CLAUDE COWORK

Bio-Research Plugin Handbook

The Complete Guide to Preclinical Research Tools and Workflows

---

VERSION 1.0.0

PLUGIN bio-research

RationalEyes.ai

contact@rationaleyes.ai

February 2026

Table of Contents

The Complete Guide to Preclinical Research Tools and Workflows

Plugin: bio-research
Version: 1.0.0
Author: Anthropic
Audience: Life sciences researchers, lab managers, bioinformaticians, and research scientists

PART I: OVERVIEW AND GETTING STARTED

1. What Is the Bio-Research Plugin?

The bio-research plugin consolidates preclinical research tools, literature databases, and analysis workflows into a single integrated package for life sciences researchers. It provides access to 10 MCP server integrations and 6 specialized analysis skills covering literature search, genomics analysis, target prioritization, clinical trial design, and research strategy.

Design Philosophy

This plugin is built for bench scientists and computational biologists who need to:

- Search and synthesize biomedical literature rapidly
- Analyze single-cell RNA-seq, bulk RNA-seq, WGS/WES, and ATAC-seq data
- Design clinical trial protocols following FDA/NIH guidelines
- Convert instrument data to standardized formats
- Make strategic research decisions using systematic frameworks

The plugin emphasizes automation, reproducibility, and best practices from the scientific community (scverse, nf-core, FDA, Fischbach & Walsh framework).

What Makes This Plugin Complex

With 88 files including extensive reference materials, the bio-research plugin is the most comprehensive plugin in the Cowork library. This complexity reflects the breadth of life sciences research:

COMPONENT	COUNT	PURPOSE
MCP Servers	10 (+2 optional)	Connect to PubMed, ChEMBL, ClinicalTrials.gov, BioRender, bioRxiv, Wiley, Synapse, Open Targets, Owkin, Benchling
Skills	6	scRNA QC, scvi-tools, Nextflow, clinical trials, Allotrope conversion, problem selection
Reference Files	47	Detailed methodology, best practices, instrument support
Scripts	35	Python utilities for data analysis and conversion
Total Files	88	Complete research toolkit

Target Users

ROLE	PRIMARY USE CASES
Research Scientists	Literature review, experimental design, data analysis
Lab Managers	Data standardization, LIMS integration, QC workflows
Bioinformaticians	Pipeline automation, single-cell analysis, integration
Clinical Researchers	Trial design, protocol generation, regulatory research
Graduate Students/Postdocs	Learning best practices, reproducible workflows

Core Capabilities

The plugin provides four major capability areas:

1. **Literature and Knowledge Discovery**
   - Search PubMed, bioRxiv, medRxiv for research articles
   - Access full-text publications through Wiley Scholar Gateway
   - Explore drug targets via Open Targets
   - Query chemical databases (ChEMBL)
   - Review clinical trials (ClinicalTrials.gov)
2. **Genomics Data Analysis**
   - Quality control for single-cell RNA-seq data
   - Deep learning models for batch correction and integration
   - Run nf-core pipelines (RNA-seq, variant calling, ATAC-seq)
   - Analyze public GEO/SRA datasets
3. **Clinical and Regulatory**
   - Generate FDA/NIH-compliant clinical trial protocols
   - Research similar trials and regulatory pathways
   - Statistical sample size calculations
4. **Data Standardization and Strategy**
   - Convert instrument output to Allotrope format
   - Systematic research problem selection framework

---

2. Installation and Setup

Prerequisites

- Claude Code or Claude Cowork desktop application
- Python 3.8+ (for analysis skills)
- Docker (for Nextflow pipelines)
- 8GB+ RAM recommended for single-cell analysis

Installation

Cowork (recommended): Open **Plugin Settings** in the Cowork desktop app, find **Bio-Research**, and click **Install.** The plugin activates immediately — no CLI required.

Claude Code CLI (alternative): If you are using Claude Code in the terminal rather than Cowork, install via:

```
claude plugins add knowledge-work-plugins/bio-research
```

> **Note:** *All standard Cowork plugins, including Bio-Research, are available from Plugin Settings with a single click. The CLI command above is only needed for Claude Code terminal users.*

Verify installation by running the start command:

```
/start
```

The start command will check which MCP servers are connected and list available skills.

MCP Server Configuration

The plugin includes 10 pre-configured MCP servers in `.mcp.json`:

SERVER	TYPE	AUTHENTICATION	AUTO-CONNECT
pubmed	HTTP	None	Yes
biorender	HTTP	OAuth (first use)	Yes
biorxiv	HTTP	None	Yes
c-trials	HTTP	None	Yes

SERVER	TYPE	AUTHENTICATION	AUTO-CONNECT
chembl	HTTP	None	Yes
synapse	HTTP	OAuth (first use)	Yes
wiley	HTTP	OAuth (first use)	Yes
owkin	HTTP	OAuth (first use)	Yes
ot (Open Targets)	HTTP	None	Yes
benchling	HTTP	*Not yet configured*	No

Note on Benchling: The Benchling MCP server has an empty URL in `.mcp.json`, indicating it's a placeholder for future configuration. Organizations using Benchling should configure their instance-specific URL.

Python Dependencies for Skills

Different skills require different Python packages. Install only what you need:

```
# Single-cell RNA QC
pip install anndata scanpy scipy matplotlib seaborn numpy

# scvi-tools
pip install scvi-tools scanpy anndata

# Nextflow (requires system tools, not Python packages)
# See Nextflow skill documentation

# Clinical Trial Protocol
pip install scipy numpy  # For sample size calculations

# Instrument Data to Allotrope
pip install allotropy pandas openpyxl pdfplumber
```

Optional Binary MCP Servers

Two additional MCP servers are available as separate binary downloads:

1. **10X Genomics txg-mcp** — Access cloud analysis data and workflows
   - Download: https://github.com/10XGenomics/txg-mcp/releases
   - File: `txg-node.mcpb`
2. **ToolUniverse (Harvard MIMS)** — AI tools for scientific discovery

- Download: https://github.com/mims-harvard/ToolUniverse/releases
- File: `tooluniverse.mcpb`

These are optional and not required for core functionality.

---

3. Quick Start Guide

Your First Session

1. Start the plugin:

```
/start
```

2. The start command displays:
   - Connected MCP servers (literature, clinical trials, drug databases)
   - Available skills (QC, scvi-tools, Nextflow, protocols, Allotrope, problem selection)
   - Suggested workflows based on common use cases
3. Choose a workflow or ask a research question directly:
   - "Search PubMed for recent CRISPR papers"
   - "Run QC on my single-cell data at ~/data/experiment.h5ad"
   - "Help me design a clinical trial protocol for a new drug"
   - "Find trials for metastatic breast cancer recruiting in California"

Common Starting Points

RESEARCH GOAL	RECOMMENDED ENTRY POINT
Literature review for a new project	"Search PubMed for [topic]" or "Find recent bioRxiv preprints on [topic]"
Analyze sequencing data	"Run QC on my scRNA-seq data" or "Set up an RNA-seq pipeline"
Drug target research	"Search ChEMBL for compounds targeting [protein]" or "Find drug targets for [disease]"
Clinical trial design	"Help me design a clinical trial for [intervention]" or "Find similar trials for [condition]"
Lab data standardization	"Convert my instrument file to Allotrope format"
Research strategy	"Help me evaluate this project idea" or "I'm stuck on my research"

Understanding the Placeholder System

Plugin files use `~category` notation to refer to tool categories generically:

PLACEHOLDER	REFERS TO	PRE-CONFIGURED TOOLS
`~literature`	Literature databases	PubMed, bioRxiv/medRxiv
`~clinical trials`	Clinical trial databases	ClinicalTrials.gov
`~chemical database`	Compound databases	ChEMBL
`~drug targets`	Target discovery platforms	Open Targets
`~scientific illustration`	Figure creation tools	BioRender
`~journal access`	Full-text article access	Wiley Scholar Gateway
`~data repository`	Research data repositories	Synapse (Sage Bionetworks)
`~AI research`	AI biology platforms	Owkin
`~lab platform`	Lab data management	Benchling (not yet configured)

See `CONNECTORS.md` in the plugin directory for complete mapping and alternative tools.

PART II: MCP SERVERS AND DATA SOURCES

4. Literature and Research Access

The plugin provides access to biomedical literature through multiple complementary databases covering peer-reviewed publications, preprints, and full-text articles.

PubMed (U.S. National Library of Medicine)

Primary source for peer-reviewed biomedical literature. Indexes 35+ million citations from MEDLINE, life science journals, and online books.

Example queries:

- "Search PubMed for CRISPR base editing publications from 2023-2024"
- "Find review articles about CAR-T cell therapy"
- "Search for papers by [author name] on [topic]"

bioRxiv and medRxiv (deepsense.ai MCP server)

Access to preprints (pre-peer-review research) in biology and medicine. Useful for:

- Latest research before formal publication
- Rapid access to COVID-19 and pandemic research
- Emerging techniques and methodologies

Example queries:

- "Find recent bioRxiv preprints on spatial transcriptomics"
- "Search medRxiv for clinical trial preprints on [drug]"

Wiley Scholar Gateway

Full-text access to academic research and publications from John Wiley & Sons journals. Requires OAuth authentication on first use.

Example queries:

- "Access the full text of this Wiley paper [DOI or title]"
- "Find Wiley articles on [topic]"

Synapse (Sage Bionetworks)

Collaborative research data management and repository platform. Useful for:

- Accessing shared datasets from research collaborations
- Storing and versioning research data
- Collaborative projects with data provenance tracking

Requires OAuth authentication on first use.

