SDS PODCAST EPISODE 817: THE POSITRON IDE, TIDY NLP AND MLOPS WITH DR. JULIA SILGE

Show Notes: http://www.superdatascience.com/817

Jon Krohn: 00:00:00 This is episode number 817 with Dr. Julia Silge, Engineering Manager at Posit.

00:00:05 Today's episode is brought to you by Gurobi, the decision intelligence leader and by ODSC, the Open Data Science Conference.

00:00:18 Welcome to the Super Data Science Podcast, the most listened to podcast in the data science industry. Each week we bring you inspiring people and ideas to help you build a successful career in data science. I'm your host, Jon Krohn. Thanks for joining me today. And now let's make the complex, simple.

00:00:49 Welcome back to the Super Data Science Podcast. Prepared to have your brain tickled today by the brilliant and articulate Dr. Julia Silge. Julia is an engineering manager at Posit, which is the company that used to be called RStudio, and they're the makers of RStudio. She also authored the bestselling O'Reilly books, Text Mining with R and Tidy Modeling with R. She previously worked as a data scientist at Stack Overflow and Datassist. Prior to joining industry, she was an academic researcher and professor at Yale University. She holds a PhD in astronomy from the University of Texas at Austin. Julia's books are awesome and so I will personally ship five physical copies of her books to people who comment or reshare the LinkedIn post that I publish about Julia's episode from my personal LinkedIn account today. Simply mention in your comment or reshare which book you'd like.

00:01:38 There's her classic, Text Mining with R, her newest book, Tidy Modeling with R, or her lesser known, but nevertheless beloved, Supervised Machine Learning for Text Analysis in R. I'll hold a draw to select the five book winners next week, so you have until Sunday, September 15th to get involved with this book contest. Today's

episode will probably appeal most to hands-on practitioners like data scientists, software developers, and ML engineers. In today's episode, Julia details the brand new open source IDE Positron that she's been developing, her favorite LLMs for code generation, the open source software libraries that make MLOps easy and her top tips for effective natural language processing, including when more traditional NLP techniques would be more effective than an LLM. All right, you ready for this magnificent episode? Let's go.

00:02:32 Julia, welcome to the Super Data Science Podcast. You are one of those mega stars in the data science space that I've wanted to have on the show for so long.

Julia Silge: 00:02:41 Oh, thank you so much for having me. Thank you. That's so kind.

Jon Krohn: 00:02:45 Of course, yeah. Where are you calling in from today?

Julia Silge: 00:02:47 I live in Salt Lake City, Utah, so that is where I am right now. It's late summer. It's hot getting to be fire season, unfortunately. But yeah, that's where I call home.

Jon Krohn: 00:03:00 Great for outdoor activities, I hear, out there.

Julia Silge: 00:03:02 Yes, just really unparalleled. I mean, in the summer it's like hiking. In the winter, of course, snow sports. It's a really lovely place to be.

Jon Krohn: 00:03:11 Nice. And so you were recommended to me most recently by Hadley Wickham, who is of course also a superstar in this space. So he was recently on episode number 779 and shortly after that, I had the pleasure of meeting him in New York at the New York R Conference, which was really nice. Is that something... Have you ever been to the New York R Conference?

Julia Silge: 00:03:33 I have. I have. I've attended a couple times and I spoke there at least once. It's really a fun group of people and it's a group of people where they have... Or it's a conference where they often have some really big names who come through every time.

Jon Krohn: 00:03:46 Every time.

Julia Silge: 00:03:48 I don't know if our everyone's favorite Bayesian swung through to give us sort of slide free talk while you were there, but that's always kind of a highlight.

Jon Krohn: 00:03:57 Yes, Andrew Gelman?

Julia Silge: 00:03:58 Yeah, yeah, yeah.

Jon Krohn: 00:04:00 This year for the 10th anniversary... I'm going to do some injustice to probably someone out there who's listening who is also a huge name that wasn't there because I'm going to forget them in the slew of all the ones that were there this year. So we had Andrew Gelman there. There was David Robinson. There was Hadley Wickham, of course was there, Wes McKinney, Max Kuhn, Hilary Mason. It was wild. It was kind of like every talk was somebody that is iconic in the field. Yeah, it's an amazing conference to go to, highly concentrated and for whatever reason, it isn't a huge audience like you see Open Data Science Conference or with the O'Reilly conferences that used to happen before the pandemic. So it means that if you do come and attend one of these conferences, you get to meet all of these people.

Julia Silge: 00:04:50 Yeah, yeah. It's a fairly intimate group, a fairly small sort of audience, and so that is a real highlight of it. You really feel like you get to... You're no so far back from the speakers. The dynamic is really interactive, which is fun.

Jon Krohn: 00:05:11 For sure. Lots of pizza, lots of beer.

Julia Silge: 00:05:13 Yeah, that's right. That's right.

Jon Krohn: 00:05:15 So anyway, into the technical stuff to talk about today, the most exciting thing that you're working on right now is that as an Engineering Manager for Posit, which formerly known as RStudio and the makers of RStudio, you're now working there as an Engineering Manager and your project that you are leading the development of is something called Positron, which is described as a next generation IDE, integrated development environment, for data science. So that is what RStudio was many years ago. I mean, I was using RStudio since 2007 that I can... At least since then and was definitely my go-to IDE when I was primarily an R developer back then, an R data scientist, although I guess I wouldn't have used the word data scientist in 2007.

Julia Silge: 00:06:07 Yeah, right, right. Yeah.

Jon Krohn: 00:06:11 So with Positron, what are the gaps or limitations that you're addressing that aren't covered by things like RStudio, VS Code, or Jupyter Notebooks, which might be the go-to IDEs for data scientists or software developers today?

Julia Silge: 00:06:25 Yeah, yeah, yeah. If I was going to sum up the one gap I feel like that Positron is working to address, it's that there isn't something out there right now that can be one place you go to do all your data science. So Positron is not a general purpose IDE. It is specifically an IDE built to do data science. I come from a science background and I've always been someone who wrote code for my data analysis, but I've always really felt that my needs were a little different than someone who is writing general purpose code to build a website or to make a mobile app. People who write code to analyze data are different in some real ways. It's not that it's like they're worst coders or... No, no, I really do think that.

Jon Krohn: 00:07:19 Take that data scientists.

Julia Silge: 00:07:23 It's not. It's not. I don't think it's that people who write code to analyze data do a worse job writing code. It's that their needs are different and that their writing code in a different way. So folks who have been, for example, who have been using VSCode as a data science IDE have really felt that tension where they're like, "This is really general purpose," and instead, and I'm trying to kind of customizing it using extensions to fit my needs. So Positron is meant to specifically be a data science IDE. Positron is also a real driving reason why we built it. The way it is that it is a multilingual or polyglot IDE. A lot of the environments you might download to do scientific computing or data science or data analysis are built specifically for one language. I know all of us have used these. So RStudio is an example of one of these, MATLAB, Spyder. There are a lot of environments in which you would do data analysis that are just built for one language. Increasingly, I just think that's not how as many people work.

00:08:29 Many, many people use multiple languages, whether it's on one project that literally uses multiple languages over the course of a week. They pick up different projects that use different languages. Or almost certainly on the span of years or your career, you use different languages because things change in our ecosystem. Like you said, you started with R and now you use other languages. There are so many people who use combinations of R and Rust or they work on projects that's like Python plus front end kind of technologies, JavaScript, HTML, etc or almost any data science language plus SQL. Very few people, IDE that is built to use one language for very few people is that really going to fit all the needs that they have over the course of a week, a month or multiple years?

00:09:19 So Positron is built with a design such that the front end user phasing features are about the tasks you need to do, whether that is interactively write code, whether that's dealing with your plots, whether that's seeing, exploring your data in a visual way, and then there are backend language packs that provide the engines for those front end features. Positron, it's very early days for Positron. We only made it public about six weeks ago as of the day we're recording this. So it is currently shipping with support for Python and R, but it is designed in such a way that other data science languages can be added because there's a separation between the front end features and what is driving them. So we look forward to adding support for other languages. As you know, we collaborate with other data science communities or new things come up, like new exciting sort of ways of doing data science come up.

Jon Krohn: 00:10:24 So the polyglot IDE part, to me, that makes a huge amount of sense. I get that, especially as a contrast to RStudio. For people who are writing code as a data analyst or a data scientist, people who are working with data, what is different that we need specifically, relative to another software developer?

