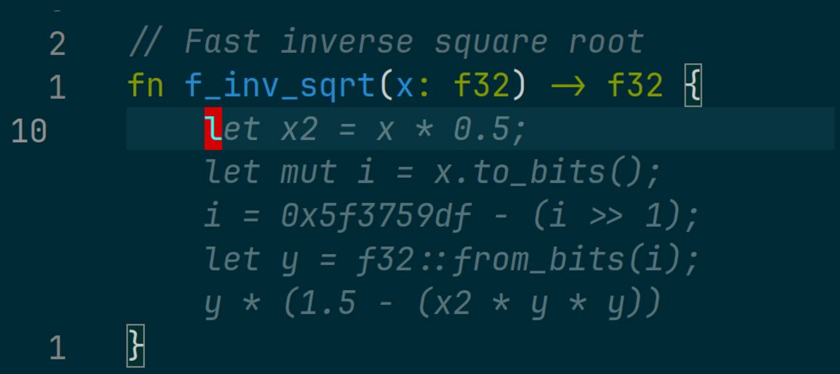
Exploring and Validating AI-Generated Programs Through Concrete Values

Kasra Ferdowsi

UC San Diego

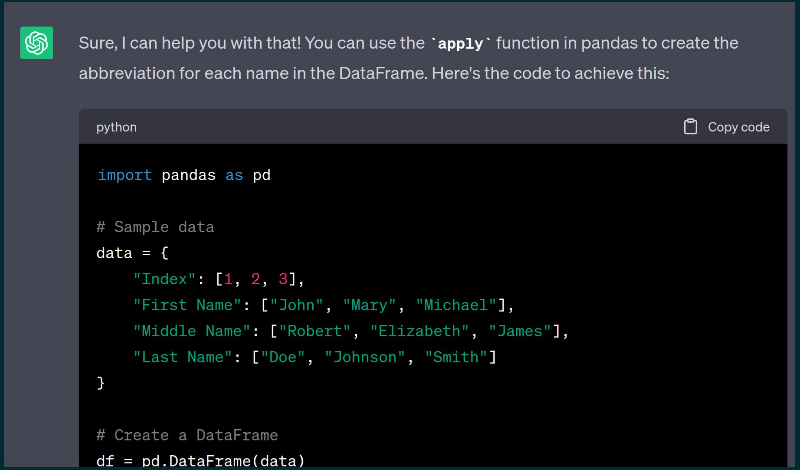
The Usability of LLM Code Generation

GitHub Copilot



```
// Fast inverse square root
fn f_inv_sqrt(x: f32) → f32 {
    let x2 = x * 0.5;
    let mut i = x.to_bits();
    i = 0x5f3759df - (i >> 1);
    let y = f32::from_bits(i);
    y * (1.5 - (x2 * y * y))
}
```

OpenAI ChatGPT



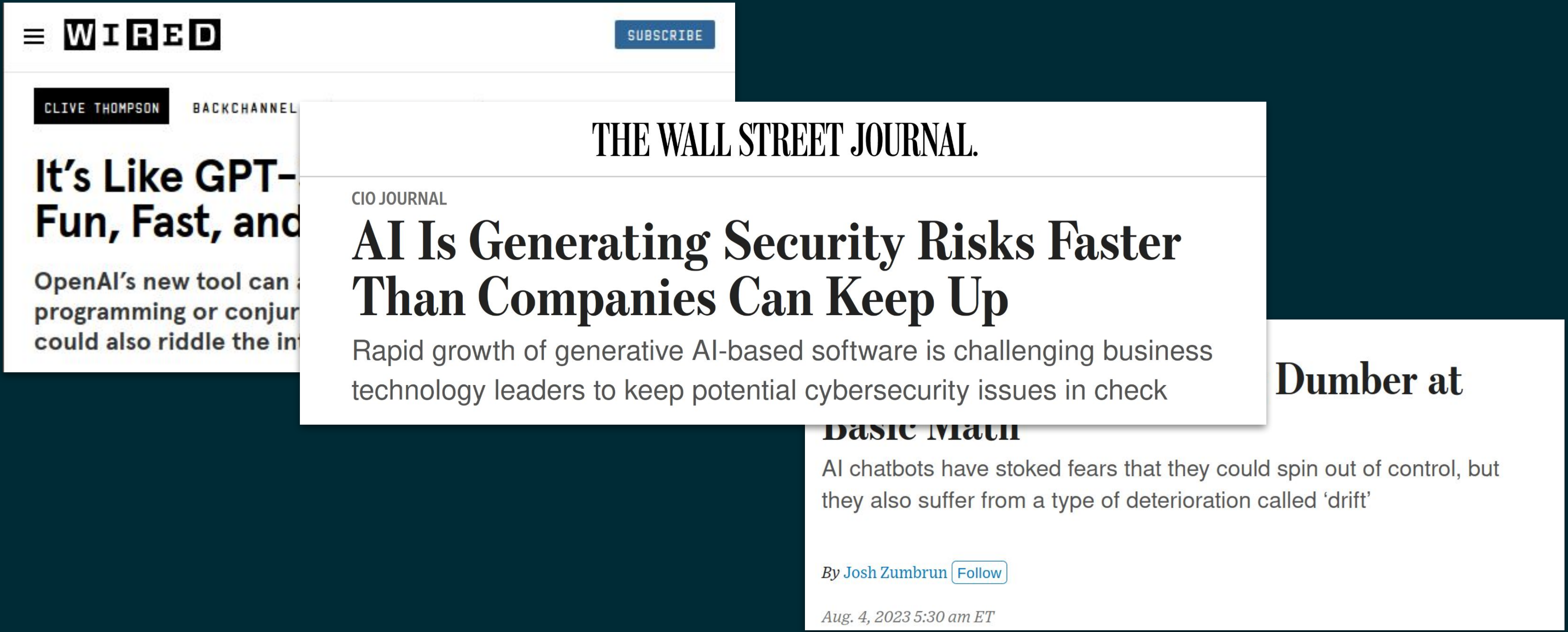
Sure, I can help you with that! You can use the `apply` function in pandas to create the abbreviation for each name in the DataFrame. Here's the code to achieve this:

```python
import pandas as pd

# Sample data
data = {
    "Index": [1, 2, 3],
    "First Name": ["John", "Mary", "Michael"],
    "Middle Name": ["Robert", "Elizabeth", "James"],
    "Last Name": ["Doe", "Johnson", "Smith"]
}

# Create a DataFrame
df = pd.DataFrame(data)
```

