posit conf 2024 conference notes

Preface

This document contains a summary of my conference notes from posit conf 2024. Talks, points and code has been reproduced to the best of my ability, but of course there is a possibility for misinterpretations due to the high density of talks, and my poor handwriting/limited understanding of the subject matter. If nothing else, I hope my writing will convey my excitement for the conference, and the novel developments for R.

Workshop - databases with R

First day of the conference was workshop day. I attended the I attended the workshop "databases with R". For course material and scripts, see https://github.com/posit-conf-2024/databases

I have collected some more workshop material that may be of interest to others: I can highlight the following: **Level up with Shiny for R Intro to Shiny for Python Intro to Quarto**

In the Databases with R workshop, we learned how to create, and send inquiries to databases using the **duckdb package**. Course material can be found using the link above, with scripts containing exercises to understand how you can use DuckDB for data wrangling. The creator of DuckDB Hannes Mühleisen gave a fantastic talk as the final keynote speaker - this talk highlights some of the advantages of using DuckDB and is worth checking out.

The course material did an excellent job of explaining how a database is structured, how you connect and disconnect, how you send queries using R syntax and how you can convert to and from SQL.

Special emphasis was also placed on the parquet file format, and how it is much faster to load than csv. They demonstrated how you could partition a dataset based on various parameters, and save it locally as parquet.

Lastly, they demonstrated how you can create a local database with duckdb, how you add data frames to it, append them and delete them.

Day 1

Updates from Posit

new Quarto features

Firstly, Charlotte Wickham presented updates from Quarto. She focused (among other things) on using Quarto for website building with custom output, and on using quarto for documentation. She presented a new Dashboard HTML format, which uses card-based layout similar to bslib for Shiny. This is enabled through the header `format: dashboard`. Multiple rows can be enabled through `## Row` . The dashboard can contain several other bslib features, including value- and input boxes. You can also add various widgets, including plotly graphs and Shiny code. A guide to dashboards can be found **here**.

A number of quarto extensions were also presented. A new PDF framework for Quarto called Typst aims to be easy to learn and work fast. It is currently in development. You enable it through `format: typst` and should be able to generate HTML quality in PDF with better breaking of multipage documents.

other features included a native Julia engine and a presentation of the scrolly-telling extension, which is discussed in detail further below. For a list of extensions, see **here**

WebR

George Stagg presented WebR, R for webassembly which can be used to test R code in the browser (available **here**). The current version contains several installed packages which can be loaded. A new feature includes Shinylive, which allows you to run Shiny code without a server.

A new package is Quarto live, which allows you to make interactive documents and course exercises in Quarto. This is enabled through `format: live-html` and by changing the header of code chunks to `{webr}`Through this code, graphs will be rendered every time document is opened, and not just when document is rendered.

Exercises can be created by adding an "Exercise label" to the code chunk as follows: `{webr} #| exercise: ex_1` Then you enable a specific exercise layout. You can also add hints through `::: {:hint exercise="ex-1"}[the hint]` and you can make a solution block with similar syntax (using quarto syntax). You can also create custom feedback algorithms, tailored to pre-defined error types.

WebR is also designed to render well on a phone screen automatically, which also makes it suitable for things like creating exercises. You can also create dashboard-like experiences with Webassembly and observable JS. Github repo **here**.

Snowflake and databricks

Last bit was about infrastructure management, and how posit products are supported by the two systems. Posit workbench is available for Snowflake as app in infrastructure. They have simplified database connections to the systems by adding support for them to odbc package. Emphasis was also placed on application deployment, how Connect has native support for authentication, and how you can custom tailor access rights.

Navigating a career in data science

This session discussed various aspects of how to build a career in data science. First talk focused on network building, and how technical skills is only part of building a professional career. You can build a network by setting small, tangible goals: talk to someone at an event, join a ressource group, join an online community, etc. Building a career is also about finding a niche, and the presenter gave some suggestions to roles you can aim for: subject matter expert, the "person who knows everyone", data workflow person, or manager role. Path should depend on passion, what sets you apart, and what you can bring to the table. Lastly, the presenter gave some advice on how you should "know your worth and ask for it" when it comes to salary (for example by talking to others of similar experience level as you), and by finding new projects or developments you can advocate for to continue your personal development.

