# Reproducible Research Workflow Guide

# Reproducible Research Workflow Guide

This guide outlines best practices for creating reproducible research workflows in the IDEEAS Lab. Following these practices ensures that your research can be understood, verified, and built upon by others.

## Core Principles

### 1. Everything is Documented

* Document your thinking, decisions, and processes
* Write code comments that explain the “why,” not just the “what”
* Keep a research log of key decisions and discoveries
* Document your computational environment

### 2. Everything is Version Controlled

* Use Git for all code, documentation, and text files
* Make frequent, meaningful commits with clear messages
* Use branches for experimental work
* Tag important milestones and releases

### 3. Everything is Automated

* Use scripts instead of manual processes
* Create reproducible computational environments
* Automate data processing and analysis pipelines
* Use continuous integration when appropriate

### 4. Everything is Accessible

* Organize files logically and consistently
* Use clear, descriptive file and variable names
* Provide clear instructions for running your code
* Share code and data when possible and appropriate

## Project Structure Template

project-name/
├── README.md # Project overview and quick start
├── LICENSE # License for code and data
├── .gitignore # Files to ignore in version control
├── environment.yml # Conda environment specification
├── requirements.txt # Python package requirements
├── Makefile # Automation scripts
│
├── data/
│ ├── raw/ # Original, immutable data
│ ├── interim/ # Intermediate processed data
│ ├── processed/ # Final, analysis-ready data
│ └── external/ # External datasets
│
├── notebooks/
│ ├── exploratory/ # Jupyter notebooks for exploration
│ ├── reports/ # Notebooks that generate reports
│ └── archive/ # Old notebooks for reference
│
├── src/ # Source code for the project
│ ├── \_\_init\_\_.py
│ ├── data/ # Scripts to download or generate data
│ ├── features/ # Scripts to turn raw data into features
│ ├── models/ # Scripts to train models and make predictions
│ ├── visualization/ # Scripts to create visualizations
│ └── utils/ # Utility functions and helpers
│
├── models/ # Trained and serialized models
├── reports/ # Generated analysis reports
│ ├── figures/ # Generated graphics and figures
│ └── tables/ # Generated tables
│
├── docs/ # Documentation
│ ├── data-dictionary.md # Description of data variables
│ ├── methodology.md # Detailed methodology
│ └── analysis-plan.md # Pre-registered analysis plan
│
└── tests/ # Unit tests for your code
 ├── \_\_init\_\_.py
 ├── test\_data.py
 ├── test\_features.py
 └── test\_models.py

## Version Control Best Practices

### Repository Setup

# Initialize repository
git init
git add README.md
git commit -m "Initial commit: Add README"

# Set up remote repository
git remote add origin https://github.com/ideeas-lab/project-name.git
git push -u origin main

### Commit Message Guidelines

Use clear, descriptive commit messages:

# Good examples
Add data cleaning script for survey responses
Fix bug in statistical analysis function
Update README with installation instructions

# Poor examples
Update
Fix stuff
Changes

### Branching Strategy

# Create feature branch
git checkout -b feature/data-analysis
# Work on feature
git add .
git commit -m "Add initial data analysis script"
# Push and create pull request
git push origin feature/data-analysis

### .gitignore Template

# Data files (add specific exceptions as needed)
data/raw/\*
data/interim/\*
data/processed/\*
!data/raw/.gitkeep
!data/interim/.gitkeep
!data/processed/.gitkeep

# Jupyter Notebook checkpoints
.ipynb\_checkpoints/

# Python
\_\_pycache\_\_/
\*.py[cod]
\*$py.class
\*.so
.Python
env/
venv/
.venv/

# IDE
.vscode/
.idea/
\*.swp
\*.swo

# OS
.DS\_Store
Thumbs.db

# Temporary files
\*.tmp
\*.temp
\*~

# Sensitive information
.env
config/secrets.yml

## Environment Management

### Conda Environment

Create environment.yml:

name: project-name
channels:
 - conda-forge
 - defaults
dependencies:
 - python=3.9
 - pandas
 - numpy
 - matplotlib
 - seaborn
 - scikit-learn
 - jupyter
 - pip
 - pip:
 - specific-pip-package==1.0.0

