Vantage6 Algorithms - Complete Playbook

Version: 1.0.0
Last Updated: 2025-11-20
Status: ✅ Production Ready

Table of Contents

Overview

This playbook provides complete instructions for using the MIDN federated imputation algorithms with the vantage6 framework. The algorithms enable privacy-preserving missing data imputation across multiple healthcare institutions without sharing raw patient data.

What's Included

- **SIMI**: Single Imputation for Missing Data (one column at a time)
- **SIMICE**: Single Imputation for Multiple Columns (multiple columns simultaneously)
- **Docker Containers**: Pre-built, ready-to-deploy algorithm containers
- **Test Suite**: Comprehensive testing framework
- **Documentation**: Complete guides and examples

Key Features

✅ **Privacy-Preserving**: Only aggregated statistics shared, never raw data
✅ **Federated Learning**: Multiple institutions collaborate without data sharing
✅ **Dockerized**: Easy deployment and scaling
✅ **Production Ready**: Fully tested and documented
✅ **Flexible**: Supports both continuous and binary data

Quick Start

5-Minute Setup

```
# 1. Clone/navigate to the project
cd vantage6_algorithms

# 2. Build containers
./build.sh

# 3. Verify containers
docker images | grep -E "(simi|simice)-algorithm"

# 4. Test locally
python test_local.py

# 5. Ready to deploy!
```

That's it! Your algorithms are ready for vantage6 deployment.

Prerequisites

Required

- **Docker** 20.10+ (Install Docker)
- **Python** 3.11+ (for local testing)
- **Vantage6 Server** (for production deployment)
- **Git** (to clone repository)

Optional (for local testing)

- **Python packages**: numpy, pandas, scipy
- **Jupyter Notebook** (for interactive exploration)

System Requirements

- **RAM**: Minimum 4GB (8GB recommended)
- **Disk Space**: 2GB for containers
- **Network**: Internet access for pulling base images

Installation & Setup

Step 1: Verify Prerequisites

```
# Check Docker
docker --version
# Should show: Docker version 20.10 or higher

# Check Python
python3 --version
# Should show: Python 3.11 or higher

# Check if vantage6 is available (optional)
pip list | grep vantage6 || echo "vantage6 not installed (OK for local testing)"
```

Step 2: Build Docker Containers

```
cd vantage6_algorithms

# Build both algorithms
./build.sh

# Expected output:
# ✓ SIMI build successful
# ✓ SIMICE build successful
```

Build Time: ~5-10 minutes (first time), ~1 minute (subsequent builds with cache)

Step 3: Verify Installation

```
# List built images
docker images | grep -E "(simi|simice)-algorithm"

# Test SIMI container
docker run --rm simi-algorithm:latest python -c "from algorithm import master_simi; print('✓ SIM

# Test SIMICE container
docker run --rm simice-algorithm:latest python -c "from algorithm import master_simice; print('✓
```

Step 4: Run Local Tests

```
# Run comprehensive test suite
python test_local.py

# Expected: All tests pass
# ✓ PASSED: Remote Functions
# ✓ PASSED: SIMI Gaussian
# ✓ PASSED: SIMI Logistic
# ✓ PASSED: SIMICE
```

Algorithm Details

SIMI (Single Imputation)

Purpose: Impute missing values in a single target column using federated learning.

When to Use

- ✅ Missing values in one column
- ✅ Need multiple imputation datasets
- ✅ Continuous or binary target variable
- ✅ Privacy-preserving collaboration required

Parameters

Parameter	Type	Required	Description	Example
`target_column_index`	int	Yes	1-based index of column to impute	`2` (second column)

Parameter	Type	Required	Description	Example
`is_binary`	bool	Yes	`True` if target is binary (0/1), `False` for continuous	`False`
`imputation_trials`	int	Yes	Number of imputed datasets to generate	`10`

How It Works

1. **Central node** identifies missing values in target column
2. **Remote nodes** compute local statistics (Gaussian or Logistic regression)
3. **Central node** aggregates statistics from all nodes
4. **Central node** fits regression model on aggregated data
5. **Central node** generates multiple imputed datasets using predictive distribution

Mathematical Foundation

- **Gaussian Method**: Uses linear regression with normal error distribution
- **Logistic Method**: Uses logistic regression for binary outcomes
- **Privacy**: Only aggregated statistics (XX, Xy, yy) shared, never raw data

SIMICE (Single Imputation Multiple Columns)

Purpose: Impute missing values in multiple columns simultaneously using iterative federated learning.