Workflow: Comprehensive Literature Review

1. Start with broad PubMed search: "Search PubMed for [topic] review articles from last 5 years"
2. Expand to preprints: "Find bioRxiv preprints on [topic] from 2024"
3. Access full text: "Get full text of key papers via Wiley Scholar Gateway"
4. Create visual summary: "Use BioRender to create a figure summarizing the mechanisms"

---

5. Drug Discovery and Clinical Databases

ChEMBL (deepsense.ai MCP server)

Bioactive drug-like compound database from EMBL-EBI. Contains:

- 2.3+ million compounds
- Bioactivity data from 1.9+ million assays
- Target, mechanism, and pathway information

Use cases:

- Find compounds targeting a specific protein
- Search by chemical structure or similarity
- Explore known bioactivities and mechanisms
- Identify existing drugs for repurposing

Example queries:

- "Search ChEMBL for EGFR inhibitors"
- "Find compounds similar to aspirin in ChEMBL"
- "What bioactivity data exists for [compound name]?"

Open Targets (deepsense.ai MCP server)

Drug target discovery and prioritization platform integrating:

- Genetics and genomics data

- Somatic mutations
- RNA expression
- Pathways and interactions
- Animal models
- Known drugs

Use cases:

- Prioritize drug targets for a disease
- Understand target-disease associations
- Explore genetic evidence for targets
- Assess target safety and tractability

Example queries:

- "Find drug targets for Alzheimer's disease"
- "What is the evidence for PCSK9 as a cardiovascular target?"
- "Show me targets with genetic support for [disease]"

ClinicalTrials.gov (deepsense.ai MCP server)

NIH/NLM clinical trial registry with 450,000+ studies worldwide. Comprehensive search and analysis capabilities.

Search dimensions:

- **By condition:** "Find trials for pancreatic cancer"
- **By intervention:** "Find pembrolizumab trials"
- **By sponsor:** "What trials is Novartis running?"
- **By location:** "Find recruiting trials in Boston"
- **By phase:** "Show Phase 3 diabetes trials"
- **By eligibility:** "Trials for 65-year-old female diabetic patient"

Advanced features:

- Detailed trial information (NCT ID lookup)
- Endpoint analysis across therapeutic areas
- Investigator discovery
- Sponsor pipeline analysis

Example queries:

- "Find recruiting Phase 2 trials for metastatic breast cancer"

- "Show me trial details for NCT04267848"
- "What endpoints are commonly used in Alzheimer's trials?"
- "Who are the leading investigators in CAR-T cell trials?"

See Section 10 (Clinical Trial Protocol Generation) for integration with protocol design workflows.

Workflow: Target-to-Clinic Research

1. Identify targets: "Find drug targets for [disease] using Open Targets"
2. Explore compounds: "Search ChEMBL for compounds targeting [top target]"
3. Review clinical progress: "Find clinical trials testing [compound] or [target]"
4. Assess landscape: "Analyze endpoints used in [disease] trials"

---

6. Visualization and Lab Platforms

BioRender (Scientific Illustration)

Create publication-quality scientific figures and diagrams. OAuth authentication required on first use.

Use cases:

- Mechanism of action diagrams
- Experimental workflows
- Cell signaling pathways
- Graphical abstracts
- Presentation slides

Example queries:

- "Create a BioRender figure showing CRISPR mechanism"
- "Generate a schematic of the CAR-T manufacturing process"

Owkin (AI for Biology)

AI-powered platform for histopathology analysis and drug discovery. OAuth authentication required on first use.

Use cases:

- Histopathology image analysis
- Biomarker discovery

- Patient stratification
- Drug response prediction

Benchling (Lab Data Management)

Lab notebook, inventory, and workflow platform. Currently configured as a placeholder (URL not set in `.mcp.json`).

Organizations using Benchling should:

1. Configure the MCP URL in `.mcp.json`
2. Set up OAuth authentication
3. Access lab notebooks, sequences, protocols, and inventory data

Potential use cases:

- Query experimental protocols
- Access sequence libraries
- Retrieve assay data
- Manage sample inventory

PART III: SKILLS DEEP DIVE

7. Single-Cell RNA-seq Quality Control

Skill Overview

Automated quality control for single-cell RNA-seq data following scverse best practices. Implements MAD (Median Absolute Deviation) based filtering with comprehensive visualizations.

When to Use

- Quality control or QC on scRNA-seq data
- Filter low-quality cells from 10X Genomics or similar datasets
- Assess data quality before downstream analysis
- Follow scverse/scanpy best practices
- Identify and remove outliers using statistical methods

Supported Input Formats

FORMAT	SOURCE	NOTES
`.h5ad`	AnnData format (scanpy/Python workflows)	Preferred format
`.h5`	10X Genomics Cell Ranger output	Auto-detected

Quick Start

```
# Default approach (recommended)
python3 scripts/qc_analysis.py experiment.h5ad

# For 10X Genomics files
python3 scripts/qc_analysis.py raw_feature_bc_matrix.h5

# Custom thresholds
python3 scripts/qc_analysis.py experiment.h5ad \
    --mad-counts 4 \
    --mad-genes 4 \
    --mad-mt 2.5 \
    --mt-threshold 10
```

Two Approaches

APPROACH	WHEN TO USE	FLEXIBILITY	COMPLEXITY
Complete Pipeline (`qc_analysis.py`)	Standard QC workflow	Command-line parameters	Low
Modular Building Blocks (`qc_core.py`, `qc_plotting.py`)	Custom workflows, conditional logic	Full Python control	Medium

Approach 1: Complete QC Pipeline (Recommended)

The convenience script `scripts/qc_analysis.py` runs a full QC workflow:

1. Calculate QC metrics (counts, genes, MT%, ribosomal%, hemoglobin%)
2. Apply MAD-based filtering with configurable thresholds
3. Filter low-detection genes
4. Generate comprehensive visualizations
5. Save filtered and annotated datasets

Parameters

PARAMETER	DEFAULT	PURPOSE
`--mad-counts`	5	MAD threshold for total counts (very permissive)
`--mad-genes`	5	MAD threshold for gene detection
`--mad-mt`	3	MAD threshold for mitochondrial %
`--mt-threshold`	8	Hard cutoff for mitochondrial %
`--min-cells`	20	Minimum cells for gene retention
`--mt-pattern`	`mt-,MT-`	Mitochondrial gene prefixes
`--ribo-pattern`	`Rpl,Rps,RPL,RPS`	Ribosomal gene prefixes
`--hb-pattern`	`^Hb[^(p)]|^HB[^(P)]`	Hemoglobin gene pattern

Outputs

All results saved to `<input_basename>_qc_results/` directory:

FILE	CONTENT
`qc_metrics_before_filtering.png`	Pre-filtering distributions and scatter plots
`qc_filtering_thresholds.png`	MAD threshold visualizations with outliers highlighted
`qc_metrics_after_filtering.png`	Post-filtering quality metrics
`<basename>_filtered.h5ad`	Clean dataset ready for downstream analysis
`<basename>_with_qc.h5ad`	Original data with QC annotations preserved

QC Metrics Explained

METRIC	WHAT IT MEASURES	TYPICAL RANGE	RED FLAGS
Total counts	UMI/reads per cell	500-50,000	Bimodal distribution (quality mixing)
Genes detected	Genes with ≥1 count	200-5,000	Very low (<200) indicates failure
Mitochondrial %	Fraction of reads from MT genes	<5% (up to 10-15% for metabolically active cells)	High MT% = dying/stressed cells
Ribosomal %	Fraction from ribosomal genes	Varies	Context-dependent
Hemoglobin %	Fraction from hemoglobin genes	Low for non-blood tissues	High = blood contamination

Why MAD Instead of Fixed Thresholds?

Fixed thresholds (e.g., "remove cells with <500 genes") fail because:

- Different protocols yield different ranges
- Different tissues have different characteristics
- Fixed values are arbitrary and not data-driven

MAD is robust to outliers and adapts to your dataset:

```
MAD = median(|X - median(X)|)
Outlier bounds = median ± n_MADs × MAD
```

Default Thresholds (Deliberately Permissive)

- **5 MADs for counts/genes (log-transformed):** Very permissive to retain rare populations
- **3 MADs for MT%:** More stringent because high MT% strongly indicates dying cells
- **8% hard threshold for MT%:** Additional conservative cutoff

The permissive defaults err on keeping cells because:

1. Rare populations might have extreme but valid values
2. Modern normalization methods handle moderate quality variation
3. Easier to filter more later than recover lost cells

Species-Specific Considerations

SPECIES	MT GENES	RIBOSOMAL GENES	HEMOGLOBIN GENES
Mouse	mt-* (lowercase)	Rpl*, Rps*	Hb* (not Hbp1)
Human	MT-* (uppercase)	RPL*, RPS*	HB* (not HBP1)
Other	Adjust patterns with `--mt-pattern`, `--ribo-pattern`, `--hb-pattern`		

When to Adjust Thresholds

More stringent (lower MADs):

- High ambient RNA contamination
- Many low-quality cells visible in plots
- Quality-driven clustering in downstream analysis

More permissive (higher MADs):

- Studying rare cell populations
- High-quality dataset
- Cell types naturally have extreme values (neurons with high MT%)

Tissue-specific:

- Brain/neurons: Higher MT% threshold (10-15%)
- Blood: More stringent MT% (5-8%)
- Tumor samples: More permissive due to biological variation

Approach 2: Modular Building Blocks

For custom workflows, use functions from `scripts/qc_core.py` and `scripts/qc_plotting.py`:

```
import anndata as ad
from qc_core import (
    calculate_qc_metrics,
    detect_outliers_mad,
    apply_hard_threshold,
    filter_cells,
    filter_genes
)

# Load data
adata = ad.read_h5ad('data.h5ad')

# Calculate metrics
calculate_qc_metrics(adata, inplace=True)

# Custom filtering logic
high_mt = apply_hard_threshold(adata, 'pct_counts_mt', 10, operator='>')
adata_filtered = filter_cells(adata, ~high_mt)

# Filter genes
filter_genes(adata_filtered, min_cells=10, inplace=True)
```

Available Utility Functions

From `qc_core.py`:

- `calculate_qc_metrics()` — Calculate all QC metrics
- `detect_outliers_mad()` — MAD-based outlier detection
- `apply_hard_threshold()` — Apply cutoffs (>, <, >=, <=)
- `filter_cells()` — Apply boolean mask to filter cells
- `filter_genes()` — Filter by detection threshold
- `print_qc_summary()` — Print summary statistics

