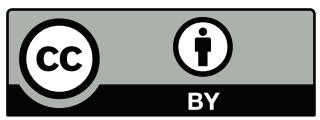
*TEX& gnuplot cookbook

Daniel Torregrosa

daniel.torregrosa @ insight-centre.org

5th June 2019



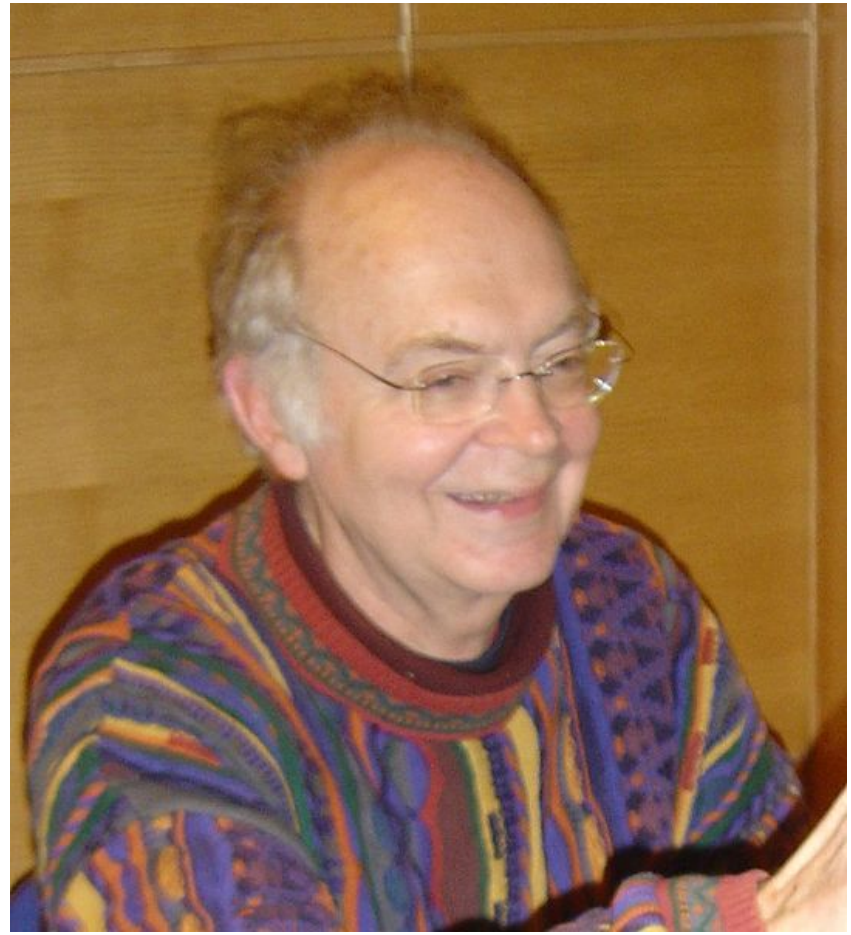
If you find any mistakes, or want to add something to the slides, feel free to get in touch with me.

Contents

Disclaimer

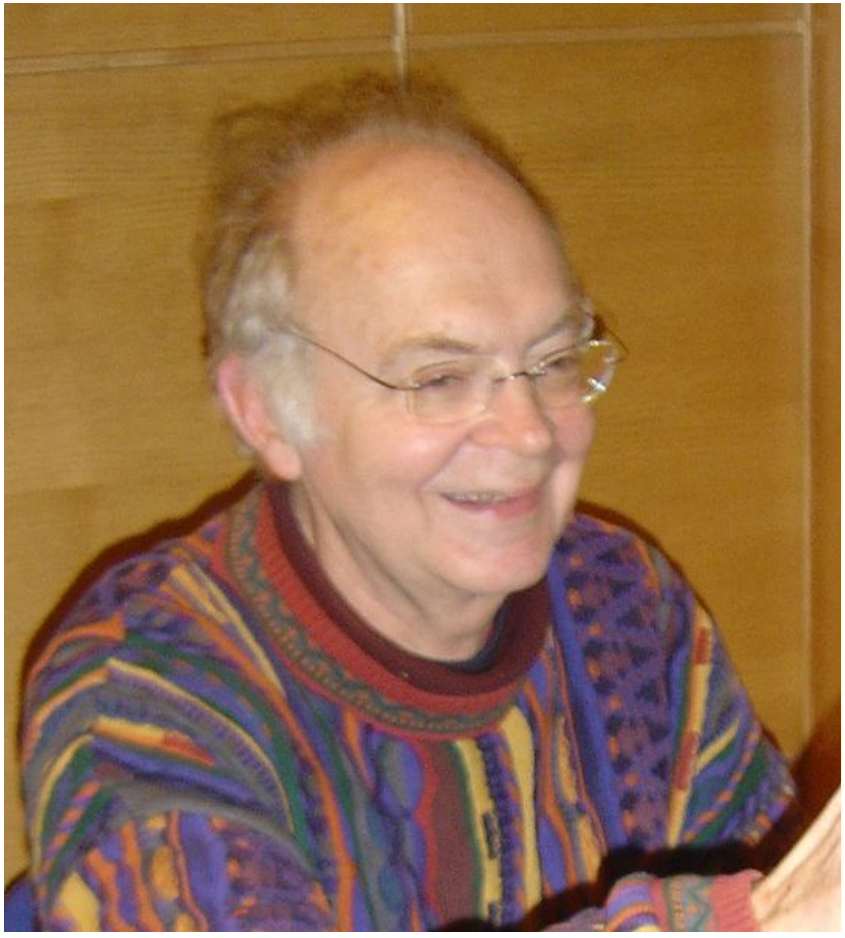
- The main objective is to show what TEX and `gnuplot` are capable of
- Keep the slides as reference for later
- Ask at any time

Outline



https://fr.wikipedia.org/wiki/Fichier:Donald_Knuth_DSC00624.jpg

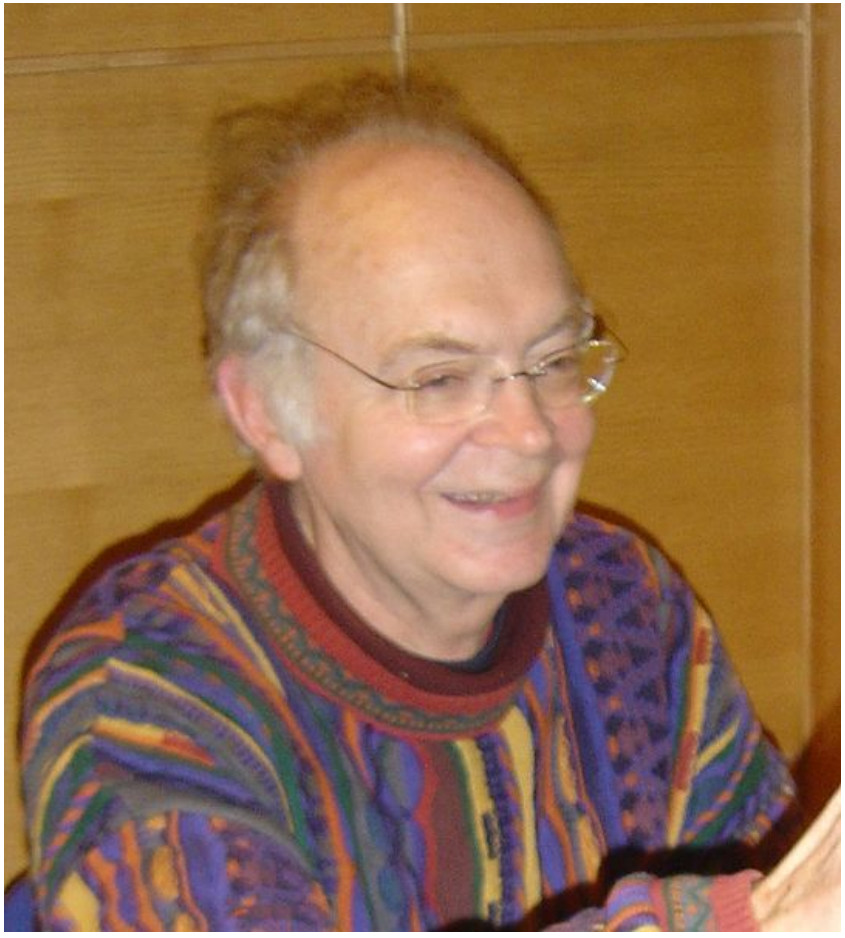
高德納 (Gāo dé nà)



https://fr.wikipedia.org/wiki/Fichier:Donald_Knuth_DSC00624.jpg

Donal Knuth

- Writer of *The Art of Computer Programming*
- Popularised big-O notation
- Defined literary programming



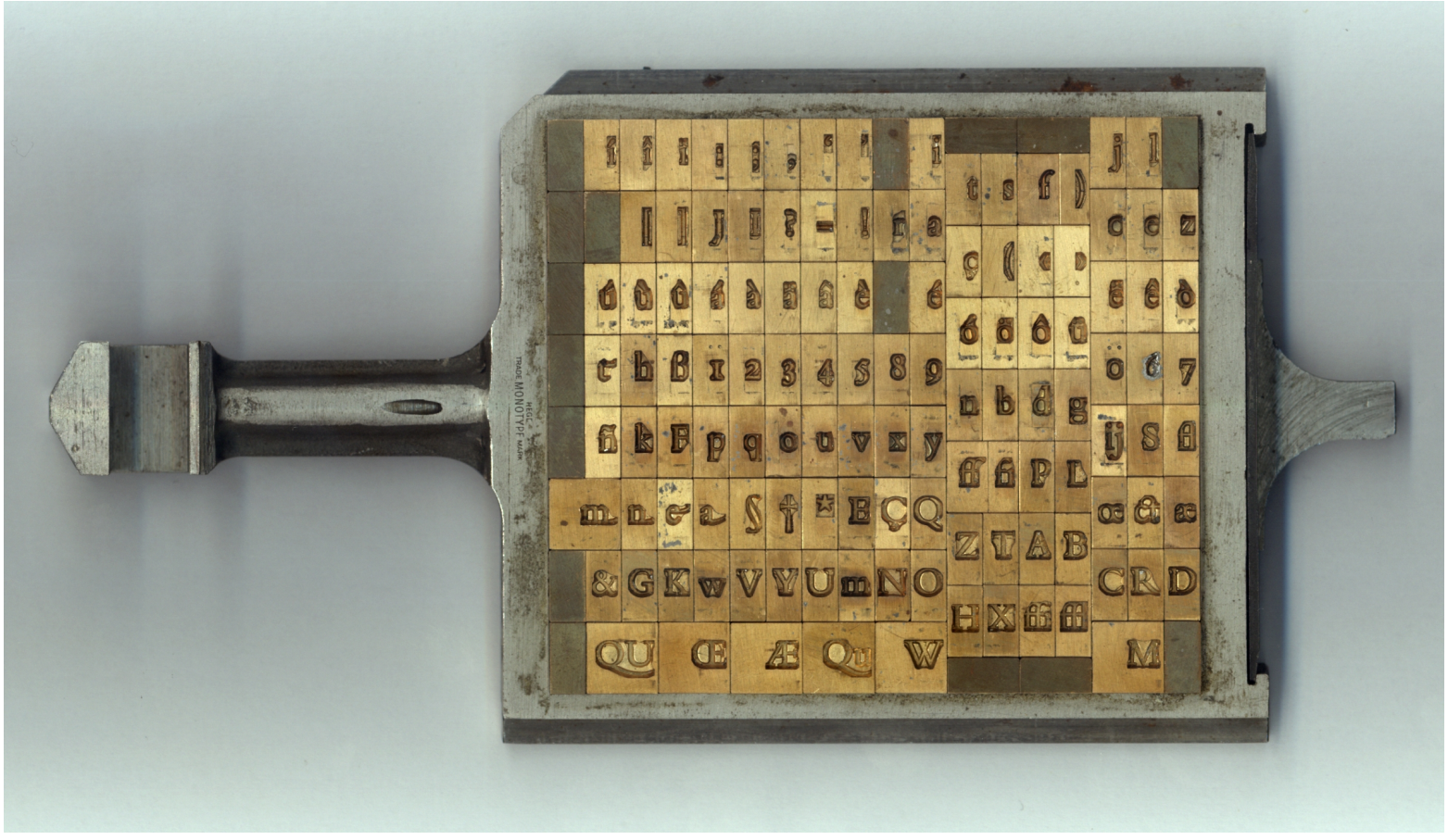
https://fr.wikipedia.org/wiki/Fichier:Donald_Knuth_DSC00624.jpg

The Art of Computer Programming

- The first volume of *The Art of Computer Programming* was published in 1968 using hot metal typesetting

The Art of Computer Programming

- The first volume of *The Art of Computer Programming* was published in 1968 using hot metal typesetting



The Art of Computer Programming

- The first volume of *The Art of Computer Programming* was published in 1968 using hot metal typesetting
- In 1977, Knuth found the phototypeset galley proofs for the second volume inferior, as hot metal typesetting had fell out of use
- Unhappy with both typesetting and fonts, he …

TEX history

- In 1977, Knuth writes a memo describing TEX
- In 1978, a first version of TEX was implemented in `SAIL`
- In 1982, TEX82 (sometimes TEX `2.0`) became Turing complete
- In 1984, the TEXbook instructs the reader to pronounce TEX as /tɜx/, from τέχνη
  - "It's the *ch* sound in Scottish words like *loch* or German words like *ach*; it's a Spanish *j* and a Russian *kh* [X]" *Donal Knuth, TEXbook, 1984*
- In 1989, TEX `3.0` became *ready to use* (feature frozen)

TeX 3.0 features

- Turing complete
- Macro and token based language
- Expansion of macros is almost side-effect free
- Tail recursion makes it very efficient
- Written in WEB, that combines TeX and a subset of Pascal
  - See *TeX: The Program*

METAFONT

- Makes fonts from strokes with finite-width pens and filled regions
- Computer Modern is the most famous example (also the default font in TeX)
- Produces typefaces (rasterised glyph)
- Mostly superseded by vector-based font systems (e.g. Postscript, TrueType, OpenType)
- "…asking an artist to become enough of a mathematician to understand how to write a font with 60 parameters is too much." *Donald E. Knuth, 1996*

Other trivia

- The *Device independent file format* (DVI) was designed by David Fuchs and implemented by Knuth;
  - TEX outputs `.dvi` files
  - Mostly superseded by postscript, a turing-complete stack-based format developed by Adobe to communicate with printers

- "At the time of my death, it is my intention that the then-current versions […] should become TEX, Version π and METAFONT, Version e, respectively. From that moment on, all 'bugs' will be permanent 'features.'"
  Donal Knuth, The future of TEX and METAFONT, 1990

Outline

Why *TeX

- Content/presentation split
- Typesetting math
- Automatising: no need to typeset half the document because you added a new figure
- Reference management via BibTeX
- Design: can you compete with the combined contributions of hundreds of developers extremely passionate about design over 30 years?

