
GuIT meeting, Pisa 2025: A visit by Doris and Erik

Doris Behrendt, Erik Nijenhuis

1 Why Pisa, why GuIT?

Doris had attended the GuIT (the Italian-speaking \TeX user group) meeting in Turin in 2019, representing DANTE e.V. (the German-speaking \TeX user group) board and delivering birthday greetings to Claudio Beccari. She also wrote an article about it in DANTE's journal [1]. Because she had fond memories of this conference, she decided to visit the Italian user group again, this time to attend the one-day meeting in Pisa, and invited Erik to join. Together we captured notes, context, and photos, and we prepared this short report for the wider community. The theme for the day was practical: how to use \TeX and friends effectively across typical workflows, from mathematics and graphics to bibliographies, presentations, and even modern alternatives.

2 The program at a glance

The meeting ran on a well-balanced schedule, with ample time for questions and coffee. Here is the day (November 5, 2025) as we experienced it:

- 09:00 *Conference opening and welcome speech.*
- 09:15 *L^AT_EX and its tools, part I: General introduction and mathematics* (Massimiliano Dominici).
- 10:00 *L^AT_EX and its tools, part II: Graphics and bibliographies* (Massimiliano Dominici).
- 10:45 Coffee break.
- 11:15 *L^AT_EX and its tools, part III: Presentations with Beamer* (Grazia Messineo).
- 12:00 *L^AT_EX and its tools, part IV: Using L^AT_EX on line* (Salvatore Vassallo).
- 12:45 *L^AT_EX and its tools, part V: Typesetting a PhD thesis with L^AT_EX* (Massimiliano Dominici).
- 13:30 Lunch break.
- 14:30 *Parallel poems* (Claudio Beccari and Cecilia Benassi).
- 15:00 *Symbolic links* (Claudio Beccari).
- 15:30 Coffee break.
- 15:45 *Linear regression* (Battista Benciolini).
- 16:15 *Typst: A new alternative to L^AT_EX* (Roberto Giacomelli).
- 17:00 GuIT general meeting.
- 18:00 End of meeting.

The slides of the talks (most of them in English) are linked on the meeting website: guitex.org/home/en/guit-meeting-2025.

3 Hands-on L^AT_EX: five practical parts

The morning blocks were an approachable yet thorough walk through core topics that newcomers and seasoned users alike appreciate when (re)discovering L^AT_EX today.

Part I: General introduction and mathematics.

Massimiliano Dominici contrasted word processors with L^AT_EX's compile-view workflow and stressed semantic document structure: a clean preamble, logical sectioning, labels/refs, indexing with `imakeidx`, multilingual support via `babel`, and disciplined use of floats. For mathematics he highlighted inline vs. display math, alignment and numbering with `amsmath` (e.g., `align`), and encouraged defining small, reusable macros (`\newcommand`, `\DeclareMathOperator`) to keep notation consistent. Building on long-standing guidance, the session also echoed ISO-oriented conventions for numbers and units (`siunitx`) and the preferred treatment of the differential d (or d) in integrals.

Attribution. The talk was given by Massimiliano Dominici. The first slide deck was authored by Gianluca Pignalberi and Massimiliano Dominici; the second set (from 2019) was prepared by Enrico Gregorio.

Part II: Graphics and bibliographies. Massimiliano Dominici framed two pillars of scholarly documents: clear graphics and structured references. For graphics he contrasted external, vector-first workflows with programmatic drawing inside L^AT_EX. A practical route is *Inkscape* plus the `TeXText` plug-in: draw in SVG, insert L^AT_EX labels and equations that use your document's fonts, export to PDF/SVG and include with `graphicx`. For in-document drawing, he surveyed the spectrum from low-level `picture/pict2e` and PostScript-based `PSTricks` to `PGF/TikZ`, highlighting readable path syntax, node placement, and the `PGFplots` stack for data-to-figure plots that compile reproducibly.

For bibliographies he mapped the choices: a manual `thebibliography` environment for short lists; classic `BIBTEX` databases; and modern `biblatex+biber` for Unicode-aware, publisher-specific styling. The emphasis was on keeping references as *structured metadata* (`.bib` files with fields such as `author`, `title`, `doi`, `url`) and letting tools handle sorting, language-aware punctuation, and cross-references (such as `BIBLATEX`'s `@xdata` and related). Typical builds invoke `pdflatex` → `biber` → `pdflatex` twice. He showed how small option sets (for example, `backend=biber`, `style=authoryear` with selective field clearing) deliver consistent output across venues. Finally, he noted that the same `.bib` archives power non-L^AT_EX pipelines (e.g., via `Pandoc`), and that

helper tools like `zblbuild` can reduce configuration friction for newcomers.

