Package 'Capsule'

May 7, 2026

Type Package

Title Comprehensive Reproducibility Framework for R and Bioinformatics Analysis

Version 0.2.0

Description A comprehensive reproducibility framework designed for R and bioinformatics workflows. Automatically captures the entire analysis environment including R session info, package versions, external tool versions ('Samtools', 'STAR', 'BWA', etc.), 'conda' environments, reference genomes, data provenance with smart checksumming for large files, parameter choices, random seeds, and hardware specifications. Generates executable scripts with 'Docker', 'Singularity', and 'renv' configurations. Integrates with workflow managers ('Nextflow', 'Snakemake', 'WDL', 'CWL') to ensure complete reproducibility of computational research workflows.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.0.0)

Imports renv, jsonlite, digest, yaml, cli, utils

Suggests testthat (>= 3.0.0)

BugReports https://github.com/SAADAT-Abu/Capsule/issues

RoxygenNote 7.3.3

NeedsCompilation no

Author Abu SAADAT [aut, cre] (ORCID: <https://orcid.org/0000-0001-9636-2513>, affiliation: Università degli Studi della Campania Luigi Vanvitelli)

Maintainer Abu SAADAT <saadatabu1996@gmail.com>

Repository CRAN

Date/Publication 2025-11-11 10:00:29 UTC

Contents

capture_environment 2

capture_hardware . . . 3
capture_session . . . 4
capture_system_libraries . . . 5
compare_snapshots . . . 5
create_renv_lockfile . . . 6
create_repro_report . . . 7
export_for_cwl . . . 7
export_for_nextflow . . . 8
export_for_snakemake . . . 9
export_for_wdl . . . 9
generate_docker . . . 10
generate_repro_script . . . 11
generate_singularity . . . 12
get_conda_env_info . . . 13
get_data_lineage . . . 13
get_param_history . . . 14
get_reference_info . . . 14
get_seed_history . . . 15
get_tool_versions . . . 16
init_capsule . . . 16
list_reference_sources . . . 17
list_snapshots . . . 18
restore_conda_env . . . 18
restore_seed . . . 19
set_seed . . . 20
snapshot_packages . . . 21
snapshot_workflow . . . 22
track_conda_env . . . 23
track_data . . . 24
track_external_tools . . . 25
track_params . . . 26
track_reference_genome . . . 26
verify_data . . . 28

Index **29**

`capture_environment` *Capture Environment State*

Description

Captures the current global environment state including objects and their types

Usage

```
capture_environment(
  output_file = NULL,
  include_values = FALSE,
  max_size = 1024 * 1024
)
```

Arguments

output_file	Character. Path to save environment info. If NULL, returns as list.
include_values	Logical. Whether to include object values (for small objects). Default FALSE.
max_size	Numeric. Maximum object size (in bytes) to include values. Default 1MB.

Value

A list containing environment information

Examples

```
## Not run:
x <- 1:10
y <- "test"
capture_environment("env_state.json")

## End(Not run)
```

---

capture_hardware *Capture Hardware Information*

Description

Capture hardware specifications including CPU, RAM, and GPU information. Useful for documenting computational resources used in analysis.

Usage

```
capture_hardware(output_file = NULL)
```

Arguments

output_file	Character. Path to save hardware info. If NULL, returns as list.

Value

List containing hardware information

Examples

```
## Not run:
capture_hardware("hardware_info.json")

## End(Not run)
```

`capture_session` *Capture Complete Session Information*

Description

Captures comprehensive R session information including R version, platform, loaded packages, system information, and locale settings.

Usage

```
capture_session(output_file = NULL, format = c("json", "yaml", "rds"))
```

Arguments

`output_file`	Character. Path to save the session info. If NULL, returns as list.
`format`	Character. Output format: "json", "yaml", or "rds". Default is "json".

Value

A list containing session information, invisibly returned

Examples

```
## Not run:
# Capture session info to JSON
capture_session("session_info.json")

# Capture and return as list
info <- capture_session()

## End(Not run)
```

capture_system_libraries *Capture System Libraries*

Description

Capture version information for system libraries that R packages depend on (e.g., libcurl, libxml2, BLAS/LAPACK implementations)

Usage

```
capture_system_libraries(output_file = NULL)
```

Arguments

output_file	Character. Path to save library info. If NULL, returns as list.