Why not *TeX

- WYSIWYG editors require almost no skills or knowledge, producing *good enough* output
- Hard to find answers to some problems because they can be easily done in an incompatible way (specially XeTeX and LuaTeX specific solutions)
- 30 years of development left a lot of waste
- Sometimes LaTeXis very obtuse in the way it works
  - `Overfull \hbox (21.67pt too wide)`
  - Position of floats
  - Use of `\fragile` and `\robust`
  - `\makeatletter`

Corollary

- TEX is a typesetting program, not a program for typesetting

- WYSIWYG editors are programs for typesetting
  - Arguably, libreoffice-like editors are quite bad at it
  - inkscape-like programs should be used for proper typesetting

Outline

TEX

- TEX undergone a feature freeze in 1989
- TEX is a typesetting *engine* with primitives
- TEX is also a binary that takes TEX files and outputs `.dvi`
- Over the years, different engines were developed:
  - ε-TEX, that improved TEX significantly
  - pdfTEX, that can output both `dvi` and `pdf` formats
  - X∃TEX, with support for unicode and modern font formats(`otf`). Results are mostly faithful to ε-TEX
  - LuaTEX, that exposes a secondary Lua interface for building macros. There are significant changes in how different elements are rendered (e.g. hyphenation, ligatures, etc.)

See https://tex.stackexchange.com/questions/222286/what-are-the-incompatibilities-of-pdftex-xetex-and-luatex

TEX

- TEX undergone a feature freeze in 1989
- TEX is a typesetting *engine* with primitives
- TEX is also a binary that takes TEX files and outputs `.dvi`
- Over the years, different engines were developed:
  - ε-TEX, that improved TEX significantly
  - pdfTEX, that can output both `dvi` and `pdf` formats
  - XƎTEX (/ˈziːtɜx/), with support for unicode and modern font formats(`otf`). Results are mostly faithful to ε-TEX
  - LuaTEX, that exposes a secondary Lua interface for building macros. There are significant changes in how different elements are rendered (e.g. hyphenation, ligatures, etc.)

See https://tex.stackexchange.com/questions/222286/what-are-the-incompatibilities-of-pdftex-xetex-and-luatex

Distribution files

- Distribution files (`ltx` extension) provide several macros to ease working with TEX
- Plain TEX is the simplest one
- LATEX, started by Leslie Lamport, is an enhanced collection of macros (*document preparation system*), and the most used one (currently LATEX 2_ε)
- ConTEXt was developed concurrently with LuaTEX
- This document is typeset with XELATEX (XETEX engine with LATEX 2_ε macros)

Classes and packages

- Class
  - `.cls` extension
  - All documents should have exactly 1 class, selected using `documentclass`
- Package
  - `.sty` extension
  - Any number of packages can be included using `usepackage`
- Distribution (`.ltx`), classes (`.cls`) and packages (`.sty`) are all TEX format files

Document classes

- `article` for articles
- `book` for books
- `beamer` for slides
- Hundreds more `https://ctan.org/topic/class`
- Each one defines layouts, macros, fonts, etc.

Packages

- Packages further extend LaTeX functionality
- Several packages to be explored later

Primitives

- Tokens: just about anything not starting with \
- Commands: anything starting with \
- ▶ More on \

Toggles vs macro vs environment

- There is no real distinction between these commands!
- Toggle: `\centering`, `{\bfseries text in bold}`
- Macro: `\item`, `\textbf{text in bold}`
- Environment: `\begin{frame}`,

```
\newenvironment{boldenv}%
{\bfseries}%
{}
...
\begin{boldenv} text in bold \end{boldenv}
```

(not by default in LaTeX, ▶ Defining macros)

Environments

- Basically scopes
- {} delimits an environment
- From *outside*, an {} delimited block is just a token
- `\begin{myEnv}`... `\end{myEnv}` implicitly creates an environment:
  - A macro with the same name (`myEnv`) that can take parameters
  - A scoped block (the content between begin and end)
  - A macro with the same name starting with end (`endmyEnv`)
- E.g. you can call `\center`, that is called where `\begin{center}` is used, but will likely break the rest of the document

Dealing with hungry macros

- Macros usually eat the next token (usually, a space)
- LaTeX macros also eat blocks enclosed in [] (optional parameters)
  - That is, those defined with `\newcommand` instead `\let` or `\def`
- Sometimes {} is enough to stop expansion, e.g. `\TeX{}`
- The correct way to stop macro expansion is using `\relax`

Character category codes

0	Escape	\	1	Begin group	{
2	End group	}	3	Math shift	$
4	Alignment	&	5	End-of-line	\n
6	Parameter	#	7	Superscript	^
8	Subscript	_	9	Ignored	
10	Space	%20, \t	11	Letters	abc
12	Other	1.+	13	Active character	~
14	Comment	%	15	Invalid input	`[DEL]`

Other character

- Both code 11 (letters) and 12 (other) get rendered when part of a token
- But, macro names can only be composed of letters
- TEX has no namespacing mechanism
- Hence, we can make *protected* macros by temporarily telling TEX that a certain *other* is now a *letter*, e.g.

```
\makeatletter
\newcommand{@myprotectedcommand}{...}
\makeatother
...
\@myprotectedcommand

[Massive error log due to writing something unparseable]
```

Active character

- Active characters are considered commands (just like any other escaped sequence of letters)
- The only default active character is ~, that is a non breaking space
- _ and ^ seem but are not active, they have its own category (7 and 8)!
- You can play with this, but it will bite

Variables

- LaTeX defines some variables, e.g. for keeping track of counters

- Some documents or packages define variables such as `author`, `title`, `subtitle`, etc.

“Variables”

- LaTeX defines some variables, e.g. for keeping track of counters
- Some documents or packages define macros such as `author`, `title`, `subtitle`, etc.
- Actually, TeX does not support variables at all!
- Done with macros, e.g. `acknowledgement`

  ▶ defined in this document

Other common caveats

- Should escape: # $ % & _ { }
- Should use `text*` command: \ ~ ^
- `\^` `\~` add accents to the next letter
- < > should be typed `\textless\textgreater`
  - unless your engine can use `UTF-8`, e.g. XƎTEX
  - beamer adds special parameters enclosed in <> for some macros
- `‘’` or `‘‘’’` for proper “quotation”! `"` will break documents
- Likewise, `\cdots` for … (do not ...)
- `--` for en–dash, `---` for em—dash

Compilation times

- LaTeX compilation is optimised by cutting some corners, “hacky” commands (e.g. `\verb`) should only be used in `fragile` blocks (that will take longer to compile)
- Macros are better defined before the start of the document (or in fragile blocks)
- Robust macros (`\robust\bfseries`) do not need to be protected with fragile (but will take longer to compile)
- Adding `[draft]` to the document class will make compilation faster, but images are placeholders, links do not work, etc.
- Multiple compilations: LaTeX should be invoked multiple times, with (likely) invocations of `bibtex` and other libraries in between. Use `latexmk`, `xelatexmk`, etc. for best results

Whitespace is fun

- One or more spaces will be considered a word delimiter
- One line break will be considered a word delimiter too!
- Two line breaks will actually break the line

```
\begin{itemize}
\item␣One␣or␣more␣␣␣spaces
will␣be␣considered
a␣word␣delimiter
\item␣One␣line␣break
will␣be␣considered
a␣word␣delimiter␣too!
\item␣Two␣line␣breaks

will␣actually␣break␣the␣line
\end{itemize}
␣␣␣␣
```

Whitespace is fun II

- ~ (active character!) represents a non-breaking space
  - E.g. `Section~\ref{label}` will never put the number in the next line
- `\\` forces a line break
- `\break` breaks the line without filling it (usually results in bad typesetting)
- `\clearpage` and `\newpage` force a page break
  - Some styles (e.g. book) offer a `\newoddpage` command
- Spaces (`\vspace{10pt}`, `\hspace{2ex plus 1ex minus 1ex}`) add a fixed space
- Skips are predefined spaces, e.g. `\smallskip`
- Fills (`\vfill`, `\hfill`) take all the free space

Sectioning

- LaTeX has several sectioning depths:
  -1. `\part`
  0. `\chapter`
  1. `\section`
  2. `\subsection`
  3. `\subsubsection`
  4. `\paragraph`
  5. `\subparagraph`
- Some only available in some classes, e.g. -1 and 0 only available in `book`
- Package `titlesec` can be used to configure how are they rendered
- Adding an * to the macro will create an unnumbered section that will not appear in the table of contents (e.g. `\section*`)
- `\tableofcontents` inserts the table of contents (and can be configured)

Macro*

- Many packages offer macro variants that end in *, e.g.
  - `section*`, a `section` without number
  - `figure*`, a page-width figure in two-column documents
  - `caption*`, a figure caption that does not start with **Figure 4:**

- But, those are macros manually defined by the different packages

- The * has no specific meaning, do not assume macro* always exist

Modularity

- `\input` *copies and pastes* the content of a different file in this position
- `\include` is similar, but recursive calls are forbidden and forces a page break
- `\includeonly` before `\include` can limit the includes (for faster compilation times)

Comments

- The character % (character code 14) makes the parser ignore the rest of the line
- Beware in math mode! Incorrectly typed 100% may break the document
- Newline can be ignored writing % at the end of the line; this is sometimes needed when defining macros

Lists

- `itemize` for unnumbered lists
- `enumerate` for numbered lists
- `\renewcommand{\theenumi}{\Roman{enumi}}%` can be used for roman numbers
  - Also `arabic` (default), `roman`, `alph` and `Alph`

→ bullets can be replaced with (almost) anything

- `description` for better management of label bullets
  - Optional `align` parameter to align labels

Font sizes

- LaTeX offers several pre-defined font sizes

  `\Huge` `\huge` `\LARGE` `\Large`
  `\large` `\normalsize` `\small` `\footnotesize`
  `\scriptsize` `\tiny`

- Other packages can add more, e.g. `beamer`'s `\VERYHUGE`
- Whole document *base* font size can be altered, e.g. `\documentclass[11pt]{article}`
- `\fontsize{size}{baselineskip}\selectfont` to choose an arbitrary size
- Package `anyfontsize` can be used for arbitrary sizes
  - Raster fonts might not have all sizes
  - Vector fonts can be scaled arbitrarily

Font faces

- Usually, several predefined styles exist, e.g.
  - Bold: `\textbf{}`, `\bfseries`
  - Italics: `\textit{}`, `\itshape`, `\emph` (nested `\emph` toggle italics)
  - Monospace (typewriter): `\texttt{}`
- Do not use `\bf`, `\it`, etc. those are deprecated and do not play nicely with each other
- **bold** *italics* **bold** ***italics***
- All these predefined macros use `\fontfamily{family}\selectfont`
- `http://www.tug.dk/FontCatalogue/`
- ▶ Recipe for adding new fonts (such as Ubuntu in this slides)

Self referencing

- Numbered environments such as sections, floats, etc. can be labelled using `\label{name}`
- You can cross-reference using `\ref{name}` at any other point of the document (even before the label!)
- Package `hyperref` adds `autoref`, that also adds a clickable link, and `nameref`, that will add the closest name in the outline, e.g. `\autoref{labelling}` produces 40 and `\nameref{labelling}` produces Basics (the name of the section)
- Package `fancyref` adds `fref`, that, if using the correct naming schema, will automatise *context information*, e.g. `\fref{sec:basics}` produces section 3 on page 20

Math

- `\(\pi+\frac{w_{\{x,y\}}}{i^{2e}}\)` can be used for inline math $\pi + \frac{w_{\{x,y\}}}{i^{2e}}$

- `\[\pi+\frac{w_{\{x,y\}}}{i^{2e}}\]` can be used for displayed math

$$\pi + \frac{w_{\{x,y\}}}{i^{2e}}$$

- TEX uses `$x$` and `$$x$$` for inline and display modes respectively; the latter is not supported in LATEX and the former will produce more obscure error reports if something goes wrong

Math II

- `\mathrm` for math roman, e.g. $\mathrm{probability}(x)$
- `\mathcal` for math calligraphic, e.g. $\mathcal{P}(x)$
- Brackets autoresize (with some help)!

$$\left\{ \begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 6 & 7 & 8 \end{array} \right)$$

```
\[
\left \{ \begin{tabular}{ccc}
1&2&3\\
4&5&6\\
6&7&8\end{tabular} \right )
\]
```

- `\begin{equation}` can be used for labelled displayed math that can be used for self references

Floats

- `figure` and `table` environments are *floats*
- Floats will never be rendered before they appear in the `.tex` file
- When a float is encountered, it is evaluated for placement; if it fails (and it usually does), it will be queued
- At page end, all queued floats are evaluated
- At document end (or `clearpage`) all floats are printed regardless

Float positions

- Floats can be requested to be placed:
  - where they appear with `\begin{figure}[h]`
  - at top or bottom of the page with `\begin{figure}[tb]`
  - at a special float page with `\begin{figure}[p]`. This option is incompatible with the rest, and the float will always appear at the next page (unless it does not fit!)