Next session was about GitHub, and how you can use it to highlight your professional profile. The presenter highlighted GitHub as an ideal way to build a junior CV because it demonstrates tangible skills, and it shows you know good development practices. Put projects on Github early in the process so they can be displayed. you should aim for your code to be easy to run, and be understandable. Repo essentials include modular code, all work in code (no Excel shortcuts/ missing bits), organized, documented, and that it makes sense. Think a lot about what projects you want on the Github landing page, as it will default to some of your early work if you don't actively set it - Githubs default display is rarely the best choice for you.

Talk no. 3 was about building a pragmatic data team. In a team it is easy to doubt whether you are a "real" data scientist - the speaker highlighted the importance of being confident, as it is very easy to add qualifiers. It is easy to accidentally create a "value continuum". The presenter shared a vision for a "pragmatic team" model that focused on ad hoc solving. Expectations for team should be set carefully and honestly, as mismatch is high risk if these are not aligned. It is also important to say "no" when needed, and to not make "heroic sacrifices" in the name of the project, and to avoid a "you built it, you own it" mentality. The speaker encouraged the audience to make their own fate instead of letting others define what is means to be a data scientist.

Last talk of the session was about finding yourself in a manager position, and how to adapt to that (something I found some resonance in, I must confess). The title was "oops, I am a manager", and the "oops" was added because first leadership responsibility is often given as a "battlefield promotion". The speaker suggested that many organizations lack management training for early stages, and tends to take themselves a little too seriously. He talked about a concept he called "minimal viable process":

- "minimal" because heavy process tends to paralyse things
- "viable" because someone is paying the bills, and they expect viable progress
- "progress" which is how things get done

The speaker talked further about Process and distilled it to the following formula: Gather and order –> do stuff –> deliver

- Gather and order is about considering all stakeholders and the level of input they provide. Specify as much detail as needed (but only as needed)
- Do stuff: leave as much space as possible (avoid micromanagement). Keep meeting frequency (and who needs to be there) as low as it needs to be
- Deliver: beyond final product, other aspects of delivery are too often neglected - remember for example to credit team for wins. And reflect on the process as leader.

There is no right process. Observe and reflect together as team. Do retrospectives. What went well/didn't work well. Ask team "which part of the process kills your soul the most". He presented "popcorn method" - (problems, options, possible experiments). Work to battle excuses: - "not us, them"–> time for collaboration - "we don't have time" –> change the process - what doesn't work? - "it won't make a difference" –> then try again and learn from last - "management won't let us" –> challenge this with management

Whats new with Tidymodels

Session was kicked off with a presentation by Hannah Frick, who talked about time-to-event data. recently, survival analysis was added to the tidymodel framework, where data is stored as a "surv" object. The flow was presented (`set_mode("censored_regression")` and `set_engine("survival")` followed be `recipe` and `predict`. They further expanded the framework to include time-to-event data.

Second talk was also about time to event data, by Max Kuhn. Evaluating time-to-event is hard, because an event may never occur, and hence outcome data can be missing. the presenter showed a censoring model, a "reverse kaplan-meier" plot. Compute probability of sampling at time tau. Finally, framework was presented for Brier Score Code (Basic "Yardstick" framework). He also discussed area under ROC curve, which is more straight-forward, but requires indicators, probabilities, and weights at time tau. In summary, he argued these models are complicated to evaluate, but can be handled by tidymodels w. clear syntax for complex statistics. (the first two talks are a bit shaky, as I am not experienced in using tidymodels, and therefore a bit rusty).

