DOE Pipeline Comprehensive Documentation

Last Updated: 2024 **Pipeline Version:** 2.0 (Modified to load pre-split fan dataframes) **Status:** Production Ready

---

Quick Start

```bash
# Navigate to workspace
cd /Users/vblake/doe2

# Activate virtual environment (if not already active)
source venv/bin/activate

# Run full pipeline
python doep.py
```

Output Location: `./outputs/` directory

Table of Contents

Pipeline Overview

The DOE (Design of Experiments) Pipeline automates a comprehensive statistical analysis of interface temperature data across different fan speed ranges, transceiver manufacturers, and rack units.

```
INPUT (CSV Files)
        ↓
   [STEP 1-3: Data Loading & Cleaning]
        ↓
   [STEP 4-6: Fan Speed Balancing]
        ↓
   [STEP 7-9: DOE Design & Modeling]
```

```
         ↓
    [STEP 10-13: Report Generation]
         ↓
    [STEP 14-16: Visualization & Output]
         ↓
OUTPUT (HTML, PDF, PowerPoint)
```

Modified Pipeline (Current Version)

This version loads pre-split fan dataframes directly:

```
fan_low_df.csv ──→ Data Cleaning ──→ Balance ──→ DOE Analysis ──→ Re
fan_high_df.csv ─┘
```

16-Step Pipeline Process

Step 1: Load Fan Speed Dataframes from CSV

Input: `outputs/fan_low_df.csv`, `outputs/fan_high_df.csv`
Output: `fan_low_df`, `fan_high_df` (pandas DataFrames)

```
# Direct load from pre-split dataframes
fan_low_df = pd.read_csv('outputs/fan_low_df.csv')
fan_high_df = pd.read_csv('outputs/fan_high_df.csv')
```

Module: `importcsv.py` → `load_csv_data()` (adapted for fan dataframes)

Description: Loads pre-split fan speed data directly from CSV files. This optimized approach bypasses the original load→split pipeline, starting analysis with already-separated low and high fan speed datasets.

Step 2: Show Data Summary (Low Speed Fans)

Input: `fan_low_df` (pandas DataFrame)
Output: Console summary of data structure

```
# Display head/tail and statistics
importcsv.show_data_summary(fan_low_df)
```

Module: `importcsv.py` → `show_data_summary()`

Description: Displays data structure including row count, column names, first/last rows, and data types for low speed fan dataset. Validates data integrity before processing.

Output Example:

```
DATA SUMMARY
=====================================================================
Total rows imported: 12,469
Total columns: 47
Column names:
  1. row_id
  2. SFP_manufacturer
  3. device_id
  4. rack_unit
  5. Interface_Temp
  ... (42 more columns)
```

Step 3: Remove Missing Data (Low Speed)

Input: `fan_low_df` (pandas DataFrame)
Output: `fan_low_df` with NaN rows removed

```
# Remove rows with any missing values
fan_low_df = importcsv.remove_missing_data(fan_low_df)
```

Module: `importcsv.py` → `remove_missing_data()`

Description: Removes all rows containing NaN (missing) values. Reports before/after row counts and number of rows removed. Ensures data completeness for statistical analysis.

Step 4: Clean Transceiver Manufacturer (Low Speed)

Input: `fan_low_df` (pandas DataFrame)
Output: `fan_low_df` with standardized manufacturer names

```
# Analyze and clean SFP manufacturer categories
fan_low_df = clean.clean_tman(fan_low_df)
```

Module: `clean.py` → `clean_tman()`

Description: Standardizes transceiver manufacturer names into 10 primary categories. Groups less common manufacturers into "Others" category. Ensures consistent categorical levels for DOE analysis.

Step 5: Create Fan Speed Histogram (Low Speed)

Input: `fan_low_df` (pandas DataFrame)
Output: `outputs/fan_speed_histogram_low.png`

```
# Generate histogram visualization of fan speeds
prep.create_fan_speed_histogram(fan_low_df, 'Low Speed Fans', 'outpu
```

Module: `prep.py` → `create_fan_speed_histogram()`

Description: Creates interactive histogram of fan speed distribution for low-speed dataset. Includes statistics (mean, std dev, count) and saves as PNG for documentation.