- `!` can be used to override LaTeX parameters for good float positions (`[h!]`)

Float headache

- The most common problems with floats arise from too big floats
  - A too big float never fits, hence never leaves the queue
  - Once 18 (in LaTeX) floats are queued, compilation fails
  - If document end of `clearpage` are reached, all floats are printed together
  - Check `resizebox` package (and use on this presentation) for tips on how to automatically fit floats

- The second most common problem with floats appears when too many floats are clumped together
  - Do not try to overpower LaTeX using `[h!]` everywhere
  - Instead, write floats before they are referenced (or better, `include` them for clarity!)

Float captions

- Floats are always numbered
- The number in the caption can be hidden using the `\caption*{}` command instead
- Floats can be labelled for self referencing
- Counters `\thefigure` and `\thetable` keeps the number of figures and tables respectively

Figure

- A float whose caption will start with *Figure*
- `includegraphics` for including raster and vector images
- Center with `\centering`
- `\begin{center} ... \end{center}` will add unnecesary vertical space!
- Format support for `includegraphics` depends on TEX flavour
  - `.eps` has widespread support
  - `.pdf` can be used with pdfLATEX or XELATEX
  - `.png` can be used with `graphix` package

Tables

- A float whose caption will start with *Table*
- Actual tables are created using
  `\begin{tabular}{spec}data\end{tabular}`
- `spec` is a list of column types
  - `c` for a centered column
  - `r` for a right-aligned column
  - `l` for a left-aligned column
  - `|` for a vertical strut
  - Packages can define new column types too

Tables II

- The data portion should have as many rows (delimited with \\) with the same number of columns as spec defines (delimited with &)

111	2	3
4	555	6
7	8	999

```
\begin{tabular}{r|c||l}
111 & 2 & 3 \\
4 & 555 & 6 \\ \hline
7 & 8 & 999 \\
\end{tabular}
```

Multirow and multicolumn

- `\multicolumn{number}{spec}{data}` creates a multi-column cell
  - `number` *discounts* &, that is, a three cell multi-column in a four column table will only require one more &
  - It cannot be larger than the number of columns left
  - `spec` can have only one of `lcr`, with optional `|` on each side

- `\multirow{number}{*}{text}` creates a multi-row cell
  - Unlike `multicolumn`, `multirow` does not *discount* & on other rows
  - The second field is the width; `*` will use the natural width of the cell
  - Requires the package `multirow`

- ▶ Table example

Widow and orphan lines

- `\widowpenalty10000` and `\clubpenalty10000` prevent widow and orphan lines
- 10000 is `other`, hence no {} needed!
- This will *only* be needed if a package broke something, TEX is great at preventing widow and orphan lines by default

Outline

`siunitx`

- SI units, pretty numbers, automatic rounding and padding, decimal number alignment in tables
- Highly configurable, e.g. `group-separator={,}`, `table-format=2.2`
- The macro `\num{}` can be used to typeset numbers
- Alternatively, `detect-all=true` will do its best to automatically typeset numbers
- The column `S` can be used in tables
  - Non-numeric columns have to be protected with {}
  - Automatic detection fails if the number has commas!

`booktabs`

- Adds `toprule`, `midrule`, `bottomrule` to tables, with `c` versions

- Top and bottom rules are not vertically centered

Table example

		BLEU			METEOR			chrF3		
Generic Dataset	SMT models	$M1_{eval}$	$M2_{eval}$	$Wiki_{eval}$	$M1_{eval}$	$M2_{eval}$	$Wiki_{eval}$	$M1_{eval}$	$M2_{eval}$	$Wiki_{eval}$
	Baseline	6.39	12.34	12.34	12.34	12.34	12.34	12.34	12.34	12.34
	$M1_{dev}$	**1.23**	12.34	12.34	**12.34**	12.34	12.34	12.34	12.34	12.34
	$M2_{dev}$	1.23	**12.34**	12.34	12.34	**12.34**	12.34	12.34	**47.70**	**12.34**
	$Wiki_{dev}$	1.23	1.23	**12.34**	12.34	12.34	**22.10**	**12.34**	12.34	12.34
	NMT models	$M1_{eval}$	$M2_{eval}$	$Wiki_{eval}$	$M1_{eval}$	$M2_{eval}$	$Wiki_{eval}$	$M1_{eval}$	$M2_{eval}$	$Wiki_{eval}$
	Baseline	1.23	1.23	8.20	1.23	12.34	12.34	12.34	12.34	12.34
	$M1_{dev}$	**12.34**	1.23	1.23	**12.34**	12.34	1.23	**37.00**	12.34	12.34
	$M2_{dev}$	1.23	**12.34**	1.23	12.34	**12.34**	12.34	20.90	**12.34**	12.34
	$Wiki_{dev}$	1.00	1.23	**12.34**	1.23	1.23	**12.34**	1.23	1.23	**12.34**
	NMT_{BPE} models	$M1_{eval}$	$M2_{eval}$	$Wiki_{eval}$	$M1_{eval}$	$M2_{eval}$	$Wiki_{eval}$	$M1_{eval}$	$M2_{eval}$	$Wiki_{eval}$
	Baseline	4.29	12.34	12.34	12.34	12.34	26.90	12.34	12.34	12.34
	$M1_{dev}$	**12.34**	1.23	1.23	**12.34**	12.34	12.34	**12.34**	12.34	12.34
	$M2_{dev}$	1.23	**12.34**	12.34	12.34	**12.34**	23.90	40.70	**12.34**	12.34
	$Wiki_{dev}$	1.23	1.23	**12.34**	1.23	1.23	**12.34**	12.34	12.34	**12.34**

Using `siunitx` , `booktabs` , `multicolumn` , `multirow` , `rotatebox` , `resizebox`

`tabularx`

- New table environment with a mandatory parameter (width) and new `X` column
- First, normal columns get assigned width as usual (i.e. enough to fit the largest cell)
- Then, `X` columns get an equivalent share of the remaining space

1	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do

`tabularx`

- New table environment with a mandatory parameter (width) and new `X` column
- First, normal columns get assigned width as usual (i.e. enough to fit the largest cell)
- Then, `X` columns get an equivalent share of the remaining space

1	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	Second	Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

`verbatim`

- Does not parse the contents
- Both environment `verbatim` and macro `verb`
- Needs `fragile`

`microtype`

- Several micro-typographic extensions
- Kerning, character protusion, font expansion...
- Not needed in XeLaTeX or LuaTeX

`todonotes`

- Easy TODO notes
- Create a `todomine` macro that will
  - Add text where the TODO is written to know where it starts exactly
  - Change the background to white
  - Make the text small
  - Add a TODO *header*

```
\newcommand{\todomine}[1]{\%Todo:\thetodoCounter\%
\todo[color=white]{\small\textbf%
{Me: \thetodoCounter } #1}}
...
\todomine{Improve wording here}
```

`graphicx` and `xcolor`

- `xcolor` adds several ways of managing colours (mixing, shades, colours by name, etc.)
  - `\color{green!40!red}`
  - `\color[wave]{485}`

- `graphix`
  - Slight overlap with `xcolor`, adds some colours by name
  - `rotatebox`, `scalebox`, `resizebox` macros
  - `includegraphics` as a replacement for `include` with trim, clip, scale, rotate options

listings

- Pretty print code

```
1 import numpy as np
2
3 def incmatrix(genl1,genl2):
4     m = len(genl1)
5     n = len(genl2)
6     M = None #to become the incidence matrix
7     VT = np.zeros((n*m,1), int)  #dummy variable
8     [...]
```

asmfonts / amssymb

- Extra math fonts or symbols
- http://milde.users.sourceforge.net/LUCR/Math/mathpackages/amssymb-symbols.pdf
- http://detexify.kirelabs.org/classify.html

hyperref

- Configurable urls
- Implicitly clickable cross-references (both `\ref` and `cite`)
- Also redefines `\url{URL}` as `\href{URL}{URL}`
- Use `\hypersetup{key=value,}` to configure, e.g.

```
\hypersetup{ #no box
    colorlinks  = true,
    urlcolor    = blue,
    linkcolor   = blue,
    citecolor   = red
}
```

```
\hypersetup{ #smallcap links
     frenchlinks    = true
}
\hypersetup{ #black links
    hidelinks       = true
}
```

`inputenc`

- `\usepackage[utf8]{inputenc}`
- Only for LaTeX (XeTeX refuses non-`UTF-8` documents)
- Using anything different from `UFT-8` will bite (and you deserve it)
- LaTeX defaults to `ASCII`

`fontenc`

- `\usepackage[T1]{fontenc}`
- Only for LaTeX (XeTeX uses `fontspec`)
- Required for proper hypenation/kerning
- `OTx` fonts use 128-bit fonts
- `Tx` fonts use 256-bit fonts
- This and `inputenc` are the #1 reason to move away from LaTeX (see Vietnamese use case)

Vietnamese example

- In both cases, use a font that supports Vietnamese, e.g. *Linux Libertine O*
- In XeLaTeX you need to
  - Declare the font `\newfontfamily\vietfont [Ligatures=TeX]{Linux Libertine O}`
  - `{\vietfont Phưng Hồng.}` = Phưng Hồng.
- In LaTeX you need to
  - `\usepackage[T5,T1]{fontenc}` respecting the order, otherwise `T5` will be default for the document
  - `H{\‘{\^o}}ng, Ph{\fontencoding{T5}\selectfont ư} {\fontencoding{T5}\selectfont ơ}ng` = Phưng Hồng
  - You will have to load the vietnamese-compatible `T5` font, likely have to create a `.map` for it (see `https://www.overleaf.com/latex/examples/example-custom-font/htswqdkhqxjk`)

Modifying a default family font with `fontspec`

- Include the fonts as `.ttf` files
- Use the `fontspec` package
- `Path` should have the relative path, the rest of parameters only take file names
- All parameters optional, but the main font, e.g. navy blue `Ubuntu mono` as mono font:

```
\setmonofont[Path = beamerthemeinsight/ubuntu/,
    BoldFont=UbuntuMono-B.ttf,
    ItalicFont=UbuntuMono-RI.ttf,
    BoldItalicFont=UbuntuMono-BI.ttf,
    Color={000080}
]{UbuntuMono-R.ttf}
```

Adding a custom family with `fontspec`

- e.g. adding a light *font family* for the *acknowledgements* part
- Defining parameters such as `BoldFont`, etc.
- Check package documentation for options:
  - Small caps, *slanted*, etc.
  - Different fonts for different sizes (e.g. Fraktur that becomes a Serif when under 8pt)

```
\newfontfamily\light[Path = beamerthemeinsight/ubuntu/,
Ligatures=TeX,
BoldFont=Ubuntu-R.ttf, % No bold light ttf
ItalicFont=Ubuntu-LI.ttf,
BoldItalicFont=Ubuntu-RI.ttf,
]{Ubuntu-L.ttf}
```

tikz

- Ti*k*Z (Ti*k*Z ist *kein* Zeichenprogramm) is a high-level language to produce vectorial graphics
- PGF is the low-level language
- Based in METAPOST (that is to METAFONT what XeTeX is to TeX)
- Has its own package manager (`\usetikzlibrary{}`)
- `http://www.texample.net/tikz/examples/` has a lot of examples

