered out, by an if statement, yielding the following formula:

$$= if(\$A\$2 = '''';'''' ; \$A\$2) \tag{2}$$

The copy sheet thus consists of such a formula in each cell.

We call *ReStoRunT transformation sheet* of S a *ReStoRunT copy sheet* of S that has been modified, *e.g.* by moving cells or deleting cells or adding content such as names and labels. A relationship between the source sheet and the ReStoRunT transformation sheet can be viewed as a transformation function applied on the source sheet, returning the ReStoRunT transformation sheet.

Note: Two cells in the ReStoRunT transformation sheet can reference the same cell in the source sheet.

3 ReStoRunT by example

Within this section, we will describe ReStoRunT via an example that covers the inner workings of ReStoRunT and give an insight into the outcomes.

3.1 The transformation task

In the scenario, Alice and Bob have agreed on a common format. Denoting values measured for two enzymes (called E1, E2), measured at time points 5, 10, 15, and 20 minutes, and as the measurement can (at times) vary by batch, the batch number is noted. Only Alice needs an average between the experiments for each time point. Each measurement value is accompanied by the possibility to make notes. A hypothetical file may look like the following. The numbers have been picked, of course, so that the reader can easily see how the transformation moves cells:

Batch#	55			
time(min)	E1	E2	Average	Notes
5	11	21	16	Note1
10	12	22	17	Note2
15	13	23	18	Note3
20	14	24	19	Note4

Alice needs this data form.

time (min)	5	10	15	20
E1	11	12	13	14
E2	21	22	23	24
Notes	Note1	Note2	Note3	Note4
Batch # gel	55			

Bob has this data form.

As a consequence, Bob needs to apply transformations in order to share his data with Alice.

3.2 Transformation without ReStoRunT

Imagine that it is a one-off exchange of data. Alice needs Bob's data in her format (to feed it into some software or some agreed-on standard format), but currently, there is only one file, and Bob does not want to waste time before knowing there are more files of the same kind.

So, Bob transforms his data manually. In MS Excel, the easiest way to do this is to cut and paste "special", and transpose the data on the way. So he marks data at the *first* line down to the *fifth*, takes an empty sheet and pastes them into that empty sheet, choosing to *transpose* the matrix, to the *second* line of the sheet (Left Table). We replace the empty column with line averages using the formula `=AVERAGE(B1:C1)` (Right Table).

time(min)	E1	E2		Notes
5	11	21		Note1
10	12	22		Note2
15	13	23		Note3
20	14	24		Note4

The transformation of Bob's table.

time(min)	E1	E2	Average	Notes
5	11	21	16	Note1
10	12	22	17	Note2
15	13	23	18	Note3
20	14	24	19	Note4

We replace the empty column with line averages.

And then we notice that the batch number is missing, which we also have to add on top via two cut-and-paste operations. By this, we have achieved Alice's format.

3.3 Weaknesses purely manual transformation

For this one time, this is the most simple that can be done. The goal is met without any overhead. However, if there is any doubt about the accuracy of the transformation ("Did Bob *mispaste*?"), Alice will have to check the original file, which may still be somewhere on Bob's hard disk.

It would have been simpler to verify, had Bob used a script or a computational workflow to modify the data. However, for a one-time transformation spending (in real-life-sized cases) several hours for preparing and testing the script would have been prohibitive.

With the ReStoRunT approach, a couple of minutes of additional work in preparing the sheet will suffice, and the rest will work as before. And the result will be a workbook that (1) contains Bob's original data sheet, and (2) contains a sheet that holds the data in Alice's format. It is a ReStoRunT *transformation sheet* This sheet contains the information in a

way that (3) enables each cell to trace the origin into Bob's sheet. And finally, (4) new Bob format sheets can be transformed in the same way into Alice format sheets, reusing the sheet described in (2).

3.4 Copy sheet and transformation sheet

In the previous section, we have described our goal. Creating a *transformation sheet* that contains the data in Alice's format and that can be reused to transform other data in Bob's format into Alice's format.