When to Use

- ✅ Missing values in multiple columns
- ✅ Columns may be correlated
- ✅ Need iterative refinement
- ✅ Privacy-preserving collaboration required

Parameters

Parameter	Type	Required	Description	Example
`target_column_indexes`	list[int]	Yes	1-based indices of columns to impute	`[2, 4, 5]`
`is_binary_list`	list[bool]	Yes	Binary flags for each column	`[False, True, False]`

Parameter	Type	Required	Description	Example
			(same length)	
`imputation_trials`	int	Yes	Number of imputed datasets to generate	`5`
`iteration_before_first_imputation`	int	Yes	Iterations before first imputation	`10`
`iteration_between_imputations`	int	Yes	Iterations between imputations	`5`

How It Works

1. **Initialization**: Initialize missing values with column means
2. **Iterative Process**:
   - For each target column:
     - Remote nodes compute statistics
     - Central aggregates statistics
     - Update regression coefficients
     - Update imputed values
3. **Multiple Imputation**: Generate multiple datasets at specified intervals
4. **Convergence**: Iterate until convergence or max iterations

Mathematical Foundation

- **Iterative Regression**: Alternates between fitting models and imputing values
- **Multiple Imputation**: Generates multiple complete datasets for uncertainty quantification
- **Privacy**: Only aggregated statistics shared at each iteration

Testing

Test Suite Overview

The project includes comprehensive test suites:

1. **test_local.py**: Unit tests with mock vantage6 client

2. **test_comprehensive.py**: Extended test suite with edge cases
3. **test_with_real_data.py**: Tests with actual sample data (if available)

Running Tests

```
# Basic unit tests
python test_local.py

# Comprehensive tests
python test_comprehensive.py

# With real data (if sample files exist)
python test_with_real_data.py
```

Test Coverage

✅ Functionality Tests

- Algorithm execution
- Statistics aggregation
- Missing value imputation
- Binary data handling
- Multiple imputation trials

✅ Data Handling Tests

- Numpy array input
- Missing value handling
- Binary data constraints
- Data serialization

✅ Edge Cases

- Small datasets
- All missing values
- No missing values
- Single/multiple columns

✅ Integration Tests

- Mock vantage6 client
- RPC function calls
- Multi-node aggregation
- End-to-end workflows

Test Data

Synthetic Test Data

The test suite generates synthetic data with:

- Configurable sample sizes
- Configurable missing rates
- Both continuous and binary variables
- Realistic correlations

Real Sample Data (Optional)

If available in `MIDN_R_PY/samples/`:

- `SIMI_Cent_Cont.csv`: SIMI central continuous data
- `SIMI_Remote_Cont_1.csv`: SIMI remote node 1 data
- `SIMICE_Cent_Cont.csv`: SIMICE central continuous data
- `SIMICE_Remote_bin_1.csv`: SIMICE remote binary data

Expected Test Results

```
✓ PASSED: Remote Functions
✓ PASSED: SIMI Gaussian
✓ PASSED: SIMI Logistic
✓ PASSED: SIMICE

Total: 4/4 tests passed
🎉 All tests passed! Algorithms are ready for vantage6 deployment.
```

Deployment

Step 1: Push to Container Registry (Optional)

If using a remote registry:

```
# Tag images
docker tag simi-algorithm:latest your-registry.com/simi-algorithm:v1.0.0
docker tag simice-algorithm:latest your-registry.com/simice-algorithm:v1.0.0

# Push images
docker push your-registry.com/simi-algorithm:v1.0.0
docker push your-registry.com/simice-algorithm:v1.0.0
```