From `qc_plotting.py`:

- `plot_qc_distributions()` — Comprehensive QC visualizations
- `plot_filtering_thresholds()` — Threshold overlay plots
- `plot_qc_after_filtering()` — Post-filtering plots

Example Custom Workflows

Only metrics and visualization, no filtering:

```
adata = ad.read_h5ad('input.h5ad')
calculate_qc_metrics(adata, inplace=True)
plot_qc_distributions(adata, 'qc_before.png', title='Initial QC')
```

Apply only MT% filtering:

```
adata = ad.read_h5ad('input.h5ad')
calculate_qc_metrics(adata, inplace=True)
high_mt = apply_hard_threshold(adata, 'pct_counts_mt', 10, operator='>')
adata_filtered = filter_cells(adata, ~high_mt)
```

Different thresholds for different cell types:

```
# Assumes cell_type annotation exists
neurons = adata.obs['cell_type'] == 'neuron'
neuron_qc = apply_hard_threshold(adata[neurons], 'pct_counts_mt', 15)
other_qc = apply_hard_threshold(adata[~neurons], 'pct_counts_mt', 8)
```

Next Steps After QC

Typical downstream analysis:

1. Ambient RNA correction (SoupX, CellBender)
2. Doublet detection (scDblFinder, scrublet)
3. Normalization (log-normalize, scran)
4. Feature selection and dimensionality reduction
5. Clustering and cell type annotation

Reference Materials

FILE	CONTENT
`references/scverse_qc_guidelines.md`	Detailed QC methodology, parameter rationale, troubleshooting
`scripts/qc_analysis.py`	Complete pipeline script
`scripts/qc_core.py`	Core QC functions
`scripts/qc_plotting.py`	Visualization functions

Best Practices

1. Always inspect visualizations before/after filtering
2. Be permissive with initial filtering
3. Consider tissue-specific factors (metabolically active cells have higher MT%)
4. Check gene annotation patterns match your species
5. Iterate if needed — QC parameters may need adjustment

---

8. scvi-tools: Deep Learning for Single-Cell Analysis

Skill Overview

Deep learning toolkit for single-cell omics using probabilistic models. Covers batch correction, data integration, multi-modal analysis, label transfer, and RNA velocity.

When to Use

- Batch correction and data integration (scVI, scANVI)
- Multi-modal data analysis (CITE-seq with totalVI, multiome with MultiVI)
- ATAC-seq analysis (PeakVI)
- Spatial transcriptomics deconvolution (DestVI)
- Label transfer and reference mapping (scANVI, scArches)
- RNA velocity (veloVI)
- System-level batch correction (sysVI)

Model Selection Guide

DATA TYPE	MODEL	PRIMARY USE CASE	REFERENCE FILE
scRNA-seq	**scVI**	Unsupervised integration, DE, imputation	`scrna_integration.md`
scRNA-seq + labels	**scANVI**	Label transfer, semi-supervised integration	`label_transfer.md`
CITE-seq (RNA+protein)	**totalVI**	Multi-modal integration, protein denoising	`citeseq_totalvi.md`
scATAC-seq	**PeakVI**	Chromatin accessibility analysis	`atac_peakvi.md`
Multiome (RNA+ATAC)	**MultiVI**	Joint modality analysis	`multiome_multivi.md`

DATA TYPE	MODEL	PRIMARY USE CASE	REFERENCE FILE
Spatial + scRNA reference	**DestVI**	Cell type deconvolution	`spatial_deconvolution.md`
RNA velocity	**veloVI**	Transcriptional dynamics	`rna_velocity_velovi.md`
Cross-technology	**sysVI**	System-level batch correction	`batch_correction_sysvi.md`

Quick Decision Tree

```
Need to integrate scRNA-seq data?
├── Have cell type labels? → scANVI
└── No labels? → scVI

Have multi-modal data?
├── CITE-seq (RNA + protein)? → totalVI
├── Multiome (RNA + ATAC)? → MultiVI
└── scATAC-seq only? → PeakVI

Have spatial data?
└── Need cell type deconvolution? → DestVI

Have pre-trained reference model?
└── Map query to reference? → scArches

Need RNA velocity?
└── veloVI

Strong cross-technology batch effects?
└── sysVI
```

Critical Requirements

1. **Raw counts required:** scvi-tools models require integer count data

```
adata.layers["counts"] = adata.X.copy()  # Before normalization
scvi.model.SCVI.setup_anndata(adata, layer="counts")
```

2. **HVG selection:** Use 2000-4000 highly variable genes

```
sc.pp.highly_variable_genes(
    adata,
    n_top_genes=2000,
    batch_key="batch",
    layer="counts",
    flavor="seurat_v3"
)
adata = adata[:, adata.var['highly_variable']].copy()
```

3. **Batch information**: Specify batch_key for integration

```
scvi.model.SCVI.setup_anndata(adata, layer="counts", batch_key="batch")
```

CLI Scripts

Modular scripts for common workflows:

SCRIPT	PURPOSE	EXAMPLE USAGE
`validate_adata.py`	Check data compatibility	`python scripts/validate_adata.py data.h5ad --batch-key batch`
`prepare_data.py`	QC, filter, HVG selection	`python scripts/prepare_data.py raw.h5ad prepared.h5ad --batch-key batch`
`train_model.py`	Train any scvi-tools model	`python scripts/train_model.py prepared.h5ad results/ --model scvi`
`cluster_embed.py`	Neighbors, UMAP, Leiden	`python scripts/cluster_embed.py adata.h5ad results/`
`differential_expression.py`	DE analysis	`python scripts/differential_expression.py model/ adata.h5ad de.csv`
`transfer_labels.py`	Label transfer with scANVI	`python scripts/transfer_labels.py ref_model/ query.h5ad results/`
`integrate_datasets.py`	Multi-dataset integration	`python scripts/integrate_datasets.py results/ data1.h5ad data2.h5ad`

Example Workflow: Batch Correction with scVI

```
# 1. Validate input data
python scripts/validate_adata.py raw.h5ad --batch-key batch --suggest

# 2. Prepare data (QC, HVG selection)
python scripts/prepare_data.py raw.h5ad prepared.h5ad \
    --batch-key batch --n-hvgs 2000

# 3. Train model
python scripts/train_model.py prepared.h5ad results/ \
    --model scvi --batch-key batch

# 4. Cluster and visualize
python scripts/cluster_embed.py results/adata_trained.h5ad results/ \
    --resolution 0.8

# 5. Differential expression
python scripts/differential_expression.py results/model \
    results/adata_clustered.h5ad results/de.csv --groupby leiden
```

Python Utilities (`scripts/model_utils.py`)

Importable functions for custom workflows:

FUNCTION	PURPOSE
`prepare_adata()`	Data preparation (QC, HVG, layer setup)
`train_scvi()`	Train scVI or scANVI
`evaluate_integration()`	Compute integration metrics
`get_marker_genes()`	Extract DE markers
`save_results()`	Save model, data, plots
`auto_select_model()`	Suggest best model
`quick_clustering()`	Neighbors + UMAP + Leiden

Workflow Reference Files

12 detailed reference documents in `references/`:

FILE	CONTENT	MODELS COVERED
`environment_setup.md`	Installation, GPU, version info	All

FILE	CONTENT	MODELS COVERED
data_preparation.md	Formatting data for any model	All
scrna_integration.md	scVI/scANVI batch correction	scVI, scANVI
atac_peakvi.md	PeakVI for accessibility	PeakVI
citeseq_totalvi.md	totalVI for protein+RNA	totalVI
multiome_multivi.md	MultiVI for RNA+ATAC	MultiVI
spatial_deconvolution.md	DestVI spatial analysis	DestVI
label_transfer.md	scANVI reference mapping	scANVI
scarches_mapping.md	Query-to-reference mapping	scArches
batch_correction_sysvi.md	Advanced batch methods	sysVI
rna_velocity_velovi.md	veloVI dynamics	veloVI
troubleshooting.md	Common issues and solutions	All

Installation and Environment

```
# Install scvi-tools
pip install scvi-tools

# For GPU support (recommended for large datasets)
pip install scvi-tools[cuda]

# Verify installation
python -c "import scvi; print(scvi.__version__)"
```

GPU acceleration significantly improves training time for large datasets (>50,000 cells).

Key Resources

- scvi-tools Documentation
- scvi-tools Tutorials
- Model Hub
- GitHub Issues

Best Practices

1. Always use raw counts, never normalized data

2. Select 2000-4000 HVGs (not full gene set)
3. Specify batch_key for integration tasks
4. Validate data with `validate_adata.py` before training
5. Save models and latent representations for reproducibility
6. Use GPU for datasets >50,000 cells
7. Consult reference files for model-specific guidance

---

9. Nextflow Pipelines for Sequencing Data

Skill Overview

Run nf-core bioinformatics pipelines (rnaseq, sarek, atacseq) on local or public sequencing data. Designed for bench scientists without specialized bioinformatics training.

When to Use

- Analyze RNA-seq, WGS/WES, or ATAC-seq data
- Process local FASTQ files
- Reanalyze public GEO/SRA datasets
- Need differential expression, variant calling, or peak calling
- Want reproducible, containerized workflows

Workflow Checklist

```
□ Step 0: Acquire data (if from GEO/SRA)
□ Step 1: Environment check (MUST pass)
□ Step 2: Select pipeline (confirm with user)
□ Step 3: Run test profile (MUST pass)
□ Step 4: Create samplesheet
□ Step 5: Configure & run (confirm genome)
□ Step 6: Verify outputs
```

Supported Pipelines

DATA TYPE	PIPELINE	VERSION	GOAL	KEY OUTPUTS
RNA-seq	`rnaseq`	3.22.2	Gene expression, differential expression	Gene counts, TPM values
WGS/WES	`sarek`	3.7.1	Germline/somatic variant calling	VCF files, BAM files

DATA TYPE	PIPELINE	VERSION	GOAL	KEY OUTPUTS
ATAC-seq	`atacseq`	2.1.2	Chromatin accessibility	Peak calls, coverage tracks

Step 0: Acquire Data (GEO/SRA)

For public datasets, use the `sra_geo_fetch.py` script:

```
# 1. Get study info
python scripts/sra_geo_fetch.py info GSE110004

# 2. Download (interactive mode to select samples)
python scripts/sra_geo_fetch.py download GSE110004 -o ./fastq -i

# 3. Generate samplesheet
python scripts/sra_geo_fetch.py samplesheet GSE110004 \
    --fastq-dir ./fastq -o samplesheet.csv
```

The interactive download mode is critical for studies with multiple data types (e.g., RNA-seq + ATAC-seq). It prompts you to select which samples to download.