Julia Silge: 00:10:41 Yeah. So I think one piece that is very different is that the process of writing code is more exploratory, is more interactive. That's not wrong or bad. That is actually just the fact that instead of getting a spec from a product manager and building a product, that's not what data scientists, data analysts do. You start with data and you often don't know what you can or should do in detail until you start that process. And if you have a code writing process that is more exploratory, you need more supports for writing in that interactive exploratory mode. Some things that support that are things like a truly, truly, fully featured interactive console. Of course that does exist in

various ways. People get at that in various ways when they use Notebooks or using say, a Python REPL.

00:11:39 But to get to a truly fully featured interactive console where what happens in the console is then reflected in the rest of where you're working... Like say in Positron, we have what we call a Variables pane. If you come from RStudio, you may be familiar with something called an Environment pane where you see all the things you. And it updates right as you change things. Or the plots that you see, you have them all right there, you can scroll through them, if you change and make a new plot, you see it pop up there and you have that really interactive way of working. Some of the other things that I know really make a difference for people, help inside of the IDE where you are working. You're working, you're like, "Ah, wait. What is the function signature?" Or "Maybe I want to look at the docs for this." So instead of having to get out of a flow state and go somewhere else and read docs on the website, you can open up help right there and copy, paste, go right back and forth and stay in that kind of flow state.

00:12:48 Another thing is you're building interactive apps. You need a way to have that right there and that it updates as you change your code versus having some sort of build process, going somewhere looking at a browser. There's really quite a lot that if we put it together, we can make people more productive. The company that I work for, Posit, the company formerly known as our RStudio, Posit, it's a really fun place to work as someone who likes thinking about the process that people bring to their task because I do think we are huge believers in code first data science. Not no code solutions. Not GUI based tool. People who do data work should be writing code and at the same time their needs are different. Their needs are different, and so we can... Pretty much every single thing my company does is deeply informed by this belief or how

deeply we know that data practitioners are different and that's good and fine, and we can make them more productive by building tools that are specifically for the kinds of tasks they need to do.

Jon Krohn: 00:14:02 In a recent episode of this podcast, the mathematical optimization guru Jerry Yurchisin joined us to detail how you can leverage mathematical optimization to drive commercial decision-making, giving you the confidence to deliver provably optimal decisions. This is where Gurobi Optimization comes into play. Trusted by most of the world's leading enterprises, Gurobi's cutting-edge optimization solver, lightweight APIs, and flexible deployment simplify the data-to-decision journey. And, thankfully, if you're new to mathematical optimization approaches, Gurobi offers a wealth of resources for data scientists, including hands-on training, comprehensive Jupyter-notebook examples, and extensive, free online courses. Check out Episode #813 of this podcast to learn more about mathematical optimization and all of these great resources from Gurobi. That's Episode #813.

00:14:50 It makes a huge amount of sense to me. You also talk about in addition to these kinds of things that are great in an IDE that are interactive, that allow us as data scientists, data analysts doing exploratory data analysis that are super helpful. So the kinds of things that you mentioned like a fully featured interactive console with updated variable pane, so useful and something that for example for me is missing from Jupyter Notebooks, which is what I primarily use today. Obviously having plots in a way that you can scroll through them separately from your code, it sounds like, whereas in my Jupyter Notebook, it's all just kind of one giant stream of information. Then obviously Jupyter Notebooks don't have that kind of built-in help or docs, although some implementations like Google Colab do that.

Julia Silge: 00:15:43 No, there's an interesting sort of set of interact or overlapping tools in this space. I do think Positron is existing in somewhere that's different than some of these slightly other places, like someone maybe who is in a situation where they want a lot, they're needing a lot of guardrails. Often people will start in Jupyter Notebooks, but often as you grow in sophistication, you end up being frustrated by some of those guardrails and constraints and it's like, "Well, I need a more fully featured IDE. I need something. But if I go straight to something like VS Code, it's actually quite challenging to try to get it to be a good fit." I mean, speaking of VS Code. We should say right now, so Positron is now among the list of IDEs that are built on top of the infrastructure that powers VS Code.

00:16:47 There's a repo out there, it's called Code OSS. It's open source components that are used to build what you might be familiar with as the Microsoft branded Visual Studio Code. There's actually quite a number now of IDEs that are forks of this open source of Code - OSS, the open source components that are used to build Visual Studio Code. I think it's actually super interesting because it's like having that as open source has almost made it like a commodity, like the building blocks for making an IDE and it has lowered the bar to people building new IDEs in really interesting ways because you can say, "Okay, what do I want to experiment with as a differentiator when it comes to making an IDE?" One that I bet you and your listeners may have heard of or experimented with is Cursor. Cursor is another fork of this open source code and its sort of take on things is AI first, like LM coding assistant first. Then our approach was data science specific.

00:17:59 We think people who do data science are different from people who write general purpose code, so let's make an IDE just for them. It means that you can build a new IDE

with a smaller team, I think, than you would have in other ways because you can concentrate on the things you think are differentiators and then the pieces that are true for anyone who's writing code. We all need to save files. We all need to look at git merge conflicts. You get that by building on top of this infrastructure. The other sort of really interesting piece for building on top of this infrastructure is that you can then make the whole wide, wide world of VS Code compatible extensions. You can make those available to your users. So for example, what this means in our case is people who need a data science specific IDE now have access to that whole world of extension.

00:19:00 So if you're like, "Well, I do data science but I want to use the Databricks extension," or whatever, you can end up getting that extensibility, which has not been true of many of the other kinds. If you look at the category of RStudio, Spyder, those are quite difficult to extend and customize. So it's an interesting choice and I think it's pretty interesting seeing how we're having this flourishing of IDEs because of the open source nature of the underlying infrastructure for VS Code.

Jon Krohn: 00:19:36 Very nice. And yes, so not only do you have Code - OSS as a kind of a backbone that's providing building blocks for Positron and offering that kind of extensibility through all the VS Code extensions, like you mentioned Databricks there, any number of extensions that people might want to be able to import.

Julia Silge: 00:19:52 Rainbow Tabs. Whatever you want, it's out there.

Jon Krohn: 00:19:56 In addition to all those things, the Positron project itself is open source, so if people are listening and they want to be contributing right now at the time of recording, there are 27 people, including I can see your face as one of those

GitHub contributors, listeners can go and they can contribute to this developing and very exciting project.

Julia Silge: 00:20:16 Yeah, so Positron is licensed such that it is source available. Anyone can come and look at the source, change it, contribute to it, and it is also licensed such that it is free to use, including for commercial purposes. You can use it, of course, in academia, for personal projects, but you can use it at work. It is licensed in such a way that it is free to use in your work as a data scientist, data analyst. So it's free to get it so you can read the code. There's real benefit to that kind of model for building and making software.

Jon Krohn: 00:20:55 I'm a big Posit fan. They have sponsored the show in the past and they may sponsor the show again in the future, so there's that obvious kind of bias, but I-

Julia Silge: 00:21:03 Sure, sure.

Jon Krohn: 00:21:04 I feel confident that I would be saying these great things about Posit regardless, because it's so awesome to have companies like Posit out there that are supporting people like you to be able to be working a huge portion of your time, I imagine, on open source projects like this, that then anyone can go and contribute to, anyone can go and use. There's already thousands of stars on this GitHub repo despite only having been released six weeks ago at the time of recording, which is wild. So yeah, thanks Posit for that kind of system and Posit, it is a public benefit corporation.

Julia Silge: 00:21:39 Yeah. Yeah, it's really interesting. So some people hear that and they're like, what does that mean? It does not mean a nonprofit. So we're not a nonprofit like Wikipedia. Some of the really famous PBCs are outdoor companies. Basically, what a PBC means is that what you write into your governance, how is your company governed, what

are the decisions, that you make explicit, here are the things we take into account when we make decisions. It turns out regular corporate governance, it's actually sort of against the rules to use any considerations other than stakeholder value. That's actually how they're set up. That's the only thing. So PBC organized companies is that you make explicit, here are the things we take into consideration. And of course, one of them is stakeholder value, making sure the company is financially stable and sustainable. So our PBC charter talks about the data science community in general, not just our customers who give us money, but the data science community in general.

00:22:50 That means we can explicitly say, "We make a decision to improve or to make the situation better for data science users in general." It's like the analogy to think of is not nonprofit. But rather like many famous outdoor companies who make outdoor products back to my living in the west where very outdoorsy kind of city, many of them are set up as PBC, or some of them are set up as PBCs so that they can... Then often they'll say, "The environment," or 'the outdoors," that they can use that to make decisions for their company. It's pretty great as a employee. No company is perfect. No place will they always do everything that you agree with, but working somewhere where leadership has made their values explicit and then set up so that they can make decisions aligned with their values is pretty great.