Step 6: Repeat Steps 2-5 for High Speed Fans

Input: `fan_high_df` (pandas DataFrame)
Output: Cleaned `fan_high_df` + histogram

```
# Apply same cleaning pipeline to high-speed fans
importcsv.show_data_summary(fan_high_df)
fan_high_df = importcsv.remove_missing_data(fan_high_df)
fan_high_df = clean.clean_tman(fan_high_df)
prep.create_fan_speed_histogram(fan_high_df, 'High Speed Fans', 'out
```

Module: `importcsv.py`, `clean.py`, `prep.py`

Description: Parallel processing of high-speed fan data through identical cleaning pipeline. Ensures both datasets undergo same transformations before balancing and analysis.

Step 7: Balance Dataframes by Manufacturer

Input: `fan_low_df`, `fan_high_df` (cleaned DataFrames)
Output: `balanced_low_df`, `balanced_high_df` (balanced DataFrames)

```
# Balance both dataframes to equal sample sizes per manufacturer
balanced_low_df, balanced_high_df = balance.balance_dataframes(fan_l
```

Module: `balance.py` → `balance_dataframes()`

Description: Stratified random sampling ensures equal representation of each manufacturer in both low and high-speed datasets. Improves DOE design orthogonality and statistical power.

Balancing Method:

- Find minimum count across all manufacturers
- Sample exactly that many observations per manufacturer
- Preserve stratification across both datasets

Step 8: Export Balanced Dataframes to CSV

Input: `balanced_low_df`, `balanced_high_df`
Output: `outputs/balanced_low_df.csv`, `outputs/balanced_high_df.csv`

```
# Save balanced dataframes for reproducibility
export.export_fan_dfs_to_csv(balanced_low_df, balanced_high_df, 'out
```

Module: `export.py` → `export_fan_dfs_to_csv()`

Description: Persists balanced dataframes to CSV format for reproducibility and auditing. Enables rerunning downstream analysis without rebalancing.

Step 9: Create Visualization: Fan Speed Comparison

Input: `balanced_low_df`, `balanced_high_df`
Output: `outputs/fan_hl_histogram.html` (interactive Plotly)

```
# Create side-by-side histogram comparison
viz.create_fan_hl_histogram(balanced_low_df, balanced_high_df, 'outp
```

Module: `viz.py` → `create_fan_hl_histogram()`

Description: Interactive Plotly visualization comparing fan speed distributions. Shows statistics (mean, std dev, variance, count) for both populations. HTML file enables zooming and inspection.

Step 10: Create DOE Design Setup

Input: `balanced_low_df`, `balanced_high_df`
Output: `doe_df` (combined with speed indicator), design summary

```python
# Prepare design of experiments framework
doe_df = doe.setup_doe_design(balanced_low_df, balanced_high_df)
```

Module: `doe.py` → `setup_doe_design()`

Description: Combines balanced dataframes and adds Fan_Speed_Range indicator ('L' or 'H'). Displays factor level summary and confirms data ready for DOE analysis.

Design Factors:

- **Transceiver_Manufacturer:** 10 categorical levels
- **Fan_Speed_Range:** 2 levels (Low, High)
- **Rack_Unit:** Continuous, treated as 42 discrete levels (1-42)

Step 11: Create Full Factorial Design Table

Input: `doe_df`
Output: `outputs/doe_design.html`, `design_table` DataFrame

```python
# Generate all factor combinations
design_table = doe.create_full_factorial_design(doe_df, 'outputs')
```

Module: `doe.py` → `create_full_factorial_design()`

Description: Generates all 840 possible factor combinations (10 manufacturers × 42 rack units × 2 speeds). Saves as HTML table for reference.