Step 1: Environment Check (REQUIRED)

```
python scripts/check_environment.py
```

All checks must pass. Common fixes:

PROBLEM	FIX
Docker not installed	Install from https://docs.docker.com/get-docker/
Docker permission denied	`sudo usermod -aG docker $USER` then re-login
Docker daemon not running	`sudo systemctl start docker`
Nextflow not installed	`curl -s https://get.nextflow.io | bash && mv nextflow ~/bin/`
Nextflow version < 23.04	`nextflow self-update`
Java not installed / < 11	`sudo apt install openjdk-11-jdk`

Do NOT proceed until all checks pass.

Step 2: Select Pipeline

Auto-detect from data:

```
python scripts/detect_data_type.py /path/to/data
```

Pipeline-specific details:

- `references/pipelines/rnaseq.md` — RNA-seq workflow, aligners, quantification
- `references/pipelines/sarek.md` — Variant calling, germline vs somatic
- `references/pipelines/atacseq.md` — Peak calling, chromatin accessibility

Step 3: Run Test Profile (REQUIRED)

Validates environment with small test data:

```
# RNA-seq
nextflow run nf-core/rnaseq -r 3.22.2 \
    -profile test,docker --outdir test_rnaseq

# Sarek
nextflow run nf-core/sarek -r 3.7.1 \
    -profile test,docker --outdir test_sarek

# ATAC-seq
nextflow run nf-core/atacseq -r 2.1.2 \
    -profile test,docker --outdir test_atacseq
```

Verify completion:

```
ls test_output/multiqc/multiqc_report.html
grep "Pipeline completed successfully" .nextflow.log
```

If test fails, see `references/troubleshooting.md`.

Step 4: Create Samplesheet

Generate automatically:

```
python scripts/generate_samplesheet.py /path/to/data <pipeline> -o samplesheet.csv
```

The script:

- Discovers FASTQ/BAM/CRAM files
- Pairs R1/R2 reads
- Infers sample metadata
- Validates before writing

Samplesheet Formats

rnaseq:

```
sample,fastq_1,fastq_2,strandedness
SAMPLE1,/abs/path/R1.fq.gz,/abs/path/R2.fq.gz,auto
```

sarek:

```
patient,sample,lane,fastq_1,fastq_2,status
patient1,tumor,L001,/abs/path/tumor_R1.fq.gz,/abs/path/tumor_R2.fq.gz,1
patient1,normal,L001,/abs/path/normal_R1.fq.gz,/abs/path/normal_R2.fq.gz,0
```

atacseq:

```
sample,fastq_1,fastq_2,replicate
CONTROL,/abs/path/ctrl_R1.fq.gz,/abs/path/ctrl_R2.fq.gz,1
```

Step 5: Configure & Run

Check genome availability:

```
python scripts/manage_genomes.py check GRCh38
# If not installed:
python scripts/manage_genomes.py download GRCh38
```

Common genomes: GRCh38 (human), GRCh37 (legacy), GRCm39 (mouse), R64-1-1 (yeast), BDGP6 (fly)

Pipeline-Specific Options

Confirm with user:

- **rnaseq:** aligner (star_salmon recommended, hisat2 for low memory)
- **sarek:** tools (haplotypecaller for germline, mutect2 for somatic)
- **atacseq:** read_length (50, 75, 100, or 150)

Run Command

```
nextflow run nf-core/<pipeline> \
    -r <version> \
    -profile docker \
    --input samplesheet.csv \
    --outdir results \
    --genome <genome> \
    -resume
```

Key flags:

- `-r` : Pin version
- `-profile docker` : Use Docker containers
- `--genome` : iGenomes reference key
- `-resume` : Continue from checkpoint if interrupted

Resource limits (if needed):

```
--max_cpus 8 --max_memory '32.GB' --max_time '24.h'
```

Step 6: Verify Outputs

Check completion:

```
ls results/multiqc/multiqc_report.html
grep "Pipeline completed successfully" .nextflow.log
```

Key Outputs by Pipeline

rnaseq:

- `results/star_salmon/salmon.merged.gene_counts.tsv` — Gene counts
- `results/star_salmon/salmon.merged.gene_tpm.tsv` — TPM values
- `results/multiqc/multiqc_report.html` — QC summary

sarek:

- `results/variant_calling/*/` — VCF files
- `results/preprocessing/recalibrated/` — BAM files
- `results/multiqc/multiqc_report.html` — QC summary

atacseq:

- `results/macs2/narrowPeak/` — Peak calls
- `results/bwa/mergedLibrary/bigwig/` — Coverage tracks
- `results/multiqc/multiqc_report.html` — QC summary

Resume Failed Run

```
nextflow run nf-core/<pipeline> -resume
```

Nextflow checkpoints intermediate results, so re-running continues from where it stopped.

Reference Materials

FILE	CONTENT
`references/geo-sra-acquisition.md`	Downloading public GEO/SRA data
`references/troubleshooting.md`	Common issues and fixes
`references/installation.md`	Environment setup
`references/pipelines/rnaseq.md`	RNA-seq pipeline details
`references/pipelines/sarek.md`	Variant calling details
`references/pipelines/atacseq.md`	ATAC-seq details

Scripts Inventory

SCRIPT	PURPOSE
`check_environment.py`	Validate Docker, Nextflow, Java
`detect_data_type.py`	Auto-detect pipeline from data
`generate_samplesheet.py`	Create samplesheet from directory
`manage_genomes.py`	Check/download reference genomes
`sra_geo_fetch.py`	Download GEO/SRA data
`utils/file_discovery.py`	FASTQ file discovery
`utils/ncbi_utils.py`	NCBI API utilities
`utils/sample_inference.py`	Sample metadata inference

SCRIPT	PURPOSE
`utils/validators.py`	Samplesheet validation

Disclaimer

This skill is a prototype demonstrating integration of nf-core pipelines for automated workflows. Users are responsible for validating computational analyses and ensuring environment compatibility. Not officially endorsed by nf-core.

Attribution

When publishing, cite the appropriate pipeline. Citations available in each nf-core repository's CITATIONS.md file.

Licenses

- nf-core pipelines: MIT License
- Nextflow: Apache License, Version 2.0
- NCBI SRA Toolkit: Public Domain

---

10. Clinical Trial Protocol Generation

Skill Overview

Generate FDA/NIH-compliant clinical trial protocols for medical devices or drugs using a modular, waypoint-based architecture.

When to Use

- "Create a clinical trial protocol"
- "Generate protocol for [device/drug]"
- "Help me design a clinical study"
- "Research similar trials for [intervention]"
- Developing FDA submission documentation

Two Modes

MODE	STEPS	DELIVERABLE	USE CASE
Research Only	0-1	Comprehensive research summary	Background research before protocol

MODE	STEPS	DELIVERABLE	USE CASE
Full Protocol	0-5	Complete protocol document	Full protocol generation

Architecture: Waypoint-Based Design

All data stored in `waypoints/` directory as JSON/markdown files:

```
waypoints/
├── intervention_metadata.json          # Intervention info, status
├── 01_clinical_research_summary.json  # Similar trials, FDA guidance
├── 02_protocol_foundation.md          # Sections 1-6
├── 03_protocol_intervention.md        # Sections 7-8
├── 04_protocol_operations.md          # Sections 9-12
├── 02_protocol_draft.md               # Complete protocol (concatenated)
├── 02_protocol_metadata.json          # Protocol metadata
└── 02_sample_size_calculation.json    # Statistical calculations
```

Modular Subskill Steps

STEP	FILE	CONTENT
0	`00-initialize-intervention.md`	Collect device/drug information
1	`01-research-protocols.md`	Clinical trials research, FDA guidance
2	`02-protocol-foundation.md`	Sections 1-6 (foundation, design, population)
3	`03-protocol-intervention.md`	Sections 7-8 (intervention details)
4	`04-protocol-operations.md`	Sections 9-12 (assessments, statistics, operations)
5	`05-concatenate-protocol.md`	Final document generation

Execution Flow

1. Display welcome message with mode selection
2. Wait for user to choose:
   - Research Only (steps 0-1)
   - Full Protocol (steps 0-5)
   - Exit
3. Execute steps sequentially (on-demand loading of subskills)
4. After step 4, pause for review before final concatenation

Research Only Mode Output

Formatted markdown artifact with:

- Intervention overview
- Similar clinical trials (top 5-10 with NCT IDs)
- FDA regulatory pathway recommendation
- FDA guidance documents
- Study design recommendations
- Sample size considerations
- Key insights and next steps

Full Protocol Mode Output

Complete protocol document following NIH/FDA template:

1. Protocol Overview
2. Background and Rationale
3. Study Objectives
4. Study Design
5. Study Population
6. Intervention/Treatment
7. Study Procedures
8. Safety Monitoring
9. Data Analysis
10. Study Operations
11. Regulatory Considerations
12. References

Prerequisites

1. **Clinical trials MCP server (Required)**
   - Available tools: `search_clinical_trials`, `get_trial_details`
   - Auto-detects in Step 1
2. **FDA Database Access (Built-in)**
   - 510(k), PMA, De Novo pathways for devices
   - Drugs@FDA, Orange Book, Purple Book for drugs
   - Uses direct FDA URLs (no web search)

3. **Clinical Protocol Template**
   - Template files in `assets/` directory
   - FDA-Clinical-Protocol-Template.md
4. **Python Dependencies (for sample size calculations)**

```
pip install scipy numpy
```

Sample Size Calculator

`scripts/sample_size_calculator.py` provides validated statistical calculations for:

- Two-sample t-tests
- Proportions (two-sample)
- Survival analysis (log-rank test)
- Equivalence/non-inferiority trials

Example usage (via protocol skill, not direct invocation):

```
from sample_size_calculator import calculate_sample_size

result = calculate_sample_size(
    test_type='two_sample_ttest',
    alpha=0.05,
    power=0.80,
    effect_size=0.5
)
```

Waypoint Resume Capability

If interrupted, the skill automatically:

1. Detects completed steps from `intervention_metadata.json`
2. Resumes from next incomplete step
3. Preserves all prior work

Rich Initial Context Support

Users can provide extensive documentation when initializing (Step 0):

- Technical specifications
- Research data
- Background materials

- Existing protocols

This context is preserved in `intervention_metadata.json` under `initial_context` field and referenced in later steps.

