Machine Learning (2020) 109:307–325
https://doi.org/10.1007/s10994-019-05841-y

Predictive spreadsheet autocompletion with constraints

Samuel Kolb[1] · Stefano Teso[1] · Anton Dries[1] · Luc De Raedt[1]

Received: 26 November 2018 / Revised: 22 May 2019 / Accepted: 6 September 2019 /
Published online: 25 October 2019

Abstract
Spreadsheets are arguably the most accessible data-analysis tool and are used by millions of people. Despite the fact that they lie at the core of most business practices, working with spreadsheets can be error prone, usage of formulas requires training and, crucially, spreadsheet users do not have access to state-of-the-art analysis techniques offered by machine learning. To tackle these issues, we introduce the novel task of *predictive spreadsheet autocompletion*, where the goal is to automatically predict the missing entries in the spreadsheets. This task is highly non-trivial: cells can hold heterogeneous data types and there might be unobserved relationships between their values, such as constraints or probabilistic dependencies. Critically, the exact prediction task itself is not given. We consider a simplified, yet non-trivial, setting and propose a principled probabilistic model to solve it. Our approach combines black-box predictive models specialized for different predictive tasks (e.g., classification, regression) and constraints and formulas detected by a constraint learner, and produces a maximally likely prediction for all target cells that is consistent with the constraints. Overall, our approach brings us one step closer to allowing end users to leverage machine learning in their workflows without writing a single line of code.

Keywords Spreadsheets autocompletion · Bayesian networks · Constraint learning · Machine learning

Editors: Karsten Borgwardt, Po-Ling Loh, Evimaria Terzi, Antti Ukkonen.

This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. [694980] SYNTH: Synthesising Inductive Data Models).

✉ Stefano Teso
stefano.teso@cs.kuleuven.be

Samuel Kolb
samuel.kolb@cs.kuleuven.be

Anton Dries
anton.dries@cs.kuleuven.be

Luc De Raedt
luc.deraedt@cs.kuleuven.be

[1] KU Leuven, Leuven, Belgium

1 Introduction

Spreadsheets are the workhorse of business and industry. They support a huge user base, composed of end users with widely different goals and degrees of competence (Lawson et al. 2009). Managing to automate the workflow of these users, even partially, will have a significant impact on all sectors of business. This explains the recent outburst of research and applications of artificial intelligence, machine learning and inductive programming on spreadsheets (Gulwani 2011; Gulwani et al. 2015; Kolb et al. 2017; Devlin et al. 2017). For instance, the BigML (https://bigml.com) extension for Google Sheets integrates standard learning algorithms and workflows into spreadsheet interfaces with the goal of lowering the threshold to predictive analysis for laymen. These approaches, however, assume that the user has some degree of technical competence (either for choosing a predictive model or for writing spreadsheet formulas) which is often not the case in practice; indeed the vast majority of spreadsheet users are neither professional programmers nor data scientists (Scaffidi et al. 2005) and cannot write even trivial formulas (Gulwani et al. 2012).

To tackle this issue, we formulate the problem of *predictive spreadsheet autocompletion*, namely the problem of predicting or suggesting the next values that the user wants to enter given a set of tables in a spreadsheet. The basic assumption is that the user may not be entirely competent with spreadsheets or data analysis, although user guidance can in principle be leveraged, if available.

Glancing at any spreadsheet dataset immediately reveals that this is a very hard problem. Cell values can have arbitrary data types, can be very sparse, and can be mutually constrained by unobserved formulas. Most importantly, spreadsheets are used to perform all kinds of tasks, from bookkeeping to data analysis, and so the underlying data generating process (and the corresponding prediction task) can be almost arbitrary. Notice that this problem is beyond the reach of standard spreadsheet applications, which often implement a limited form of "autocompletion" using, e.g., propagation rules. This problem is also significantly more general than missing-data imputation (Van Buuren 2018), which targets individual data matrices, whereas spreadsheets can hold multiple related tables and formulas, and thus have a much less restricted data generating process.

We present a novel approach, PSyChe (Predictive Spreadsheets with Constraints), which casts predictive autocompletion as a constrained probabilistic inference problem. In order to keep the task manageable, it is assumed that the data exhibits regularities and that it is entered in a systematic manner, as is often the case in decision making applications. At a high level, PSyChe is given black-box access to a set of heterogeneous base predictors trained for different tasks, e.g., multi-class prediction or regression.[1] By construction, these base predictors may output mutually inconsistent values for the same cell. Crucially, PSyChe leverages the TaCLe constraint learner to extract spreadsheet formulas and constraints that hold among the observed cells (Kolb et al. 2017). The proposed approach combines the base predictions into a coherent, consistent joint completion of the target cells by reasoning about the confidences in the predictions and the potential dependencies among the cells. This is accomplished by solving a probabilistic inference problem subject to the learned constraints.

Thus PSyChe can assist the user in (1) entering values, by suggesting completions; (2) making use of formulas without typing them, by learning formulas from the data and applying them during inference; (3) automatically obtaining predictions for particular cells.

[1] A variety of specialized predictors—potentially pre-trained on task-specific datasets—can be included for handling specific cases, e.g., dates, email addresses, or simple data analysis pipelines.

Table 1 Two tables in a spreadsheet

Type	Country	June	July	Aug	Total	Profit
Vanilla	UK	610	190	670	1470	YES
Banana	UK	170	690	520	1380	YES
Chocolate	UK	560	320	140	1020	YES
Banana	DE	610	640	320	1570	NO
Stracciatella	UK	300	270	290	860	NO
Chocolate	FR	430	350	?	?	?
Banana	DE	250	650	?	?	?
Chocolate	BE	210	280	?	?	?

Type	ProdTime
Chocolate	60
Banana	40
Stracciatella	70
Vanilla	40

Summarizing, our main contributions are:

1. Introducing the novel problem of predictive spreadsheet autocompletion;
2. A formalization of the problem in terms of joint probabilistic inference under constraints;
3. An implementation based on the TACLE constraint learner, which can deal with formulas and constraints occurring in our setting;
4. An empirical evaluation over (1) real-world spreadsheets taken from the EUSES corpus (Fisher and Rothermel 2005) and the formula-rich benchmark spreadsheets of Kolb et al. (2017); (2) real-world machine learning datasets from OpenML (Vanschoren et al. 2013); and (3) synthetically generated data.

The paper is structured as follows. In the next Section we formalize the problem of predictive spreadsheet autocompletion. Next, we introduce the TACLE constraint learner and discuss our extensions. We present PSYCHE in Sect. 2 and evaluate our implementation in Sect. 5. Finally, we overview the related work in Sect. 6 and then conclude with some final remarks.

2 Predictive spreadsheet autocompletion

Let us introduce *predictive spreadsheet autocompletion* using the simplified spreadsheet in Table 1. It contains information about the sales of particular flavors of ice-cream in different countries and months, as well as information about the production time taken to produce one unit of ice-cream. Now, decisions need to be made about which flavors of ice-cream to retain in which countries, based on the total sales, costs and profitability. However, the top table is incomplete, as some of the values for August are not yet available, which is problematic for the decision making process.

Automatically completing the spreadsheet requires a number of steps: (1) discover the formula H_T stating that Total is equal to the sum of June, July and August; (2) find a predictive model f_A for the column August using the available data; (3) find a model f_P for Profit using the available data; (4) impute the missing values for August using f_A; (5) impute Total using H_T; and (6) impute Profit using f_P.

Formally, the spreadsheet autocompletion problem can be stated as follows:

> ***Given*** *a set of tables in a spreadsheet, a set of heterogeneous predictors* $\mathcal{F}$*, a set of formulas and constraints* $\mathcal{H}$ *learned from the tables, and a range of n target empty cells*, ***find*** *an autocompletion for the target cells, that is, an assignment of values* $v_1, \ldots, v_n$ *to the cells*.

