Evaluating MIDST, A System to Support Stigmergic Team Coordination

KEVIN CROWSTON, JEFFREY SALTZ, NIRA J SITAULA, and YATISH HEGDE, Syracuse University School of Information Studies

Data science teams working on a shared analysis face coordination problems such as dividing up the work to be done, monitoring performance and integrating the pieces. Research on distributed software development teams has raised the potential of stigmergic coordination, that is, coordination through a shared work product in place of explicit communication. The MIDST system [6] was developed to support stigmergic coordination by making individual contributions to a shared work product visible, legible and combinable. In this paper, we present initial studies of a total of 40 student teams (24 using MIDST) that shows that teams that used MIDST did get the planned benefits for their work, did seem to coordinate at least in part stigmgerically and performed better on an assigned project.

CCS Concepts: • Human-centered computing → Computer supported cooperative work; Empirical studies in collaborative and social computing; • Information systems → Data analytics.

Additional Key Words and Phrases: stigmergic coordination; translucency; awareness; data-science teams

ACM Reference Format:

1 INTRODUCTION

This paper presents initial experiences using the MIDST (Modular Interactive Data Science Tool) system [6], a web-based application intended to support coordination for data-science teams. Data-science projects have goals such as identifying correlations and causal relationships, classifying and predicting events, identifying patterns and anomalies or inferring probabilities, interest and sentiment [7]. A common data-science programming language is R [1]; analyses are performed by writing programs in the R language that take data as input and output analysis results.

While a small data-science project can be carried out by a single person, large analyses need the effort of a team. Task coordination is a key challenge for team members [10], e.g., dividing the project into manageable pieces (e.g., data cleaning, model fitting, visualization), assigning pieces to team members, tracking progress, dealing with interdependencies and integrating the pieces to a final project. For instance, model fitting needs to be run on a cleaned data set, so a team member writing code for model fitting needs to know what the cleaned data set looks like (e.g., what variables are available) and to retrieve the new data set when changes are made. A further complication is that data analyses are often iterative, that is, the results of one analysis may suggest additional analyses or changes to prior steps. For instance, fitting a model may identify outliers in the data or needed transformations that should be part of the data cleaning.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

© 2020 Copyright held by the owner/author(s).
2573-0142/2020/8-ART111
https://doi.org/10.1145/xxxx.xxxx
The goal of the MIDST system is to support a team of data scientists collaborating on developing a data analysis script. It is a web-based application that supports developing and running a shared code base and project management. The findings of our initial studies across 40 student project teams (24 using MIDST) are three-fold. First, there is evidence that the system does support the planned affordances. Second, the system does seem to enable members of the teams using it to coordinate their work. And finally, the teams using the system achieved better results on their project than those who didn’t. We present in turn the intended features of MIDST, our study methodology and findings before concluding with some reflections on our initial studies and suggestions for future studies.

2 PRIOR WORK

The current paper builds on the development of the MIDST system [6]. The novelty of the MIDST system is that it sought to support stigmergic coordination, that is, coordination based on signals from the shared work in addition to or rather than through explicit communication. Stigmergy was formulated initially to explain the behaviour of social insects (e.g., ants or wasps) that achieve coordinated results (e.g., deploying workers to a food source or building a nest) without explicit communication [11]. Instead of talking to each other, the insects follow simple behavioural rules to react to changes in the environment made by other insects (e.g., ants exploiting a food source by following the scent trail left by another ant). More recently, stigmergy has been invoked to explain human behaviours: the formation of trails in a field as people follow paths initially laid down by others (similar to ant trails), or markets, as buyers and sellers interact through price signals [20].

In CSCW specifically, Christensen [3–5] discussed how architects and builders coordinate their tasks through “the material field of work” such as drawings, building on earlier work in CSCW focusing on coordination through work such as changes in shared databases [25]. Stigmergy has been suggested in particular as an interpretation of how free/libre open source software (FLOSS) developers coordinate [2], what Kalliamvakou et al. called a “code-centric collaboration” perspective [13]. FLOSS developers interact in part through the code that they are developing, managed with source code control systems that provide status about the state of the code and development. Stigmergic coordination is valuable for software developers as it enables good coordination of the the development process while reducing the need for explicit communication, which can be costly, especially in a distributed environment. The design of MIDST drew specifically on the experiences of FLOSS developers. However, stigmergy has been argued as a mechanism in online work more generally. For instance, Elliot [9] argued that “[c]ollaboration in large groups is dependent on stigmergy,” with the specific example of authoring on Wikis.

The MIDST system was designed to support stigmergic coordination of the work of data-science teams by providing three system affordance, namely visibility, legibility and combinability of individual contributions made to the shared work product. To introduce the system, we discuss these in turn.

2.1 Visibility

A first requirement for stigmergic coordination is that the work and the status of the work done by team members must be made visible to other team members. MIDST supports visibility of work in several ways. First, MIDST organizes work into projects; the creator of a project can invite others to view and contribute to it. The breakdown of work within the project is presented in a shared network view, shown in Figure 1. This view represents the data analysis as a workflow, the nodes representing individual units of analysis. The flow of data through the system (and thus the dependencies of code modules) is visualized via the node connections in the network view. The
system shares changes to the workflow with all users who have the project open as they are made, e.g., a new node added to the workflow appears in real time for other users.

Second, the system also shares code, thus making users’ work mobile and visible to others. When a node is opened, it reveals a Jupyter Notebook\(^1\) [16] containing the R code for the node, as shown in Figure 2. Data from the connections is made available in a variable to the R code in the node. A user can execute the entire network, a single node (assuming the prior nodes have been run to create the input data) or a single cell in the Jupyter Notebook (e.g., for debugging).

---

\(^1\)The version of MIDST described in [6] used a plain text editor rather than Jupyter Notebooks for this view.
To avoid sharing unfinished and perhaps unusable code, the contents of a node are updated only on demand. A user who has made changes can push the new code to other team members when it is ready, for them to test or to use. Other users are notified that new code is available, but can defer pulling it, e.g., until they are ready to integrate any changes.

