Guide for Managing AMORE Website[1]

Ingebjørg A. Iversen

2025-11-18

[1]Thanks to Alina and Heemin for your contributions in developing this guide.

Table of contents

1 Introduction 5

1.0.1 Who this guide is for: 5

1.0.2 Important contacts 5

1.1 How AMORE website is programmed and hosted 5

1.2 Git, Shinyapp, and netlify 5

1.2.1 Git (Version Control & Collaboration) 6

1.2.2 GitHub repository 6

1.2.3 ShinyApps.io (Shiny App Hosting) 6

1.2.4 Netlify (Website Hosting) 6

1.3 RStudio and Quarto 7

1.3.1 The Different Languages 7

1.4 Nettskjema 7

2 Getting started 9

2.0.1 Good to know before starting 9

2.1 Prerequisites 9

2.2 Access Methods to Git - three options 9

2.2.1 Authentication Setup 10

2.2.2 Method 1: For Collaborators (Command Line in bash or powershell) 12

2.2.3 Method 2: or External Contributors (Command Line in bash or powershell - Fork Workflow) 13

2.2.4 Method 3: Method 3: RStudio Git Integration (For Both Collaborators and Contributors) 15

2.3 Dependencies 16

3 Project structure and file explanations 18

3.1 Root files 19

3.1.1 index.qmd 20

3.1.2 _quarto.yml 20

3.1.3 LICENSE . . . 20
3.1.4 README.md . . . 21
3.1.5 .gitignore . . . 21
3.1.6 .node-version . . . 22
3.1.7 Package.json . . . 23
3.1.8 netlify.toml . . . 23
3.1.9 netlify-setup-quarto.sh . . . 24
3.1.10 install-quarto.sh . . . 24
3.2 .quarto/ folder . . . 25
3.3 assets/ folder . . . 26
3.4 pages/ folder . . . 26
3.5 LMAs/ folder . . . 27
3.6 docs/ folder . . . 27
3.7 scripts/ folder . . . 28
3.8 shiny-meta/ folder . . . 28
3.8.1 How app.R works: . . . 29
3.9 _site/ folder . . . 30

4 Styling system SCSS 32
4.0.1 Overview . . . 32
4.0.2 Structure of styles.scss . . . 33

5 Adding new projects 34
5.1 What do the project file contain? . . . 34
5.1.1 Before Building the template . . . 35
5.2 Building the template . . . 36
5.3 Mandatory sections . . . 36
5.3.1 YAML header . . . 36
5.3.2 Title and Information container . . . 39
5.3.3 Abstract . . . 42
5.3.4 Resources . . . 42
5.4 Optional sections . . . 48
5.4.1 Current results . . . 49
5.4.2 Inclusion and exclusion criteria . . . 57
5.4.3 Methodology summary . . . 58
5.4.4 Search strategy . . . 58
5.5 Previewing your work . . . 59

6 Adding new metadata alternatives 61
6.1 Add a new filter category . . . 61
6.1.1 Step 1: Filter tab button . . . 62
6.1.2 Step 2: Filter panels . . . 63
6.1.3 Step 3: Filter section(s) . . . 64
6.1.4 Step 4: Server logic . . . 66
6.1.5 Step 5: Add categorization logic (optional) . . . 72
6.2 Adding a new subcategory to an existing field . . . 73
6.2.1 Overview . . . 73
6.2.2 Step 1: Update checkbox choices . . . 73
6.2.3 Step 2: Add categorization logic (if applicable) . . . 74
6.3 Update the guide document YAML header . . . 75
6.3.1 Location . . . 75
6.3.2 What to update . . . 75
6.3.3 For new top-level categories . . . 76
6.4 Update _lma.template.qmd (Recommended) . . . 76
6.4.1 Why update the template? . . . 76
6.4.2 What to update . . . 76
6.5 Update Nettskjema contact form . . . 77
6.5.1 Which form to update . . . 77
6.5.2 Why update Nettskjema? . . . 77
6.5.3 What to update . . . 77
6.5.4 Steps to update the form: . . . 77
6.5.5 Important considerations . . . 78
6.6 Summary: Complete update workflow . . . 78
6.6.1 1. Update app.R . . . 78
6.6.2 2. Update guide document . . . 78
6.6.3 3. Update _lma.template.qmd (recommended) . . . 78
6.6.4 4. Update Nettskjema . . . 78
6.6.5 Verification checklist . . . 78

7 Troubleshooting 80
7.1 Recovery . . . 80
7.1.1 Recovery Files . . . 80
7.2 Common Problems and Solutions . . . 81
7.2.1 YAML Syntax Errors . . . 83
7.2.2 Styling Problems . . . 83
7.3 Recovering Previous Versions . . . 86
7.4 Recovering from Mistakes . . . 87

8 Website maintenance 88
8.1 Third-Party Services Overview . . . 88
8.1.1 Netlify . . . 88
8.1.2 ShinyApps.io . . . 88
8.1.3 GitHub . . . 89
8.1.4 Nettskjema . . . 89
8.1.5 Outlook.com Email . . . 90
8.1.6 Zenodo . . . 90
8.2 Login Credentials Management . . . 91
8.2.1 Storage Location . . . 91
8.3 Automated Maintenance Script . . . 91
8.3.1 Overview . . . 91
8.3.2 Explanation of script: . . . 91

Chapter 1

Introduction

The AMORE (Active Monitoring of Oxytocin Research Evidence) website is a platform for hosting and managing living meta-analyses for oxytocin research. This guide provides comprehensive instructions for lab members and administrators to manage, update, and maintain the website.

Live Website: https://amore-project.org

1.0.1 Who this guide is for:

- Lab members adding new LMA projects
- Website administrators maintaining platform

1.0.2 Important contacts

Ingebjørg A. Iversen

email: iaiversen@hotmail.com, ingebjai@uio.no

Daniel Quintana

email: daniel.quintana@psykologi.uio.no, danielqu@uio.no

1.1 How AMORE website is programmed and hosted

AMORE was built using RStudio and Quarto. Git is used for version control, collaboration and hosting website in GitHub repository. Netlify is deploying the website and accessing the script from the github repository. Shinyapp host the app.R (backend to the directory hosting living meta-analysis on AMORE). Nettskjema delivers contact forms used as medium between platform users and the AMORE team.

1.2 Git, Shinyapp, and netlify

The AMORE platform consists of three interconnected services:

1.2.1 Git (Version Control & Collaboration)

Repository: iaiversen/AMORE-webpage.

Stores all source code. Tracks changes with commit history. Enables collaboration through pull requests.

Important: Never commit sensitive data (API keys, passwords)

1.2.2 GitHub repository

Repository: AMORE-webpage

Repository URL: https://github.com/iaiversen/AMORE-webpage

Created by github account: iaiversen (Ingebjørg Anjadatter Iversen)

Collaborators: dsquintana (Daniel quintana), AMSartorius (Alina Sartorius), HeeminK (Heemin Kang).

What collaborators (Alina, Dan, Heemin) can do:

- Review, comment, approve, merge and close pull requests by external contributors

What only repository owner can do:

- Add/remove collaborators
- Change repository settings (make public/private, etc.)
- Delete the repository
- Transfer ownership

1.2.2.1 Repository License

AMORE-webpage repository has the MIT license. It gives open access. The LICENSE file is in the root folder (AMORE-webpage).

1.2.3 ShinyApps.io (Shiny App Hosting)

Hosts the Living Meta-Analysis directory (app.R) Free tier limitations: limited active hours, connection timeouts URL: https://meta-oxytocin.shinyapps.io/shiny-meta/ Embedded in Living_meta-analysis_Directory.qmd via iframe

1.2.3.1 Shinyapp.io login credentials (only for BNE lab members)

Info found in Teams –> manuscript –> AMORE –> Website technicals –> AMORE_credentials.xlsx

1.2.4 Netlify (Website Hosting)

Automatically deploys from GitHub Builds site from _site folder after Quarto render Domain: amore-project.org Configuration in netlify.toml

Deployment workflow:

Make changes locally in RStudio. Test by rendering (quarto render or Render button) Commit and push to GitHub. Netlify automatically rebuilds and deploys. For Shiny app: Deploy separately using rsconnect::deployApp() or push publish button in the right top corner of the terminal.

1.2.4.1 Netlify login credentials (only for BNE lab members)

info found In Teams –> manuscript –> AMORE –> Website technicals –> AMORE_credentials.xlsx

1.3 RStudio and Quarto

RStudio is your primary development environment. Install:

R (version 4.0.0+) from CRAN RStudio Desktop from Posit Quarto CLI - Usually bundled with RStudio, or install from quarto.org

1.3.1 The Different Languages

The AMORE website is built using multiple programming languages and frameworks that work together: Quarto is multilingual, Just specify in the code chunk what language is used.

Languages in use:

- Quarto Markdown (.qmd): Primary content files for pages
- SCSS/CSS: Styling and responsive design
- JavaScript: Interactive functionality and client-side behavior
- R: Shiny app backend, data processing, and setup scripts
- HTML: Embedded within Quarto files for custom components Shell scripts: Deployment and automation tasks

1.4 Nettskjema

Ediorial right for nettskjema forms:**

- ingebjai@uio.no
- danielqu@uio.no

The users reach out to the AMORE team through Nettskjema. They find the nettskjema through links on the AMORE website or directly on the contact page.

The Nettskjema forms are:

1. Contact form AMORE
    - Contact button on home page (index.qmd) transfers platform users to this form.
    - Redirects to the other three forms (Propose your project, Update an existing living meta-analysis or Get help or share feedback).
2. Propose your project
3. Get help or share feedback
4. Update an existing living meta-analysis

Emails of responses to any of these forms goes to:

- ingebjai@student.sv.uio.no
- daniel.quintana@psykologi.no

NB! Dan streamlines the nettskjema responses to the AMORE expert steering committee for evaluation of projects or other inquiries.

If you need editorial right to the nettskjema, reach out to Dan.

Chapter 2

Getting started

This section guides you through setting up your development environment, providing the essential setup instructions.

2.0.1 Good to know before starting

Throughout this guide, you'll see code blocks with different labels that indicate language used, or where/how to run the code. An important note is that the bash blocks indicate Git commands, but the interface does not need to be Bash. It needs to be a Git friendly Terminal such as Bash, Powershell, or RStudio Terminal.

bash blocks
{bash} The bash code blocks indicates commands are for command line interfaces. Bash, powershell and RStudio Terminal are all command line interfaces. (I personally use Windows powershell as my terminal). The blocks contain Git commands that work in all mentioned command interfaces.

2.1 Prerequisites

- R (Version 4.0.0 or higher) - programming language
- Rstudio - Integrated Development Environment (IDE) for R.
- Quarto - Document publishing system for creating websites or reports
- Github account - (needs to be authenticated on your computer)
- Git version installed on your computer (not browser version).

2.2 Access Methods to Git - three options

Choose the method that matches your access level and preferred workflow:

Method 1: For Collaborators (Command Line in bash or powershell)

Method 2: For External Contributors (Command Line in bash or powershell - Fork Workflow)

Method 3: RStudio Git Integration (For Both Collaborators and Contributors)

2.2.0.1 Understanding Access Levels

Collaborators: - Have direct write access to the AMORE repository - Can push changes directly to the main repository - Changes go live on the website immediately (after Netlify rebuild) - Can review and merge pull requests from external contributors - To become a collaborator: Contact Ingebjørg Iversen (GitHub: iaiversen) to request access

External Contributors: - Work through the fork workflow (Method 2) - Cannot push directly to the AMORE repository - Must submit pull requests for review - Any collaborator can review and merge your contributions - Ideal for one-time contributions or occasional updates

2.2.1 Authentication Setup

Before you can push changes to the repository, you need to set up authentication on your computer. Even though you've been added as a collaborator on GitHub, your computer needs a way to prove your identity when pushing changes. Once configured, you'll be able to push changes to any GitHub repository you have access to.

You have two options: Personal Access Token (PAT) or SSH keys. Choose one method below.

2.2.1.1 Option 1: Personal Access Token (PAT)

Step 1: Generate a Personal Access Token on GitHub

1. Go to GitHub and log in
2. Click your profile picture (top right) → Settings
3. Scroll down to Developer settings (bottom of left sidebar)
4. Click Personal access tokens → Tokens (classic)
5. Click Generate new token → Generate new token (classic)
6. Give your token a descriptive name (e.g., "AMORE-webpage-access")
7. Set expiration (recommend 90 days or 1 year)
8. Under Select scopes, check the `repo` box (this gives full control of private repositories)
9. Scroll down and click Generate token
10. IMPORTANT: Copy the token immediately - you won't be able to see it again!

Step 2: Use the token when pushing

When you try to push for the first time:

```
git push origin main
```

Git will prompt you for credentials: - Username: Enter your GitHub username - Password: Paste your Personal Access Token (NOT your GitHub password)

Step 3: Store credentials (optional but recommended)

To avoid entering the token every time, you can store it using Git Credential Manager:

On Windows:

```
git config --global credential.helper wincred
```

On Mac:

```
git config --global credential.helper osxkeychain
```

On Linux:

```
git config --global credential.helper store
```

After running this command, the next time you enter your token, it will be saved securely.

2.2.1.2 Option 2: SSH Keys - More convenient for frequent use

Step 1: Check if you already have SSH keys

Open PowerShell (Windows) or Terminal (Mac/Linux) and run:

```
ls ~/.ssh
```

If you see files named `id_rsa` and `id_rsa.pub` (or `id_ed25519` and `id_ed25519.pub`), you already have SSH keys and can skip to Step 3.

Step 2: Generate SSH keys (if you don't have them)

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

Press Enter to accept the default file location. You can optionally add a passphrase for extra security (or press Enter to skip).

Step 3: Copy your public key

On Windows (PowerShell):

```
cat ~/.ssh/id_ed25519.pub | clip
```

On Mac:

```
cat ~/.ssh/id_ed25519.pub | pbcopy
```

On Linux or if the above don't work:

```
cat ~/.ssh/id_ed25519.pub
```

Then manually copy the output.

Step 4: Add SSH key to GitHub

1. Go to GitHub → Settings (click your profile picture)
2. Click SSH and GPG keys (in the left sidebar)
3. Click New SSH key
4. Give it a title (e.g., "My Work Laptop")
5. Paste your public key in the "Key" field
6. Click Add SSH key

Step 5: Test the SSH connection to GitHub

```
ssh -T git@github.com
```

You should see a message like: “Hi [username]! You’ve successfully authenticated…”

That’s it for initial setup! When you clone repositories using SSH URLs (like ‘git@github.com:…‘), they’ll automatically use your SSH keys.

2.2.2 Method 1: For Collaborators (Command Line in bash or powershell)

If you are a direct collaborator with write access:

```
{bash}
#| label: collaborator-initial-setup
#| description: Clone repository and install dependencies for collaborators
#| eval: false
#| echo: true

# Clone the repository directly
git clone https://github.com/iaiversen/AMORE-webpage.git
cd AMORE-webpage



# Install R dependencies
Rscript Setup.R
```

Daily Workflow:

```
{bash}
#| label: collaborator-daily-workflow
#| description: Standard git workflow for collaborators with write access
#| eval: false
#| echo: true

# Get latest changes before starting work
git pull origin main

# Make your changes
# ... edit files ...

# Stage your changes
git add .

# Commit with descriptive message
git commit -m "Add new project: [Project Name]"

# Push directly to main repository
git push origin main
```

To check if you’re a collaborator: - Visit: https://github.com/iaiversen/AMORE-webpage/settings - If you can access Settings, you’re a collaborator - If you get “404 Not Found”, use Method 2 below

---

2.2.3 Method 2: or External Contributors (Command Line in bash or powershell - Fork Workflow)

If you are NOT a direct collaborator:

Initial Setup (One Time):

Step 1: Fork the Repository 1. Go to https://github.com/iaiversen/AMORE-webpage 2. Click the Fork button (top right corner) 3. This creates your own copy at: https://github.com/[YOUR-USERNAME]/AMORE-webpage

Step 2: Clone YOUR Fork

```{bash}
#| label: fork-clone-setup
#| description: Clone your forked repository to your local machine
#| eval: false
#| echo: true

# Clone your fork to your computer
git clone https://github.com/[YOUR-USERNAME]/AMORE-webpage.git
cd AMORE-webpage
```

Step 3: Link to Main Repository

```{bash}
#| label: fork-upstream-setup
#| description: Link your fork to the original repository as upstream
#| eval: false
#| echo: true

# Add the original repository as "upstream"
git remote add upstream https://github.com/iaiversen/AMORE-webpage.git

# Verify your remotes
git remote -v
# Should show:
# origin    https://github.com/[YOUR-USERNAME]/AMORE-webpage.git (your fork)
# upstream  https://github.com/iaiversen/AMORE-webpage.git (main repo)
```

Understanding Your Fork Setup

When you fork and clone, you're working with TWO GitHub repositories:

Your fork ('origin'):

- Your personal copy: 'https://github.com/[YOUR-USERNAME]/AMORE-webpage' - You have full push access here - When you run 'git push', changes go here - Think of this as your workspace

The main AMORE repository ('upstream'):

- The official repository: 'https://github.com/iaiversen/AMORE-webpage' - You CANNOT push directly here - You can only pull updates from here - This is where the live website gets updated from

To get your changes into the main AMORE repository: - Go to GitHub and create a Pull Request (see "Creating a Pull Request" below) - AMORE maintainers review and merge your changes - Only then do your changes appear on the live website

To get updates from the main AMORE repository:

```
git pull upstream main
git push origin main  # Update your fork with the latest changes
```

Regular Workflow (After Initial Setup)

Step 1: Before starting new work - sync with upstream

Always get the latest changes from the main repository first:

```
#| label: fork-sync-upstream
#| description: Sync your local repository with the main AMORE repository
#| eval: false
#| echo: true

# Get latest changes from main AMORE repository
git pull upstream main

# Push these updates to your fork
git push origin main
```

Step 2: Make your changes

Edit files, create new content, etc.

Step 3: Commit and push to YOUR fork

```
#| label: fork-commit-push
#| description: Commit changes and push to your fork
#| eval: false
#| echo: true

# Stage your changes
git add .

# Commit with a descriptive message
git commit -m "Add new project page for [project name]"

# Push to YOUR fork (not the main repository)
git push origin main
```

Step 4: Create a Pull Request

1. Go to your fork on GitHub: ‘https://github.com/[YOUR-USERNAME]/AMORE-webpage‘ 2. You’ll see a yellow banner: “This branch is X commits ahead of iaiversen:main” 3. Click **Contribute** → **Open pull request** 4. Fill in the Pull Request form: - **Title:** Brief description of changes (e.g., “Add meta-analysis project page for oxytocin and trust”) - **Description:** Explain what you changed and why 5. Click **Create pull request

Step 5: Wait for review

- AMORE maintainers will review your changes - They may ask for modifications or clarifications - Once approved, they’ll merge your changes - Your changes will then appear on the live website after Netlify rebuilds it

2.2.4 Method 3: Method 3: RStudio Git Integration (For Both Collaborators and Contributors)

RStudio has built-in Git integration that provides a visual interface for Git operations. This method works for both collaborators (Method 1) and fork users (Method 2).

Method 3: Prerequisites

- Complete the Authentication Setup section above

- Method 2 users only: Complete the fork setup (fork the repository on GitHub first)

Initial Setup: Clone in RStudio

Step 1: Create a new project from version control

1. In RStudio: File → New Project → Version Control → Git
2. Repository URL:
   - Method 1 (collaborators): ‘https://github.com/iaiversen/AMORE-webpage‘
   - Method 2 (fork users): ‘https://github.com/[YOUR-USERNAME]/AMORE-webpage‘
3. Project directory name: Leave as ‘AMORE-webpage‘
4. Create project as subdirectory of: Choose your location (e.g., ‘Documents/‘)
5. Click Create Project

RStudio will download all files and open the existing ‘AMORE.Rproj‘ project with Git already connected.