The Usability of LLM Code Generation



WIRED

SUBSCRIBE

CLIVE THOMPSON BACKCHANNEL

It's Like GPT-
Fun, Fast, and

OpenAI's new tool can
programming or conjur
could also riddle the in

THE WALL STREET JOURNAL.

CIO JOURNAL

AI Is Generating Security Risks Faster Than Companies Can Keep Up

Rapid growth of generative AI-based software is challenging business technology leaders to keep potential cybersecurity issues in check

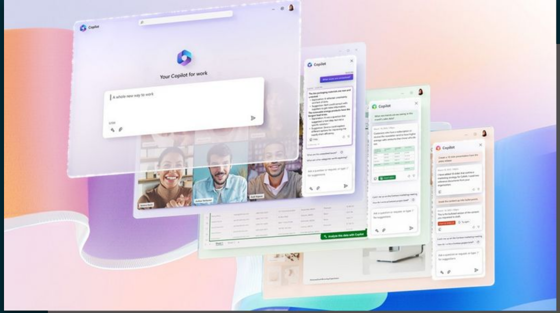
Dumber at
Basic Math

AI chatbots have stoked fears that they could spin out of control, but they also suffer from a type of deterioration called 'drift'

By Josh Zumbrun Follow

Aug. 4, 2023 5:30 am ET

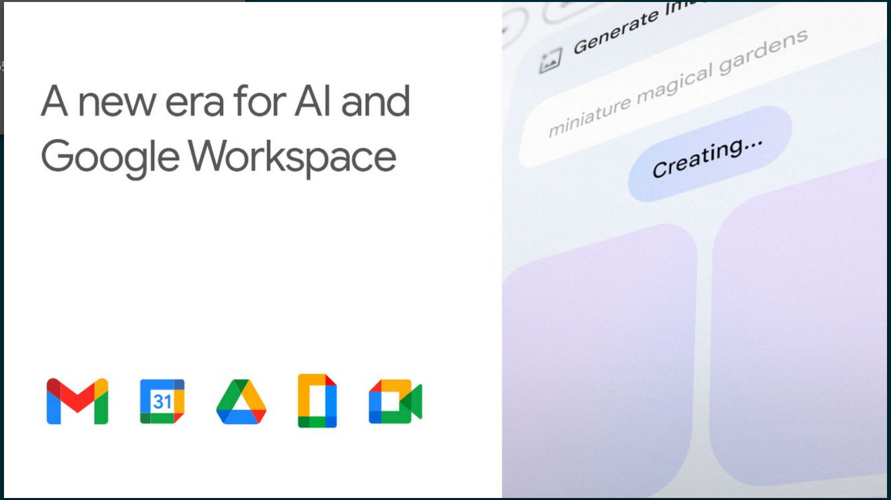
The Usability of LLM Code Generation



News · Microsoft 365 · Enterprise · 7 min read

Introducing Microsoft 365 Copilot—A whole new way to work

By Colette Stallbaumer, General Manager, Microsoft 365



A new era for AI and Google Workspace

In Summary…

Validating AI-generated programs is becoming a part of our lives,

So *programmers* and *end users* alike need affordances for doing so!

Overview

LEAP:

Live Exploration of AI-Generated Code

Programmers

ColDeco:

An End User Spreadsheet Inspection Tool for AI-Generated Code

End users

Overview

LEAP:

Live Exploration of AI-Generated Code

ColDeco:

An End User Spreadsheet Inspection Tool for AI-Generated Code

1. The Cost of Validation
2. LEAP demo
3. User Study

Overview

LEAP:

Live Exploration of AI-Generated Code

1. The Cost of Validation
2. LEAP demo
3. User Study

ColDeco:

An End User Spreadsheet Inspection Tool for AI-Generated Code

The Cost of Validation

Programmers using AI-generated code...

- Spend significant time *validating* code suggestions,
- Have trouble evaluating the correctness of generated code,
- Choose validation strategies based on *time cost*, and so
- Both *under- and over-rely* on AI code suggestions.

[Barke et al. 2023, Liang et al. 2023, Mozannar el al. 2022, Vaithilingam et al. 2022]

Live
Programming

The Cost of Validation

Does *Live Programming* offer a good interaction for *validating* AI-generated code?

Overview

LEAP:

Live Exploration of AI-Generated Code

ColDeco:

An End User Spreadsheet Inspection Tool for AI-Generated Code

1. The Cost of Validation
2. LEAP demo
3. User Study

Overview

LEAP:

Live Exploration of AI-Generated Code

1. The Cost of Validation
2. LEAP demo
3. User Study

ColDeco:

An End User Spreadsheet Inspection Tool for AI-Generated Code

User Study

How does Live Programming affect...