Jon Krohn: 00:23:42 Yeah, it is for sure. So yeah, really cool. Thanks Posit for doing that. To go into more of, I guess, the history of Posit and things developed in the past, obviously we've already talked about how Posit used to be called RStudio and that's because they were developers of that hugely... It was like the de facto IDE of choice for basically everyone when you're using R. It just worked so nicely. So with RStudio and Positron both being developed by Posit now,

how do you see these two tools coexisting and complementing each other?

Julia Silge: 00:24:15 Yeah, so I think the best thing to think about this is to just really acknowledge how new Positron is. Positron is a teeny tiny baby, and it is actually only available... As of when we're recording this, it's only available like getting an installer from GitHub actions. I would not say it is ready for prime time. Positron is today, for example, not available to our paying customers, like our people who pay for our products like Workbench. It's not even in there yet. It's in what we're calling a public beta. It's in beta here. In RStudio, it is like rock solid, stable software. It's got a decade plus of real world use on it. We expect, particularly, for an R user, RStudio is probably going to be the best choice for many people's work today and for quite a while to come. If you are happy with your current code editing experience, there's no call from me or anyone on the Positron to switch today. I think right now, if I were to say who should try it out today, it's someone who's a little adventurous, maybe they're an early adopter type and probably someone who has some motivating reason, like they are someone who writes R and C++ together. That would be an example, but like you're using multiple languages together or you do a lot of maybe Python plus Rust or you do a lot of data science plus front-end work and you are using those multiple languages and you're frustrated by the things that are currently open to you.

00:26:09 So my company is committed to maintenance, and including new features where appropriate for RStudio, for the long term. We expect it to be used for people's real work for quite a number of years to come. We made the call to build on top of a new architecture because of some of the limitations. You could have maybe envisioned what if you made RStudio, but for Python, and it was really RStudio, but built for Python. That would've been another

sort of way that could have moved forward. There were a couple of reasons why that was not the call. One of them is that the architecture of RStudio is deeply, deeply entwined with R. So much so that the R session is deeply ... and it would've taken a lot of work to unravel that. You can't just throw Python in there. If you have ever been an Rstudio user, you may have had the experience of the crash with the little bomb and the whole thing just goes down. Right. Are you familiar? Do you know what I'm talking about, Jon?

Jon Krohn: 00:27:19 I actually haven't seen that.

Julia Silge: 00:27:20 Okay. All right.

Jon Krohn: 00:27:21 I guess I'm grateful.

Julia Silge: 00:27:24 If you crash R because of too much data or RAM or whatever, if you crash R, the whole thing goes down. The whole thing goes down. With the new architecture that we're building on, which in many ways is more modern and more separated, if R crashes, the IDE stays up. You lose, of course, things you have made in your interactive R session, but the rest of everything stays. You don't lose everything else. So A, RStudio's not going anywhere for a very long time and we really did think carefully about if we're going to make a next generation thing or support people who are using more than one language, what should we do next? There were a lot of things on the table an I think that what we landed on ... of course it has it's upsides and downsides, but I think it's a really good choice where we landed. It's playing out well.

Jon Krohn: 00:28:22 It sounds like that ability to not have the whole system crash maybe is part of what makes this a next generation IDE.

Julia Silge: 00:28:29 Yeah, for sure. I would definitely agree with that.

Jon Krohn: 00:28:31 So what's next? You already support R and Python. Do you have some sense already, can you disclose to us on air maybe what programming languages you're thinking of supporting next?

Julia Silge: 00:28:41 I am willing to say this, not because the plans are immediate, but because I think it's really clear what makes sense next. And that is probably Julia, the language that shares my name, but I do not use. If you look at our GitHub repo and you search to find the most upvoted issue that has come up, at least as of today, when I looked at it, it was like, "Hey, can you support Julia?" I'm just predicting that will be the next thing that happens, not because we have started work on it or I'm committing us to it, but because it seems that is what the community would most like next. There are already Julia Jupyter kernels. Positron is built using a lot of standard protocols, so that work can be modular, where it can be reusable. If there is an already existing Jupyter kernel, which there is of course for Julia, then the amount of work to get it to work into Positron is not enormous. It's not enormous. To be clear, just to emphasize, this is not me saying we have started on this.

Jon Krohn: 00:30:01 Yeah, yeah, yeah.

Julia Silge: 00:30:01 But that I'm predicting that'll be the next thing that happens because that's where the most community interest is.

Jon Krohn: 00:30:08 I've had the pleasure of speaking at ODSC West many times over the years, and without question, ODSC West is one of my favorite conferences. Always held in San Francisco, this year it'll be taking place from October 29th to 31st. As the leading technical AI conference, ODSC West brings together hundreds of world-class AI experts, speakers, and instructors. This year's offering will feature hands-on sessions with cutting edge

techniques from machine learning, AI agents, AI engineering, LLM training and fine tuning, RAG and more. Whatever your skill level beginner or experienced practitioner, you'll leave ODSC West with new in-demand AI skills. Register now at odsc.com/california and use our special code podcast for an additional 10% off on your pass.

00:30:55 Yeah, it makes sense that there is a lot of community interest and also overlap in terms of existing components that you can make use of because this is probably obvious to some portion of listeners, but maybe for some for others it isn't. But Jupyter Notebooks that is, I don't know, I don't think portmanteau is the right English word, but it's a blend of Julia, Python and R, is what makes up that word Jupyter.

Julia Silge: 00:31:21 Yeah. Yeah, yeah. So people who use a Jupyter Notebook may not be super aware of what's driving this. So every time you have a Jupyter Notebook, there's a Jupyter kernel and the Jupyter kernel uses something called the Jupyter protocol. And so the Jupyter protocol is a way that a backend and a frontend can communicate. So it's like the frontend, think of your notebook, is going to ask a question, and then the backend gives an answer and it goes back and forth with a standardized set of questions and answers. That's called the Jupyter protocol. You can do that in a notebook, but you can also use that protocol in other ways and that's what Positron does. It uses that protocol as a way to drive those components that we just talked about, like the variables pane, the plots, the viewer. You can use Jupyter to, and by that I mean the Jupyter kernel and the Jupyter protocol to build a fully interactive console.

Jon Krohn: 00:32:23 All right, so this might be a really dumb question for me, but we talked about how Positron is also built on Code - OSS. So does that mean that there is already an in-built

connection between Code - OSS and the Jupyter protocol, or is that something that you stitched together for Positron?

Julia Silge: 00:32:39 That is something we stitched together. Sometimes when people hear about what we're doing with Positron, they say, "Well, why didn't you just build a set of extensions? Why didn't you just provide extensions so that someone who can come to VS Code and can use it as a data science user in this really fluent way?" VS Code is amazing. I have spent quite a lot of time in its innards over the past year. Incredible piece of engineering, it is built with limited extension points. So what a pure extension can do is quite limited. Our hypothesis, our thing that we believe is that, after having tried it out, is that if to get what a data science user truly needs, what they really need to be effective as that work, is not possible only with extensions. It's mainly because of the way things are connected to each other, the way that you want your console, your plots, your variables, your viewer, maybe the pane you use to connect to databases, you want all of that in sync and talking to each other in a way that's not possible only through extensions. If you ask one of these other forks a question, "Why did you fork? Instead of building an extension, why did you fork?" They'll have a similar answer like, "Oh, well our hypothesis is, that what our user wants is X and that's not possible through the existing extension points."

Jon Krohn: 00:34:08 Very cool. That makes a ton of sense to me. Going back just a tiny little bit, when I was asking you what language might be supported next or what's next for Positron, I am not surprised that the answer was Julia. But I actually expected, because this was before we'd done any of this discussion of Jupyter protocol, and so I expected, my guess was that it was going to be SQL.

Julia Silge: 00:34:32 Oh. So that's also really an interesting idea. We would be interested in hearing folks feedback on that kind of thing. So right now, the way that SQL is handled is that we have something called a connections pane that allows you to manage your SQL connection, see your tables, see them in a really nice visual way, but then you would write Python or you would write R. In Python, it supports SQLite and the SQLAlchemy, these ways of getting from SQL in and out of Python. But I think a pure Python, I'm sorry, a pure SQL experience would also be really interesting and is something that we have had internal discussions about. The thing here is that the execution model is really different. When you are executing SQL queries, you're not executing them in some kind of kernel, like a Python kernel, an R kernel or this possible Julia kernel, you're actually executing it back in the database. But I think there is an interesting idea of a data science IDE that is built to use SQL.