The example above indicates the challenges involved in predictive autocompletion, which we briefly overview:

- First, spreadsheets are very heterogeneous. Imputing a single cell requires to deal with arbitrary data types, be they discrete or numerical. Only a handful of filled cells may be available, and may occur in irregular patterns.
- The values of different cells are often interdependent, because of unobserved formulas and constraints, or because of statistical dependencies. In particular, constraints and formulas are pervasive in real-world spreadsheet usage, and cannot be ignored.
- The dependencies are often scattered across different tables, which need to be (implicitly) joined before prediction can be attempted.
- Finally, the data generation process underlying the observed tables is essentially arbitrary. The actual predictive task may require us to solve classification, regression, ranking, or other more specialized predictive tasks. This information is *not* provided in advance.

To the best of our knowledge, no predictive model can deal with all of these issues, as discussed in Sect. 6.

Of course, we do not aim at solving all of these problems in the present paper. In order to keep the task manageable, we focus on cases where the data exhibits regularities and is entered in a systematic manner. This is often the case in decision making applications. Although our approach does support some forms of pre-processing (e.g. automatically joining related tables, as explained in Sect. 3), we also assume that basic data wrangling and cleaning has been been performed, so that the tables can be read off from the spreadsheet. We mention some tools that automate this step in Sect. 6. Finally, since the prediction task is not given, we assume to have access to a set of heterogeneous "base predictors", which rely on different cues and biases and are suited for the different candidate tasks. These can be either learned on the fly on the current spreadsheet, or be trained on larger datasets in an offline fashion.

Under these assumptions, we frame predictive spreadsheet autocompletion as the problem of combining the base predictions in a manner consistent with the learned constraints. This requires one to take into consideration both the relative performance of the base predictors, as well as the constraints observed to hold in the target tables, in order to avoid completing cells with unfeasible values.

In the following, we introduce PSYCHE, a novel probabilistic approach to spreadsheet autocompletion that leverages these ideas to solve many of the above problems. While doing so, we will assume a basic understanding of directed graphical models (Koller and Friedman 2009) and machine learning for classification and regression. Before discussing PSYCHE, we overview the TACLE constraint learner (Kolb et al. 2017), which lies at the core of our approach, and our extensions to it.

3 Learning formulas and constraints with TACLE

In many applications, repetitive tasks can be automated by specifying a formal model (e.g., computing summations or counts in spreadsheets), but the end users often lack the background

to build a model upfront. Constraint learning facilitates this step by automatically extracting a model from data. Several constraint learning approaches have been designed for learning constraint programs and mathematical optimization models, see, e.g., (Rossi and Sperduti 2004; Bessiere et al. 2005, 2016; Beldiceanu and Simonis 2012). TACLE (Kolb et al. 2017) is an approach specifically tailored for the spreadsheet setting. In particular, it detects formulas and constraints that hold in (complete or partial) spreadsheets, and leverages ideas from the program synthesis literature to significantly speed up the learning step.

Our approach, PSYCHE, uses TACLE to acquire constraints and formulas from the target spreadsheet. These are then used to both predict the value of missing cells (as in our toy example) and to avoid suggesting unfeasible completions. We proceed by discussing TACLE.

The TACLE*constraint learner.* At a high-level, TACLE has access to a set of *constraint templates* that specify which types of formulas it should look for. Every template is made up of three parts: syntax, signature and definition. For instance, the template for column-wise sum has *syntax* $B_2 = \text{SUM}_{\text{col}}(\boldsymbol{B}_1)$. Its *signature* specifies that B_2 is a column, $\boldsymbol{B}_1$ a set of consecutive columns, that their heights match, and that all cells should contain numerical values. By assigning specific (ranges of) columns to $\boldsymbol{B}_1$ and B_2, TACLE obtains an actual constraint, which encodes the fact that the ith cell assigned to B_2 is equal to the sum of the ith row of the columns assigned to $\boldsymbol{B}_1$. The *definition* computes the actual output from the inputs. TACLE has support for a multitude of widely used spreadsheet constraint templates applying to both rows and columns. Constraint learning then amounts to instantiating the available templates on all applicable ranges of rows and columns, and verifying whether the signature and definition hold.

In order to deal with the large number of possible instantiations, TACLE searches only over *tables* of equally-sized rows and/or columns, and *blocks*, which are continuous ranges of rows or columns with the same type (e.g., numeric or textual) and individual vectors (a row or a column) of type-consistent cells. TACLE leverages constraint programming to implement the enumeration step, as well as some clever strategies to prune assignments to the templates, significantly speeding up the search.

TACLE natively includes templates for many formulas that frequently appear in real-world spreadsheets, including: arithmetic operations on rows (e.g. $B_2 = \text{SUM}_{\text{row}}(B_1)$), columns ($\text{SUM}_{\text{col}}$), and element-wise operations ($+$, $-$, $\times$, $/$), as well as more complex operations such as group-and-sum and group-and-count (SUMIF and COUNTIF, respectively), lookup, aggregates (MIN, MAX, AVERAGE), as well as proper constraints (e.g. ORDERED), among others. See Table 2 for a selection of constraints supported by TACLE.

We remark that TACLE can learn both formulas and constraints. The distinction is as follows. Formulas (e.g., SUM) are functional constraints, meaning that the output value is uniquely determined by the input arguments. Instead, "proper" constraints (e.g. ORDERED) instead are not functional, as they admit multiple feasible output values. Both formulas and constraints appear frequently in real-world spreadsheets, and our approach is designed to leverage both.

4 Autocompletion with PSYCHE

In this section, we present PSYCHE, our approach to predictive spreadsheet autocompletion. We start off by discussing the simplest case of autocompletion, namely imputing a single cell, and later on generalize this to the full setting with multiple cells.

Table 2 A *selection* of formulas and constraints learned by TACLE and their intuitive meaning and signature. Adapted from Kolb et al. 2017

Syntax	Meaning
$B_3 = B_1\ \{+, -, \times, /\}\ B_2$	Element-wise arithmetic
B_1, B_2, B_3 are all numeric and have the same length	
$B_2 = \{\text{SUM, MIN, MAX, AVERAGE}\}_{\text{col}}(\boldsymbol{B}_1)$	Column-wise aggregates
B_1, B_2 are numeric; B_1 has at least 2 columns and as many rows as elements in B_1	
$B_2 = \{\text{SUM, MIN, MAX, AVERAGE}\}\text{IF}(B_{fk}, B_{pk}, B_1)$	Group-by-like aggregation
B_{fk}, B_{pk} are discrete; B_1, B_2 are numeric; $\text{ALLDIFFERENT}(B_{pk})$	
B_1, B_{fk} and B_{pk}, B_2 have the same length and orientation	
B_1, B_{fk} have the same table; B_{pk}, B_{fk} have have different tables	
$B_2 = \text{LOOKUP}(B_{fk}, B_{pk}, B_1)$	Lookup mapping
B_{fk}, B_{pk} are discrete; B_{fk}, B_{pk} have the same type; $\text{FK}(B_{fk}, B_{pk})$	
B_1, B_{fk} and B_{pk}, B_2 have the same length, table and orientation	
$\text{SERIES}(B)$	Values increase by 1
B is integer-typed and a permutation of values 1..length(B)	
$\text{ORDERED}(B)$	Values are monotonically increasing

As for notation, we will represent all cells by random variables (RVs). These will be written in upper case (e.g. X), and their values in lower case (x). Ordered sets of RVs will be written in bold $\boldsymbol{X}$, and the ith element of a set will be indicated as X_i. $P(x)$ will often be used as a shorthand for $P(X = x)$.