Third, to support team project management, MIDST tracks and displays the owner and status of each node. In the current version, only the node’s owner can push code changes; others can only view and run the node and make temporary changes (though any user can change the ownership of a node). A node’s development status is one of proposed, not started, in progress, validating or completed. Nodes are created in the proposed status without an owner. For instance, team members might start work by brainstorming an initial plan for the project as a set of proposed nodes. As part of the project planning, nodes can be assigned to a team member to work on and their status changed to not started. When a node is opened for the first time, it is automatically moved to in progress. If the node is unassigned, it is assigned to the user who first opens it. Nodes are manually moved to validating to signal that the code is ready to be checked by another team member or to completed to signal that work is done. Figure 3 shows which statuses have an assigned user and the expected status transitions (though a user can manually move a node to any status from any status).

Changes in these node attributes are visually shared as they happen in the network view (via the colouring and other node decorations), in the notebook view (via messages from MIDST) and in a task kanban view (Figure 4). The node status can be updated by dragging it in the kanban view or from a menu on the notebook view. Nodes with execution errors are shown with a ‘!’ icon on a node in the network view. These displays allow team members to get an update on the status of their project at a glance.

Changes in these node attributes are visually shared as they happen in the network view (via the colouring and other node decorations), in the notebook view (via messages from MIDST) and in a task kanban view (Figure 4). The node status can be updated by dragging it in the kanban view or from a menu on the notebook view. Nodes with execution errors are shown with a ‘!’ icon on a node in the network view. These displays allow team members to get an update on the status of their project at a glance.

2.2 Legibility

A second prerequisite for stigmeric coordination is that other users must be able to recognize the purpose of a contribution to the shared work to know what it means and how to work with it, that is, the contribution must be legible. It has been noted that as an analysis grows it often becomes difficult to navigate and understand [21], which is an impediment for efficient and effective development and maintenance. For example, prior research has revealed that many data-science analyses are difficult to understand, and that even the original analyst can struggle to understand their prior effort [14, 21].
In Ref [6], legibility was described by saying that the work had to be of a known genre, meaning that it exhibited socially-recognized regularities of form that provide useful signals of the purpose of a contribution [19]. For instance, in scholarly publishing there are different kinds of documents, each with a clear form and associated purpose, e.g., an article vs. an article review. By recognizing the form, an informed reader knows what they can do with the document. Ref [6] argued that source code could also be considered as having a genre, e.g., code for the user interface, data manipulation, etc. and that being able to recognize from its form what the code did made it easier to work with. Similarly, while describing the challenge of creating maintainable R applications, Malviya et al. [18] state that a well-defined workflow (i.e., where to put / get data and how to produce, collect and report the results) can help improve maintainability.

There are two ways in which MIDST improves legibility of data-science task contributions. First, MIDST supports the notion that each node has a status, which Ref [6] argued is one form of genre, in that a "not started" node requires different work than one that is in progress or ready for validation. Second, there is an implicit set of genres of code nodes in the workflow defined in the network view. Data flows from reading data to cleaning data to exploratory analysis to more advanced machine learning to visualizations to outputs. Where the node is within the network provides context as to the type of work likely being done within that node. Collectively, these genres helps the team member understand the context and goal of the node, independent of the actual details within the node. Ref [6] noted that MIDST’s support for genres was incomplete in that code nodes are just notebooks that can potentially do anything, but the expectation is they are more useful to other team members if they do one recognizable thing, an organization of work that is encouraged by the workflow.

2.3 Combinability

The final prerequisite for stigmergic coordination is combinability of work, meaning that one person’s contribution can easily to incorporated with others into a working whole. Ref [12] speaks similarly of FLOSS growing through a process of superposition as contributions layer on top of each other. However, the designers of MIDST observed that the lack of modularity of R scripts make them difficult to combine. Further, other research has shown that modularity is not a familiar concept for data scientists and that teams struggle to achieve effective task modularity [23].

MIDST supports combinability by encouraging users to create modular components. Specifically, MIDST supports modularity via the network view, in which nodes of R code have clearly defined

![Fig. 4. Kanban view in MIDST](image_url)
input and output ports to connect to other nodes. These input and output ports act as an interface definition, clarifying the work to be accomplished by the node. With respect to merging work, MIDST supports an easy way to push and pull code. When appropriate, MIDST reminds a user to push the code they’ve developed to the central repository and then reminds the other team members to pull. Initial results from another study of MIDST [23] suggest that use of MIDST does improve the modularity of an analysis.

2.4 Problem statement
The main contribution of the prior work described above was to develop a theory for the affordances hypothesized to support stigmergic coordination (visibility, legibility and combinability) and to translate these affordances to system design. In this paper, we seek to examine system usage empirically.

A particular interest is whether teams can make effective use of the developed functionality. Ref [6] noted that the affordances for stigmergic coordination are socio-technical, meaning that social practices are necessary to make the technical features effective. For instance, genres are social rather than natural; learning to recognize them is part of learning a group practice. (Consider the problem of students not recognizing the difference between a research article and a blog post, which to experienced researchers are entirely distinct genres.) Similarly, learning to write modular code takes practice. FLOSS developers have a set of such practices that support being able to coordinate via the code, such as a “atomic commits” (i.e., a contribution should fix one bug rather than many so it can be easily evaluated) and “not breaking the build” (i.e., a contribution should not leave the code base in an unusable state that would block the work of other users). For MIDST to be effective, team members need a similar set of practices around working with shared code.

The specific questions addressed in this paper are:

1. Do teams use the system to collaborate?
2. Do teams using the MIDST system develop work practices that use the system features, that is, do they use it to a) obtain visibility of other team members’ work, b) create legible work contributions and c) combine their individual work contributions?
3. Are teams using MIDST able to coordinate their work stigmergically?
4. Do teams using MIDST produce better analyses than teams that don’t?

3 METHOD
In this section, we describe the methods adopted to answer our research questions: the overall study design, the subjects, the data collected and the analyses.