00:35:47 This is maybe a little bit further out, but the idea of SQL, then you're executing in the database and then you bring it into something that provides support for a reporting and dashboards, which for us would be Cordo. Because Cordo does have support for pure JavaScript visualization through Observable. And so you could execute SQL against a backend and then bring it into something, like if you're going to write a report, make a dashboard, make a set of slides, you could end up with a data science experience that actually does not go through a kernel, but instead executes queries against the actual SQL, brings it over. And then actually you could do your reporting and your dashboard needs or whatever, actually just directly in JavaScript with using Observable. So very interesting stuff that could be on the horizon for really different kinds-

Jon Krohn: 00:36:40 Exciting.

Julia Silge: 00:36:41 Yeah. Yeah, really exciting.

Jon Krohn: 00:36:43 Looking forward to seeing where this project goes. Obviously, we will have a link to the GitHub repo for Positron in the show notes so people can check that out. Moving on to another big passion project of yours as well as Posit's is MLOps tooling. So you have defined MLOps as a set of practices to deploy and maintain ML models in production reliably and efficiently. How has the tooling that you have developed, so tools like Tidy models and Vetiver ensured MLOps practices?

Julia Silge: 00:37:17 I love talking about MLOps because I think it's a phrase that has a lot of hype around it still, and people aren't sure what it means. People aren't sure and that's partly why when we started building tools, we wanted to be really explicit about what do we mean when we say MLOps. There's the process of developing a model, which you might use scikit-learn or tidymodels or PyTorch to develop a model. Let's call that model development. And then at some point you are done, you have your model, and then what do you do with it next? What are you even supposed to do with it next? I think that first piece, you can learn about that in a class. People will teach it to you if you go get a master's of data science, you can read books about it. But when you're done, what do you do? What do do with it? Where are you supposed to put the model? What does production mean?

00:38:18 People are like, "I'm afraid to ask, but what actually is production? What does that mean?" I think one of the reasons I love working on tools for MLOps is that I really like thinking about people's really practical workflows. What do they actually do? What are they actually trying to do and can we think about that from a systems thinking kind of way, where how is it fitting together? For the team that I built working on MLOps, what we wanted to focus on is your model is trained. You have used the

appropriate statistical methods to decide what your model is. It's done, where do you go from there? Where do you go from there? We ended up highlighting three tasks. The first thing you need to do is you need to version that model. So much like you need to version code, you need to version a model so that you can keep track of change. If you were to retrain that model next month or next year, how do you know what the differences are, how do you keep track of the metadata around it? So that's that first piece version.

00:39:28 The second piece is deploy. That's a word that again, people are like, "What does that even mean? I don't know." Deploy means, a simple way to think about it is get the model off of your laptop, instead of being somewhere in the development environment where you built the model, instead you have lifted the model up with all the computational needs it has and you put it somewhere else. Often the best way to do that for the middle 80% of users is using REST APIs in some kind of containerized environment, whether that's literally Docker or something else. That's deploy. So you version your model, you're doing the prep work so that you can keep track of what are the characteristics of the model. Deploy the model, that's the process of wrapping it up and getting it where it needs to go away from your dev environment into a place where it can be integrated into the general infrastructure of your organization.

00:40:30 And then the third big piece is monitoring. So once you have put that model somewhere else, how do you watch it over the longterm? Because machine learning models are pretty interesting in that they are both statistical artifacts and they are software artifacts. So think of like a mobile app or something, it's mainly just a software artifact. If you were to monitor how it's doing, you would monitor what's its uptime, how fast is it responding? That's the software artifact of it.

Jon Krohn: 00:41:10 You mentioning that just caused me to realize a question that I should have asked you about Positron as well.

Julia Silge: 00:41:18 Yes. Okay.

Jon Krohn: 00:41:18 Which is, what kinds of discussions have you had around ... so right now, Positron is that it's just that software development artifact.

Julia Silge: 00:41:26 Yes.

Jon Krohn: 00:41:26 But I imagine just in the way that GitHub has Cpilot and Google has Gemini built into Colab, that must be something that you discuss as well where there would be a machine learning artifact built right in.

Julia Silge: 00:41:39 So as of today, in its early stage, Positron doesn't come built in with LLM-based code assistant type tools. Instead, we have built really, so that you get a great integration experience with LLM-based coding assistant tools that are available. People probably immediately will be like, "What about Copilot?" It is true that as of today when we're recording this, Copilot is licensed in such a way that it can only be used in more proprietary Microsoft tools. If we're literally talking about Copilot that is usable in the Microsoft build of VS Code, but as of this recording is not available in Positron. What you do have access to is to this burgeoning ecosystem of alternative LLMs. There are quite a number of ways of interacting with LLMs that are aligned with different people's preferences. Three that I know people have had quite good experiences, one of them is called Continue. One of them is called Tabnine, and one of them is called Codeium, with an E. They're different takes on what it means to have an LLM-based tool help you write code. A cool thing about these three is that you actually have different choices about the backend that you use there. You can use one of the OpenAI backends, but you can use alternatives as well.

You can in fact, use a fully open-source model if that is what-

Jon Krohn: 00:43:16 Code Llama.

Julia Silge: 00:43:18 Yeah, yeah. Right. In fact, you can use a local model such that if you have constraints maybe around compliance, that you can't send your data or code far away. So if the question is literally, " Does Copilot work?" Unfortunately, as of this moment, the answer is no. But if you're like, "What is our vision for LLM-based code assistance?" Our vision for that is excellent integrations. Excellent integrations and choice, so that people can decide what aligns with their own either personal or organizational priorities when it comes to the details of the LLM itself and then the details of how you're interacting with the LLM.

Jon Krohn: 00:44:01 Did you know that the number one thing hiring managers look at are the projects you've completed? That's why building a strong portfolio in machine learning and AI is crucial to your success. At Super Data Science, you'll learn how to start your portfolio on platforms like Hugging Face and GitHub, filling it with diverse projects. In expert-led live labs, you'll complete an exciting new project every week. Plus, through community-driven projects, you'll tackle real-world, multi-week assignments while working in a team. Get hands-on experience with projects like retail demand forecasting, building an AI model from scratch, deploying your own LLM in the cloud and many more. Start your 14 day free trial today and build your portfolio with superdatascience.com.

00:44:42 That's a really exciting answer because that's probably what developers love the most is to-

Julia Silge: 00:44:47 Yeah.

Jon Krohn: 00:44:47 ... hear that they can have whatever their preferred choice is. It means that behind the scenes they can be using the LLM of their choice, it could be an open AI API, or it could be Anthropic, or it could be a completely open-source implementation like Code Llama, something that you could have running locally or that you could be using through a third-party provider.

Julia Silge: 00:45:09 Yeah.

Jon Krohn: 00:45:11 So you mentioned three integrations-

Julia Silge: 00:45:14 So the first one is Tabnine, so Continue ... not to pick a favorite, but Continue. People bring different sets of priorities here. So I'm not saying the one that I like the best is the best one. So Continue, Codeium with an E in it, C-O-D-E-I-U-M and then Tabnine. So these are ones that I know people have had good experiences with in Positron already.

Jon Krohn: 00:45:42 Awesome. That is really helpful. I love hearing that. I'm eating into our MLOps discussion time, but we can now go back to where-

Julia Silge: 00:45:53 No, no it is really relevant. Yeah, it's really relevant, because that third piece for what does LOps mean is monitoring. When you're monitoring a machine learning model that's in production, it means different things depending on who you're talking to because it is a software artifact, meaning that you do have to measure things like uptime latency for a machine learning model. You do need to measure that because they are software artifacts, but they are also statistical artifacts. You put a model into production and you could monitor things about its software characteristics like metrics, like latency and uptime. And it could be doing great according to those metrics, but over time, say something in the world changed, such that the actual underlying relationship

between your inputs and your outputs change, over time, your statistical metrics could just be falling off a cliff. If you don't monitor your models, you don't know that that's happening because machine learning models and production are particularly prone to silent failure. Because by failure we don't only mean the software characteristics of them, we also mean the statistical character.

00:47:10 It can be giving you predictions that are nonsense because the world has changed, and the model that you originally changed a month or a year ago, no longer applies. So those are the three pieces when it comes to MLOps that we focus on providing tooling for. We assume that you start with a model that's trained. Again, we think that people bring a variety of sets of perspectives to how they get there and so we are fairly agnostic to how you got there. This is one of the real differentiators between the projects I've worked on, which it's called Vetiver versus a project like MLflow. We come in at a different place and provide data science practitioners more flexibility in how they get started. We have version deploy monitor, version deploy monitor, so that's what Vetiver provides support for. Vetiver is a framework for MLOps in both Python and R. There's very parallel implementations there. What that means is that if as a practitioner, if you prefer to use tidymodels in R for one kind of statistical problem, but you like to use PyTorch in Python for another kind of statistical problem, you can deploy them both with the same tool.