Step 12: Fit Full DOE Model

Input: `doe_df`
Output: Full model results, ANOVA table, parameter estimates

```python
# Fit full factorial model with all interactions
full_model, full_results, full_summary = doe.fit_doe_model(doe_df, '
```

Module: `doe.py` → `fit_doe_model()`

Description: Fits full-factorial regression model treating all factors as categorical:

```
Interface_Temp ~ C(Manufacturer) + C(Rack_Unit) + C(Speed) +
                 C(Manufacturer):C(Rack_Unit) +
                 C(Manufacturer):C(Speed) +
```

```
                   C(Rack_Unit):C(Speed) +
                   C(Manufacturer):C(Rack_Unit):C(Speed)
```

Model Statistics:

- Parameters: 820
- R^2: 0.3897
- Adjusted R^2: 0.3830
- F-statistic: 58.42
- p-value: < 0.001

Step 13: Fit Reduced DOE Model

Input: `doe_df`, `full_results`
Output: Reduced model results, statistics

```python
# Fit reduced model with significant terms only (p ≤ 0.05)
reduced_model, reduced_results, reduced_summary = doe.fit_reduced_dc
    doe_df, full_results, alpha=0.05, output_dir='outputs'
)
```

Module: `doe.py` → `fit_reduced_doe_model()`

Description: Removes non-significant terms from full model, retaining only $p \leq 0.05$. Achieves 45% parameter reduction while maintaining R^2 within 1% of full model.

Reduced Model:

- Parameters: 451 (-45%)
- R^2: 0.3852 (-1.1%)
- MSE: 1.7460 (+0.1%)
- Significant terms: 25

Step 14: Generate HTML Reports

Input: Full/reduced model results
Output: `outputs/doe_analysis_report.html`, `outputs/doe_analysis_reduced.html`

```python
# Create detailed HTML reports with tables and diagnostics
# (automatically called during model fitting)
```

Module: `doe.py` → `create_doe_report()`, `create_reduced_doe_report()`

Description: Generates comprehensive HTML reports including:

- Model fit statistics and diagnostic tables
- ANOVA tables (Type I sequential)
- Lack-of-fit test results
- Parameter estimates with 95% confidence intervals
- Interaction plots with visual diagnostics

Step 15: Generate PDF Reports

Input: HTML reports
Output: `outputs/*_summary.pdf`

```
# Convert HTML reports to PDF using multiple generators
pdf_generator_plotly.create_reduced_model_pdf_enhanced(
    'outputs/doe_analysis_reduced.html',
    'outputs/doe_analysis_reduced_summary.pdf'
)
```

Module: `pdf_generator_plotly.py` → `create_reduced_model_pdf_enhanced()`

Description: Converts HTML reports to PDF format with embedded Plotly charts. Includes model fit diagrams, parameter tables, and leverage plots. Professional formatting for distribution.

Step 16: Create PowerPoint Presentations

Input: HTML/PDF reports
Output: PowerPoint files (.pptx)

```
# Generate PowerPoint presentations from analysis reports
powerpoint_generator.create_full_model_powerpoint(
    'outputs/doe_analysis_report.html',
    'outputs/doe_analysis_report.pptx'
)

powerpoint_generator.create_reduced_model_powerpoint(
    'outputs/doe_analysis_reduced.html',
    'outputs/doe_analysis_reduced.pptx'
)

powerpoint_generator.create_comparison_powerpoint(
    'outputs/doe_analysis_report.html',
```

```
    'outputs/doe_analysis_reduced.html',
    'outputs/doe_model_comparison.pptx'
)
```

Module: `powerpoint_generator.py` → Multiple functions

Description: Creates three PowerPoint presentations:

1. **Full Model Analysis** (doe_analysis_report.pptx)
   - 4 slides + leverage plots
   - All 820 parameters and interactions
2. **Reduced Model Analysis** (doe_analysis_reduced.pptx)
   - 4 slides + leverage plots
   - Streamlined 451-parameter model
3. **Comparison Presentation** (doe_model_comparison.pptx)
   - Side-by-side metrics
   - Model effectiveness comparison
   - Recommendation summary

Module Reference

13 Python Modules

1. importcsv.py (3 functions, 100 lines)

Purpose: CSV data import and initial summary

Function	Description
`load_csv_data()`	Load raw CSV into pandas DataFrame
`show_data_summary()`	Display head, tail, column names, row/column counts
`remove_missing_data()`	Remove rows with any NaN values

Used in Steps: 1, 2, 3, 6

---

2. clean.py (2 functions, 78 lines)

Purpose: Data cleaning and validation

Function	Description

`analyze_device_vendors()`	Analyze and summarize vendor distribution
`clean_tman()`	Standardize transceiver manufacturer names (10 categories)