Setup and activation:

# Create environment
conda env create -f environment.yml

# Activate environment
conda activate project-name

# Update environment file
conda env export > environment.yml

### Python Requirements

Create requirements.txt:

pandas==1.3.3
numpy==1.21.2
matplotlib==3.4.3
seaborn==0.11.2
scikit-learn==1.0.1
jupyter==1.0.0

## Code Organization

### Function Documentation

def clean\_survey\_data(df, remove\_incomplete=True):
 """
 Clean survey response data by handling missing values and outliers.

 Parameters
 ----------
 df : pandas.DataFrame
 Raw survey data with responses
 remove\_incomplete : bool, default True
 Whether to remove rows with incomplete responses

 Returns
 -------
 pandas.DataFrame
 Cleaned survey data

 Examples
 --------
 >>> cleaned\_data = clean\_survey\_data(raw\_data, remove\_incomplete=False)
 """
 # Implementation here
 pass

### Configuration Management

Create config.py:

"""Configuration settings for the project."""

# Data paths
RAW\_DATA\_PATH = "data/raw/"
PROCESSED\_DATA\_PATH = "data/processed/"
FIGURES\_PATH = "reports/figures/"

# Analysis parameters
RANDOM\_SEED = 42
TEST\_SIZE = 0.2
N\_FOLDS = 5

# Model parameters
MODEL\_PARAMS = {
 'random\_forest': {
 'n\_estimators': 100,
 'random\_state': RANDOM\_SEED
 }
}

### Logging Setup

import logging

# Configure logging
logging.basicConfig(
 level=logging.INFO,
 format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
 handlers=[
 logging.FileHandler('analysis.log'),
 logging.StreamHandler()
 ]
)

logger = logging.getLogger(\_\_name\_\_)

## Data Management

### Data Pipeline Structure

def main():
 """Main data processing pipeline."""
 logger.info("Starting data processing pipeline")

 # Load raw data
 raw\_data = load\_raw\_data()
 logger.info(f"Loaded {len(raw\_data)} raw records")

 # Clean data
 clean\_data = clean\_survey\_data(raw\_data)
 logger.info(f"Cleaned data: {len(clean\_data)} records remaining")

 # Feature engineering
 features = create\_features(clean\_data)
 logger.info(f"Created {features.shape[1]} features")

 # Save processed data
 save\_processed\_data(features)
 logger.info("Data processing complete")

if \_\_name\_\_ == "\_\_main\_\_":
 main()

### Data Validation

def validate\_data(df):
 """Validate data quality and structure."""
 assert not df.empty, "DataFrame is empty"
 assert 'participant\_id' in df.columns, "Missing participant\_id column"
 assert df['participant\_id'].nunique() == len(df), "Duplicate participant IDs"

 # Check for expected value ranges
 assert df['age'].between(18, 100).all(), "Age values out of expected range"

 logger.info("Data validation passed")

## Analysis Documentation

### Analysis Plan Template

Create docs/analysis-plan.md:

# Analysis Plan

## Research Questions
1. Primary research question
2. Secondary research questions

## Hypotheses
- H1: [Specific hypothesis]
- H2: [Specific hypothesis]

## Variables
### Dependent Variables
- Variable 1: [Description, measurement]
- Variable 2: [Description, measurement]

### Independent Variables
- Variable 1: [Description, measurement]
- Variable 2: [Description, measurement]

## Statistical Analysis Plan
### Descriptive Statistics
- [What descriptive analyses will be conducted]

### Inferential Statistics
- [What statistical tests will be used]
- [Multiple comparison corrections]
- [Effect size measures]