Step 2: Register with Vantage6 Server

```
from vantage6.client import UserClient

# Connect to vantage6 server
client = UserClient(
    server_url="https://your-vantage6-server.com",
    api_key="your-api-key"
)

# Get collaboration ID
collaborations = client.collaboration.list()
collab_id = collaborations[0]['id']  # Use your collaboration ID

# Register SIMI
simi_algorithm = client.algorithm.create(
    name="simi",
    image="simi-algorithm:latest",  # or registry path
    description="Single Imputation for Missing Data",
    version="1.0.0",
    collaboration_id=collab_id
)

# Register SIMICE
simice_algorithm = client.algorithm.create(
    name="simice",
    image="simice-algorithm:latest",  # or registry path
    description="Single Imputation for Multiple Columns",
    version="1.0.0",
    collaboration_id=collab_id
)
```

Step 3: Verify Registration

```
# List registered algorithms
algorithms = client.algorithm.list()
for alg in algorithms:
    print(f"{alg['name']}: {alg['image']}")
```

Usage Examples

Example 1: SIMI - Continuous Variable

Scenario: Impute missing values in a continuous variable (e.g., blood pressure) across 3 hospitals.

```
from vantage6.client import UserClient

client = UserClient("https://your-server.com", "api-key")

# Create task
task = client.task.create(
    name="Blood Pressure Imputation",
    image="simi-algorithm:latest",
    input_={
        'target_column_index': 3,      # Blood pressure column
        'is_binary': False,             # Continuous variable
        'imputation_trials': 10         # Generate 10 imputed datasets
    },
    organizations=[hospital1_id, hospital2_id, hospital3_id],
    database="patient_data"
)

# Monitor task
import time
while True:
    result = client.task.get(task['id'])
    if result['status'] in ['completed', 'failed']:
        break
    time.sleep(5)

# Get results
if result['status'] == 'completed':
    output = result['result']
    imputed_datasets = output['imputed_datasets']
    print(f"Generated {len(imputed_datasets)} imputed datasets")
```

Example 2: SIMI - Binary Variable

Scenario: Impute missing values in a binary outcome (e.g., disease present/absent).

```
task = client.task.create(
    name="Disease Status Imputation",
    image="simi-algorithm:latest",
    input_={
        'target_column_index': 5,      # Disease status column
        'is_binary': True,              # Binary variable
        'imputation_trials': 5          # Generate 5 imputed datasets
    },
    organizations=[org1, org2],
    database="clinical_data"
)
```

Example 3: SIMICE - Multiple Columns

Scenario: Impute missing values in multiple correlated variables simultaneously.

```
task = client.task.create(
    name="Multi-Column Imputation",
    image="simice-algorithm:latest",
    input_={
        'target_column_indexes': [2, 4, 6],           # Three columns
        'is_binary_list': [False, True, False],        # Mixed types
        'imputation_trials': 5,
        'iteration_before_first_imputation': 10,       # 10 iterations before first imputation
        'iteration_between_imputations': 5             # 5 iterations between imputations
    },
    organizations=[org1, org2, org3],
    database="research_data"
)
```

Example 4: Using Results

```
# Get task results
result = client.task.get(task_id)
output = result['result']

# Access imputed datasets
imputed_datasets = output['imputed_datasets']

# Each dataset is a list of lists (can be converted to numpy array)
import numpy as np
first_dataset = np.array(imputed_datasets[0])

# Use datasets for downstream analysis
# Example: Pool results across multiple imputations
mean_values = np.mean([np.array(ds) for ds in imputed_datasets], axis=0)
```

Troubleshooting

Common Issues

Issue 1: Container Build Fails

Symptoms: `docker build` fails with file not found errors

Solutions:

```
# Ensure you're in the correct directory
cd vantage6_algorithms

# Check that Core/ directory exists
ls -la Core/

# Verify build script permissions
chmod +x build.sh

# Try building manually
docker build --build-arg ALGORITHM=SIMI -t simi-algorithm:latest -f Dockerfile .
```

Issue 2: Import Errors in Container

Symptoms: `ModuleNotFoundError` when running container

Solutions:

```
# Check installed packages
docker run --rm simi-algorithm:latest pip list

# Verify requirements.txt
cat SIMI/requirements.txt

# Rebuild container
docker build --no-cache --build-arg ALGORITHM=SIMI -t simi-algorithm:latest -f Dockerfile .
```