Step 2: Set up upstream remote (external collaborators)

If you’re using a fork, connect to the main repository:

1. In RStudio: Tools → Terminal → New Terminal
2. Run:

```
git remote add upstream https://github.com/iaiversen/AMORE-webpage.git
```

Daily Workflow in RStudio

You’ll see a Git tab in RStudio (usually upper-right pane).

Making and Saving Changes

1. Edit your files in the project
2. Git tab: Check boxes next to files you want to save
3. Click Commit
4. Write a descriptive commit message
5. Click Commit button
6. Click Push

Understanding What “Push” Does

Collaborators:

- Push sends changes directly to the main AMORE repository

- Changes go live on the website after Netlify rebuilds (usually 2-3 minutes)

Fork users:

- Push sends changes only to YOUR fork

- You must create a Pull Request on GitHub to propose changes to the main repository

- See “Creating a Pull Request” section in Method 2

Getting Latest Changes

Ccollaborators:

- Click the Pull button (blue down arrow) in the Git tab

Fork users:

To get updates from the main AMORE repository:

1. In RStudio: Tools → Terminal → New Terminal

2. Run:

```
git pull upstream main
```

Click Push in the Git tab to update your fork

2.2.4.1 Reopening Your Project

After initial setup:

- File → Open Project → Recent Projects** → select AMORE

- Or File → Open Project → navigate to ‘AMORE.Rproj‘

- Or double-click ‘AMORE.Rproj‘ in your file system

RStudio remembers all Git settings - no reconfiguration needed!

2.3 Dependencies

Install all required packages:

Open `Setup.R` and run the entire script.

```
{r}
#| label: Setup script for dependencies
#| description: copy paste into r console
#| eval: false
#| echo: true

#You need to be in the root folder AMORE
source("scripts/Setup.R") # write this in console
# Or just find the script yoursekf and run it.
```

The setup.R script installs:

Core packages:

- `rmarkdown`, `knitr`, `quarto` - Document rendering

- `shiny`, `rsconnect` - Shiny app deployment
- `DT`, `yaml`, `fs`, `httr`, `jsonlite` - Data handling
- `bslib`, `sass` - Styling
- `tinytex` - LaTeX/PDF support

2.3.0.1 Verify Installation

Test that everything works:

```{bash}
#| label: Verify installation in r console
#| eval: false
#| echo: true

("quarto --version") # Check Quarto system
quarto::quarto_render("index.qmd") # Test rendering a single page
```

Chapter 3

Project structure and file explanations

```bash
#| label: Root folder (AMORE-webpage) and the file structure
#| eval: false
#| echo: true

AMORE-webpage/
├── index.qmd                          # Homepage (root level)
├── _quarto.yml                        # Main configuration file
├── LICENSE
├── README.md
├── .gitignore
├── .node-version
├── package.json
├── netlify.toml                       # Netlify deployment config
├── netlify-setup-quarto.sh            # Quarto setup for Netlify
├── install-quarto.sh                  # Quarto installation script
├── AMORE.Rproj                        # RStudio project file
│
├── .quarto/                           # Quarto build cache (auto-generated)
│   ├── idx/                           # Index files
│   ├── xref/                          # Cross-references
│   ├── listing/
│   ├── preview/
│   └── _freeze/
│
├── assets/
│   ├── favicons/                      # All favicon files
│   │   ├── favicon.ico
│   │   ├── apple-touch-icon.png
│   │   ├── favicon-96x96.png
│   │   ├── site.webmanifest
│   │   └── amore.favicon.ico
│   │
│   ├── images/                        # Logo and visual assets
│   │   ├── logo.unfinished_nowhite- Copy.png
│   │   ├── LiMAs.jpg
│   │   └── [other image files]
│   │
│   └── styles/
│       └── styles.scss                # Main stylesheet
```

```
│
├── pages/                              # All content pages
│   ├── about.qmd
│   ├── contact.qmd
│   ├── guidelines.qmd
│   ├── Living_meta-analysis_Directory.qmd
│   ├── Protocol_checklist.qmd
│   ├── Resources.qmd
│   └── Standardization.qmd
│
├── LMAs/                               # Living Meta-Analysis projects
│   ├── _lma-template.qmd               # Template for new LMAs
│   └── project pages files
│
├── docs/                               # Documentation (working files)
│   ├── _AMORE_Guide.qmd                # Website management guide
│   ├── _recoveryfile.app.qmd
│   └── _recoveryfile.qmd
│
├── scripts/                            # Utility scripts
│   └── Setup.R                         # R dependencies installer
│
├── shiny-meta/                         # Shiny application
│   ├── app.R                           # Main Shiny app
│   ├── deploy.R                        # Deployment script
│   ├── README.md                       # Shiny app documentation
│   ├── .Renviron                       # Saves shiny meta tokens
│   │
│   ├── rsconnect/                      # Shiny deployment config
│   │   └── shinyapps.io/
│   │       └── meta-oxytocin/
│   │           └── shiny-meta.dcf
│   │
│   └── www/                            # Shiny app assets
│       └── amore.favicon.ico
│
└── _site/                              # Generated website (DO NOT EDIT)
    ├── index.html
    ├── search.json
    ├── assets/
    ├── pages/
    ├── LMAs/
    ├── shiny-meta/
    └── site_libs/
```

3.1 Root files

Root files are located directly in the `AMORE-webpage/` folder. These files control the fundamental behavior, configuration, and metadata of the entire website project. They define how the site is built, deployed, and maintained.

Key characteristics of root files:

- Control project-wide settings and behavior
- Required for the website to function properly

- Should be modified carefully as changes affect the entire site
- Most are configuration or metadata files (not content)

3.1.1 index.qmd

The homepage of the AMORE website. This is the first page visitors see when they navigate to https://amore-project.org.

Type: Quarto markdown document (content file)

CSS classes used directly in index.qmd:

- `.content-section` - Main container with transparent background on gradient
- `.header-logo` - Logo image styling (centering, max-width)
- `.subtitle-text` - Centered white subtitle text with specific font size
- `.primary-button` - Call-to-action button styling
- `.contact-buttons` - Button container for centering
- `.BNL-text` - Text styling for lab attribution

Global styles that also affect this page:

- `body` - Gradient animated background (visible behind `.content-section`)
- Typography rules (`p`, `a`, headings)
- `.primary-button:hover` - Hover effects on the CTA button
- Responsive breakpoints that adjust layout on mobile/tablet

3.1.2 _quarto.yml

The main configuration file that controls how the entire website is built and structured.

Type: YAML configuration file

Important notes:

- Changes here affect the entire site
- Always test after editing with `quarto preview`
- Incorrect YAML indentation will break the site
- Navbar order controls menu item order

3.1.3 LICENSE

Defines the legal terms under which the AMORE website code and content can be used, modified, and distributed.

Type: Plain text legal document

License: MIT open source license. MIT license explanations.

3.1.4 README.md

Brief description of the AMORE website project displayed on the GitHub repository page.

Type: Markdown document

Purpose:

- First thing people see on GitHub
- Explains what the project is

3.1.5 .gitignore

Purpose: Tells Git which files and folders to ignore (not track or upload to GitHub).

Type: Plain text configuration file

Purpose:

- Keeps repository clean
- Prevents uploading generated/temporary files
- Reduces repository size
- Protects sensitive information

```
{r}
#| label: .gitignore contents
#| eval: false
#| echo: true

# R files
*.log
*.tex
*.aux
.Rproj.user
.Rproj.user/
.Rhistory
.RData
.Ruserdata
*.Rproj

# Quarto and website specific files
_site/
/.quarto/
**_cache/
**.bak
*_error.scss
*.backup
*.ipynb
/_site/
/_freeze/
/_book/
/*_files/

# Generated HTML files in SOURCE directories (not _site)
```

```
/*.html
/pages/*.html
/LMAs/*.html
/docs/*.html

# Generated dependency folders in SOURCE directories
/*_files/
/pages/*_files/
/LMAs/*_files/
/docs/*_files/



# OS files
.DS_Store
Thumbs.db

# Test files

docs/_recoveryfile.qmd
docs/_recoveryfile.app.qmd

# Generated files
_quarto_internal_scss_error.scss

# Backup files
*.backup
*.bak
```

Why exclude these:

- RStudio files are user-specific
- _site/ is generated, not source code
- Cache files speed up local rendering but aren't needed in repo
- Reduces repository size

Never add to .gitignore:

- Source files (.qmd, .R, .scss)
- Configuration files (_quarto.yml, netlify.toml)
- Assets (images, logos)

3.1.6 .node-version

Specifies which Node.js version to use for building the website.

Type: Plain text version file

Purpose:

- Ensures consistent Node.js version across environments

- Used by Netlify during deployment
- Prevents version compatibility issues

Current content:

- Single line with Node.js version number (e.g., "16")

When to edit:

- Updating Node.js requirements
- Fixing deployment issues related to Node.js version
- Usually leave unchanged unless necessary

3.1.7 Package.json

Defines npm scripts and Node.js dependencies for building and deploying the website.

Type: JSON configuration file

Purpose:

- Contains build scripts for Netlify automated deployment
- Manages project metadata
- Defines commands to install Quarto and render the site

Key sections:

- `name:` - Project identifier "amore-project"
- `version:` - Project version number
- `scripts:` - Build commands
  - `prebuild` - Installs Quarto before building
  - `build` - Renders the Quarto site

3.1.8 netlify.toml

Configuration file for Netlify deployment settings.

Type: TOML configuration file

Purpose:

- Tells Netlify how to build and deploy the website
- Specifies build commands and output directory
- Sets environment variables

Key settings:

- `publish = "_site"` - Where built files are located
- `command = "npm run build"` - Build command to execute
- `NODE_VERSION = "16"` - Node.js version for building

3.1.9 netlify-setup-quarto.sh

Bash script that installs Quarto on Netlify's build servers.

Type: Shell script (`.sh`)

Purpose:

- Downloads and installs specific Quarto version
- Runs during Netlify build process
- Ensures Quarto is available for rendering

Key actions:

- Downloads Quarto v1.3.450 tarball
- Extracts and installs to `~/.local/quarto`
- Creates symbolic link to binary
- Verifies installation

Version note:

- Script uses v1.3.450 for Netlify deployment
- Local development can use newer Quarto versions
- Version mismatch did not cause issued for myself, but if rendering issues, perhaps look into this

Important:

- Called by `package.json` prebuild script
- Must be executable (`chmod +x`)

3.1.10 install-quarto.sh

Bash script that installs Quarto locally (alternative installation method).

Type: Shell script (`.sh`)

Purpose:

- Downloads and installs Quarto v1.6.40
- For local development setup
- Installs to user's home directory

Key actions:

- Downloads Quarto tarball from GitHub

- Extracts to `$HOME/.local/bin`., a temporary cache directory created during local Quarto installation. The $HOME directory is safe to delete.

  Type: System directory
- Adds to PATH
- Verifies installation with `quarto --version`

When to use:

- Setting up development environment
- Alternative to downloading from Quarto website
- Linux/Unix systems

When to edit:

- Updating to newer Quarto version

Note:

- Different version than `netlify-setup-quarto.sh`
- Not used in deployment (local only)

3.2 .quarto/ folder

Quarto's build cache and internal processing files (auto-generated).

Type: System cache directory

Purpose:

- Stores rendering cache for faster builds
- Contains cross-reference indexes
- Holds frozen computation results
- Internal Quarto processing files

Structure:

Important notes:

- Auto-generated - recreated on each render
- Already excluded in `.gitignore`
- NB! Never edit manually
- Safe to delete if troubleshooting build issues
- Speeds up subsequent renders

When to delete:

- Fixing strange rendering errors
- Clearing cache after major changes
- Will be recreated automatically on next `quarto render`

3.3 assets/ folder

Contains all static files used across the website (images, styles, icons).

Seperated into three subfolders:

- favicorns
  - Contains all browser icons and Progressive Web App (PWA) assets.
  - Utilized in _quarto.yml file
- images
  - all images across the website are saved in this folder
- styles
  - Contains the styles.scss sheet that instructs the rendering process of design rules for the website

3.4 pages/ folder

Contains all main content pages of the website (except homepage).

Type: Content directory

Purpose:

- Houses all `.qmd` files for site pages
- Each file = one webpage
- Linked from navbar in `_quarto.yml`

Files:

- `about.qmd` - About AMORE, mission, team, steering committee
- `contact.qmd` - Contact form links
- `guidelines.qmd` - Step-by-step workflow for publishing LMAs
- `Living_meta-analysis_Directory.qmd` - Embedded Shiny app directory
- `Protocol_checklist.qmd` - Interactive checklist tool
- `Resources.qmd` - External resources and tools
- `Standardization.qmd` - AMORE framework requirements

3.5 LMAs/ folder

Contains all Living Meta-Analysis project pages.

Type: Content directory (project pages)

Purpose:

- Individual `.qmd` file for each LMA project
- Template file for creating new projects
- Fetched by Shiny app for directory listing

YAML metadata structure: Each project file contains metadata that the Shiny app reads:

Important notes:

- Shiny app reads these files via GitHub API
- Metadata must follow template structure
- Files starting with `_` are excluded from rendering

3.6 docs/ folder

Contains internal documentation and working files (not published to website).

Type: Documentation directory

Purpose:

- Working documents for website management
- Draft files and recovery backups
- Internal guides and notes

Files:

- `_AMORE_Guide.qmd` - Website management guide (this document)
- `_recoveryfile.qmd` - Backup/recovery file
- `_recoveryfile.app.qmd` - Backup/recovery file

Important notes:

- Files starting with `_` are not rendered to website
- Already excluded in `.gitignore` (for `.html` versions)
- For internal use only
- Not linked in navbar
- Safe to add more documentation files here

When to use:

- Creating internal documentation
- Draft content before moving to pages/
- Testing new features
- Reference materials for maintainers

3.7 scripts/ folder

Contains utility scripts for project setup and maintenance.
Type: Utility scripts directory
Purpose:

- Automate setup tasks
- Maintenance scripts
- Helper utilities

Files:

- `Setup.R` - Installs all required R packages and dependencies

Key functions in Setup.R:

- Installs R packages (rmarkdown, knitr, shiny, quarto, etc.)
- Installs TinyTeX for PDF generation
- Checks for existing installations before installing
- Loads required libraries

3.8 shiny-meta/ folder

Contains the Shiny application for the Living Meta-Analysis Directory.
Type: Shiny app directory
Purpose:

- Interactive searchable directory of all LMA projects
- Fetches project data from GitHub
- Filters and displays LMA metadata
- Deployed on shinyapps.io

Structure:

```{bash}
#| label: shiny-meta directory
#| eval: false
#| echo: true

shiny-meta/
├── app.R           # Main Shiny application code
├── deploy.R        # Deployment script for shinyapps.io
├── .Renviron       # Keeps shiny-meta tokens safe
├── README.md       # Shiny app documentation
├── rsconnect/      # Deployment configuration
└── www/            # Static assets (favicon)
```

Key files:

- `app.R` - Complete UI and server logic
- `deploy.R` - Contains shinyapps.io account credentials
- `rsconnect/` - Auto-generated deployment settings
- .Renviron **Security critical file** containing:
  * 'SHINYAPPS_ACCOUNT' - Your shinyapps.io account name * 'SHINYAPPS_TOKEN' - Authentication token for deployment * 'SHINYAPPS_SECRET' - Secret key for deployment

3.8.1 How app.R works:

1. app.R fetches `.qmd` files from `LMAs/` folder via GitHub API
2. Extracts YAML metadata from each project
3. Provides filtering by outcomes, population, methodology
4. Displays paginated results with links to project pages

- Adding New Filter Categories
  - [To be expanded: Updating the Shiny app to include new metadata fields]

3.8.1.1 Deployment:

- Hosted at: https://meta-oxytocin.shinyapps.io/shiny-meta/
- Non-paid server at shinyapps.io, paid server will improve page loading
- Embedded in: `pages/Living_meta-analysis_Directory.qmd`

3.8.1.2 .Renviron

This file is personal with the tokens from the shinyapp account needed to deploy the app. All users that will deploy anew the app.R to the server on the shinyapp account meta-oxytocin must create their own .Renviron file with personal token and secret. Which is collected by login in to the account at shinyapp.io.

The meta-oxytocin account on shinyapp.io from Posit uses unpaid version, to enable multiple account users, the account subscription must be upgraded to professional plan.

Setting up deployment credentials for shinyapp:

1. Create a ‘.Renviron‘ file in the ‘shiny-meta/‘ directory: “‘bash # .Renviron file structure SHINYAPPS_ACCOUNT=your-account-name SHINYAPPS_TOKEN=your-token-here SHINYAPPS_SECRET=you secret-here “‘
2. Get your credentials from shinyapps.io: * Log in to https://www.shinyapps.io/ * Navigate to Account meta-oxytocin → Tokens * Show/Generate token * Copy the account name, token, and secret
3. In R, run ‘deploy.R‘ which will use these credentials

Important notes:

- Updates automatically when new LMA files added to GitHub
- Requires shinyapps.io account for deployment

3.9 _site/ folder

Contains the generated/rendered website files (auto-generated, do not edit).

Type: Build output directory

Purpose:

- Final HTML files ready for deployment
- Compiled CSS and JavaScript
- Copied assets and resources
- Generated during build process

Structure:

```{bash}
#| label: _site directory
#| eval: false
#| echo: true

_site/
├── index.html          # Rendered homepage
├── search.json         # Search index
├── assets/             # Copied assets
├── pages/              # Rendered page HTML files
├── LMAs/               # Rendered LMA project pages
├── shiny-meta/         # Copied Shiny app files
└── site_libs/          # Quarto dependencies (Bootstrap, JS)
```

Important notes:

- Auto-generated - recreated on every `quarto render`
- Never edit files here - changes will be overwritten
- Already excluded from Git (in `.gitignore`)

Two separate build processes:

Local build (your computer):

- Purpose: Preview and test changes
- Command: `quarto preview` or `quarto render`
- Creates: `_site/` folder locally
- Result: For testing only, not deployed

Netlify build (production):

- Purpose: Deploy live website
- Triggered: Automatically on GitHub push
- Process: Netlify clones repo → runs `npm run build` → generates fresh `_site/`
- Result: This is what visitors see at amore-project.org

Chapter 4

Styling system SCSS

4.0.1 Overview

The AMORE website uses SCSS (Sassy CSS) for styling, which provides more power and flexibility than regular CSS. The main stylesheet is located at `assets/styles/styles.scss`. This file is automatically compiled to CSS when you render the Quarto site.

Here you find scss rules

What styles.scss controls:

- Global variables (colors, fonts)
- Typography system
- Layout components (hero sections, grids, cards)
- Navigation and footer
- Interactive elements (tabs, filters, buttons)
- Page-specific styles (index, contact, project pages)
- Responsive breakpoints (mobile, tablet, desktop). This means the layout adjusts to different screen sizes.
- Animations (gradient shifts, hover effects)

Linked in:

- `_quarto.yml` under `theme:` and `css:`
- Applied to all pages automatically

Important notes:

- Changes affect entire website
- Test thoroughly after edits and before commiting
- Uses SCSS features (variables, nesting)

- Compiled during `quarto render`

File Paths in SCSS

When referencing images or other files in your SCSS, understanding file paths is crucial:

```
{scss}
#| label: file path
#| eval: false
#| echo: true

/* CORRECT - Goes up one folder level from assets/styles/ to assets/ */
background-image: url('../images/logo.png');

/* INCORRECT - Would look in assets/styles/images/ */
background-image: url('images/logo.png');
```

4.0.2 Structure of styles.scss

The file is organized into logical sections marked with comments:

1. Global Variables - Colors, fonts, and sizing used throughout
2. Body & Animations - Background effects and page-level styling
3. Typography - All text styling (headings, paragraphs, links)
4. Components - Reusable pieces (buttons, cards, navigation)
5. Page-Specific - Styles for particular pages (contact, about, etc.)
6. Responsive Design - Adjustments for different screen sizes

- Advanced Customization
  - This section covers advanced modifications for experienced users.
- Custom CSS Modifications
  - [To be expanded: How to modify styles.scss for custom designs]

Chapter 5

Adding new projects

This section guides you through adding a new living meta-analysis project to the AMORE website. Each project requires a `.qmd` file with properly formatted metadata that allows the Shiny app directory to filter and display the project correctly.

