Demo: Loading Macros via `include-in-header`

This page demonstrates the `include-in-header` strategy for loading macros. The YAML front matter of this file includes:

```yaml
format:
  html:
    include-in-header:
      - macros-header.html   # MathJax macro config script
  pdf:
    include-in-header:
      - macros.qmd           # raw LaTeX goes directly into preamble
  revealjs:
    output-file: demo-include-in-header-slides.html
    scrollable: true
    center: false
    include-in-header:
      - macros-header.html   # loader that reads macros.qmd
  docx: default
```

> **Note (HTML/RevealJS):** `macros-header.html` is a small loader script (not a second macro-definition file). It reads `macros.qmd`, converts macro definitions to a MathJax config object, and sets `MathJax.tex.macros` before typesetting. For RevealJS, it also forces the built-in Reveal math plugin to use MathJax 3. This requires `macros.qmd` to be published as a static resource in the site output. View the RevealJS slides output.
>
> **Note (PDF):** For PDF, `macros.qmd` can be included directly since its raw LaTeX goes into the document preamble.
>
> **Note:** This demo uses `macros-header.html` / `macros.qmd` because both files are at the root of this repository. When using this repository as a Git submodule

(cloned into a `macros/` subdirectory), replace with `macros/macros-header.html` and `macros/macros.qmd`.

The same configuration can be placed in a shared `_quarto.yml` to apply project-wide:

```yaml
format:
  html:
    include-in-header:
      - macros/macros-header.html
  pdf:
    include-in-header:
      - macros/macros.qmd
  revealjs:
    include-in-header:
      - macros/macros-header.html
```

This inserts the macro definitions for each output format:

- **PDF**: `macros.qmd` goes into the LaTeX preamble via `include-in-header` — macros work, including `\providecommand` (note that LaTeX's font-encoding initialization restores `\v` as a text caron accent at `\begin{document}`, so `\v{x}` produces a text accent error in math mode under `include-in-header` unless re-asserted in the preamble; use `\vec{x}` or `\vecf{x}` for robust cross-format vector notation).
- **HTML**: `macros-header.html` is inserted via `include-in-header` and loads macros from `macros.qmd` into `MathJax.tex.macros`.
- **RevealJS**: `macros-header.html` does the same loader step and switches the Reveal math plugin to MathJax 3, so slide math uses the same `macros.qmd` source.
- **docx**: Word documents do not use a LaTeX preamble or MathJax header, so `include-in-header` has no effect on math rendering for docx. The `docx: default` entry above produces docx output without macro support.

Example math

The table below uses macros defined with `\def` in `macros.qmd`, which work in all output formats that support MathJax or LaTeX:

Description	Code	Output
Greek: alpha	`\a`	α
Greek: beta	`\b`	β
Greek: gamma	`\g`	γ
Greek: lambda	`\l`	λ
Greek: sigma	`\s`	σ

Description	Code	Output
Log-likelihood	`\llik`	ℓ
Independent	`X \ind Y`	$X \perp\!\!\!\perp Y$
IID distributed	`X_i \siid \Normal(\m, \ss)`	$X_i \sim_{\text{iid}} \mathrm{N}(\mu, \sigma^2)$
Vector: mu	`\vm`	μ
Hat beta	`\hb`	$\hat{\beta}$
Sigma-algebra	`\salg`	$\mathcal{S}$
Borel sigma-algebra	`\borel`	$\mathcal{B}$

Color macros

`macros.qmd` defines a one-argument macro for each of the 19 LaTeX base colors (`\red`, `\blue`, `\green`, `\orange`, …), using the `\color` switch form (`{\color{red}{#1}}`), which is supported by MathJax (HTML/RevealJS) and lualatex (PDF):

Description	Code	Output
Red	`\red{x}`	$\red{x}$
Blue	`\blue{y}`	$\blue{y}$
Green	`\green{z}`	$\green{z}$
Orange	`\orange{\mu}`	$\orange{\mu}$
Combined	`\red{x} + \blue{y} = \green{z}`	$\red{x} + \blue{y} = \green{z}$
Nested macros	`X_i \sim_{\red{\ind}} \blue{\Normal}(\orange{\mu_i}, \green{\ss})`	$X_i \sim_{\red{\perp\!\!\!\perp}} \blue{\mathrm{N}}(\orange{\mu_i}, \green{\sigma^2})$

> **Note (docx):** as described above, the `include-in-header` strategy does not load macros for docx output, so the color macros (like all other macros) are unavailable there. For docx, use the include shortcode strategy instead.