Issue 3: Task Fails with "Method Not Found"

Symptoms: Vantage6 task fails with method routing errors

Solutions:

- Verify `wrapper.py` correctly routes method names
- Check that method names match between master and RPC calls
- Review task logs: `client.task.get(task_id).get('log')`

Issue 4: No Results Returned

Symptoms: Task completes but no results

Solutions:

```
# Check task status
result = client.task.get(task_id)
print(f"Status: {result['status']}")
print(f"Result: {result.get('result')}")

# Check logs
logs = result.get('log', [])
for log_entry in logs:
    print(log_entry)
```

Issue 5: Data Access Errors

Symptoms: Cannot access data at remote nodes

Solutions:

- Verify database name matches at all nodes
- Check node permissions
- Ensure data is accessible via vantage6's data interface

- Review node configuration

Debug Mode

Enable verbose logging:

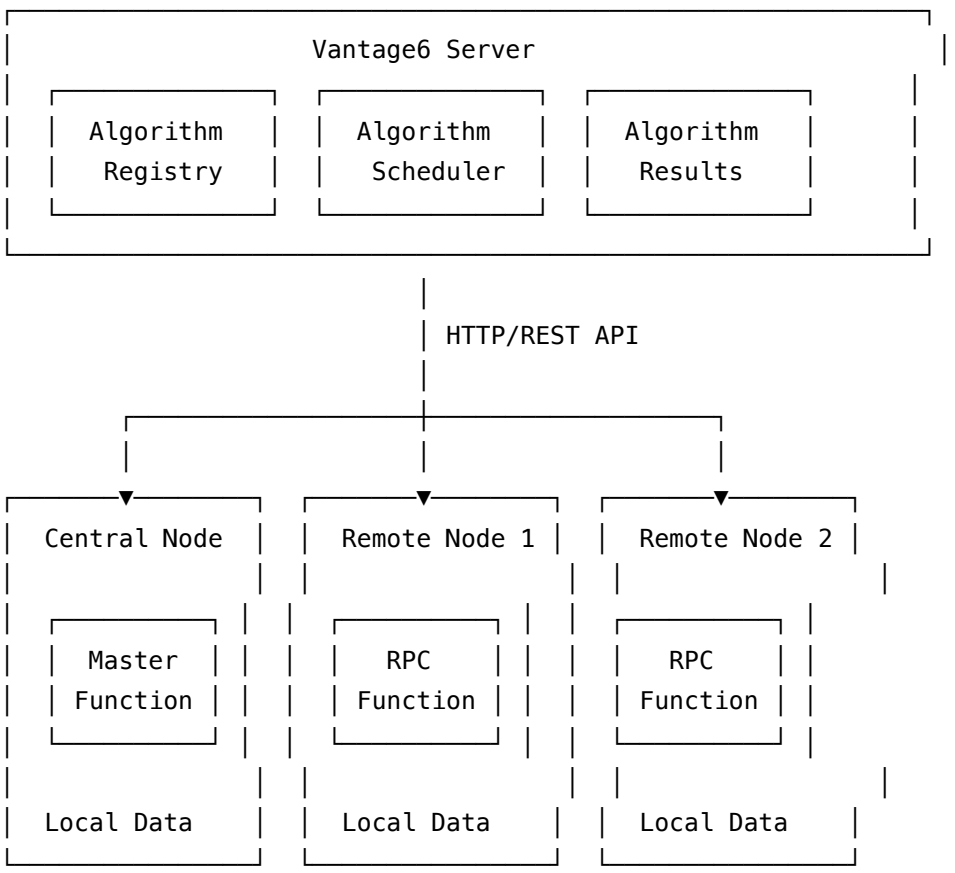
```
import logging
logging.basicConfig(level=logging.DEBUG)

# Create task with debug
task = client.task.create(
    ...,
    input_={..., 'debug': True}  # If supported
)
```

Getting Help

1. **Check Logs**: Always review task logs first
2. **Test Locally**: Use `test_local.py` to verify algorithms work
3. **Verify Setup**: Run `./build.sh` and verify containers
4. **Review Documentation**: Check `INTEGRATION_GUIDE.md` for details