Jon Krohn: 00:48:35 Nice.

Julia Silge: 00:48:35 You can provide your, say your software engineer collaborator with the same API that looks the same no matter how you trained it. The other differentiator for Vetiver is it is built ... this aligns so much with the other things we've said ... but it is built for a data science

practitioner to use. It is not built with a general software engineer user in mind, it is built with a data science user in mind. Different orgs make different decisions about this, like who is responsible for getting a model the last mile, like getting it deployed. At really large organizations, there's whole teams that that is their whole job.

00:49:21 But for many medium size organizations or even small ones, it's like, who should do this? Who should do this? My hypothesis is that the best person to do it is the person who has the most domain knowledge about the model. If we can give that person the tool so that they can hand it off a little later in the process, not hand off some really raw thing, but actually be the one that packages up, deploys the model, that we end up with better machine learning practice overall because they have the most knowledge about the model and how it works.

Jon Krohn: 00:49:54 Yeah. Thank you for that tour of Vetiver. It sounds like the compatibility, having support for both Python and R in one place, allowing me ... there are still some things that I love to do in R. I primarily use Python these days, but there's things like particularly for me around creating visualizations that are with a ggplot library.

Julia Silge: 00:50:13 For sure. For sure.

Jon Krohn: 00:50:14 So, much better than anything.

Julia Silge: 00:50:14 Unmatched. Unmatched.

Jon Krohn: 00:50:17 Unmatched. And so, yeah, there's reasons for me to be using both together. And so, it's great that with Vetiver I can be deploying both languages together and monitoring across all those three key steps that you outlined there, version, deploy, and monitor. So, moving on to our next topic area. It still is tidy. It's a tidy topic. So, with Vetiver you're talking about tidy models in there. This one is all

about tidytext. So, you've written several books, several bestselling books in fact, and one of them, Text Mining with R: A Tidy Approach, features the tidytext natural language processing library. And interestingly enough, it also includes Jane Austen's Complete Works with an R package that you wrote, which is Jane Austen R.

Julia Silge: 00:51:04 Yes, that's right. That's right. Some of your listeners are probably familiar with the stickers, like the hex stickers that the R community just loves and loves to put on. And I made a hex sticker for the Jane Austen R Library, and it's her signature with colors or whatever. And I just said, I love it. I love it.

Jon Krohn: 00:51:24 And this is a complete tangent from where I was going, but are you a big lover of Jane Austen books?

Julia Silge: 00:51:29 I'm a super fan. I'll be honest, no. So, the story of tidytext is very intertwined with my story of just getting into data science in general. When I was making this career transition from the random stuff I was doing before into data science, I was thinking about, okay, I have this weird resume. What can I do? How can I set myself up so that people who are in... this is about 10 years ago, who people who are in this at the time, new-ish field of data science, how can I be compelling that yes, I can do this. I'm someone who can do one of these jobs? And so, I was working on what I thought of as the time as a blog, a way to show people the things I can work on. I envision myself sitting down with a hiring manager and talking through these projects like a portfolio. And as I was thinking about what will be compelling to people, and I thought about, okay, the stuff that I really know about, stuff that I really know about and care about that's personal to me. And so, my blog, all the posts are still left. If you go to the earliest, earliest posts, some of those are using data from Utah, like Salt Lake City and Utah data because I was like, okay, I go to the public data portal, I pull something

that's about county differences in health. So, I started out there. And then I very quickly started thinking about, well, I mean, anyone who knows me knows I love Jane Austen. This is one of the great loves of my life, my whole life since I was 12 years old or something. I should see what kind of analysis can I do out there to do with Jane Austen's work?

00:53:23 Jane Austen's works are in the public domain, which means you can just get the text of them. And I started doing some initial exploration. I was having a great time, and then I was introduced to Via, at the time, the thriving data science social media scene. And I've always introduced to David Robinson, who you mentioned earlier. He lives in New York, a big part of the NY R community. And David Robinson, I'll be honest, changed my life. Dave reached out about collaborating because he was excited about some of the stuff he saw me doing was, "I think there's an opportunity here to build tooling that is tidy versus style tooling, but applying to text." I was quite new to the R community at the time, and I didn't have as much background as he did in terms of what does that mean, tidy verse style tooling? But he had. He had built things like Broom and had already had that experience.

00:54:31 And so, we worked together, we collaborated together, and we actually met for the first time at what was called an uconference run by the organization rOpenSci, which is an amazing organization about supporting open science through R. There's pyOpenSci is a similar organization for Python. Anyway, we met at an R conference in person. And he was like, "Hey, do you want to build an R package to do text analysis from a tidy verse perspective?" And I was like, "Okay, sounds great. Let's do it." And we had something working by the end of three days. That was the core of what tidytext became. And then over both Dave and I at the time were really loving writing publicly and putting a lot of stuff out there to help people know how to

use our tools. So, we were doing really interesting, or at least we thought things we were really interested in. We were writing a lot.

00:55:32 I did a lot really digging deeper into the Jane Austen stuff, comparing to other books. This was like 2015. Dave did this analysis of Trump's tweets at the time that went super viral, that used our tooling. So, we were having all these blog posts. And we, at one point, looked at each other, were like, "What if we wrote a book? What if we wrote a book?" And we started basically with taking some of these stuff we had already written either long form documentation, package vignettes, blog posts when we started putting them together. And then how will we shift? How will we reorganize? How will we make this flow from one thing to the other? And how do we write an introduction? How do we wrap this thing up? And we wrote that book really fast. I have since worked on other books, and the book that Dave and I wrote together came together fast and because it just was right. It was just right. And it was huge for me. It was huge for me. It was huge for my career. I love that book. I love Jane Austen. Yeah, no, that's a big part of my story is how that all came together.

Jon Krohn: 00:56:48 I said, we will be doing a book giveaway. You don't know this yet, Julia, but when we have authors on the show, we often do book giveaways. And so, we will be doing one for your books.

Julia Silge: 00:56:58 Exciting.

Jon Krohn: 00:56:59 Yeah. Yeah. Yeah. When this episode goes live, people who have been listening to the audio version, it will be in my intro because you also wouldn't know this, Julia, but after we finish recording, I use my notes from our conversation to create an intro and an outro. And in that

intro, I will have announced that people can get a physical copy of your book by... yeah.

Julia Silge: 00:57:22 Oh, delightful.

Jon Krohn: 00:57:25 All the details are in the intro as to how they can pull that off.

Julia Silge: 00:57:28 Amazing. That's delightful.

Jon Krohn: 00:57:32 Yeah. So, yeah, that was a great story to get behind the development of your first book. And now, I mean, there are lots of recent projects as well that are super interesting. You've done topic modeling for Taylor Swift lyrics. You've done measuring readability with a smog metric, which is a fun I like at juliasilge.com/blog/gobbledygook.

Julia Silge: 00:58:00 Yeah, that was really fun to work on. And I will say the Taylor Swift one was super fun to work on. I did it right the same week that the concert film came out. I have not seen Taylor Swift live, sad to report, but I went the week that that concert film came out. And it was like the first week, and I went with my kids to see it, and then I did the topic modeling and it felt very topical at the time. It was very fun. I was very interesting. And actually the results are pretty interesting.

00:58:29 So, the topic modeling looks at the textual content of lyrics. And the real takeaway from there is that there's the pandemic era albums, like Evermore. The two of them are very similar lyrically, the machine learning algorithm puts them together. The early albums all get put together because they are also very thematically very similar. And then Reputation is one that really stands out as separate. Reputation is quite distinct lyrically from these other groups. So, that was really fun to work on and certainly aligns with my own sort of how I perceive as a fan and

someone who enjoys Taylor Swift's work. It really was an interesting way to explore something else that I love. I love doing data science projects about things that I love, and I just get a lot of joy from that.

Jon Krohn: 00:59:28 Nice. Really cool to hear about that project as well. Another one that you did recently is with Stranger Things dialogue. And so, for the popular Netflix series, Stranger Things, you showcased High FREX and lift words of each season's dialogue. What do those terms mean and what did you find?

Julia Silge: 00:59:49 Yeah. Okay. So, topic modeling is a Unsupervised Machine Learning method for analyzing text. Text data is very interesting. I mean, I guess no shock that I find that very interesting. But when you think about text and natural language, there are a couple of really defining things about it. You see a lot of power laws where there are a few words, we use a ton, the and, then there are a lot of words that are used only a few times. This means you count the things you're observing in a... there's a very wide discrepancy in how many times you have counted things or observed things. And so, you have to use methods that allow you to get at that, allow you to learn something even giving that. They're sort of brute force methods where you just take out stop words, you make some cuts. But then there's much more sophisticated ways you can use to learn what words are important, what topics are about what.