Next followed a talk on fair machine learning by Simon Couch. He started by presenting the discrepancy between predicted and accurate data, where models do not predict extremes accurately, and tend to be too conservative. Called "regressivity" - models tend to average. He made the bold claim that the concept of fairness is not about math or measures, but about belief, and is context dependent. Next followed a discussion on fairness - normal paradigm would be that "a performant model is a good model". By this logic, is it good that error is distributed equally at low and high ends? This becomes problematic if the model for example predicts prices - if it is equally off at high or low ends, the low end may be disproportionally punished by inaccuraries compared to the wealthy. He countered that a percentage error is more fair. Machine learning should aim to implement as many fairness parameters as possible. He presented homeownership as an example, with fixed percentage for homeowner exemptions. Metrics evaluate model, model is part of higher system. He further talked about some applied articles of tidymodels, such as fairness of GPT detectors and hospital re-admissions. More is available at simonpcouch/conf-24

The session was closed by Emil Hvidtfelt, who talked about his Orbital package which aims to run predictions with tidymodels in databases. The goal is, of course, good predictions. Predictions in databases can be done with tidypredict, and while it offers supports for many modes, it is limited to 1 equation. Recipes support is hard for redundant calculation. Classification probabilities are awkward. As an alternative to this, the Orbital package was developed, and creates output with several equations. On databases, you can predict directly with SQL, and be run in a Shiny app too (live predicting). Cons are that not all models are supported, you don't get input checking, and it is still very new. Pros are no docking container is needed, you can run predictions in databases, and generate code. As a side-note, discussion was also presented of the Butcher package, which reduces the size of fitted objects.

Innovation with Shiny

The session started with a talk by Kiegan Rice, who talked about turning Shiny apps into a full exploration experience. Shiny can be made into looking like a website by adding a landing- and about page, and by adding nav bars, and other guiding tools. Shiny can be used to customize and enhance user experience by reacting to user actions visibly, and by adding notes that provide context. The presenter highlighted a feature that allows users to share sessions/tables as URL, which seems very nifty. She encouraged Shiny programmers to learn CSS, and to reflect on what users want to learn when building the app. The session had some nice advice for building with Shiny: Understand the data structure, build the interface, connect data and interface.

Second session was called "We can have nice shiny apps" and was presented by Greg Swinehart. The subject was how to improve Shiny UIs. He presented multiple subjects related to UI, for example how he had made a color palette based on Shiny blue, which is the bs theme default. Bslib is where active development is happening, and how you make modern apps for R. He presented a new Shiny Components Gallery for Python for UI elements to provide an overview of what can be added to Shiny apps, and a similar gallery called Shiny Layouts with different layout examples to get started with Shiny for Python.The future of Shiny building is through the card element from bslib. Templates are supplemented with a series of Youtube videos. He also briefly talked about the "fontawesome" package which seems very nifty

Next session was "Making an app a system" by Andrew Bates, who talked about data processing outside of the Shiny app itself. He talked about using YAML mapping to reference keys with values in YAML file (though I am not sure I understood the greater purpose). He presented a package called matte that provides support framework for adding data pipelines outside of Shiny.

Last talk was called "portfolio analysis and package management through Shiny" by Lovekumar Patel, who demonstrated a portfolio analysis system using Shiny and Plumber API. The talk further explored AgGrid, a javascript based grid that integrates through HTML widgets.

Keynote: Generative AI by Melissa Van Bussel

Melissa Van Bussel delivered an inspiring talk about the opportunities, dangers, and fatigues associated with generative AI use, and delivered some specific examples of how to use generative AI in science and data science. She has a youtube channel about generative AI (among others, this includes a video on how to make your own generative AI in R, and another on how to train customAI models). She shared her initial great excitement for GPT 4.0 which combines text, visual, and audio without a paywall. It has image process capability, can take pictures, upload in chat. It can convert images into quarto documents, add styling to match, read data from images. you can also upload files. It can generate summaries, explain data, export key insights. Advanced data analysis feature allows you to export data - this can generate fear of replacement for data scientists, but proper prompt engineering requires knowledge about subject matter. She detailed some features of prompt engineering, and compared it to building a plot with ggplot.

For specific examples of application she showed a quarto theme designer she made - user enters prompt about theme, and theme is generated. She also made an Rpackage sticker generator (Recraft AI).