We do so by creating a copy sheet and then modifying it manually.

As described above *ReStoRunT copy sheet* β of α is a sheet, where each c in β references the cell in the same line and column in α using formula 2 in section 2.1.

3.5 Creating a transformation sheet by manually applying a sequence of changes

We call Bob's sheet α, and its ReStoRunT copy sheet β. In section 3.2, Bob has applied his changes to α. Now we just apply the same changes to the copy sheet β, instead. The result will just *look* the same. We omit showing the table for brevity.

We have turned the copy sheet into a *transformation sheet* that shows the transformed data, and which —as shown in the next sections– embodies the transformation in a reusable manner.

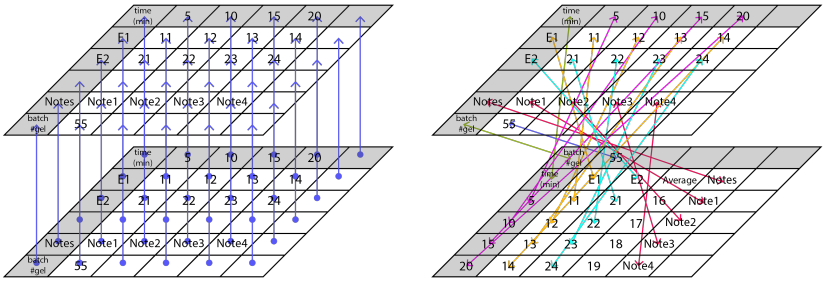

Fig. 1: Right: A ReStoRunT copy sheet referencing the original. Left: A ReStoRunT transformation sheet.

3.6 Tracing the result back to the source

Now imagine that the value 12 is in doubt. *We want to know, is there a copy-paste error?* The value 12 is in cell $\beta!B4$. Looking into the cell, the cell $\beta!B$ contains the formula = $\alpha!\$C\$2$

whose value is 12, as desired. The headers in the sheet α appear to be correct. So, after looking at the formula, we can tell where the value of $\beta!B4$ comes from. The same applies to all other cells in the sheet β.

In other words, the sheet β now contains all references to the original values in α *and* it meets Alice's format requirements. The workbook containing both α and β contains both the original values in the original format as well as the transformation. This also works with single-cell formulas and with many multi-cell formulas.

3.7 Rerunning the transformation sheet on a new data sheet

Now, assume that the data exchange between Alice and Bob has been successful. Bob has the sheets α and the transformation sheet β. He now has additional data α' that he would like to turn into Alice's format.

All he needs to do is the following two steps:

1. Create a copy β' of β that copies all the formulas. Each formula references a cell in α
2. Now do a replace on all cells in β': Replace references to α by references to α'. This can be done via a simple string replacement on all formulas. Now all these formulas reference the corresponding cells in α'.

So, by one copy and one search/replace operation, the same transformation was applied to another sheet in Bob's format. This is suitable for small numbers of workbooks and sheets.

4 Software supporting ReStoRunT

Creating a copy sheet β for a given sheet α in a workbook appears to be the most tedious and error-prone step. However, it suffices to create a workbook that contains an empty sheet α and an $n \times m$ sheet β that is a copy sheet of the empty sheet α. β can then be used as a copy sheet for any non-empty sheet of size $n \times m$ or below.

To further reduce the manual work, e created some lightweight Python tools that simplify using ReStoRunT. We chose Python as a language that is widely accepted and installed. The software is small and open source in order to invite checks by its users. [MM22] contains a repository with the following tools:

```
ReStoRunTify --infile f.xslx --outfile g.xslx
```

reads `f.xslx`, adds ReStoRunT copy sheets for each sheet in f and writes the resulting workbook to `g.xslx`.

```
IsolateReStoRunTsheet --infile f.xslx \
   --tobeisolated "TestSheet" --outfile isolated.xslx
```

Takes ReStoRunT-TestSheet from `f.xslx` and creates a workbook that contains just ReStoRunT-TestSheet and an empty TestSheet. We need the empty TestSheet, as without such a sheet, all the references in ReStoRunT-TestSheet will be broken and replaced by an error string.

```
ApplyReStoRunTsheet --infile f.xslx --sheetfile g.xslx \
         --destinationsheet "Sheet 2" --outfile o.xslx
```

takes the first ReStoRunT sheet in the sheetfile (`g.xslx`) and applies it to the sheet `--destinationsheet Sheet 2`, and then writes out the resulting workbook to the `--outfile o.xslx`.