4.1 Autocompleting a single cell

We start off with the simplest case of autocompletion, which consists of determining the most likely value v for a single cell Y,

$$\text{argmax}_v \ \ P(Y = v \mid \boldsymbol{X} = \boldsymbol{x}, \mathcal{H}) \tag{1}$$

given:

1. A set of *input cells* $\boldsymbol{X}$ with values $\boldsymbol{x}$, which will typically correspond to a selection of cells from rows and columns connected to Y, e.g., all the observed cells in the same row as Y;
2. k heterogeneous *base predictors* $\mathcal{F} = \{f_1, \ldots, f_k\}$, where each f_j is a predictive model that outputs a candidate value (or a distribution over candidate values) for Y starting from a set of input cells $\boldsymbol{X}^j \subseteq \boldsymbol{X}$;
3. ℓ *hard constraints* $\mathcal{H} = \{H_1, \ldots, H_\ell\}$ on the cells in $\boldsymbol{X}$; the hard constraints exclude certain value assignments to $\boldsymbol{X} \cup Y$, e.g., an ORDERED constraint requires that the values in the associated column be ordered from largest to smallest or vice-versa;

In the following, we discuss how the conditional distribution $P(Y = v \mid \boldsymbol{X} = \boldsymbol{x}, \mathcal{H})$ is modeled by PSYCHE and how the above components contribute to it. Before proceeding, however, we make two important remarks.

First, as mentioned above, some constraints such as SUM are functional in the sense that if $n - 1$ values and the total are given, then SUM uniquely determines the missing value.

Further, constraints such as MAX, although not strictly functional, are still formulas, i.e., the result is uniquely determined by the inputs. We model these constraints as base predictors rather than as constraints.

Second, all the observed cells in the spreadsheet that are not input cells $\boldsymbol{X}$ are used by PSYCHE for estimating different parameters. We indicate these cells as $\boldsymbol{D}$. For instance, if Y is a target cell and the corresponding input cells $\boldsymbol{X}$ are all in the same row as Y, then $\boldsymbol{D}$ includes the observed cells in all the other rows. More specifically, PSYCHE uses the cells in $\boldsymbol{D}$ to learn the hard constraints with TACLE, as well as other parameters appearing in the conditional distribution. In our experiments, we also use $\boldsymbol{D}$ to fit the base predictors.

The base predictors Let us have a closer look at the base predictors $\mathcal{F}$. We consider both deterministic predictors, such as decision trees (Breiman 2017), and discriminative probabilistic predictors, such as logistic regression (Bishop 2006), although any other predictor can be used. Multi-label predictors will be discussed in Sect. 4.2. Each base predictor f_j is represented as a conditional distribution $P(V_j = v \mid \boldsymbol{X}^j)$, where v is the value assigned to the target cell by f_j and $\boldsymbol{X}^j \subseteq \boldsymbol{X}$ are the input cells that f_j actually used.[2] If f_j is deterministic, it outputs a single value, and therefore:

$$P(V_j = v \mid \boldsymbol{X}^j = \boldsymbol{x}^j) = \delta\left\{v = f_j(\boldsymbol{x}^j)\right\}$$

Here $\delta\{\cdot\}$ is the indicator function which returns 1 if v matches the output of f_j and 0 otherwise. On the other hand, if f_j is probabilistic, then it outputs the above conditional probability automatically.

PSYCHE expects the base predictors to be given. In practice, they can be obtained by fitting any classifier or regressor on the (non-input) observed cells $\boldsymbol{D}$ in the target spreadsheet. Our experiments follow this setup and show that it works well, provided that enough training cells are available, see Sect. 5. When this is not the case, an alternative is to employ predictors pre-trained on large external corpora by adapting them to the target spreadsheet on-the-fly. We leave a proper analysis of this more elaborate pipeline to future research.

Calibrating the base predictors The base predictors can, of course, make mistakes. In many cases, the error pattern is predictable. For instance, in class-unbalanced tasks (e.g. book databases, where the same publisher is repeated for every one of its books), the base predictors may consistently over-predict the majority class.

In order to correct for this, we include a calibration step that aims at fixing systematic errors by redistributing probability mass among candidate values. It does so through a conditional distribution $P(V'_j \mid V_j, \boldsymbol{X}^j)$, where V_j is the output of the predictor and V'_j is the calibrated prediction. For discrete predictions, calibration is implemented as a categorical distribution, for maximum flexibility; continuous values are addressed in Sect. 4.3. The dependency on the input cells is ignored, i.e., $P(V'_j \mid V_j, \boldsymbol{X}^j) = P(V'_j \mid V_j)$, as to reduce the number of parameters.

PSYCHE estimates the categorical distribution from the observed cells $\boldsymbol{D}$ in a robust manner via cross-validation. More specifically, the dataset $\boldsymbol{D}$ is first split into folds. Next, since in $\boldsymbol{D}$ all of the variables are observed, it is easy to count how many times $n_{v|v'}$ the true label $V'_j = v'$ occurs given that the prediction is $V_j = v$. The counts are then averaged over folds to obtain $\tilde{n}_{v'|v}$. Laplacian smoothing by a small constant c is then applied to the average counts to obtain the conditional probability table $P(V'_j = v' \mid V_j = v) := (\tilde{n}_{v'|v} + c)/\sum_{v''}(\tilde{n}_{v''|v} + c)$. Such cross-validated averages are known to provide robust generalization guarantees (Elisseeff

[2] That is, V_j is conditionally independent of $\boldsymbol{X} \setminus \boldsymbol{X}^j$ given $\boldsymbol{X}^j$; in other words, for all possible values of $\boldsymbol{X}$, we can always write $P(V_j \mid \boldsymbol{X}) = P(V_j|\boldsymbol{X}^j)$ (Koller and Friedman 2009).

and Pontil 2003). Notice that this procedure remains the same regardless of how the base predictors were obtained, e.g., fitted on the current spreadsheet or pre-trained on external data.

Combining the calibrated predictors The next step is to use the distributions of the calibrated predictions $P(V'_j = v \mid V_j)$, $j = 1, \ldots, k$, to model the cell value distribution $P(Y = v \mid \boldsymbol{X} = \boldsymbol{x}, \mathcal{H})$ appearing in Eq. 1. For now, let us assume that no constraints are given, so that $\mathcal{H}$ can be ignored.

In order to mix together the various predictors, we opt for a mixture of experts (Jordan and Jacobs 1994; Bishop 2006), which has the form[3]:

$$P(Y = v \mid \boldsymbol{X}) := \sum_{j=1}^{k} P(V'_j = v \mid \boldsymbol{X}^j, Z = j) P(Z = j \mid \boldsymbol{X}^j)$$

Here Z is a categorical random variable taking values in $\{1, \ldots, k\}$. This results in a very flexible model. However, as the Z are latent variables, we will need to resort to expectation-maximization (EM) during estimation. Since EM can have an impact on efficiency and thus hinder reactivity in interactive scenarios, we further simplify the model by assuming that Z is marginally independent from $\boldsymbol{X}$. The distribution becomes:

$$P(Y = v \mid \boldsymbol{X}) := \sum_{j=1}^{k} w_j P(V'_j = v \mid \boldsymbol{X}^j) \tag{2}$$

The weights $w_j := P(Z = j)$, $j = 1, \ldots, k$ differentially allocate trust to predictors. The resulting model is equivalent to weighted voting (or average opinion pooling) (Clemen and Winkler 1999).

We model the distribution $P(Z = j)$ by following two principles. First, if a predictor performs badly, it should be effectively disabled by setting the corresponding weight w_j close to zero. In practice, we observe experimentally that this allows PSYCHE to effectively ignore base predictors that attempt to predict a cell using unrelated columns. Second, if there are multiple good predictors, they should be allowed to collectively contribute to the overall prediction by voting.