3.1 Study design
Our study addresses several research questions, requiring different research approaches. Initially, we set out to test MIDST using a controlled experiment in which some teams used MIDST and others used RStudio, an integrated development environment for R. However, we found two limitations to this study design. First, we were not able to obtain data about collaboration from the RStudio teams. The teams coordinated face-to-face or by email, ways that were not visible to us as researchers. As a result, many of the analyses presented below use only data from the MIDST sections. Furthermore, we discovered that not all of the teams in the MIDST condition actually used the system as intended, compromising the evaluation but also raising the question of why they didn’t. Therefore, while we present some comparison of RStudio and MIDST teams, we also compare MIDST teams that did or did not use the collaboration features.
To answer question 1, we analyze the workflows created by the teams to identify teams that made more or less use of the system. To answer question 2a (visibility), we examine MIDST feature usage data that shows that team members did (sometimes) share code and examine or run each others’ nodes. To answer question 2b (legibility), we examine the nodes that teams created and informally cluster them. To answer question 2c (combinability), we examine the networks created to show how some do combine contributions from multiple team members. Question 3 (stigmergic coordination) is partly answered by data showing that team members monitor each others’ performance via MIDST. We also use data from a survey of MIDST users about perceived benefits of the tool. As well, we compare the activities of the initial meetings of teams using MIDST and R Studio. Finally, to answer Question 4 (performance), we present an analysis of the quality of projects created using MIDST and RStudio.

3.2 Subjects

The participants in the study were data-science graduate students in an introductory masters-level data science class. While there has been little written about using students to gain insight into industry teams within the data-science context, experiments with students have been common for decades in the software development domain. In fact, students were used as subjects in 87% of the software development experiments analyzed over a representative ten-year period [26]. It is important to note that when using students as subjects, several factors are typically considered. First, "students vs. professionals" is actually a misrepresentation of the confounding effect of proficiency, and in fact differences in performance are much more important than differences in status [27]. Hence, master-level students, many of whom have several years of industry experience, can often be an appropriate choice for subjects and act as a proxy for junior-level professionals. Second, comparing across experimental conditions, using students may reduce variability because all students have about the same level of education, leading to better statistical characteristics [17]. Finally, while students might not be as experienced as practicing professionals, they can be viewed as the next generation of professionals and hence suitable subjects for studies [15, 22].

Data in this paper comes from students in 5 sections of a face-to-face introductory masters-level data-science course in the final semester of 2019 and in 4 sections of a distance version of the same course in the first quarter of 2020. We expected that the face-to-face teams would have less need for coordination support, since they could easily meet and discuss their projects. However, it was easier to observe these students and to support their use of MIDST, which was still under development at the time. We expected the tool to be more useful for the distance students, who have fewer opportunities to meet, but to be more difficult to support. Most students were in a masters program in data science with some others in business administration.

In the face-to-face class, we had 18 teams in 3 sections that used MIDST and 9 teams in 2 sections that used RStudio. Sections used a common syllabus and assignments but were taught by different faculty members; the MIDST sections were taught by two of the authors of this paper. Students choose sections, so assignment of students to sections is not random meaning that there could be undetected systematic differences among them. Project teams were formed by the instructor and included 4 or 5 students. Students in the MIDST sections were given a short tutorial demonstration of MIDST to introduce the system and technical support when there were problems. In two of the sections they worked on an individual assignment and in all sections on a team project that had to be submitted for grading in MIDST. (Earlier assignments were done in RStudio.)

In the distance class, we had 6 teams in 2 sections that used MIDST and 7 in 2 sections that used RStudio. Teams were 2 to 4 students. As with the face-to-face class, one of the authors provided an introduction to MIDST. In these sections, we were able to observe an initial 20–30 minute video-conference meeting of 8 of the teams (3 MIDST and 5 RStudio) after they were assigned.
the project and a second meeting a week later for the RStudio teams. However, teams were not required to use MIDST and in the end only one or two did. Sections were taught by different faculty members, but not by the authors.

The project in both classes was deliberately open-ended: students were given a large dataset of responses to hotel customer surveys and asked to derive some interesting insights about drivers of customer satisfaction. The data needed to be cleaned and processed to create features for the data mining. Teams were expected to use a variety of data-mining techniques and to create a number of visualizations to illustrate their findings, which were presented in class and in a final report.

The study protocol was reviewed and approved by the Syracuse University Institutional Review Board. Students in the face-to-face class were required to use MIDST as part of the class experience, but were not required to take part in the study. Their agreement was not visible to the instructor of their section. While there were certainly some challenges for students using MIDST, they also benefited from exposure to new approaches to data-science project management, as will be demonstrated through our findings.

3.3 Data
The study was multi-method, drawing on varied sources of data.

3.3.1 Workflow data. We collected the workflows and R notebooks created by the teams who used MIDST in the face-to-face sections and the R code of those in the RStudio sections. We analyzed those by counting the number of nodes created, the network characteristics, the length and types of nodes and completeness of the analyses performed.

3.3.2 Usage log data. The MIDST system logs user interaction with MIDST. We analyzed the logs for the face-to-face classes. The log data used for the analysis were collected over the last two months of the semester, ending just after the final projects were due. The log actions recorded are listed below with brief descriptions:

- view network: view the project workflow diagram.
- view node: open a node, to view its contents or to work on coding in the notebook. (Unfortunately, we were not able to log interactions with the Jupyter notebook server.)
- started node run: a node/notebook is run by a project member.
- started network run: the entire network/project is run.
- pushed nodes: push changes in notebook to share with team members.
- pulled nodes: pull the node to retrieve the most recent version of the code.

The system also logged manual changes to the development status of the node as shown in Figure 3. We analyzed updates to the status of nodes across each project to understand usage of Kanban in MIDST as well as their difference among different network structures.

3.3.3 Observational data. We observed 3 MIDST and 5 RStudio teams in the distance class working together for the first time on the project midway through the class. We further observed the RStudio teams the following week. The meetings were recorded (with the permission of the students) for analysis and lasted 20 minutes for the RStudio teams and 28 minutes for the MIDST teams. Unfortunately, because section instructors were able to allocate only limited in-class time to teamwork we were able to observe the teams only once or twice. Given the limited amount of data, in this paper we present only an initial analysis.

