DIMSpec Quick Guide - Installation

Jared M. Ragland*

2024-02-13

Getting set up quickly with DIMSpec

If your needs require only the current example database of per- and polyfluoroalkyl substances (PFAS) and not the attached toolkit, that SQLite file may be downloaded directly from the NIST Public Data Repository and, along with the technical sections in the DIMSpec User Guide and the data dictionary describing its schema (a JSON file located in the project directory), used by any platform able to interact with SQLite databases.

To get the most from DIMSpec, the software toolkit [1] provided with the project enables a wide variety of customized functionality including convenience, utility, and visualization functions written in the R language, an Application Programming Interface powered by `plumber`, web applications powered by `shiny`, and an integration of the python interface to the `rdkit` chemometrics package. A file is provided at `/config/env_glob.txt` which enables configuration customization including database title, database file name, and several other options affecting system integration. Due to its intended reuse and level of available customization, the DIMSpec project is not distributed as an R package.

The easiest way to get started with the full DIMSpec project is by forking the project repository from GitHub, though it may also be downloaded without a GitHub account as a zip file for ease of use (if so, simply extract it to a location of your choosing). Either way, consider signing in and giving the project a Star or Watch to easily keep track of updates. Install the required software (see below) and run the compliance script (e.g. `source("/R/compliance.R")` from the project directory). This should in most cases establish the computational environment supporting the toolkit. When starting from scratch this may take a while as packages are downloaded and installed.

*National Institute of Standards and Technology | Material Measurement Laboratory | Chemical Sciences Division | for assistance email pfas@nist.gov or jared.ragland@nist.gov

Requirements

The DIMSpec project [2] is distributed as an R project built around R version 4.1 and later, and leverages a large number of packages to run as designed. An integrated development environment such as RStudio [3] is highly recommended but is not required. For integration of chemometrics, which is used in several aspects of the project but is not a strict requirement, a python environment including `reticulate` and `rdkit` is required; this will in many cases be installed when the compliance script runs. This can be relaxed for those preferring to use an R chemometrics package, and if informatics is requested in the environment file but rdkit is not, ChemmineR and rcdk will be installed; note that these are not integrated into the project at this time. If you will be developing new databases using this infrastructure, SQLite and its command line interface (CLI) are required; this is what allows the R session to build a fresh database for population. As the project is hosted on GitHub, Git is recommended as the easiest method to download the project and provide version control. SQLite, its CLI, and Git must be available on your system PATH for proper functionality.

Step by Step

See the DIMSpec User Guide for full instructions and how to use DIMSpec. These instructions are a summary of those found there.

1. Install software requirements for your operating system (**required in bold**, *recommended in italics*).

   - **R v4.1 or later** (v4.3+ is recommended) (download)
   - **RTools** (required for package installation if using the Windows operating system) (download)
   - *RStudio or another integrated development environment for R* (optional) (download)
   - SQLite3 (download) and its command line interface (download) (both optional but *recommended* for users who will be building databases with DIMSpec or interacting with databases directly through the command line)
     - *Note*: SQLite3 executables may be installed at any location. On Windows systems, ensure that it is available to PATH in your system or user environment variables (e.g. `%USERPROFILE%/sqlite3` when installed to your `C:/Users/user` directory).
   - Git (download) for version control, *recommended* for advanced users
   - A lightweight database interaction interface (e.g. DBeaver Lite (download) or similar) for direct interaction with the database outside of DIMSpec
   - Miniconda with python 3.8 or later (suggested only if there are problems, the compliance script will install RMiniconda for you if chemometrics support is requested) (download)
     - *Note*: This is included mostly to enable integration of existing workflows built in Python.
     - *Note*: On Windows systems, ensure that the the check box for "Add to PATH" is clicked during installation. This can be added to your system or user environment variables later by adding the `/condabin` directory of where it was installed. To support advanced users with existing Python environments, the project variable `CONDA_PATH` can point to any conda executable; this setting is located in `/inst/rdkit/env_py.R`.

[2] This release was tested on Windows 10 and a fresh VMWare build of Ubuntu 20.04 LTS. Ubuntu carries several additional system requirements. Prior to running DIMSpec on Ubuntu, install or make sure the following are available using:
apt install -y build-essential libcurl4-openssl-dev libxml2-dev zlib1g-dev libssl-dev libsodium-dev ffmpeg libtiff-dev libpng-dev libblas-dev liblapack-dev libarpack2-dev gfortran libcairo2-dev libx11-dev libharfbuzz-dev libfribidi-dev libudunits2-dev libgeos-dev libgdal-dev libfftw3-3 libmagick++-dev
After following the R installation instructions for Ubuntu, ensure additional requirements using:
apt install -y –no-install-recommends r-cran-tidyverse r-cran-shiny