Following these principles, a simple implementation would set the weights $w_j \propto \eta \mathrm{acc}_j$ to be proportional to the estimated accuracy of the jth predictor; $\eta \in \mathbb{R}$ is a hyper-parameter. The accuracy is easy to evaluate on the training set $\boldsymbol{D}$ using cross-validation, as done for the calibrator. However, this strategy is quite lenient and may associate relatively large weights to bad predictors. For this reason, we prefer a multiplicative variant, where the weights are set to $w_j = w'_j / (\sum_\ell w'_\ell)$ for $w'_j = (1 - \eta)^{d(1-\mathrm{acc}_j)}$, d is the number of rows in $|\boldsymbol{D}|$ (where the target cell is observed) and $\eta \in (0, 1)$. This is consistent with work on prediction from expert advice and the hedge algorithm (Arora et al. 2012). The net effect is that the weight of bad predictors decreases exponentially fast and the vast majority of the probability mass is allocated to the better predictors. Of course, if too little data are available, the weights can be simply fixed to $\frac{1}{k}$.

Notice that these combiners are all symmetric (in the sense that any predictor can in principle be preferred) and satisfy the principle of "conditionally" independent alternatives", meaning that the predictions of alternative base predictors (and their calibrated counterparts) are independent given $\boldsymbol{X}$. This can be easily verified in Fig. 1.

Inference with no constraints We are finally ready to solve Eq. 1 when no constraints are present. Let Y be the value to be filled in. In order to select the most likely prediction, we pick the value with the largest probability, namely $\mathrm{argmax}_v \; P(Y = v \mid \boldsymbol{X} = \boldsymbol{x})$. The value

[3] To see the equivalence to mixture of experts, notice that, although notationally different, Y and V'_j refer to the same information, namely the value of the target cell.

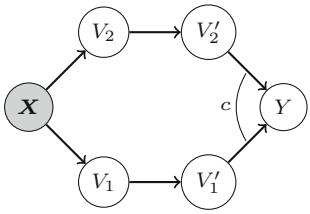

Fig. 1 Directed graphical model for one cell and two predictors. Circles represent RVs, shaded circles represent evidence. Edges denote conditional dependencies. Here c indicates the voting combiner. We represent the input cells as a single observed vector-valued node $\boldsymbol{X}$, for simplicity

distribution for each predictor f_j is obtained by marginalizing the raw predictions V_j:

$$\begin{aligned} P(V_j' = v' \mid \boldsymbol{X}) &= \textstyle\sum_v P(V_j' = v' \mid V_j = v, \boldsymbol{X}) P(V_j = v \mid \boldsymbol{X}^j) \\ &= \textstyle\sum_v P(V_j' = v' \mid V_j = v) P(V_j = v \mid \boldsymbol{X}^j) \end{aligned}$$

where we replaced $P(V_j' = v' \mid V_j = v, \boldsymbol{X})$ with $P(V_j' = v' \mid V_j = v)$. The two factors are exactly the calibrator and the base predictors discussed above. Applying the mix combiner, we obtain the distribution of the selected value Y:

$$P(Y = v \mid \boldsymbol{X}) = \textstyle\sum_{j=1}^{k} \sum_v w_j P(V_j' = v \mid V_j = v) P(V_j = v \mid \boldsymbol{X}^j) \tag{3}$$

Summarizing, the raw value distributions $P(V_j \mid \boldsymbol{X})$ are obtained in a black-box fashion from the base predictors f_j, $j = 1, \ldots, k$; the calibrators $P(V_j' \mid V_j)$ correct the individual predictions of the corresponding predictor; finally, the top-level combiner $P(Y \mid \boldsymbol{X})$ allocates trust or responsibility to competing predictors and produces a unique distribution over candidate values. Figure 1 illustrates the complete one-cell model as a directed probabilistic graphical model.

Inference under constraints We are left with explaining the role of the hard constraints. In practice, the constraints are obtained by running TACLE on the target spreadsheet. If any are found, they are used during inference to eliminate all candidate values inconsistent with them. The resulting distribution is normalized accordingly. More formally, the probability of $Y = v$ under constraints $\mathcal{H}$ is:

$$P(Y = v \mid \boldsymbol{X}, \mathcal{H}) = \tfrac{1}{Z_{\boldsymbol{X}}} P(Y = v \mid \boldsymbol{X})$$

if v and $\boldsymbol{X}$ are feasible with respect to $\mathcal{H}$, and zero otherwise. Here $Z_{\boldsymbol{X}}$ is a normalization constant that amounts to $Z_{\boldsymbol{X}} = \sum_{v \text{ and } \boldsymbol{X} \text{ are feasible w.r.t. } \mathcal{H}} P(Y = v \mid \boldsymbol{X})$. In practice, implementations may track the unnormalized distribution instead, for efficiency. Since normalization is a monotonic transformation, applying or ignoring it leaves the optimal autocompletion unchanged.

4.2 Autocompleting a spreadsheet

We have everything together to discuss the more general setting of autocompleting n cells. Let Y_i be the variable representing the value of cell i. As above, we are interested in finding the most likely autocompletion:

$$\operatorname{argmax}_{v_1, \ldots, v_n} P(Y_1 = v_1, \ldots, Y_n = v_n \mid \boldsymbol{X}, \mathcal{H}) \tag{4}$$

In this setting, we allow multi-label predictors, that is, predictors that output multiple cells at once. These can be easily modeled as $P(V_{i_1}, \ldots, V_{i_m} \mid \boldsymbol{X}^j)$, where $i_1, \ldots, i_m$ are the predicted cells. More importantly, we allow the base predictors to take *predictions* as inputs. This is essential for modeling inter-dependent cells. If such predictors were forbidden, it may be impossible to predict some or most cells in the spreadsheet.

Unfortunately, dependencies complicate the autocompletion step, because they can lead to *cycles*. That is, there may be cases where a cell Y_1 can be predicted from another cell Y_2, and then Y_2 can be equally well predicted from Y_1. To guarantee that the value of every cell is uniquely defined and that the resulting graphical model is acyclic, we forbid these cases. To do so, we fix the order in which the cells are autocompleted, so that no cell occurs twice. This is common practice in probabilistic graphical models (Koller and Friedman 2009). Now, given an order $\pi = (i_1, \ldots, i_n)$, the joint probability decomposes as:

$$\begin{aligned} &P(Y_1, \ldots, Y_n \mid \boldsymbol{X}, \mathcal{H}) \\ &\quad = P(Y_{i_1} \mid \boldsymbol{X}, \mathcal{H}) P(Y_{i_2} \mid Y_{i_1}, \boldsymbol{X}, \mathcal{H}) \ldots P(Y_{i_n} \mid Y_{i_{n-1}}, \ldots, Y_1, \boldsymbol{X}, \mathcal{H}) \end{aligned} \tag{5}$$

The factors can be easily computed. Indeed, letting $V_{i,j}$ be the prediction of the jth predictor for the ith cell and $i_s \in \pi$, then we can always rewrite its raw distribution as $P(V_{i_s,j} \mid \boldsymbol{X}^j, Y_{i_{s-1}}, \ldots, Y_{i_1})$ and compute the overall value distribution of cell i as in Eq. 3. The complete directed graphical model is illustrated in Fig. 2.

Inference with multiple cells Given an order π, PSYCHE computes the joint distribution of the n values by using the factorization in Eq. 5. More in detail, for all $i_s \in \pi$, the distribution $P(V_{i_s}, \ldots, V_{i_1} \mid \boldsymbol{X}, \mathcal{H})$ is computed by multiplying $P(V_{i_{s-1}}, \ldots, V_{i_1} \mid \boldsymbol{X}, \mathcal{H})$ and $P(V_{i_s} \mid V_{i_{s-1}}, \ldots, Vi_1, \boldsymbol{X}, \mathcal{H})$. PSYCHE takes the hard constraints $\mathcal{H}$ into account by discarding all values that are incompatible with $\mathcal{H}$ as soon as they are encountered. We note in passing that this form of probabilistic inference under constraints can be modelled with probabilistic logic programming frameworks such as ProbLog (De Raedt et al. 2007; Dries et al. 2015) and solved using efficient knowledge compilation techniques (Fierens et al. 2015). This promising research direction will be pursued in future work.