Follow AMORE-webpage –> LMAs –> _lma-template.qmd

_lma-template.qmd is a template of project file .qmd. The template is especially useful to present how code chunks are related to each other and how they must be formatted. The template presents HOW the code chunks are formatted and structured, while this guide gives instructions for use and breaking down the code chunks with explanations.

1. Locate the template: Find `_lma-template.qmd` in the LMAs folder of your project directory
2. Create your project file:
    - Duplicate the template file and rename it with a descriptive name.
    - Naming convention: Use lowercase letters with underscores, no spaces (Correct: `smith_social_cognition.qmd` | Incorrect: `Smith Social Cognition.qmd`)
3. Save location: Keep the new file in the LMAs folder alongside other project files
4. Alternatively: create new .qmd file that you save in the LMAs folder. Write scripts yourself or copy paste from this guide or template the necessary code components that you want to include.

Do NOT push to GitHub repository before the project page is finished without adding a prefix '_' to the file name like this _example_project_file.qmd. The prefix prevents the page from rendering and the _quarto.yml have a specific command to ignore files with prefixes at the beginning of the file name. However, with the prefix, the file loses access to the style instructions from styles.scss, this means that the prefix should only be added after the preview but before pushing to Git repository.

5.1 What do the project file contain?

The project .qmd files consists of sections that must be filled and structured correctly. The sections are divided into mandatory and optional sections.

5.1.0.1 Mandatory sections (must be included):

- YAML header - Contains all metadata for filtering and display in the directory
- Information container - The top blue title panel with project details (timeline, team, links)
- Abstract - Brief overview of the meta-analysis
- Resources - Links to preregistration, data, scripts, and publications

5.1.0.2 Optional sections (include if relevant):

- Current Results
- Methodology Summary
- Inclusion and Exclusion Criteria
- Search Strategy
- Study Characteristics
- References

5.1.1 Before Building the template

Before building the template understanding the YAML header format rules are necessary. The YAML header as it should be included is presented in the next section.

```{yaml}
#| label: YAML header format rules explanations
#| description: This is not supposed to be inlcuded anywhere! It is for explanatory purposes.
#| eval: false
#| echo: true

# The YAML format is very important. Incorrect indentation will result in error, or the filter system
 ↪  of the app.R (directory of projects) will be unable to correctly read the neccessary information.

## if a metadata class is irrelevant for the project there are several steps to ignore the class:
oxytocin: #example class
  route: null           # ← Won't show
  dosage: NA            # ← Won't show
  assessment_method:    # ← Empty/missing, won't show
                        # remove class altogether


YAML array format: # All three options should in theory work. If error occur, switch method.
# List with dashes without quotes
analytical_framework:
  - Frequentist
  - Bayesian
  - Mixed methods

# List with dashes AND quotes
analytical_framework:
  - "Frequentist"
  - "Bayesian"
  - "Mixed methods"
```

```yaml
# Inline array with brackets AND quotes
analytical_framework: ["Frequentist", "Bayesian", "Mixed methods"]

# ⚠ DO NOT MODIFY the format section at the bottom
# This section is required for proper page rendering:
format:
  html:
    page-layout: full
    toc: true
    # ... rest of format settings
```

5.2 Building the template

Follow this section to include all mandatory and relevant optional sections for the projects.

5.3 Mandatory sections

5.3.1 YAML header

the YAML header is the metadata section at the very top of each project file that controls how the meta-analysis project appears and gets filtered in the AMORE directory. The YAML header in not visible on the project page.

How YAML Metadata Works:

The metadata you choose determines how your project is ranked in search results - all projects appear, but those matching more filter criteria rank higher. Each metadata field that matches a user's selected filter adds 1 point to your project's relevance score. The relevance points are not presented anywhere, they are part of the backend ranking system. Some categories like "Mixed methods", "Variable dosage", and "Mixed age groups" are parent categories that automatically match all specific options within that category, so listing both the parent and specific values is redundant and doesn't increase your score.

Example: If your project uses both Frequentist and Bayesian methods, analytical_framework: "Mixed methods" gives the same score as listing all three options when users filter by any analytical framework.

The YAML header must contain the Title and Format field, the other fields are voluntary to include. However there more metadata fields included the better visibility the living meta-analysis project will have.

If you want to include new metadata options beyond the given alternatives this must first be integrated in the system. The new alternative must be added to app.R, the _lma-template.qmd and nettskjema. How to do this is explained in section Adding new metadata alternatives in this guide.

When research groups propose new projects through nettskjema "Propose your project", they are instructed to VOLUNTARY choose metadata options for their project to increase their visibility. If they choose not to answer this part of the form leave the sections blank.

```yaml
{yaml}
#| label: YAML header options
#| description: This should be at the very beginning of the project file. The filters options are
 ↪  matched with app.R. If you want to include new options this must also be added to the app's filter
 ↪  logic.# When research groups propose new projects through nettskjema "Propose your project", they
 ↪  are instructed to VOLUNTARY choose metadata options for their project to increase their
 ↪  visibility. If they choose not to answer this part of the form leave the sections blank or write
 ↪  null.
#| eval: false
#| echo: true


---
title: "Write title here"
analytical_framework: # Choose one or more
  - "Frequentist" # The meta-analysis performed a frequentist meta-analysis
  - "Bayesian" # The meta-analysis performed a bayesian meta-analysis
  - "Mixed methods" # the meta-analysis used a mix of both frequentist and bayesian methods. This is a
 ↪  parent category.
oxytocin:
  intervention: # if applicable, choose one or more:
    - "Intranasal oxytocin administration" # The meta-analysis investigates research question on
 ↪  intranasal oxytocin administration
    - "Oral oxytocin administration" # The meta-analysis investigates research question on oral
 ↪  oxytocin administration
    - "Intravenous/injection oxytocin administration" # The meta-analysis investigates research
 ↪  question on intravenous or injected oxytocin administration
    - "Environmental/behavioral oxytocin manipulation" # The meta-analysis investigates research
 ↪  question where oxytocin is measured after enviornmental or behavioral oxytocin manipulation (e.g.
 ↪  giving a hug, or gazing into your dogs eyes)
    - "Perinatal oxytocin exposure" # refers to studies examining the effects on infants/children who
 ↪  were exposed to synthetic oxytocin during the perinatal period (around birth) - typically because
 ↪  their mothers received oxytocin to induce or augment labor.
  assessment_method: # if applicable, choose one or more:
    - "Behavioral assessment" # the meta-analysis investigates behavioral outcomes associated or
 ↪  affected by oxytocin (like trust, social interaction, emotional responses)
    - "Physiological response" # the meta-analysis investigates physiological responses outcomes (e.g.
 ↪  heart rate, blood pressure)
    - "Biological sample collection" # the meta-analysis investigates outcomes where oxytociin was
 ↪  assessed with biological samples (like blood, urine, saliva, hormones)
    - "Genetic studies" # the meta-analysis investigates genetic variations related to oxytocin,
 ↪  and/or variations in genes associated with oxytocin related to other outcomes such as behavioral
 ↪  or physiological.
    - "Neural/imaging measurement" # the meta-analysis investigates neural/imaging measurments (such
 ↪  as eeg, MRI) related to oxytocin outcomes (e.g. brain activity after oxytocin administration, or
 ↪  gray matter volume in dementia pateients with different baseline oxytocin measures)
  route: # if applicable, choose one or more:
    - "Central" # Meta-analysis investigates oxytocin outcomes related to the central nervous system
 ↪  or administration that reaches the central nervous system (e.g intranasal oxytocin is believed to
 ↪  reach the brain and the central nervous system, oxytocin pathways in the brain is the central
 ↪  nervous system)
    - "Peripheral" # Meta-analysis investigates
    - "Various administration routes" # This is a parent category. The meta-analysis investiagtes
 ↪  oxytocin across both routes. Or there is uncertainty about which route will be targeted (e.g.
 ↪  research question investigating secondary effects of centrally administered oxytocin on peripheral
 ↪  oxytocin system).
  dosage: # if applicable, choose one or more:
    - "8 IU"
```

```
    - "16 IU"
    - "24 IU"
    - "32 IU"
    - "40 IU"
    - "Variable dosage" # if the study includes multiple dosage interventions this parent category can
↪  be used
population:
  status: # if applicable, choose one or more:
    - "Healthy" # The meta-analysis investigates research questions on a healthy population
    - "Clinical" # The research question(s) investigates a clinical population.
    - "Mixed" # The research question(s) under investigation is equally central to healthy and
↪  clinical populations
  age_group: # Age ranges not specified # if applicable, choose one or more:
    - "Children"
    - "Adolescents"
    - "Adults"
    - "Older Adults"
    - "Mixed Age Groups" # Parent group that covers the four other age groups. Would recommend using
↪  if age ranges in meta-analysis covers more than three groups (e.g. adolescents, adults, older
↪  adults) or if groups are poorly defined.
  clinical_type: # Describe if applicable (e.g., "Autism Spectrum Disorder", "Depression")
outcomes:
  biological: # if applicable, choose one or more:
    - "Cardiovascular" # research question investigates oxytocin related to cardiovascular outcomes
↪  (e.g., heart rate, blood pressure, heart rate variability)
    - "Neuroendocrine" # research question investigates oxytocin related to neuroendocrine outcomes
↪  (e.g., cortisol levels, sex hormones, thyroid function)
    - "Neurological" # research question investigates oxytocin related to neurological outcomes (e.g.,
↪  neurotransmitter function, neurodegenerative markers, event-related potentials)
    - "Metabolic" # research question investigates oxytocin related to metabolic outcomes (e.g.,
↪  glucose metabolism, insulin sensitivity, weight regulation, appetite hormones)
    - "Immune and Inflammatory" # research question investigates oxytocin related to immune system
↪  function or inflammatory markers (e.g., cytokine levels, immune cell activity, inflammatory
↪  biomarkers)
    - "Pain and Sensory" # research question investigates oxytocin related to pain perception, pain
↪  thresholds, or sensory processing outcomes
    - "Sleep and Circadian" # research question investigates oxytocin related to sleep physiology or
↪  circadian rhythm (e.g., sleep architecture, melatonin levels, circadian disruption)
  psychological_behavioral: # if applicable, choose one or more:
    - "Mood and Emotion" # research question investigates oxytocin related to emotional states, mood
↪  regulation, or emotional reactivity (e.g., anxiety, depression symptoms, emotional recognition)
    - "Cognition and Memory" # research question investigates oxytocin related to cognitive functions
↪  or memory performance (e.g., working memory, attention, learning, recall)
    - "Stress and Coping" # research question investigates oxytocin related to stress responses,
↪  stress regulation, or coping mechanisms (e.g., perceived stress, stress reactivity, resilience)
    - "Eating and Appetite" # research question investigates oxytocin related to eating behavior,
↪  appetite regulation, or food intake (e.g., caloric intake, food preferences, satiety)
    - "Risk and Decision-Making" # research question investigates oxytocin related to risk-taking
↪  behavior, decision-making processes, or judgment (e.g., financial decisions, risk assessment)
    - "Sleep Behavior and Quality" # research question investigates oxytocin related to subjective
↪  sleep quality, sleep patterns, or sleep-related behaviors (e.g., sleep duration, insomnia
↪  symptoms, sleep disturbances)
    - "Bonding and Attachment" # research question investigates oxytocin related to parent-infant
↪  bonding, romantic attachment, or social bonding behaviors (e.g., maternal sensitivity, attachment
↪  security, relationship quality)
    - "Trust and Cooperation" # research question investigates oxytocin related to trust behaviors,
↪  cooperative behaviors, or prosocial actions (e.g., trust game performance, sharing behavior,
↪  reciprocity)
```

```
    - "Communication and Empathy" # research question investigates oxytocin related to social
 ↪  communication abilities, empathic responses, or perspective-taking (e.g., empathy scores, theory
 ↪  of mind, social understanding)
    - "Aggression and Conflict" # research question investigates oxytocin related to aggressive
 ↪  behavior, conflict resolution, or defensive responses (e.g., aggressive tendencies, hostility,
 ↪  in-group/out-group behavior)
  clinical: # if applicable, choose one or more:
    - "Neurodevelopmental" # research question investigates oxytocin related to neurodevelopmental
 ↪  disorder symptoms or outcomes (e.g., autism spectrum disorder, ADHD, developmental delays)
    - "Mood Disorders" # research question investigates oxytocin related to mood disorder symptoms or
 ↪  treatment outcomes (e.g., depressive disorders, bipolar disorders, anxiety disorders)
    - "Psychotic Disorders" # research question investigates oxytocin related to psychotic disorder
 ↪  symptoms or outcomes (e.g., schizophrenia symptoms, paranoia, social cognition in psychosis)
    - "Addiction and Substance Use" # research question investigates oxytocin related to substance use
 ↪  disorders, addiction behaviors, or withdrawal symptoms (e.g., craving, relapse, substance use
 ↪  patterns)
    - "Eating Disorders" # research question investigates oxytocin related to eating disorder symptoms
 ↪  or pathology (e.g., anorexia nervosa, bulimia nervosa, binge eating disorder symptoms)
dois:
  preregistration: "10.XXXX/template-prereg"
  preprint: "10.XXXX/template-preprint"
  publication: "10.XXXX/template-publication"
format: ### NB!  The format field must be included.
  html:
    page-layout: full
    toc: true
    toc-title: "On this page"
    toc-location: right
    toc-depth: 3
---
```

5.3.2 Title and Information container

The title and information container is the first visible section of the project page. It contains the projects main title and information such as authors, timeline and contact. Insert correct information in the [brackets].

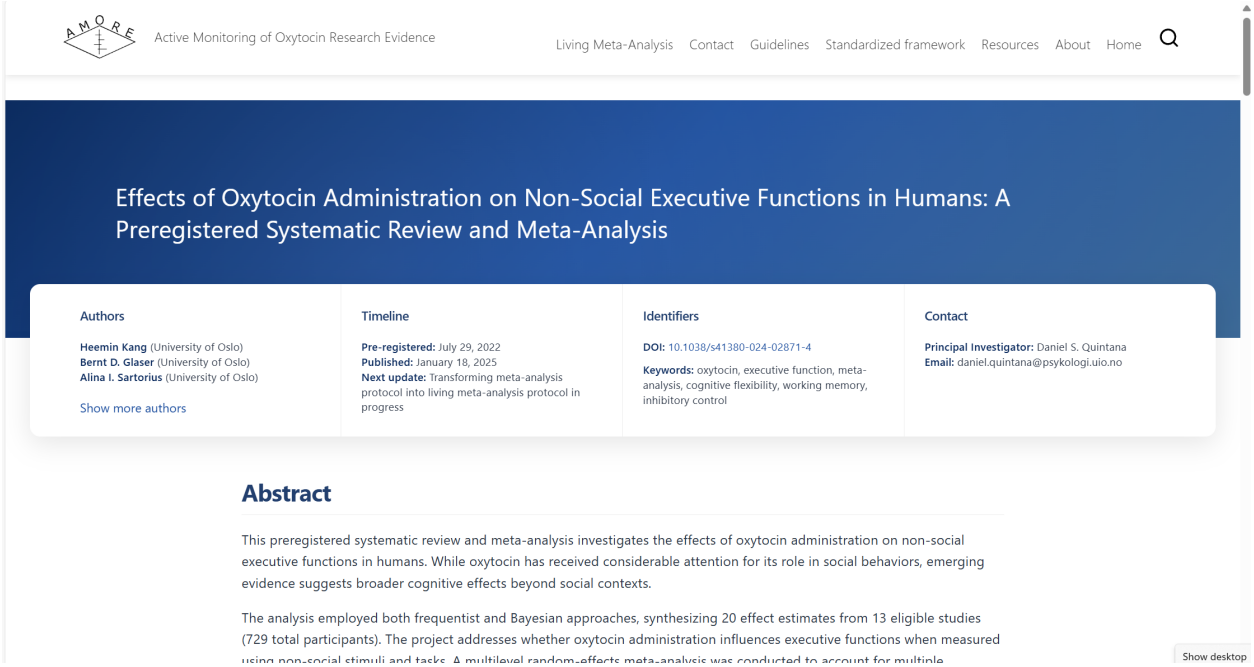


Figure 5.1: Reference image: Title and information container

```
{r}
#| label: information container
#| description: This code chunk is the title and the information box visible under the project hero
 ↪  title. Insert the required information inside the [brackets]
#| eval: false
#| echo: true


::: {.project-hero-section}
::: {.project-hero-content}
::: {.project-hero-title}

[Write title here]

:::
:::
:::

::: {.project-info-panel}
::: {.project-info-grid}

::: {.info-card .authors-card}
### Authors
<div id="author-list">
  <div id="visible-authors">
    <strong>[First Author Name]</strong> ([Affiliation])<br> ## Write authors and affiliations inside
     ↪  the brackets []
    <strong>[Second Author Name]</strong> ([Affiliation])<br>
    <strong>[Senior Author Name]</strong> ([Affiliation])
  </div>
```

```
  <div id="hidden-authors" style="display:none">
    <strong>[Fourth Author]</strong> ([Affiliation])<br> ## Write authors and affiliations inside the
     ↪  brackets []
    <strong>[Fifth Author]</strong> ([Affiliation])<br>
    <strong>[Sixth Author]</strong> ([Affiliation])<br>
    <strong>[Seventh Author]</strong> ([Affiliation])<br>
    <strong>[Eighth Author]</strong> ([Multiple Affiliations | Separated by Pipes])<br>
    <strong>[Additional Authors as needed...]</strong><br>
  </div>

  <a href="#" id="author-toggle" class="toggle-link" onclick="toggleAuthors(); return false;">+ Show
   ↪  more authors</a>

  <script>
  function toggleAuthors() {
    var hiddenAuthors = document.getElementById("hidden-authors");
    var toggleButton = document.getElementById("author-toggle");

    if (hiddenAuthors.style.display === "none") {
      hiddenAuthors.style.display = "block";
      toggleButton.textContent = "- Show fewer authors";
    } else {
      hiddenAuthors.style.display = "none";
      toggleButton.textContent = "+ Show more authors";
    }
  }
  </script>
</div>
:::

::: {.info-card .timeline-card} ## Fill information inside the [] brackets
### Timeline
**Pre-registered:** [Month DD, YYYY]
**Last update:** [Status of current analysis]
**Next update:** [Planned date or "To be determined" (XX-month update cycle)]
:::

::: {.info-card .identifiers-card} ## Fill information inside the [] brackets
### Identifiers
**DOI:** [https://doi.org/ACTUAL_DOI](https://doi.org/ACTUAL_DOI)

**Keywords:** [keyword1], [keyword2], [keyword3], [methodology], [population]
:::

::: {.info-card .contact-card} ## Fill information inside the [] brackets
### Contact
**Principal Investigator:** [PI Full Name]
**Email:** [pi.email@institution.edu](mailto:pi.email@institution.edu)
:::

:::
:::
```

5.3.3 Abstract

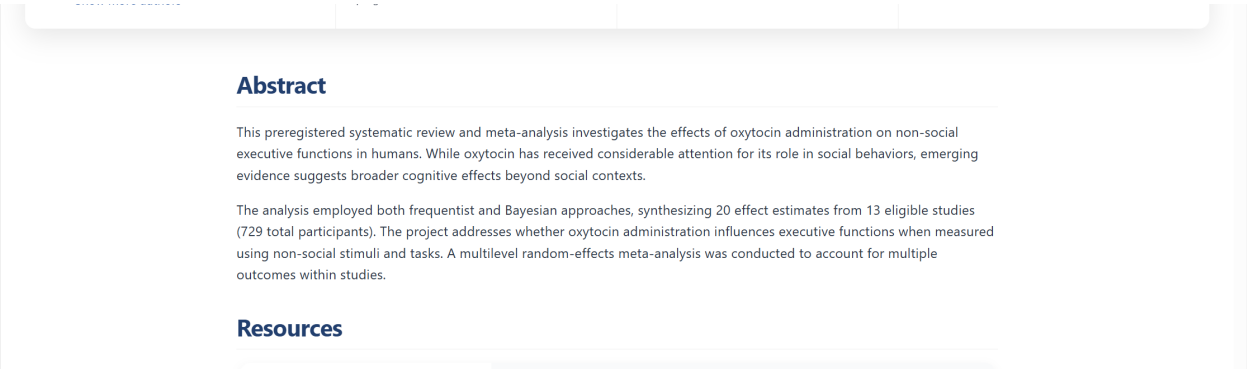


Figure 5.2: Reference image: Abstract

```
{r}
#| label: Abstract section of main content area
#| description: This is where you fill in the abstract. It should be the first header after the
 ↪  information container box.
#| eval: false
#| echo: true

## Abstract {#abstract}

write abstract under abstract header
```

5.3.4 Resources

The resources box is where all materials connected to the project is linked. The resources box is structured by publication version.