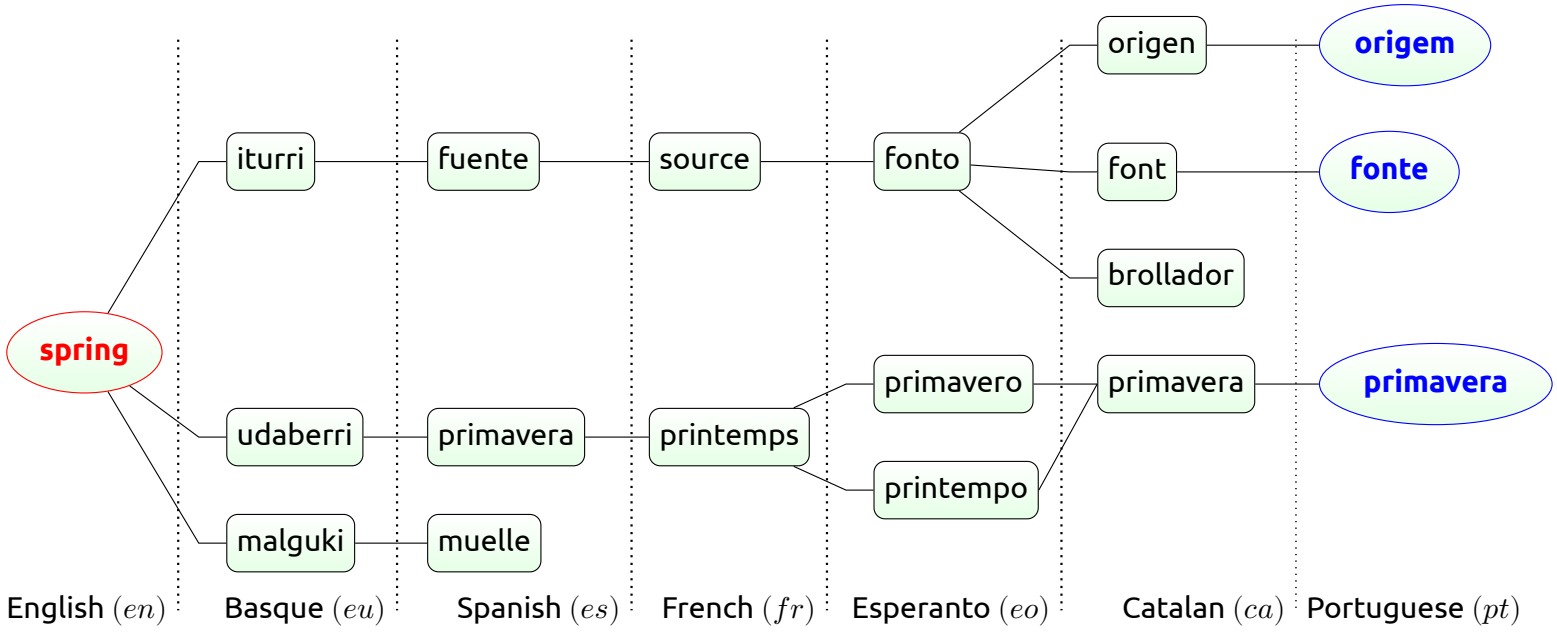
Long-short term memory

tex.stackexchange.com/questions/332747/
how-to-draw-a-diagram-of-long-short-term-memory

TikZ `foreach`

https://tex.stackexchange.com/questions/61805/tikz-using-loop-to-draw-grid-of-nodes

TikZ Forest set



Based on Sina Ahmadi's work. Requires of package `forest`

Ti*k*Z Mind map

From http://www.texample.net/tikz/examples/servers/

beamer

- Beamer is a document class to create slides
- You can do (almost) everything you do in any other TEX document
- Adds some features, such as title page
- Adds overlay specification commands using < >
- Dozens of default themes
  https://hartwork.org/beamer-theme-matrix/

Beamer environments

- `columns` environment

- `block`, `alertblock` and `examples` blocks (defined by each theme)

Title
block

Title
alertblock

Examples
Examples

Overlay specifications

- Beamer has several overlay specifications for *animations*

- The most basic command is `\pause`

Overlay specifications

- Beamer has several overlay specifications for *animations*

- The most basic command is `\pause`

<1: `uncover<3>` or `onslide<3>` have the same behaviour 1>
<2: 2>
<3: 3>
<4: `alt<6>` 4>

Overlay specifications

- Beamer has several overlay specifications for *animations*

- The most basic command is `\pause`

<1: `uncover<3>` or `onslide<3>` have the same behaviour 1>
<2: 2>
<3: 3>
<4: `alt<6>` 4>

Overlay specifications

- Beamer has several overlay specifications for *animations*

- The most basic command is `\pause`

<1: `uncover<3>` or `onslide<3>` have the same behaviour 1>
<2: `visible<4>` or `onslide+<4>` have the same behaviour 2>
<3: 3>
<4: `alt<6>` 4>

Overlay specifications

- Beamer has several overlay specifications for *animations*

- The most basic command is `\pause`

<1: `uncover<3>` or `onslide<3>` have the same behaviour 1>
<2: 2>
<3: `only<5>` or `onslide*<5>` have the same behaviour 3>
<4: `alt<6>` 4>

Overlay specifications

- Beamer has several overlay specifications for *animations*

- The most basic command is `\pause`

<1: `uncover<3>` or `onslide<3>` have the same behaviour 1>
<2: 2>
<3: 3>
<4: shows different text on slide 6 4>

Overlay specifications II

- Uncover and visible act the same if `\setbeamercovered{invisible}` is used (like in this presentation)
- Several macros and environments also have support for <>
- Create your own with `newcommand<>`
- There are different ways of writing intervals:
  - On a particular slide `<3>`, slides `<3,6>` or range `<3-5>`

Overlay specifications II

- Uncover and visible act the same if `\setbeamercovered{invisible}` is used (like in this presentation)
- Several macros and environments also have support for `<>`
- Create your own with `newcommand<>`
- There are different ways of writing intervals:
  - On a particular slide `<3>`, slides `<3,6>` or range `<3-5>`
  - From a slide onward `<3->` or up to `<-3>`

Overlay specifications II

- Uncover and visible act the same if `\setbeamercovered{invisible}` is used (like in this presentation)
- Several macros and environments also have support for `<>`
- Create your own with `newcommand<>`
- There are different ways of writing intervals:
  - On a particular slide `<3>`, slides `<3,6>` or range `<3-5>`
  - From a slide onward `<3->` or up to `<-3>`
  - On the next slide `<+>`

Overlay specifications II

- Uncover and visible act the same if `\setbeamercovered{invisible}` is used (like in this presentation)
- Several macros and environments also have support for `<>`
- Create your own with `newcommand<>`
- There are different ways of writing intervals:
  - On a particular slide `<3>`, slides `<3,6>` or range `<3-5>`
  - From a slide onward `<3->` or up to `<-3>`
  - On the next slide `<+>`
  - On the same slide as the previous specification, `<.>`

Overlay specifications II

- Uncover and visible act the same if `\setbeamercovered{invisible}` is used (like in this presentation)
- Several macros and environments also have support for `<>`
- Create your own with `newcommand<>`
- There are different ways of writing intervals:
  - On a particular slide `<3>`, slides `<3,6>` or range `<3-5>`
  - From a slide onward `<3->` or up to `<-3>`
  - On the next slide `<+>`
  - On the same slide as the previous specification, `<.>`
  - Also `begin{itemize}[<+->]` for automatic animation

Handouts

- By passing `handout` as a parameter to `documentclass`, it will create a handout, that is, a ”flattened” version of the animations
- Mileage may vary depending on how much do you (ab)use overlays
- Extra options such as multiple slides per page, borders...

Self notes

- You can also add notes to yourself by using `note`
- You can configure in which direction to ”extend” the slides
- Sadly no proper dual screen support, depends on external software, e.g. `pympress` (`https://www.scivision.dev/beamer-latex-dual-display-pdf-notes/`)

```
%\documentclass[notes]{beamer}       % print both
\documentclass[notes=only]{beamer}   % only notes
%\documentclass{beamer}              % only frames

\setbeameroption{show notes}    % Redundant with [notes]
\setbeameroption{show notes on second screen}
...
\note{Some note}
...
\note[itemize]{\item Point 1 \item Point 2}
```

Other beamer options

- `\documentclass[aspectratio=169]{beamer}` for 16:9 slides
  - You can compile twice and have two versions
  - Will not play well with horizontal rescales depending on `\textwidth`
  - Risky, many venues still use 4:3

- Self referencing:
  - `frame`s cannot be labelled using the `\label` macro
  - Instead use `\begin{frame}[label=my_label]`
  - Package `hyperref` provides `hyperlink`, needed for beamer links

Customize beamer

- Insight style in `insight.sty`
- Based on Sina's work
- Lots of default components
  - `setbeamerfont` and `usebeamerfont` for configuring fonts
  - `setbeamercolor` and `usebeamercolor` for configuring colours
- e.g. `\setbeamerfont{footer}{size=\tiny,series=\bf}` and `\setbeamercolor{footer}{fg=pantone174-6}` will automatically change all footers to tiny size, bold face, and grey colour
- Add or redefine to default components or create your own with `defbeamertemplate`, `setbeamertemplate`, `addtobeamertemplate`, etc.

BibTeX

- `ASCII`-based, one-size-fits-all bibliography management
- Dozens of fields and entry types (that define mandatory and optional fields)
- Takes only one `.bib` file as input
- Outputs `.bbl` files (basically `.tex`)
- A `.bst` file defines how the bibliography should be rendered

`biber`

- Unicode-based, one-size-fits-all bibliography management
- Dozens of fields and entry types (that define mandatory and optional fields)
- Takes multiple `.bib` file as input, but also supports other `xml`-based bibliography formats such as Zotero RDF, Endnote, etc. Also remote files over `HTTP` and `FTP`
- Outputs `.bbl` files and other formats such as `GraphViz` graphs, conversion between formats
- Better (full unicode) sorting and disambiguation
- Automatic unicode $\leftrightarrow$ LaTeX conversion (e.g. á $\leftrightarrow$ `{\'\a}`)
- 99.9% backcompatible with BibTeX format
- `.bbx`, `.cbx` and `.lbx` files determine how the bibliography should be rendered

`natbib`

- Only compatible with BibTeX
- Most stable approach
- Effectively no longer in development
- `.bst` are “a form of postfix stack language”, but loads of them already exist (and `makebst`)
- Based on natural and social sciences format, hard to have humanities-style citations

`natbib`: citing

- Multiple `cite*` flavours, check conference guidelines!
- Multiple references in the same command will also work, e.g.
  `\cite{bahdanau2014neural, sutskever2014sequence}`
- *Primitives*

`citeauthor`	for authors, condensed witn *et al.*
`citeauthor*`	for full author list
`citeyear`	for the year
`citenum`	for the position in the bibliography (some styles lack numbers in the bibliography)

`natbib` : printing the bibliography

- `bibliographystyle` to define the style
- `bibliography` to include the .bib and show the used bibliography
- If you want to add a cite to the biliography, you can use `nocite`
  - `\nocite{*}` to include the whole `.bib` file
- No default option to have cites without bibliography
  - Hide them by calling `bibliography` inside `newsavebox`
  - Use a package with the option, e.g. `nobibliography` from `bibentry` package
- Limited to one biblioraphy file

`natbib` example

```
\documentclass{article}
\usepackage{natbib}
...
Neural machine translation \cite{bahdanau2014neural}
is a paradigm
...
\bibliographystyle{mtsummit2019}  % no .bst
\bibliography{biblio}  % no .bib
```

`biblatex`

- Compatible with BibTeX and `biber`
  - Soon to be only `biber`
  - `\usepackage[backend=biber]{biblatex}`
- Extremely configurable
- Dozens of new cite types
- Configured using TeX macros
- Way less styles available, but slowly picking up

`biblatex`: citing

- Many more `cite*` commands

`footcite` for cites in footnotes

`supercite` for cites in superscripts

`volcite` for citing a particular volume of a cite

- `citefield` to get the value of any field

`biblatex`: printing the bibliography

- Style is defined as an option to the packate, e.g.
  `\usepackage[style=ieee]{biblatex}`

- Add bibliography with `addbibresource`, configurable to take multiple formats and remote locations if using `biber` backend
  - Also `bibliography` that takes one or more local `.bib` files

- Typeset the biblipraphy with `printbibliography`, that has many options:

`section` for printing the cites of a particular section

`type` for printing only entries with a particular type, e.g. `book`

`title` for configuring the title of the section

biblatex example

```
\documentclass{article}
\usepackage[style=numeric]{biblatex}
\addbibresource{xampl.bib}
\begin{document}
...
\cite{article-minimal,article-full,book-minimal,book-full}
...
\printbibliography[type=article,title={Articles}]
\printbibliography[type=book,title={Books}]
\end{document}
```

Macros

- TEX primitives will bite (unless for trivial usage)
  - `\let\foo\bar`: `\foo` is equivalent to `\bar` when `\let` was parsed
  - `\def\foo{\bar}`: `\foo` is equivalent to `\bar` whenever `\bar` is parsed
- Either will redefine already define macros
- Read the TEXbook (Chapters 9 and 20) for more info about `def`
- Do not use `let` unless you have a great understanding of TEX