Architecture

System Architecture



Algorithm Flow

1. User creates task via vantage6 API
   ↓
2. Vantage6 launches containers at each node
   ↓
3. Central node executes master_*() function
   ↓
4. Master calls remote nodes via client.create_new_task()
   ↓
5. Remote nodes execute RPC_*() functions on local data
   ↓
6. Remote nodes return aggregated statistics (not raw data)
   ↓
7. Master aggregates statistics from all nodes
   ↓
8. Master fits regression model
   ↓
9. Master generates imputed datasets
   ↓
10. Results returned via vantage6 API

Data Privacy

What is Shared:

- ✅ Aggregated statistics (XX, Xy, yy matrices)
- ✅ Regression coefficients (after aggregation)
- ✅ Imputed values (only at central node)

What is NOT Shared:

- ❌ Raw patient data
- ❌ Individual records
- ❌ Node-specific data distributions

Container Structure

```
/app/
├── algorithm.py      # Main algorithm logic
├── wrapper.py        # Vantage6 entry point
├── Core/             # Shared utilities
│   ├── LS.py         # Least squares
│   ├── Logit.py      # Logistic regression
│   └── ...
└── requirements.txt  # Dependencies
```

File Structure

```
vantage6_algorithms/
├── Dockerfile                    # Unified Dockerfile (use this!)
├── build.sh                      # Build script
├── PLAYBOOK.md                   # This file - complete guide
├── README.md                     # Overview
├── INTEGRATION_GUIDE.md          # Integration details
├── QUICK_INTEGRATION.md          # Quick 3-step guide
├── Core/                         # Shared utilities
│   ├── LS.py
│   ├── Logit.py
│   ├── transfer.py
│   └── ...
├── SIMI/                         # SIMI algorithm
│   ├── algorithm.py              # Main algorithm code
│   ├── wrapper.py                # Vantage6 entry point
│   └── requirements.txt          # Dependencies
├── SIMICE/                       # SIMICE algorithm
│   ├── algorithm.py
│   ├── wrapper.py
│   └── requirements.txt
├── test_local.py                 # Local test suite
├── test_comprehensive.py         # Comprehensive tests
└── test_with_real_data.py        # Real data tests
```

Best Practices

1. Testing

- ✅ Always test locally before deploying
- ✅ Use `test_local.py` to verify functionality
- ✅ Test with sample data similar to production

2. Deployment

- ✅ Tag images with version numbers
- ✅ Use container registry for production
- ✅ Monitor first few tasks closely
- ✅ Keep logs for debugging

3. Data Preparation

- ✅ Ensure consistent column indices across nodes
- ✅ Handle missing values appropriately
- ✅ Verify data types match (continuous vs binary)
- ✅ Check data quality before running

4. Performance

- ✅ Start with small `imputation_trials` for testing
- ✅ Monitor task execution time
- ✅ Adjust `iteration_before_first_imputation` based on convergence
- ✅ Consider data size when setting parameters

5. Security

- ✅ Use secure container registry
- ✅ Verify vantage6 server certificates
- ✅ Review data access permissions
- ✅ Monitor for unusual activity

Version History

- **v1.0.0** (2025-11-20): Initial release
  - SIMI and SIMICE algorithms
  - Unified Dockerfile

    - Comprehensive test suite
    - Complete documentation

Support & Resources

Documentation

- **This Playbook**: Complete usage guide
- **INTEGRATION_GUIDE.md**: Detailed integration steps
- **QUICK_INTEGRATION.md**: Quick start guide
- **README.md**: Overview and structure

External Resources

- Vantage6 Documentation
- Algorithm Development Guide
- Docker Documentation

Getting Help

1. Review this playbook
2. Check troubleshooting section
3. Review test results
4. Check vantage6 server logs
5. Contact support team

License

[Add your license information here]

Acknowledgments

- Original MIDN algorithms from MIDN_R_PY
- Vantage6 framework for federated learning infrastructure
- Contributors and testers

Last Updated: 2025-11-20

Status: ✅ Production Ready

Version: 1.0.0