

SPSS-MCP: A Psychological Statistics Automation System Based on Claude Code

Authors: Jingshen Xue, Zhu Tingshao, Zhu Tingshao

Date: 2026-04-10T12:38:34+00:00

Abstract

This paper introduces a method for the deep integration of Claude Code with IBM SPSS Statistics. By developing an SPSS server based on the Model Context Protocol (MCP), referred to as SPSS-MCP, Claude Code is enabled to directly invoke SPSS to perform statistical analyses. The system is built upon a unified execution engine based on the SPSS XD API, requiring IBM SPSS Statistics to be installed in a Windows environment; all statistical analyses are completed by invoking the SPSS backend through this engine. The server provides a total of 36 tools, covering common statistical methods in psychological research such as basic statistics, advanced regression, multilevel models, survival analysis, multivariate analysis, cluster analysis, and reliability analysis. This paper details the system architecture and the principles of core technical implementation, and offers best practice recommendations for psychology researchers, aiming to help researchers significantly enhance the efficiency and accessibility of statistical analysis with the assistance of AI.

Full Text

Preamble

SPSS-MCP: An Automated Psychological Statistics System Based on Claude Code Jingshen Xue^{1,2}, Tingshao Zhu^{1,2}

(1 Institute of Psychology, Chinese Academy of Sciences, Beijing 100101; 2 Department of Psychology, University of Chinese Academy of Sciences, Beijing 100049)

Abstract

This paper introduces a method for the deep integration of Claude Code with IBM SPSS Statistics by developing an SPSS server based on the Model Context

Protocol (MCP). This system, referred to as SPSS-MCP, enables Claude Code to directly invoke SPSS to perform statistical analyses. The system is built upon a unified execution engine utilizing the SPSS XD API, which requires IBM SPSS Statistics to be installed in a Windows environment; all statistical analyses are executed through this engine by calling the SPSS backend. The server provides a total of 36 tools, covering common statistical methods used in psychological research, including descriptive statistics, advanced regression, multilevel modeling, survival analysis, multivariate analysis, and cluster analysis.

Additionally, the server supports reliability analysis and other essential techniques. This paper provides a detailed description of the system architecture and the principles of its core technical implementation. Furthermore, it offers best practice recommendations specifically tailored for psychological researchers. The objective of this integration is to assist researchers in leveraging AI to significantly enhance the efficiency and accessibility of statistical analysis.

Keywords

Model Context Protocol; IBM SPSS Statistics; Large Language Model; Automated Statistical Analysis

1 Introduction

Statistical analysis is a core component of scientific research in psychology. As psychological research methods continue to evolve, researchers face increasingly complex statistical challenges. These range from basic descriptive statistics and hypothesis testing to mixed-effects models and structural equation modeling (SEM) within longitudinal designs, as well as multivariate dimensionality reduction and latent class analysis (LCA). The diversity and complexity of these methodologies require researchers to possess both a profound foundation in statistics and high proficiency in operating specialized software.

IBM SPSS Statistics remains one of the most widely used statistical software packages in the field of psychology [?, ?]. Its widespread adoption stems primarily from its mature and stable statistical engine, standardized report outputs, and a graphical user interface (GUI) that offers a low barrier to entry by minimizing programming requirements. Furthermore, it provides robust support for psychometric scale analysis, such as Cronbach's α and exploratory factor analysis (EFA). However, the interactive operational mode of SPSS limits the degree of automation in analysis workflows. With the rise of the Open Science movement, a growing contradiction has emerged between the demand for code-based reproducible research and the menu-driven operation of SPSS [?, ?]. These difficulties are prevalent in psychological research and constitute a significant obstacle, particularly for early-career researchers. Lowering the technical threshold for statistical analysis and enhancing research efficiency are critical tasks for promoting the popularization of psychological methodology.

The rapid development of Large Language Models (LLMs) offers a new pathway to address these challenges. The introduction of ChatGPT Code Interpreter (later renamed Advanced Data Analysis) by OpenAI in 2023 first demonstrated the capability of LLMs to complete data analysis tasks by writing and executing Python code [?]. Users can describe their analytical needs in natural language, and the system automatically generates code, executes calculations, and returns results, significantly lowering the technical barriers to data analysis.