Used in Steps: 4, 6

---

3. prep.py (2 functions, 113 lines)

Purpose: Data preparation and visualization

Function	Description
`calculate_fan_speed_mean()`	Compute mean of raw fan speed measurements
`create_fan_speed_histogram()`	Generate and save fan speed distribution histogram

Used in Steps: 5, 6, 9

---

4. split.py (1 function, 24 lines)

Purpose: Split data by fan speed range

Function	Description
`split_fan()`	Split dataframe into low/high speed fan populations

Used in: Original pipeline (now bypassed by loading pre-split data)

---

5. balance.py (1 function, 52 lines)

Purpose: Balance datasets by categorical level

Function	Description
`balance_dataframes()`	Stratified sampling to equalize manufacturer representation

Used in Steps: 7

---

6. export.py (1 function, 34 lines)

Purpose: Export results to CSV

Function	Description
`export_fan_dfs_to_csv()`	Save balanced dataframes to CSV files

Used in Steps: 8

---

7. viz.py (2 functions, 288 lines)

Purpose: Generate interactive visualizations

Function	Description
`create_fan_hl_histogram()`	Side-by-side fan speed distribution comparison (Plotly)
`create_ttemp_hl_histogram()`	Side-by-side interface temp distribution comparison (Plotly)

Used in Steps: 9, 16

---

8. doe.py (12 functions, 1,629 lines)

Purpose: Design of Experiments core logic

Function	Description
`setup_doe_design()`	Prepare DOE framework and combine dataframes
`create_full_factorial_design()`	Generate all factor combinations table
`fit_doe_model()`	Fit full 820-parameter factorial model
`_calculate_lack_of_fit()`	Calculate lack-of-fit test statistics
`_debug_model_comparison()`	Compare full vs reduced model parameters
`fit_reduced_doe_model()`	Fit reduced model with α=0.05 significance threshold
`_clean_label()`	Utility: format parameter labels
`_clean_formula()`	Utility: format model formula strings
`create_interaction_plots()`	Generate interaction plot visualizations
`create_doe_report()`	Generate full model HTML report
`create_reduced_doe_report()`	Generate reduced model HTML report
`convert_html_to_pdf()`	Convert HTML to PDF

Used in Steps: 10, 11, 12, 13, 14

---

9. **pdf_generator.py** (2 functions, 351 lines)

Purpose: Basic PDF report generation

Function	Description
`create_design_summary_pdf()`	Generate DOE design overview PDF
`create_analysis_summary_pdf()`	Generate model analysis summary PDF

Used in Steps: 15

---

10. **pdf_generator_enhanced.py** (1 function, 578 lines)

Purpose: Enhanced PDF with visual extractions

Function	Description
`create_reduced_model_pdf_with_visuals()`	Extract HTML/PDF images into formatted PDF report

Used in Steps: 15

---

11. **pdf_generator_plotly.py** (5 functions, 830 lines)

Purpose: Advanced PDF with Plotly chart capture

Function	Description
`extract_plotly_chart_as_image()`	Convert Plotly figures to PNG images
`create_model_formula_string()`	Generate model formula display text
`create_parameters_table()`	Create parameter summary table
`create_reduced_model_pdf_enhanced()`	Generate comprehensive reduced model PDF
`(additional helper functions)`	Support visualization extraction

Used in Steps: 15

---

12. **powerpoint_generator.py** (14 functions, 1,233 lines)

Purpose: PowerPoint presentation generation

Function	Description
`extract_model_fit_plot_from_pdf()`	Extract model diagram from PDF
`extract_model_diagram_image()`	Wrapper for model diagram extraction
`extract_base64_images_from_html()`	Decode base64 images from HTML
`extract_html_tables()`	Parse tables from HTML using pandas
`extract_interaction_plots_from_html()`	Extract Plotly interaction plots
`create_title_slide()`	Create formatted title slide
`create_equation_slide()`	Create model equation display slide
`create_content_slide()`	Create text/table/image content slide
`add_image_to_slide()`	Utility: add image to existing slide
`create_full_model_powerpoint()`	Generate full model presentation
`create_reduced_model_powerpoint()`	Generate reduced model presentation
`add_side_by_side_leverage_comparisons()`	Create comparison slides with paired leverage plots
`create_comparison_powerpoint()`	Generate full vs reduced comparison presentation
`convert_html_to_powerpoint()`	Orchestrate all PowerPoint conversions