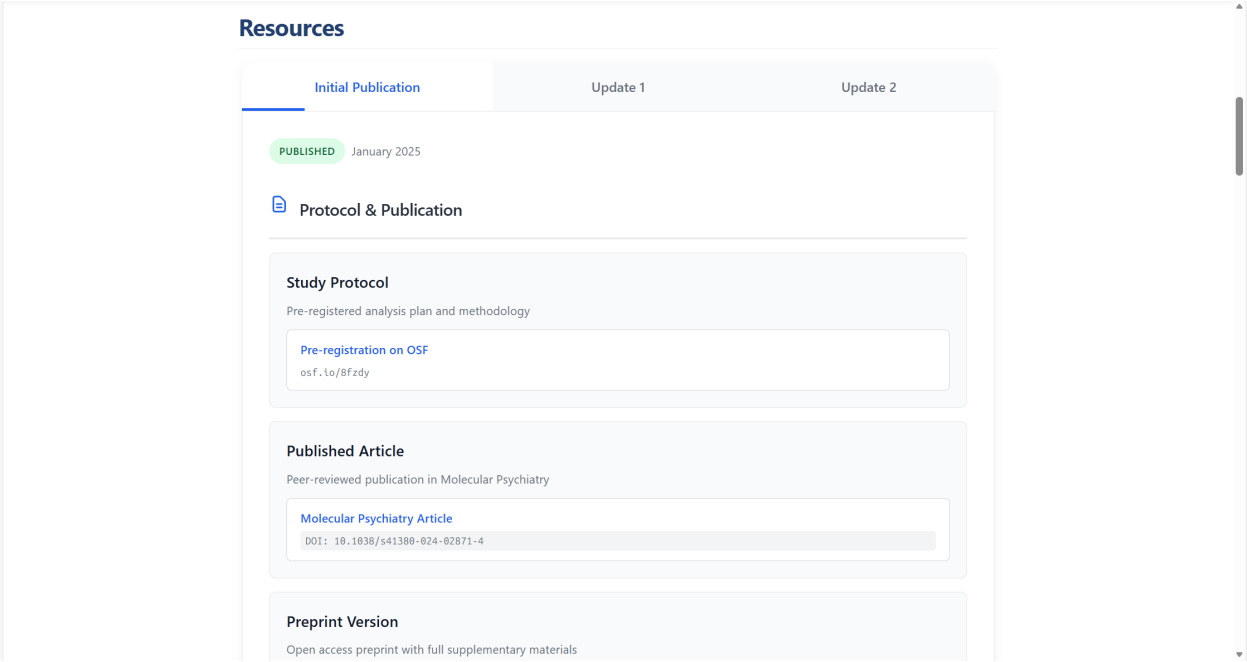


Figure 5.3: Reference image: Resources box for all materials connected to project.

The Resources section uses tabbed navigation to organize materials from the initial publication and subsequent updates. Subsequent sections will present how to build the Resources html code structure supported by JavaScript. HTML structure and indentation must be preserved exactly - only replace content within `[brackets]`.

5.3.4.1 Part 1: Tab Navigation (DO NOT EDIT)

```
<body>
    <div class="resources-container">

        <!-- TAB NAVIGATION BUTTONS -->
        <div class="tab-navigation">
            <button class="tab-button active" onclick="switchTab('initial')">Initial
            ↪  Publication</button>
            <button class="tab-button" onclick="switchTab('update1')">Update 1</button>
            <button class="tab-button" onclick="switchTab('update2')">Update 2</button>
        </div>
```

Critical: Keep all HTML tags, classes, and structure unchanged.

5.3.4.2 Part 2: Initial Publication Tab Structure

5.3.4.3 2A: Timeline Indicator

```
        <!-- INITIAL PUBLICATION TAB -->
        <div id="initial" class="tab-content active">

            <!-- Timeline indicator for initial publication -->
```

```
            <div class="timeline-indicator">
                <div class="status-badge status-published">Published</div>
              <span>[Month Year]</span>  <!-- EDIT: Replace with actual date, e.g., "January 2024" -->
            </div>
```

EDITABLE: Only the content inside `<span>[Month Year]</span>`

5.3.4.4 2B: Resource Categories

Each resource category follows this structure:

Standard Category Headers (Use as-is):

- Protocol & Publication
- Data & Analysis
- Related Research

Optional: Custom Category If you need a different category name, you can modify the category title while keeping the structure:

```
            <div class="resource-category">
                <div class="category-header">
                    <svg class="category-icon" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9
                        ↪  12h6m-6 4h6m2 5H7a2 2 0 01-2-2V5a2 2 0 012-2h5.586a1 1 0 01.707.293l5.414
                        ↪  5.414a1 1 0 01.293.707V19a2 2 0 01-2 2z"></path>
                    </svg>
                    <h3 class="category-title">Protocol & Publication</h3>  <!-- ✏ OPTIONAL EDIT:
                    ↪  Change category name if needed -->
                </div>
```

Keep unchanged: All `<div>` tags, classes, and the entire `<svg>` element

5.3.4.5 2C: Resource Items

Template for each resource item:

```
<!-- Study Protocol Item -->
                <div class="resource-item">
                    <div class="resource-header">
                        <div class="resource-info">
                            <div class="resource-title">Study Protocol</div>
                            <!-- ✏ EDIT: Replace "Study Protocol" with your resource type -->

                            <div class="resource-description">[Write short description of study
                            ↪  protocol, e.g., "Preregistered analysis plan outlining inclusion
                            ↪  criteria and analytical approach"]</div>
                            <!-- ✏ EDIT: Replace entire [bracketed text] with your description -->

                            <div class="resource-links">
                              <a href="[INSERT_FULL_URL_HERE]" class="resource-link" target="_blank">
                                <!-- ✏ EDIT: Replace [INSERT_FULL_URL_HERE] with actual URL -->
```

```
                                <div class="link-title">[Link name, e.g., "Preregistration on
                                ↪  OSF"]</div>
                                <!-- ✏ EDIT: Replace [bracketed text] with link name -->

                                <div class="link-url">[Clean URL, e.g., "osf.io/8fzdy"]</div>
                                <!-- ✏ EDIT: Replace [bracketed text] with display URL -->
                            </a>
                        </div>
                    </div>
                </div>
            </div>
```

Editing Guidelines:

- MUST EDIT: All content within `[brackets]`
- CAN EDIT: Text between `<div class="resource-title">` and `</div>`
- DO NOT EDIT: Any HTML tags, class names, or structure
- PRESERVE: All indentation exactly as shown

5.3.4.6 2D: Multiple Links Example

If a resource has multiple links, duplicate the `<a>` block:

```
<div class="resource-links">
    <a href="[INSERT_FULL_URL_HERE]" class="resource-link" target="_blank">
        <div class="link-title">[Paper title or description]</div>
        <div class="link-doi">[DOI, e.g., "10.1234/example"]</div>
    </a>
    <!-- ✏ ADD MORE: Copy and paste this <a> block for additional links -->
    <a href="[INSERT_FULL_URL_HERE]" class="resource-link" target="_blank">
        <div class="link-title">[Second paper title or description]</div>
        <div class="link-doi">[DOI, e.g., "10.5678/example"]</div>
    </a>
</div>
```

5.3.4.7 2E: Adding/Removing Resource Items

To add a new resource item: Copy an entire resource item block (from `<div class="resource-item">` to its closing `</div>`) and paste within the same category.

To remove a resource item: Delete the entire block from `<div class="resource-item">` to its matching closing `</div>`.

Critical: Maintain proper indentation when adding/removing items.

5.3.4.8 Part 3: Update Tabs

Both Update 1 and Update 2 follow the same structure until they are published. Once an update is published, replace the content with the full structure from Part 2 (Initial Publication Tab).

5.3.4.9 Update 1 Tab:

```
<!-- UPDATE 1 TAB -->
        <div id="update1" class="tab-content">

            <!-- Coming soon notice -->
            <div class="future-notice">
                <strong>Update 1 Coming Soon</strong> • Expected: [Month Year] (updates every [XX]
                ↪ months)
                <!-- ✏️ EDIT: Replace [Month Year] and [XX] with actual dates -->
            </div>

            <!-- Timeline indicator for update 1 -->
            <div class="timeline-indicator">
                <div class="status-badge status-coming-soon">Coming Soon</div>
                <span>Next update within [XX] months</span>
                <!-- ✏️ EDIT: Replace [XX] with your update cycle timeframe -->
            </div>
        </div>
```

5.3.4.10 Update 2 Tab:

```
        <!-- UPDATE 2 TAB -->
        <div id="update2" class="tab-content">

            <!-- Coming soon notice -->
            <div class="future-notice">
                <strong>Update 2</strong> • Expected: [Month Year] (updates every [XX] months)
                <!-- ✏️ EDIT: Replace [Month Year] and [XX] with actual dates -->
            </div>

            <!-- Timeline indicator for update 2 -->
            <div class="timeline-indicator">
                <div class="status-badge future-notice">Coming Soon</div>
                <span>Next update within [XX] months</span>
                <!-- ✏️ EDIT: Replace [XX] with your update cycle timeframe -->
            </div>
        </div>
```

5.3.4.11 Adding New Update Tabs

If you need to add Update 3, Update 4, etc., follow these steps:

5.3.4.12 Step 1: Add the tab button

In the Tab Navigation section (Part 1), add a new button:

```
        <div class="tab-navigation">
            <button class="tab-button active" onclick="switchTab('initial')">Initial
            ↪ Publication</button>
            <button class="tab-button" onclick="switchTab('update1')">Update 1</button>
            <button class="tab-button" onclick="switchTab('update2')">Update 2</button>
```

```
            <button class="tab-button" onclick="switchTab('update3')">Update 3</button>
            <!-- ⚠ ADD MORE: Copy the button line and change the number -->
        </div>
```

Critical:

- Change `switchTab('updateX')` where X is your update number
- Change the button text to match (e.g., "Update 3")

5.3.4.13 Step 2: Add the tab content section

Before the closing **`</div>`** of the resources-container and before the JavaScript, add:

```
<!-- UPDATE 3 TAB -->
        <div id="update3" class="tab-content">
            <!-- ⚠ IMPORTANT: The id must match the name used in switchTab() -->

            <!-- Coming soon notice -->
            <div class="coming-soon-notice">
                <strong>Update 3 Coming Soon</strong> • Expected: [Month Year] (updates every [XX]
                ↪ months)
                <!-- ⚠ EDIT: Replace [Month Year] and [XX] with actual dates -->
            </div>

            <!-- Timeline indicator -->
            <div class="timeline-indicator">
                <div class="status-badge status-coming-soon">Coming Soon</div>
                <span>Next update within [XX] months</span>
                <!-- ⚠ EDIT: Replace [XX] with your update cycle timeframe -->
            </div>
        </div>
```

Critical matching requirements:

- The `id="update3"` must match the button's `onclick="switchTab('update3')"`
- Keep the exact same structure for all future updates

5.3.4.14 When an Update is Published

When Update 1 (or any update) is published, replace the "Coming Soon" content with the full publication structure:

BEFORE (unpublished):

```
        <div id="update1" class="tab-content">
            <div class="coming-soon-notice">
                <strong>Update 1 Coming Soon</strong> • Expected: January 2026
            </div>
            <div class="timeline-indicator">
                <div class="status-badge status-coming-soon">Coming Soon</div>
                <span>Next update within 24 months</span>
            </div>
        </div>
```

AFTER (published):

```
        <div id="update1" class="tab-content">

            <!-- Timeline indicator for update 1 -->
            <div class="timeline-indicator">
                <div class="status-badge status-published">Published</div>
                <span>January 2026</span>  <!-- ✏️ EDIT: Actual publication date -->
            </div>

            <!-- PROTOCOL & PUBLICATION CATEGORY -->
            <div class="resource-category">
                <div class="category-header">
                    <svg class="category-icon" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9
                        ↪  12h6m-6 4h6m2 5H7a2 2 0 01-2-2V5a2 2 0 012-2h5.586a1 1 0 01.707.293l5.414
                        ↪  5.414a1 1 0 01.293.707V19a2 2 0 01-2 2z"></path>
                    </svg>
                    <h3 class="category-title">Protocol & Publication</h3>
                </div>

                <!-- Add all resource items here, following Part 2 structure -->

            </div>
```

Simply copy the entire structure from the Initial Publication tab (Part 2) and paste it, replacing all the "Coming Soon" content.

```
    <!-- JAVASCRIPT FOR TAB SWITCHING -->
    <script>
        function switchTab(tabId) {
            // Hide all tab contents
            const contents = document.querySelectorAll('.tab-content');
            contents.forEach(content => content.classList.remove('active'));

            // Remove active class from all buttons
            const buttons = document.querySelectorAll('.tab-button');
            buttons.forEach(button => button.classList.remove('active'));

            // Show selected tab content
            document.getElementById(tabId).classList.add('active');

            // Add active class to clicked button
            event.target.classList.add('active');
        }
    </script>
```

5.4 Optional sections

These sections are optional but highly recommended to include to make the project pages more informative and interesting.

5.4.1 Current results

The Current Results section presents your meta-analysis findings in a structured, visually organized format. It uses color-coded containers to communicate different types of findings and includes options for displaying important outcomes.

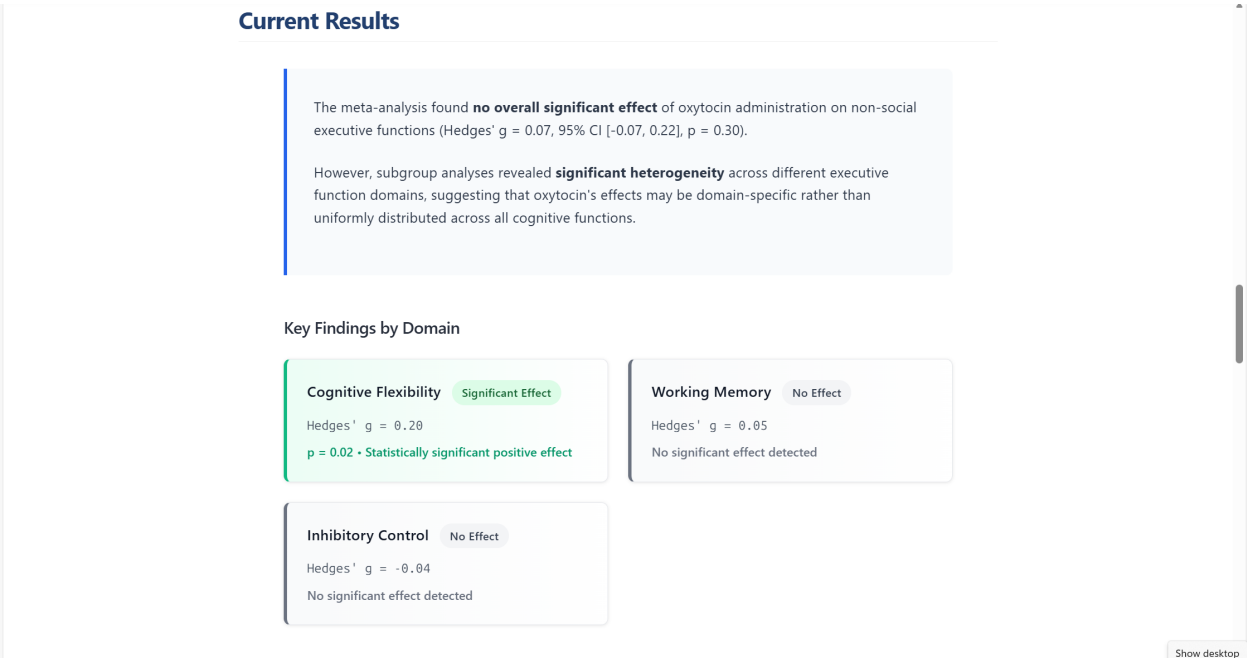


Figure 5.4: Reference image: Example from Kang et al. on how Current results section

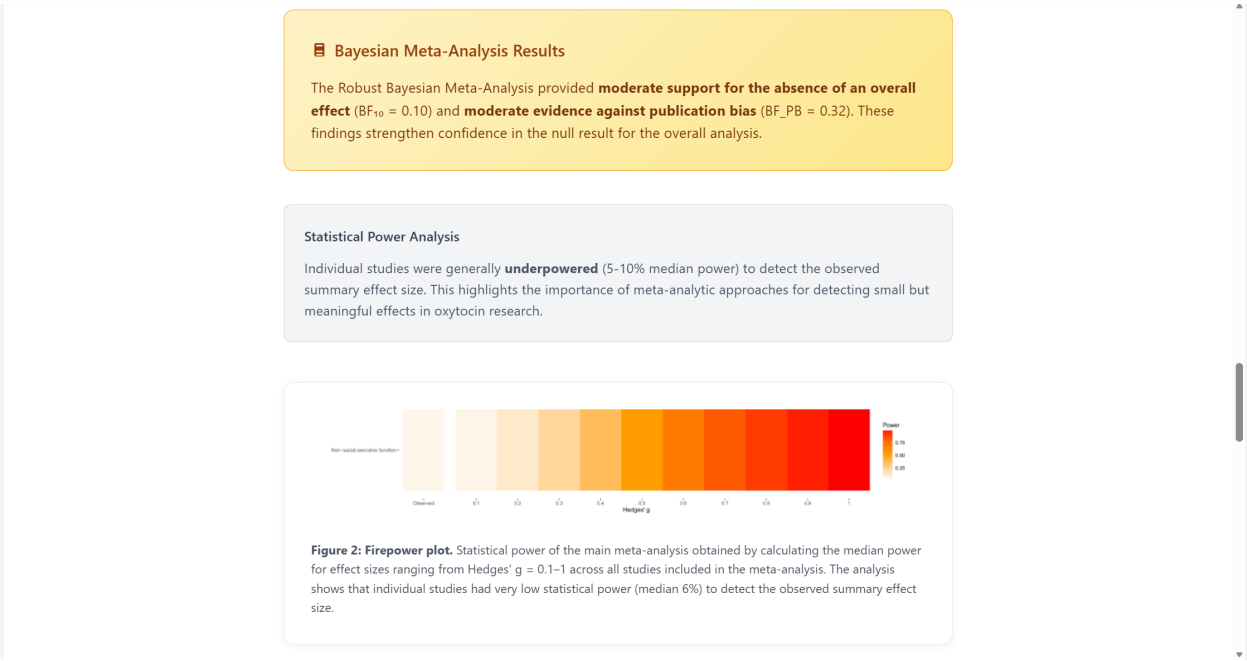


Figure 5.5: Reference image: Example from Kang et al. on how Current results section

Important: The entire results section must be wrapped in `<div class="results-container">` tags.