Value

List containing system library information

Examples

```
## Not run:
capture_system_libraries("system_libs.json")

## End(Not run)
```

compare_snapshots *Compare Two Workflow Snapshots*

Description

Compare two Capsule snapshots to identify differences in packages, parameters, data files, and other tracked components

Usage

```
compare_snapshots(snapshot1, snapshot2, output_file)
```

Arguments

snapshot1	Character. Name of first snapshot
snapshot2	Character. Name of second snapshot
output_file	Character. Path to save comparison report (required).

Value

List containing comparison results

Examples

```
## Not run:
compare_snapshots("analysis_v1", "analysis_v2",
                  output_file = tempfile(fileext = ".md"))

## End(Not run)
```

---

create_renv_lockfile *Create renv Lockfile*

Description

Generate an renv-compatible lockfile for package reproducibility

Usage

```
create_renv_lockfile(output_file, project_path = ".")
```

Arguments

output_file	Character. Path to save lockfile (required).
project_path	Character. Path to project. Default is current directory.

Value

Path to created lockfile

Examples

```
## Not run:
create_renv_lockfile(output_file = tempfile(fileext = ".lock"))

## End(Not run)
```

create_repro_report *Create Reproducibility Report*

Description

Generate a comprehensive markdown report documenting all reproducibility information

Usage

```
create_repro_report(
  output_file,
  analysis_name = NULL,
  include_package_list = TRUE
)
```

Arguments

output_file	Character. Path to save the report (required).
analysis_name	Character. Name of the analysis
include_package_list	Logical. Include full package list. Default TRUE.

Value

Path to generated report

Examples

```
## Not run:
create_repro_report(tempfile(fileext = ".md"), "main_analysis")

## End(Not run)
```

export_for_cwl *Generate CWL (Common Workflow Language) Input*

Description

Export Capsule data in YAML format suitable for CWL workflows

Usage

```
export_for_cwl(output_file)
```

Arguments

output_file	Character. Path to save inputs (required).

Value

List containing input data

Examples

```
## Not run:
export_for_cwl(tempfile(fileext = ".yml"))

## End(Not run)
```

---

export_for_nextflow *Export Capsule Data for Nextflow*

Description

Export all Capsule tracking data in a format suitable for Nextflow pipelines

Usage

```
export_for_nextflow(output_file, include_checksums = TRUE)
```

Arguments

output_file	Character. Path to save manifest (required).
include_checksums	Logical. Include file checksums. Default TRUE.

Value

List containing manifest data

Examples

```
## Not run:
export_for_nextflow(tempfile(fileext = ".json"))

## End(Not run)
```

export_for_snakemake *Export Capsule Data for Snakemake*

Description

Export all Capsule tracking data in YAML format for Snakemake pipelines

Usage

```
export_for_snakemake(output_file, include_checksums = TRUE)
```

Arguments

output_file	Character. Path to save config (required).
include_checksums	Logical. Include file checksums. Default TRUE.

Value

List containing config data

Examples

```
## Not run:
export_for_snakemake(tempfile(fileext = ".yaml"))

## End(Not run)
```

export_for_wdl *Create WDL (Workflow Description Language) Config*

Description

Export Capsule data in JSON format suitable for WDL workflows

Usage

```
export_for_wdl(output_file)
```

Arguments

output_file	Character. Path to save config (required).

Value

List containing config data

Examples

```
## Not run:
export_for_wdl(tempfile(fileext = ".json"))

## End(Not run)
```

`generate_docker` *Generate Docker Configuration*

Description

Generate a Dockerfile and docker-compose.yml for complete environment reproducibility

Usage

```
generate_docker(
  output_dir,
  r_version = NULL,
  base_image = "rocker/r-ver",
  system_deps = NULL,
  project_name = "reproflow-project",
  include_rstudio = FALSE
)
```

Arguments

`output_dir`	Character. Directory to save Docker files (required).
`r_version`	Character. R version to use. Default is current R version.
`base_image`	Character. Base Docker image. Default "rocker/r-ver"
`system_deps`	Character vector. System dependencies to install
`project_name`	Character. Name for the project
`include_rstudio`	Logical. Include RStudio Server. Default FALSE.

Value

List of generated file paths

Examples

```
## Not run:
generate_docker(
  output_dir = tempdir(),
  project_name = "my_analysis",
  system_deps = c("libcurl4-openssl-dev", "libxml2-dev")
)

## End(Not run)
```

`generate_repro_script` *Generate Reproducible Script*

Description

Generate an executable R script that includes all reproducibility information including package versions, seeds, parameters, and data verification.