Attribution. The slides for this part drew on material by Agostino De Marco (graphics) and Guido Milanese (bibliographies), originally presented at the 2019 GuIT meeting in Turin.

Part III: Presentations with Beamer. Beamer remains a community staple. Drawing on their 2019 tutorial, Grazia Messineo and Salvatore Vassallo focused on designing clear, well-timed slides: avoid dark gradients and overcrowded frames; plan realistically; aim for “one slide = one idea”. They reviewed core structure (`\documentclass{beamer}`), sections driving navigation, frames with titles/subtitles and several modes for different outputs (`beamer`, `handout`, `article`). Themes were presented with restraint (outer/inner, colour, font). Beamer's overlays power progressive disclosure — `\pause`, `\onslide<.>`, `\only<.>`, `\uncover<.>`, and `\alert<.>` — to pace explanations without clutter; transparency can be tuned via `\setbeamercovered{transparent=...}`. For code and listings, they noted that verbatim causes *fragile* frames. Images are included as usual (`\includegraphics`) and can themselves follow overlay specifications. A brief note on fonts (e.g., using `\usefonttheme{professionalfonts}` with `lxfonts`) reinforced typography consistency. The overarching message matched the day's theme: let semantics, pacing, and simplicity carry the talk; use effects sparingly to serve clarity.

Part IV: Using L^AT_EX online. Cloud editors lower the entry barrier for teaching and teamwork: no local installation, usable from any device, and real-time collaboration. In this session Salvatore Vassallo focused on *Overleaf* and *CoCalc* as accessible starting points — both with free tiers suitable for courses and light projects.

Overleaf is the most widely used online L^AT_EX platform: integrated PDF preview, comments, version history, and easy sharing (including Git). The free plan permits one collaborator per project. Its simplicity is also its main constraint: projects are essentially a flat folder, and customization beyond the document preamble is limited.

CoCalc, by contrast, is a full GNU/Linux environment in the browser that happens to include a L^AT_EX editor alongside R, SageMath, Octave, Python, and more. Even on the free plan (with limited CPU/RAM), it allows multiple collaborators and enables workflows closer to a local setup: a personal `texmf` tree, custom config files, separate image/data

folders, and even terminal compiles with specific options. The interface offers Italian localization, and optional desktop apps can connect to your projects.

Across both platforms the advice was pragmatic: use shared templates, bibliography tools, and sensible compile settings; and keep local and cloud projects synchronized to preserve reproducibility.

Part V: Typesetting a PhD thesis. Here the spotlight was on the TOPtesi bundle by Claudio Beccari—a well-documented set of classes and extensions for bachelor, master, and PhD theses (slides by Beccari; session presented by Massimiliano Dominici). The name TOP stands for *Technical Institute of Turin* (Politecnico di Torino/Torino Politecnico) and “tesi” is Italian for “thesis”. In practice you load it with `\documentclass[...]{toptesi}` and pick concise options: body size (`corpo=10pt` by default), duplex printing (`twoside`), the thesis type (`tipotesi=doctoral` or Politecnico’s `scudo`), and an optional `mybibliography` flag if you plan to override the default numeric style. The class takes care of institutional identity (title pages, logos, metadata hooks) and keeps students on rails. We walked through a typical structure—front matter (title page, ToC/LoF/LoT), main matter (chapters and appendices), and back matter (bibliography, nomenclature, indexes)—with a reminder that `\includeonly` and subfolders help compile large projects quickly. Indexing is simple because `imakeidx` is already in place: declare indexes with `\makeindex`, sprinkle `\index` entries, and let the tool run. Bibliographies use a `.bib` database processed by `biber`; the default is a numeric “IEEE-like” style, and `mybibliography` opens the door to custom styles when needed.

PDF/A and metadata. Beccari emphasized output that is archivable. One robust path is `pdfx` (with a small `pdfxmetadata` block set *before* loading it) to produce PDF/A-2b. Erik noted he used this for years, but nowadays prefers `\DocumentMetadata` to target PDF/A-4f. `pdfx` still earns its keep, though, because a few `hyperref`/viewer quirks (e.g., how copyright and metadata display) sometimes call for its tooling. Either way, plan the metadata early and keep it versioned alongside your sources for reproducibility.

Student’s view. TOPtesi felt refreshingly straightforward: it bundles the right defaults (math packages, indexing, hyperlinks) so students can focus on content instead of yak-shaving. In hindsight, Erik wished he had this overview when he was studying; it would have saved time chasing package combinations (`imakeidx`, bibliography options, and friends).