Used in Steps: 16

---

13. doep.py (Main orchestration script)

Purpose: Pipeline orchestration and execution

Main Functions:

- Calls all 12 steps in sequence
- Error handling and logging
- Output validation

Entry Point: `if __name__ == "__main__":`

Data Flow Diagram

```
┌──────────────────────────────────┐
│  INPUT: CSV FILES                │
│  - fan_low_df.csv  (12,469 rows) │
│  - fan_high_df.csv  (3,848 rows) │
└──────────────────────────────────┘
         │
         │
         ▼
    ┌────────────────────────────────┐
    │ STEP 1-3: IMPORT & CLEAN       │
    │ - Load CSVs                     │
    │ - Remove NaN rows               │
    │ - Standardize manufacturer      │
    └─────────────────────────────────┘
         │
         │
         ▼
    ┌────────────────────────────────┐
    │ STEP 4-6: REPEAT FOR HIGH      │
    │ Parallel cleaning pipeline      │
    │ for high-speed fan data         │
    └─────────────────────────────────┘
         │
         │
         ▼
    ┌────────────────────────────────┐
    │ STEP 7: BALANCE BY LEVEL       │
    │ Stratified random sampling      │
    │ Equal observations/manufacturer│
    └─────────────────────────────────┘
         │
         │
         ▼
    ┌────────────────────────────────┐
    │ STEP 8-9: VISUALIZE & EXPORT   │
    │ - Export to CSV                │
    │ - Create comparison plots       │
    └─────────────────────────────────┘
         │
         │
         ▼
    ┌────────────────────────────────┐
    │ STEP 10-13: DESIGN & ANALYZE   │
    │ - Setup DOE framework          │
    │ - Create design table          │
    │ - Fit full model (820 params)  │
    │ - Fit reduced model (451)      │
    └─────────────────────────────────┘
         │
         │
         ▼
    ┌────────────────────────────────┐
    │ STEP 14: GENERATE REPORTS      │
```

```
│ - HTML with tables/plots      │
│ - Statistics & diagnostics    │
└───────────────────────────────┘
     │
     ▼
┌───────────────────────────────┐
│ STEP 15: CREATE PDFs          │
│ - Convert HTML to PDF         │
│ - Embed visualizations        │
└───────────────────────────────┘
     │
     ▼
┌───────────────────────────────┐
│ STEP 16: CREATE POWERPOINTS   │
│ - Full model presentation     │
│ - Reduced model presentation  │
│ - Comparison presentation     │
└───────────────────────────────┘
     │
     ▼
┌───────────────────────────────┐
│ OUTPUT: DELIVERABLES          │
│ Location: ./outputs/          │
│                               │
│ CSV Files:                    │
│ - balanced_low_df.csv         │
│ - balanced_high_df.csv        │
│ - doe_design.html             │
│                               │
│ HTML Reports:                 │
│ - doe_analysis_report.html    │
│ - doe_analysis_reduced.html   │
│                               │
│ PDF Reports:                  │
│ - doe_analysis_*_summary.pdf  │
│                               │
│ PowerPoint Files:             │
│ - doe_analysis_report.pptx    │
│ - doe_analysis_reduced.pptx   │
│ - doe_model_comparison.pptx   │
│                               │
│ Visualizations:               │
│ - fan_*_histogram*.png/html   │
│ - interaction_plots.html      │
└───────────────────────────────┘
```

Key Statistics

Codebase Metrics

Metric	Value
Total Modules	13
Total Functions	48
Total Lines of Code	6,189
Significant Lines (non-comment/blank)	5,467
Documentation Lines	722

Module Complexity

Module	Lines	Complexity	Functions
doe.py	1,629	High	12
powerpoint_generator.py	1,233	High	14
pdf_generator_plotly.py	830	Medium	5
pdf_generator_enhanced.py	578	Medium	1
pdf_generator.py	351	Medium	2
viz.py	288	Medium	2
prep.py	113	Low	2
clean.py	78	Low	2
balance.py	52	Low	1
export.py	34	Low	1
split.py	24	Low	1
importcsv.py	100	Low	3
Total	**6,189**	**Mixed**	**48**