1. Code Correctness
2. Over-/Under-Reliance on AI
3. Cognitive Load

Between Subjects study:

17 Participants

2 Conditions:

1. AI
2. AI + LP

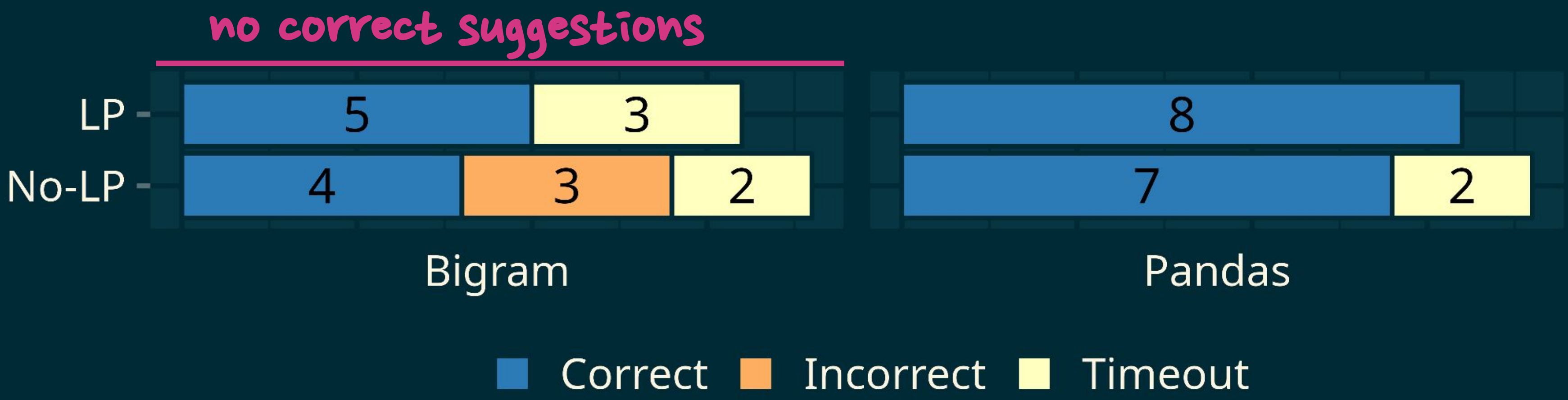
→ NO-LP

LP

Tasks

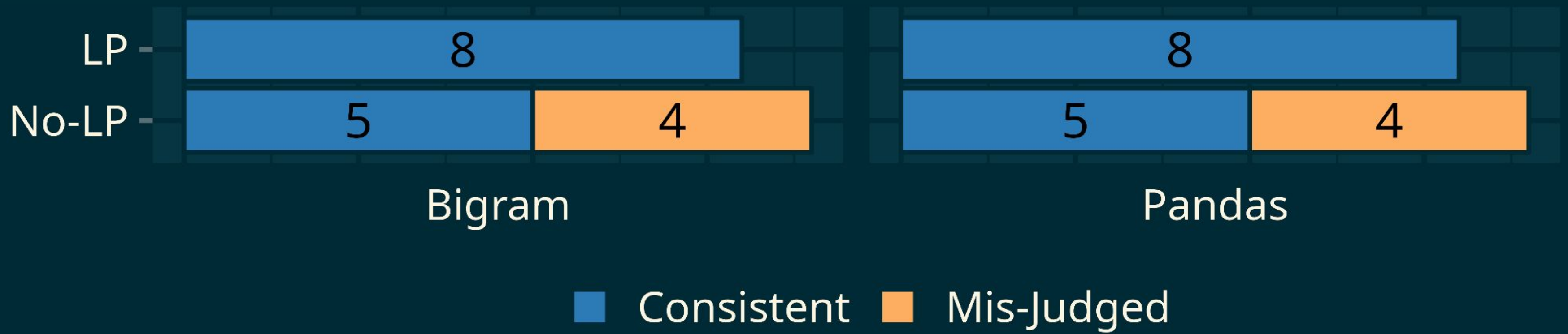
	API-Heavy	Algorithmic
Fixed-prompt	Pandas	Bigrams
Open-prompt	Box Plot	String Rewriting

RQ1: Correctness



Live programming helps validate suggestions!
(But does not help fix incorrect ones)

RQ2: Over-/Under-reliance



6 no-LP vs 0 LP participants **mis-judged** correctness of their solutions

RQ2: Over-/Under-reliance

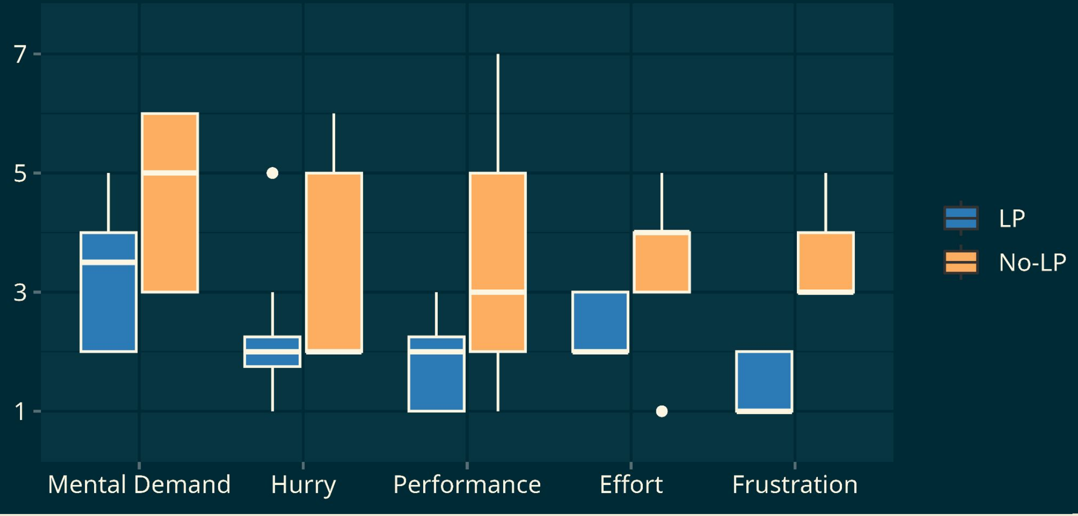
"it was **easy to understand** the behavior of a code suggestion because the little boxes on the side allowed for you to preview the results." (P3)

"it **saved me the effort** of writing multiple print statements." (P1)

Live programming reduces over-/under-reliance on AI, by lowering the *cost of validation*.