Usage

```
generate_repro_script(
  script_file,
  source_script = NULL,
  analysis_name = "analysis",
  include_renv = TRUE,
  include_data_check = TRUE,
  include_session_info = TRUE
)
```

Arguments

`script_file`	Character. Path to save the generated script
`source_script`	Character. Original analysis script to include
`analysis_name`	Character. Name for this analysis
`include_renv`	Logical. Include renv initialization. Default TRUE.
`include_data_check`	Logical. Include data verification. Default TRUE.
`include_session_info`	Logical. Include session info at end. Default TRUE.

Value

Path to generated script

Examples

```
## Not run:
generate_repro_script(
  "analysis_reproducible.R",
  source_script = "analysis.R",
  analysis_name = "main_analysis"
)

## End(Not run)
```

`generate_singularity` *Generate Singularity Definition File*

Description

Generate a Singularity/Apptainer definition file for HPC environments. Singularity is commonly used in HPC clusters where Docker is not available.

Usage

```
generate_singularity(
  output_dir,
  r_version = NULL,
  base_image = "rocker/r-ver",
  conda_env = NULL,
  system_deps = NULL,
  project_name = "reproflow-project"
)
```

Arguments

`output_dir`	Character. Directory to save Singularity files (required).
`r_version`	Character. R version to use. Default is current R version.
`base_image`	Character. Base Docker image. Default "rocker/r-ver"
`conda_env`	Character. Path to conda environment file. Optional.
`system_deps`	Character vector. System dependencies to install
`project_name`	Character. Name for the project

Value

List of generated file paths

Examples

```
## Not run:
generate_singularity(
  output_dir = tempdir(),
  project_name = "my_analysis",
  system_deps = c("samtools", "bwa")
)

## End(Not run)
```

get_conda_env_info *Get Conda Environment Info*

Description

Retrieve information about tracked conda environments

Usage

```
get_conda_env_info(
  env_name = NULL,
  registry_file = ".capsule/conda_registry.json"
)
```

Arguments

env_name	Character. Specific environment name, or NULL for all
registry_file	Character. Path to conda registry

Value

List of environment information

get_data_lineage *Get Data Lineage*

Description

Retrieve complete lineage information for tracked data

Usage

```
get_data_lineage(
  data_path = NULL,
  registry_file = ".capsule/data_registry.json"
)
```

Arguments

data_path	Character. Path to data file. If NULL, returns all lineage.
registry_file	Character. Path to provenance registry.

Value

List containing lineage information

Examples

```
## Not run:
# Get lineage for specific file
lineage <- get_data_lineage("data/mydata.csv")

# Get all lineage
all_lineage <- get_data_lineage()

## End(Not run)
```

`get_param_history` *Get Parameter History*

Description

Retrieve parameter tracking history

Usage

```
get_param_history(
  analysis_name = NULL,
  registry_file = ".capsule/param_registry.json"
)
```

Arguments

`analysis_name`	Character. Specific analysis name, or NULL for all
`registry_file`	Character. Path to parameter registry

Value

List of parameter records

`get_reference_info` *Get Reference Genome Information*

Description

Retrieve information about tracked reference genomes

Usage

```
get_reference_info(
  genome_build = NULL,
  registry_file = ".capsule/reference_registry.json"
)
```

Arguments

genome_build	Character. Specific genome build, or NULL for all
registry_file	Character. Path to reference registry

Value

List of reference genome information

Examples

```
## Not run:
# Get all tracked references
get_reference_info()

# Get specific reference
get_reference_info("GRCh38")

## End(Not run)
```

---

get_seed_history *Get Seed History*

Description

Retrieve seed tracking history

Usage

```
get_seed_history(
  analysis_name = NULL,
  registry_file = ".capsule/seed_registry.json"
)
```

Arguments

analysis_name	Character. Specific analysis name, or NULL for all
registry_file	Character. Path to seed registry

Value

List of seed records

get_tool_versions *Get External Tool Versions*

Description

Retrieve version information for previously tracked external tools

Usage

```
get_tool_versions(
  tool_name = NULL,
  registry_file = ".capsule/tools_registry.json"
)
```

Arguments

tool_name	Character. Specific tool name, or NULL for all tools
registry_file	Character. Path to tools registry

Value

List of tool version information

Examples

```
## Not run:
# Get all tracked tools
get_tool_versions()

# Get specific tool
get_tool_versions("samtools")

## End(Not run)
```

init_capsule *Initialize Capsule in Project*

Description

Initialize Capsule reproducibility framework in the current project. Creates necessary directory structure and configuration files.