01:00:48 So, a topic model is a multi-level hierarchical model. And the mental thing you can think about is that a topic is made out of a mixture of words. And words can be in more than one topic. And then a document is made of a mixture of topics. So, with the Taylor Swift example, I think I did it so that songs or maybe lines, well, one or the other, songs or lines with the document and then you say, okay, so which songs are made of which topics? And

the topics are made of different words. So, you end up with the most probable words, but those are often the same across a lot of topics. So, in Stranger Things, I think I did it as lines, like lines of dialogue.

01:01:37 And if you think about, have a group of people talking, the most common words are always just common words that people use speaking. It's different than the words if you were reading prose. So, it's not so much the and and, but more like you and words you use as you talk. Our spoken word is different than we do written language. So, dialogue. So, those are the most probable words. So, if you look at, oh, the topics all have the same, most probable words, but that's normal. That's normal and okay.

01:02:08 And so, these other metrics like lift and FREX help you get at metrics of what words are special or unique for different topics. So, they're like different statistics. Like FREX is high frequency and exclusivity. So, it means words that are used a lot, i.e for high frequency, but also have high exclusivity, meaning you see them in some topics but not others. High lift is words that you see a lot relative to how often they're used. So, if you had a word that was used a lot, but high lift for that word would have to be really high, really high compared to something that was used only a few times.

01:02:50 So, one thing I remember from that Stranger Things is that some of these high... if you're looking across the seasons, what are these high FREX, high lift words? You see the words about that monster that they called Dart, the little funny monster that got lost in the house, that one pops up in that season because high lift, high frequency, high exclusivity. When there were things that only happened in the first season, like they talked a lot more about the upside down or something. And those popped up in the early seasons because it was high

exclusivity to those. So, topic models are great. They are complicated models. And it's interesting thinking about when are they useful in the era of LLM based tools. And I think they're most useful when you have medium-sized text data. By that I mean like 5,000, 10,000. You have text data, that many documents.

01:03:48 So, think of a document often in a real application as a survey response or something along that line. So, you have on the order of 5,000, 10,000, and you are interested in looking for what are the topics, what are these about? You can, of course, look to LLM based tools for summarization, which can give you another sort of way of doing it. But I tend to think they're most useful in situations either where you have compliance reasons that you can't use LLM based tools, or that you have more medium-sized type data, or you have needs for higher statistical rigor than I threw it into an LLM and got something out. So, I think even in the era of LLM based text tools, I think it never gets away the need for doing EDA for text. And that's exactly what the tidytext package is all about, is about doing EDA for text. I would say it's a bad idea to just throw text or analyzing into an LLM based tool for summarization without also doing EDA first. And it's also interesting to think about when would you use which kinds of tools and what are your needs? These are all tools. And adding more tools is great, but we have to know when it's appropriate to apply them.

Jon Krohn: 01:05:11 Those are really great points you made there at the end around why you would use an LLM versus topic modeling. These kinds of more now what you might say are traditional natural language processing techniques. And yeah, some great points there. If you want higher statistical rigor, if you want to be doing exploratory data analysis, which maybe you should be doing before you are using an LLM.

Julia Silge: 01:05:28 Yep. Yep. Yep. Now, I would argue yes.

Jon Krohn: 01:05:30 Mid-size data. It's also probably going to be a lot less expensive.

Julia Silge: 01:05:33 Oh, absolutely. Absolutely. 100%. Because it is true. LLM-based tools are expensive to use, either literally per API call or in the expertise of running a local one. Yeah, nope, that's absolutely true. Yep.

Jon Krohn: 01:05:51 So, one of the questions that was brought up in our research by our researcher, Serg Masis, that I was really interested in asking you around NLP, was that you have pointed out how practitioners, including myself, tend to use pre-made lists of stop words before they start doing NLP analysis. So, maybe quickly give us your definition of stop words and then tell us why I should stop using a pre-made list.

Julia Silge: 01:06:17 So, stop words are lists of words that people are like, "Oh, those are not important." I can take them out. And so, they are words like the, and, of. And so, in English, a conservative stop word list would be on the order of a 100 and a more aggressive stop word list would be on the order of like a 1,000. And the way that these lists were made is that they're old. They're old. These lists come from the mid-20th century typically, and they were made by taking, for the time, huge corpuses of language and counting up words, looking at the top 1,000, and deciding where a cutoff should be. And then a person decided where to make the cutoff. And a person decided, "Are there words that I should or should not keep in?"

01:07:10 There are so many problems with stop words. So, A, some of them literally have typos in them. And if you are like, well, is that good or bad? If someone had a typo for... I don't know, I can't think of a long enough word, but let's say if someone had a typo like ADN. Maybe I do want to

take that out. But it's strange that actually there are some words that have typos in there. Another thing that happens is that because these were created from corpuses of language, like many books that were put together, you actually end up with evidence of gender bias just in the stop word lists. Because let's say we take a whole huge number of books. There are more uses in those books of he than she. There are more uses in those books of his than her. And some of those stop word lists actually have for some of those sets of... you make a list of all the English pronouns, they have all of them for masculine, and they only have three quarters of them for the feminine version because of where the cutoff was.

01:08:19 So, you're like, oh, man. Even though you're like, boy, this is the simplest thing you can do, make a list of words. They are all of our challenges around data analysis, data science, data sources, they show up even in this dead simplest thing you can do. Now, I now avoid using lists of stop words when I do topic models because they will always end up being the most probable words. But we just talked about the most probable words are never very interesting. You need to look at these other statistics, give you a better sense of what topics are really about. When I do supervised machine learning, I often leave them in as well because it turns out it's actually informative. The way some the documents use even those boring words, it can be predictive. They can actually have predictive information. Like if you're doing classification, how it uses those boring words can help.

01:09:25 If I'm doing EDA, I sometimes take them out. If I'm trying to show the top most common words and I just am like, well, let me take out those boring words. I do sometimes still take them out there. But I often will supplement it with even EDA approaches that provide a ways for seeing differences across groups that do not depend on taking out those stop words. So, some examples of that are

looking for log odds of words. What are the highest log odds words? I have a package for this that's called tidy low for tidy log odds. And that's actually an interesting approach. Anytime you're looking at differences of counts across groups, it doesn't have to be language. Applying it to language is really great.

01:10:11 I still use things like TF-IDF as an exploratory tool, which I bet many people have heard of. And I've written about what it means, and people can dig into that more. So, that's my pitch. That's my pitch. My pitch is they suffer from the same problems that almost any data science process suffers from, even though they are so simple. And if you're doing unsupervised smart learning machine or supervised machine learning, you probably want those words. And if you're doing EDA, there are alternative approaches that give you better answers.

Jon Krohn: 01:10:43 Very cool answer. Something that I have been teaching for years, something that I was aware of that is an issue in stop words is that for some particular application you might have, you at least need to be looking through the list of stop words. I mean, you've made a good argument to not using them at all, but something at least that I have been saying for years is that you need to know what the stop words are in there. So, for example, if you're doing sentiment analysis, then one of your stop words not, you're going to be pulling out the word not. So, that's one of the most critical words and figuring out the sentiment of a document.

Julia Silge: 01:11:22 Totally. Totally. My first book that I wrote with Dave has an example of this. It's back to Jane Austen. So, it turns out one of the most commonly used stop words has the word miss on the stop word list. Or no, I'm sorry. This is sentiment analysis. Sorry. So, miss is on in the sentiment analysis list as a negative word. So, this is slightly different. It's not a stop word list, but a sentiment lexicon

list. But they're the very similar constraints at play. So, miss is on the list as a negative word, like I miss you or you miss that... I don't know.

Jon Krohn: 01:12:01 You miss quarterly earnings.

Julia Silge: 01:12:02 Yeah, yeah, yeah. Right, right, right. It's on the list as a negative word. So, if you look at Jane Austin novels and you do sentiment analysis using one of these lexicons, it shows up. It's like, oh, the word that is driving negative sentiment in a top 10 way a lot is the word miss. But of course, in Jane Austen's works, that's how everyone is referred to. It's all like Ms. Bennett, everyone is miss, everyone is miss. So, they're not negative at all. Those are neutral words. So, that's a way that applies to sentiment analysis.

01:12:41 And actually, the more sophisticated tools, sentiment analysis tools now, which they do better, but they are actually not immune from this problem because it's just basically fancier counting and fancier linear algebra. They do better, but they are not immune from this exact problem, that words in their training data were used a certain way. And if it's used in a different way, it's like mismatch between your training data and the data you're using it on. And that's one of the real downsides of using the pre-trained models, is that either they're too general, or use a different way, or it's similar problems.