If no order π is given, then PSYCHE performs inference for all possible orders, and then chooses the overall most likely joint autocompletion $v_1, \ldots, v_n$. This automatically determines the best order, too. Notice that the number of possible orders π is exponential in the number of target cells n, but the latter can be always restricted to be small enough (e.g. 4 or 5), regardless of whether the target cells are selected automatically by the spreadsheet application or by the user of the system. This also limits the runtime of the procedure. If a large number of cells are to be jointly predicted, we can also fall back to a fixed-ordering such as left-to-right, which mimics the behavior of human input. In our experiments, for instance, predicting three cells takes about a second on average.

Estimating all the components Estimating the parameters of the joint distribution in Eq. 5 is accomplished as follows. Let's start from the single-cell case. Fitting $P(Y \mid \boldsymbol{X})$ involves works as following:

- For each base predictor $f_j \in \mathcal{F}$:
  - Fit the calibrator $P(V'_j \mid V_j)$ by estimating the conditional probability table of via maximum likelihood estimation.
    As discussed above, this amounts to splitting the training set $\boldsymbol{D}$ into folds, counting the co-occurrence of predictions v' and actual values v in the fold, averaging the

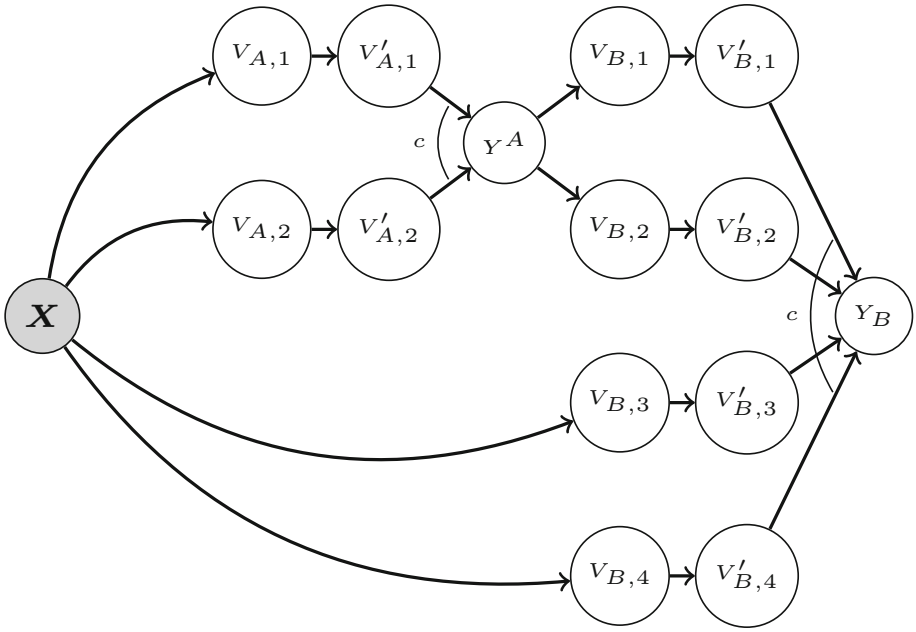

Fig. 2 Directed graphical model for two cells A and B, according to the order $\pi = \{A, B\}$. Top, from left to right: two models predict cell A from $\boldsymbol{X}$, and then two models cell B from A, producing $V_{B,1}$ and $V_{B,2}$. Bottom: two models predict cell B from $\boldsymbol{X}$, giving $V_{B,3}$ and $V_{B,4}$

counts across folds, computing the conditional probability table, and then applying Laplacian smoothing to it.
  - Estimate the accuracy acc_j of f_j on $\boldsymbol{D}$ using cross-validation as above.
- Compute the weights of the combiner (Eq. 2) by using the accuracies estimated in the previous step.

The multiple cell case boils down to fitting all the factors Eq. 5 on the observed cells $\boldsymbol{D}$. For instance, consider two cells in the order $\pi = (1, 2)$. Notice that we do allow cell 2 to be predicted from cell 1, but not vice versa. In this simple case, the calibrators of all base predictors for cell 1 are first estimated on $\boldsymbol{D}$, and so is their combiner; next, the same is done for the base predictors and combiner of cell 2. This guarantees that all the required data are available at all times.[4]

4.3 Handling continuous values

So far we have considered completing cells with discrete values only. Continuous values are considerably more difficult to handle, chiefly because all point values $v \in \mathbb{R}$ of a continuous RV Y have zero probability (Koller and Friedman 2009), and—depending on the choice of continuous distributions—finding the most likely value may require using non-convex continuous or continuous-discrete optimization.

[4] In principle, the training set of cell 2 should also include the predictions for cell 1, but we ignore this step, for simplicity and efficiency. This did not seem to have a huge effect in our experiments (data not reported).

In order to avoid these problems, we compute a confidence interval ε_j for every base regressor f_j, chosen so that the probability that the real value falls within ε_j distance from the prediction is large.[5] Now, we model $P(Y = v \mid V_j, \boldsymbol{X})$ as a discrete distribution $P(Y \in [V_j - \epsilon_j, V_j + \epsilon_j] \mid \boldsymbol{X})$. The intuition is that high-quality regressors will have small confidence intervals, so that they only "support" predictions close to their own; on the other hand, bad regressors will have large intervals and therefore support even predictions far away from their own. In order to avoid non-convex optimization, we also restrict the combiner used for choosing among alternative regressors to picking the most accurate one. This is equivalent to using a very steep value for η in the combiner equation, which leads to all weights to be zero except the one of the best regressor. This model, despite being somewhat restrictive, has the advantage of enabling reasoning on continuous values in a crisp probabilistic manner; in the empirical analysis we will show that it also works well in practice.

5 Empirical evaluation

In this section we address the following research questions:

Q1 How does PSYCHE compare against state-of-the-art ensemble predictors?
Q2 How do different combiners influence accuracy and calibration?
Q3 What is the benefit of allowing interdependent predictors (chaining)?
Q4 What is the effect of adding formulas?
Q5 Can our method provide suggestions in real time?

The code for the complete experimental setup is available at: [will be made available upon acceptance].

Base predictors As base predictors we use decision trees for categorical data and both regression trees as well as linear regression for numerical data (picking the best scoring one). We train all models using scikit-learn (Pedregosa et al. 2011) using a maximum depth of 4 for tree based methods (including random forests) and default parameters otherwise. For every spreadsheet, the models are trained and cross-validated on the first i rows, to obtain the predictors and confusion matrices to be used to predict the $(i + 1)$th row (i is dependent on the data-set) and we report results averaged over all predictions. For single-cell predictions or unchained predictions we choose 5 random subsets of the non-target columns to act as features and train an applicable base predictor for every subset. If prediction-chaining (i.e., interdependent predictors) is enabled, predictors can additionally use a subset of the columns we aim to predict. For the possible subsets we train two base classifiers that each use a random subset of the input column as well as the subset of target columns classifiers. The parameters of the calibrators and accuracy/score are estimated using leave-one-out cross-validation and the smoothing constant is set to $c = 0.1$.

Data preparation We performed our evaluation on four data-sets: (1) *euses* ($i = 10$), a subset of 86 spreadsheets from the EUSES corpus (Fisher and Rothermel 2005), a large collection of real-world spreadsheets scraped from the Web; (2) *tacle* ($i = 10$), a private set of 50 spreadsheets used to benchmark the TACLE constraint learner; (3) *openml* ($i = 40$), a set of 99 CSV files from OpenML with up to 20 features, no missing data and 50 to 100 instances (Vanschoren et al. 2013); and (4) *ice-cream* ($i \in [10, 15]$), a set of 20 synthetically generated noisy spreadsheets based on the running example with a formula (total sales,

[5] In practice, this can be obtained by cross-validating the L_1 loss on the dataset D^j.