5 Advantages and limitations of the ReStoRunT approach

Using simple and well-known means, we have reached a useful way of storing Excel transformations for reuse in small series. These transformations are stored within the workbook and are platform-independent. The representation can be used to create other software for larger series of documents. This is our priority in future work on this topic.

MS Excel has the functionality to trace back formula references to their origins. That makes ReStoRunT more useful, as one can see *visually* which cells depend on which other cells.

ReStoRunT works for arbitrarily large, finite-sized sheets. It is applicable for all use cases where the maximum size of the matrix to be transformed can be determined beforehand. In this paper, we described the transformation as the translation of cells. But also normalisation and other formulas that concern a small, finite number of cells (like the average in our example) are something that is tackled using ReStoRunT.

ReStoRunT uniquely works on the layout of sheets, so far, it does not make use of labels or other content within the sheet.

We see as the main limitations (i) that very large numbers of cells will slow down Excel and (ii) that there are some Excel area functions that make it hard to trace back. For example: Sorting a column will yield results, but it will be hard to trace back which value *really* was the third biggest value in a cell set containing 30.000 cells. This can be countered by cascading ReStoRunT sheets thereby extending the detail of traces of changes.

ReStoRunT is using basic Excel formulas, RStoRunT works on Google Sheets, LibreOffice, as well as Apple Numbers and Gnumeric.

6 State of the art compared to ReStoRunT

For re-applying changes, MS Excel has a built-in macro recorder. When using it, it is hard to build working code without modifying the recorded VBA macro afterwards. This is problematic, as the recording needs to be done by experimental scientists who cannot practise this task sufficiently to reach proficiency. Also, the intended users cannot be expected to be fluent in VBA. In addition, a drawback of recorded Excel macros is that many spam filters mark .xlsm (Excel with Macro) files as SPAM, as the powerful embedded VBA code poses a security risk. Furthermore, the receiver is asked if they want to run the security risk of using the macro. Especially novices will not be equipped to take this decision. In addition to that, the resulting transformation code is platform dependent, as it is expressed in Visual Basic for Applications (VBA). Only through an analysis of a given macro a proficient reader will be able to find out what field was the source of a change. The reader will have to *invert* the operations done while recording the macros to find the source of data.

In contrast, one click on the formula in a ReStoRunT sheet will show which cells in the original data sheet contributed to the current value. Furthermore, ReStoRunT needs only key Excel mechanisms for functioning that function across a wide range of spreadsheet tools, as stated above.

InSituTrac [As13] is a comprehensive Excel add-in for recording changes to Excel files. The purposes are tracing provenance and re-applying changes. The comprehensive solution features visualising types and sequences of changes to Excel sheets. Functionality-wise it goes far beyond ReStoRunT. The Excel add-in centres around a ribbon in Excel that gives access to the recording and exploration functionality.

In contrast, ReStoRunT is cross-platform, packages both original and result in one workbook, and the provenance information can be perused without resorting to any add-in.

Google Sheets [Go] are a cloud service for spreadsheets that provide macro recording and change tracking information. However, recorded macros are not exported alongside an Excel export of Google Sheets. So the relation between Sheet and Macro is lost. Copying workbooks provide a way to get a script into another workbook. However, again the users are asked to take uncomfortable security decisions.