Data Minimization Strategy

Each step implements aggressive summarization:

- Keep: Top-N results (5-10 max), key facts, concise rationale
- Discard: Raw MCP query results, full FDA guidance text, lower-ranked results

This keeps waypoint files manageable (<15KB JSON, <50KB markdown).

Disclaimers

⚠️ **IMPORTANT:** This tool provides preliminary protocol based on NIH/FDA guidelines. It does NOT constitute:

- Official FDA or IRB determination or approval
- Medical, legal, or regulatory advice
- Substitute for professional biostatistician review
- Substitute for FDA Pre-Submission meeting
- Guarantee of regulatory or clinical success

Required before proceeding:

- Biostatistician review and sample size validation
- FDA Pre-Submission meeting (Q-Submission for devices, Pre-IND for drugs)
- IRB review and approval
- Clinical expert and regulatory consultant engagement
- Legal review of protocol and informed consent
- Completion of all [TBD] items in protocol

Reference Materials

FILE	CONTENT
`references/00-initialize-intervention.md`	Step 0 instructions
`references/01-research-protocols.md`	Step 1 research workflow
`references/02-protocol-foundation.md`	Step 2 sections 1-6
`references/03-protocol-intervention.md`	Step 3 sections 7-8

FILE	CONTENT
`references/04-protocol-operations.md`	Step 4 sections 9-12
`references/05-concatenate-protocol.md`	Step 5 final assembly
`assets/FDA-Clinical-Protocol-Template.md`	Protocol template
`scripts/sample_size_calculator.py`	Statistical calculations

Best Practices

1. Start with Research Only mode to validate approach before full protocol
2. Provide as much initial context as possible in Step 0
3. Review each waypoint file after generation
4. Pause after Step 4 to review protocol draft before final concatenation
5. Have biostatistician review sample size calculations
6. Plan FDA Pre-Submission meeting before finalizing protocol

---

11. Instrument Data to Allotrope Conversion

Skill Overview

Convert laboratory instrument output files (PDF, CSV, Excel, TXT) to Allotrope Simple Model (ASM) JSON format or flattened 2D CSV. Supports 40+ instrument types including cell counters, spectrophotometers, plate readers, qPCR, and chromatography systems.

When to Use

- Standardize instrument data for LIMS systems
- Prepare data for upload to data lakes
- Generate parser code for data engineers
- Convert lab data to machine-readable format
- Ensure data interoperability across platforms

Note: This is an example skill demonstrating data engineering task automation. Customize for your organization by modifying reference files, connecting to schema registries via MCP, or extending scripts for proprietary formats.

Workflow Overview

1. Detect instrument type (auto-detect or user-specified)

2. Parse file using allotropy library or flexible fallback
3. Generate outputs:
   - ASM JSON (full semantic structure)
   - Flattened CSV (2D tabular format)
   - Python parser code (for handoff to data engineers)
4. Validate and deliver

Output Format Selection

FORMAT	BEST FOR	CHARACTERISTICS
ASM JSON	LIMS systems, data lakes, archival	Full semantic structure, ontology URIs, validates against Allotrope schemas
Flattened CSV	Quick analysis, Excel, systems without JSON support	2D table, one measurement per row, metadata repeated
Both	Maximum flexibility	Generate both for different downstream uses

Supported Instruments (see `references/supported_instruments.md` for complete list)

CATEGORY	INSTRUMENTS
Cell Counting	Vi-CELL BLU, Vi-CELL XR, NucleoCounter NC-200, Countess II
Spectrophotometry	NanoDrop One/Eight/8000, Lunatic, SpectraMax
Plate Readers	SoftMax Pro, EnVision, Gen5, CLARIOstar, Synergy, Infinite
ELISA	SoftMax Pro, BMG MARS, MSD Workbench, Victor Nivo
qPCR	QuantStudio, Bio-Rad CFX, LightCycler
Chromatography	Empower, Chromeleon, ChemStation
Electrophoresis	TapeStation, Fragment Analyzer, Bioanalyzer

Detection and Parsing Strategy

Tier 1: Native allotropy parsing (PREFERRED)

Always try allotropy first. Check available vendors:

```
from allotropy.parser_factory import Vendor

# List all supported vendors
for v in Vendor:
    print(f"{v.name}")

# Common vendors:
# AGILENT_TAPESTATION_ANALYSIS
# BECKMAN_VI_CELL_BLU
# THERMO_FISHER_NANODROP_EIGHT
# MOLDEV_SOFTMAX_PRO
# APPBIO_QUANTSTUDIO
```

Tier 2: Flexible fallback parsing

Only if allotropy doesn't support the instrument. This fallback:

- Does NOT generate calculated-data-aggregate-document
- Does NOT include full traceability
- Produces simplified ASM structure

Tier 3: PDF extraction

For PDF-only files, extract tables using pdfplumber, then apply Tier 2 parsing.

Critical: Calculated Data Handling

Separate raw measurements from calculated/derived values:

- **Raw data** → `measurement-document` (direct instrument readings)
- **Calculated data** → `calculated-data-aggregate-document` (derived values)

Calculated values MUST include traceability via `data-source-aggregate-document`:

```
"calculated-data-aggregate-document": {
  "calculated-data-document": [{
    "calculated-data-identifier": "SAMPLE_B1_DIN_001",
    "calculated-data-name": "DNA integrity number",
    "calculated-result": {"value": 9.5, "unit": "(unitless)"},
    "data-source-aggregate-document": {
      "data-source-document": [{
        "data-source-identifier": "SAMPLE_B1_MEASUREMENT",
        "data-source-feature": "electrophoresis trace"
      }]
    }
  }]
}
```

Common Calculated Fields by Instrument

INSTRUMENT	CALCULATED FIELDS
Cell counter	Viability %, cell density dilution-adjusted values
Spectrophotometer	Concentration (from absorbance), 260/280 ratio
Plate reader	Concentrations from standard curve, %CV
Electrophoresis	DIN/RIN, region concentrations, average sizes
qPCR	Relative quantities, fold change, ΔΔCt

See `references/field_classification_guide.md` for detailed guidance on classifying fields as raw vs. calculated.

Validation

Always validate ASM output:

```
python scripts/validate_asm.py output.json
python scripts/validate_asm.py output.json --reference known_good.json
python scripts/validate_asm.py output.json --strict  # Warnings as errors
```

Validation Rules:

- Based on Allotrope ASM specification (December 2024)
- Last updated: 2026-01-07
- Source: https://gitlab.com/allotrope-public/asm

Soft Validation Approach:

Unknown techniques, units, or sample roles generate **warnings** (not errors) for forward compatibility. Use `--strict` mode for stricter validation.

What validation checks:

- Correct technique selection
- Field naming conventions (space-separated, not hyphenated)
- Calculated data has traceability
- Unique identifiers for measurements
- Required metadata present
- Valid units and sample roles

Pre-Parsing Checklist

Before writing a custom parser, ALWAYS:

1. Check if allotropy supports it — use native parser if available
2. Find a reference ASM file — check examples or ask user
3. Review instrument-specific guide — check reference materials
4. Validate against reference — run `validate_asm.py --reference <file>`

Common Mistakes to Avoid

MISTAKE	CORRECT APPROACH
Manifest as object	Use URL string
Lowercase detection types	Use "Absorbance" not "absorbance"
"emission wavelength setting"	Use "detector wavelength setting" for emission
All measurements in one document	Group by well/sample location
Missing procedure metadata	Extract ALL device settings per measurement

Code Export for Data Engineers

Generate standalone Python scripts:

```
python scripts/export_parser.py \
    --input "data.csv" \
    --vendor "VI_CELL_BLU" \
    --output "parser_script.py"
```

The exported script:

- Has no external dependencies beyond pandas/allotropy
- Includes inline documentation
- Can run in Jupyter notebooks
- Is production-ready for data pipelines

Scripts Inventory

SCRIPT	PURPOSE
`convert_to_asm.py`	Main conversion script
`flatten_asm.py`	ASM → 2D CSV conversion
`export_parser.py`	Generate standalone parser code
`validate_asm.py`	Validate ASM output quality

Reference Materials

FILE	CONTENT
`references/supported_instruments.md`	Full instrument list with Vendor enums
`references/asm_schema_overview.md`	ASM structure reference
`references/field_classification_guide.md`	Where to put different field types
`references/flattening_guide.md`	How flattening works

Example Use Cases

Example 1: Vi-CELL BLU file

```
User: "Convert this cell counting data to Allotrope format"
[uploads viCell_Results.xlsx]

Process:
1. Detect Vi-CELL BLU (95% confidence)
2. Convert using allotropy native parser
3. Output:
   - viCell_Results_asm.json (full ASM)
   - viCell_Results_flat.csv (2D format)
   - viCell_parser.py (exportable code)
```

Example 2: LIMS-ready flattened output

```
User: "Convert this ELISA data to a CSV I can upload to our LIMS"

Process:
1. Parse plate reader data
2. Generate flattened CSV with columns:
   - sample_identifier, well_position, measurement_value, measurement_unit
   - instrument_serial_number, analysis_datetime, assay_type
3. Validate against LIMS import requirements
```

Installation

```
pip install allotropy pandas openpyxl pdfplumber
```

Best Practices

1. Always validate output before delivery
2. Use native allotropy parsers when available
3. Clearly separate raw from calculated data
4. Include full traceability for calculated values
5. When uncertain about field classification, ask user for clarification
6. Reference `field_classification_guide.md` for guidance

---

12. Scientific Problem Selection Framework

Skill Overview

Systematic framework for research problem selection based on Fischbach & Walsh's "Problem choice and decision trees in science and engineering" (Cell, 2024). Conversational approach for ideation, troubleshooting, and strategic decisions.