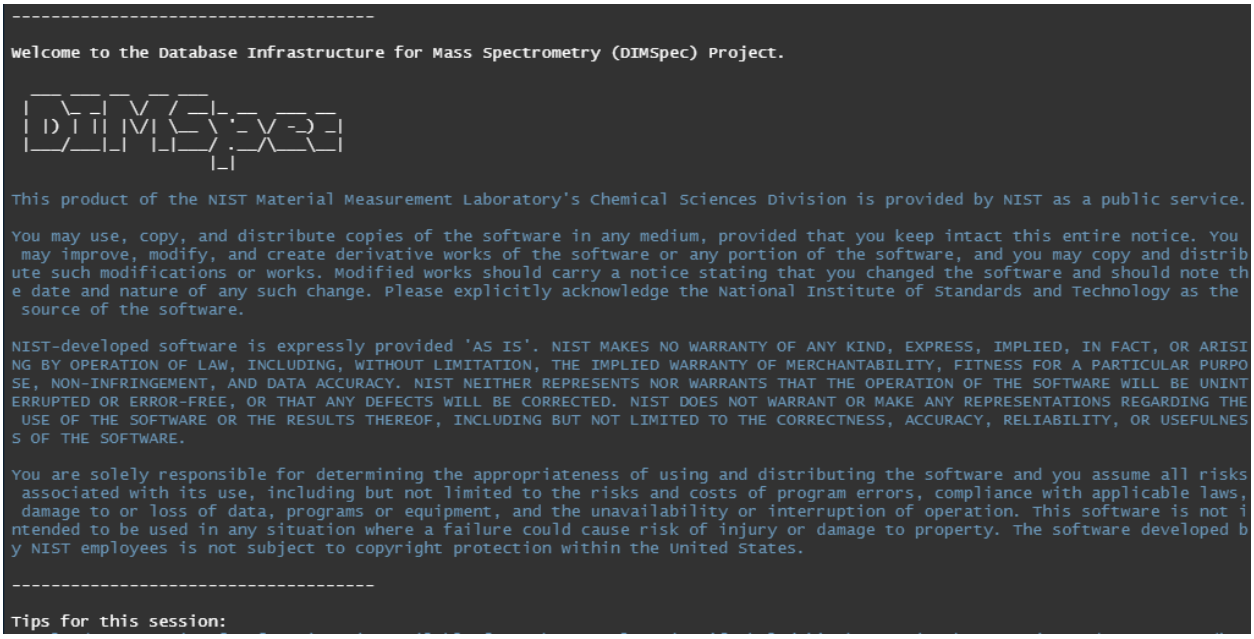
[3] Any mention of commercial products is for information only; it does not imply recommendation or endorsement by NIST.

2. (Optional, advanced) Customize any settings for your implementation (see the Project Set Up section of the DIMSpec User Guide).

3. Run the script at `/R/compliance.R` to establish the compute environment, which by default will:

    - Install (if necessary) and load all necessary packages.
    - Integrate Python and `rdkit` functionality through `reticulate`.
    - Connect to a local SQLite database (one is included with the project download).
    - Establish logging and argument verification.
    - Launch a `plumber` API for database communication with applications.
    - Make three `shiny` web applications available.

These steps may take a while the first time. If DIMSpec is installed correctly and ready for use, after running the compliance script you should see the following splash screen in the console:



Once installed, all aspects of DIMSpec should be ready to use. Tips for the current session will be displayed below the splash screen. From here you may start any of the included web applications (e.g. `start_app("msmatch")`), view the help documentation for all included functions with `fn_guide()`, or open the full user guide with `user_guide()`. Start a new DIMSpec session in the future by running the compliance script again.

Potential Installation Issues

You may run into issues or feedback when installing DIMSpec. These may include requests to allow certain applications through your firewall (particularly on Windows) or package installation issues. The compliance script should install everything needed to run DIMSpec. Package installation issues are often system-dependent and can be fixed in the console. If the console displays messages along the lines of `there is no package called 'package'` you may force install it with `install.packages("package")` . Once you have done so, run the compliance script again.

Developing With DIMSpec

More information on developing with DIMSpec is provided in the full DIMSpec User Guide or the Quick Guide - Developing with DIMSpec

A Note on Environments

While the provided convenience script for environment establishment (located at `/R/compliance.R`) should in most cases establish the proper environment for you, it is admittedly using a large number of packages. To increase flexibility across various computing and operating systems without modifying libraries that may already be in place, a resolved environment is not provided. The most common place where difficulties may be encountered therefore is package version conflicts and setting up the Python environment providing `rdkit` integration. For troubleshooting purposes, the list of required packages can be located in the `DEPENDS_ON` environment variable in the `/config/env_R.R` file. Generally, package issues in R will print a meaningful error message to the console and guide users toward a resolution. These occasionally arise due to conflicting dependencies; the order in which packages are installed in the compliance script should minimize this. *On Windows systems, some R packages require that RTools be installed first.* RStudio should provide a streamlined experience for installing packages compared with base R. Conflicts will be displayed in the console; identify package issues such as installation failures by looking for messages such as

Speed Concerns

The full DIMSpec project can take a while to load at first. To speed this up, it is recommended that the compliance script be sourced prior to engaging in any development work with DIMSpec. Two developer resources are provided in the project that may slow down execution times. Both logging and argument validation are performed by default, which increases execution time but provides a wealth of information to assist with development. These may be turned off individually In the environment settings by setting `LOGGING_ON` or `VERIFY_ARGUMENTS` to `FALSE`, respectively; set `MINIMIZE` to `TRUE` to turn off both at once.

Development Assistance Functions

Navigating a project such as DIMSpec, especially one with several environment files determining behaviors, can be daunting. There are two functions specifically built with developers in mind to assist with opening project files from anywhere. Use the `open_env()` function to open an environment file for editing with the only argument being the environment you wish to edit; valid values are “R”, “global”, “logging”, “rdkit”, “shiny”, or “plumber”. This leverages `open_proj_file()`, a more generic function to open any file in the project by partial name match. Both of these are tied to the project directory and leverage the `here` package.

Additionally, since this project is NOT a package, custom functions to assist with R function help documentation are also provided. Use `fn_guide()` to open the DIMSpec function help index in your system browser, or use `fn_help(X)` where `X` is the name of a function (quoted or unquoted) to open that documentation directly (if using RStudio, this will open in the help pane).