4 Afternoon: creativity, data, and alternatives

After lunch, we enjoyed *Parallel poems* (Claudio Beccari and Cecilia Benassi), a humanities-flavoured demonstration of setting bilingual or multilingual poetry in parallel columns. They introduced a compact environment, `ParallelStanza`, that pairs an original on the left with a translation on the right, each as an independent typographic box. Each stanza pair is treated as a non-breakable unit (much like a displayed equation), and you repeat the environment for as many pairs as the poem needs. A brief example from Homer’s *Iliad* (Greek–Italian) showed how optical alignment can be preserved even when lines are not in one-to-one correspondence.

Practical tips: keep the page `\raggedbottom` to avoid vertical stretching; if the next pair will not fit, force a `\newpage` rather than allow uneven whitespace; try to keep each bilingual pair together on one page. On fonts and engines, the presenters noted Beamer’s default T1 setup limits some scripts (e.g., Cyrillic/Romanian extensions); for broader coverage, use a Unicode engine with `fontspec` and appropriate language support via `babel` or `polyglossia`.

Symbolic links. Claudio Beccari showed how symbolic links (*symlinks*) can simplify shared resources across many L^AT_EX projects—especially a common bibliography. A symlink is a filesystem entry that *points* to a real file: it looks like a local file, but edits affect the single master copy. On Unix-like systems they are created with `ln -s <target> <link>`; on Windows, with administrative privileges, use `mklink <link> <target>`. Create the link *in the project folder that should contain* the virtual file; the target path can be relative (e.g., `../database.bib`) or absolute.

Benefits: one master database, no duplication, consistent citations everywhere. In practice, keep a single `database.bib` and, inside each document folder, make a link (any name ending in `.bib`) to that file; recompiling picks up updates automatically. Good habits still matter: ensure unique citation keys, avoid duplicate records, and use `\nocite{*}` sparingly. Beccari also recommended automation with `latexmk`-style engines so T_EX and Biber/BIBT_EX rerun as needed until cross-references settle. This linking approach complements the alternative we discussed over dinner, namely configuring `TEXMFHOME` when juggling multiple private trees (see below).

The discussion continued with practical alternatives. Erik noted that while a single symlink can solve a quick setup, when working with multiple personal or project trees it is often cleaner to configure your T_EX search path via environment variables—

in particular `TEXMFHOME`. By setting `TEXMFHOME` you can keep private styles, classes, and bib files in separate trees and switch contexts without touching the system tree. This keeps setups portable across machines and reduces surprises when moving between projects. Erik previously demonstrated this workflow in India and documented it in the proceedings [3]. In later discussion, Massimiliano also noted that a suitably configured `TEXMFLOCAL` tree can play a similar role for shared resources, avoiding the need for multi-entry `TEXMFHOME` paths when several users or projects rely on the same custom material.

Linear regression. Battista Benciolini presented the mathematics and a practical implementation of linear regression á la \LaTeX , released as the package `linearregression` (CTAN; first released June 2024, updated December 2024). The aim is to perform the entire workflow—data ingestion, computation, and plotting—inside \LaTeX . Most of the computations require only \LaTeX 3 syntax.

The package allows plotting as well as computing the parameters of three common “best fit” lines approximating a set of 2D points. This approximation is called linear regression and Battista pointed out that this problem was treated by Karl Pearson in his article “On Lines and Planes of Closest Fit to Systems of Points in Space” in 1901 [4].

There are three common cases: classical regression (minimizing vertical errors), inverse regression (swap axes, which means minimizing horizontal errors), and symmetric/total regression (minimizing orthogonal distances). In his talk, Battista presented the package together with the math, which in the first two cases is quite simple but in the symmetric case is somewhat extensive. We refer the reader to the package documentation or the slides of the talk which can be found online [2].

The pipeline is run with a handful of commands: `\lrfilename` selects a file, `\lrcomputation` computes moments and estimates, `\lrplotparameters` tunes the graphic, `\lrplot` draws the point cloud and/or fitted lines (see Figure 1), and `\lrprint` emits a small summary table (see Table 1).

The code is written in \LaTeX 3 and interleaves reading and computation to keep memory usage modest. Current limitations include sparse diagnostics for degenerate cases and limited integration with larger PGF/TikZ figures, but it adheres closely to modern \LaTeX 3 practices and works well as a pedagogical example.

The session had the flavour of a friendly contest: how far can one go inside \LaTeX alone? It was fun and instructive; nonetheless, Doris remarked that

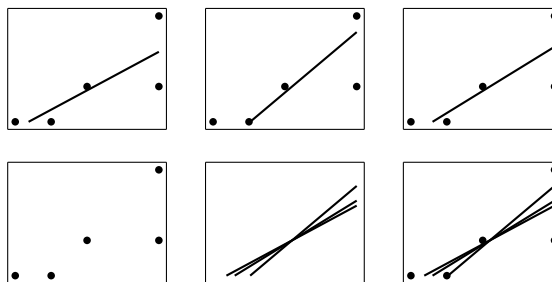


Figure 1: Linear regression example from the package `linearregression`. First row: classical (first), inverted classical (second) and symmetric case (third). Second row: variants with points only, all three lines only, points together with all three lines.