For analysis, the meeting was divided into two-minute slices that were coded for (i) the type of coordination and work being performed, (ii) the type of data-science task being performed and (iii) for other interactions. A 2-minute slice could have more than one code if there was more than one
kind of activity. We chose to code in 2-minute increments to simplify the coding task for the coders while still capturing data granular enough to be of interest. The specific codes are:

(i) Type of coordination or individual work:

- Status update: when a team member updates the others on their progress and status of the task assigned.
- Plan future work: discussion of planned future work for the project.
- Code / interfaces: discussion about the R code or the interface between two pieces of code.
- Individual work: working individually.
- Integrate code: discussion on combining pieces of codes of the project.

(ii) Type of data science task:

- Understand problem: discussion of problem formulation and brainstorming about steps to solution.
- Data preparation: cleaning, filling missing values, excluding columns and/or rows of data, handling outliers or other data preparation work.
- Modelling: discussion of machine learning or data-mining approaches to be used.
- Evaluation or visualization: visualizing the data and/or outcomes of data processing.
- Deploying code: sharing the code with team members.

(iii) Other activities:

- Socialization: time spent on social discussion.
- Discussion of tools: time spent discussing issues or problems with the tool, in particular, difficulties using MIDST.

The video observations were coded by graduate students who had previously taken the course and had worked with MIDST. The coders went through a brief training on the coding system that included discussions of how to recognize each type of work, coding a short example discussion, review of the code and discussion of disagreements or uncertainties and review of the process of recording the codes. Each meeting was coded by three coders. Coders could make free-form comments to describe the activity occurring in a time slot in addition to using the preset codes listed above.

For every 2-minute slice, we merged together the codes from the three coders, tracking how often each code was applied to the slice. We kept only codes that were applied two or more times (i.e., at least two coders agreed that the particular type of activity occurred in that time slot). Codes applied only by a single coder were dropped. The coders applied a total of 441 codes (counting after merging) across 284 minutes (142 2-minute slots) in the 13 recorded meetings. Of these, 267 had no match, leaving 174 code for analysis (after merging the repeated codes). The MIDST meetings had a total of 71 codes applied, the 1st week RStudio meetings 59 and the 2nd week RStudio meetings 44. These 174 codes were applied by two or more coders, so they represent more than half of the codes applied, but it is apparent that the level of agreement among the coders is only middling. The coding could be made more reliable, e.g., with more intensive training of the coders. However, as we have only limited data (e.g., only a few groups for one or two meetings) and this analysis plays a small part in this paper, it did not seem worthwhile to pursue improvement at this point in the project. We focus instead on the aspects of the meetings for which there is sufficient agreement.

3.3.4 Survey data. We surveyed the students about their experiences using the tools and coordinating with their team members on the project. We used the same survey as in Ref [6]. The survey asked a series of quantitative and qualitative questions about the impression of the utility of the tool, the team process and coordination and satisfaction with results. We received 118 responses
from the face-to-face class (41 from students in MIDST sections and 77 from RStudio sections) but only 13 from the distance class (6 for MIDST and 7 for RStudio).

4 FINDINGS
In this section we present findings from the data analysis to answer the research questions posed above.

4.1 Question 1: Do teams use MIDST for team work?
Our first question is whether the teams actually used MIDST’s features to support their team work. Establishing which teams did and did not use MIDST is important for this paper for two reasons. First, if some teams in fact did not use the tool, then including their data will paint a misleading picture of the tool’s performance. Second, determining usage makes it possible to compare users and non-users.

To assess whether teams actually used MIDST, we first examined the workflows they created. For this analysis we focus on the 18 face-to-face teams in the MIDST condition. The median project included 12 nodes (ranging from 4 to 21); the median node contained 7 Jupyter notebook cells and 29 lines of code (with a wide range, from 1 to 116 cells and 1 to 1217 lines of code). We present the workflows as a directed graph as shown in 5. The color of the node in these figures represents the owner of the node (unlike in the actual system, where colour represents the development status). The size of the node is proportional to the number of times an action was logged for that node, which suggests the quantity of effort or interaction spent in MIDST.

Students in the face-to-face section were required to submit the code for their projects in MIDST. However, a few teams apparently did the project mostly or entirely outside of MIDST (e.g., in RStudio) and simply copied their individual R code files into MIDST to submit, as shown in Figures 5 and 6. Note in particular the small size of the nodes, indicating few activities logged in MIDST. As most of the teams were composed of 4 students, we considered that workflows with fewer than 8 nodes (i.e., at least one team member with just one node) only used MIDST to submit their assignment. Three of the 18 projects fell into this category.

Fig. 5. A workflow represented as a directed graph; colour codes indicate the node owner and size represents the number of activities logged on the node.

Fig. 6. A MIDST workflow where the four disconnected nodes with little activity suggest that the work was done individually and outside of MIDST.
Even when networks had multiple nodes per student, in three cases the networks were disconnected, evidence that the team’s work was done separately rather than integrated. The network diagram of two of those projects are shown in Figure 7 and Figure 8.

To quantify the structural properties of the remaining workflows, we computed the betweenness centrality for each node. Computation of the betweenness centrality of a node starts by identifying the shortest path between all pairs of nodes; the betweenness centrality of a node is the fraction of such paths on which it sits. For this computation, we considered the workflows as undirected graphs. A node with high betweenness is a hot-spot in the analysis as it is part of the processing of data from or to many other nodes.

The distribution of betweenness scores was quite skewed, with most nodes have betweenness of 0 and only one or two having a high score. We divided the remaining projects by the number of nodes with betweenness greater than 0.4, chosen as it divided the high-score nodes from the majority. Table 1 shows the statistics for each project: the total number of nodes, the number of nodes with betweenness greater than 0.4 and basic statistics for the betweenness scores.

Four projects had only 1 node with betweenness > 0.4. The diagram of two such projects are shown in Figure 9 and Figure 10. Examining these workflows, we see some evidence of division of labour among the team members: specifically, it seems that one team member took care of data cleaning and preparation of a dataset on which a series of analyses were run. However, in these cases the analyses were mostly run separately.

Finally, the remaining 9 projects had 2 or more nodes with a betweenness centrality > 0.4. The diagram of two example projects are shown in Figure 11 and Figure 12. These projects show a high level of interconnected work. Furthermore, the size of some of the nodes suggests that development of at least some of the code was done in MIDST.