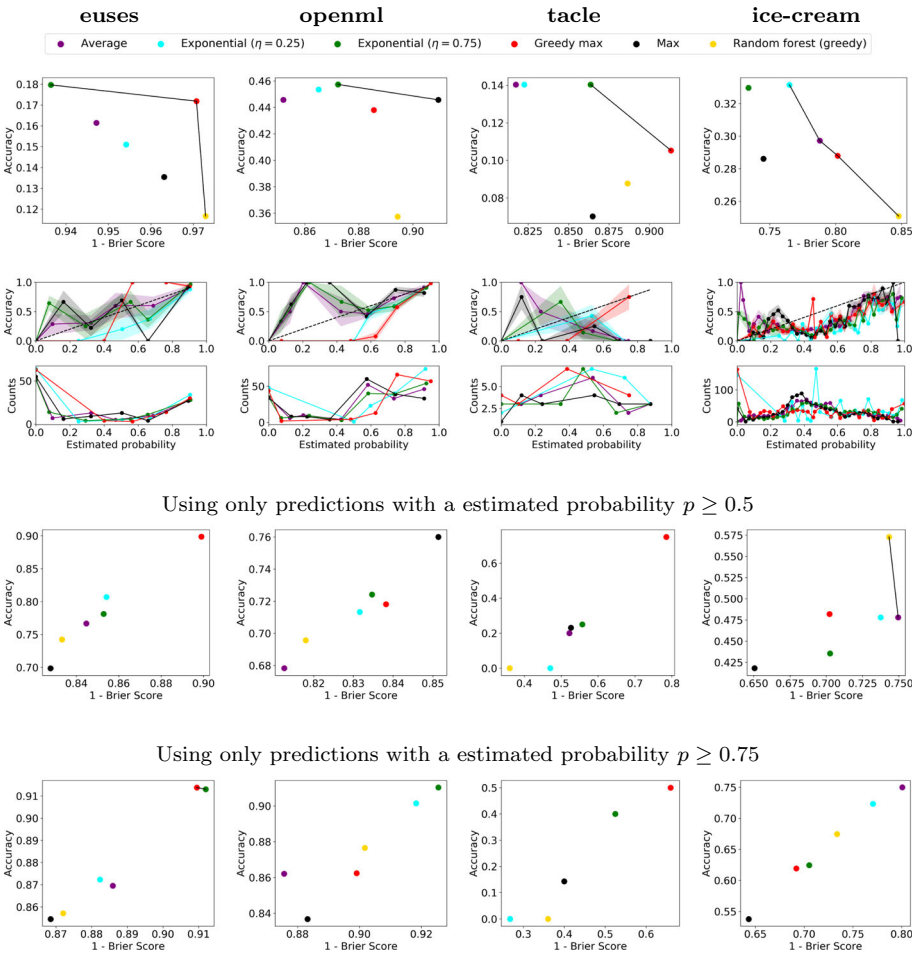

Fig. 3 The pareto curves in the first row show how different mixing strategies perform in terms of accuracy and calibration on our four data-sets. The calibration curves (second row) group predictions in bins based on their estimated probability and plot the average accuracy (with colored bands indicating the standard error on the mean) over the average estimated probability per bin. The lower half of those plots show the number of predictions per bin. Finally, rows 3 and 4 show pareto curves for predictions matching a threshold on the estimated probability of 0.5 and 0.75, respectively. Every column corresponds to one data-set and the legend at the top of the table is common for all plots (Color figure online)

best quarter) and noisy relationships (for august and profit). We pre-processed the *euses* spreadsheets by removing comments, blank columns, *etc.*, to allow PSYCHE to detect the tables from the data and to allow TACLE to detect formulas and constraints.

The *ice-cream* data-set approaches best the setting in which we believe PSYCHE will be used, while the *euses* data-set provides the most realistic set of spreadsheets. However, *euses* contains only few (detectable) formulas and interesting relationships. Therefore, we include *tacle* as data with a higher number of formulas and *openml* as data suited for machine learning.

	euses			**openml**	
	$\neg formulas$	$formulas$		$\neg formulas$	$formulas$
$\neg chaining$	0.205 ± 0.012	0.212 ± 0.012	$\neg chaining$	0.741 ± 0.009	0.735 ± 0.009
$chaining$	0.210 ± 0.012	0.218 ± 0.013	$chaining$	0.754 ± 0.009	0.753 ± 0.009
	tacle			**ice-cream**	
	$\neg formulas$	$formulas$		$\neg formulas$	$formulas$
$\neg chaining$	0.176 ± 0.029	0.196 ± 0.031	$\neg chaining$	0.609 ± 0.012	0.628 ± 0.013
$chaining$	0.196 ± 0.033	0.197 ± 0.031	$chaining$	0.656 ± 0.013	0.668 ± 0.013

Fig. 4 By examining the average score ($\pm$ the standard error on the mean), we can see that both predictor-chaining and using formulas generally have a positive effect—especially for the formula-richer data-sets in the bottom row

Combiners In our experiments we compare four different combiners: *average*, *exponential*, *max* and *greedy max* on the task of predicting categorical cells in our data-sets. The average combiner uses equal weights for all predictors, the exponential combiner implements the multiplicative variant with exponential weights (we consider $\eta = 0.25$ and $\eta = 0.75$ and cap d at 10 predictions), the max combiner sets the weight of the most accurate predictor to 1 and all others to 0, and the greedy combiner forgoes the calibration step, greedily picking the most accurate predictor at every step and using the accuracy of that predictor as self-assessed probability. Both max and greedy allocate responsibility to one predictor for every cell to predict while the other combiners (the sum combiners) actually combine the predictions of several predictors.

Evaluating success We evaluate PSYCHE by measuring and reporting the accuracy on categorical cells. For numeric cells we instead measure the L_1-distance between the actual solution and the predicted solution normalized by the distance between the largest and smallest value observed for the column of the predicted cell. Finally, we report the score as $\max(1 - distance, 0)$. When predicting multiple cells an order needs to be fixed to prevent cyclic dependencies. In our experiments on chaining and formulas we predict sets of 3 consecutive cells, consider all possible orders for the prediction task and pick the order that yields the most confident prediction.

Before proceeding, we remark that, due to the generality of the prediction task, it may be difficult to estimate the performance of approaches for predictive spreadsheet autocompletion using standard techniques, such as cross-validations. Notice also that the semi-relational nature of spreadsheets implies that, due to constraints/relations among different rows, it may be impossible to split them into truly independent folds. In our evaluation, we circumvent these issues by restricting the experiments to classification and regression tasks, and by disallowing constraints across rows.

Measuring calibration For every prediction, PSYCHE outputs the estimated probability value p that the prediction is correct and we measure the accuracy acc (which will be 0 or 1 for a single categorical prediction). The combiners influence both the prediction—and thus the accuracy—as well as the estimated probability value. To measure which of the combiners predicts probabilities more in line with the actual accuracy we use the Brier (quadratic) scoring rule (Brier 1950). Given n predictions each with an estimated probability and measured accuracy (p_i, acc_i), the Brier Score is computed as $\sum_{i=1}^{n} \frac{1}{n}(p_i - \text{acc}_i)^2$. The Brier Score is minimal when the estimated probability matches the true probability that a classifier predicts the right answer. We consider this classifier *well calibrated*. In our plots we use $1 - Brier\ Score$ such that, as with the accuracy, higher values are better.