Table 1: Left: data. Right: the result of `\lrprint`.

| | | |
|-----|---|---------------|
| 1 3 | Data File | data.txt |
| 2 3 | Number of points | 5 |
| 3 4 | Mean values of the coordinates | 3.2 |
| 5 4 | Minimum values of the coordinates | 1 |
| 5 6 | Maximum values of the coordinates | 5 |
| | | 6 |
| | Second order moments | abscissa 2.56 |
| | | mixed 1.4 |
| | | ordinate 1.2 |
| | Slope and intercept of optimal line (estimated with errors in ordinate) | 0.5468 |
| | | 2.25 |
| | Slope and intercept of optimal line (estimated with errors in abscissa) | 0.8571 |
| | | 1.2571 |
| | Unit vector along the line | 0.8476 |
| | | 0.5306 |
| | Slope and intercept of optimal line (estimated with symmetric regression) | 0.626 |
| | | 1.9967 |

for this kind of numerical analysis a few lines in Python (`numpy/pandas` or `scikit-learn/pytorch`) are often quicker, with \LaTeX then shining as the place to explain and present the results.

Typst: A new alternative to \LaTeX . Roberto Giacomelli gave a compact tour of *Typst*, a modern typesetting system written in Rust and positioned as a user-friendly, programmable alternative to \LaTeX . Born from a 2019 effort to rethink the pipeline from scratch, Typst runs locally or on the web app and produces PDF, HTML, SVG, and PNG from UTF-8 `.typ` sources with fast, incremental compilation. The design blends Markdown-like brevity with structured document concepts and a clear internal scripting model.

Markup essentials: Typst distinguishes content [...], Typst code introduced with #, and maths delimited by `$. . . $/$$. . . $$`; these modes mix freely, which makes small computations inline natural (for example, `$$\sqrt{2} \approx \#calc.\sqrt{2}$$`).

Variables and functions are first-class via `#let` (e.g., `#let box = rect(fil:lblue)[...]; #box`), and global styles can be tuned dynamically with the `#set` rule (think key–value options with scoping). Small examples shown included generating a times table with `table(...)` and contrasting imperative vs. functional factorial definitions; the point was not the algorithms, but that document logic and content live together.

Workflow and ecosystem: With VS Code + Tiny-mist (or the browser app), authors see live updates as they type—useful in teaching. Packages and templates are fetched from the built-in *Universe* repository on demand (for instance, initializing a slides project from a template is a one-liner). The documentation is thorough and error messages are friendly, lowering the first hurdle for newcomers while keeping enough depth for power users. The discussion that followed was practical.

Our shared conclusion: it is healthy for the ecosystem to explore new tools, while \TeX continues to shine in long-form, mathematically rich documents and archival stability. Many participants expressed curiosity about ideas we might learn from each other.

5 Conversations with GuIT

We kept things informal and mostly listened between sessions. It was a joy to see people learning and teaching \LaTeX in Italian, and to watch practical examples spark questions across the room. A recurring theme was simply sharing slide decks and example sources after the meeting so others can reuse them in classrooms.

6 Takeaways

Two small, practical things we would like to do more of after Pisa:

1. Provide minimal, well-commented starter templates for common tasks (assignments, theses, posters).
2. Share slide decks and sample sources promptly so others can learn and build on them.

7 Grazie mille, Pisa!

Our thanks to GuIT for the warm welcome and kind hospitality, to all speakers for clear, useful talks, and to the participants whose questions made the day lively. We visited on our own initiative and were happy to be received so kindly. We look forward to future collaborations between our communities.



Figure 2: Dinner after the meeting in Pisa.

From front right, counter-clockwise: Doris Behrendt, Grazia Messineo, Erik Nijenhuis, Salvatore Vassallo, Massimiliano Dominici, Silvia Cazzato, Roberto Giacomelli, Battista Benciolini.

References

- [1] D. Behrendt. Tagungsbericht GuIT meeting. *Die \TeX nische Komödie*, (1):8–13, 2020. archiv.dante.de/DTK/PDF/komoedie_2020_1.pdf
- [2] B. Benciolini. La regressione lineare in \LaTeX , 2025. www.guitex.org/home/images/meeting2025/BencioPisa2025c.pdf
- [3] E. Nijenhuis. Xerdi’s documentation project: Intertwined documents. *TUGboat* 46(2), 2025. tug.org/TUGboat/tb46-2/tb143nijenhuis-doceng.html
- [4] K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

◇ Doris Behrendt, Erik Nijenhuis
 \TeX Users Group
tug.org