5.4.1.1 Part 1: Section Header (DO NOT EDIT)

```
## Current Results {#current-results}
<body>
    <div class="results-container">
```

Critical: Keep the section header and opening tags exactly as shown.

5.4.1.2 Part 2: Overall Findings Display

EDIT:

- `[OUTCOME]` - Replace with your outcome measure (e.g., "social cognition", "trust behavior")
- `[EFFECT SIZE MEASURE]` - replace with effect size measure (e.g. hedges' g, cohend's d)
- `[VALUE]` - Replace with effect size
- `[LOWER]` - Replace with lower confidence interval bound
- `[UPPER]` - Replace with upper confidence interval bound
- `[P-VALUE]` - Replace with p-value
- `[consistent across domains / varied by subdomain]` - Choose one or write your own description
- `[ADDITIONAL CONTEXT]` - Add relevant interpretation (e.g., "stronger effects observed in clinical populations")

Choose ONE option that best describes your overall meta-analytic findings:

5.4.1.3 Option A: No Overall Effect (Blue)

```
        <!-- Option A: No Overall Effect -->
        <div class="overall-findings" style="background: #f0f9ff; border-left: 4px solid #3b82f6;">
            <p>The meta-analysis found <strong>no overall significant effect</strong> of oxytocin
            ↪  administration on [OUTCOME] ([EFFECT SIZE MEASURE] = [VALUE], 95% CI [[LOWER],
            ↪  [UPPER]], p = [P-VALUE]).
        </div>
```

5.4.1.4 Option B: Significant Positive Effect (Green)

```
        <!-- Option B: Significant Positive Effect -->
        <div class="overall-findings" style="background: #f0fdf4; border-left: 4px solid #10b981;">
            <p>The meta-analysis found a <strong>significant positive effect</strong> of oxytocin
            ↪  administration on [OUTCOME] (Hedges' g = [VALUE], 95% CI [[LOWER], [UPPER]], p =
            ↪  [P-VALUE]).</p>
            <p>This effect was [consistent across domains / varied by subdomain], with [ADDITIONAL
            ↪  CONTEXT].</p>
```

```
</div>
```

5.4.1.5 Option C: Significant Negative Effect (Red)

```
<!-- Option C: Significant Negative Effect -->
<div class="overall-findings" style="background: #fef2f2; border-left: 4px solid #ef4444;">
    <p>The meta-analysis found a <strong>significant negative effect</strong> of oxytocin
    ↪ administration on [OUTCOME] (Hedges' g = [VALUE], 95% CI [[LOWER], [UPPER]], p =
    ↪ [P-VALUE]).</p>
    <p>This effect suggests that oxytocin [INTERPRETATION OF NEGATIVE EFFECT].</p>
</div>
```

5.4.1.6 Option D: Mixed/Heterogeneous Effects (Yellow/Orange)

```
<!-- Option D: Mixed/Heterogeneous Effects -->
<div class="overall-findings" style="background: #fffbeb; border-left: 4px solid #f59e0b;">
    <p>The meta-analysis revealed <strong>substantial heterogeneity</strong> in oxytocin's
    ↪ effects on [OUTCOME] (Hedges' g = [VALUE], 95% CI [[LOWER], [UPPER]], I² =
    ↪ [VALUE]%).</p>
    <p>Effects varied significantly by [MODERATOR], suggesting that oxytocin's impact is
    ↪ highly context-dependent.</p>
</div>
```

Important: Delete/Do not include the options you don't use. Keep only the one that matches your findings.

5.4.1.7 Part 3: Key Findings by Domain

This section displays findings broken down by specific domains or subgroups.

5.4.1.8 Section Wrapper (DO NOT EDIT)

```
<!-- KEY FINDINGS SECTION WRAPPER -->
<div class="key-findings-section">
    <h3 class="key-findings-title">Key Findings by Domain</h3>

    <div class="findings-grid">
```

5.4.1.9 Finding Card Options

Use as many cards as needed for your domains. Each card represents one domain/subgroup finding.

EDIT:

- `[DOMAIN NAME]` - Name of the domain/subgroup (e.g., "Social Cognition", "Trust Behavior")
- `[VALUE]` - Effect size for this domain
- `[EFFECT SIZE MEASURE]` - Effect size measure (e.g. Hedges g, Cohend's d)

- `[P-VALUE]` - P-value for this domain

Card Type A: Significant Positive Effect (Green)

```
<!-- Card Type A: Significant Positive Effect -->
<div class="finding-card significant" style="background: #f0fdf4;">
    <div class="finding-domain">
        [DOMAIN NAME]
        <span class="effect-size-badge positive-effect">Significant Effect</span>
    </div>
    <div class="finding-effect">[EFFECT SIZE MEASURE] = [VALUE]</div>
    <div class="finding-significance">p = [P-VALUE] • Statistically significant
    ↪ positive effect</div>
</div>
```

Card Type B: No Significant Effect (Gray)

```
<!-- Card Type B: No Significant Effect -->
<div class="finding-card non-significant" style="background: #f9fafb;">
    <div class="finding-domain">
        [DOMAIN NAME]
        <span class="effect-size-badge neutral-effect">No Effect</span>
    </div>
    <div class="finding-effect">Hedges' g = [VALUE]</div>
    <div class="finding-significance">No significant effect detected</div>
</div>
```

Card Type C: Significant Negative Effect (Red)

```
<!-- Card Type C: Significant Negative Effect -->
<div class="finding-card negative-significant" style="background: #fef2f2;">
    <div class="finding-domain">
        [DOMAIN NAME]
        <span class="effect-size-badge negative-effect">Significant Negative
        ↪ Effect</span>
    </div>
    <div class="finding-effect">Hedges' g = [NEGATIVE VALUE]</div>
    <div class="finding-significance">p = [P-VALUE] • Statistically significant
    ↪ negative effect</div>
</div>
```

Card Type D: Trending/Marginal Effect (Yellow)

```
<!-- Card Type D: Trending/Marginal Effect -->
<div class="finding-card non-significant" style="background: #fffbeb;">
    <div class="finding-domain">
        [DOMAIN NAME]
        <span class="effect-size-badge neutral-effect">Trending</span>
    </div>
    <div class="finding-effect">Hedges' g = [VALUE]</div>
  <div class="finding-significance">p = [P-VALUE] • Marginally significant trend</div>
</div>
```

5.4.1.10 Closing the Key Findings Section (DO NOT EDIT)

```
            </div>
        </div>
```

Instructions:

- Copy and paste as many finding cards as you need
- Mix and match card types (A, B, C, D) based on your results
- Keep cards in the order that makes most sense for your narrative
- Delete/do not include unused card type examples

5.4.1.11 Part 4: Bayesian Analysis Results

Choose ONE option that matches your Bayesian analysis results (or delete/skip this entire section if you didn't conduct Bayesian analysis):

EDIT:

- `[VALUE]` (first) - Replace with BF value (e.g., "0.23")
- `[moderate/strong]` - Choose appropriate strength descriptor
- `[VALUE]` (second) - Replace with BF_PB value
- `[ADDITIONAL INTERPRETATION]` - Add context about what this means for your research question

5.4.1.12 Option A: Evidence Against Effect (Blue)

```
        <!-- Option A: Evidence Against Effect (Null Result) -->
        <div class="bayesian-analysis" style="background: linear-gradient(135deg, #dbeafe 0%, #bfdbfe
        ↪  100%);">
            <div class="bayesian-title">
                <svg width="20" height="20" fill="currentColor" viewBox="0 0 20 20">
                   <path fill-rule="evenodd" d="M6 2a2 2 0 00-2 2v12a2 2 0 002 2h8a2 2 0 002-2V4a2 2 0
                    ↪  00-2-2H6zm1 2a1 1 0 000 2h6a1 1 0 100-2H7zm6 7H7a1 1 0 100 2h6a1 1 0 100-2zm0
                    ↪  4H7a1 1 0 100 2h6a1 1 0 100-2z" clip-rule="evenodd"/>
                </svg>
                Bayesian Meta-Analysis Results
            </div>
            <div class="bayesian-content">
                The Robust Bayesian Meta-Analysis provided <strong>moderate support for the absence of
↪  an effect</strong> ($BF_{10}$ = [VALUE]) and <strong>[moderate/strong] evidence against publication
↪  bias</strong> (BF_PB = [VALUE]). These findings strengthen confidence in the null result.
            </div>
        </div>
```

5.4.1.13 Option B: Evidence For Effect (Green)

```
        <!-- Option B: Evidence For Effect -->
        <div class="bayesian-analysis" style="background: linear-gradient(135deg, #d1fae5 0%, #a7f3d0
        ↪  100%);">
            <div class="bayesian-title">
                <svg width="20" height="20" fill="currentColor" viewBox="0 0 20 20">
                   <path fill-rule="evenodd" d="M6 2a2 2 0 00-2 2v12a2 2 0 002 2h8a2 2 0 002-2V4a2 2 0
                     ↪  00-2-2H6zm1 2a1 1 0 000 2h6a1 1 0 100-2H7zm6 7H7a1 1 0 100 2h6a1 1 0 100-2zm0
                     ↪  4H7a1 1 0 100 2h6a1 1 0 100-2z" clip-rule="evenodd"/>
                </svg>
                Bayesian Meta-Analysis Results
            </div>
            <div class="bayesian-content">
                The Robust Bayesian Meta-Analysis provided <strong>strong evidence for the
↪  effect</strong> (BF₁₀ = [VALUE]), supporting the frequentist findings. [ADDITIONAL
↪  INTERPRETATION].
            </div>
        </div>
```

5.4.1.14 Option C: Inconclusive Evidence (Yellow)

```
        <!-- Option C: Inconclusive Bayesian Evidence -->
        <div class="bayesian-analysis" style="background: linear-gradient(135deg, #fef3c7 0%, #fde68a
        ↪  100%);">
            <div class="bayesian-title">
                <svg width="20" height="20" fill="currentColor" viewBox="0 0 20 20">
                   <path fill-rule="evenodd" d="M6 2a2 2 0 00-2 2v12a2 2 0 002 2h8a2 2 0 002-2V4a2 2 0
                     ↪  00-2-2H6zm1 2a1 1 0 000 2h6a1 1 0 100-2H7zm6 7H7a1 1 0 100 2h6a1 1 0 100-2zm0
                     ↪  4H7a1 1 0 100 2h6a1 1 0 100-2z" clip-rule="evenodd"/>
                </svg>
                Bayesian Meta-Analysis Results
            </div>
            <div class="bayesian-content">
                The Robust Bayesian Meta-Analysis provided <strong>inconclusive evidence</strong>
↪  (BF₁₀ = [VALUE]), suggesting the data cannot clearly distinguish between the presence and absence
↪  of an effect.
            </div>
        </div>
```

5.4.1.15 Part 5: Statistical Notes

Add as many statistical note boxes as needed to highlight important methodological or analytical points. Common options provided:

5.4.1.16 Power Analysis Note (Gray)

```
        <!-- Power Analysis Note -->
        <div class="statistical-note" style="background: #f3f4f6; border-left: 3px solid #6b7280;">
            <div class="statistical-note-title">Statistical Power Analysis</div>
            <div class="statistical-note-content">
```

```
            Individual studies were generally <strong>underpowered</strong> ([PERCENT]% median
↪ power) to detect the observed summary effect size. This highlights the importance of meta-analytic
↪ approaches for detecting small but meaningful effects.
        </div>
    </div>
```

EDIT:

- `[PERCENT]` - Replace with median power percentage (e.g., "23")

5.4.1.17 Heterogeneity Note (Yellow)

```
    <!-- Heterogeneity Note -->
    <div class="statistical-note" style="background: #fef3c7; border-left: 3px solid #f59e0b;">
        <div class="statistical-note-title">Heterogeneity Analysis</div>
        <div class="statistical-note-content">
            Substantial heterogeneity was observed across studies (I² = [VALUE]%, τ² = [VALUE]),
↪ suggesting that [INTERPRETATION OF HETEROGENEITY SOURCE].
        </div>
    </div>
```

EDIT:

- `[VALUE]` (first) - Replace with I^2 percentage (e.g., "67")
- `[VALUE]` (second) - Replace with ^{2} value (e.g., "0.15")
- `[INTERPRETATION OF HETEROGENEITY SOURCE]` - Explain likely sources (e.g., "methodological differences across studies may account for variation in effect sizes")

5.4.1.18 Publication Bias Note (Blue)

```
    <!-- Publication Bias Note -->
    <div class="statistical-note" style="background: #dbeafe; border-left: 3px solid #3b82f6;">
        <div class="statistical-note-title">Publication Bias Assessment</div>
        <div class="statistical-note-content">
            [Evidence of publication bias was/was not detected] using [METHOD]. [INTERPRETATION
↪ AND IMPLICATIONS].
        </div>
    </div>
```

EDIT:

- `[Evidence of publication bias was/was not detected]` - Choose one
- `[METHOD]` - Replace with your assessment method (e.g., "Egger's test and funnel plot inspection")
- `[INTERPRETATION AND IMPLICATIONS]` - Explain what this means for interpreting your results

Instructions:

- Use all three note types, some, or none depending on your analysis
- You can add custom note boxes by copying the structure and changing the title and content
- Delete/Skip any notes that don't apply to your analysis

5.4.1.19 Part 6: Figure Display

Use this section to display key visualizations.

EDIT:

- `[IMAGE_URL]` - Path to your image file (e.g., "images/forest_plot.png" or URL)
- `[DESCRIPTIVE ALT TEXT]` - Accessibility text describing the image (e.g., "Forest plot showing effect sizes across 24 studies")
- `[NUMBER]` - Figure number (e.g., "1", "2")
- `[FIGURE TITLE]` - Title of the figure (e.g., "Forest Plot of Oxytocin Effects on Trust")
- `[DETAILED CAPTION]` - Full caption explaining the figure

```
<!-- FIGURE SECTION -->
<div class="figure-section" style="margin: 3rem 0; text-align: center;">
    <div class="figure-container" style="background: white; border: 1px solid #e5e7eb;
    ↪  border-radius: 12px; padding: 2rem; box-shadow: 0 2px 8px rgba(0, 0, 0, 0.05);">
        <img src="[IMAGE_URL]"
             alt="[DESCRIPTIVE ALT TEXT]"
             style="max-width: 100%; height: auto; border-radius: 8px; margin-bottom: 1rem;">
        <div class="figure-caption" style="font-size: 0.9rem; color: #4b5563; line-height:
        ↪  1.6; text-align: left; margin-top: 1rem;">
            <strong>Figure [NUMBER]: [FIGURE TITLE].</strong> [DETAILED CAPTION DESCRIBING THE
            ↪  FIGURE AND KEY FINDINGS]
        </div>
    </div>
</div>
```

Instructions:

- Copy and paste this entire block for each figure you want to display
- Update figure numbers sequentially
- Ensure image files are in your project directory

5.4.1.20 Part 7: Pre-Registration Notice (Optional)

If your analysis hasn't been conducted yet, use this instead of Parts 2-6:

```
<!-- No results yet -->
<div class="current-results-container">
    <div class="preregistered-notice">
        <p><strong>This is a pre-registered protocol.</strong> Initial analysis has not yet
        ↪  been conducted. Results will be posted when the analysis is completed and the
        ↪  pre-print is published.</p>
    </div>
</div>
```

When to use: Only for pre-registered protocols before initial analysis is complete.

When to remove: Delete this section entirely once you have results to report, and replace with Parts 2-6.

5.4.1.21 Part 8: Closing Tags (DO NOT EDIT)

```
    </div>
</body>
```

CRITICAL: Always include these closing tags at the end of your results section.

5.4.2 Inclusion and exclusion criteria

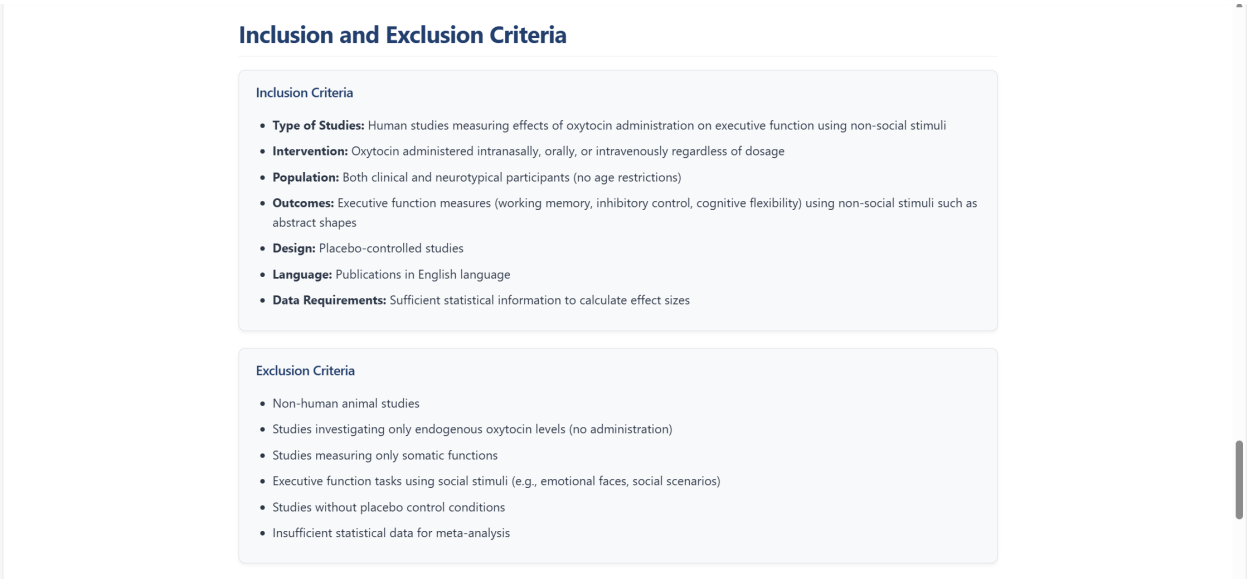


Figure 5.6: Reference image: inclusion and exclusion boxes

```
{r}
#| label: Inclusion and Exclusion Criteria
#| description: Here you can create two seperate boxes with the inclusion and exclusion criteria for
 ↪  the meta-analysis.
#| eval: false
#| echo: true


## Inclusion and Exclusion Criteria {#criteria}

::: {.criteria-box}
### Inclusion Criteria

- **Type of Studies:** [Study designs included, e.g., "Randomized controlled trials", "Original
 ↪  research written in English"]
- **Intervention:** [Specific intervention criteria, e.g., "Intranasal oxytocin administration"]
- **Population:** [Target population, e.g., "Healthy adults aged 18-65"]
- **Outcomes:** [Primary and secondary outcomes, e.g., "Social behavior measures", "Validated scales
 ↪  for trust"]
- **Data Requirements:** [Statistical requirements, e.g., "Must report means and standard deviations
 ↪  or effect sizes"]
:::
```

```
::: {.criteria-box}
### Exclusion Criteria

- [Specific exclusion criteria relevant to the research question]
- [Additional exclusion criteria]
- [Overlap/duplicate publication handling]
:::
```

5.4.3 Methodology summary

```
{r}
#| label: Methodology summary of main content area
#| description: A section handy to explain a summary of the methodology of the living meta-analysis.
#| eval: false
#| echo: true

## Methodology Summary {#methodology}

This living meta-analysis follows the PRISMA guidelines for systematic reviews.

The quality of included studies will be assessed using [specific quality assessment tool, e.g.,
 ↪  "Cochrane Risk of Bias tool for RCTs"]. Studies will be independently rated by two reviewers
 ↪  [describe rating categories and scoring system].

Statistical analyses will be conducted using [statistical software] with the [specific packages].
 ↪  [Type of effects model] will be used with [specific estimator]. Heterogeneity will be evaluated by
 ↪  calculating [heterogeneity measures]. Publication bias will be assessed using [bias assessment
 ↪  methods].
```