Q1 *How does* PSYCHE*compare against state-of-the-art ensemble predictors*? In order to evaluate the accuracy of PSYCHE, we consider first the task of predicting a single cell and compare our method to a baseline version that uses random forests (Breiman 2001), trained on all columns not containing the prediction target, as base predictors. The random forest implementation uses 10 estimators. By comparing random forests against PSYCHE using decision trees and different combiners, we observe that PSYCHE usually achieves similar or superior accuracy than random forests (Fig. 3, row 1). The cross-validated accuracy of the random forest does turn out to be a reasonably well-calibrated probability estimate for the accuracy on unseen data.

When considering all predictions, the overall accuracy can be quite low—especially for the euses and tacle data-sets. This is the case because in these spreadsheets most of the columns are not predictable (e.g., names, addresses, …), might contain a large number of values or some of the input columns contain previously unseen values. This points at the importance of the ability to estimate the probability that a prediction is correct, i.e., knowing when a prediction should be made and when not.

Q2 *How do different combiners influence accuracy and calibration*? Our experiments using various combiners on different data-sets show that the sum combiners generally achieve higher accuracy, while the max combiners tend to yield better calibrated predictions (Fig. 3, row 1). On most data-sets the greedy combiner is more calibrated than the max combiner, however, on the *openml* dataset more training data is available and the max combiner performs better than the greedy one both in terms of accuracy and calibration. This difference is due to the fact that the combiners using the calibration step (all except greedy) need to fit more parameters. A higher η-value for the exponential combiner yields larger differences in weights while lower values yield results more similar to the unweighted average combiner.

By grouping predictions in fixed-size bins we can visually compare the average accuracy with the average estimated probability per bin, which shows for what estimated probabilities PSYCHE is over- or under-confident. Generally, the combiners tend more towards slightly over-confident predictions, however, the sum-based combiners can be under-confident for lower estimated probability values (Fig. 3, row 2).

When considering only predictions with a high estimated probability, the exponential combiners achieve good accuracy and calibration values for most data-sets (Fig. 3, rows 3 and 4). For the data-sets with few training instances (rows)—euses and tacle—the greedy max combiner generally scores highest. It does, however, not provide an actual probability distribution to, e.g., suggest likely alternative values. These plots also illustrate that on the tacle data-set all combiners are badly calibrated for higher estimated probabilities, as the average accuracy for predictions estimated to be at least 75% correct lies below 0.5.

Q3 *What is the benefit of allowing interdependent predictors*? By running two versions of our method, one of which disallows prediction-chaining, on the task of predicting any 3 consecutive cells on the spreadsheets in our data-sets, we can compare their performance. We expect that prediction chaining increases the accuracy, whenever multiple consecutive columns can be successfully predicted, as, for example, in the case of our *ice-cream* experiment. Our experiments show that, indeed, prediction chaining increases the accuracy of the experiments (Fig. 3, row 3).

Q4 *What is the effect of adding formulas*? Formulas are a form of specialized predictors, geared toward the spreadsheet setting. We evaluated the effect of formulas by on the task of predicting any 3 consecutive cells on the spreadsheets in our data-sets as for **Q3**. The potential for increased accuracy depends strongly on the number of usable formulas discovered in a

spreadsheet. Additionally, some of the formulas such as sum can also be recovered using linear regression. Our experiments show that for we can achieve moderate to major accuracy improvements on our data-sets (Fig. 3, row 3) and the best results are obtained when both chaining and formulas are used. Due to its flexible design, PSYCHE can readily integrate additional constraint learners, equation discovery systems or program synthesis algorithms to further improve the accuracy of its predictions.

Q5 *Can our method provide suggestions in real time* Excluding TACLE, executing the whole pipeline on our datasets—that is, training the base predictors and obtaining the joint prediction—takes on average less than a second for single cell predictions and about a second for 3 cells. Constraint learning with TACLE can take tens of seconds. However, this can be amortized as TACLE discovers constraints globally and does not need to be rerun for every prediction task. Additionally, we extended TACLE to include a timeout per constraint template, meaning that it can be run quickly to obtain initial formulas and optionally can be rerun with increased timeouts in the background (Fig. 4).

6 Related work

There are several strands of related work. First, PSYCHE has been strongly influenced by work on combining machine learning and automatic programming with spreadsheets. Most spreadsheet software includes a limited form of autocompletion: if the values in a row or column follow some simple pattern (e.g., constant or progressive), the next values in the pattern can be automatically filled in. But this is clearly a very limited setting. The recent work of Gulwani on programming-by-example and program synthesis (Gulwani 2011; Gulwani et al. 2017) showed that strong prior knowledge is key to autocompletion in non-toy applications. The seminal FlashFill automatically completes columns from a few examples of the desired values. Several methods follow this idea and adapt it to different tasks (e.g., data wrangling Raza and Gulwani 2017). This insight is leveraged by TACLE (Kolb et al. 2017) to induce constraints from spreadsheets. PSYCHE is extending this idea towards autocompletion of arbitrary cells in spreadsheets using predictive models, formulas, and constraints.

Very relevant are also the BigML platform and related efforts, which integrate standard learning algorithms, workflows, and visualisations with easy to use spreadsheet interfaces. The goal of these approaches is to lower the entry point to data analysis for non-experts, for instance by allowing to quickly analyze features (columns) and predictive models (for a single column at a time). Predictive spreadsheet autocompletion goes beyond this, by automating predictions without any user intervention—although user guidance can be leveraged, if available, by adapting the combiners, calibrators, and (if possible) predictors whenever the user fills in or changes any cells. BigML is also restricted in predicting a single column at a time, it does not consider the issue of chaining predictions, and it uses a very simplistic missing value imputation strategy.[6] Although rather general and useful, these approaches neither address nor solve predictive spreadsheet autocompletion. Most importantly, PSYCHE is meant to integrate black-box classifiers obtained from potentially disparate sources into a single collective predictor. BigML, in contrast, offers only a handful of classifiers (e.g. decision trees or extensions thereof) that are trained directly on the data itself.

PSYCHE is related to techniques for imputing missing values in data tables (Scheuren 2005; Van Buuren 2018), which encompass both statistical (Van Buuren 2007) and machine learning approaches (Stekhoven and Bühlmann 2011). These methods can impute multiple

[6] See the documentation at https://bit.ly/2Bvc8De, retrieved on November 2018.

data entries, and reorder the rows as to optimize the pattern of cells to be imputed with respect to the observed ones. Our idea of chaining predictions for different cells is similar in spirit to the "fully conditional specification" used in multiple imputation (Van Buuren 2007). Missing value imputation, however, differs from predictive spreadsheet autocompletion in several key aspects. First, spreadsheets often include multiple related tables, and individual tables may have different directionalities (e.g. column-based, row-based, or heterogeneous). This can be readily handled by PSyCHE. In contrast, imputation approaches assume the data to be a single table in attribute-value format. Second, formulas and constraints play a key role in spreadsheets, but are absent in standard imputation settings. As a consequence, imputation methods do not consider the issue of consistency in practice. PSyCHE, on the other hand, is specifically designed to acquire formulas and constraints (via an extension of TACLE) and use them during inference to ensure consistency.

Another class of related work consists of ensemble methods in machine learning (Dietterich 2000), which learn multiple predictive models and combine them to make predictions. Most ensemble methods focus on a single target attribute, although there also exist multiple classifier methods that combine predictions for multiple attributes. One example is the recent MERCS system (Van Wolputte et al. 2018) which uses ensembles of multi-target decision trees to predict any cell in an attribute-value table. There are two differences with such techniques. First, PSyCHE is able to learn and use constraints (across multiple tables), and second, it is based on a probabilistic framework that serves to integrate these predictions.

Finally, relational learning methods (Muggleton and De Raedt 1994; Yin et al. 2006; Nickel et al. 2011) share some similarities with our approach, in that they can handle multiple tables and potentially constraints. However, they do not integrate black-box base predictors into the inference, which is instrumental to the performance of the overall pipeline.