Recent studies have further systematically evaluated the potential of LLMs in the fields of statistics and data science. A review by Rahman et al. [?] outlined the design paradigms and performance boundaries of LLM-based data science agents. The “Data Interpreter” framework proposed by Hong et al. [?] demonstrated the systematic application of LLM agents in end-to-end data science tasks. Nascimento et al. [?] specifically evaluated the performance of mainstream LLMs, such as Claude and GPT-4, in data science code generation tasks. However, existing LLM data analysis solutions generally rely on Python or R code generation paradigms, which involve significant transition costs for psychological researchers accustomed to the SPSS workflow. Researchers must learn entirely new programming languages and statistical libraries (such as `scipy`, `statsmodels`, or `pingouin`). Moreover, differences often exist between Python-generated statistical results and SPSS outputs in terms of formatting, precision, and psychometric conventions, which can affect the comparability and professional acceptance of research findings.

The Model Context Protocol (MCP) is an open standard released by Anthropic in November 2024, designed to solve the standardization issues of integrating LLM applications with external tools and data sources [?, ?]. Adopting a client-server architecture, MCP defines three core capabilities: Tools, Resources, and Prompts. MCP servers communicate with MCP hosts (such as Claude Desktop or Claude Code) via standard input/output (stdio) or HTTP/SSE transport, eliminating the redundant work of developing separate integration interfaces for every LLM application.

Following its release, the MCP protocol was rapidly adopted by major AI vendors: OpenAI officially integrated MCP into its product line in March 2025, Google DeepMind announced support for the protocol during the same period, and development platforms such as Replit and Sourcegraph have also connected to the ecosystem. The rapid growth of the MCP ecosystem indicates that protocol-based tool integration represents a major direction for the functional expansion of AI assistants. FastMCP [?] is a lightweight Python implementation of the MCP protocol, providing a decorator-based tool registration mechanism and built-in asynchronous support, which greatly simplifies the complexity of developing MCP servers.

SPSS-MCP is built upon the FastMCP framework. Claude Code is an agentic programming tool designed for terminal and development environments, capable of reading project context, editing files, executing commands, and connecting to external tools, databases, and analytical services via MCP. Consequently,

researchers can directly use natural language to propose tasks—for example, “Perform a reliability analysis on this data and report the results” or “Test the regression effects of satisfaction, motivation, age, and years of service on performance.” The system then automatically handles tool calls, parameter organization, and result integration.

Compared to traditional statistical software that relies on menu operations or manual syntax writing, this “natural language—tool call—result return” interaction mode significantly lowers the barrier to using statistical analysis systems. It also allows non-expert programmers to conveniently invoke existing analytical capabilities. The official Claude Code documentation identifies the integration of external tools and data sources via MCP as one of its core extension methods and explicitly supports natural language-driven multi-step task execution.

This paper proposes an MCP-based SPSS server that connects Claude Code to IBM SPSS Statistics for statistical analysis, thereby creating an end-to-end workflow from natural language instructions to SPSS syntax generation, engine execution, and result retrieval. The system is built on a unified execution engine using the SPSS XD API and implements a set of tools and toolchain organization examples covering common methods in psychological research. Additionally, this paper provides key implementation details and usage recommendations, and releases the project code under the MIT license to facilitate reuse and extension.

2 System Architecture and Implementation

2.1 Overall Architecture

The overall architecture of SPSS-MCP is composed of the following core modules: Claude Code (the MCP host), the FastMCP server layer (server.py), the SPSS execution engine (spss_{runner}.py), the file I/O engine (sav_{reader}.py), the method registry (comprising method_{registry}.py, method_{templates}.py, and method_{validation}.py), and the IBM SPSS Statistics backend.

