# Documentation for `mlmacros.sty`

Maximilian Soelch

November 8, 2018

The `mlmacros` package provides convenient and readable macros for commonly used syntax when writing scientific articles about machine learning.

## 1 Subindexing sequence data

When writing about algorithms on sequence data, we often refer to variables at a specific time by subindexing, e. g. $x_t, x_{t-1}, x_{T+2}$, or $x_{i:j} = (x_i, \ldots, x_j)$ for a sequence.

This package provides shortcuts for the most common indexes:

`\tm`, `\tp`, `\ts`, `\tsm`, `\tsp`, `\Tm`, `\Tp`, `\Ts`, `\Tsm`, `\Tsp`, `\tsub`

| | | | | |
|---|---|---|---|---|
| `x\tm` | `x\tp` | `x\ts` | `x\tsm` | `x\tsp` |
| $x_{t-1}$ | $x_{t+1}$ | $x_{1:t}$ | $x_{1:t-1}$ | $x_{1:t+1}$ |
| `x\tm{k}` | `x\tp{k}` | | `x\tsm{k}` | `x\tsp{k}` |
| $x_{t-k}$ | $x_{t+k}$ | | $x_{1:t-k}$ | $x_{1:t+k}$ |
| `x\Tm` | `x\Tp` | `x\Ts` | `x\Tsm` | `x\Tsp` |
| $x_{T-1}$ | $x_{T+1}$ | $x_{1:T}$ | $x_{1:T-1}$ | $x_{1:T+1}$ |
| `x\Tm{k}` | `x\Tp{k}` | | `x\Tsm{k}` | `x\Tsp{k}` |
| $x_{T-k}$ | $x_{T+k}$ | | $x_{1:T-k}$ | $x_{1:T+k}$ |
| `x\tsub{k}` | | | `x\tsub{i}{j}` | |
| $x_k$ | | | $x_{i:j}$ | |

The logic behind the commands is the following:

- Start with the desired upper or lower case `t`.
- Optionally add an `s` for a sequence starting at 1.
- Add `m` or `p` for minus or plus.
- Optionally add an argument to replace the default 1.

Notice that the plus and minus signs are slightly shortened, cf.

$x_{t-1:T+K}$

$x_{t-1:T+K}$

Use `\shortminus` or `\shortplus` to make use of these. You can also use `\tminus` or `\tplus` instead of `t\shortminus` or `t\shortplus`, respectively.

## 2 \variables

Macros can help a great deal to stick with keeping notational conventions while maintaining readability of the source file. However, with an increasing number of conventions, maintenance of the macros becomes just as hard.

The `\variables` macro eases some of that pain. It is called in the preamble with a list of variables and dynamically creates macros based on this list. For example, the call `\variables{x,y,A}` automatically provides the following macros:

| | | | |
|---|---|---|---|
| `x` | `\xseq` | `\bx` | `\bxseq` |
| | $\bar{x}$ | $\boldsymbol{x}$ | $\bar{\boldsymbol{x}}$ |
| `y` | `\yseq` | `\by` | `\byseq` |
| | $\bar{y}$ | $\boldsymbol{y}$ | $\bar{\boldsymbol{y}}$ |
| `A` | `\Aseq` | `\bA` | `\bAseq` |
| | $\bar{A}$ | $\boldsymbol{A}$ | $\bar{\boldsymbol{A}}$ |

It works with more complicated replacements, such as greek letters or other math symbols. Calling `\variables[mean,std]{\mu,\sigma}` provides commands such as `\mean` for $\mu$, `\bmeanseq` for $\bar{\boldsymbol{\mu}}$ or `\stdseq` for $\bar{\sigma}$. Notice that the list delimiter currently needs to be a comma *without enclosing spaces*.

More complicated macro substitutions currently do not work with `\variables`. However, you can exploit the underlying command `\varmacros` for every pair: `\varmacros{E}{\mathcal{E}}` for commands like `\bE`, which yields $\mathcal{E}$.

Further automatically created macros can easily be edited and/or extended by adjusting the `mlmacros.sty` file.

## 3 \probdists

The `\probdists` macro provides a rich macro for probability distributions.

The call `\probdists{p}` provides a command that can be used in any of the following ways:

| `\p` | `\p{x}` | `\p{x}{z}` | `\p[y]{x}{z}` | `\p[y]{x}` | `\p[y]` |
|---|---|---|---|---|---|
| $p$ | $p(x)$ | $p(x \mid z)$ | $p_y(x \mid z)$ | $p_y(x)$ | $p_y$ |

The `\p` macro dynamically decides which parts to display. This allows for rapid adjustment of the syntax. The parentheses adjust dynamically via internal usage of `\mleft` and `\mright`, unless starred commands (`\p*` etc.) are used,

$$p\left(\frac{x}{y}\right) \text{ vs. starred } p(\tfrac{x}{y}).$$

`\p` ignores arguments not given in braces, e. g. `\p{x}z` yields $p(x)z$.

Of course, `probdists` can be called with a list of letters, e. g. `\probdists{p,q}`. Similarly as the `\variables` macro, we can use, e. g. Greek letters, by passing optional arguments: `\probdists[policy]{\pi}` provides macros such as `\policy{u}{x}` for $\pi(u \mid x)$.

At this point, complicated constructions like $\mathbb{P}$ can be achieved by calling the underlying `\MkProbDist`, e. g. `\MkProbDist{P}{\mathbb{P}}` for $\mathbb{P}(x)$.