5.4.4 Search strategy

```
{r}
#| label: Search strategy of main content area
#| description: Write the search strings used on different search engines like PubMed or similar for
 ↪  reproducibility. It is very nice to have the search strings on the project page as this is
 ↪  apparently not always documented as makes reproducibility more difficult.
#| eval: false
#| echo: true


## Search Strategy {#search-strategy}

::: {.search-strategy-box}
<span class="search-db">write database here:</span>
write [search AND string] here

<span class="search-db">EMBASE:</span>
'Pitocin®' OR 'Syntocinon®' OR 'synthetic oxytocin' AND 'labour/labor' OR 'birth' OR 'perinatal' OR
 ↪  'prenatal' OR 'obstetric'

<span class="search-db">Web of Science:</span>
```

```
('Pitocin®' OR 'Syntocinon®' OR 'synthetic oxytocin' AND 'labour/labor' OR 'birth' OR 'perinatal' OR
 ↪  'prenatal' OR 'obstetric') AND ('delivery' AND 'augment*' OR 'induc*')

<span class="search-db">MEDLINE:</span>
'Pitocin®' OR 'Syntocinon®' OR 'synthetic oxytocin' AND 'labour/labor' OR 'birth' OR 'perinatal' OR
 ↪  'prenatal' OR 'obstetric' AND 'antepartum'

:::
```

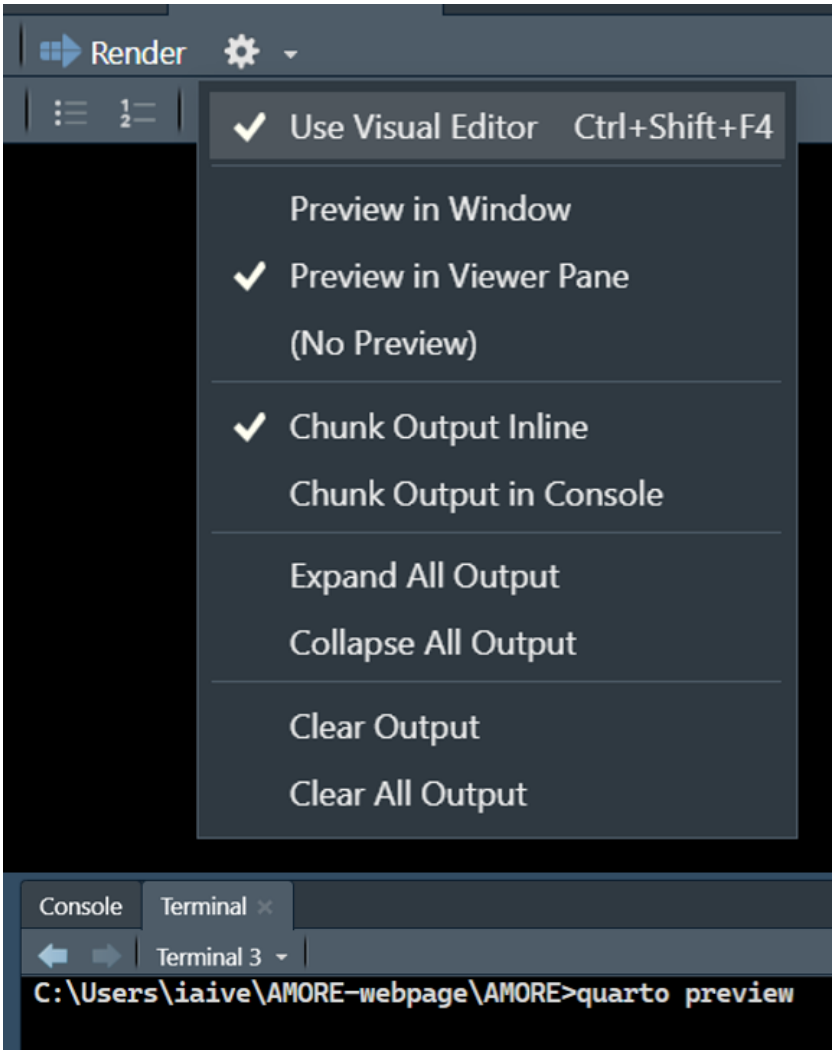
5.5 Previewing your work

It is very important to preview the work BEFORE pushing to Github repository.

You can preview by clicking the render button located at the the top bar of the RStudio interface, or by writing quarto preview in the terminal.

You can preview in Viewer pane OR in browser version. I recommend adjusting the viewer pane size when previewing to check how the content adjusts to different screen sizes.

IMPORTANT: Save project .qmd with prefix '_' while working on it. Only after it is ready to be published do you remove the '_projectfilename.qmd' to 'projectfilename.qmd' in the LMA folder and push to github repository. You can push to Github while working for version control with the prefix.


Render
Use Visual Editor Ctrl+Shift+F4
Preview in Window
Preview in Viewer Pane
(No Preview)
Chunk Output Inline
Chunk Output in Console
Expand All Output
Collapse All Output
Clear Output
Clear All Output
Console
Terminal
Terminal 3
C:\Users\iaive\AMORE-webpage\AMORE>quarto preview

Chapter 6

Adding new metadata alternatives

If you plan on expanding or adjusting the metadata options, there are three steps needed to complete that task.

The directory's backend is app.R, and app.R collects information from the YAML header in each .qmd file. To maintain control over and provide overview of all YAML metadata options, new categories or category options must be logged and written into the guide document's YAML header options with explanations. Lastly, the options must be made available to project proposals in Nettskjema.

Therefore, three steps must be completed to include a new metadata option:

- Add to app.R
- Add to guide document YAML header in Chapter "Adding new projects" (and preferably also _lma.template.qmd)
- Add to Nettskjema

6.1 Add a new filter category

If you are adding a new subcategory to an existing field (like adding "Eating Disorders" under Clinical Outcomes), you can skip this step. However, if you are creating an entirely new metadata category (for example, a new top-level filter like "Geographic Region" or "Funding Source"), you need to complete these steps:

- Step 1: Filter tab button - Add a clickable tab in the UI
- Step 2: Filter panel - Create the actual panel that opens when the tab is clicked
- Step 3: Filter section(s) - Add the checkbox groups or other inputs within the panel
- Step 4: Server logic - Add the new filter to:
  - The `observeEvent` that resets pagination when filters change
  - The scoring/filtering logic in `filtered_data()` reactive
  - The search text combination (if it should be searchable)

- Step 5: Metadata extraction - Update the functions that parse .qmd files to extract your new field from YAML

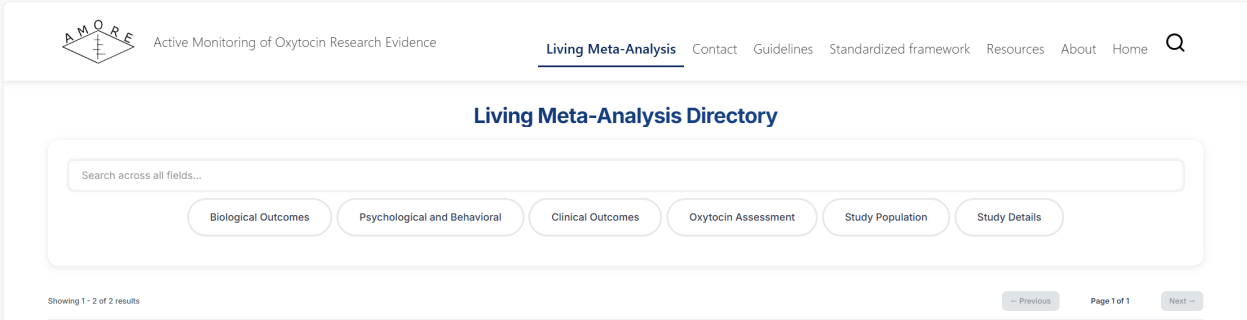


Figure 6.1: Filter tabs on the directory website page. If you want to include a new filter category here, this must be added to app.R and the YAML header.

```yaml
---
title: "Write title here"
analytical_framework: # Choose one or more
  - "Frequentist"
  - "Bayesian"
  - "Mixed methods"
oxytocin:
  intervention: # if applicable, choose one or more:
    - "Intranasal oxytocin administration"
    - "Oral oxytocin administration"
    - "Intravenous/injection oxytocin administration"
    - "Environmental/behavioral oxytocin manipulation"
    - "Perinatal oxytocin exposure"
  assessment_method: # if applicable, choose one or more:
    - "Behavioral assessment"
    - "Physiological response"
    - "Biological sample collection"
    - "Genetic studies"
    - "Neural/imaging measurement"
  new_metadata_category:
    - "add category options"
```

6.1.1 Step 1: Filter tab button

In the code below, the existing tabs in app.R are shown as examples. Add your new tab at the end, following the same format, as shown here with the brackets [your new metadata field]:

```
# Filter tabs with simpler approach
          div(class = "filter-tabs",
              tags$button("Biological Outcomes",
                          class = "filter-tab",
                          type = "button",
                          `data-target` = "biological"),
              tags$button("Psychological and Behavioral",
                          class = "filter-tab",
                          type = "button",
                          `data-target` = "psychological"),
```

```
            tags$button("Clinical Outcomes",
                        class = "filter-tab",
                        type = "button",
                        `data-target` = "clinical"),
            tags$button("Oxytocin Assessment",
                        class = "filter-tab",
                        type = "button",
                        `data-target` = "oxytocin"),
            tags$button("Study Population",
                        class = "filter-tab",
                        type = "button",
                        `data-target` = "population"),
            tags$button("Study Details",
                        class = "filter-tab",
                        type = "button",
                        `data-target` = "details")
            tags$button("[your new metadata field]",
                        class = "filter-tab",
                        type = "button",
                        `data-target` = "[datatarget]")
        ),
```

6.1.2 Step 2: Filter panels

After adding your filter tab button, you need to create the actual panel that will appear when users click on that tab. This panel contains the filter section(s) with checkboxes or other input options.

The panel's `id` must match the `data-target` value you used in Step 1. For example, if your `data-target` was "geographic_region", your panel `id` should be "geographic_region-panel".

Below, the Psychological & Behavioral panel is shown as an example, followed by a template for your new panel:

```
# Psychological & Behavioral Panel
            div(class = "filter-panel", id = "psychological-panel",
                div(class = "filter-section",
                    h4("Psychological and Behavioral Outcomes", class = "filter-title"),
                    checkboxGroupInput("psychological_behavioral_outcomes", NULL,
                                       choices = c("Mood and Emotion" = "mood_emotion",
                                                   "Cognition and Memory" = "cognition_memory",
                                                   "Stress and Coping" = "stress_coping",
                                                   "Eating and Appetite" = "eating_appetite",
                                                   "Risk and Decision-Making" = "risk_decision",
                                                   "Sleep Behavior and Quality" =
                                                    ↪  "sleep_behavior_quality",
                                                   "Bonding and Attachment" = "bonding_attachment",
                                                   "Trust and Cooperation" = "trust_cooperation",
                                                 "Communication and Empathy" = "communication_empathy",
                                                   "Aggression and Conflict" = "aggression_conflict"),
                                       selected = NULL
                    )
                )
            ),

            # New Panel
```

```
            div(class = "filter-panel", id = "[datatarget-panel]",
                div(class = "filter-section",
                    h4("[Title of category]", class = "filter-title"),
                    checkboxGroupInput("[relevant_to_title_outcome]", NULL,
                                       choices = c("[Subcategory option]" = "[subcategory_option]",
                                                   "[Subcategory option]" = "[subcategory_option]",
                                                   "[Subcategory option]" = "[subcategory_option]",
                                                   "[Subcategory option]" = "[subcategory_option]",
                                                   "[Subcategory option]" = "[subcategory_option]"),
                                       selected = NULL
                    )
                )
            ),
```

Important notes:

- The panel `id` format is: `[your_datatarget]-panel` (e.g., "geographic_region-panel")
- The `checkboxGroupInput` ID (first parameter) should be descriptive and match what you'll use in server logic (e.g., "geographic_region_filter")
- In the `choices`, the format is `"Display Text" = "internal_value"`
- The display text is what users see; the internal value is what gets stored in the data

6.1.3 Step 3: Filter section(s)

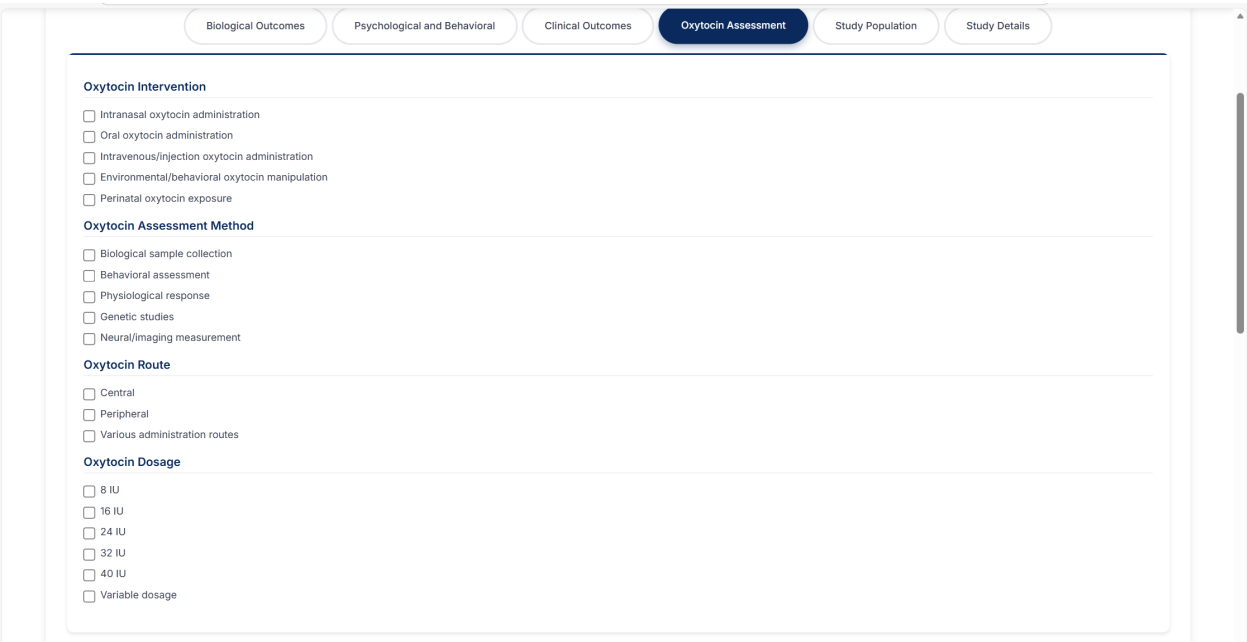


Figure 6.2: A filter panel can contain one or more sections. This illustrates a panel "Oxytocin Assessment" with multiple sections.

A filter panel can contain one or multiple filter sections. Each filter section has its own title and set of checkbox options. This is useful when a metadata category has multiple related dimensions.

For example, the Oxytocin Assessment panel contains four separate sections: Intervention, Assessment Method, Route, and Dosage. Each section filters on a different aspect of oxytocin assessment.

If your new metadata category has multiple related dimensions, you can add multiple `filter-section` divs within the same panel. Below, the Oxytocin Assessment panel with its four sections is shown as an example, followed by a template for your new panel with multiple sections:

```
# Oxytocin Assessment Panel
              div(class = "filter-panel", id = "oxytocin-panel",
                  div(class = "filter-section",
                      h4("Oxytocin Intervention", class = "filter-title"),
                      checkboxGroupInput("oxytocin_intervention", NULL,
                                         choices = c("Intranasal oxytocin administration",
                                                     "Oral oxytocin administration",
                                                     "Intravenous/injection oxytocin administration",
                                                     "Environmental/behavioral oxytocin manipulation",
                                                     "Perinatal oxytocin exposure"),
                                         selected = NULL
                      )
                  ),
                  div(class = "filter-section",
                      h4("Oxytocin Assessment Method", class = "filter-title"),
                      checkboxGroupInput("assessment_method", NULL,
                                         choices = c("Biological sample collection",
                                                     "Behavioral assessment",
                                                     "Physiological response",
                                                     "Genetic studies",
                                                     "Neural/imaging measurement"),
                                         selected = NULL
                      )
                  ),
                  div(class = "filter-section",
                      h4("Oxytocin Route", class = "filter-title"),
                      checkboxGroupInput("oxytocin_route", NULL,
                                         choices = c("Central",
                                                     "Peripheral",
                                                     "Various administration routes"),
                                         selected = NULL
                      )
                  ),
                  div(class = "filter-section",
                      h4("Oxytocin Dosage", class = "filter-title"),
                      checkboxGroupInput("oxytocin_dosage", NULL,
                                         choices = c("8 IU",
                                                     "16 IU",
                                                     "24 IU",
                                                     "32 IU",
                                                     "40 IU",
                                                     "Variable dosage"),
                                         selected = NULL
                      )
                  )
              ),



# New Panel
              div(class = "filter-panel", id = "[datatarget-panel]",
                  div(class = "filter-section",
```

```
            h4("[Title of category]", class = "filter-title"),
            checkboxGroupInput("[relevant_to_title_outcome]", NULL,
                               choices = c("[Subcategory option]",
                                           "[Subcategory option]",
                                           "[Subcategory option]",
                                           "[Subcategory option]",
                                           "[Subcategory option]"),
                               selected = NULL
            )
        ),
        div(class = "filter-section",
            h4("[Title of category]", class = "filter-title"),
            checkboxGroupInput("[relevant_to_title_outcome]", NULL,
                               choices = c("[Subcategory option]",
                                           "[Subcategory option]",
                                           "[Subcategory option]"),
                               selected = NULL
            )
        ),
        div(class = "filter-section",
            h4("[Title of category]", class = "filter-title"),
            checkboxGroupInput("[relevant_to_title_outcome]", NULL,
                               choices = c("[Subcategory option]",
                                           "[Subcategory option]",
                                           "[Subcategory option]"),
                               selected = NULL
            )
        ),
        div(class = "filter-section",
            h4("[Title of category]", class = "filter-title"),
            checkboxGroupInput("[relevant_to_title_outcome]", NULL,
                               choices = c("[Subcategory option]",
                                           "[Subcategory option]",
                                           "[Subcategory option]",
                                           "[Subcategory option]"),
                               selected = NULL
            )
        )
    ),
```

Important notes:

- Each `checkboxGroupInput` needs a unique ID (the first parameter)
- You can have as many or as few filter sections as needed - just one section is perfectly fine
- Each section's checkbox choices follow the same format: `"Display Text" = "internal_value"`

6.1.4 Step 4: Server logic

After creating the UI elements for your new filter, you need to add the corresponding server-side logic so that the filter actually works. This involves three main updates to the server code:

1. Reset pagination observer - Add your new filter to the list that triggers page reset when changed
2. Filtering and scoring logic - Add logic to filter entries and calculate relevance scores based on your new metadata

3. Search text combination - Include your new field in the searchable text so users can find it through the search bar

Each of these updates ensures your new filter integrates seamlessly with the existing search and filtering functionality.

6.1.4.1 Step 4a: Reset pagination observer

When users change any filter, the app should reset to page 1 of the results. To enable this for your new filter, add it to the `observeEvent` list that monitors filter changes.