Usage

```
init_capsule(
  project_path = ".",
  use_renv = TRUE,
  use_git = TRUE,
  create_gitignore = TRUE
)
```

Arguments

project_path	Character. Path to project directory. Default is current directory.
use_renv	Logical. Initialize renv for package management. Default TRUE.
use_git	Logical. Initialize git if not already present. Default TRUE.
create_gitignore	Logical. Create/update .gitignore. Default TRUE.

Value

Invisible NULL

Examples

```
## Not run:
# Initialize Capsule in current directory
init_capsule()

# Initialize without renv
init_capsule(use_renv = FALSE)

## End(Not run)
```

---

list_reference_sources

List Common Reference Genome Sources

Description

Display a helpful list of common reference genome sources

Usage

```
list_reference_sources()
```

Value

No return value, called for side effects (displays reference sources)

Examples

```
list_reference_sources()
```

`list_snapshots` *List Available Snapshots*

Description

List all available snapshots with basic metadata

Usage

```
list_snapshots()
```

Value

Data frame with snapshot information

Examples

```
## Not run:
list_snapshots()

## End(Not run)
```

`restore_conda_env` *Restore Conda Environment*

Description

Restore a conda environment from a previously exported environment file

Usage

```
restore_conda_env(
  env_file = "conda_environment.yml",
  env_name = NULL,
  use_mamba = FALSE,
  force = FALSE
)
```

Arguments

`env_file`	Character. Path to environment YAML file. Default "conda_environment.yml"
`env_name`	Character. Name for the new environment. If NULL, uses name from file.
`use_mamba`	Logical. Use mamba instead of conda. Default FALSE.
`force`	Logical. Remove existing environment if it exists. Default FALSE.

Value

Logical. TRUE if successful, FALSE otherwise

Examples

```
## Not run:
# Restore environment from file
restore_conda_env("conda_environment.yml")

# Use mamba for faster installation
restore_conda_env("conda_environment.yml", use_mamba = TRUE)

# Force recreate if exists
restore_conda_env("conda_environment.yml", force = TRUE)

## End(Not run)
```

---

restore_seed *Restore Random Seed*

Description

Restore a previously tracked random seed

Usage

```
restore_seed(analysis_name, registry_file = ".capsule/seed_registry.json")
```

Arguments

analysis_name	Character. Name of analysis to restore seed from
registry_file	Character. Path to seed registry

Value

The seed value (invisibly)

Examples

```
## Not run:
# Restore previously tracked seed
restore_seed("simulation_1")

## End(Not run)
```

set_seed *Set and Track Random Seed*

Description

Set a random seed and track it for reproducibility. Note: This function is explicitly designed to set random seeds as requested by the user.

Usage

```
set_seed(
  seed = NULL,
  kind = NULL,
  normal.kind = NULL,
  sample.kind = NULL,
  analysis_name = NULL,
  registry_file,
  set_seed = TRUE
)
```

Arguments

seed	Numeric. Random seed to set. If NULL, generates random seed.
kind	Character. RNG kind (see ?set.seed). Default NULL uses current.
normal.kind	Character. Normal RNG kind. Default NULL uses current.
sample.kind	Character. Sample RNG kind. Default NULL uses current.
analysis_name	Character. Name to associate with this seed
registry_file	Character. Path to seed registry (required).
set_seed	Logical. If TRUE, actually sets the seed. If FALSE, only tracks it. Default TRUE.

Value

The seed value (invisibly)

Examples

```
## Not run:
# Set and track a specific seed
set_seed(12345, analysis_name = "simulation_1",
         registry_file = tempfile(fileext = ".json"))

# Generate and track a random seed
set_seed(analysis_name = "bootstrap_analysis",
         registry_file = tempfile(fileext = ".json"))

## End(Not run)
```

snapshot_packages *Track Package Versions and Dependencies*

Description

Creates a comprehensive snapshot of all installed packages, their versions, dependencies, and sources for reproducibility.

Usage

```
snapshot_packages(
  output_file = NULL,
  include_dependencies = TRUE,
  only_attached = FALSE
)
```

Arguments

output_file	Character. Path to save package info. If NULL, returns as list.
include_dependencies	Logical. Include dependency tree. Default TRUE.
only_attached	Logical. Only track attached packages. Default FALSE.