Claude Code communicates with the FastMCP server via the MCP protocol (using stdio transport). Based on the results of tool capability detection, the server routes requests to either the SPSS execution engine (for tasks requiring SPSS) or the file I/O engine (for metadata reading tools). Each SPSS analysis session persists .spv and .sps files within the SPSS_{{{RESULTS}}}_{DIR}} directory. The system executes analyses according to the following workflow (see Figure 1 [Figure 1: see original paper]):

[Figure 1: see original paper] SPSS XD API Execution Workflow

2.2 Tool Design Patterns

SPSS-MCP provides a total of 36 MCP tools, which are categorized into nine major functional classes (see). The design of these tools covers the entire workflow

of quantitative psychological research, ranging from initial data preparation to the final generation of statistical reports. All 36 statistical analysis tools follow a standardized design framework: Capability Detection → Syntax Construction → SPSS Execution → Output Parsing → Markdown Return.

Specifically, the utility function first invokes `_{{require}}_{{spss}}(ctx)` to verify the availability of SPSS; if the environment is unavailable, it returns an error message immediately. Subsequently, it calls the corresponding `build_{{syntax}}` function to construct the command.

To enhance system scalability and maintainability, this project introduces a registry-based method management mechanism. Methods that are used relatively infrequently but still hold significant application value—such as Cox regression, discriminant analysis, and two-step clustering—are managed through a centralized method registry (`method_{{registry}}.py`).

The registry defines a structured parameter schema (JSON Schema), an SPSS syntax template (`method_{{templates}}.py`), and parameter validation rules (`method_{{validation}}.py`) for each method. This architectural design ensures that adding new methods does not require modifications to the core server code; instead, new functionality can be integrated simply by declaring the method definition within the registry.

Furthermore, the system provides the `spss_{{run}}_{{syntax}}` tool, which allows for the direct execution of arbitrary custom SPSS syntax. This tool is designed for analysis requirements that fall outside the scope of existing pre-packaged tools, as well as for exploratory analysis and advanced customized analysis scenarios.

Functional Features and Specifications

Category	Tools	Description
Data Management	<code>spss_{{check}}_{{status}}</code> , <code>spss_{{list}}_{{variables}}</code> , <code>spss_{{read}}_{{metadata}}</code> , <code>spss_{{read}}_{{data}}</code> , <code>spss_{{file}}_{{summary}}</code>	Handles metadata collection, variable definition, and data import, and data reviews.
Descriptive Analysis	<code>spss_{{frequencies}}</code> , <code>spss_{{descriptives}}</code> , <code>spss_{{crosstabs}}</code>	Generates frequency tables, descriptive statistics, and contingency tables.
Hypothesis Testing	<code>spss_t_{{test}}</code> , <code>spss_{{anova}}</code> , <code>spss_{{correlations}}</code> , <code>spss_{{nonparametric}}_{{tests}}</code>	Performs t-tests, ANOVA, Pearson/Spearman correlation, and non-parametric tests.

Category	Tools	Description
Advanced Modeling	<code>spss_{regression}</code> , <code>spss_{{logistic}}_{regression}</code> , <code>spss_{mixed}</code> , <code>spss_{{ordinal}}_{regression}</code> , <code>spss_{genlin}</code>	Supports multiple linear regression, logistic regression, LMM, and GEE.
Survival Analysis	<code>spss_{{kaplan}}_{meier}</code> , <code>spss_{{cox}}_{regression}</code>	Used for survival curves and hazard rate evaluation.
Multivariate Analysis	<code>spss_{factor}</code> , <code>spss_{discriminant}</code> , <code>spss_{manova}</code>	Supports EFA, discriminant analysis, and MANOVA.
Cluster Analysis	<code>spss_{{cluster}}_{hierarchical}</code> , <code>spss_{{twostep}}_{cluster}</code>	Performs hierarchical and two-step clustering.
Psychometrics	<code>spss_{{reliability}}_{calculate}</code> , <code>spss_{{compute}}_{scale}</code>	Calculates Cronbach's alpha and composite scale scores.
System Utilities	<code>spss_{{run}}_{syntax}</code> , <code>spss_{{validate}}_{syntax}</code> , <code>spss_{{list}}_{supported}</code>	Executes custom syntax files, validates registry syntax files, and lists supported methods.

