# Instructions for Processing FTICR-MS Data using CoreMS

## Table of Contents

## Overview

Due to recent efforts from researchers at EMSL to improve formula assignment and overall processing of high-resolution mass spectrometry data, we are moving away from using Formularity/Formultitude to generate accessible FTICR-MS data and towards **CoreMS**. **CoreMS** is a framework used to analyze many types of mass spectrometry data and there is no single *right* way to use it (though there are plenty of *wrong* ways…). **CoreMS** provides several benefits over our previous workflow:

1) We have finer control over each step of the analytical process from peak calling to formula assignment
2) It is open-source and non-proprietary,
3) It is agnostic towards operating system because it is written using Python.

This instruction set will walk you through how to install CoreMS and all prerequisites, run our SFA's CoreMS script, and regenerate our published datasets (or apply these methods to your own datasets). By default, our CoreMS-based pipeline accepts a variety of data types including raw FIDs from Bruker instruments, XML/mass lists generated using Bruker Data Analysis, and a series of miscellaneous mass lists; the script will identify the data type using the extension. This instruction set assumes limited knowledge of Python, conda/mamba, and similar programs – if you have experience with these, some of this information might be redundant!

**Disclaimer:** We are not affiliated with the CoreMS development team and any issues with this instruction set are not associated with them. Please reach out to us with any issues that arise if you are using our instructions.

## Requirements

- Reading through the CoreMS GitHub page (https://github.com/EMSL-Computing/CoreMS)
- Any computer

    - These instructions are written for Windows and macOS; if you are using Linux, the macOS instructions should be equivalent
    - Some computers will likely run through the analyses slower than others, so the computer should be recent (newer than 2020-2021)
- mamba installed via Miniforge
    - Please read through and follow the instructions that corresponds to your device on their GitHub: https://github.com/conda-forge/miniforge
    - mamba allows us to use "virtual environments" which give us complete control of versioning (e.g., things installed into one virtual environment don't impact the rest of your computer or other environments)
- R and RStudio
    - Download and install R: https://www.r-project.org/
    - Download and install RStudio: https://posit.co/download/rstudio-desktop/
- *Recommended, but optional*: An integrated development environment (IDE) of your choice – we recommend using VS Code (https://code.visualstudio.com/)
    - An IDE is a program that allows you to have a bit more control over scripting and coding that simply writing in a simple text editor. This is useful if you plan on editing the script and want to do some testing on your own.
    - VS Code is compatible with many different languages and, notably, interfaces well with virtual environments
    - Each IDE will need to be configured differently so please look up how to configure VS Code if you desire

## Instructions for Installing CoreMS

1. Download CoreMS repo from GitHub (either via the website or using git clone)
    a. Make sure you store it somewhere memorable and intentional where you won't accidentally delete it – let's call this location **/Documents/CoreMS-[*current date or version*]/**
2. Open your terminal
    a. **On Windows** – open Miniforge Prompt
    b. **On macOS** – open Terminal
3. **If on macOS:** Install mono via homebrew
    a. Please see https://brew.sh/ for information about homebrew, how to install it, and why you should use it on macOS
    b. Run the following commands in Terminal:

```
brew update
brew upgrade
brew install mono
```

4. Create new virtual environment by running the following commands in your terminal
    a. One of the advantages of virtual environments is the control of software versions. By naming this environment after the date that you install (or the version of CoreMS you are using), you can always ensure you can go back to a known working and good version.

```
mamba update --all
mamba create -n corems-[current date or version]
```

5. Activate your virtual environment and install Python v3.10 by running the following commands in your terminal

   ```
   mamba activate corems-[current date or version]
   mamba install python=3.10
   ```

6. Install dependencies by running the following command in your terminal

   ```
   mamba install pip git pythonnet psycopg2-binary
   # I suspect we're safe without psycopg2-binary, but I have yet to do
   A/B testing
   ```

7. Deactivate and reactivate the virtual environment to ensure it loads correctly
   ```
   mamba deactivate
   mamba activate corems-[current date or version]
   ```

8. Install CoreMS into your virtual environment from the git repo

   ```
   pip install git+https://github.com/EMSL-Computing/CoreMS
   # This will install the current version of CoreMS, regardless of the
   version you downloaded above
   ```