RQ3: Cognitive Load

NASA TLX metrics on the Pandas task



Live programming significantly reduced the *cognitive load* of exploration for *tasks amenable to validation by execution.*

In Summary...

Live Programming is *not a panacea*. But!

It’s really powerful for reducing the *cost of validating* AI-generated programs.

Overview

LEAP:

Live Exploration of AI-Generated Code

1. The Cost of Validation
2. LEAP demo
3. User Study

ColDeco:

An End User Spreadsheet Inspection Tool for AI-Generated Code

Overview

LEAP:

Live Exploration of AI-Generated Code

1. The Cost of Validation
2. LEAP demo
3. User Study

ColDeco:

An End User Spreadsheet Inspection Tool for AI-Generated Code

1. End User Programming
2. ColDeco Example
3. Implementation
4. User Study

Overview

LEAP:

Live Exploration of AI-Generated Code

1. The Cost of Validation
2. LEAP demo
3. User Study

ColDeco:

An End User Spreadsheet Inspection Tool for AI-Generated Code

1. End User Programming
2. ColDeco Example
3. Implementation
4. User Study

End User Programming

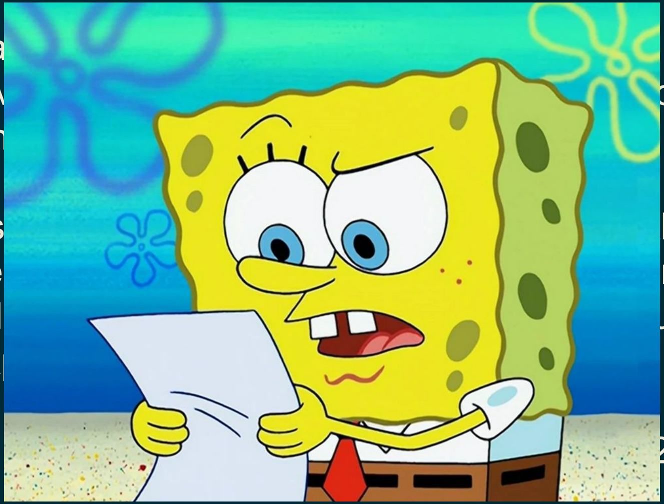
Live Programming for free

End Users can’t always read the code

Pa as
“ov ow
wh

As he
pe ng
ad ual
ex

21]



Spreadsheets & AI

Can we leverage *familiar spreadsheet concepts* for end user *validation* of AI-generated code?

Helper Columns

Filtering

Intermediate variables

Program Slicing!

Overview

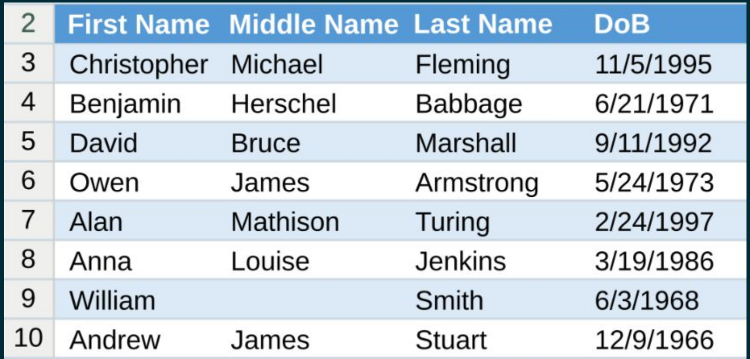
LEAP:
Live Exploration of AI-Generated Code

1. The Cost of Validation
2. LEAP demo
3. User Study

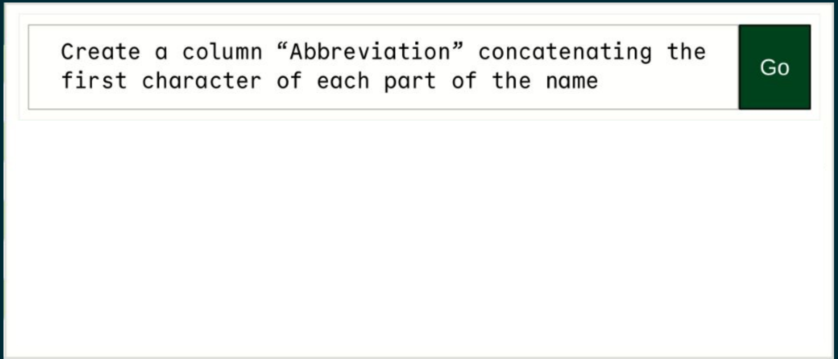
ColDeco:
An End User Spreadsheet Inspection Tool for AI-Generated Code

1. End User Programming
2. ColDeco Example
3. Implementation
4. User Study

ColDeco



2	First Name	Middle Name	Last Name	DoB
3	Christopher	Michael	Fleming	11/5/1995
4	Benjamin	Herschel	Babbage	6/21/1971
5	David	Bruce	Marshall	9/11/1992
6	Owen	James	Armstrong	5/24/1973
7	Alan	Mathison	Turing	2/24/1997
8	Anna	Louise	Jenkins	3/19/1986
9	William		Smith	6/3/1968
10	Andrew	James	Stuart	12/9/1966