Macros

- LaTeX primitives are safe(r)
  - `\newcommand{\foo}{\bar}`: `\foo` is equivalent to `\bar`, will fail if `\foo` is already defined
  - `\renewcommand{\foo}{\bar}`: `\foo` is equivalent to `\bar`, will fail if `\foo` is not yet defined
- Easily define number of parameters:

```
\renewcommand{\texttt}[1]{\colorbox{gray!10}
{\textttt{#1}}}
```

- Supports up to 1 optional argument (that will be #1):

```
\newcommand{\lawyers}[3][company]{#2, #3, and~#1}
\lawyers[H]{Dewey}{Cheatem}. % Dewey, Cheatem, and H
\lawyers{Dewey}{Cheatem} % Dewey, Cheatem, and company
```

http://joshua.smcvt.edu/latex2e/_005cnewcommand-_0026-_005crenewcommand.html

Advanced macros

- Macros with arbitrary number of parameters using low-level primitives: https://davidyat.es/2016/07/27/writing-a-latex-macro-that-takes-a-variable-number/-of-arguments/

- Macros with key-value style parameters: `pdfkeys` package, https://tex.stackexchange.com/questions/34312/how-to-create-a-command-with-key-values

Acknowledgements macro

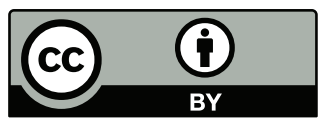
```
\makeatletter
\newcommand{\acknowledgment}%
{\@dblarg\beamer@acknowledgment}
\long\def\beamer@acknowledgment[#1]#2{%
  \def\insertacknowledgment{#2}%
  }
\makeatother
```

gnuplot

Daniel Torregrosa

daniel.torregrosa @ insight-centre.org

5th June 2019



Outline

History

- Initially created as a free-time project by several university students

- `gnuplot` is not related to the GNU project in any way
  - Proposed names include `llamaplot` and `nplot`
  - A pun on `newplot`

- It is not free software, albeit it is open source
  - Licence prohibits using the source code to create a new project

Outline

Why `gnuplot`

- Two and three dimensional plots
- Easily automatised
  - Beats copy-pasting data on spreadsheet-based software
  - Also beats 99% of plots generated with spreadsheet-based software
  - Can generate multiple formats, including `.tex` + `.eps`
- Maybe you are already using it!
  - Backend for `GNU Octave`
  - Bindings for `Java`, `Ruby`, `Python` ...
  - You can call it directly from TEX

Why not `gnuplot`

- You already use `python` and `matplotlib`
  - `gnuplot` is faster but offers limited computation options compared to full `python`
  - `matplotlib` lets you configure a lot more than `gnuplot`
  - `gnuplot` follows the `*nix` principles

Outline

Quick reference

`set` for options

`plot` for 2d plots

`splot` for 3d plots

`save` to save configuration and the latest `plot` / `splot` command

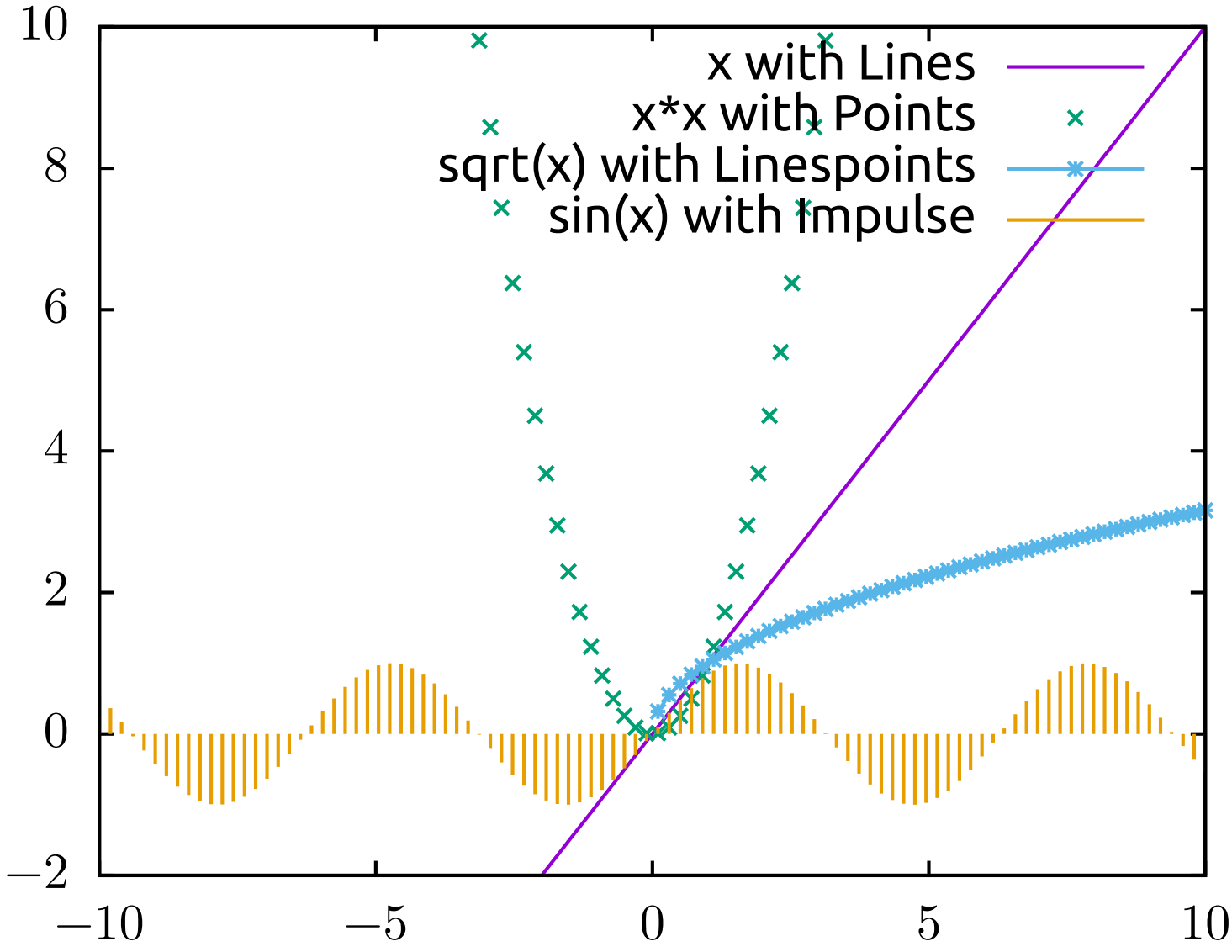
`load` to load a saved file

`help` help about any command

`show` for a fairly verbose explanation of the status of an option

- Many commands have shorthands, e.g. `with linespoints linestyle 2` can be typed `w lp ls 2`

Functions and types



Important stuff

- `set title 's'` : title of the plot
- `set xlabel 's'` : label the x axis, similar command for y axis
- `set samples x{,y}` : maximum of rendered points, default 100,100
  - For best results, it should be at least the size of the biggest data set
  - Be careful with functions!
- `set xrange [min:max]` : defines the range of the x axis
  - Either value can be `*` to auto-adjust
  - Either value can be empty, to keep the previous value
- `unset x` : returns `x` to factory value, e.g. `unset xrange` is equivalent to `set range [*:*]`

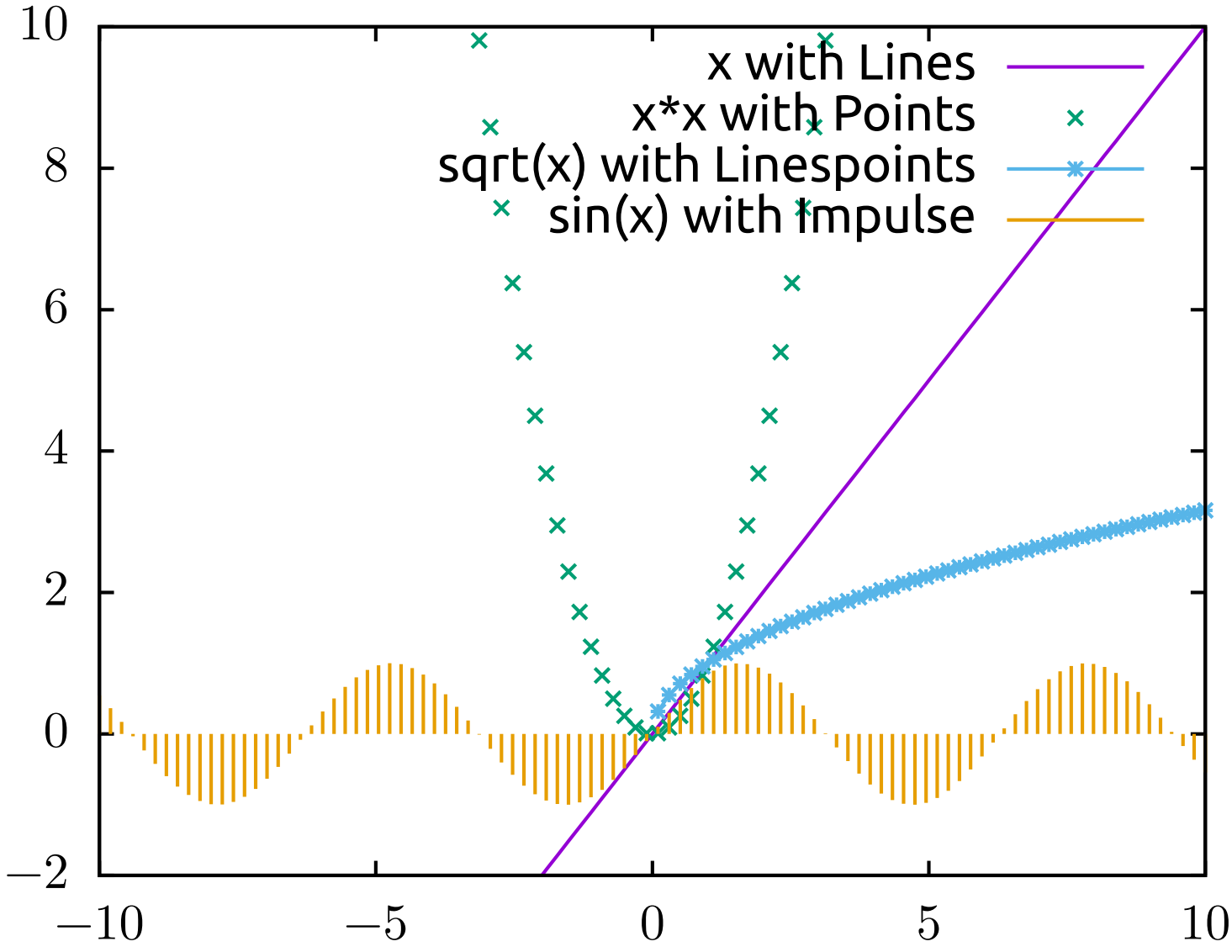
`tics`

- Tics can be drawn `in` or `out`
- Tics can be `mirror` ed on the opposite axis (or `nomirror`)
- Tics can be scaled to be bigger or smaller
  - Scale 0 = keeps the labels but no drawn tic
- Tics can be defined in 4 ways:
  - `autofreq` will do its best (usually too many tics)
  - `<incr>` to show a tic at fixed intervals
  - `<start>, <incr> {, <end>}` same but with start and end ranges
  - `('label1' pos1, 'label2' pos2 ...)` will write the corresponding label at the corresponding tic
- `add` can be used to incrementally define tics
- `format 'fs'` can be used to format, with `'fs'` as a `'printf'` -like expression