She next shared some tools to make data communication easier: Scribe, a tool to create step-by-step turorials without copy pasting. Can be embedded as quarto but that requires professional version. Second tool was called descript, which allows you to edit video and audio as if editing word documents,. You can remove filler words or long silences or even make a voice clone.

For the AI fearful she talked about responsible use of AI, with a basis in the guide made by the canadian government. They introduced a set of principles called FASTER - Fair: output should be fair, and comply with decency - do not use AI content to make decisions that can impact people or material. Review outputs manually and removed biased content. Accountable: Don't use AI as search engines, don't use them for a skill you do not possess. Verify outputs. Consult legal departments or review policy. Determine if training data was obtained legally or ethically sound Secure: Abstract as much as possible, use synthetic data, never upload sensitive information. Read the terms of service (really!) Transparant: Keep detailed record on how AI was used. Add watermark to images and add disclaimers Educated: Understand tools before using them. Learn how to make efficient prompts Relevant: Consider if AI is the best tool for the task?

Day 2

Keynote: Future of Data science by Allan Downey

Allan Downey talked about how the public has perceived data science, how expectations have risen and fallen, and how news outlets creates an atmosphere of pessimism where data on many fronts actually suggests that we should be more hopeful about a lot of things. He recommended a book by Hans Rosling on the subject. Data suggests that young people are less happy than previous generations, due to misinformation by negative media. Particularly he recommended "Not the end of the World" and "Stop reading the news". He encouraged us to use data to understand the world better. Many in the audience (particularly younger people) were very triggered by his message, and accused him of forgetting his own privelege as a white male.

What's new with Shiny?

AI and Shiny - Joe Cheng

I was excited to hear Joe Cheng, the original developer of Shiny, speak. He talked abotu what Shiny has to offer AI and vice versa. The talk was based on a specific request on how to upload video to GPT-4 through Shiny. Created package called "shinymedia" for python with functions called `input_video_clip()` and `audio_spinner()` to return response from AI. He then demonstrated how he had written a script that can analyze video and get audio response. He recorded the room, asked GPT-4 what it looked like he was doing, and it replied "it looks like you are giving a presentation to a large audience. It looks like fun!"

He further demonstrated how chatbot can be added to shiny session (a sidebar that serves as a chat). Chatbot has no access to data, it only controls SQL queries. It can be used to explore data (it can only send queries and receive data, it has no access to raw data). The model could do things they did not expect, like identifying and removing outliers. They added button to plots that allow AI interpretation of plots - works for base, plotly and ggplot. Generates text about the plot. You can ask follow-up questions in chat, as output contains chat window. Demo here. They realize SQL is stunningly good for connecting LLM to data app. LLM can easily write SQL and it is human friendly. He encouraged the audience to start coding with LLM - they are not “stochastic parrots”, but also not “machines that reason”. Stay skeptical and curious.

An assistant for Shiny - Winston Chang
Winston Chang talked further about AI assistants and Shiny. They made an assistant (ShinyAssistant) that can build simple demo apps for R and Python. It can make errors, but is capable of iterating. It can help you debug error messages. He highlighted it should be used as an assistant, not as a replacement for a competent designer.

He also talked about Shiny Live, which now runs in browser, not remote server. This means sessions can be shared with link. He talked about embedding.io which allows you to crawl, chunk and vectorize websites so their content can be used for LLMs. You create a collection of sites, and then query it with their API. Can be coupled to chatbot allowing you to send queries to collection.

As the others, he emphasized that you should never submit confidential data or openAI API key in chat

Supercharge your shiny for Python app: unleashing jupyter widgets for interactivity by Carson Sievert
This talk focused on Jupyter widgets. First bit focused on quak package which turns data frames interactive.

Widget binding can either be one-way binding (python to JS) or two-way binding (python and JS both connected). Discussed some pros and cons of this, and how ipywidgets allow two-way binding. Shinywidgets is ipywidgets in Shiny. has `render_widget()` function that allows widgets. Showed some examples of for applications, and discussed how this works.

Dataframes for Shiny with Python
Compared various data.table renderings for Shiny: - DT: nice interfact, fast, filtering, no editing, - rHandsontable: no filtering, but editing - Excel-like interface - Reactable: very fast, highly customizable, but no editing - GT: all aspects of table modifiable, built in structured approach like ggplot. Later it got an interactivity feature.