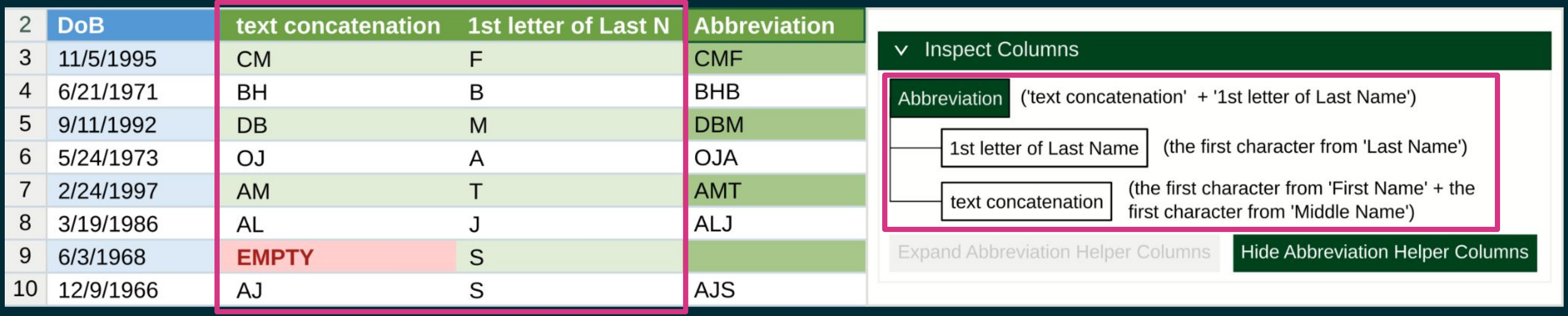


Create a column “Abbreviation” concatenating the first character of each part of the name

```
df['Abbreviation'] = \
    df['First Name'].str[0] + \
    df['Middle Name'].str[0] + \
    df['Last Name'].str[0]
```

* [Liu and Sarkar et al. 2023]

ColDeco



2	DoB	text concatenation	1st letter of Last N	Abbreviation
3	11/5/1995	CM	F	CMF
4	6/21/1971	BH	B	BHB
5	9/11/1992	DB	M	DBM
6	5/24/1973	OJ	A	OJA
7	2/24/1997	AM	T	AMT
8	3/19/1986	AL	J	ALJ
9	6/3/1968	EMPTY	S	
10	12/9/1966	AJ	S	AJS

∨ Inspect Columns

Abbreviation ('text concatenation' + '1st letter of Last Name')

1st letter of Last Name (the first character from 'Last Name')

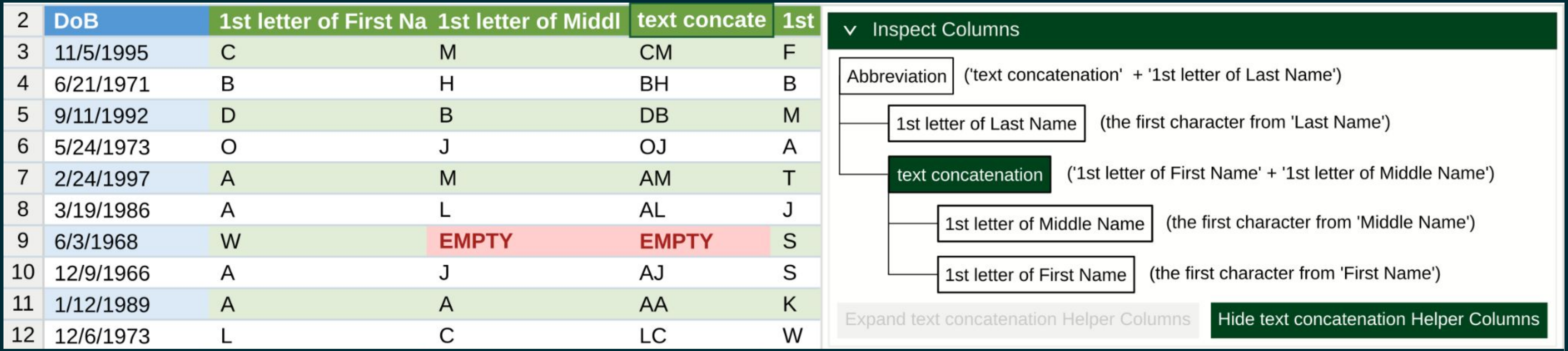
text concatenation (the first character from 'First Name' + the first character from 'Middle Name')

Expand Abbreviation Helper Columns
Hide Abbreviation Helper Columns

Helper columns

“Decomposed” Description

ColDeco



2	DoB	1st letter of First Na	1st letter of Middl	text concate	1st
3	11/5/1995	C	M	CM	F
4	6/21/1971	B	H	BH	B
5	9/11/1992	D	B	DB	M
6	5/24/1973	O	J	OJ	A
7	2/24/1997	A	M	AM	T
8	3/19/1986	A	L	AL	J
9	6/3/1968	W	EMPTY	EMPTY	S
10	12/9/1966	A	J	AJ	S
11	1/12/1989	A	A	AA	K
12	12/6/1973	L	C	LC	W

Abbreviation ('text concatenation' + '1st letter of Last Name')

- 1st letter of Last Name (the first character from 'Last Name')
- text concatenation ('1st letter of First Name' + '1st letter of Middle Name')
  - 1st letter of Middle Name (the first character from 'Middle Name')
  - 1st letter of First Name (the first character from 'First Name')

Expand text concatenation Helper Columns
Hide text concatenation Helper Columns

one summary row per behavior of the code

only referenced columns shown

Overview

LEAP:

Live Exploration of AI-Generated Code

1. The Cost of Validation
2. LEAP demo
3. User Study

ColDeco:

An End User Spreadsheet Inspection Tool for AI-Generated Code

1. End User Programming
2. ColDeco Example
3. Implementation
4. User Study

Implementing Helper Columns

Goal:

> Given a pandas programs of the form `df[<name>] = <expr>`, extract intermediate sub-exprs representing row-wise operations

Solution*:

1. Identify subexpressions that can be written as `Series` representing a column,
2. Assign them to new columns in the `Dataframe`, and
3. Replace the original subexpression with a column reference.

* Basically, A-Normal Form conversion for Dataframe programs.