When to Use

- Pitch a new research idea
- Troubleshoot a stuck project
- Evaluate project risks
- Plan research strategy
- Navigate decision trees
- Choose what scientific problem to work on

Three Entry Points

1. **Pitch an idea for a new project** — work it up together
2. **Share a problem in a current project** — troubleshoot together
3. **Ask a strategic question** — navigate the decision tree together

This conversational entry meets scientists where they are.

Core Framework Concepts

The Central Insight: Problem Choice >> Execution Quality

Even brilliant execution of a mediocre problem yields incremental impact. Good execution of an important problem yields substantial impact.

The Time Paradox: Scientists typically spend:

- **Days** choosing a problem
- **Years** solving it

This imbalance limits impact. These skills help invest more time choosing wisely.

Evaluation Axes:

- **X-axis:** Likelihood of success
- **Y-axis:** Impact if successful

Skills help move ideas rightward (more feasible) and upward (more impactful).

The 9 Skills Overview

SKILL	PURPOSE	OUTPUT	TIME
1. Intuition Pumps	Generate high-quality research ideas	Problem Ideation Document	~1 week
2. Risk Assessment	Identify and manage project risks	Risk Assessment Matrix	3-5 days
3. Optimization Function	Define success metrics	Impact Assessment Document	2-3 days
4. Parameter Strategy	Decide what to fix vs. keep flexible	Parameter Strategy Document	2-3 days
5. Decision Tree Navigation	Plan decision points and altitude dance	Decision Tree Map	2 days
6. Adversity Response	Prepare for crises as opportunities	Adversity Playbook	2 days
7. Problem Inversion	Navigate around obstacles	Problem Inversion Analysis	1 day
8. Integration & Synthesis	Synthesize into coherent plan	Project Communication Package	3-5 days
9. Meta-Framework	Orchestrate complete workflow	Complete Project Package	1-6 weeks

Skill Workflow

```
SKILL 1: Intuition Pumps
         ↓ (generates idea)
SKILL 2: Risk Assessment
         ↓ (evaluates feasibility)
SKILL 3: Optimization Function
         ↓ (defines success metrics)
SKILL 4: Parameter Strategy
         ↓ (determines flexibility)
SKILL 5: Decision Tree
         ↓ (plans execution and evaluation)
SKILL 6: Adversity Planning
         ↓ (prepares for failure modes)
SKILL 7: Problem Inversion
         ↓ (provides pivot strategies)
SKILL 8: Integration & Communication
         ↓ (synthesizes into coherent plan)
SKILL 9: Meta-Skill
         (orchestrates complete workflow)
```

Entry Point 1: Pitch an Idea

Initial prompt: "Tell me the short version of your idea (1-2 sentences)."

Response approach: Return quick summary demonstrating understanding. Note research area and rephrase the idea highlighting its kernel.

Follow-up prompt: "Now give me a bit more detail:

1. What exactly you want to do
2. How you currently plan to do it
3. If it works, why will it be a big deal
4. What you think are the major risks"

Workflow: Guide through early stages:

- Skill 1: Intuition Pumps — Refine and strengthen the idea
- Skill 2: Risk Assessment — Identify and manage project risks
- Skill 3: Optimization Function — Define success metrics
- Skill 4: Parameter Strategy — Determine what to fix vs. keep flexible

Entry Point 2: Troubleshoot a Problem

Initial prompt: "Tell me a short version of your problem (1-2 sentences)."

Response approach: Return quick summary demonstrating understanding. Note project context and rephrase problem highlighting core essence.

Follow-up prompt: "Now give me a bit more detail:

1. The overall goal of your project
2. What exactly went wrong
3. Your current ideas for fixing it"

Workflow: Guide through troubleshooting:

- Skill 5: Decision Tree Navigation — Plan decision points
- Skill 4: Parameter Strategy — Fix one parameter at a time
- Skill 6: Adversity Response — Frame problems as opportunities
- Skill 7: Problem Inversion — Strategies for navigating obstacles

Entry Point 3: Ask a Strategic Question

Initial prompt: "Tell me the short version of your question (1-2 sentences)."

Response approach: Return quick summary demonstrating understanding. Note broader context and rephrase question.

Follow-up prompt: "Now give me a bit more detail:

1. The setting (current or future project)
2. A bit more detail about what you're thinking"

Workflow: Draw on appropriate modules:

- Skills 1-4 for future project planning
- Skills 5-7 for current project navigation
- Skill 8 for communication and synthesis
- Skill 9 for comprehensive workflow orchestration

Key Framework Principles

The Risk Paradox:

- Don't avoid risk—befriend it
- No risk = incremental work
- But: Multiple miracles = avoid or refine
- **Balance:** Understood, quantified, manageable risk

The Parameter Paradox:

- Too many fixed = brittleness
- Too few fixed = paralysis

- **Sweet spot:** Fix ONE meaningful constraint

The Adversity Principle:

- Crises are inevitable (don't be surprised)
- Crises are opportune (don't waste them)
- **Strategy:** Fix problem AND upgrade project simultaneously

Reference Materials (9 detailed guides in `references/`)

FILE	CONTENT	SEARCH PATTERNS
`01-intuition-pumps.md`	Generate research ideas	Intuition Pump #, Trap #, Phase
`02-risk-assessment.md`	Risk identification	Risk 1-5, go/no-go, assumption
`03-optimization-function.md`	Success metrics	Generality, Learning, optimization, impact
`04-parameter-strategy.md`	Parameter fixation	fixed, float, constraint, parameter
`05-decision-tree.md`	Decision tree navigation	altitude, Level, decision
`06-adversity-planning.md`	Adversity response	adversity, crisis, ensemble
`07-problem-inversion.md`	Problem inversion strategies	Strategy, inversion, goal
`08-integration-synthesis.md`	Integration and synthesis	narrative, communication, story
`09-meta-framework.md`	Complete workflow	Phase, workflow, orchestrate

Who Should Use These Skills

USER	WHEN	FOCUS	TIMELINE
Graduate Students	Thesis projects, qualifying exams	Skills 1-3 (ideation, risk, impact) + Skill 9	2-4 weeks
Postdocs	New position, independent projects, fellowships	All skills, emphasizing independence and risk	1-2 weeks
Principal Investigators	New lab, new direction, mentoring, grants	Skills 1, 3, 4, 6 (ideation, impact, parameters, adversity)	Ongoing

USER	WHEN	FOCUS	TIMELINE
Startup Founders	Company inception, pivots, investor pitches	Skills 1-4 + Skill 8 (communication)	1-2 weeks, revisit quarterly

Expected Outcomes

Immediate (after completing workflow):

- Clear project vision
- Honest risk assessment
- Contingency plans
- Communication materials ready
- Confidence in problem choice

6-Month:

- Faster decisions (have framework)
- Productive adversity handling
- No existential crises (risks mitigated)

2-Year:

- Published results or strong progress
- Avoided dead-end projects
- Career aligned with goals
- **Time well-spent** (ultimate measure)

Foundational Reference

Fischbach, M.A., & Walsh, C.T. (2024). "Problem choice and decision trees in science and engineering." *Cell*, 187, 1828-1833.

Based on course BIOE 395 taught at Stanford University.

Design Principles

1. **Conversational Entry** — Meet users where they are
2. **Thoughtful Interaction** — Ask clarifying questions
3. **Literature Integration** — Use PubMed at strategic points
4. **Concrete Outputs** — Every skill produces tangible documents
5. **Building Specificity** — Progressive detail through targeted questions

6. **Flexibility** — Skills work independently, sequentially, or iteratively
7. **Scientific Rigor** — Evidence-based claims

Best Practices

1. Start with short version, then add detail
2. Use literature search to validate generality and feasibility
3. Be honest about risks and assumptions
4. Fix one meaningful parameter, let others float
5. Plan for crises as opportunities
6. Document decisions and rationale
7. Iterate based on new information

PART IV: WORKFLOW PATTERNS

13. Common Research Workflows

The bio-research plugin supports six major research workflow patterns combining multiple skills and MCP servers.

Workflow 1: Literature Review and Synthesis

Goal: Comprehensively review literature on a topic and synthesize findings.

Steps:

1. Search PubMed for peer-reviewed articles
2. Search bioRxiv for recent preprints
3. Access full-text via Wiley Scholar Gateway
4. Synthesize findings into summary
5. Create visual summary with BioRender

Commands:

```
"Search PubMed for CRISPR base editing reviews from 2023-2024"
"Find bioRxiv preprints on base editing from 2024"
"Access full text of [key paper DOI]"
"Create BioRender figure showing base editing mechanisms"
```

Typical timeline: 2-3 days

Workflow 2: Single-Cell RNA-seq Analysis

Goal: Complete single-cell RNA-seq analysis from raw data to biological insights.