Shiny for python now contains render data frame support. Still early stage, but high potential, allows selecting in table, and rendering in plot. Briefly talked about reactlog -a package to evaluate reactive inputs in shiny (similar to Targets flowchart). Also talked about empowered Renderer - it contains consolidated reactive values and methods. Talked about various Python table features that will be added in Shiny going fowards.

Python Rgonomics - Emily Riederer

This talk recommended a number of Python tools to get R users started in Python data science.

- `Polars` is a good package to explore data - syntax is similar to dplyr. Showed examples to support,

- `Plotnine` is a robust clone of ggplot2. `seaborne.objects` shows some more python-specific features for visualizations.

She further argued that quarto with the current python support is good alternative to jupyter. Dependency management can be handled with `pdm` package.

She also spent some time discussing VS code and positron, and how it creates "customizable data-first" developer experience.

`Cookiecutter` allows you to structure projects from templates. `ErrorLens` aggressively highlights errors in IDE. `Ruff` adds lint and style options.

Pour some glitter on it - custom Quarto Outputs

Transfering from the Python session, I missed some of the talk by Meghan Hall, but it was about designing and deploying quarto templates . She talked about why to spend time on the process. It can generate a common visual identity for organization, and add custom features like conficentiality banner. She highlighted how DevTools can be used to collect inspiration. Use the inspect elements feature (right-click and inspect) to see how an object is made with css. She talked about how to share custom templates . One of my main take aways was that I should learn more css. She also encouraged the creation of internal packages.

Closeread: bringing Scrollytelling to Quarto - Andrew Bray

This talk was a fantastic eye-opener. It focused on scrolly-telling, where figures will appear, zoom and disappear as you work through text. Allows you to tell a story. Their quarto extension is called "closeread". "Fenced div" syntax:

```
::: {#cr-myid . myclass myattr = "val"}
Bla bla
:::
```

Check it on https://closeread.netlify.app/ Once the plugin is installed, you change the quarto format to closeread-html. He explained the concept of limelock (adding |to text to preserve formatting such as new line)

Example on how to use closeread: You want `cr-myid` from above to appar so you call it as `@cr-myid` in the text. It seems like you need to leave two breaks after. You can highlight and zoom by calling things within the cr-chunks. Example:

```
::: {#cr-myid . myclass myattr = "val"}
Bla bla
an important (piece of info){#cr-important} you want to highlight
```

Can be called in the following section: `:::{.cr-section}`
`@cr-myid This will show the full text`
`[@cr-myid]{hlz="cr-important} This will highlight the important bit from line 2`

Scrollytelling uses spatial cross-references. Allows reader to focus on what is important instead of telling them. Particularly useful for graphics. If we for example have a map called `@cr-map` we can pan by adding the following commands `[@cr-map]{pan-to"70%,-10% scale-by=2.7}` In this

case the map is created as `:::{#cr-map}`
`![](map-name.png)`
`:::`

Commands include pan-to (pan across sticky), scale-by (scale sticky up or down), hlz (highlight and zoom to line), highlight (highlight line or spans) and zoom-to (zoom to line)

It can also be used to study code by adding code chunks within the CR format, using the calssical ```` ```{r} ```` format (include echo and message with #|). Lines of code can be highlighted with `highlight= 1,3`

Data visualizations: Idea> Process > Sharing

Creating multi-figure visualizations with Patchwork - Thomas Lin Pedersen
First session was a musing on design philosophy, using Patchwork as a case. The goal of patchwork was to make it intuitive to build multi-panel figures. The goal is to make API-first package design. The advice from Pedersen was to "have a mission", "embrace the no", and aim for "powerful simplicity".

The talk was mostly on design philosophy but he did give one piece of advice that seems very useful - the `free` function that helps alleviate misalignment during patchwork construction, where an axis title will floating due to plot alignment. He also highlighted the `collect` function that can be used in plot layout settings to ensure you only have one legend and/or axis in your assembled plot.