Implementing Helper Columns

```
df['Abbreviation'] = df['First Name'].str[0] + df['Last Name'].str[0]
                              Series                    Series
```

```
df['$fresh1']      = df['First Name'].str[0]
df['$fresh2']      = df['Last Name'].str[0]
df['Abbreviation'] = df['$fresh1'] + df['$fresh2']
```

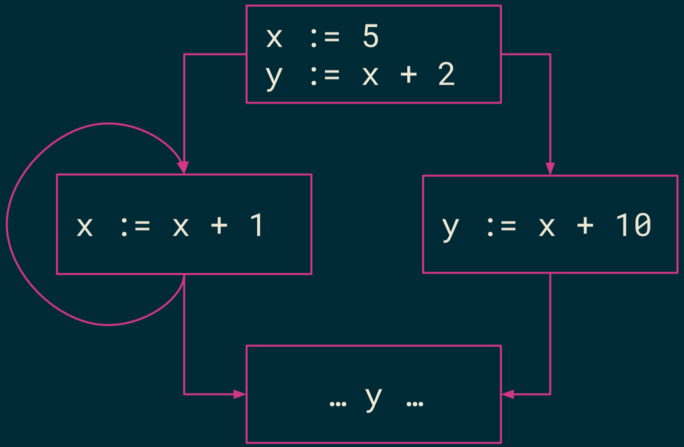
Implementing Helper Columns

```
df["Popular"] = df.apply(lambda x:\
     "Yes" if x["votes"] > 10000 and x["vote_avg"] >= 8 else "No"\
     , axis=1)
```

```
df["$fresh1"] = df.apply(lambda x: x["votes"] > 10000, axis=1)
df["$fresh2"] = df.apply(lambda x: x["vote_avg"] >= 8, axis=1)
df["Popular"] = df.apply(lambda x: "Yes" if x["$fresh1"] and x["$fresh2"] else "No", axis=1)
```

Implementing Summary Rows

Dataflow analysis, program tracing, etc.?



```
x := 5
y := x + 2
```

```
x := x + 1
```

```
y := x + 10
```

```
… y …
```

Table Filtering: Predicates over the values!

vote_avg	votes	votes > 10k	Popular
8	7954	False	No
8.4	18132	True	Yes

↓

vote_avg	votes	votes > 10k	Popular
{positive}	{positive}	{isFalse}	{Enum[No]}
{positive}	{positive}	{isTrue}	{Enum[Yes]}

Implementing Summary Rows

vote_avg	votes	Popular
8	7954	No
8.4	18132	Yes

Implementing Summary Rows

1. Expand *all* helper columns.

vote_avg	votes	vote_avg >= 8	votes > 10000		Popular
8	7954	True	False	…	No
8.4	18132	True	True		Yes

Implementing Summary Rows

1. Expand *all* helper columns.
2. *Tag* the values in each column using a predetermined set of predicates:
   a. `{positive, zero, negative}`
   b. `{isTrue, isFalse}`
   c. `{empty, nonEmpty}`
   d. Enumeration Value (distinct string values)

vote_avg	votes	vote_avg >= 8	votes > 10000		Popular
{positive}	{pos…}	{isTrue}	{isFalse}	…	{enum[No]}
{positive}	{pos…}	{isTrue}	{isTrue}		{enum[Yes]}

Implementing Summary Rows

1. Expand *all* helper columns.
2. *Tag* the values in each column using a predetermined set of predicates:
   a. `{positive, zero, negative}`
   b. `{isTrue, isFalse}`
   c. `{empty, nonEmpty}`
   d. Enumeration Value (distinct string values)
3. Partition the rows based on the vector of tags.

vote_avg	votes	vote_avg >= 8	votes > 10000		Popular
{positive}	{pos…}	{isTrue}	{isFalse}	…	{enum[No]}
{positive}	{pos…}	{isTrue}	{isTrue}		{enum[Yes]}

Overview

LEAP:

Live Exploration of AI-Generated Code

1. The Cost of Validation
2. LEAP demo
3. User Study

ColDeco:

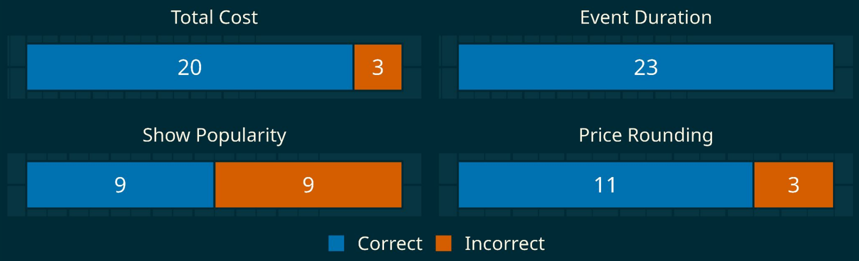
An End User Spreadsheet Inspection Tool for AI-Generated Code

1. End User Programming
2. ColDeco Example
3. Implementation
4. User Study

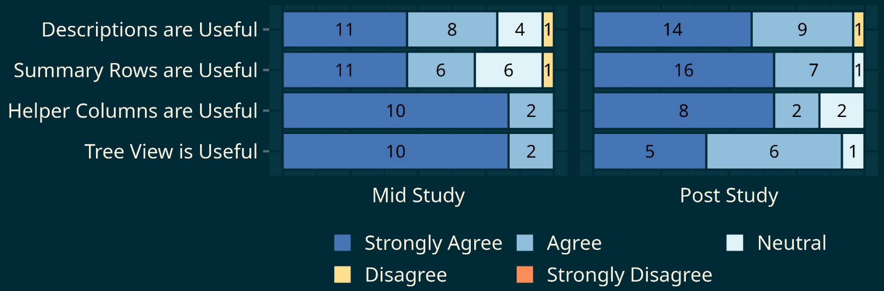
User Study

User study with 24 participants, solving 4 tasks:

Does ColDeco enable code validation by end users?



What are users' impressions of ColDeco's features?