2.3 Error Handling and Fault Tolerance

The system implements a multi-layered error handling and fault tolerance strategy. First, to prevent uncontrollable wait times when processing large-scale datasets or complex models, all SPSS subprocess calls are configured with adjustable timeout thresholds (defaulting to 120 seconds); this allows the system to proactively terminate processes and return diagnostic information if execution time extends abnormally. Second, the system employs an error classification mechanism that distinguishes between fatal errors (`__{{spss}}_{error}`), such as syntax errors or insufficient memory) and non-fatal warnings.

These warnings (`__{{spss}}_{warn}`), such as unmet analytical assumptions) are handled through distinct error returns and warning prompts, ensuring that the calling party can implement appropriate corrective or explanatory strategies. Furthermore, at the process governance level, the system automatically deletes temporary Python script files upon task completion regardless of the execution outcome, thereby mitigating the risk of disk space leaks and maintaining a clean operating environment. Finally, to address potential artifact overwriting caused by concurrent requests, all temporary files utilize a naming scheme that incorporates millisecond-level timestamps. This mechanism fundamentally prevents filename conflicts, enhancing both stability and traceability in high-concurrency scenarios.

2.4 Testing and Verification

This study selected 10 typical cases from the teaching dataset of *Application of SPSS 26.0 Statistical Analysis in Psychology and Education* (Dai & Jian, 2023) for end-to-end replication. These cases cover 10 categories of statistical methods: descriptive statistics, Pearson correlation, independent samples t-test, two-way ANOVA, exploratory factor analysis, multiple linear regression, Cronbach's α reliability analysis, Wilcoxon signed-rank test, hierarchical clustering, and analysis of covariance (ANCOVA). The replication success rate for these 10 cases was 100% (10/10). All statistical conclusions for every case—including significance levels, effect sizes, and model fit indices—were entirely consistent with the results obtained through manual SPSS operations.

2.5 Comparison with Traditional Workflows

Traditional SPSS Workflow: This process typically involves manual menu navigation or the manual writing of syntax, requiring users to constantly consult software documentation and statistical textbooks. Menu-based operations are difficult to document systematically, often necessitating the mastery of complex macro languages. Consequently, researchers must possess both proficiency in SPSS-specific operations and deep statistical expertise. Output is generally limited to the native SPSS Viewer (.spv) format.

SPSS-MCP Workflow: This modern approach utilizes natural language instructions, where AI recommends an appropriate toolchain based on the specific research design. The system provides real-time generation of statistical reports, while .sps files maintain a complete and transparent record of the analysis syntax. Users can describe batch processing requirements in plain language, shifting the focus from software mastery to a fundamental understanding of statistical concepts. Results are provided in dual Markdown and .spv formats, integrated with native SPSS functionality and proactive AI assistance.

3 Implementation and Best Practices

3.1 System Requirements and Configuration

The operation of SPSS-MCP depends on the following hardware and software environment. The operating system must be Windows 10 or Windows 11; this restriction stems from the fact that the IBM SPSS XD API only provides batch processing interfaces for Windows. The Python runtime requires version 3.10 or higher (the implementation in this paper was validated based on Python 3.12). If researchers need to utilize full statistical functions, such as inferential statistics or multivariate analysis, they must also have any version of IBM SPSS Statistics (from version 20 to 31) installed locally. It is recommended to install from source to ensure version consistency:

```
git clone https://github.com/Exekiel179/SPSS-MCP.git
cd SPSS-MCP
```

```
pip install -e .
```

After the installation is complete, run the following command to verify the environment status: `spss-mcp status`

3.2 Typical Usage Scenarios

Before formally conducting statistical analysis, it is recommended that researchers use Claude Code to perform an overall overview of their data files. By simply providing the file path, the system can automatically read variable information, sample sizes, and basic distributional characteristics. This step helps in the early detection of data entry errors, outliers, and coding inconsistencies.