9. Install Podman, an open-source alternative to Docker
   a. **On Windows:** Follow the instructions from Podman's own documentation (https://podman-desktop.io/docs/installation/windows-install)
      i. Download Podman Desktop for Windows
         1. https://podman.io/
      ii. Allow the Compose and Podman extensions to install and then click "Start Onboarding"
      iii. When it asks to install Podman, select the **Windows Linux Subsystem (WSLv2)** option and the **Install WSLv2 if not present** option
      iv. Let Podman install and restart your computer if prompted
         1. Podman may continue installing after the restart - let it do its thing
         2. Pay attention to the Autostart option - this will allow Podman to open whenever you turn on your computer
      v. Once installed, you should see a Podman Desktop window
   b. **On macOS:** Use homebrew to install Podman

      ```
      brew install podman
      brew install podman-compose
      ```

10. Load CoreMS's database Docker image into Podman
    a. **On Windows**: Open Command Prompt instead of Miniforge Prompt for this section

```
cd /Documents/CoreMS-[current date or version]/
podman machine start
podman-compose up -d
podman container list
```

   b. The last command will confirm everything worked – you should see a single entry including "corems" in the name
   c. **Note:** Occasionally, the database generation can fail but no error will appear. You'll be able to tell by 1) the scripts running *very* fast, and 2) no formulas being assigned. To fix this, you need to remove volumes in podman and then I recommend running a single sample through first. From there, you should be good!

```
podman volume ls # list volumes
podman volume rm [volume name] # remove volume using identifier
```

11. Test your CoreMS installation using the provided Python script and test data
    a. In your terminal, make sure that you have started your CoreMS environment

```
mamba activate corems-[current date or version]
```

   b. Run the "CoreMS_Test.py" script on the provided example data

```
python CoreMS_Test.py -i /Documents/CoreMS-[current date or
version]/tests/tests_data/ftms/ -o /Documents/Output -r
/Documents/CoreMS-[current date or version]/db/Hawkes_neg.ref
```

   c. If you see an output in your chosen output folder, everything worked correctly!

## Running the CoreMS Pipeline

Assuming everything above worked (or that you had CoreMS installed through another mechanism), you are ready to run our pipeline. In principle, there are two steps to run data through CoreMS and we've provided example data for you to test each step.

1. Run the CoreMS_Runner.py script on your data or the example data (Test_Raw).

```
python CoreMS_Runner.py -i /Documents/Input_Data -o
/Documents/Processed_Data -r /Documents/CoreMS-[current date or
version]/db/Hawkes_neg.ref

Required Options:
-i = input folder
-o = output directory
-r = location of reference for calibration; feel free to move the
reference file elsewhere [found in the CoreMS repo by default]

Optional Options:
-t = threshold method used in peak identification when run on a
raw data file; xmls are always run using SN (log)
```

```
-c = calibration value used for selecting points (5)
```

2. Run the CoreMS_MergeProcess.Rmd script in RStudio.
    a. Install all the required packages in R

    ```
    install.packages(c(“devtools”, “tidyverse”, “easycsv”))
    devtools::install_github("EMSL-Computing/ftmsRanalysis")
    ```

    b. Change your input directory to your processed data or our example data (Test_Processed)
        a. Currently, the script is configured to use *easycsv* (an OS agnostic package) to prompt the user to input their desired directory, though sometimes this can fail. In this case, you can set *path_to_dir* to your input directory.
    c. Click “Knit” at the top of the RStudio window

## Troubleshooting

This is not going to be a comprehensive troubleshooting document, but this will include some common errors that we’ve come across during testing. Please let us know if you find any issues!

**Podman Troubleshooting**

- **Windows:** On occasion, Podman will appear running but ultimately any command you try to run will fail (e.g., Step 10d doesn’t work). This appears to be the result of a file not generating correctly.
    - Navigate to **C:\Users\[current user]**
    - If a folder named **.ssh** is not created, please create it
    - Navigate into that folder
    - Create a new text file named **known_hosts** without any extension
    - Run `podman ps -a` in your Command Prompt; if you see an output, this fix has succeeded, if not, please send us a message
- **Windows:** Podman requires that your system has the necessary virtualization subsystems enabled. For most installations of Windows 10/11, these features will be enabled by default (for example, many newer security features require virtualization). Some Windows systems may not be completely configurated and require a couple extra steps.
    - After installing Podman (Step 9), you’ll see a message about virtualization not being enabled. This message will have an accompanying link with instructions - if you follow the instructions, you should be able to enable virtualization and proceed to Step 10.
- **Windows:** In limited situations, communication with the Docker image server is impossible resulting in Step 10d failing with an error message like “Temporary failure in name resolution”.
    - This is likely a result of your corporate security infrastructure blocking communication with the necessary server. Please reach out to IT to see if access to the necessary servers (registry-1.docker.io).
    - If you are still having issues, please let us know.