User Study

Helper Columns afford transparency:

“show-your-work button” (P19)

It makes the code "less like a black box" (P23)

Helping them "pinpoint exactly which part of the prompt is not working well" (P15)

ColDeco for Collaboration:

Explain their work to someone else (P11, P15)

Help with *understanding* complex formulas (P6, P19)

Automatically *document* spreadsheets (P6, P15)

In Summary…

Using familiar concepts can enable end users to validate code suggestions.

PL techniques can offer new affordances, even if the user doesn’t see the program!

Overview

LEAP:
Live Exploration of AI-Generated Code

Live Programming
for
Programmers

ColDeco:
An End User Spreadsheet Inspection Tool
for AI-Generated Code

PL Techniques
for
End users

References

N. Perry, M. Srivastava, D. Kumar, and D. Boneh, “Do Users Write More Insecure Code with AI Assistants?,” 2022

S. Barke, M. B. James, N. Polikarpova, “Grounded Copilot: How Programmers Interact with Code-Generating Models,” 2023

J. T. Liang, C. Yang, and B. A. Myers, “Understanding the Usability of AI Programming Assistants.” 2023

H. Mozannar, G. Bansal, A. Fourney, and E. Horvitz, “Reading Between the Lines: Modeling User Behavior and Costs in AI-Assisted Programming,” 2022

P. Vaithilingam, T. Zhang and E. Glassman, “Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models,” 2022

N. Polikarpova, “How Programmers Interact with AI Assistants,” 2023

L. Chen, M. Zaharia, and J. Zou, “How is ChatGPT’s behavior changing over time?,” 2023

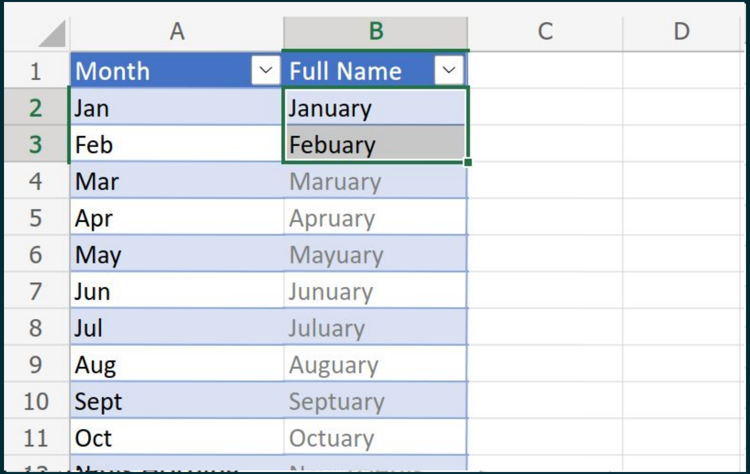
S. Lau, S. S. Ragavan, K. Milne, T. Barik, and A. Sarkar, “TweakIt: Supporting End-User Programmers Who Transmogrify Code,” 2021

M. X. Liu, A. Sarkar, C. Negreanu, B. Zorn, J. Williams, N. Toronto, and A. D. Gordon, “‘What It Wants Me To Say’: Bridging the Abstraction Gap Between End-User Programmers and Code-Generating Large Language Models,” 2023

Bonus Slides

Code Generation in the Wild

Excel FlashFill



	A	B	C	D
1	Month	Full Name		
2	Jan	January		
3	Feb	Febuary		
4	Mar	Maruary		
5	Apr	Apruary		
6	May	Mayuary		
7	Jun	Junuary		
8	Jul	Juluary		
9	Aug	Auguary		
10	Sept	Septuary		
11	Oct	Octuary		

An End User Tool:

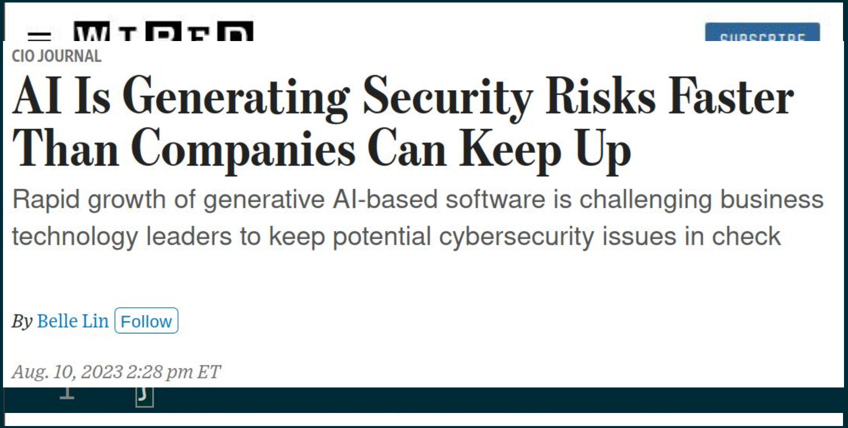
- Input-Output Examples
- Output *program* not shown

"It's a great concept, but it can also lead to lots of bad data. [...] Be very careful. [...]"

John Walkenbach
(Cited in [Mayer 2015])

Github Copilot

GitHub Copilot



CIO JOURNAL

AI Is Generating Security Risks Faster Than Companies Can Keep Up

Rapid growth of generative AI-based software is challenging business technology leaders to keep potential cybersecurity issues in check

By Belle Lin Follow

Aug. 10, 2023 2:28 pm ET

A Developer Tool:

- Code Context + Natural Language
- *Only* output program is shown

Programmers using AI-generated code…

1. Significant time validating code suggestions,

2. Trouble evaluating code correctness, and

3. Under- and over-rely on AI code suggestions.

Grounded Copilot

Acceleration	**vs.**	**Exploration**
unintentional	**Prompting**	intentional with comments / invoke side panel
"pattern matching"	**Validation**	explicit validation via elimination / execution / documentation
unit of focus (sub-expression / statement)	**Scope**	entire function + multiple alternatives
unwilling to edit	**Mismatch Tolerance**	willing to edit / debug / "rip apart" / cherry-pick