Different sections had different distributions of the teams with more-or-less complex workflows, with more of the more complex ones found in the sections that had used MIDST for an individual assignment before the project. However, the relationship between section and complexity was not statistically significant ($\chi^2(6) = 10.8, p = 0.1$).

Further evidence of the different levels of usage of MIDST comes from the use of the kanban features that allow team members to report and track the development status of nodes. The results shown in Table 2 shows that the kanban was used differently across the teams in the MIDST. The figure shows the number of times a node was moved to a particular status per project. If this project...
Table 1. Betweenness Statistics

<table>
<thead>
<tr>
<th>Project Category</th>
<th># nodes &gt; 0.4</th>
<th>Nodes Count</th>
<th>mean</th>
<th>std</th>
<th>min</th>
<th>50%</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fewer than 8 nodes</td>
<td>0</td>
<td>5</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>4</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6</td>
<td>0.333</td>
<td>0.273</td>
<td>0.000</td>
<td>0.400</td>
<td>0.600</td>
</tr>
<tr>
<td>Disconnected</td>
<td>0</td>
<td>8</td>
<td>0.006</td>
<td>0.017</td>
<td>0.000</td>
<td>0.000</td>
<td>0.048</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>15</td>
<td>0.010</td>
<td>0.027</td>
<td>0.000</td>
<td>0.000</td>
<td>0.099</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>14</td>
<td>0.031</td>
<td>0.094</td>
<td>0.000</td>
<td>0.000</td>
<td>0.346</td>
</tr>
<tr>
<td>One central</td>
<td>1</td>
<td>8</td>
<td>0.155</td>
<td>0.337</td>
<td>0.000</td>
<td>0.000</td>
<td>0.952</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>10</td>
<td>0.139</td>
<td>0.298</td>
<td>0.000</td>
<td>0.000</td>
<td>0.944</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>13</td>
<td>0.135</td>
<td>0.251</td>
<td>0.000</td>
<td>0.000</td>
<td>0.924</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>9</td>
<td>0.151</td>
<td>0.284</td>
<td>0.000</td>
<td>0.018</td>
<td>0.875</td>
</tr>
<tr>
<td>Two or more central</td>
<td>2</td>
<td>9</td>
<td>0.151</td>
<td>0.318</td>
<td>0.000</td>
<td>0.000</td>
<td>0.893</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>15</td>
<td>0.140</td>
<td>0.244</td>
<td>0.000</td>
<td>0.000</td>
<td>0.758</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>21</td>
<td>0.088</td>
<td>0.207</td>
<td>0.000</td>
<td>0.000</td>
<td>0.747</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>12</td>
<td>0.139</td>
<td>0.248</td>
<td>0.000</td>
<td>0.000</td>
<td>0.709</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>12</td>
<td>0.171</td>
<td>0.264</td>
<td>0.000</td>
<td>0.000</td>
<td>0.764</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>12</td>
<td>0.129</td>
<td>0.278</td>
<td>0.000</td>
<td>0.000</td>
<td>0.891</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>19</td>
<td>0.101</td>
<td>0.225</td>
<td>0.000</td>
<td>0.000</td>
<td>0.837</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>17</td>
<td>0.152</td>
<td>0.223</td>
<td>0.000</td>
<td>0.125</td>
<td>0.775</td>
</tr>
</tbody>
</table>

In summary, it seems that about two-thirds of the teams who were assigned to use MIDST actually used its features to support team coordination. In further analyses, we will compare teams with workflows in the four levels of complexity, which for ease of reference we will label from A to D as shown in Table 2.

management feature were used as intended, each node would have moved through all five statuses, i.e. a node is created in the proposed state, moved to not started when accepted and assigned, moved to in progress when it is worked on, then validating and completed. However, both 13 and Table 2 show that project teams didn’t always use the intervening stages but often simply moved nodes to the completed state, i.e., marking completion instead of real-time tracking of the status of each nodes in the project. Further, the counts suggest that the projects with the least developed workflows also used the project management features least.
4.2 Question 2: Usage of affordances

Our next question was whether the users of the MIDST system show evidence of being able to use the intended affordances. We consider first legibility and then visibility and combinability.

4.2.1 Legibility of work. For work contributions to be useful as a basis for coordinating work, team members must be able to recognize the purpose of a contribution and thus what they can or should do with that contribution. In the case of MIDST, the contributions are the nodes with code added to the workflow. The work in the less developed workflows is not very understandable, e.g., a node called “Analysis 1” might do anything (Figure 7). In contrast, in the more complex workflows, we see a common structure in which data are loaded and cleaned and then a set of analyses are run more or less independently. The data cleaning node has high betweenness, while the rest are low or even 0. Figure 11 is unusual in that it has four cleaning nodes, preceded by splitting the dataset and followed by merging it. Talking to the students, we learned that this team resisted division of labour: they felt it was important than all team members experience all phases of the project, data cleaning included, hence the multiple data cleaning nodes. But even in this case, the function of the nodes seems to fall into regular patterns, suggesting that team members had learned at least an initial set of genres for the work.

4.2.2 Visibility and combinability of code. Before a node can be recognized it must be visible to other team members; once it is recognized, it must be possible to combine the work. To explore the visibility and combinability of the work contributions, we examined the logs of the usage of different
Table 2. Kanban Statistics