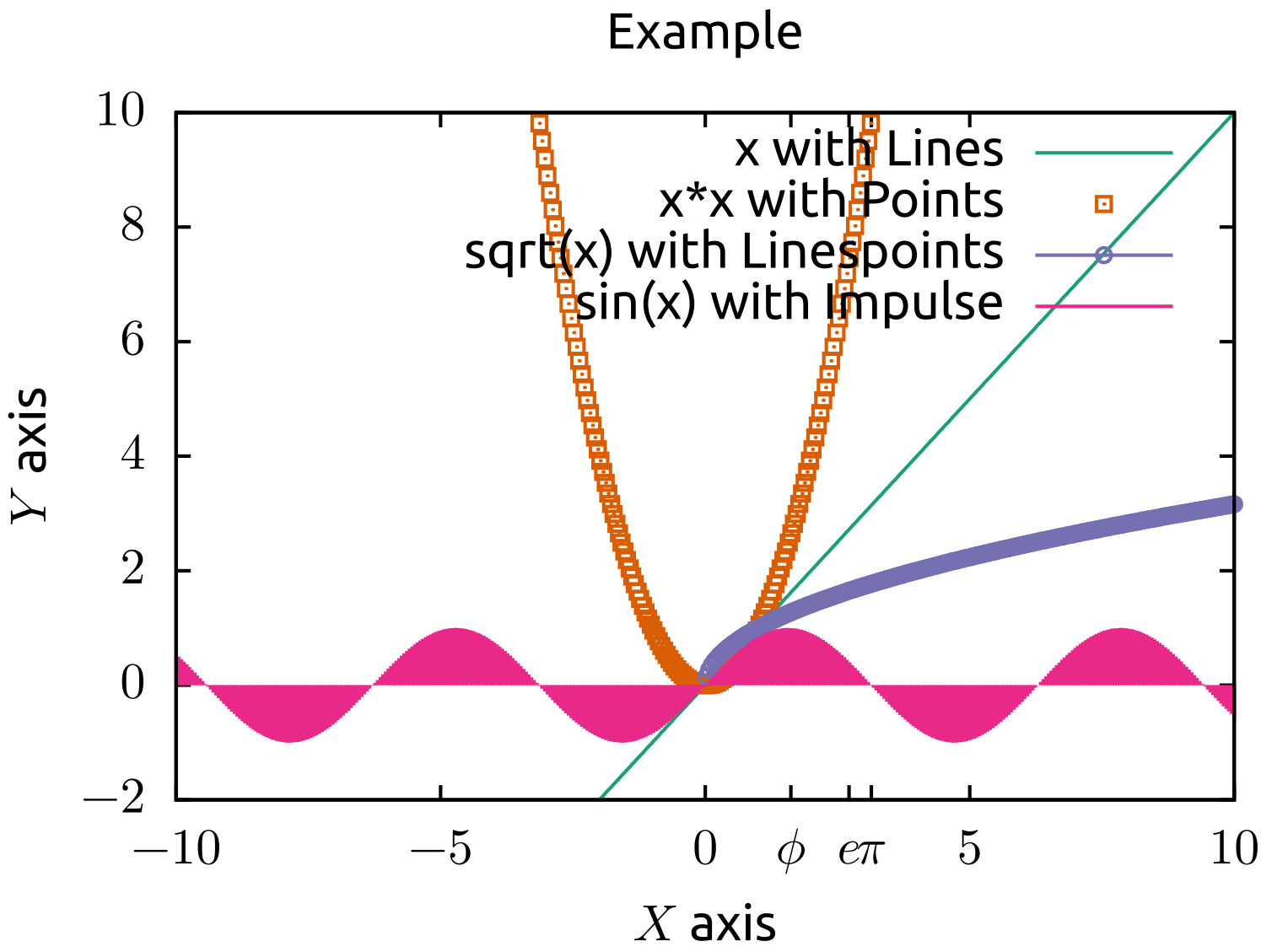
`set style line`

- `linetype` different combinations of lines and dashes
- `linecolor` colour of both line and points
  - `rgb '#RRGGBB'`
  - Some named colours available too
- `linewidth` for defining the width of the line, in points
- `pointtype` different point types
- `pointsize` for defining the size of the point, in points

All of the above



All of the above



See http://colorbrewer2.org for more colour series

Using

- Select columns from data, e.g. `using 2:3` will use the second and third columns
- Special 0-th column with the number of point
- By enclosing it in `()`, you can operate on the numbers
  - `(1)` is a literal 1
  - `($1)` is the value of the first column
  - `($2-$1)` is the value of the second column minus the value of the first column
- You can have multiple dataset in the same file by splitting them with two newlines
  - `using 1:2 index 0` will select the first block
- There is also an `every i_p:i_b:s_p:s_b:e_p:e_b` directive
  - `i` ncrement, `s` tart and `e` nd for `p` oints and `b` locks

Other plot types

- `candlesticks` (box and whisker), that require 5 columns
- `boxes`, that act like impulse but with width and can be filled with solid or transparent colors or patterns
- `image`, that generate imagemaps (e.g. heatmaps)
- `label`, that writes the third column as a label at the coordinate defined by the first and second columns
- First column/row can be ignored by adding `rowheaders` / `columnheaders`

Stats

- `stats 'file' {using} {name 'p'}` can be used to get statistics from files
- Same `using` options as *plot*, but can only process up to 2 columns
- Several variables in the form `p_variable` are assigned
  - `p` is the value of the `name` option, `STATS` if unset
  - `_x` and `_y` suffixes if using 2 columns
- File variables
  - `records`, `blocks`, `columns` (of the first line!)
- 1-column variables
  - `min`, `max`, `mean`, `median`, `stdev`...
- 2-column variables
  - `correlation`, `slope` / `intercept` (linear fit)...

Table

- `set table 'file'`
- Will *plot* to a file with a maximum of points defined by `set samples` in the `x y{ z} r` format
  - `r` is either `i` n-range, `o` ut-of-range or `u` ndefined
- You can then `plot 'file'` to draw the plot
- Useful for reusing data or modifying it, e.g.
  - The values of a binning/smoothing
  - The output of a function
- The table will have as many blocks as plots, e.g. `plot x, x**2` will generate a file with 2 blocks

Multiple axes

- Every plot has two x and y axes
- `x2label`, `x2tics`, `x2range`
- `plot 'file' axes x1y2`
- Set `nomirror` for all tics
- `set link x2 via x**2 inverse sqrt(x)`

Multiple plots

- `set multiplot` can be used to print multiple plots in the same *page*
- `set multiplot layout 2,3 {margins ...}` for automatic overlay
  - `set multiplot next` can be used to skip one of the predefined spots
- `set origin x,y; set size x,y` for manual overlay of each plot
  - `0,0` is the bottom-left corner

Key (legend)

- Plots with empty title are not shown in the key
- `set key vertical maxrow 1` for horizontal key
- The key can be set at a relative position (inside/outside, top/bottom/left/right/center/etc.) or absolute (at 0.2, 0.3)
  - The coordinates are **x and y coordinates in the plot** by default!
  - You can choose relative position in the graph (`at graph 0,0`) or whole image (`at screen 0,0`), with `0,0` = bottom left corner

Terminals

- gnuplot has several `terminals` available
- Default terminal is the interactive `qt` or `x11` (depending on system defaults)
- Notable terminals are

`pngcairo` outputs `.png`

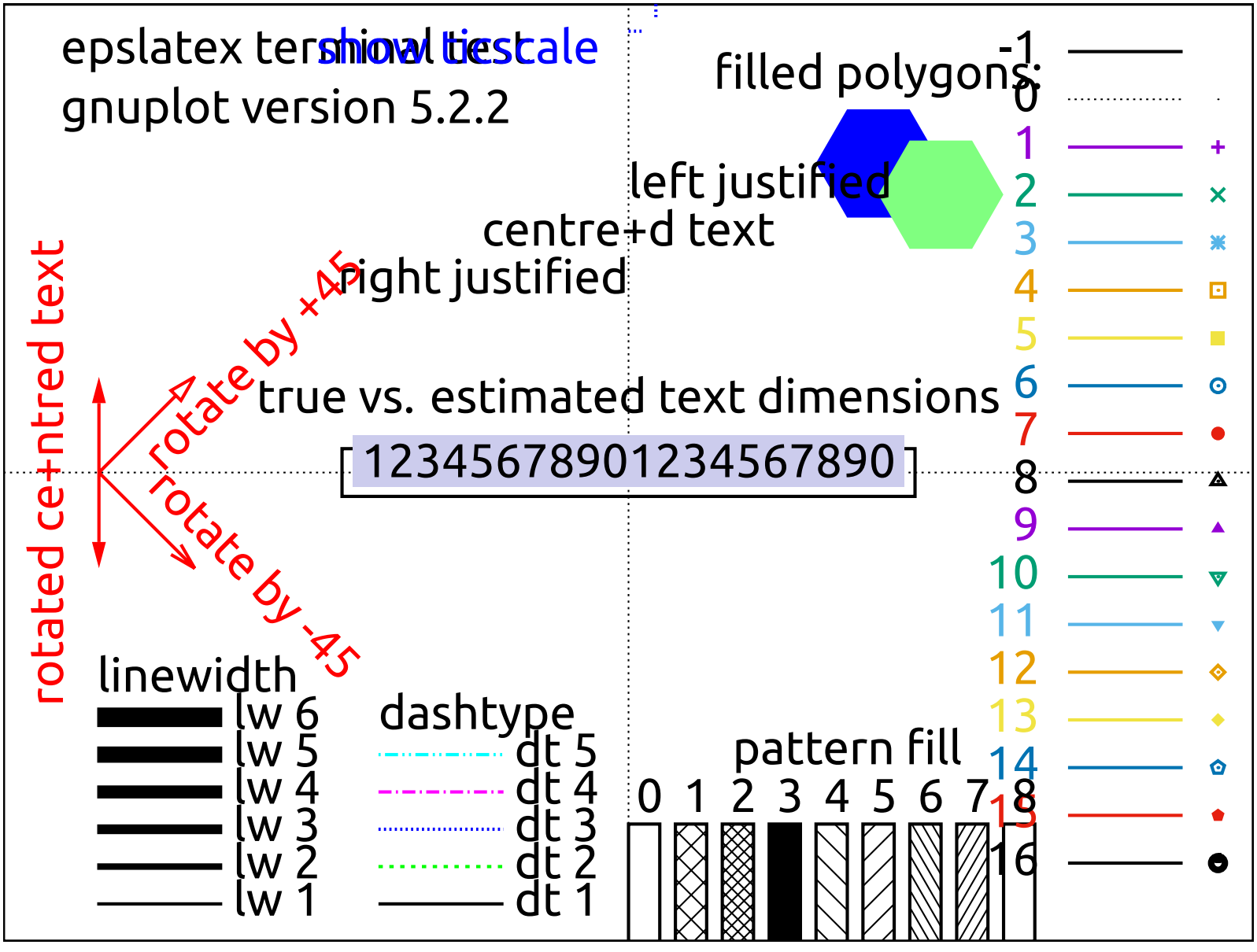
`epslatex` outputs a `.tex` and a `.eps` file, `input` the `.tex` file for inserting the image, will automatically inherit fonts!

`svg` for `.svg`

`canvas` for partially interactive `html5` canvas objects

- Many behaviour differences between terminals
- Multiple parameters can be configured, depending on the terminal, e.g. mono/color, size, base linewidth, etc.
  - `enhanced` means *enhanced* text, e.g. `a_b` becomes a subscript. Will mess up your LaTeX!

Test


epslatex terminal test
gnuplot version 5.2.2
show tics
filled polygons:
left justified
centre+d text
right justified
rotated ce+ntred text
rotate by +45
rotate by -45
true vs. estimated text dimensions
12345678901234567890
linewidth
lw 6
lw 5
lw 4
lw 3
lw 2
lw 1
dashtype
dt 5
dt 4
dt 3
dt 2
dt 1
pattern fill
0 1 2 3 4 5 6 7 8
-1
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Test with `dumb` terminal

```
gnuplot> set terminal test
Terminal type is now 'dumb'
Options are 'feed  size 79, 24 aspect 2, 1 mono'
gnuplot> test

+-dumb  terminal test-----------------show ticscale-----------------1-+---+--+
|                                    $$$     filled polygons:      0 +...+ .|
| gnuplot version 5.2.2              :             XXXX            1 ***** A|
|                                    :             XXXXXXX         2 ##### B|
| ^            .>                    :            XXXXXXXXX        3 $$$$$ C|
| |          ..                      left justifieXXXXXXXXXX       4 %%%%% D|
| |        ..                   centre+d text       XXXXXXXX        5 @@@@@ E|
| |      ..             right justified:               XXXX         6 &&&&& F|
| |    ..                              :                             7 ===== G|
| |  ..                                :                             8 ***** H|
| |..                true vs. estimated text dimensions             9 ***** I|
+.cannot rotate text.....+-------------------+.................10.#####.J+
| |  ..                                :                           11 $$$$$ K|
| |    ..                              :                 n+1       12 %%%%% L|
| |      ..                            Enhanced text:   x0          13 @@@@@ M|
| |        ..                          :               Bold Italic  14 &&&&& N|
| |          ..                        :                            15 ===== O|
| |            .                       :                            16 ***** P|
| |             >                      :                            17 ***** Q|
|                                      :      pattern fill          18 ##### R|
|                                              8                     19 $$$$$ S|
|                                      ||||||||||                    20 %%%%% T|
|                                      ||||||||||                       |
+----linewidthlw 6-----dashtype dt 5---++++++++++---------------------------+
```

Coding

- Variables `x=2`
- Functions `max(x, y) = (x > y ? x : y)`
- Conditional
  `if (condition) something; else other thing;`
- Loops
  - `do for [i=1:10] {…}`
  - `plot for [i=1:10] …`
- Simple string manipulation
  - `a.'.txt'` concatenates
  - `words("a b c") = 3`
  - `word("a b c", 2) = b` (1-indexed)
- `system("ls")` for system calls

Outline

When to use …

`points` for x/y plots, e.g. price of car vs max speed

- Throw some fitted curves in (if it makes sense)

`lp` for data with trends, e.g. speed vs fuel use

- Only lines makes it very hard to read in b/w
- By using points on a subset of points you improve the readability
- Even better: manually set labels near points

`bars` : for clustered data (e.g. histograms) or one axis is nominal e.g. car sales by brand

- Can be rotated 90° to improve readability

`box&whisker` : distribution of data, e.g. profit per car by year

- Hides nonnormal distributions
- Better when supported with a violin plot!