The predictive spreadsheet completion problem tackled in the present paper is more challenging than the present approaches, as it requires to combine predictive models and learned constraints. The semi-relational nature of the data and the fact that the underlying prediction task is unknown immediately rule out most machine learning approaches. In PSyCHE we assume that the target spreadsheet has been pre-processed into a manageable format. This is not always realistic, and one direction for further research is concerned with the automatic detection of useful structure in the spreadsheet that can be used to obtain such manageable formats automatically. The combination of predictive auto-completion with data wrangling yields an automated data scientist, a direction of research that we are actively pursuing, see De Raedt et al. (2018).

7 Conclusion

We introduced the novel problem of predictive spreadsheets autocompletion with the aim of partially automating the spreadsheet workflows of end-users. This task is more challenging than standard prediction and imputation problems, for several reasons: the data generation process can be almost arbitrary, formulas and constraints play a central role, and the data is spread across multiple tables. We address these challenges by proposing a probabilistic model that combines task-specific base predictors and learned constraints (e.g. spreadsheet formulas) to produce a consistent joint prediction for all target empty cells. This setup handles unobserved dependencies among heterogeneous cells and tables. Suitable modeling assumptions are made to handle data scarcity. We show empirically that our proposed approach, PSyCHE, produces reasonable suggestions in an efficient manner when applied to real-world

spreadsheets from the EUSES corpus. The integration of PSYCHE into existing spreadsheet applications is discussed.

Key future extensions to this work include: (1) leveraging historical data and specialized pre-trained predictors, (2) exploiting the relational structure of spreadsheet to transfer predictors across tables, and (3) most importantly, integrating the proposed method into spreadsheet applications by devising a suitable protocol for providing suggestions to the user and learning from the received feedback. All extension fit naturally in the presented framework and would improve the coverage and quality of the completions in this data scarce setting.

References

Arora, S., Hazan, E., & Kale, S. (2012). The multiplicative weights update method: A meta-algorithm and applications. *Theory of Computing*, *8*(1), 121–164.

Beldiceanu, N., & Simonis, H. (2012). A model seeker: Extracting global constraint models from positive examples. In *Principles and practice of constraint programming* (pp. 141–157). Springer.

Bessiere, C., Daoudi, A., Hebrard, E., Katsirelos, G., Lazaar, N., Mechqrane, Y., et al. (2016). New approaches to constraint acquisition. In C. Bessiere, et al. (Eds.), *Data mining and constraint programming* (pp. 51–76). Cham: Springer.

Bessiere, C., Coletta, R., Koriche, F., & O'Sullivan, B. (2005). A sat-based version space algorithm for acquiring constraint satisfaction problems. In J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, & L. Torgo (Eds.), *Machine learning: ECML 2005* (pp. 23–34). Berlin: Springer.

BigML Home Page. Retrieved 30 April 2019 https://bigml.com.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Berlin: Springer.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Breiman, L. (2017). *Classification and regression trees*. Abingdon: Routledge.

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthey Weather Review*, *78*(1), 1–3.

Clemen, R. T., & Winkler, R. L. (1999). Combining probability distributions from experts in risk analysis. *Risk Analysis*, *19*(2), 187–203.

De Raedt, L., Blockeel, H., Kolb, S., Teso, S., & Verbruggen, G. (2018). Elements of an automatic data scientist. In *International symposium on intelligent data analysis* (pp. 3–14). Springer.

De Raedt, L., Kimmig, A., & Toivonen, H. (2007). Problog: A probabilistic prolog and its application in link discovery. In *Proceedings 20th international joint conference on artificial intelligence*.

Devlin, J., Uesato, J., Bhupatiraju, S., Singh, R., Mohamed, A.r., & Kohli, P. (2017). Robustfill: Neural program learning under noisy i/o. In *International conference on machine learning* (pp. 990–998).

Dietterich, T.G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems* (pp. 1–15). Springer.

Dries, A., Kimmig, A., Meert, W., Renkens, J., Van den Broeck, G., Vlasselaer, J., & De Raedt, L. (2015). Problog2: Probabilistic logic programming. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 312–315). Springer.

Elisseeff, A., & Pontil, M. (2003). Leave-one-out error and stability of learning algorithms with applications. *NATO Science Series Sub Series III Computer and Systems Sciences*, *190*, 111–130.

Fierens, D., Van den Broeck, G., Renkens, J., Shterionov, D., Gutmann, B., Thon, I., et al. (2015). Inference and learning in probabilistic logic programs using weighted boolean formulas. *Theory and Practice of Logic Programming*, *15*(3), 358–401.

Fisher, M., & Rothermel, G. (2005). The euses spreadsheet corpus: A shared resource for supporting experimentation with spreadsheet dependability mechanisms. In *ACM SIGSOFT software engineering notes*, vol. 30, (pp. 1–5). ACM.

Gulwani, S. (2011). Automating string processing in spreadsheets using input–output examples. In *ACM SIGPLAN notices*, vol. 46, (pp. 317–330). ACM.

Gulwani, S., Harris, W. R., & Singh, R. (2012). Spreadsheet data manipulation using examples. *Communications of the ACM*, *55*(8), 97–105.

Gulwani, S., Hernández-Orallo, J., Kitzelmann, E., Muggleton, S. H., Schmid, U., & Zorn, B. (2015). Inductive programming meets the real world. *Communications of the ACM*, *58*(11), 90–99.

Gulwani, S., Polozov, O., Singh, R., et al. (2017). Program synthesis. *Foundations and Trends® in Programming Languages*, *4*(1–2), 1–119.

Jordan, M. I., & Jacobs, R. A. (1994). Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, *6*(2), 181–214.
Kolb, S., Paramonov, S., Guns, T., & De Raedt, L. (2017). Learning constraints in spreadsheets and tabular data. *Machine Learning*, *106*(9–10), 1441–1468.
Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. Cambridge: MIT press.
Lawson, B. R., Baker, K. R., Powell, S. G., & Foster-Johnson, L. (2009). A comparison of spreadsheet users with different levels of experience. *Omega*, *37*(3), 579–590.
Muggleton, S., & De Raedt, L. (1994). Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, *19*, 629–679.
Nickel, M., Tresp, V., & Kriegel, H. P. (2011). A three-way model for collective learning on multi-relational data. *ICML*, *11*, 809–816.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.
Raza, M., & Gulwani, S. (2017). Automated data extraction using predictive program synthesis. In *AAAI* (pp. 882–890).
Rossi, F., & Sperduti, A. (2004). Acquiring both constraint and solution preferences in interactive constraint systems. *Constraints*, *9*(4), 311–332.
Scaffidi, C., Shaw, M., & Myers, B. (2005). Estimating the numbers of end users and end user programmers. In *Visual languages and human-centric computing, 2005 IEEE symposium on IEEE* (pp. 207–214).
Scheuren, F. (2005). Multiple imputation: How it began and continues. *The American Statistician*, *59*(4), 315–319.
Stekhoven, D. J., & Bühlmann, P. (2011). MissForest—Non-parametric missing value imputation for mixed-type data. *Bioinformatics*, *28*(1), 112–118.
Van Buuren, S. (2007). Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical Methods in Medical Research*, *16*(3), 219–242.
Van Buuren, S. (2018). *Flexible imputation of missing data*. Boca Raton: Chapman and Hall/CRC.
Van Wolputte, E., Korneva, E., & Blockeel, H. (2018). Mercs: Multi-directional ensembles of regression and classification trees. In *AAAI*.
Vanschoren, J., van Rijn, J. N., Bischl, B., & Torgo, L. (2013). OpenML: Networked Science in machine learning. *SIGKDD Explorations*, *15*(2), 49–60. https://doi.org/10.1145/2641190.2641198.
Yin, X., Han, J., Yang, J., & Philip, S. Y. (2006). Efficient classification across multiple database relations: A crossmine approach. *IEEE Transactions on Knowledge & Data Engineering*, *6*, 770–783.