<table>
<thead>
<tr>
<th>Project Categories</th>
<th>Kanban States</th>
<th>Used by projects</th>
<th>Modules Count mean</th>
<th>std</th>
<th>min</th>
<th>50%</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Fewer than 8 nodes (Total of 3 projects)</td>
<td>1. proposed</td>
<td>1 of 3</td>
<td>1.00</td>
<td>NA</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>2. not started</td>
<td>1 of 3</td>
<td>1.00</td>
<td>NA</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>3. in progress</td>
<td>2 of 3</td>
<td>1.50</td>
<td>0.71</td>
<td>1.0</td>
<td>1.5</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>4. validating</td>
<td>1 of 3</td>
<td>1.00</td>
<td>NA</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>5. completed</td>
<td>1 of 3</td>
<td>2.00</td>
<td>NA</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>B. Disconnected (Total of 3 projects)</td>
<td>1. proposed</td>
<td>3 of 3</td>
<td>2.33</td>
<td>1.15</td>
<td>1.0</td>
<td>3.0</td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td>2. not started</td>
<td>2 of 3</td>
<td>3.00</td>
<td>2.83</td>
<td>1.0</td>
<td>3.0</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td>3. in progress</td>
<td>2 of 3</td>
<td>4.50</td>
<td>0.71</td>
<td>4.0</td>
<td>4.5</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td>4. validating</td>
<td>2 of 3</td>
<td>1.00</td>
<td>0.00</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>5. completed</td>
<td>3 of 3</td>
<td>7.33</td>
<td>3.06</td>
<td>4.0</td>
<td>8.0</td>
<td>10.0</td>
</tr>
<tr>
<td>C. One central node (Total of 4 projects)</td>
<td>1. proposed</td>
<td>1 of 4</td>
<td>2.00</td>
<td>NA</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>2. not started</td>
<td>2 of 4</td>
<td>2.00</td>
<td>1.41</td>
<td>1.0</td>
<td>2.0</td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td>3. in progress</td>
<td>2 of 4</td>
<td>2.50</td>
<td>0.71</td>
<td>2.0</td>
<td>2.5</td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td>4. validating</td>
<td>3 of 4</td>
<td>3.67</td>
<td>2.31</td>
<td>1.0</td>
<td>5.0</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td>5. completed</td>
<td>3 of 4</td>
<td>8.67</td>
<td>3.79</td>
<td>6.0</td>
<td>7.0</td>
<td>13.0</td>
</tr>
<tr>
<td>D. Two or more central nodes (Total of 8 projects)</td>
<td>1. proposed</td>
<td>3 of 8</td>
<td>3.00</td>
<td>1.00</td>
<td>2.0</td>
<td>3.0</td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td>2. not started</td>
<td>4 of 8</td>
<td>4.50</td>
<td>4.51</td>
<td>1.0</td>
<td>3.0</td>
<td>11.0</td>
</tr>
<tr>
<td></td>
<td>3. in progress</td>
<td>7 of 8</td>
<td>3.71</td>
<td>2.43</td>
<td>1.0</td>
<td>4.0</td>
<td>7.0</td>
</tr>
<tr>
<td></td>
<td>4. validating</td>
<td>6 of 8</td>
<td>3.83</td>
<td>2.86</td>
<td>1.0</td>
<td>3.5</td>
<td>7.0</td>
</tr>
<tr>
<td></td>
<td>5. completed</td>
<td>8 of 8</td>
<td>9.75</td>
<td>6.82</td>
<td>1.0</td>
<td>10.5</td>
<td>21.0</td>
</tr>
</tbody>
</table>

system features. We divided these by the different groups A–D developed above. We considered running a single node, viewing the workflow and viewing a node as individual actions. Plots of the counts of these activities per project team are shown in Figure 14. We note that the counts of these activities are higher in the teams with more complex workflows, reflecting that the members of those teams did more of their work in MIDST. An ANOVA shows that counts of viewing the network and a node are significantly different across the four conditions ($F(3, 14) = 6.3, p = 0.006$ for viewing the network and $F(3, 14) = 4.1, p = 0.03$ for viewing a node) but not for running a single node.\(^2\)

We consider pulling and pushing code and running the entire workflow (i.e., including the code of other team members) as indications of collaborative development, specifically actions to make work visible to others and to combine work with their code. Again, as shown in Figure 15, these activities are more common for teams with more complex workflow. As noted earlier, the more complex workflows are also more interconnected, evidence that team members were building on each others’ work. However, an ANOVA shows that the differences in these counts among the groups are not statistically significant.

Finally, there is evidence that team members made use of the visibility of code, specifically by viewing or running a node that is assigned to another team member, as shown in Figures 16 and 17. However, only the counts of viewing someone else’s node is statistically significantly different.

\(^2\)The counts for all eight actions are statistically significantly different when comparing teams in groups A and B combined to the teams in groups C and D combined.
across the conditions ($F(3, 14) = 4.4, p = 0.02$). Collectively, the frequency of these actions suggest that those who used MIDST did experience its affordances for visibility and combinability.

4.3 **Question 3: Do teams coordinate stigmergically?**

The evidence of the previous section suggests that members of that teams that used MIDST did use the system features to view each others code and track progress on the project. The third question we wish to explore is whether teams using MIDST are actually able to coordinate stigmergically, that is, via the shared work.

Evidence for this question comes first from the survey. We report only on the face-to-face teams for which we had a large number of responses. A problem we encountered in the analysis of the surveys is that they were anonymous, meaning that we did not know which team a respondent was in, only which condition, MIDST or RStudio. And as noted above, only about 2/3 of the MIDST
teams seem to have actually used the tool to support their teamwork. As a result, comparison of the quantitative data was inconclusive. We focus instead on a qualitative analysis of open-ended questions about how the students coordinated their teamwork and how they found the status of tasks that other team members were working on. These questions were analyzed by one of the authors by looking for mentions of specific coordination technologies or approaches in the free-text answers.

For the RStudio teams, the most common answer for how the team coordinated (mentioned by about 1/3 of respondents who provided an answer) was face-to-face meetings, typically once or twice a week. Some mentioned the strategy of dividing work into independent pieces to reduce the need to coordinate. For finding task status, meetings or Trello were about as common (students were advised to use Trello in the course). Students also mentioned using Whatsapp or another chat program to stay in touch. A few students specifically noted problems sharing code, e.g., “Sharing code was tough and clunky” or “Rstudio didn’t let us share the code, so we had to download the code shared by the team members and then integrate it into our own code and run it”. Some mentioned using Google Docs or Github to share code or work status, along with occasional mentions of a number of other tools (e.g., Blackboard or Teams). About 1/2 of the respondents noted communications problems in their team. A common theme here was the difficulty of finding a time to meet given peoples’ busy schedules.

In contrast, only a few respondents in the MIDST condition mentioned meetings or use of any other coordination technology as a way to coordinate. Most reported using MIDST as the way to find task status. And reports of communications problems were rare, though complaints about slacking team members were equally common in the two conditions. In summary, the qualitative data from the survey suggests that the MIDST teams faced less difficulty coordinating their work and needed less support to do, which is consistent with being able to use stigmergic coordination.