When to use … II

`heatmap` : 3d data in 2d, e.g. confusion matrix

- Hard to read unless data is sparse

`stacked` : like lines but

- Less intra-series resolution
- Easier to compare inter-series

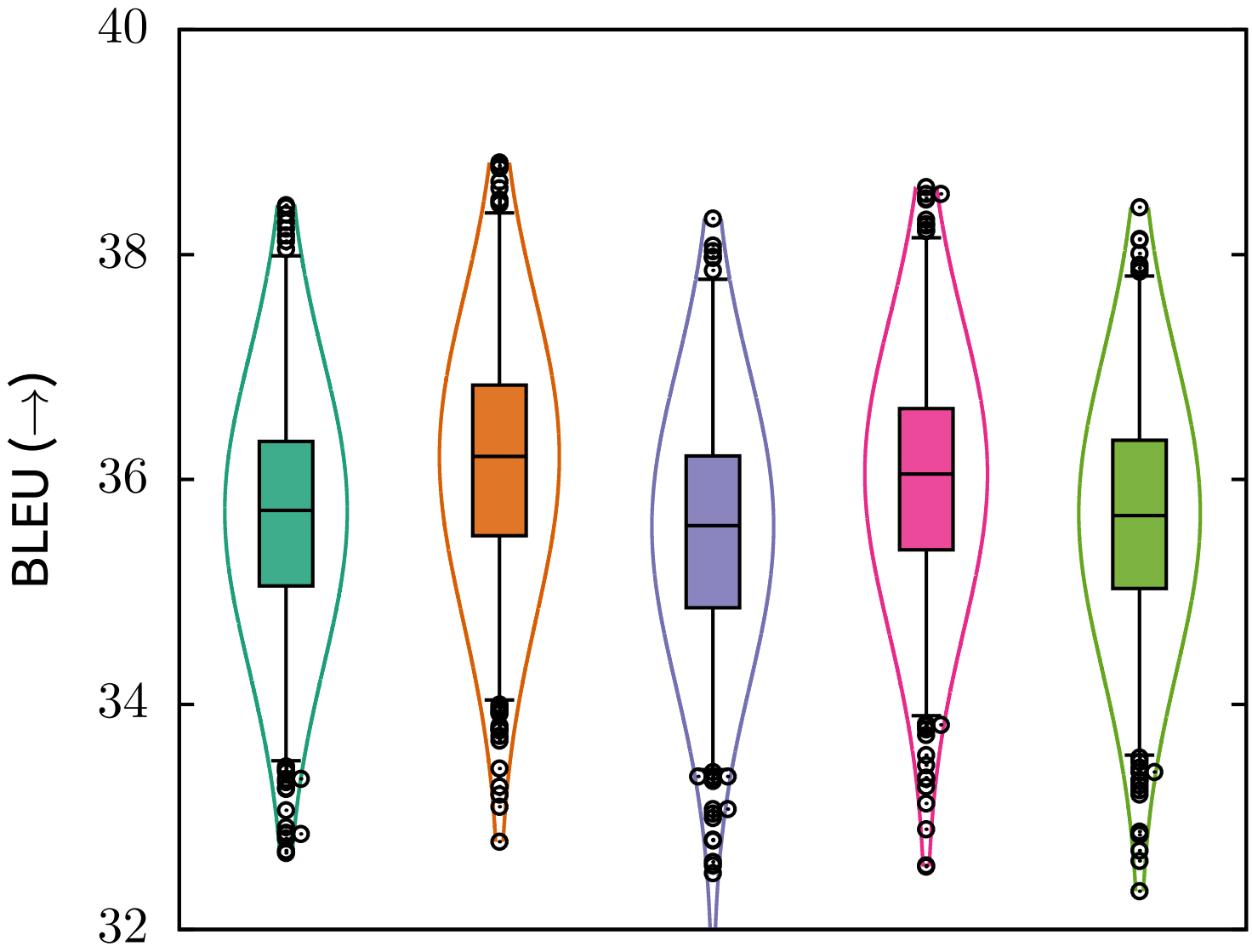
`circle` : special kind of `point` where the third column defines the size of the point, e.g. bubble charts

`pie charts` : no

`table` : sometimes a plot is not the best way

- All plots trade resolution for readability
- If you want 100% precise readings, use a table
- Wrongly adding a table to your presentation is a nice way of losing the audience

Box and violin plots



Box and violin plots I

```
set samples 3000
set boxwidth 0.25 absolute
set style fill solid 0.85 border lt -1
set style boxplot fraction 0.975 pt 6

bleuFiles=system("ls -1 stats/*Bleu.data")

unset xrange
do for [i=1:words(bleuFiles)] {
    set table 'bleu'.i
    plot word(bleuFiles,i) using 1 smooth kdensity \
    bandwidth 1
}
unset table
```

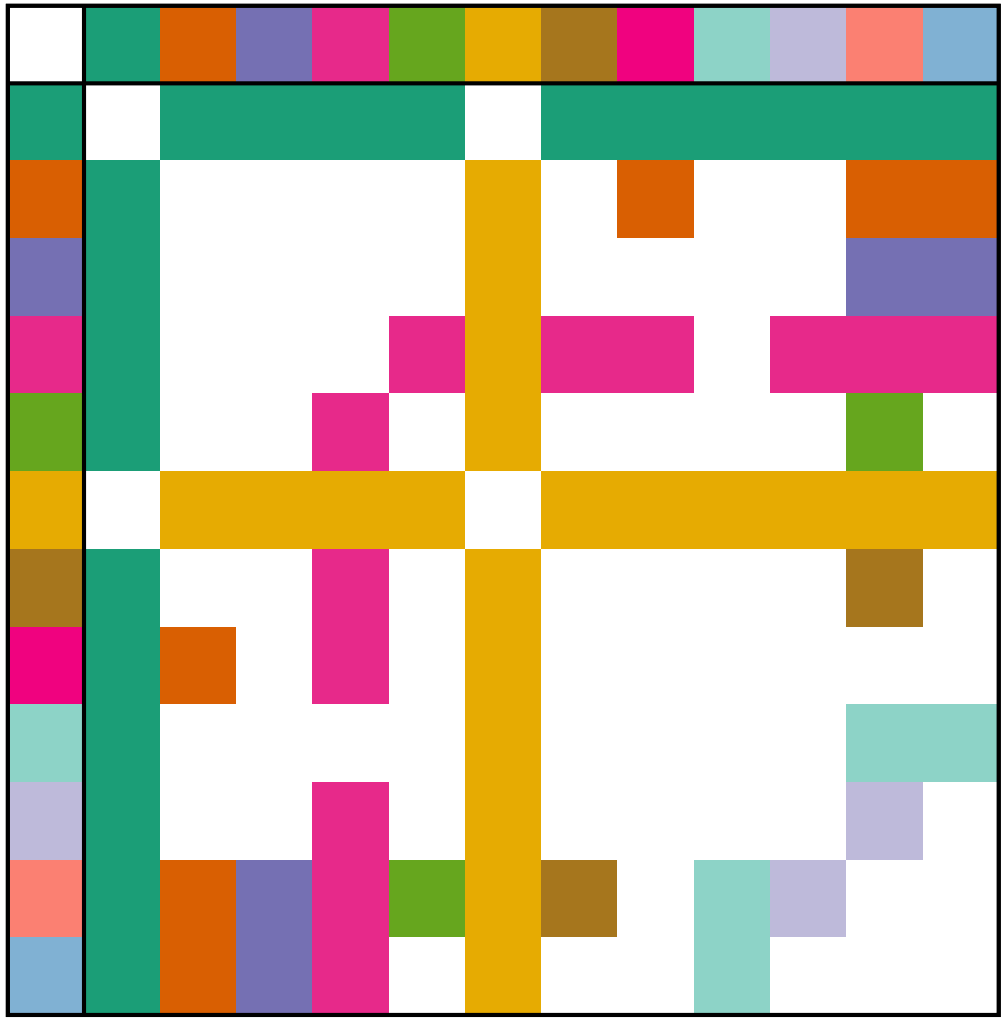
Adjusting the number of tics

```
mintics = 5

max(x, y) = (x > y ? x : y)
min(x, y) = (x < y ? x : y)
roundTo(x, y) = ceil(x/y)*y

maxV=0
minV=99999999
do for [i=1:words(bleuFiles)] {
    stats 'bleu'.i using 1 name 'y' nooutput;
    maxV=max(maxV, y_max);
    minV=min(minV, y_min);
}
set ytics roundTo((maxV-minV)/minTics, 1)
```

Box and violin plots II

```
tics=(0.5+words(bleuFiles))
set xrange [0.5:tics]

set ylabel "BLEU ($\rightarrow$)"
unset xtics
unset yrange
plot for [i=1:words(bleuFiles)] 'bleu'.i using \
        (i-$2/1000.):1 w lines ls (i) t '', \
    for [i=1:words(bleuFiles)] 'bleu'.i using \
        (i+$2/1000.):1 w lines ls (i) t '', \
    for [i=1:words(bleuFiles)] word(bleuFiles,i) \
        using (i):1 with boxplot fc ls i t ''
```

Heatmaps



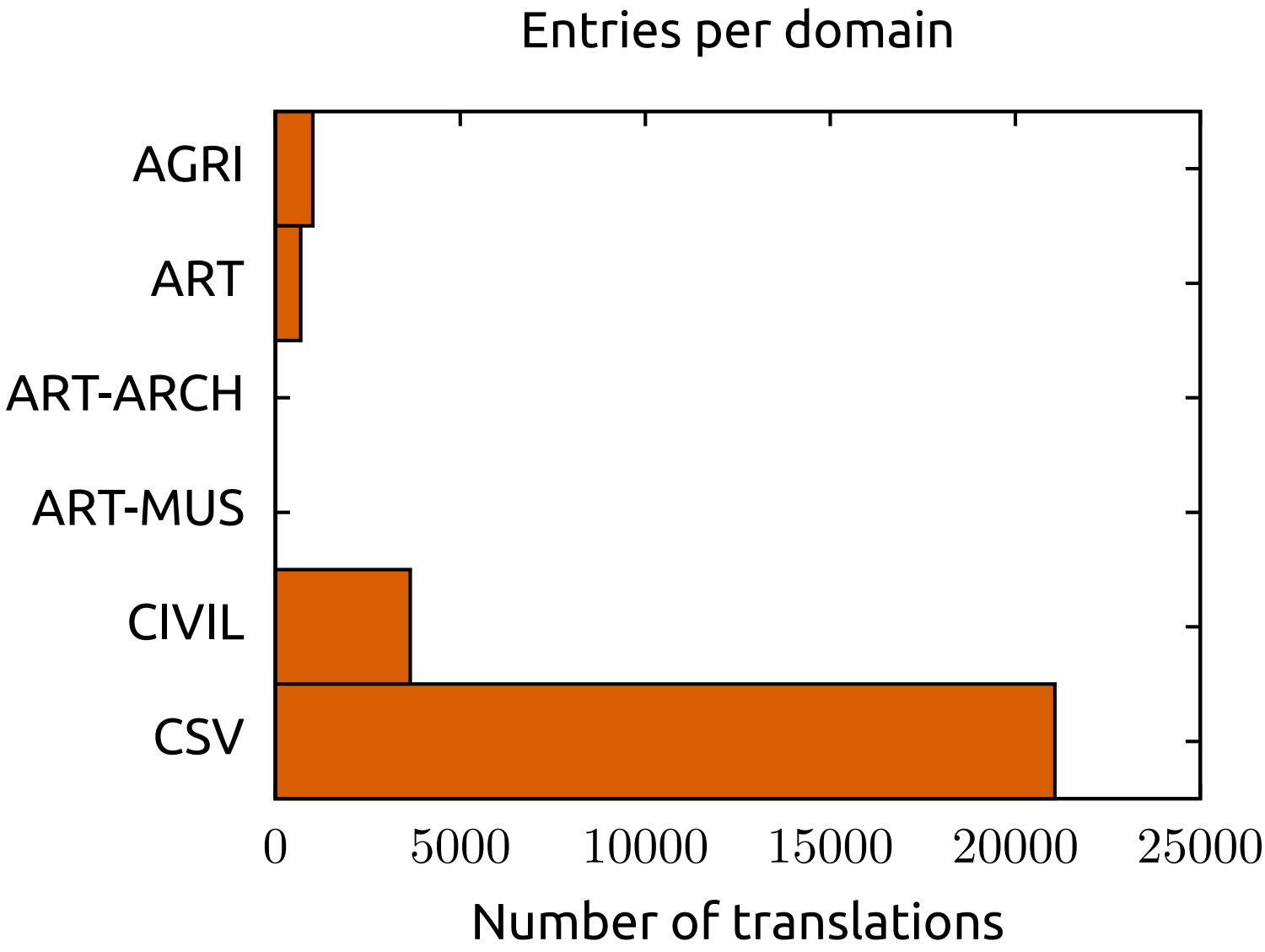
Heatmaps I

```
unset colorbox
set cbrange [-1:15]
set palette maxcolors 17
set palette model RGB defined (-1 '#FFFFFF',
  0 '#1B9E77', 1 '#D95F02', 2 '#7570B3',
  3 '#E7298A', 4 '#66A61E', 5 '#E6AB02',
  6 '#A6761D', 7 '#F0027F', 8 '#8DD3C7',
  9 '#BEBADA', 10 '#FB8072', 11 '#80B1D3',
  12 '#FDB462', 13 '#B3DE69', 14 '#FCCDE5',
  15 '#FF5555')
```

