**GitHub Standards and Workflow Document**

**1. Repository Naming Conventions**

* **Repository Name**:
  + Use **kebab-case** (lowercase words separated by hyphens).
  + Be concise but descriptive.
  + Include the project name, type, or purpose.
  + Examples:
    - delta-smelt-analysis
    - fish-population-model
    - data-visualization-template
* **Private vs. Public**:
  + Use **private** repositories for projects involving unpublished data or sensitive information.
  + Use **public** repositories for projects that are published, open source, or available to collaborators.

**2. File and Folder Naming Conventions**

* **General Guidelines**:
  + Use **snake\_case** for files and directories (e.g., data\_cleaning\_script.R).
  + Avoid spaces and special characters.
  + Include version numbers (e.g., model\_v1.0.R).
* **Standard Folders**:
  + **data/**: Raw and processed data (use subfolders: raw/ and processed/).
  + **scripts/**: Analysis or processing scripts.
  + **docs/**: Documentation, protocols, and notes.
  + **results/**: Generated outputs like figures, tables, and reports.
  + **tests/**: Scripts or files for testing the workflow.

**3. Repository Setup Workflow**

When creating a new repository, follow these steps:

**Options for a New Repository**

1. **Initialize Repository**:
   * Check **“Add a README file”** to provide an overview of the project.
   * Check **“Add .gitignore”**:
     + Use a template that fits your project’s language or framework (e.g., R, Python).
   * Check **“Choose a license”**:
     + Select an appropriate open-source license like MIT or GPL for public repositories.
2. **Standard Files**:
   * Always include:
     + README.md (project overview, instructions, and contributors).
     + LICENSE (clarifies how others can use the code).
     + .gitignore (excludes sensitive or unnecessary files, e.g., .RData, \*.log).

**4. Workflow for Push/Pull/Commits**

**Branching Strategy**

* **Main Branch**: Use main for production-ready code.
* **Feature Branches**: Create feature branches for new tasks or bug fixes:
  + Naming: feature/<description> or fix/<issue-id>.
  + Example: feature/add-temperature-model or fix/issue-45.

**Pull and Push Workflow**

1. **Clone the Repository**:

bash

Copy code

git clone https://github.com/org/repo.git

cd repo

1. **Create a New Branch**:

bash

Copy code

git checkout -b feature/<branch-name>

Example:

bash

Copy code

git checkout -b feature/add-analysis-script

1. **Add Changes**:

bash

Copy code

git add <file\_name> # or use 'git add .' to stage all changes

1. **Commit Changes**:
   * Use descriptive commit messages.
   * Example format:

php

Copy code

<type>(<scope>): <message>

* + - feat: New feature
    - fix: Bug fix
    - docs: Documentation updates
    - style: Code style changes
    - refactor: Code restructuring
    - test: Adding or updating tests
    - chore: Maintenance tasks

Example:

bash

Copy code

git commit -m "feat(data-cleaning): add missing value imputation"

1. **Push to Remote**:

bash

Copy code

git push origin feature/<branch-name>

1. **Open a Pull Request (PR)**:
   * Provide a clear description of changes.
   * Link to relevant issues.
   * Assign reviewers for code review.
2. **Merge PR to main**:
   * After approval, merge using GitHub’s interface.
3. **Pull Changes**:
   * Sync your local repository:

bash

Copy code

git checkout main

git pull origin main

**5. Commit Message Standards**

Follow this structured format for commit messages:

**Format:**

php

Copy code

<type>(<scope>): <description>

**Examples:**

* **New Feature**:

scss

Copy code

feat(model): add population growth simulation

* **Bug Fix**:

scss

Copy code

fix(visualization): correct axis labels on plots

* **Documentation**:

scss

Copy code

docs(README): update usage instructions

**6. Collaboration Guidelines**

1. **Pull Before Push**:
   * Always pull the latest changes from the remote branch before pushing:

bash

Copy code

git pull origin main

1. **Resolve Conflicts Locally**:
   * If there’s a conflict, resolve it locally using your editor or terminal.
2. **Code Reviews**:
   * Submit PRs for all changes.
   * Use GitHub’s **Reviewers** feature to assign at least one reviewer.
   * Follow up on feedback promptly.

**7. Standardized Repository Files**

**README.md Template**

markdown

Copy code

# Project Title

## Description

Brief overview of the project.

## Setup Instructions

1. Clone the repository:

```bash

git clone https://github.com/org/repo.git

cd repo

1. Install dependencies:

bash

Copy code

Rscript -e "install.packages(c('tidyverse', 'ggplot2'))"

**Usage**

Describe how to run scripts or analyses.

**Contributors**

* Name (Role)
* Name (Role)

**License**

MIT License

markdown

Copy code

### \*\*.gitignore Example (for R Projects)\*\*