Below, the existing filters are shown, followed by where to add your new filter showed with [brackets]:

```
  # Reset to page 1 when filters change
  observeEvent(list(
    input$search_text,
    input$biological_outcomes,
    input$psychological_behavioral_outcomes,
    input$clinical_outcomes,
    input$oxytocin_intervention,
    input$assessment_method,
    input$oxytocin_route,
    input$oxytocin_dosage,
    input$population_status,
    input$population_age,
    input$analysis_framework,
    [input$newfilter_id]
  ), {
    current_page(1)
  })
```

Important note:

- The input ID must match the `checkboxGroupInput` ID you created in Step 2 or 3

6.1.4.2 Step 4b, Part 1: Initialize the data frame structure

When the app starts, it creates an empty data frame to store metadata from all LMA files. You need to add a column for your new metadata field here.

Find the `parse_qmd_metadata` function and locate where `meta_df` is initialized. Add your new field to this list:

```
# Initialize empty data frame
meta_df <- data.frame(
  Title = character(),
  Status = character(),
  Framework = character(),
  OxytocinIntervention = character(),
  AssessmentMethod = character(),
  OxytocinRoute = character(),
  OxytocinDosage = character(),
  PopulationStatus = character(),
```

```
  PopulationAge = character(),
  PopulationClinicalType = character(),
  BiologicalOutcomes = character(),
  PsychologicalBehavioralOutcomes = character(),
  ClinicalOutcomes = character(),
  [NewField] = character(),
  LastUpdated = character(),
  Abstract = character(),
  Filename = character(),
  stringsAsFactors = FALSE
)
```

Important notes:

- Add your field in a logical location (e.g., group it with similar fields)
- Use `character()` as the data type

6.1.4.3 Step 4b, Part 2: Extract from YAML

After initializing the data frame, you need to tell the app how to extract your new field from the YAML header of each .qmd file. This happens in the same `parse_qmd_metadata` function.

Find the section where fields are extracted and add your extraction code:

```
# Extract multi-value fields with array support
oxytocin_intervention <- safe_extract(meta, c("oxytocin", "intervention"))
oxytocin_intervention_str <- extract_array_as_string(oxytocin_intervention)

assessment_method <- safe_extract(meta, c("oxytocin", "assessment_method"))
assessment_method_str <- extract_array_as_string(assessment_method)

# Extract your new field
new_field <- safe_extract(meta, c("[parent_category]", "[field_name]"))
new_field_str <- extract_array_as_string(new_field)
```

Important notes:

- Replace `[parent_category]` with the top-level YAML key (e.g., “oxytocin”, “population”, “outcomes”)
- Replace `[field_name]` with your specific field (e.g., “intervention”, “route”)
- The `_str` variable converts the data to a string format for storage

6.1.4.4 Step 4b, Part 3: Add to the entry list

After extracting the data from YAML, you need to add it to the entry list that gets created for each .qmd file. This happens further down in the same function.

Find the section where the entry list is created and add your new field:

```
entry <- list(
  Title = meta$title %||% "Untitled",
  Status = status,
  Framework = framework_str,
```

```
  OxytocinIntervention = oxytocin_intervention_str,
  AssessmentMethod = assessment_method_str,
  OxytocinRoute = oxytocin_route_str,
  OxytocinDosage = oxytocin_dosage_str,
  PopulationStatus = population_status_str,
  PopulationAge = population_age_str,
  PopulationClinicalType = safe_extract(meta, c("population", "clinical_type")),
  BiologicalOutcomes = biological_outcomes_str,
  PsychologicalBehavioralOutcomes = psychological_behavioral_outcomes_str,
  ClinicalOutcomes = clinical_outcomes_str,
  [NewField] = [new_field_str],
  LastUpdated = last_updated,
  Abstract = abstract,
  Filename = filename
)
```

Important notes:

- The name on the left (e.g., `NewField`) should match what you used in Part 1
- The value on the right should be the `_str` variable you created in Part 2

6.1.4.5 Step 4b, Part 4: Add to the data frame conversion

After creating entry lists for all files, the app converts them into a data frame. You need to add your new field to this conversion.

Find the section where the data frame is created from the entry list and add your field:

```
meta_df <- do.call(rbind, lapply(meta_list, function(x) {
  data.frame(
    Title = x$Title,
    Status = x$Status,
    Framework = x$Framework,
    OxytocinIntervention = x$OxytocinIntervention,
    AssessmentMethod = x$AssessmentMethod,
    OxytocinRoute = x$OxytocinRoute,
    OxytocinDosage = x$OxytocinDosage,
    PopulationStatus = x$PopulationStatus,
    PopulationAge = x$PopulationAge,
    PopulationClinicalType = x$PopulationClinicalType,
    BiologicalOutcomes = x$BiologicalOutcomes,
    PsychologicalBehavioralOutcomes = x$PsychologicalBehavioralOutcomes,
    ClinicalOutcomes = x$ClinicalOutcomes,
    [NewField] = x$[NewField],
    LastUpdated = x$LastUpdated,
    Abstract = x$Abstract,
    Filename = x$Filename,
    stringsAsFactors = FALSE
  )
}))
```

Important notes:

- The field name must match exactly what you used in Parts 1 and 3
- The format is `FieldName = x$FieldName`

6.1.4.6 Step 4b, Part 5: Add scoring logic

The app ranks search results by how many filter criteria each LMA matches. You need to add scoring logic for your new filter so it contributes to the relevance ranking.

Find the `filtered_data` reactive and locate the scoring section. Add your scoring code after the existing filters:

```
# Score for analysis framework
if (length(input$analysis_framework) > 0) {
  framework_scores <- sapply(1:nrow(meta_df), function(i) {
    as.numeric(check_multi_value_match(meta_df$Framework[i],
                                       input$analysis_framework))
  })
  score_vector <- score_vector + framework_scores
}

# Score for [your new filter]
if (length(input$[newfilter_id]) > 0) {
  new_scores <- sapply(1:nrow(meta_df), function(i) {
    as.numeric(check_multi_value_match(meta_df$[NewField][i],
                                       input$[newfilter_id]))
  })
  score_vector <- score_vector + new_scores
}
```

Important notes:

- Replace `[newfilter_id]` with the input ID from your checkboxGroupInput (Step 3)
- Replace `[NewField]` with the column name from your data frame (Part 1)
- This gives 1 point per match for ranking

6.1.4.7 Step 4b, Part 6: Add to search text combination

To make your new field searchable through the main search bar, add it to the combined search text.

Find the text search section in `filtered_data` and add your field:

```
# Apply enhanced text search
if (!is.null(input$search_text) && input$search_text != "") {
  search_text_combined <- paste(
    meta_df$Title,
    meta_df$Abstract,
    meta_df$BiologicalOutcomes,
    meta_df$PsychologicalBehavioralOutcomes,
    meta_df$ClinicalOutcomes,
    meta_df$OxytocinIntervention,
    meta_df$AssessmentMethod,
    meta_df$OxytocinRoute,
    meta_df$OxytocinDosage,
    meta_df$PopulationStatus,
    meta_df$PopulationAge,
    meta_df$PopulationClinicalType,
    meta_df$Framework,
    meta_df$[NewField],
```

```
    sep = " "
  )
```

6.1.4.8 Step 4b, Part 7: Add to total active filters count

The relevance badge shows how many criteria an LMA matches. Add your filter to the count calculation.

Find the `total_active_filters` calculation and add your filter:

```
# Calculate total active filters for badge display
total_active_filters <- length(input$biological_outcomes) +
  length(input$psychological_behavioral_outcomes) +
  length(input$clinical_outcomes) +
  length(input$oxytocin_intervention) +
  length(input$assessment_method) +
  length(input$oxytocin_route) +
  length(input$oxytocin_dosage) +
  length(input$population_status) +
  length(input$population_age) +
  length(input$analysis_framework) +
  length(input$[newfilter_id])
```

6.1.4.9 Step 4c: Add to display output

To make your new metadata field visible in the search results, add it to the display section where LMA entries are shown to users.

Find the `output$lma_list` section and locate where metadata is displayed. Add your field in a logical location:

```
div(class = "lma-meta",
    if (!is_empty_value(data$Framework[i])) span("Analytical Framework: ",
    ↪  format_display_value(data$Framework[i])),
    if (!is_empty_value(data$OxytocinIntervention[i])) span("Oxytocin Intervention: ",
    ↪  format_display_value(data$OxytocinIntervention[i])),
    if (!is_empty_value(data$AssessmentMethod[i])) span("Oxytocin Assessment: ",
    ↪  format_display_value(data$AssessmentMethod[i])),
    if (!is_empty_value(data$OxytocinRoute[i])) span("Oxytocin Route: ",
    ↪  format_display_value(data$OxytocinRoute[i])),
    if (!is_empty_value(data$OxytocinDosage[i])) span("Oxytocin Dosage: ",
    ↪  format_display_value(data$OxytocinDosage[i])),
    if (!is_empty_value(data$BiologicalOutcomes[i])) span("Biological Outcomes: ",
    ↪  format_display_value(data$BiologicalOutcomes[i])),
    if (!is_empty_value(data$PsychologicalBehavioralOutcomes[i])) span("Psychological and Behavioral
    ↪  Outcomes: ", format_display_value(data$PsychologicalBehavioralOutcomes[i])),
    if (!is_empty_value(data$ClinicalOutcomes[i])) span("Clinical Outcomes: ",
    ↪  format_display_value(data$ClinicalOutcomes[i])),
    if (!is_empty_value(data$[NewField][i])) span("[Display Label]: ",
    ↪  format_display_value(data$[NewField][i])),
    if (!is_empty_value(data$PopulationStatus[i])) span("Population Status: ",
    ↪  format_display_value(data$PopulationStatus[i])),
    if (!is_empty_value(data$PopulationAge[i])) span("Population Age: ",
    ↪  format_display_value(data$PopulationAge[i])),
    if (!is_empty_value(data$PopulationClinicalType[i])) span("Clinical Type: ",
    ↪  format_display_value(data$PopulationClinicalType[i])),
```

```
    if (!is_empty_value(data$LastUpdated[i])) span("Last Updated: ", data$LastUpdated[i])
),
```

Important notes:

- Replace `[NewField]` with your field name from the data frame
- Replace `[Display Label]` with the user-friendly text you want to show (e.g., "Geographic Region:")
- Place it in a logical location among related fields

6.1.5 Step 5: Add categorization logic (optional)

Most fields work with simple direct matching and don't need this step. However, if your new field has:

- Subcategories that should match various related terms (like "cardiovascular" matching "heart rate", "blood pressure", etc.)
- Hierarchical relationships (like "Variable dosage" being a parent that matches all specific dosages (8IU, 24IU))

Then you need to add categorization logic.

6.1.5.1 For subcategory matching:

Find the `categorize_outcome` function and add your new category type:

```
categorize_outcome <- function(outcome_text, category_type) {
  if (is.na(outcome_text)) return(FALSE)

  outcome_lower <- tolower(outcome_text)

  if (category_type == "cardiovascular") {
    return(any(sapply(c("hrv", "heart rate", "blood pressure", "cardiac", "cardiovascular"),
                      function(term) grepl(term, outcome_lower))))
  } else if (category_type == "neuroendocrine") {
    return(any(sapply(c("cortisol", "hormone", "endocrine", "stress marker", "testosterone",
     ↪  "estrogen"),
                      function(term) grepl(term, outcome_lower))))
  }

  # your new category
  } else if (category_type == "[your_category_type]") {
    return(any(sapply(c("[keyword1]", "[keyword2]", "[keyword3]", "[keyword4]"),
                    function(term) grepl(term, outcome_lower))))

  # ... other categories ...

  return(FALSE)
}
```

6.1.5.2 For hierarchical matching:

Find the `check_hierarchical_match` function and add parent categories if needed:

```
hierarchical_parents <- list(
  dosage = c("Variable dosage"),
  route = c("Various administration routes"),
  population_status = c("Mixed"),
  analytical_framework = ("Mixed methods"),
  population_age = c("Mixed age groups"),
  [your_field] = c("[Parent Category Name]")
)
```

Important notes:

- Skip this step entirely if your field uses simple exact matching
- Only add categorization if users need to search by broader terms
- Only add hierarchical matching if you have parent categories that should match all children

6.2 Adding a new subcategory to an existing field

When to use: Adding a new option to an existing filter category (e.g., adding "Eating Disorders" under Clinical Outcomes, or adding "48 IU" to Oxytocin Dosage)

6.2.1 Overview

Adding a subcategory only requires updating three locations in app.R:

1. The checkbox choices in the filter panel UI
2. The categorization logic (if using semantic matching)
3. No changes to data extraction or server logic are needed

6.2.2 Step 1: Update checkbox choices

Find your filter panel in the UI section and add your new option to the `choices` list.

Example - Adding "Eating Disorders" to Clinical Outcomes:

```
# Clinical Outcomes Panel
div(class = "filter-panel", id = "clinical-panel",
    div(class = "filter-section",
        h4("Clinical Outcomes", class = "filter-title"),
        checkboxGroupInput("clinical_outcomes", NULL,
                           choices = c("Neurodevelopmental" = "neurodevelopmental",
                                       "Mood Disorders" = "mood_disorders",
                                       "Psychotic Disorders" = "psychotic_disorders",
                                       "Addiction and Substance Use" = "addiction_substance",
                                       "Eating Disorders" = "eating_disorders"),  # â† Already added
                           selected = NULL
        )
    )
),
```

Template - Add your new subcategory:

```
checkboxGroupInput("[field_id]", NULL,
                   choices = c("[Existing Option 1]" = "[existing_option_1]",
                               "[Existing Option 2]" = "[existing_option_2]",
                               "[Your New Option]" = "[your_new_option]"),  # â† Add here
                   selected = NULL
)
```

Important notes: - The format is `"Display Text" = "internal_value"` - Display text is what users see in the checkbox - Internal value should use lowercase with underscores (e.g., "eating_disorders") - Don't forget the comma after the previous line!

6.2.3 Step 2: Add categorization logic (if applicable)

Most fields don't need this step. Only add categorization logic if:

- Your field uses semantic matching (like Biological, Psychological/Behavioral, and Clinical Outcomes)
- You want the filter to match various related terms in the metadata

For fields with exact matching (like Oxytocin Dosage, Population Age, Analytical Framework), skip this step entirely.

6.2.3.1 When categorization is needed

Find the `categorize_outcome` function and add your new category with relevant search terms.

Example - The existing "eating_disorders" categorization:

```
categorize_outcome <- function(outcome_text, category_type) {
  if (is.na(outcome_text)) return(FALSE)

  outcome_lower <- tolower(outcome_text)

  # ... other categories ...

  } else if (category_type == "eating_disorders") {
    return(any(sapply(c("anorexia", "bulimia", "eating disorder", "binge"),
                      function(term) grepl(term, outcome_lower))))
  }

  return(FALSE)
}
```

Template - Add your new categorization:

```
  } else if (category_type == "[your_new_option]") {
    return(any(sapply(c("[keyword1]", "[keyword2]", "[keyword3]", "[keyword4]"),
```

```
                    function(term) grepl(term, outcome_lower))))
```

Important notes:

- The `category_type` must match the internal value from Step 1
- Add keywords that should trigger a match (lowercase)
- Keywords can be single words or phrases
- The function searches for partial matches, so "eating disorder" will match "eating disorders"

6.3 Update the guide document YAML header

After adding your new metadata field to app.R, you must document the available options in the guide so that project creators know what values they can use.

6.3.1 Location

The YAML metadata documentation is in the "Adding new projects" chapter of the guide document (`_AMORE_Guide_pdf.qmd`).

6.3.2 What to update

Find the section titled "YAML header" which contains a complete example of all available metadata options. This is the reference that users consult when creating their project files.

6.3.2.1 Example: Adding "Eating Disorders" to Clinical Outcomes

Locate the relevant section in the YAML header documentation and add your new option:

```yaml
outcomes:
  clinical: # if applicable, choose one or more:
    - "Neurodevelopmental"
    - "Mood Disorders"
    - "Psychotic Disorders"
    - "Addiction and Substance Use"
    - "Eating Disorders"  # ← Added
```

Important notes: - Keep the formatting consistent with existing entries - Include the option in quotes if other options use quotes - Maintain alphabetical order if the list is alphabetized - Add clarifying comments if the option needs explanation

---

6.3.3 For new top-level categories

If you added a completely new filter category (not just a subcategory), you need to add the entire section to the YAML documentation.

Example: Adding a new "Geographic Region" category:

```yaml
geographic_region: # if applicable, choose one or more:
  - "North America"
  - "Europe"
  - "Asia"
  - "South America"
  - "Africa"
  - "Australia/Oceania"
  - "Multiple regions"
```

Place it in a logical location within the YAML structure, typically grouped with related metadata fields.

6.4 Update _lma.template.qmd (Recommended)

While updating the guide document is mandatory, updating the template file is highly recommended. The template file (`_lma.template.qmd` in the LMAs folder) is what project creators copy when creating new project pages.

6.4.1 Why update the template?

- Provides immediate reference for users
- Reduces errors from typing options manually
- Shows the exact format needed
- Makes it easier for users to select valid options

6.4.2 What to update

The template contains the same YAML header structure as documented in the guide. Update it in the same way:

Location: `/LMAs/_lma.template.qmd`

Section: YAML header at the very top of the file

6.4.2.1 Example: Adding "Eating Disorders"

```yaml
---
title: "Write title here"
analytical_framework:
  - "Frequentist"
  - "Bayesian"
  - "Mixed methods"
# ... other sections ...
outcomes:
```

```yaml
  clinical: # if applicable, choose one or more:
    - "Neurodevelopmental"
    - "Mood Disorders"
    - "Psychotic Disorders"
    - "Addiction and Substance Use"
    - "Eating Disorders"  # ← Add here
# ... rest of template ...
---
```

Important notes: - The template YAML must match what's documented in the guide - The template YAML must match what app.R expects - Any mismatch will cause confusion or errors

---

6.5 Update Nettskjema contact form

The final step is updating the Nettskjema form so that new project proposals can include your new metadata option.

6.5.1 Which form to update

Form name: "Propose your project"

Who has edit access: - ingebjai@uio.no (Ingebjørg) - danielqu@uio.no (Daniel Quintana)

6.5.2 Why update Nettskjema?

When researchers propose new projects, they fill out a Nettskjema form that collects metadata about their project. This metadata is then used to create their project page. If your new option isn't available in the form, users can't select it during project proposal.

6.5.3 What to update

The form has a section for metadata selection where users choose from predefined options that match the YAML structure.

6.5.4 Steps to update the form:

1. Log into Nettskjema at nettskjema.no with appropriate credentials
2. Find the "Propose your project" form
3. Locate the metadata question section - this is where users select their project's metadata (outcomes, population, oxytocin details, etc.)
4. Add your new option in the same style as the previous option
5. Save and test the form to ensure the new option appears

6.5.5 Important considerations

Consistency is critical: The options in Nettskjema must exactly match: - The options in app.R filter checkboxes - The options documented in the guide - The options in _lma.template.qmd

Formatting: - Use the display text (e.g., "Eating Disorders") not the internal value (e.g., "eating_disorders") - Match capitalization and punctuation exactly - Keep the same order as in the template when possible

Optional vs Required: - Metadata questions in Nettskjema are voluntary - Users can skip sections that don't apply to their project - This is fine - blank fields simply won't display in the directory

---

6.6 Summary: Complete update workflow

To add a new metadata option to the AMORE system:

6.6.1 1. Update app.R

- Add to UI filter choices
- Add to server logic (data extraction, scoring, display)
- Add categorization if using semantic matching

6.6.2 2. Update guide document

- Document the new option in the YAML header example
- Include in the "Adding new projects" chapter
- Add explanatory comments if needed

6.6.3 3. Update _lma.template.qmd (recommended)

- Add the new option to the template YAML header
- Ensure formatting matches the guide

6.6.4 4. Update Nettskjema

- Add the new option to the "Propose your project" form
- Place in the appropriate metadata section
- Test the updated form

6.6.5 Verification checklist

After completing all updates:

☐ New option appears in directory filters
☐ Filtering works correctly in the Shiny app
☐ Option is documented in the guide
☐ Option is available in the template file
☐ Option can be selected in Nettskjema

☐ Test project file with new option displays correctly
☐ All four sources match exactly (app.R, guide, template, form)

Chapter 7

Troubleshooting

This section proposes different solutions to common errors and problems you may encounter when working on AMORE-website.