Participants

n = 17

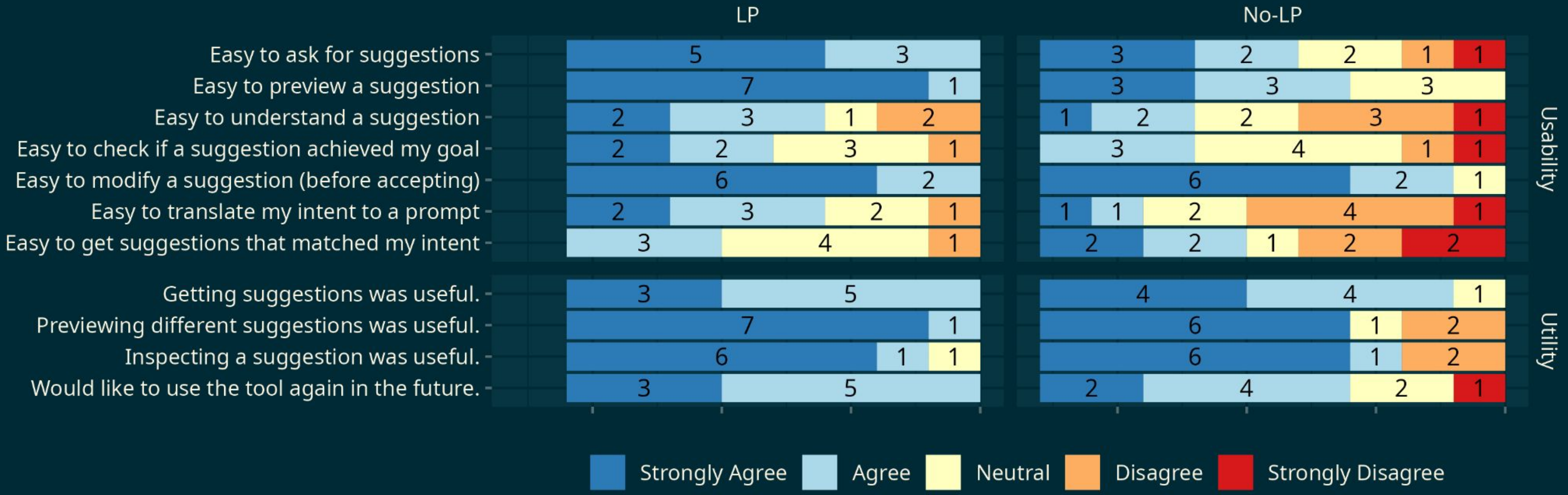
Occupation:

- 15 academia
- 2 industry

Python Usage:

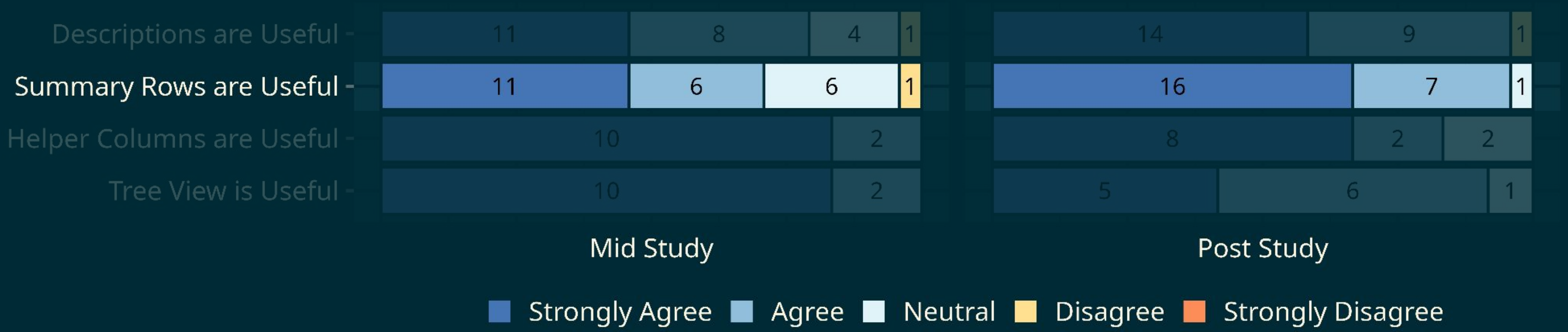
- 2 occasionally
- 8 regularly
- 7 almost every day

RQ4: Users' Impressions



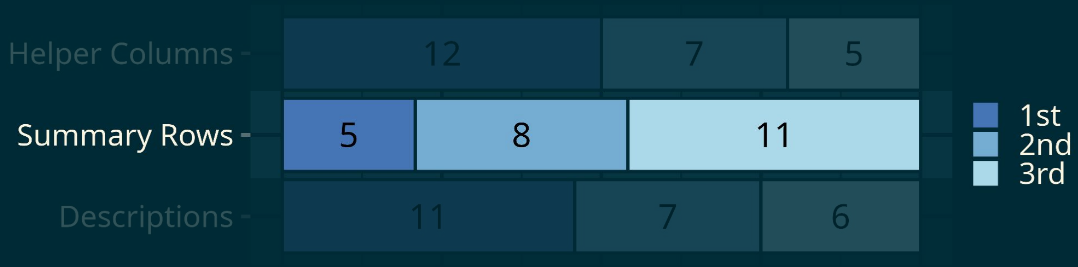
LEAP was more *usable* and more *useful.*

User Impressions



Users liked ColDeco

User Impressions



Usability of Summary Rows

"I **don’t really understand it**, so I wanted to look at the table myself." (P6)

“It **brings the different outcomes and behaviors to the front of the screen** very quickly.” (P16)

"I think I **didn’t understand summary rows** before this [...] Maybe I **got used to it** because it’s my fourth time using this program" (P14)

Summary Rows had a steeper learning curve