Value

A list containing package information

Examples

```
## Not run:
# Track all installed packages
snapshot_packages("package_manifest.json")

# Track only attached packages
snapshot_packages("packages.json", only_attached = TRUE)

## End(Not run)
```

snapshot_workflow *Create Complete Workflow Snapshot*

Description

Create a comprehensive snapshot of the entire workflow including session info, packages, data, parameters, and generate all reproducibility artifacts.

Usage

```
snapshot_workflow(
  snapshot_name = NULL,
  analysis_name = "analysis",
  source_script = NULL,
  description = NULL,
  generate_docker = TRUE,
  generate_script = TRUE,
  generate_report = TRUE
)
```

Arguments

snapshot_name	Character. Name for this snapshot. Default is timestamp.
analysis_name	Character. Name of the analysis
source_script	Character. Path to main analysis script
description	Character. Description of this workflow
generate_docker	Logical. Generate Docker configuration. Default TRUE.
generate_script	Logical. Generate reproducible script. Default TRUE.
generate_report	Logical. Generate reproducibility report. Default TRUE.

Value

List containing paths to generated files

Examples

```
## Not run:
# Create complete workflow snapshot
snapshot_workflow(
  snapshot_name = "analysis_v1",
  analysis_name = "main_analysis",
  source_script = "analysis.R",
  description = "Initial analysis run"
```

```
)

## End(Not run)
```

`track_conda_env` *Track Conda Environment*

Description

Export and track a conda environment specification for reproducibility. Works with both conda and mamba.

Usage

```
track_conda_env(env_name = NULL, output_file, use_mamba = FALSE, registry_file)
```

Arguments

`env_name`	Character. Name of conda environment. If NULL, uses active environment.
`output_file`	Character. Path to save environment file (required).
`use_mamba`	Logical. Use mamba instead of conda. Default FALSE.
`registry_file`	Character. Path to conda registry (required).

Value

List containing environment information

Examples

```
## Not run:
# Track currently active conda environment
track_conda_env(output_file = tempfile(fileext = ".yml"),
                registry_file = tempfile(fileext = ".json"))

# Track specific environment
track_conda_env(env_name = "bioinfo_env",
                output_file = tempfile(fileext = ".yml"),
                registry_file = tempfile(fileext = ".json"))

# Use mamba instead
track_conda_env(use_mamba = TRUE,
                output_file = tempfile(fileext = ".yml"),
                registry_file = tempfile(fileext = ".json"))

## End(Not run)
```

`track_data` *Track Data Provenance*

Description

Records comprehensive provenance information for data files including checksums, sources, timestamps, and metadata. Supports fast hashing for large files.

Usage

```
track_data(
  data_path,
  source = c("downloaded", "generated", "manual", "reference", "other"),
  source_url = NULL,
  description = NULL,
  metadata = NULL,
  fast_hash = TRUE,
  size_threshold_gb = 1,
  registry_file
)
```

Arguments

`data_path`	Character. Path to data file or directory.
`source`	Character. Source of the data (e.g., "downloaded", "generated", "manual", "reference").
`source_url`	Character. URL if data was downloaded. Optional.
`description`	Character. Description of the data. Optional.
`metadata`	List. Additional metadata. Optional.
`fast_hash`	Logical. Use faster xxHash for large files (>1GB). Default TRUE.
`size_threshold_gb`	Numeric. Size threshold (GB) for using fast hash. Default 1.
`registry_file`	Character. Path to provenance registry (required).

Value

A list containing data provenance information

Examples

```
## Not run:
# Track a downloaded dataset
track_data("data/mydata.csv",
  source = "downloaded",
  source_url = "https://example.com/data.csv",
  description = "Customer data from API",
```

```
  registry_file = tempfile(fileext = ".json")
)

# Track generated data
track_data("results/simulation.rds",
  source = "generated",
  description = "Monte Carlo simulation results",
  registry_file = tempfile(fileext = ".json")
)

# Track large file with fast hashing
track_data("data/large_file.bam",
  source = "generated",
  fast_hash = TRUE,
  registry_file = tempfile(fileext = ".json")
)

## End(Not run)
```

`track_external_tools` *Track External Bioinformatics Tools*

Description

Track versions of external command-line tools commonly used in bioinformatics pipelines (e.g., samtools, STAR, BWA, etc.)