Jon Krohn: 01:13:19 Awesome. Great. Insightful answer there. We got a ton from you. My stop word question has now been answered, and I will stop using them. You've made a clear case. Built in gender bias, these kinds of just weird statistical phenomena around where exactly the cutoff was, what data they were trained on. And so, yeah, using something like TF-IDF is something that I've been using a lot historically, but now you also mentioned tidylo.

Julia Silge:	01:13:49	Yeah, and there's implementations of that in Python as well. If you look up the tidylo documentation, it is based on a paper that was originally implemented in Python. So, that's available in Python as well.
Jon Krohn:	01:14:02	Excellent. Really quickly, and we are running out of recording time here, so hopefully you don't mind if we run over a few minutes.
Julia Silge:	01:14:10	Yeah. Yeah, yeah, I'm good.
Jon Krohn:	01:14:11	But we've talked a lot about tidy modeling throughout this episode. We haven't in this episode defined that. What are the tidy principles? And why should they matter to our listeners?
Julia Silge:	01:14:24	That's great. Okay, so I'm going to speak about it a little bit higher level. What are tidy data principles? And so, I'm sure Hadley would've been able to speak to this, and maybe he did.
Jon Krohn:	01:14:34	Yeah, he would have.
Julia Silge:	01:14:35	Maybe he did. But the tidy data principles are a set of ways of dealing with data. So, you want to have one observation per row. So, let's say you were measuring the temperature of something over time, and you already had five sensors, and every time you make a new observation, you make a new row, you don't make a new column, you make a new row. So, your data ends up long and skinny, not super wide. So, one observation per row, one table for observation. It's actually, if you have listeners who come from the world of databases and database world, it is actually the same as the normal forms of data for databases. How do you make tidy data, tidy or what is a structure?

01:15:26 And the idea there is that when you have tools for tidy data, you can have standardized tools instead of writing a lot of bespoke code every single time. The tidy verse is a collection of our packages that embraces tidy data principles. And that means we have our data in this form. We give you tools. There's tools for converting back and forth between getting to tidy data, getting out of tidy data. But then if you want to do visualization, if you want to build models, you can get your data into tidy form and then do those next steps that you need to take. The tidy verse set of our packages also has some values around reusable data structures, that we don't want to make a new data structure for every single problem. Instead, we want to have a standard set of data structures and get our data into that, then a tool to get our data into that structure so we can have more modular code.

01:16:23 So modularity, reusable structures. So I have not myself worked much on tidy verse packages, but I did work on tidy models packages, and that is a set of packages that applies these kinds of tidy verse priorities and principles. One observation per row, reusable code, modular code so that you don't have to even write. You're like, "Oh, I need to do something slightly different. I'm going to have to write it from scratch." No. We give you these pieces of modular code so that you can put it together in the way that you need to. So tidy models is a framework for machine learning and modeling in R. The two main things is it adopts these tidy verse principles to give someone a fluent way to get from EDA into machine learning. And also has some really great priorities around statistical guardrails to keep you from doing the wrong things, to keep you using good practices around data. Data hygiene is a big one. Right?

01:17:31 It is crazy. I think most of us know, we need to split our data into training and testing sets. Right? But it is pretty wild that even in today's... With machine learning

knowledge being as broad as it is, people still trip up on data leakage kinds of issues. And tidy models is built with that, first and foremost. Can we keep you from leaking your data when you don't mean to? I think the biggest piece there is real explicit adoption of data pre-processing, data engineering as part of your model. You can't think about that as something you do separate. You cannot do it before you split your data. You cannot do it before you re sample. If you're making cross-validation folds or re sampling folds, your pre-processing steps, your data engineering steps have to happen inside of a loop that is doing re sampling.

01:18:24 So that is tidy models. That is tidy models. And I was really excited to... I was working on that team, the first two- ish years that I worked at RStudio, and then Posit. Then moved a little later to MLOps stuff. And then for the past year I've been working on Positron. You can probably tell I'm not someone who has one life passion. I am a little bit of a generalist. I am a little bit of a generalist. I like to learn... If I were to say one sort of, like what puts all that together, it's that I really care about people's real workflows. How are they approaching their real problem and how can we think about the problem they're solving and the tools they're using at a systems level? So if I think about what connects, from text analysis to MLOps to building a freaking IDE, what I think connects those for me is I really like applied work. I really like practical work. I really like thinking about how people approach their task in a very concrete nitty-gritty kind of way.

Jon Krohn: 01:19:31 Awesome. It's nice to get that insight from your background as well and how everything ties together. But most importantly in that recap was getting that overview of the tidy verse and tidy models. And we did certainly in Hathley's most recent episode, 779, talk about the tidy verse, but I don't remember for sure if we talked about tidy models very much, so thank you for those insights as

well. I did post a week prior to you coming on the show. I posted on social media on my LinkedIn and Twitter accounts. I still call it Twitter, and I posted that you would be coming on the show. There was a huge amount of engagement, hundreds upon hundreds of reactions, tens of thousands of impressions. And we did have some interesting questions as well.

Julia Silge: 01:20:18 Oh, nice.

Jon Krohn: 01:20:18 So I've got two for you.

Julia Silge: 01:20:20 Okay.

Jon Krohn: 01:20:21 The first one here is from Luke Morris, who is a healthcare data scientist at the Stanford University School of Medicine. Luke says that you are awesome and that your book with Emil... I'm going to butcher his last name, Hvitfeldt.

Julia Silge: 01:20:34 Oh, Emil Hvitfeldt. Yes. Yes.

Jon Krohn: 01:20:37 Hvitfeldt. There you go. So that book, "Supervised Machine Learning for Text Analysis in R". That was Luke's North Star on his graduate capstone project. And so his question is, as a major #TidyTuesday contributor, what's been the biggest oh wow moments you've had digging through these weekly data sets?

Julia Silge: 01:20:58 Okay. I love this question because actually something comes directly to mind for me. There was a data set in 2020, a Tidy Tuesday data set that was from back in 2020 that was about the voyages of enslaved Africans. It was the data set on the last several decades of the Trans-Atlantic slave trade that... It happened during 2020 and it was like, "Let's explore this and let's learn this more." And it showed that... I didn't know this going in, but it turns out there's this exact example of Simpson's paradox.

Classic stats level, intro stats Simpson's paradox, where it's the mean year of arrival and age. It looked like they were going up together, that when these enslaved Africans were arriving on the Western Hemisphere it looked like over time people were older as they were arriving. But it turned out just to be Simpson's paradox because the earlier years they were proportionally more women than men and they were bringing boys earlier and all this kind of thing.

01:22:26 So first of all, pretty heavy topic, pretty heavy deep topic where you're like, " This is rough actually looking at this data." And then I went through this example, this modeling. And no one had told me ahead of time, and in fact I don't think anyone had looked at this particularly, but you make a first initial plot and you're like, "Oh, look, people are getting older with time." But it actually turns out was the opposite and it was just Simpson's paradox. And so that really sticks with me partly because it was a heavy topic and I really remembered it. And because I literally just discovered... I mean, I just found the Simpson's paradox example as I was going. I'm like, "It's real. Everyone, Simpson's paradox is real."

Jon Krohn: 01:23:08 Yeah. It's the Simpson's paradox being, if I remember correctly, you alluded to it there is that it appears that there is a correlation one way between two factors, but then when you condition upon some third factor in this case, sex.

Julia Silge: 01:23:24 Yes.

Jon Krohn: 01:23:25 It turns out that actually, well, that you had more women coming over and women tend to be younger-

Julia Silge: 01:23:30 Later. Yes. Yes. Yeah.

Jon Krohn: 01:23:32 Yeah. Yeah.

Julia Silge: 01:23:34 So it flips. The sign of the effect flips when you end up controlling for this thing that it turns out it is important. Yes. Great experience.

Jon Krohn: 01:23:41 Yeah. So in fact controlling for gender, the slaves were getting younger.

Julia Silge: 01:23:46 That's right. That's right. That's exactly right. That's exactly right.

Jon Krohn: 01:23:49 Yeah, that is an interesting, although you're right, a pretty heavy topic, and so we'll see if the second question ends up coming down.

Julia Silge: 01:24:01 I don't know. Hopefully we can be a little bit lighter this time.

Jon Krohn: 01:24:06 So yeah, so this one is from Otto Hansen. He's a data engineer and data scientist based in the Netherlands and Otto says that he's looking forward to this episode. His question would be, how does Julia reflect on the rise of AI as an aid for teaching future generations of data science students? Can AI in your view, Julia, either be a force for good in teaching students how to code, or do you think that AI poses a threat to properly teaching or mentoring future data science students and leaders?