Steps:

1. QC with single-cell-rna-qc skill
2. Batch correction and integration with scvi-tools
3. Clustering and visualization
4. Differential expression analysis

5. Cell type annotation

Commands:

```
# Step 1: QC
python scripts/qc_analysis.py experiment.h5ad

# Step 2-3: Integration and clustering
python scripts/prepare_data.py experiment_filtered.h5ad prepared.h5ad --batch-key batc
python scripts/train_model.py prepared.h5ad results/ --model scvi
python scripts/cluster_embed.py results/adata_trained.h5ad results/

# Step 4: Differential expression
python scripts/differential_expression.py results/model results/adata_clustered.h5ad d
```

Typical timeline: 1-2 days for computation, 1-2 weeks for biological interpretation

---

Workflow 3: Public Dataset Reanalysis

Goal: Reanalyze published GEO/SRA dataset with modern pipelines.

Steps:

1. Search literature to identify dataset (PubMed)
2. Download data from GEO/SRA (Nextflow skill)
3. Run nf-core pipeline (RNA-seq, ATAC-seq, or variant calling)
4. Verify outputs and interpret results

Commands:

```
# Step 1: Find the dataset
"Search PubMed for [original paper]"

# Step 2: Download
python scripts/sra_geo_fetch.py info GSE110004
python scripts/sra_geo_fetch.py download GSE110004 -o ./fastq -i

# Step 3: Run pipeline
python scripts/check_environment.py
nextflow run nf-core/rnaseq -r 3.22.2 -profile test,docker --outdir test
nextflow run nf-core/rnaseq -r 3.22.2 -profile docker --input samplesheet.csv --outdir
```

Typical timeline: 1 day download, 1-3 days computation

Workflow 4: Drug Target Discovery

Goal: Identify and prioritize drug targets for a disease.

Steps:

1. Identify targets with Open Targets
2. Find compounds with ChEMBL
3. Review clinical trials (ClinicalTrials.gov)
4. Research publications (PubMed)
5. Synthesize target assessment

Commands:

```
"Find drug targets for Alzheimer's disease using Open Targets"
"What is the genetic evidence for BACE1 as an Alzheimer's target?"
"Search ChEMBL for BACE1 inhibitors"
"Find clinical trials testing BACE1 inhibitors"
"Search PubMed for BACE1 inhibitor clinical trial results"
```

Typical timeline: 2-3 days

Workflow 5: Clinical Trial Protocol Development

Goal: Generate complete FDA/NIH-compliant clinical trial protocol.

Steps:

1. Research mode: similar trials and FDA guidance
2. Protocol foundation (sections 1-6)
3. Protocol intervention (sections 7-8)
4. Protocol operations (sections 9-12)
5. Final document generation

Commands:

```
/start  # Invoke clinical-trial-protocol skill
[Select "Full Protocol" mode]
[Provide intervention information]
[Review waypoint files after each step]
[Generate final protocol]
```

Typical timeline: 3-5 days (research + writing)

Integration with MCP servers:

- ClinicalTrials.gov: Find similar trials, analyze endpoints
- PubMed: Research efficacy and safety data
- FDA databases: Identify regulatory pathway and guidance documents

---

Workflow 6: Lab Data Standardization for LIMS

Goal: Convert instrument data to standardized format for LIMS upload.

Steps:

1. Identify instrument type
2. Convert to Allotrope ASM format
3. Generate flattened CSV for LIMS
4. Validate output
5. (Optional) Export parser code for data engineers

Commands:

```
"Convert this Vi-CELL file to Allotrope format"
[Skill detects instrument, converts, generates both JSON and CSV]
python scripts/validate_asm.py output_asm.json
```

Typical timeline: Minutes to hours depending on file size

---

14. Data Analysis Pipelines

Pipeline Pattern 1: Sequencing Data End-to-End

```
Raw FASTQ files
    ↓
[Nextflow skill] Environment check
    ↓
[Nextflow skill] Run nf-core pipeline (rnaseq/sarek/atacseq)
    ↓
QC reports + aligned data
    ↓
Downstream analysis (differential expression, variant calling, peak calling)
```

Pipeline Pattern 2: Single-Cell Integration

```
Multiple scRNA-seq datasets
    ↓
[Single-cell QC skill] QC each dataset independently
    ↓
[scvi-tools skill] Prepare data (HVG selection)
    ↓
[scvi-tools skill] Train scVI model for integration
    ↓
[scvi-tools skill] Clustering and visualization
    ↓
Integrated, batch-corrected dataset
```

Pipeline Pattern 3: Multi-Modal Single-Cell

```
CITE-seq data (RNA + protein)
    ↓
[Single-cell QC skill] QC RNA modality
    ↓
[scvi-tools skill] totalVI for multi-modal integration
    ↓
Integrated RNA + protein analysis
```

---

15. Integration Strategies

Strategy 1: Progressive Disclosure

Start with high-level analysis, drill down as needed:

1. **Week 1:** Literature review (PubMed, bioRxiv) → identify key questions
2. **Week 2:** Preliminary analysis (QC, basic stats) → validate approach

3. **Week 3:** Deep analysis (integration, DE, advanced models)
4. **Week 4:** Synthesis and visualization

Strategy 2: Parallel Workflows

Run complementary analyses simultaneously:

- **Track A:** Wet lab (experiments) → Instrument data conversion → LIMS upload
- **Track B:** Computational (sequencing) → Nextflow pipelines → scvi-tools analysis
- **Track C:** Literature and strategy → PubMed search → Scientific problem selection

Strategy 3: Iterative Refinement

Use framework for systematic iteration:

1. **Ideation** (Problem selection skill) → Define hypothesis
2. **Pilot** (Small-scale analysis) → Test feasibility
3. **Risk assessment** (Problem selection skill) → Evaluate obstacles
4. **Full analysis** (Skills + MCP servers) → Execute
5. **Adversity response** (Problem selection skill) → Navigate challenges
6. **Synthesis** (Problem selection skill) → Communicate results

---

PART V: ADVANCED USAGE

16. Customization and Extension

Approach A: Direct Modification

Edit plugin files directly for quick experiments or personal customizations.

Example: Add tissue-specific QC thresholds

Edit `skills/single-cell-rna-qc/SKILL.md`:

```
## Tissue-Specific QC Thresholds

### Brain/Neurons
- MT% threshold: 10-15% (neurons have higher MT content)
- Genes detected: 1000-7000 (neurons express more genes)

### Blood
- MT% threshold: 5-8% (stricter for blood cells)
- Genes detected: 500-3000

### Tumor
- More permissive thresholds due to biological variation
- Review histograms carefully for bimodal distributions
```

Approach B: Fork the Plugin

Copy entire plugin, rename, modify for organization-wide use.

Example: Create pharma-specific bio-research plugin

1. Copy `bio-research/1.0.0/` to `bio-research-pharma/1.0.0/`
2. Update `plugin.json`:

```
{
  "name": "bio-research-pharma",
  "version": "1.0.0",
  "description": "Bio-research plugin adapted for pharmaceutical R&D"
}
```

3. Add pharma-specific skills:

- GMP compliance for lab data
- Regulatory endpoints for clinical trials
- Drug development milestone tracking

4. Modify clinical trial protocol skill for IND/NDA submissions

Approach C: Complementary Skills via CLAUDE.md

Add project-specific knowledge without touching plugin.

Example: Add CRO-specific protocols

Create `.claude/skills/cro-protocols/SKILL.md`:

```
---
name: cro-protocols
description: CRO-specific protocols and SOPs for sequencing services
---

# CRO Protocols

## Sample Submission Requirements

[CRO-specific requirements]

## Data Delivery Formats

[Expected output formats]

## QC Standards

[CRO's QC thresholds and acceptance criteria]
```

Add to `CLAUDE.md`:

```
## CRO Integration

When analyzing data from [CRO name], always apply cro-protocols skill
for QC thresholds and data format expectations.
```

---

17. Troubleshooting and Best Practices

Common Issues and Solutions

ISSUE	SKILL	SOLUTION
MCP server not connecting	All	Check `.mcp.json` configuration, re-authenticate OAuth servers
Python package missing	scvi-tools, QC, Clinical trials	Install requirements: `pip install -r requirements.txt`
Docker permission denied	Nextflow	`sudo usermod -aG docker $USER`, then re-login
Out of memory	scvi-tools, Nextflow	Reduce dataset size, use GPU for scvi-tools, adjust Nextflow resource limits
MAD thresholds too strict	Single-cell QC	Increase MAD values (5→7 for counts/genes, 3→5 for MT%)
scvi-tools requires counts	scvi-tools	Ensure `adata.layers["counts"]` contains raw counts before normalization
Nextflow pipeline fails	Nextflow	Check test profile first, review `.nextflow.log`, see troubleshooting.md
Allotrope validation fails	Instrument conversion	Check field naming, ensure calculated data has traceability
Waypoint file missing	Clinical trials	Resume from last completed step, waypoints auto-save after each step

Best Practices

For Single-Cell Analysis:

1. Always run QC before downstream analysis
2. Use raw counts for scvi-tools, never normalized data
3. Select 2000-4000 HVGs for scvi-tools models
4. Validate data with `validate_adata.py` before training
5. Save models and latent representations for reproducibility

For Nextflow Pipelines:

1. Always run environment check first (`check_environment.py`)
2. Run test profile before real data
3. Use `-resume` flag to continue interrupted runs
4. Pin pipeline version with `-r` flag
5. Check MultiQC report for quality issues

For Clinical Trials:

1. Start with Research Only mode before full protocol
2. Provide detailed initial context in Step 0
3. Review each waypoint file after generation
4. Have biostatistician review sample size calculations
5. Plan FDA Pre-Submission meeting before finalizing

For Data Conversion:

1. Always try native allotropy parser first
2. Validate ASM output before delivery
3. Clearly separate raw from calculated data
4. Include full traceability for calculated values
5. When uncertain, ask user for field classification

For Problem Selection:

1. Start with short version, then add detail
2. Use literature search to validate claims
3. Be honest about risks and assumptions
4. Fix one meaningful parameter
5. Plan for crises as opportunities

PART VI: REFERENCE MATERIALS

18. Complete File Inventory

Plugin Structure: 88 Total Files

Root Level (5 files)

```
.claude-plugin/plugin.json    # Plugin metadata
.mcp.json                      # MCP server connections (10 servers)
CONNECTORS.md                  # Placeholder mappings
README.md                      # Plugin documentation
commands/start.md              # Start command
```