Descriptive statistics are a fundamental requirement for the methods section of a research paper. Claude Code can recognize natural language expressions such as “descriptive statistics” or “mean and standard deviation,” automatically invoking `spss_{descriptives}` or `spss_{frequencies}` to generate standardized tables.

When performing group comparisons, researchers can describe their research questions in a relatively colloquial manner. Claude Code will automatically select the appropriate method based on the nature of the grouping variables and supplement the analysis with post-hoc comparisons (such as Tukey’s HSD) when an ANOVA is significant.

For studies utilizing scales, it is usually necessary to report reliability and validity indicators. Claude Code can perform Cronbach’s α analysis (via the `spss_{reliability}_{alpha}` tool) and Exploratory Factor Analysis (via the `spss_{factor}` tool), automatically outputting key results such as factor loading matrices, explained variance, and the KMO test.

3.3 Natural Language Interaction Example

The following example demonstrates the typical workflow of Claude Code automatically invoking MCP tools upon receiving a natural language request.

User: I have a dataset of employee performance (`employee_{performance}.sav`). I want to investigate the impact of satisfaction, motivation, age, and years of experience on performance. Additionally, I would like to check if there are differences based on gender and education level.

Claude: Understood. I will first confirm the environment and data structure, then plan the analytical path. - [Invoke] `spss_{check}_{status}` → SPSS available - [Invoke] `spss_{file}_{summary}`(...) → 80 cases, 15 variables - [Invoke] `spss_{read}_{metadata}`(...) → Variable labels and value labels confirmed

Claude: I plan to proceed in the following order: Descriptive Statistics → Reliability Analysis → Normality Testing → Correlation Analysis → Multiple

Regression → t-test → ANOVA. Should I begin?

User: Go ahead and start.

Claude: [Executes `spss_{descriptives}`] | Variable | M | SD | Skewness | | :
-| :-| :-| :-| | Age | 39.91 | 11.47 | -0.18 | | Years of Experience | 17.84 | 11.63
| -0.11 | | Performance Score | 93.60 | 8.43 | -1.26 |

[Claude continues with Reliability Analysis, Correlation, and Regression, providing APA-style interpretations and identifying issues like multicollinearity between Age and Experience.]

4 Conclusion

This paper introduces SPSS-MCP: an open-source system based on the Model Context Protocol (MCP) that integrates IBM SPSS Statistics into Claude Code to support natural language-driven statistical analysis. Utilizing the SPSS XD API as a unified execution engine, the system provides a total of 36 tools. Through a registry mechanism, these tools cover the standard methodologies commonly used in psychological research.

The core contribution of SPSS-MCP lies in the implementation and validation of an operational workflow: the Large Language Model (LLM) generates SPSS syntax based on user intent, which is then executed by the SPSS statistical engine to produce results. Building on this foundation, the system supports auditing and replication by persistently saving `.sps` and `.spv` artifacts. Compared to general LLM data analysis solutions that primarily focus on Python or R code generation, this implementation path is more compatible with existing SPSS workflows, helping to reduce transition costs and maintain consistency in output standards. As the MCP ecosystem grows rapidly, SPSS-MCP aims to provide a valuable reference for improving research efficiency in psychology and related social sciences.

References

- Anthropic. (2024). Model Context Protocol: An open standard for connecting AI assistants to data systems. Anthropic Technical Report. <https://modelcontextprotocol.io>
- Field, A. (2018). *Discovering statistics using IBM SPSS Statistics* (5th ed.). SAGE Publications.
- Hong, S., et al. (2024). Data Interpreter: An LLM Agent For Data Science. arXiv:2402.18679.
- Open Science Collaboration. (2015). Estimating the reproducibility of psychological science. *Science*, 349(6251), aac4716.
- Pallant, J. (2020). *SPSS survival manual: A step by step guide to data analysis using IBM SPSS* (7th ed.). Routledge.

Rahman, M., et al. (2025). LLM-Based Data Science Agents: A Survey of Capabilities, Challenges, and Future Directions. arXiv:2510.04023.

Note: Figure translations are in progress. See original paper for figures.

Source: ChinaXiv – Machine translation. Verify with original.