Documentation Coverage

Category	Count	Coverage
Functions with docstrings	48	100%
Docstring format	PEP 257	Standard
Module docstrings	13	100%

Examples in docstrings	15+	31%

Pipeline Performance Characteristics

- **Input Data Size:** ~16,300 rows
- **Analysis Time:** ~5-10 minutes (depending on model fitting)
- **Output Size:** ~50-100 MB (HTML + PDF + PowerPoint)
- **Memory Footprint:** ~500 MB peak

Design of Experiments Details

DOE Model Specification

Full Model (820 parameters)

```
Interface_Temp ~ C(Transceiver_Manufacturer) + C(Rack_Unit) + C(Fan_
                 C(Transceiver_Manufacturer):C(Rack_Unit) +
                 C(Transceiver_Manufacturer):C(Fan_Speed_Range) +
                 C(Rack_Unit):C(Fan_Speed_Range) +
                 C(Transceiver_Manufacturer):C(Rack_Unit):C(Fan_Spee
```

Model Statistics:

- $R^2 = 0.3897$
- Adjusted $R^2 = 0.3830$
- F-statistic = 58.42
- p-value < 0.001
- Residual Std. Error = 1.3208
- DoF (Residual) = 7,382

Reduced Model (451 parameters)

Selection Criterion: p-value ≤ 0.05

```
Interface_Temp ~ C(Transceiver_Manufacturer) + C(Rack_Unit) +
                 C(Transceiver_Manufacturer):C(Rack_Unit) +
                 C(Rack_Unit):C(Fan_Speed_Range)
```

Model Statistics:

- $R^2 = 0.3852$ (-1.14% vs full)
- Adjusted $R^2 = 0.3816$ (-0.04% vs full)
- F-statistic = 108.36 (+85.3% vs full)

- p-value < 0.001
- MSE = 1.7460 (+0.13% vs full)

Factor Levels

Factor	Type	Levels	Description
Transceiver_Manufacturer	Categorical	10	Device vendors (+ "Others")
Fan_Speed_Range	Categorical	2	"L" (< 9,999 rpm), "H" (≥ 10,000 rpm)
Rack_Unit	Categorical*	42	Integer positions 1-42

*Treated as categorical for main effects despite continuous source

Statistical Tests

1. **ANOVA (Type I - Sequential)**
   - Partitions variance by factor
   - Tests main effects first, then interactions
2. **Lack-of-Fit Test**
   - Pure error from replicate observations
   - Assesses adequacy of model form
   - LOF p-value: 0.200 (model adequate)
3. **Confidence Intervals**
   - 95% CI for all parameter estimates
   - Enables precision assessment

Key Findings

1. **Manufacturer Effect:** Highly significant (p < 0.001)
   - Different manufacturers produce significantly different temperatures
   - 217.44 F-statistic value
2. **Rack Unit Effect:** Highly significant (p < 0.001)
   - Position in rack affects temperature
   - 6.02 F-statistic value
3. **Interaction Effects:**
   - Manufacturer × Rack Unit: Significant (p < 0.001)
   - Rack Unit × Speed: Marginal (p = 0.200)

Additional Resources

📚 DOE Pipeline Documentation

- **DOEP_SETUP.md** - Installation and environment setup guide
- **DOEP_LIB_REQS.md** - Complete library requirements reference
- **DOCSTRING_PLAN.md** - Comprehensive documentation planning guide
- **outputs/** - Generated reports and visualizations

Support & Troubleshooting

Common Issues

Issue: `ModuleNotFoundError: No module named 'statsmodels'`

- **Solution:** Run `pip install -r requirements.txt`

Issue: PDF generation fails with pdfkit error

- **Solution:** Install wkhtmltopdf: `brew install --cask wkhtmltopdf` (macOS)

Issue: PowerPoint generation incomplete

- **Solution:** Check that python-pptx is installed: `pip install python-pptx`

Performance Optimization

- Run on multi-core system for faster model fitting
- Reduce iterations for testing with smaller dataframes
- Clear outputs directory before re-running for fresh results

---

Document Version: 1.0
Last Modified: 2024
Maintained By: DOE Pipeline Development Team