### Model Specifications
- [Specific models to be fit]
- [Model assumptions to be checked]

## Sample Size and Power
- [Power analysis results]
- [Minimum detectable effect size]

### Results Documentation

def document\_results(results, filename):
 """Document analysis results in a structured format."""
 with open(f"reports/{filename}", 'w') as f:
 f.write("# Analysis Results\n\n")
 f.write(f"Analysis conducted on: {datetime.now()}\n\n")

 for test\_name, result in results.items():
 f.write(f"## {test\_name}\n")
 f.write(f"- Test statistic: {result['statistic']:.3f}\n")
 f.write(f"- p-value: {result['p\_value']:.3f}\n")
 f.write(f"- Effect size: {result['effect\_size']:.3f}\n\n")

## Testing and Validation

### Unit Testing

Create tests/test\_data.py:

import unittest
import pandas as pd
from src.data.clean import clean\_survey\_data

class TestDataCleaning(unittest.TestCase):

 def setUp(self):
 """Set up test data."""
 self.sample\_data = pd.DataFrame({
 'participant\_id': [1, 2, 3, 4],
 'age': [25, 30, None, 35],
 'response': ['A', 'B', 'C', None]
 })

 def test\_clean\_survey\_data\_removes\_missing(self):
 """Test that missing data is handled correctly."""
 result = clean\_survey\_data(self.sample\_data, remove\_incomplete=True)
 self.assertEqual(len(result), 2) # Should remove rows with missing data

 def test\_clean\_survey\_data\_keeps\_missing(self):
 """Test that missing data is kept when specified."""
 result = clean\_survey\_data(self.sample\_data, remove\_incomplete=False)
 self.assertEqual(len(result), 4) # Should keep all rows

if \_\_name\_\_ == '\_\_main\_\_':
 unittest.main()

### Run Tests

# Run all tests
python -m pytest tests/

# Run specific test file
python -m pytest tests/test\_data.py

# Run with coverage
python -m pytest --cov=src tests/

## Automation and Reproducibility

### Makefile Template

.PHONY: data features models reports clean test

# Default target
all: data features models reports

# Data processing
data:
 python src/data/download\_data.py
 python src/data/clean\_data.py

# Feature engineering
features: data
 python src/features/build\_features.py

# Model training
models: features
 python src/models/train\_model.py

# Generate reports
reports: models
 python src/visualization/make\_plots.py
 jupyter nbconvert --to html notebooks/reports/final\_report.ipynb

# Run tests
test:
 python -m pytest tests/

# Clean generated files
clean:
 rm -rf data/interim/\*
 rm -rf data/processed/\*
 rm -rf models/\*
 rm -rf reports/figures/\*

# Set up environment
setup:
 conda env create -f environment.yml
 conda activate project-name
 pip install -e .

### README Template

# Project Name

Brief description of the project and its goals.

## Getting Started

### Prerequisites
- Python 3.9+
- Conda or pip

### Installation
```bash
# Clone repository
git clone https://github.com/ideeas-lab/project-name.git
cd project-name

# Set up environment
conda env create -f environment.yml
conda activate project-name

# Install package in development mode
pip install -e .

### Usage

# Run full analysis pipeline
make all

# Run individual steps
make data
make features
make models
make reports

## Project Structure

[Describe the organization of files and directories]

## Data

[Describe the data sources and structure]

## Methods

[Brief overview of methodology]

## Results

[Summary of key findings]

## Contributing

[Guidelines for contributors]

## License

[License information] ```

## Sharing and Publication

### Pre-publication Checklist

* All code is documented and tested
* Data is properly documented with data cards
* Analysis is reproducible from raw data
* Sensitive information is removed or protected
* Code repository is clean and organized
* README provides clear instructions
* License is specified
* Dependencies are clearly specified

### Data and Code Sharing

* Use appropriate repositories (GitHub, Zenodo, etc.)
* Include DOIs for permanent citation
* Follow journal and funder requirements
* Consider embargo periods if needed
* Provide clear usage guidelines

**Remember**: Reproducibility is not just about the final product - it’s about creating sustainable practices that make your research more efficient, reliable, and impactful throughout the entire process.