Excemplify [Sh13] was a tool for doing traceable changes to Excel files in the frame of Immunoblot experiments. However, this was a large piece of configurable bespoke software with the problems we described above, *i.e.* it needed too much configuration work for each format change.

openRefine [te22] is a tool built for cleaning dirty data. It is a separate application to be installed in user space. It provides functionality to import sheets and then modify them using point 'n' click as well as multi-cell operations. The traces of such modifications can be recorded, stored, imported, and reused. However, the transformation is not shipped with

the data, and openRefine first imports data into one structure, and then re-exports them. This makes it hard to work with complex workbooks.

Workflow tools such as KNIME [Be09] and Galaxy [Af18] provide rich table functionality. But just as using Python's Pandas library [te20], R [R 22], R tidyverse [Wi19], and another tooling, creating transformers in these tools does not happen by simple recording and needs to be tested to a greater extent than recording based solutions. Some of them also import the table into an intermediate format, thus losing the formatting information of the initial table.

[Wo11] is a tool for adding ontology information to spreadsheets, a complementary approach. It can read workbooks, add hidden sheets with ontology information and then store the sheet. These data can then subsequently be read by other tools. RightField's use is complementary to the tools described above.

To our knowledge, ReStoRunT has its use in the space of Spreadsheet-related tools, being a useful addition because it is simple, not in the cloud, and doing quality control using a ReStoRunT sheet does not need anything beyond standard software.

7 Conclusion

We argued that transforming Excel files and similar spreadsheets is an important task in experimental biological work. The difficulty lies in the fact that one needs many *different* transformations that need to be *traced* and possibly *rerun* several times, but not rerun often enough to warrant a large development or configuration effort.

For this task, we have proposed ReStoRunT, *i.e.* recording and storing transformations such that results can be traced to their origin and finally can be rerun, *i.e.* applied to new data.

ReStoRunT can be used entirely manually as a set of Excel practises or complemented via tooling, of which we present an initial version. We hope to help scientists in sharing their experimental data in a harmonized manner.

Acknowledgements

Müller and Mertová are funded by HITS and the Klaus Tschira Foundation, KTS. Müller had been co-funded by MESI-STRAT. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 754688. Much of this work came from discussions with *Ines Heiland* and *Cecilia Barile* about their needs for Excel transformations and traceability. Another input came from programming work that Mertová and Müller did for ASSR, the Samaritans of Slovakia. We thank Stefan Giulini of ASSR for a guided tour of their needs and their application in real life.

References

[Af18] Afgan, E.; Baker, D.; Batut, B.; van den Beek, M.; Bouvier, D.; Čech, M.; Chilton, J.; Clements, D.; Coraor, N.; Grüning, B. A.; Guerler, A.; Hillman-Jackson, J.; Hiltemann, S.; Jalili, V.; Rasche, H.; Soranzo, N.; Goecks, J.; Taylor, J.; Nekrutenko, A.; Blankenberg, D.: The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. Nucleic Acids Res. 46/W1, W537–W544, 2018.

[As13] Asuncion, H. U.: Automated data provenance capture in spreadsheets, with case studies. Future Generation Computer Systems 29/8, Including Special sections: Advanced Cloud Monitoring Systems & The fourth IEEE International Conference on e-Science 2011 — e-Science Applications and Tools & Cluster, Grid, and Cloud Computing, pp. 2169–2181, 2013, ISSN: 0167-739X, URL: https://www.sciencedirect.com/science/article/pii/S0167739X13000691.

[Be09] Berthold, M. R.; Cebron, N.; Dill, F.; Gabriel, T. R.; Kötter, T.; Meinl, T.; Ohl, P.; Thiel, K.; Wiswedel, B.: KNIME - the Konstanz Information Miner: Version 2.0 and Beyond. SIGKDD Explor. Newsl. 11/1, pp. 26–31, Nov. 2009, ISSN: 1931-0145, URL: http://doi.acm.org/10.1145/1656274.1656280.

[CK04] Carroll, J.; Klyne, G.: Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation, https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, W3C, Feb. 2004.