Single-Cell RNA QC Skill (5 files)

```
skills/single-cell-rna-qc/
├── SKILL.md                   # Skill definition and workflow
├── LICENSE.txt
├── references/
│   └── scverse_qc_guidelines.md
└── scripts/
    ├── qc_analysis.py         # Complete pipeline
    ├── qc_core.py             # Core QC functions
    └── qc_plotting.py         # Visualization functions
```

scvi-tools Skill (21 files)

```
skills/scvi-tools/
├── SKILL.md                    # Skill definition and model selection
├── LICENSE.txt
├── references/ (12 files)
│   ├── atac_peakvi.md
│   ├── batch_correction_sysvi.md
│   ├── citeseq_totalvi.md
│   ├── data_preparation.md
│   ├── environment_setup.md
│   ├── label_transfer.md
│   ├── multiome_multivi.md
│   ├── rna_velocity_velovi.md
│   ├── scarches_mapping.md
│   ├── scrna_integration.md
│   ├── spatial_deconvolution.md
│   └── troubleshooting.md
└── scripts/ (8 files)
    ├── cluster_embed.py
    ├── differential_expression.py
    ├── integrate_datasets.py
    ├── model_utils.py
    ├── prepare_data.py
    ├── train_model.py
    ├── transfer_labels.py
    └── validate_adata.py
```

Nextflow Skill (18 files)

```
skills/nextflow-development/
├── SKILL.md                    # Skill definition and workflow checklist
├── LICENSE.txt
├── references/ (7 files)
│   ├── geo-sra-acquisition.md
│   ├── installation.md
│   ├── troubleshooting.md
│   └── pipelines/
│       ├── atacseq.md
│       ├── rnaseq.md
│       └── sarek.md
└── scripts/ (10 files)
    ├── check_environment.py
    ├── detect_data_type.py
    ├── generate_samplesheet.py
    ├── manage_genomes.py
    ├── sra_geo_fetch.py
    └── utils/
        ├── __init__.py
        ├── file_discovery.py
        ├── ncbi_utils.py
        ├── sample_inference.py
        └── validators.py
```

Clinical Trial Protocol Skill (9 files)

```
skills/clinical-trial-protocol/
├── SKILL.md                    # Orchestrator and workflow
├── LICENSE.txt
├── references/ (6 files)
│   ├── 00-initialize-intervention.md
│   ├── 01-research-protocols.md
│   ├── 02-protocol-foundation.md
│   ├── 03-protocol-intervention.md
│   ├── 04-protocol-operations.md
│   └── 05-concatenate-protocol.md
├── assets/
│   └── FDA-Clinical-Protocol-Template.md
└── scripts/
    └── sample_size_calculator.py
```

Instrument Data to Allotrope Skill (11 files)

```
skills/instrument-data-to-allotrope/
├── SKILL.md                    # Skill definition and conversion workflow
├── LICENSE.txt
├── requirements.txt
├── references/ (4 files)
│   ├── asm_schema_overview.md
│   ├── field_classification_guide.md
│   ├── flattening_guide.md
│   └── supported_instruments.md
└── scripts/ (4 files)
    ├── convert_to_asm.py
    ├── export_parser.py
    ├── flatten_asm.py
    └── validate_asm.py
```

Scientific Problem Selection Skill (12 files)

```
skills/scientific-problem-selection/
├── SKILL.md                    # Entry points and framework overview
├── LICENSE.txt
└── references/ (9 files)
    ├── 01-intuition-pumps.md
    ├── 02-risk-assessment.md
    ├── 03-optimization-function.md
    ├── 04-parameter-strategy.md
    ├── 05-decision-tree.md
    ├── 06-adversity-planning.md
    ├── 07-problem-inversion.md
    ├── 08-integration-synthesis.md
    └── 09-meta-framework.md
```

File Count Summary

COMPONENT	FILES
Root/Plugin Config	5
Single-Cell RNA QC	5
scvi-tools	21
Nextflow	18
Clinical Trial Protocol	9
Instrument to Allotrope	11

COMPONENT	FILES
Scientific Problem Selection	12
Total	**88**

---

19. Command Reference

Start Command (`/start`)

Location: `commands/start.md`

Purpose: Initialize bio-research environment and display available tools.

Execution:

1. Display welcome message
2. Check connected MCP servers (tests connectivity)
3. List available skills
4. Mention optional binary MCP servers
5. Ask user what they're working on
6. Suggest workflow based on common use cases

Typical Output:

```
Bio-Research Plugin

Your AI-powered research assistant for the life sciences...

Literature & Data Sources:
✓ PubMed (biomedical literature)
✓ bioRxiv/medRxiv (preprints)
...

Skills:
• Single-Cell RNA QC
• scvi-tools
• Nextflow Pipelines
• Clinical Trial Protocol
• Instrument Data Converter
• Scientific Problem Selection

What are you working on today?
```

PART VII: APPENDICES

20. Optional Binary MCP Servers

Two additional MCP servers available as separate binary downloads:

10X Genomics txg-mcp

Purpose: Access 10X Genomics cloud analysis data and workflows

Installation:

1. Download `txg-node.mcpb` from https://github.com/10XGenomics/txg-mcp/releases
2. Install via drag-and-drop into Claude Desktop or manual configuration

Capabilities:

- Access cloud analysis results
- Query Cell Ranger outputs
- Retrieve processed datasets
- Integration with 10X platform

Category placeholder: `~genomics platform`

ToolUniverse (Harvard MIMS)

Purpose: AI tools for scientific discovery from Harvard Medical School

Installation:

1. Download `tooluniverse.mcpb` from https://github.com/mims-harvard/ToolUniverse/releases
2. Install via drag-and-drop into Claude Desktop or manual configuration

Capabilities:

- Access curated AI tool database
- Find tools for specific scientific tasks
- Explore method recommendations

Category placeholder: `~tool database`

---

21. Resource Links and Citations

Plugin Resources

- Plugin repository: (anthropics/knowledge-work-plugins)
- Issue tracker: (GitHub repository issues)
- Documentation: This handbook

MCP Server Documentation

SERVICE	DOCUMENTATION
PubMed	https://pubmed.ncbi.nlm.nih.gov/
bioRxiv	https://www.biorxiv.org/
BioRender	https://www.biorender.com/
ChEMBL	https://www.ebi.ac.uk/chembl/
Open Targets	https://www.opentargets.org/
ClinicalTrials.gov	https://clinicaltrials.gov/
Synapse	https://www.synapse.org/
Wiley	https://onlinelibrary.wiley.com/
Owkin	https://owkin.com/

Scientific Framework Citations

scverse (Single-Cell Analysis)

- Luecken, M.D., & Theis, F.J. (2019). Current best practices in single-cell RNA-seq analysis: a tutorial. *Molecular Systems Biology*, 15(6), e8746.
- https://www.sc-best-practices.org/

scvi-tools

- Gayoso, A., et al. (2022). A Python library for probabilistic analysis of single-cell omics data. *Nature Biotechnology*, 40, 163-166.
- https://docs.scvi-tools.org/

nf-core

- Ewels, P.A., et al. (2020). The nf-core framework for community-curated bioinformatics pipelines. *Nature Biotechnology*, 38, 276-278.

- https://nf-co.re/

Allotrope

- Allotrope Foundation ASM specification (December 2024)
- https://gitlab.com/allotrope-public/asm

Problem Selection Framework

- Fischbach, M.A., & Walsh, C.T. (2024). Problem choice and decision trees in science and engineering. *Cell*, 187, 1828-1833.

nf-core Pipeline Citations

When publishing results, cite the appropriate pipeline:

- **rnaseq:** See https://github.com/nf-core/rnaseq/blob/3.22.2/CITATIONS.md
- **sarek:** See https://github.com/nf-core/sarek/blob/3.7.1/CITATIONS.md
- **atacseq:** See https://github.com/nf-core/atacseq/blob/2.1.2/CITATIONS.md

Tool Licenses

- scvi-tools: BSD 3-Clause License
- nf-core pipelines: MIT License
- Nextflow: Apache License, Version 2.0
- Allotropy: Apache License, Version 2.0
- NCBI SRA Toolkit: Public Domain

Conclusion

The bio-research plugin represents the most comprehensive integration of preclinical research tools available in the Cowork ecosystem. With 88 files spanning literature search, genomics analysis, clinical trial design, data standardization, and research strategy, it provides life sciences researchers with a complete toolkit for modern research workflows.

Key Takeaways

1. **Start Simple:** Use the `/start` command and choose a workflow that matches your immediate needs
2. **Build Skills:** Master one skill at a time (start with literature search or QC)
3. **Combine Tools:** The real power comes from chaining skills and MCP servers together
4. **Customize:** Adapt the plugin to your organization's specific needs

5. **Iterate:** Use the problem selection framework to make strategic decisions

Getting Help

- Consult reference files within each skill for detailed methodology
- Use troubleshooting guides for common issues
- Review this handbook for workflow patterns and best practices
- Check MCP server documentation for specific tool capabilities

Next Steps

1. Install the plugin: `/install anthropics/knowledge-work-plugins bio-research`
2. Run the start command: `/start`
3. Choose your first workflow from Section 13
4. Explore skills as you encounter relevant research questions
5. Customize for your organization using patterns from Section 16

The bio-research plugin is designed to grow with you—from simple literature searches to complex multi-modal analyses, from quick QC checks to full clinical trial protocols. Start where you are, and expand as your research demands.

Document Information

- **Version:** 1.0.0
- **Word Count:** ~8,100 words
- **Files Analyzed:** 88 (complete plugin inventory)
- **Date Prepared:** February 2026
- **Prepared By:** Agent analysis of bio-research plugin v1.0.0

END OF HANDBOOK

RationalEyes.ai

contact@rationaleyes.ai

Intelligent Automation for Knowledge Work

Bio-Research Plugin Handbook — Version 1.0.0