*Warning:* The macros provided by `probdists` overwrite any previously created macro of the same name.

# 4 `\expc, \var, \loss, \cost, \entropy`

For expectation and variance, this package provides overloaded macros similar to the dynamic macros provided by `\probdists` in section 3.

The application is straightforward:

| `\expc{x}` | `\expc{x}{z}` | `\expc[y]{x}{z}` | `\expc[y]{x}` |
|---|---|---|---|
| $\mathbb{E}[x]$ | $\mathbb{E}[x \mid z]$ | $\mathbb{E}_y[x \mid z]$ | $\mathbb{E}_y[x]$ |

and analogously

| `\var[y]{x}{z}` | `\loss[y]{x}{z}` | `\cost[y]{x}{z}` | `\entropy[y]{x}{z}` |
|---|---|---|---|
| $\mathrm{Var}_y[x \mid z]$ | $\mathcal{L}_y(x \mid z)$ | $\mathcal{C}_y(x \mid z)$ | $\mathbb{H}_y(x \mid z)$ |

All comments from section 3 apply here as well.

# 5 Drafting a paper

This section collects some macros on top of the `todonotes` package. When a reference is missing in the text, you can use `\addref` . You can also add a list argument that will be turned into bullets .

Following →this blog post by Eliezer Yudkowsky, Yudkowsky, 2007, it's a good practice to say "And then magically , we get A", or "by magic , B follows" to make the missing links and arguments explicit. This package provides the two corresponding macros `\magic``\magically`, which can also be used like `\magic{idea to fix this}` to yield magic .

Add reference(s).

Add reference(s).
- missingref1
- missingref2

Science > Magic!

Science > Magic!

Science > Magic!
idea to fix this

# 6 Syntactic sugar

## 6.1 Math

Some general, commonly used math expressions are provided by short, readable macros:

- `mathbb` wrappers: `\NN` for $\mathbb{N}$, `\RR` for $\mathbb{R}$, `\PP` for $\mathbb{P}$, `\EE` for $\mathbb{E}$.
- `mathcal` wrappers: `\mcX` for $\mathcal{X}$, `\mcU` for $\mathcal{U}$, `\mcZ` for $\mathcal{Z}$, `\mcD` for $\mathcal{D}$.
- `\data` for $\mathcal{D}$.
- `\ind` for $\mathbb{1}$.
- `\left\right` wrappers:

  `\abs{}` : $\left|\frac{a}{b}\right|$ `\set{}` : $\left\{\frac{a}{b}\right\}$ `\interval{}` $\left[\frac{a}{b}\right]$

- Kullback-Leibler divergence: `\kl{p}{q}` for $\mathrm{KL}(p \parallel q)$. The parentheses adjust to the height of the arguments unless you use `\kl*`.
- Gaussian/Normal distributions: `\gauss{0,1}` for $\mathcal{N}(0, 1)$. The parentheses adjust to the height of the arguments unless `\gauss*` is used. Overloaded, can be used for conditionals as well: `\gauss{x}{\mu, \sigma}` gives $\mathcal{N}(x \mid \mu, \sigma)$.
- Integrals: `\dint` for nicer integrals:

  `\int x \dint x:` $\int x \, \mathrm{d}x$ `\int x dx:` $\int x \mathrm{d}x$

- Equations that are to be proven: `x \shallbe y` for $x \stackrel{!}{=} y$.

### 6.2 Other

One of the most cumbersome typesetting issues is the correct spacing after abbreviations.

This packages provides macros that automatically apply correct spacing and follow →this guide for

- *with respect to*: `derivative \wrt input` ⇝ derivative w. r. t. input,
- *exempli gratia*: `algrorithms, \eg EM` ⇝ algorithms, e. g. EM; also `\Eg` for E. g.,
- *id est*: `156 members, \ie almost all` ⇝ 156 members, i. e. almost all; also `\Ie` for I. e.,
- *et cetera*: `Germany, France, \etc` ⇝ Germany, France, etc.

The latter command checks if a period follows, i. e. `\dots Germany, France, \etc. Other countries \dots` yields ... Germany, France, etc. Other countries ...

## 7 Style of this document

The default style of this document uses Palantino as the main font via

```
\usepackage[osf,sc]{mathpazo}
\linespread{1.05}\selectfont
```

with some extra spacing between lines. Moreover, the Euler math font is used:

```
\usepackage[euler-digits]{eulervm}
```

The `amsmath` package is loaded with the `fleqn` for left flush of equations.

All these changes were taken from the highly recommended →`classicthesis` package by André Miede.

Other recommended nice-to-have packages for convenient and good-looking typewriting are `cleveref` for extremely clever referencing, `nicefrac` for better handling of inline fractions, and `microtype`, which takes care of small typesetting issues that improve the overall document appearance.

# 8 TEXstudio autocomplete

Along with this style file and example document, a `cwl` file is provided. By adding it to your TEXstudio, autocomplete for the static macros is provided. At this point, autocomplete of the dynamically produced `variables` and `probdists` is not included.

Under Windows 10, it has to be copied to the directory

`%appdata%\texstudio\completion\user`.

Under Ubuntu 16.04 and 18.04, it has to be copied to the directory

`~/.config/texstudio`.

# References

Yudkowsky, Eliezer (Aug. 29, 2007). *Say Not "Complexity"*. URL: `https://www.lesswrong.com/posts/kpRSCH7ALLcb6ucWM/say-not-complexity` (visited on 04/25/2018).