Julia Silge: 01:24:37 Yeah, this is a really great question and I have my use of LLM-based tools, which... So I'm going to be a little specific because AI is one of those words that's like what on earth does it mean? What on earth does it mean? So when I have used LLM-based tools for code assistance, I feel like it has overall helped me. And I think that it's partly because of when it has come in my life and also partly of how I have used it. So if I were to say use Copilot or one of the alternatives to Copilot, it is most helpful to me when I am trying to do something in a language I am a little bit less familiar with. So I would say in terms of

my own background these days I write a lot of TypeScript, I write a little bit of Rust, I write a little bit of Python, I write a medium amount of R. And I write a lot of YAML.

01:25:40 But anyway, and a big proportion of those languages are not languages I have used my whole life or even 10 years. I've used them. A good proportion of those I've used less than 10 years. And so if I know what I want to do and I can't remember syntax for how to do it in a certain language, that's actually been quite helpful for me. If I am doing R, especially if I'm writing EDA, tidy verse EDA code for R, I'm faster than the prompts and I feel like it gets in my way and I'm like, ah, this is so frustrating. That's where if I would say, what am I most productive at? What am I, my expertise where it is the highest? That's probably what I'm the fastest at writing. And I think that's partly because those tools are so well-designed, like the tidy verse is so well-designed that it is a really good fit for how I think about data and analyzing data.

01:26:37 So that's an example where I'm like, I don't actually find it that helpful in my strongest competency. I find it quite helpful in my slightly where I'm still, I'm a little bit slower, I would maybe have to look something up. It is quite helpful in that area. Now let's talk about teaching. Let's talk about teaching. I don't know that it would be that helpful in the long run for a learner to use it when they're learning their first language. I think that it may... Because you maybe don't have the ability to evaluate as much. It spits something out and you're like, is this good or bad? I don't know. So I don't know that it's so useful for someone trying to get competence with their first language. I think there are some real how can it... What's an effective or good use in a learning environment? And also what do we want someone to learn?

01:27:37 Maybe I could envision someone teaching a class where it's like, Hey, we're going to learn to code in Python and

you all have one of these things installed. Okay, write a comment that says, I don't know. Let's picture a very intro class, write a comment that says, write a for loop that does blah, blah, blah. And then it'll pop up the thing and maybe someone could teach saying, look at it, read it, predict what will happen when you run it, and then do that. So I think it's possible to use these in teaching environments that are going to be good. Because of my own background, it's a little hard for me to envision it being really useful for a learner of a first language. I see tons of people around me using it, those tools really useful for additional languages. Okay.

01:28:27 Now, that was all about code. What about other uses? What about other uses? I have not had great experience with using these tools in other uses. If I try to have it write something for me, I hate the voice that it's written in. I'm like, ugh, I hate it. No, that doesn't sound like me at all. That sounds like this sort of gross machine. They all have this kind of voice when they're generating text prose, when they're generating prose, and I hate it. I'm like, no way would I send an email that says that. That's ridiculous. So I've had less luck with it. In text generation outside of code, I've had pretty good experiences with them, text generation in code. I think it will take some real thoughtful educational people to think about how do we use these in really helpful ways to help people get the real skills. I mean the real skills when it comes to code is not writing syntax.

01:29:29 That's kind of almost what gets in the way. And if we can use... Can these tools help us solve that? Maybe, maybe. So overall I put myself in the I'm not like morally opposed to these tools, although of course there are all kinds of questions around the data that was used in to have trained them, the licensing. There are some complicated issues, but I would not say I am in principle opposed to them. I have had use using them. In some ways I'm kind

of skeptical about how they are used broadly. And of course the fact that it can just generate unlimited amounts of text and then we can just put all that text on the internet. There are some real... Are we going into a information where we cannot find good information because there's too much literally text being generated as poor quality? Very interesting questions. I'm not someone who feels like I have the answers. I certainly have opinions based on my own experiences of playing around with them.

Jon Krohn: 01:30:28 You touched on a lot of different topics there. We could probably spend a whole episode with me kind of digging into those. But yeah, it's nice or it's interesting to hear that at least for the software development education use case, you do see a lot of potential. So do I as well. And for me personally, it's these kinds of in the flow tools like Gemini and Google Colab have been amazing for me to be developing software way more quickly. I mean, instead of having to search over the web to find some Stack Overflow answer, it's not exactly in the same situation with the same version and obviously different variable names and I have to kind of figure things out now. It's often just a click of a button to getting something to work. I'm constantly blown away. It is interesting with the tone thing that you mentioned there with natural language generation.

01:31:18 That is something I think that is getting better. In fact, just last night at the time of recording, Natalie, who does operations management for our podcast, she said to me, "Wow, Google Gemini just suddenly something changed." Google Gemini is her preferred LLM and she uses it specifically for emails. And I think she provides context of ideal emails that she's written in the past. So she's like, she wants her tone to be mimicked. She was like recently, it just-

Julia Silge: 01:31:47 It really improved.

Jon Krohn: 01:31:48 -In the last few days.

Julia Silge: 01:31:50 That'll be really interesting to see how these tools develop moving forward.

Jon Krohn: 01:31:54 Yeah, for sure. That is it for our audience questions. My penultimate question for you, one that I always ask our guests is if you have a book recommendation for us. Obviously we know about your books, but maybe you have someone else's book to recommend to us.

Julia Silge: 01:32:08 So based on that discussion we just actually had, I'm going to pull one off of my shelf here because it is, I think so relevant. So it is called "The Programmer's Brain" and it is by Felienne Hermans. And it is all about how can you learn to be a better programmer? And how do people program, how can you increase your own skill in programming? And it is super interesting to read it. It kind of came out before the rise of the LLM based tool. So it does not address those, but it is extremely interesting reading it now through the lens of these tools being available. And what are the real skills? When it comes to being someone who writes code, what are the real skills, like the real skills that people are using and how can you make yourself more effective?

01:33:01 So I'm going to say that is my recommendation because of that discussion we just had about where are these tools helpful. I think if you or your readers look at it with that mindset, you'll be like, ah, interesting. This can come in here. This can come in here in this process and really make me better. Or, oh, this is why I'm really frustrated when I try to use it here because it's not supporting that kind of work.

Jon Krohn:	01:33:26	Excellent recommendation. I love it. And final final question is how people should follow you after this episode, Julia?
Julia Silge:	01:33:33	Yeah, so I would say YouTube is a great place. Me on YouTube. My blog, I post there. In terms of social media, the places that I am hanging out these days are LinkedIn, Blue Sky, and Mastodon.
Jon Krohn:	01:33:47	Awesome. Thank you, Julia. This has been an amazing episode. You've also been really generous with your time. We've gone well over the scheduled recording slot, so thank you so much. We really appreciate all the insights you had today. And yeah, maybe we can check in a couple of years and see how your projects are coming along, maybe how Positron is coming.
Julia Silge:	01:34:05	That would be so exciting. Thank you so much for having me.
Jon Krohn:	01:34:14	Boom. So much rich, actionable detail in today's episode with Julia Silge. In it, she filled us in on how the polyglot Positron IDE is designed from the ground up to be ideal for people who do exploratory data analysis, including updated variable panes and key consideration given to Data Viz. She told us how Continue, Tabnine, and Codeium are her favorite LLMs for code generation. She told us how NLP should be used instead of LLMs when we need higher statistical rigor when carrying out EDA or working with larger data sets. And how we should use TF-IDF or her tidylo library in lieu of stop words because of issues like typos and demographic biases. As always, you can get all the show notes including the transcript for this episode, the video recording, any materials mentioned on the show, the URLs for Julia's social media profiles, as well as my own at superdatascience.com/817.

01:35:07 And thanks to everyone on the Super Data Science podcast team. We've got our podcast manager, Ivana Zibert, media editor Mario Pombo, operations manager Natalie Ziajski, researcher Serg Masis, writers Dr. Zara Karschay and Silvia Ogweng, and founder Kirill Eremenko. Thanks to all of them for producing another magnificent episode for us today. And for enabling that super team to create this free podcast for you, we're deeply grateful to our sponsors. You can support this show by checking out our sponsors links. Please do that. You can find them in the show notes. And if you yourself are ever interested in sponsoring an episode, you can get the details on how by making your way to jonkrohn.com/podcast. Otherwise, please share, review, subscribe, and so on. But most importantly, I just hope you'll keep on tuning in. I'm so grateful to have you listening and hope I can continue to make episodes you love for years and years to come. Till next time, keep on rocking it out there and I'm looking forward to enjoying another round of the Super Data Science Podcast with you very soon.