![Fig. 18. Frequency of codes applied to time for MIDST and R-Studio teams in first project week](image1)

![Fig. 19. Frequency of codes applied to time for R-Studio teams in second project week](image2)
Further evidence comes from the observation of team meetings in the distance class and the analysis of how the teams spent their time together. Figure 18 shows the frequency of different kinds of work for the teams using MIDST vs. those using RStudio in the first meeting and Figure 19 for just the RStudio teams in the second meeting. The codes are defined in section 3. The MIDST and RStudio distributions are statistically significantly different ($\chi^2(10) = 56.2, p < 0.001$). Comparing the two distributions, it seems that the MIDST teams spend less time than the RStudio teams on coordination activities towards the right of the figure (e.g., status update or planning future work) and more time on data-science work towards the left of the figure (e.g., coding and interfaces, data preparation, modelling), consistent with a shift from explicit to stigmergic coordination. On the other hand, the distributions of the two RStudio meetings are not statistically significantly different ($\chi^2(9) = 11.8, p = 0.2$), suggesting that the coding has captured something real about how these teams meet.

4.4 Question 4: Do teams using MIDST do better projects?

The final question is if using MIDST enables teams to do better work. We have already provided evidence of one advantage, namely better integration of the project outputs. The teams who really used MIDST are not just putting together slides reporting separate efforts. To further evaluate the analysis done within MIDST, we counted:

- The number of nodes in the workflow
- The number of machine-learning algorithms used (e.g., Support Vector Machine, Decision Trees). There was an expectation for the project that teams would use a minimum of 3 algorithms.
- The number of visualizations created (to represent the number of attributes evaluated via exploratory analysis). There was an expectation of a minimum of 15 visualizations.
- The number of map visualizations created. There was an expectation of minimum of 1 mapping visualizations.

We again divided the teams into four groups based on the complexity of their workflows. We also compare against the final projects of the teams using RStudio. Note that number of nodes (or files for RStudio) and number of analyses are independent, because a single node can contain multiple models and visualizations. Table 3 shows the fraction of the project teams that met the expected standard. Table 4 gives the count of the number of each analysis and the number of nodes in the workflow. The projects with more complex workflows did more in terms of the count of analyses performed. The differences between the number of analyses done by teams in groups A and B vs. groups C and D were statistically significant (using a Welch Two Sample t-test in R, $t(9.2) = -4.2, p = 0.002$, $t(14.9) = -5.0, p < 0.001$ and $t(11.1) = -2.3, p = 0.04$ for the number of models, visualizations and map visualizations respectively).

<table>
<thead>
<tr>
<th># of Projects</th>
<th>% of projects above expectations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3 Models</td>
</tr>
<tr>
<td>RStudio</td>
<td>9</td>
</tr>
<tr>
<td>Fewer than 8 nodes</td>
<td>4</td>
</tr>
<tr>
<td>Disconnected networks</td>
<td>2</td>
</tr>
<tr>
<td>1 node with betweenness &gt; 0.4</td>
<td>4</td>
</tr>
<tr>
<td>2+ nodes with betweenness &gt; 0.4</td>
<td>8</td>
</tr>
</tbody>
</table>
Table 4. Average (and standard deviation) of count of analyses performed per project by complexity of workflows

<table>
<thead>
<tr>
<th>Complexity</th>
<th># of Projects</th>
<th># of nodes or files</th>
<th>Models</th>
<th>Viz</th>
<th>Map Viz</th>
</tr>
</thead>
<tbody>
<tr>
<td>RStudio</td>
<td>9</td>
<td>3.8 (2.7)</td>
<td>3.4 (1.0)</td>
<td>15.1 (5.3)</td>
<td>0.7 (0.7)</td>
</tr>
<tr>
<td>Fewer than 8 nodes</td>
<td>4</td>
<td>4.5 (1.3)</td>
<td>2.3 (0.5)</td>
<td>10.3 (2.1)</td>
<td>0.0 (0.0)</td>
</tr>
<tr>
<td>Disconnected network</td>
<td>2</td>
<td>14.5 (0.7)</td>
<td>3.0 (0.0)</td>
<td>6.0 (5.7)</td>
<td>0.5 (0.7)</td>
</tr>
<tr>
<td>1 node with betweenness &gt; 0.4</td>
<td>4</td>
<td>10.0 (1.9)</td>
<td>3.8 (0.4)</td>
<td>22.3 (12.5)</td>
<td>3.8 (3.7)</td>
</tr>
<tr>
<td>2+ nodes with betweenness &gt; 0.4</td>
<td>8</td>
<td>15.7 (3.8)</td>
<td>4.6 (0.7)</td>
<td>35.6 (9.8)</td>
<td>9.7 (11.9)</td>
</tr>
</tbody>
</table>

The RStudio teams fall in the middle of the range. The RStudio teams were not statistically significantly different from projects in either groups A and B combined or groups C and D combined for models and map visualizations but were different from both for visualizations ($t(10.3) = -2.5, p = 0.03$ comparing RStudio projects to the projects in groups A and B and $t(16.2) = 3.4, p = 0.003$ to groups C and D).

5 DISCUSSION

The results presented above suggest that MIDST does help teams to work together in a different way. As shown in Figure 18, teams seem to spend their initial time together differently. Strikingly, MIDST teams seem to spend less time on coordination, e.g., doing status updates, developing mutual understanding of the problem or discussing future work. Rather, they seem to be able to more immediately work on the task at hand, working on data preparation, coding, different kinds of analyses and integrating results. Even though the data come from the first team meeting, it seems that having a shared work environment allows the team members to actually start working together.

In contrast, the members of the RStudio teams will be working individually on their parts of the project in RStudio; there is no shared space for the team. During a face-to-face discussion, one team member could share their copy of RStudio with the others, but only one person could be entering code and at the end of the sessions, the others would not necessarily have advanced on their parts of the project. By the second meeting of the RStudio teams, they are spending some time on the task, but they still seem to have a greater need for explicit coordination. In short, while the RStudio teams spent their time talking about work, the MIDST teams were able to start doing it.