Usage

```
track_external_tools(tools = NULL, registry_file)
```

Arguments

`tools`	Character vector of tool names to track. If NULL, tracks common tools.
`registry_file`	Character. Path to tools registry. Default ".capsule/tools_registry.json"

Value

List containing tool version information

Examples

```
## Not run:
# Track common bioinformatics tools
track_external_tools(registry_file = tempfile(fileext = ".json"))

# Track specific tools
track_external_tools(c("samtools", "bwa", "STAR"),
                     registry_file = tempfile(fileext = ".json"))

## End(Not run)
```

track_params *Track Analysis Parameters*

Description

Record analysis parameters and configuration settings for reproducibility

Usage

```
track_params(params, analysis_name = NULL, description = NULL, registry_file)
```

Arguments

params	Named list of parameters to track
analysis_name	Character. Name/identifier for this analysis
description	Character. Description of what these parameters control
registry_file	Character. Path to parameter registry (required).

Value

List containing parameter information

Examples

```
## Not run:
# Track model parameters
params <- list(
  learning_rate = 0.01,
  epochs = 100,
  batch_size = 32,
  model_type = "neural_network"
)
track_params(params, "model_training", "Deep learning model parameters",
             registry_file = tempfile(fileext = ".json"))

## End(Not run)
```

track_reference_genome *Track Reference Genome*

Description

Track reference genome files, annotations, and indices for reproducibility. This is critical for genomics/transcriptomics pipelines where the exact reference version affects results.

Usage

```
track_reference_genome(
  fasta_path,
  gtf_path = NULL,
  gff_path = NULL,
  genome_build = NULL,
  species = NULL,
  source_url = NULL,
  indices = list(),
  metadata = list(),
  registry_file,
  data_registry_file
)
```

Arguments

fasta_path	Character. Path to reference genome FASTA file
gtf_path	Character. Path to GTF annotation file. Optional.
gff_path	Character. Path to GFF annotation file. Optional.
genome_build	Character. Genome build identifier (e.g., "GRCh38", "mm10")
species	Character. Species name (e.g., "Homo sapiens", "Mus musculus")
source_url	Character. URL where reference was downloaded from
indices	Named list. Paths to aligner indices (STAR, BWA, etc.)
metadata	List. Additional metadata about the reference
registry_file	Character. Path to reference registry (required).
data_registry_file	Character. Path to data registry for tracking files (required).

Value

List containing reference genome information

Examples

```
## Not run:
track_reference_genome(
  fasta_path = "ref/GRCh38.fa",
  gtf_path = "ref/gencode.v38.annotation.gtf",
  genome_build = "GRCh38",
  species = "Homo sapiens",
  source_url = "https://www.gencodegenes.org/",
  indices = list(
    star = "ref/STAR_index/",
    bwa = "ref/bwa_index/GRCh38"
  ),
  registry_file = tempfile(fileext = ".json"),
  data_registry_file = tempfile(fileext = ".json")
```

```
)

## End(Not run)
```

`verify_data` *Verify Data Integrity*

Description

Verify that tracked data files have not been modified by comparing checksums

Usage

```
verify_data(data_path = NULL, registry_file = ".capsule/data_registry.json")
```

Arguments

`data_path`	Character. Path to specific file, or NULL to verify all tracked files.
`registry_file`	Character. Path to provenance registry.

Value

Logical. TRUE if data is unchanged, FALSE otherwise

Examples

```
## Not run:
# Verify specific file
verify_data("data/mydata.csv")

# Verify all tracked files
verify_data()

## End(Not run)
```

Index

capture_environment, 2
capture_hardware, 3
capture_session, 4
capture_system_libraries, 5
compare_snapshots, 5
create_renv_lockfile, 6
create_repro_report, 7

export_for_cwl, 7
export_for_nextflow, 8
export_for_snakemake, 9
export_for_wdl, 9

generate_docker, 10
generate_repro_script, 11
generate_singularity, 12
get_conda_env_info, 13
get_data_lineage, 13
get_param_history, 14
get_reference_info, 14
get_seed_history, 15
get_tool_versions, 16

init_capsule, 16

list_reference_sources, 17
list_snapshots, 18

restore_conda_env, 18
restore_seed, 19

set_seed, 20
snapshot_packages, 21
snapshot_workflow, 22

track_conda_env, 23
track_data, 24
track_external_tools, 25
track_params, 26
track_reference_genome, 26

verify_data, 28