7.1 Recovery

Important: All files starting with _ are in `.gitignore` and won't be committed to version control.

Git ensures version control so earlier version pushed to Git can be retrieved.

7.1.1 Recovery Files

The project includes backup/testing files for safe experimentation:

7.1.1.1 `_recoveryfile.qmd`

- Purpose: Test major page changes before applying to live files
- Usage: Remove _ to render, test, then re-add _ to hide
- Location: /docs folder

7.1.1.2 `_recoveryapp.R`

- Purpose: Test Shiny app updates before deploying
- Usage:
    1. Copy `app.R` to `_recoveryapp.R`
    2. Make changes to `_recoveryapp.R`
    3. Test by running app from `_recoveryapp.R`
    4. If successful, copy code back to `app.R`
- Location: `/docs`

7.2 Common Problems and Solutions

7.2.0.1 Authentication Error: "Invalid username or password" or "Password authentication is not supported"

Example of the issue:

```
$ git push origin main
Username for 'https://github.com': user-name@example.com
Password for 'https://user-name@example.com@github.com':
remote: Invalid username or password
fatal: Authentication failed for 'https://github.com/user/repo.git/'
```

This error occurs when authentication is not set up correctly or GitHub cannot verify your credentials.

Common Causes and Solutions:

7.2.0.2 1. Using Password Instead of Personal Access Token (PAT)

GitHub no longer accepts account passwords for Git operations via HTTPS. You must use a Personal Access Token instead.

Solution: If you haven't set up authentication yet, go back to the Authentication Setup section and follow the instructions for either PAT or SSH.

If you already have a PAT but are still seeing this error, make sure you're entering the token (not your GitHub password) when prompted for a password.

7.2.0.3 2. Hyphens in Email or Username

Hyphens in email addresses or usernames are valid but can sometimes cause issues if improperly handled by Git.

Solution: Ensure your email is correctly configured in Git:

```
git config --global user.email "user-name@example.com"
git config --global user.name "Your Name"
```

7.2.0.4 3. Incorrect Cached Credentials

If you previously entered wrong credentials, Git may have cached them.

Solution: Clear the cached credentials:

On Windows:

```
git credential-manager-core erase
# Or
git config --global --unset credential.helper
```

On Mac:

```
git credential-osxkeychain erase
host=github.com
protocol=https
[Press Return twice]
```

On Linux:

```
git config --global --unset credential.helper
```

After clearing, the next time you push, Git will prompt for credentials again. Enter your GitHub username and your Personal Access Token (not password).

7.2.0.5 4. Switch from HTTPS to SSH Authentication

If HTTPS continues to cause problems, switching to SSH is often more reliable.

Solution:

First, set up SSH keys (see Authentication Setup - Option 2: SSH Keys in section Getting started), then update your repository's remote URL:

```
cd AMORE-webpage
git remote set-url origin git@github.com:iaiversen/AMORE-webpage.git
```

Or for your fork (Method 2 users):

```
git remote set-url origin git@github.com:[YOUR-USERNAME]/AMORE-webpage.git
```

Verify the change:

```
git remote -v
```

You should see `git@github.com:...` instead of `https://github.com/...`

7.2.0.6 5. Verify Your Authentication Setup

Test your authentication:

For HTTPS (PAT):

```
git ls-remote https://github.com/iaiversen/AMORE-webpage.git
```

If prompted, enter your username and PAT. If it lists branches, authentication works.

For SSH:

```
ssh -T git@github.com
```

You should see: "Hi [username]! You've successfully authenticated..."

7.2.1 YAML Syntax Errors

Symptoms: - Render fails with “YAML error” message - Page shows blank content - Console shows parsing errors

Common Causes:

```yaml
{yaml}
#| label: YAML troybleshooting
#| description: If the YAML header shows error try to see if any of these troubleshooting steps helps
 ↪  solve your problem.
#| eval: false
#| echo: true

# β—Œ Missing quotes
status: Preregistered

# βœ" Correct
status: "Preregistered"

# β—Œ Incorrect array syntax
intervention: "Value 1", "Value 2"

# βœ" Correct
intervention:
  - "Value 1"
  - "Value 2"

# β—Œ Inconsistent indentation
outcomes:
  biological:
  - "Item"  # Wrong indent

# βœ" Correct
outcomes:
  biological:
    - "Item"  # Correct indent
```

Solution: 1. Check R console for specific error line 2. Validate YAML at: http://www.yamllint.com/ 3. Ensure consistent 2-space indentation 4. Use quotes for all string values

7.2.2 Styling Problems

Symptoms: - Colors don’t match - Layout broken - Responsive design fails - Changes don’t appear

Solution Steps:

7.2.2.1 Inspect element

What is Inspect Element?

Inspect Element is a powerful browser tool that lets you peek “under the hood” of any webpage to see exactly how it’s built and styled. Think of it as X-ray vision for websites.

What Inspect Element Shows You:

1. HTML Structure: The actual code that creates the page

2. CSS Styles: All the styling rules affecting each element
3. Which styles are actually applied: When multiple rules conflict, see which one "wins"
4. Box model: Margins, padding, and spacing around elements
5. Console errors: JavaScript errors or warnings

How to Open Inspect Element

In Any Browser:

- Right-click on any element → Select "Inspect" or "Inspect Element"

In RStudio Preview:

1. Render your page (it opens in Viewer or browser)
2. Click "Show in new window" icon (opens in system browser)
3. Now use inspect element as normal
4. Note: RStudio's built-in viewer has limited inspect functionality

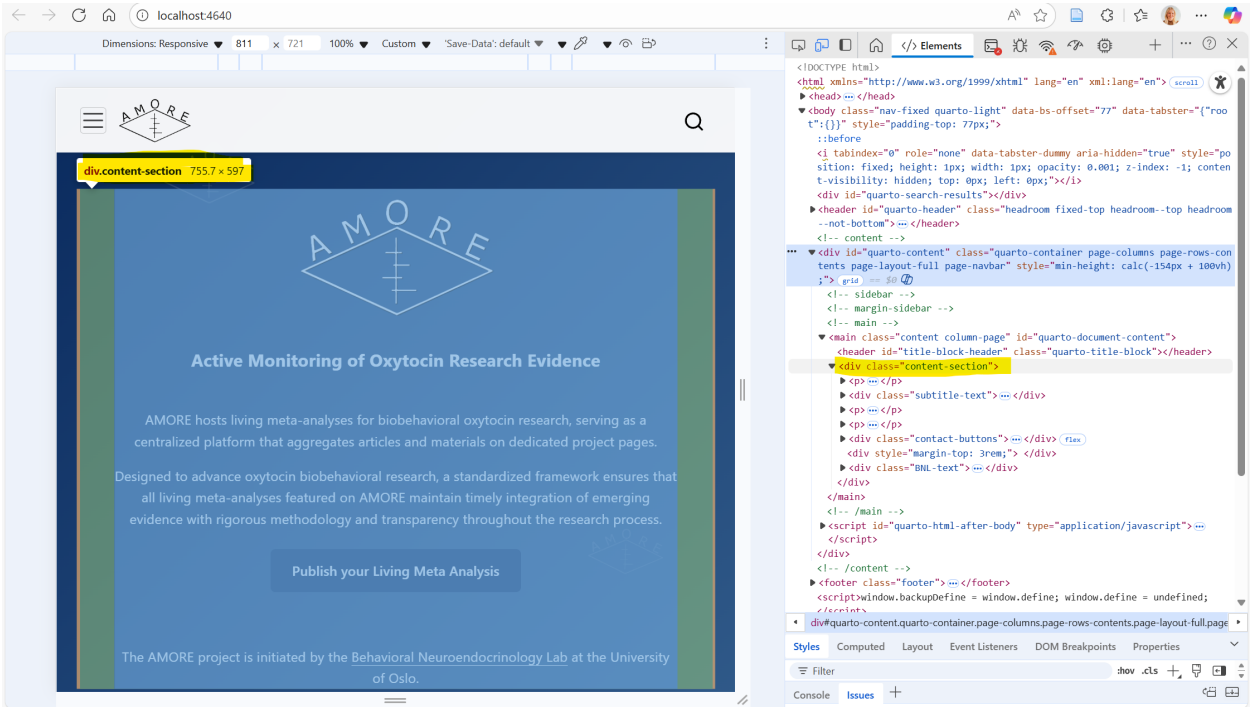
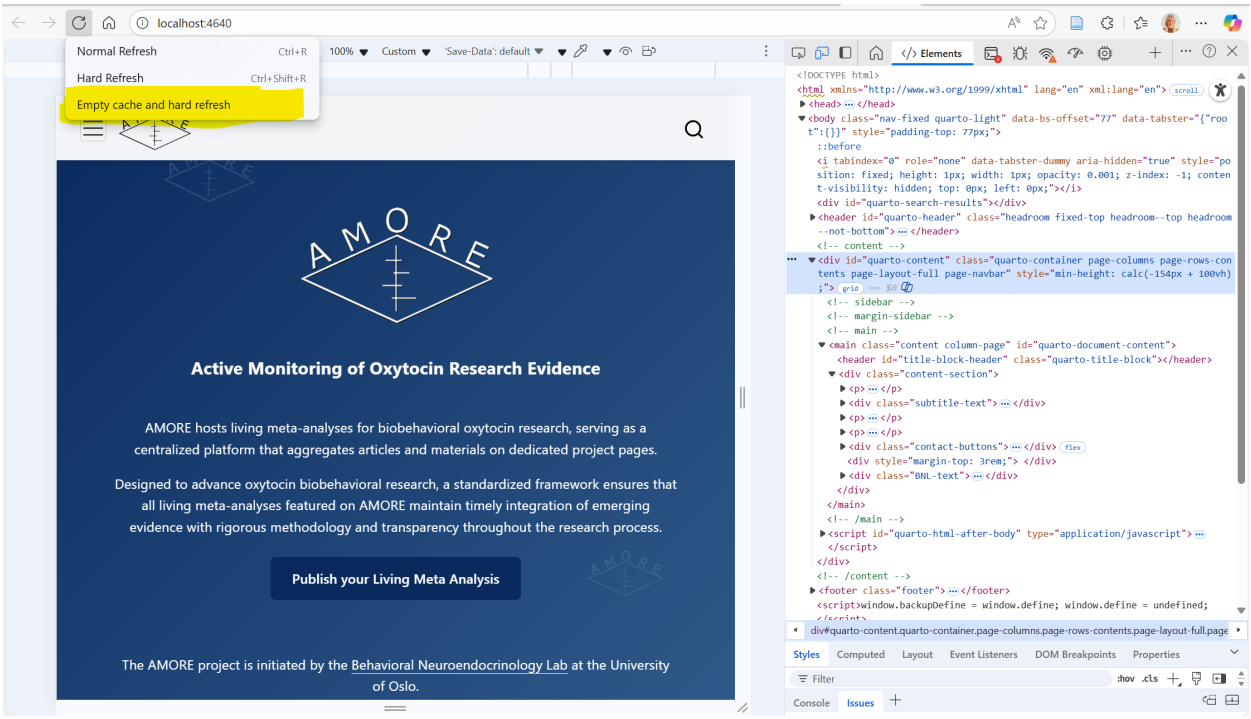


Figure 7.1: Inspect element in browser

7.2.2.2 Cache Issues:

1. Save all files in RStudio
2. Restart R: `Session >> Restart R`

3. Render to browser preview - Right-click page Inspect Element
4. Clear browser cache: - - Right-click refresh button - “Empty Cache and Hard Refresh”
5. Re-render page



7.2.2.3 CSS Specificity:

```{scss}
#| label: scss troubleshooting
#| description: see if changing specificity will help
#| eval: false
#| echo: true

/* Too general - might not apply */
.box {
  color: red; }

/* More specific - will apply */
.content-section .box {
  color: red !important;  /* Use !important sparingly */ }
```

7.2.2.4 Syntax Errors:

```{scss}
#| label: scss troubleshooting
#| description: look for syntax error
#| eval: false
#| echo: true

/* Missing semicolon */
.class {
  color: red
  margin: 10px; }

/* Correct */
.class {
  color: red;
  margin: 10px; }
```

7.3 Recovering Previous Versions

7.3.0.1 Undo uncommitted changes:

```{bash}
#| label: Undo committed changes with git command
#| description: write code into command line interface (Bash, Powershell, RStudio Terminal)
#| eval: false
#| echo: true

# Discard changes to specific file
git checkout -- filename.qmd

# Discard all uncommitted changes
git reset --hard HEAD
```

7.3.0.2 Restore file to previous version:

```{bash}
#| label: Restore file to previous versions with git command
#| description: write code into command line interface (Bash, Powershell, RStudio Terminal)
#| eval: false
#| echo: true

# Find the commit hash
git log --oneline

# Restore file from specific commit git checkout [commit-hash] -- filename.qmd

# Example:
git checkout a1b2c3d -- LMAs/project.qmd
```

7.4 Recovering from Mistakes

7.4.0.1 Accidentally deleted file:

```
{bash}
#| label: Version control with Git - recover accidentally deleted file
#| description: write code into command line interface (Bash, Powershell, RStudio Terminal).
#| eval: false
#| echo: true

# Restore from last commit
git checkout HEAD -- filename.qmd
```

7.4.0.2 Committed wrong files:

```
{bash}
#| label: Version control with Git - commited wrong file
#| description: write code into command line interface (Bash, Powershell, RStudio Terminal)
#| eval: false
#| echo: true

# Undo last commit, keep changes
git reset --soft HEAD~1

# Re-stage correct files
git add correct-files.qmd git commit -m "Corrected commit"
```

7.4.0.3 Pushed broken code:

```
{bash}
#| label: Version control with Git - go back to last committed version
#| description: write code into command line interface (Bash, Powershell, RStudio Terminal)
#| eval: false
#| echo: true

# Revert to previous working commit
git revert [commit-hash] git push origin main
```

Chapter 8

Website maintenance

This section provides the correct information needed to maintain a smooth running website. The section provides instructions on updating packages and installations necessary for the project, and the necessary maintenance of third party services used to host and deploy AMORE.

8.1 Third-Party Services Overview

AMORE's infrastructure relies on several third-party services. Each service has specific maintenance requirements and renewal schedules.

8.1.1 Netlify

- Purpose: Domain hosting and website deployment
- Cost: Domain name (annual payment required), deployment (free tier)
- Website: https://www.netlify.com

8.1.1.1 Maintenance Tasks:

- Renew domain name annually (Needs updated billing information, billing can be autorenewed after billing information is updated)
- Monitor build logs for deployment errors
- Check deploy settings if website fails to update

8.1.2 ShinyApps.io

- Purpose: Hosting the Living Meta-Analysis Directory Shiny application
- Account: meta-oxytocin
- Cost: Free tier
- Dashboard: https://www.shinyapps.io/admin/#/dashboard
- URL: https://meta-oxytocin.shinyapps.io/shiny-meta/

8.1.2.1 Maintenance Tasks:

- Monitor app usage (free tier has limits, might be necessary with paid version to improve server capacity). Free tier usage limits:
  - 25 active hours per month
  - 5 applications maximum
  - Automatic sleep after 15 minutes of inactivity
- Check error logs if app becomes unresponsive
- Redeploy if changes are made to `app.R`

8.1.3 GitHub

- Purpose: Version control and source code repository
- Repository: https://github.com/iaiversen/AMORE-webpage
- Cost: Free

8.1.3.1 Maintenance Tasks:

- Regularly commit and push changes
- Review pull requests if collaborators
- Keep repository organized (clear branch structure)
- Monitor repository size (GitHub has storage limits)

8.1.4 Nettskjema

- Purpose: Contact forms and data collection
- Provider: University of Oslo
- Cost: Free (institutional service)
- Renewal: Requires renewal after a set period of active years (check in Nettskjema)

8.1.4.1 Maintenance Tasks:

- Ensure form links are working
- Update form fields if project requirements change
- Export and back up form responses periodically
- Monitor embedded contact forms, perhaps better to change to old solution if embedded solution is loading to slow, or shows other errors. Follow instructions below to change back to old version.

How to change contact form solution:

Step 1. Move to folder /pages

Step 2. Rename contact.qmd to _embedded_contact.qmd

Step 3. Rename _redirected_contact.qmd to contact.qmd

Step 4. Push changes to Github repository AMORE-webpage

Nothing else needs to be done. quarto.yml will fetch the file named contact.qmd. The _redirected_contact.qmd redirects the website users to nettskjema's own website to fill the form.

8.1.5 Outlook.com Email

- Purpose: Login credentials for Netlify and ShinyApps.io
- Importance: Critical - this email provides access to necessary services

8.1.5.1 Maintenance Tasks:

- Keep email account secure
- accounts can be transferred to more secure users if neccessary.This can be done inside netlify and shinyapps.io. The procedures are easy.
  - in Netlify go to Profile - edit settings –> insert new profile credentials
  - In shinyapps.io go to Account - Settings - Transfer Account
    - Only necessary changes is later to change the .Renviron tokens and password if the meta-oxytocin account is transferred to a new Posit user.
- Monitor for service notifications
- Keep inbox organized to avoid missing important alerts (add profile on local computer or log in to browser version every now and then to monitor).

8.1.6 Zenodo

- Purpose: Archived backups of GitHub repository
- Website: https://zenodo.org
- User: iaiversen
- Cost: Free
- Features:
  - Each archive receives a unique DOI
  - Permanent preservation of repository snapshots
  - Citable versions for academic references
- Maintenance Tasks:
  - Create new archive after major release

8.2 Login Credentials Management

8.2.1 Storage Location

All login credentials are stored in:

- Location: Teams folder → AMORE → Website technicals
- Filename: `AMORE_credentials.xlsx`
- Access: Restricted to Teams member of BNE lab

8.3 Automated Maintenance Script

8.3.1 Overview

The AMORE project includes a comprehensive maintenance script (`scripts/Maintenance.R`) that automates the process of updating R packages and testing the website functionality. This script is designed to make maintenance safer and more systematic by:

- Creating automatic backups before changes
- Offering different update strategies based on risk tolerance
- Running comprehensive post-update tests
- Generating detailed maintenance logs

8.3.2 Explanation of script:

The script will run interactively, prompting you for decisions at key points.

Important: Always commit all your work to Git before running maintenance updates.

Section 1: Pre-Update Checks and Backup

The script first performs safety checks:

1. Git Status Check: Ensures your working directory is clean
2. Package Backup: Creates a snapshot of all current package versions
    - Saved as `docs/packages-backup-YYYYMMDD.csv`
    - Allows you to rollback if something breaks
3. Outdated Packages: Identifies which packages need updating
    - Shows current vs. available versions
    - Flags major version updates (higher risk)
4. Critical Packages: Highlights AMORE-essential packages

Section 2: Package Updates

You'll be presented with 6 update strategies

Option	Strategy	Risk Level	Recommended For
0	Skip updates (just test)	None	Testing current setup
1	Critical AMORE packages only	Lowest	Regular maintenance
2	Critical + high priority	Low	Recommended approach
3	All packages except major versions	Medium	Comprehensive updates
4	Everything including major versions	High	When major changes needed
5	Custom selection	Variable	Advanced users

Section 3: Post-Update Testing

After updates, the script runs comprehensive tests:

1. Basic Page Rendering: Tests `index.qmd` renders correctly
2. Project Page Rendering: Tests an LMA file renders correctly
3. Full Site Render: Renders entire website (can take a few minutes)
4. Shiny App Loading: Verifies the Shiny app loads without errors
5. ShinyApps.io Connection: Tests deployment credentials (if configured)
6. Link Checking: Optionally scans for broken links (can be slow)