Heatmaps II

```
set grid front linetype -1 lw 1
set xtics scale 0 ("" 0.5)
set ytics scale 0 ("" 0.5)
set yrange [*:*] reverse
plot 'Bleu.significance' matrix with image t ''
```

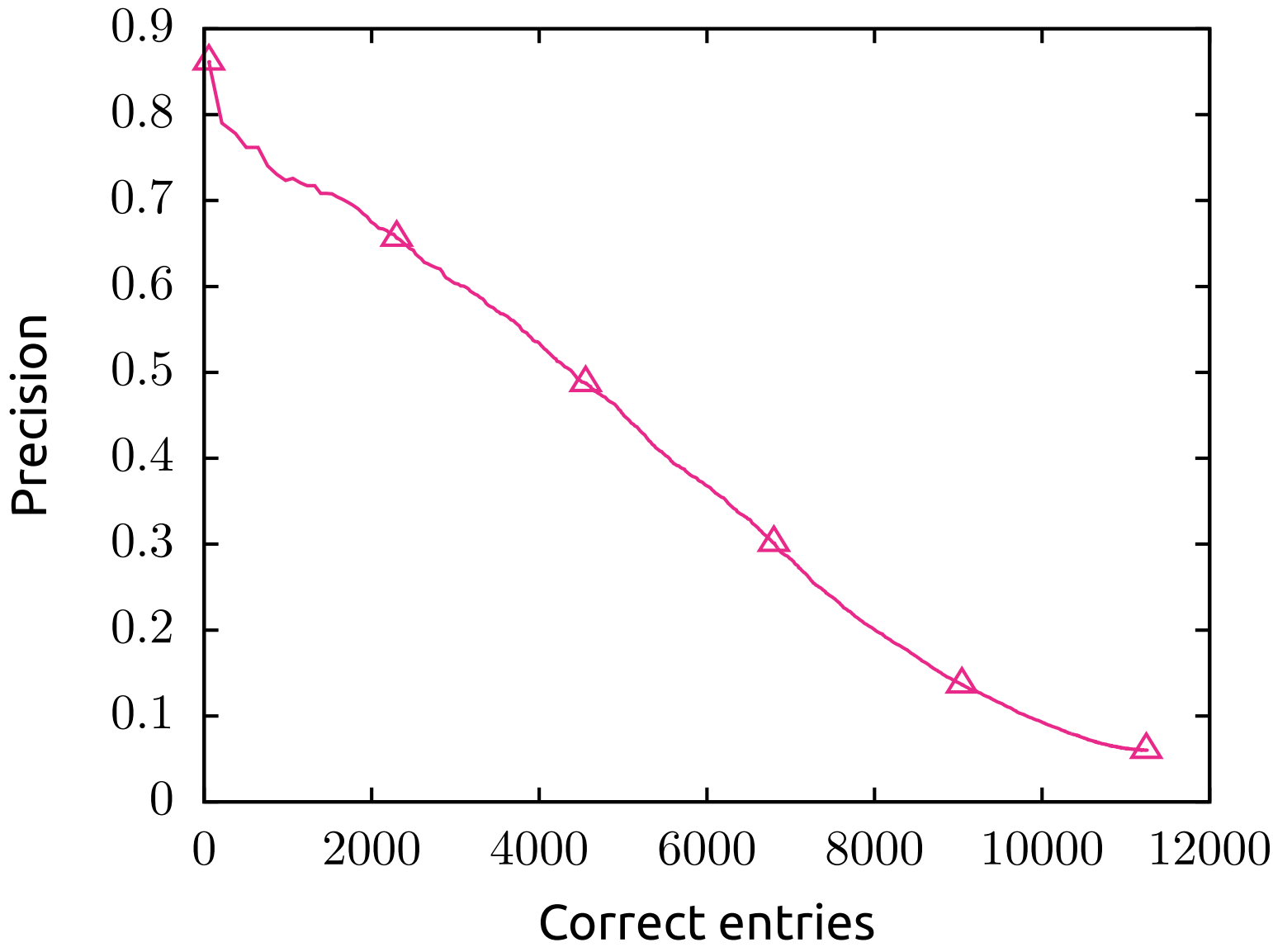
Horizontal bars



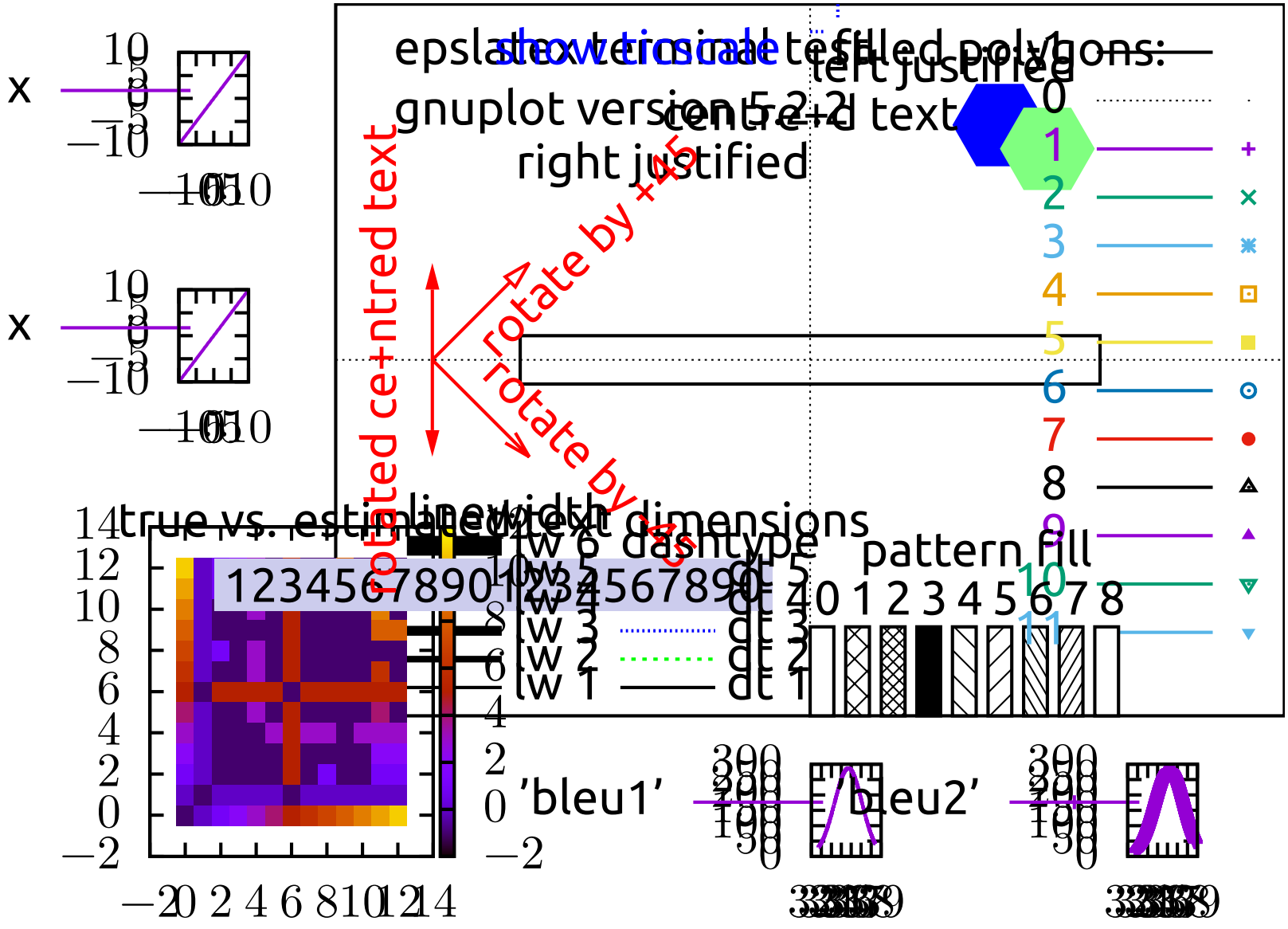
Horizontal bars I

```
set xrange[0:*]
set yrange[*:*] reverse
set title 'Entries per domain'
set xlabel '# of translations'
set ylabel ''
plot 'dom_count' u (0):($0):(0):2:($0-.5):($0+.5):ytic(1) \
  w boxxyerr ls 2 t ''
```

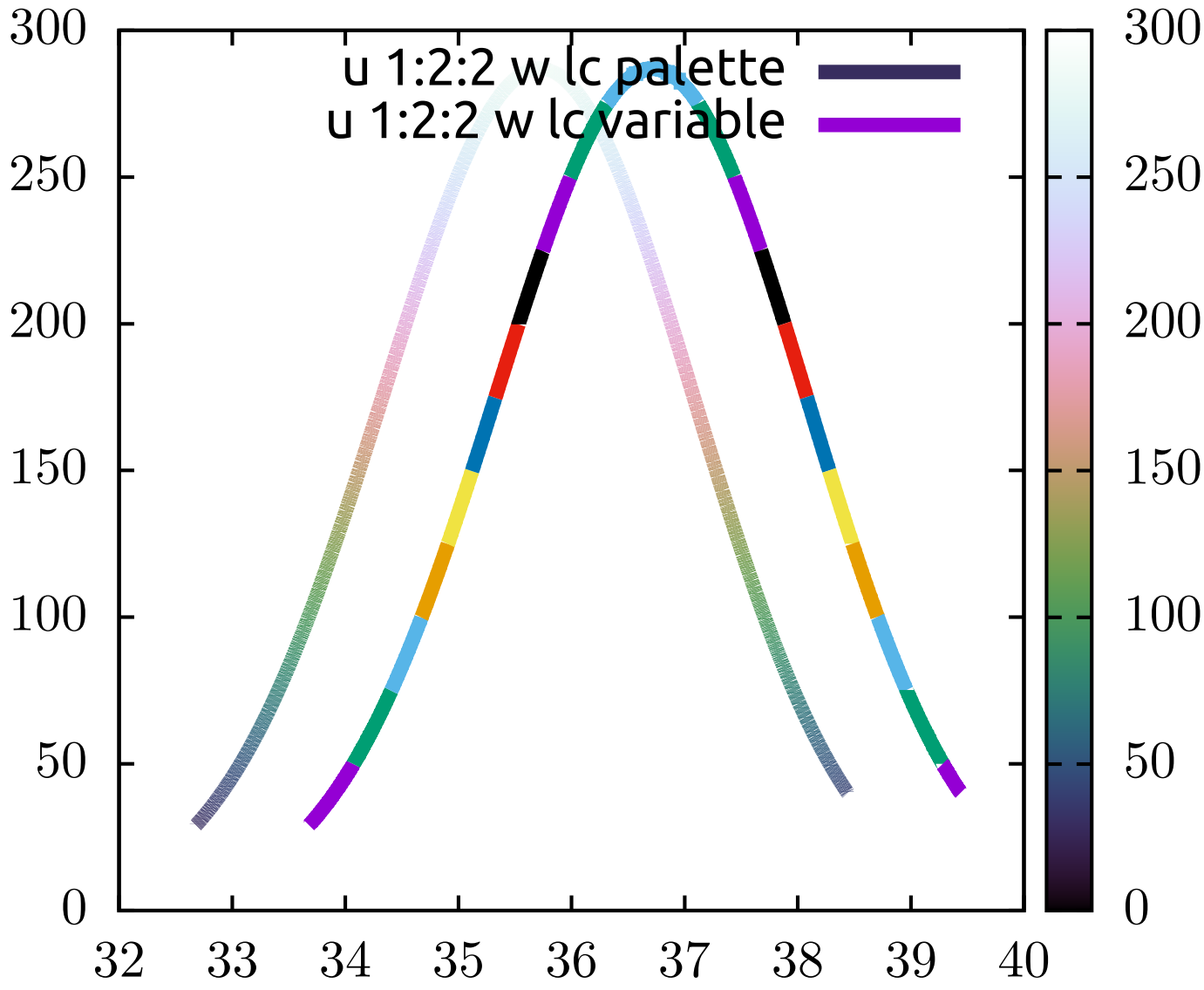
Subset


Precision
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2000
4000
6000
8000
10000
12000
Correct entries

Multiplot


x
x
epslatex terminal
gnuplot version 5.2.2
show trig scale
left justified
centre+d text
right justified
rotated ce+ntred text
rotate by +45
rotate by -45
true vs. estimated
linewidth
text dimensions
lw 6 dashtype
lw 5
lw 4
lw 3
lw 2
lw 1
dt 5
dt 4
dt 3
dt 2
dt 1
pattern fill
'bleu1'
'bleu2'

Palette and variable



More info

- The `help` command has a lot of information

- Lots of demos
  http://gnuplot.sourceforge.net/demo_5.2/