[FA22] FAIRsharing.org: MIBBI; Minimum Information for Biological and Biomedical Investigations, https://fairsharing.org/3518, [Online, accessed 2022-12-02], 2022.

[Go] Google Workspace, G.: Google Sheets: Online Spreadsheet Editor, URL: https://www.google.com/sheets/about/.

[Hu03] Hucka, M.; Finney, A.; Sauro, H. M.; Bolouri, H.; Doyle, J. C.; Kitano, H.; Arkin, A. P.; Bornstein, B. J.; Bray, D.; Cornish-Bowden, A.; Cuellar, A. A.; Dronov, S.; Gilles, E. D.; Ginkel, M.; Gor, V.; Goryanin, I. I.; Hedley, W. J.; Hodgman, T. C.; Hofmeyr, J.-H.; Hunter, P. J.; Juty, N. S.; Kasberger, J. L.; Kremling, A.; Kummer, U.; Novère, N. L.; Loew, L. M.; Lucio, D.; Mendes, P.; Minch, E.; Mjolsness, E. D.; Nakayama, Y.; Nelson, M. R.; Nielsen, P. F.; Sakurada, T.; Schaff, J. C.; Shapiro, B. E.; Shimizu, T. S.; Spence, H. D.; Stelling, J.; Takahashi, K.; Tomita, M.; Wagner, J.; Wang, J.; Forum, S. B. M. L.: The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics 19/4, pp. 524–531, Mar. 2003.

[MM22] Mueller, W.; Mertová, L.: ReStoRunT GitHub repository, https://github.com/mertova/ReStoRunT, [Online, accessed 2022-12-01], 2022.

[R 22] R Core Team: R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2022, URL: https://www.R-project.org/.

[Sh13] Shi, L.; Jong, L.; Wittig, U.; Lucarelli, P.; Stepath, M.; Mueller, S.; D'Alessandro, L.; Klingmüller, U.; Müller, W.: Excemplify: a flexible template based solution, parsing and managing data in spreadsheets for experimentalists. J Integr Bioinform. 2/10, p. 220, Apr. 2013.

[te20] pandas development team, T.: pandas-dev/pandas: Pandas, version latest, Feb. 2020, URL: https://doi.org/10.5281/zenodo.3509134.

[te22] openRefine team: openRefine, http://openrefine.org/, [Online, accessed 2022-12-01], 2022.

[VD09] Van Rossum, G.; Drake, F. L.: Python 3 Reference Manual. CreateSpace, Scotts Valley, CA, 2009, ISBN: 1441412697.

[Wi16] Wilkinson, M. D.; Dumontier, M.; Aalbersberg, I. J.; Appleton, G.; Axton, M.; Baak, A.; Blomberg, N.; Boiten, J.-W.; da Silva Santos, L. B.; Bourne, P. E., et al.: The FAIR Guiding Principles for scientific data management and stewardship. Scientific data 3/, 2016.

[Wi19] Wickham, H.; Averick, M.; Bryan, J.; Chang, W.; McGowan, L. D.; François, R.; Grolemund, G.; Hayes, A.; Henry, L.; Hester, J.; Kuhn, M.; Pedersen, T. L.; Miller, E.; Bache, S. M.; Müller, K.; Ooms, J.; Robinson, D.; Seidel, D. P.; Spinu, V.; Takahashi, K.; Vaughan, D.; Wilke, C.; Woo, K.; Yutani, H.: Welcome to the tidyverse. Journal of Open Source Software 4/43, p. 1686, 2019.

[Wi23] Wikipedia: VisiCalc, 2023, URL: https://en.wikipedia.org/wiki/VisiCalc, visited on: 01/22/2023.

[Wo11] Wolstencroft, K.; Owen, S.; Horridge, M.; Krebs, O.; Mueller, W.; Snoep, J.; du Preez, F.; C., G.: RightField: embedding ontology annotation in spreadsheets. Bioinformatics 14/27, pp. 2021–2, July 2011.