From idea to code to image: Creative data visualizations in R - Georgios Karamanis
This talk focused on getting inspiration for great visualizations. If you see something you like, spend a minute investigating why it works and what you like. He likes to recreate visualizations made with other tools. He encouraged users to explroe the various `geom` functions in ggplot, and to check out `ggforce` if you want to draw solid lines. He also encouraged the use of `ggplot_build()`. He encouraged participation in challenges, and to explore color schemes and fonts. And he recommended that if you are in doubt about a design, subtract and simplify if possible.

Animated web graphics in Quarto with Svelte and other tools - James Goldie
James Goldie talked about using Observable Javascript (OJS) in Quarto . To enable OJS chunks replace `{r}` with `{ojs}`of course you need to familarize yourself with Javascript syntaxt for this. He spent some time on "observable plot" as an alternative to ggplot. You can load in R or python objects with `ojs_define(object)` or read it in from csv `data = FileAttachment("datafile.csv") .csv({ typed: true})` Observable plot has build in tool tips that we can make accessible to individual marks `Plot.tip()`

Following this part, he talked about simulating animation and conditional content - the example was adding projection to turn map into a globe. OJS is inherently reactive, so you can for example make a globe rotate continuously (he gave an example by defining a value angle that change based on sys.time and then added `rotate:[angle, -10]`).

He emphasized that if/else works differently in ojs because things are natively reactive. Every time condition changes, they will update so he showed an example where a timer changed text, an image or a drawing continously back and forth. You can also make content respond to window size and change accordingly. He stated that a limitation of OJS is that you can't easily transition from A to B because when you switch to B, A is destroyed. He then presented his vision for "declarative graphics" - focus on what to draw not how to do it. He presented `Sverto`, so you can make smooth transitions when a parameter changes (so no abrupt transfer but a gradual one). It looked incredible (uses Svelte and D3.js). Check it out here: https://sverto.jamesgoldie.dev . The workflow is 1. import svelte components, 2. tell Sverto where to add it, 3. update component with OJS.

He gave some additional websites to check out: Quarto docs: https://quarto.org/docs/interactive/ojs Observable Plot docs: https://oservablehq.com/plot observable Inputs docs: https://observablehq.com/documentation/inputs/overview

Be kind, rewind - Ellis Hughes
His talk foucused on the `camcorder` package, which saves plots as you build them instead of doing it manually, if you for example want to demonstrate a process. A bunch of images can be saved and rerun with `gg_playback()`. Camcorder can also be used for animations. https://github.com/thebioengineer/camcorder He also called attention to the `flametree` package https://github.com/djnavarro/flametree which allows user to make generative art in R.

Keynote: Data wrangling like a boss with DuckDB by Hans Mühleisen

This exciting keynote was about `duckdb` and the challenges that led to specific design choices in the framework. Hans Mühleisen's first main message was that if data fits in memory, you generally should not put it in a database as this will generally be slower and more frustrating. He discussed various csv readers as an alternative - he benchmarked with a 800 mB file and the various csv readers could read it in between 3 (fread) and 22 sec (read.csv base R). His mission was to create something that was fast and intuitive and that led to duckdb. It is an in-process analytical system so it doesn't require setup or server - you just install it with pip or renv. It is a fast, free, and fully featured data management system and wins benchmarks at 50 gB. He showed how it has one of the best csv readers (comparable to fread).

He then presented `duckplyr` - using the syntax of `dplyr` - he showed examples where duckplyr could bring down runtime from 15 sec to 1.5 sec. Duckdb has native support for parquet files, and if you have multiple files, it can be concatenated into a duckdb file for easy subsequent loading. Duckplyr is still in development and will fall back on dplyr if it encounters an unexpected issue. You can also create various macros of things that you do a lot.

He then described how he sees us as moving towards the end of big data. Hardware improvements mean we will move towards a future where data can be processed on a small laptop - a "data singularity". He believes single node computations is the future.

Lastly he talked about duckdb extensions where they will open duckdb to community extensions, which subsequently can be installed from duckdb. They aim to become their own ecosystem.