As the project progresses, the evidence from section 4.2 is that the system does seem to enable teams that use it to see each others’ status and code, which the qualitative survey results suggest enable them to coordinate their teamwork more easily. As well, MIDST makes it easier to develop modular code that is easier to understand and to integrate. Overall, it appears that teams using MIDST accomplish a more thorough analysis.

Of course, these results are indicative rather than corroborative. The middling level of agreement on the coding of the meetings is a concern (though the stability of coding from one week to the next for the RStudio teams suggest it has captured something real about the way the teams work). The lack of a control group for the within-MIDST analysis precludes conclusions about causality. For instance, the apparent relation between the use of MIDST features and the complexity of the workflow and the thoroughness of the analysis could be spurious. While there’s no reason in principle why a team could not do a thorough analysis in RStudio (even if they submit it via MIDST), it could be that teams trying to manage the time they spend on the course do both a less thorough job on the project and avoid dealing with an experimental tool. Still, our conclusions
are based on multiple sources of data and types of analysis. No single result is conclusive but the combination of multiple forms of evidence pointing in the same direction may be more persuasive.

The result that six of teams in the MIDST condition made minimal use of MIDST was somewhat unexpected. To hypothesize reasons for this disuse we note that intention to use a technology can be predicted by the technology’s perceived usefulness and perceived ease of use [8].

Starting with ease of use, it has to be admitted that during the semester we were attempting to evaluate the system with the face-to-face sections, it was still being worked on and did have occasional bugs. (Indeed, the primary reason for working with a face-to-face class was to be better able to support students when they had problems.) A number of the comments on the survey allude to these problems. Further, as a research prototype, its functionality is not at the level of a commercial project such as RStudio, which all the students experienced first. The system was more stable for the distance students, but there was also much less opportunity to introduce the system and to support users, also impacting ease of use.

While ease of use might be an explanation by itself, it also seems that a number of teams didn’t perceive the tool as useful, which we found surprising given the difficulty of sharing code experienced by the RStudio teams and the ease of doing so in MIDST. One explanation is that MIDST is based on the notion of a shared work product. However, not all teams attempted to create an integrated product, being content instead to run separate analyses that were integrated only by being presented together.

Whether this lack of interdependence among work products is characteristic of data science in general or just of student projects is not clear. In our discussion of genres above, we noted a common structure of a central node for data cleaning and feature development followed by many more-or-less independent analyses of the data that might in fact not require much coordination. On the other hand, we also noted that data analysis is often iterative, where the findings of one analysis might have implication for another. However, those implications are at a different level than the flow of data that is captured in the workflow. It is not clear how best to capture and share these insights as they are developed.

Another explanation may lie in the socio-technical nature of the affordances need to support stigmergic coordination. As noted above, the technical features of MIDST need to be supported by team practices. It could be that the teams are not used to working together and so did not have the work practices needed to make MIDST effective. For instance, students may not be in the habit of sharing code they write with others or of tracking others’ progress on distributed work, making that the visibility provided by MIDST less useful.

With regards to combinability, we already noted the difficulties of writing modular and maintainable analysis code, problems that seem to be experienced by data scientists in general, as noted in literature review. While MIDST has support for modularity, it could do more to encourage it. To encourage both writing more focused nodes and more legible ones, a possible MIDST feature is a pre-populated library of standard nodes, e.g., nodes for regression, visualizations and so on.

It is also worth noting that the students in the sections that used MIDST for an individual assignment before doing the project tended to have more complex workflows for the project. Though the relationship is not statistically significant, it does suggest that it simply takes a bit of time and practice to learn how to work in a new way. A takeaway here is that teams need training in how to work as a team as well as in the particular features of MIDST. The growing literature on team processes for data science [24] suggests that these problems are not unique to students, but working with students offers an opportunity to address them. The data science course that was the site for our study does include some training, e.g., on the importance of modularity and on kanban as a project management approach; this material could be extended and tied to features of MIDST that they support.
6 CONCLUSION

We plan to further evaluate the use and benefits of MIDST with a larger sample that enables more thorough statistical analysis of differences and a more rigourously-implemented study design that supports causal conclusions. To do so, we seek to make the system more acceptable to users. The developers of MIDST have already made advances in ease of use, e.g., by incorporating Jupyter notebook as the coding interface. We also need to address the perceived usefulness of the system by making it clearer what problem the system solves. If the system is tested with students, the project requirements might be tweaked to reward more integrated solutions. Our experience with the difficulties students faced working together suggests teamwork practices that might be encouraged.

In future work, we also plan to use the system as the basis for building additional support for data-science teams. A particular interest is the possibility of adding non-human members to a team. An automated system could assess data quality, e.g., spotting outliers or problems with missing data, suggest transformations to correct skew or more ambitiously, notice bias in the data. It could create additional data columns, e.g., breaking up complex data into components. A system could suggest additional actions for an analysis, e.g., useful visualizations or modelling approaches given what it knows about the data or diagnostics for a user-selected analysis. If the assumptions of a test are violated, it could suggest an alternative, e.g., a non-parametric test instead of a parametric one.

Having the system work in the same work space as the users would support stigmergic coordination between the human user and automated system. Interventions in the process could be made by creating a note on notebook cell with suggested changes or creating additional cells or entire nodes, e.g., the cells to run and interpret diagnostics for an analysis or a node to create a visualization. The system could communicate intent or certainty by adding comments to the code. If the system intervenes by providing code to run, the user could edit the code if not appropriate. To make cells function smoothly together (i.e., combinability) does require some additional work, e.g., identifying which variables hold the necessary data. For a user to be able to use suggestions made by an automated system, they need to be able to recognize what those contributions do and how to use them. Such legibility could be explicitly supported, e.g., by commenting in the code.

Finally, the apparently beneficial impact of MIDST for the data-science teams that used it raises the question of where else a stigmergic approach might be useful. The principles of visibility, legibility and combinability are ones that can be transferred from domain to domain. But given that stigmergic coordination is embedded in the particular work, supporting it generically seems difficult. Future research could look for ways to generalize across different kinds of work products to support broader classes of tasks and teams.

ACKNOWLEDGMENTS

This work was partially supported by a grant from the US National Science Foundation, IIS 16–18444.

REFERENCES


