# Instructions for Processing FTICR Data using CoreMS

**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**

1. Download CoreMS repo from GitHub (either via the website or using git clone)
   1. 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
   1. **On Windows** – open Miniforge Prompt
   2. **On macOS** – open Terminal
3. **If on macOS:** Install mono via homebrew
   1. Please see <https://brew.sh/> for information about homebrew, how to install it, and why you should use it on macOS
   2. Run the following commands in Terminal
      1. brew update
      2. brew upgrade
      3. brew install mono
4. Create new virtual environment by running the following commands in your terminal
   1. mamba update --all
   2. mamba create -n corems-[current date or version]
      1. 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.
5. Activate your virtual environment and install Python v3.10 by running the following commands in your terminal
   1. mamba activate corems-[current date or version]
   2. mamba install python=3.10
6. Install dependencies by running the following command in your terminal
   1. mamba install pip git pythonnet psycopg2-binary
      1. I suspect we're safe without the psycopg2-binary, but I have yet to do A/B testing
7. Deactivate and reactivate the virtual environment to ensure it loads correctly
   1. mamba deactivate
   2. mamba activate corems-[current date or version]
8. Install CoreMS into your virtual environment from the git repo
   1. pip install git+https://github.com/EMSL-Computing/CoreMS
      1. This will download the current version of CoreMS, regardless of the version you downloaded above
9. Install Podman, an open-source alternative to Docker
   1. **On Windows:** Follow the instructions from Podman’s own documentation (<https://podman-desktop.io/docs/installation/windows-install>)
      1. You will also need to ensure you have “Compose” setup (<https://podman-desktop.io/docs/compose/setting-up-compose>)
   2. **On macOS:** Use homebrew to install Podman
      1. brew install podman
      2. brew install podman-compose
10. Load CoreMS’s database Docker image into Podman
    1. **On Windows**: Open Command Prompt instead of Miniforge Prompt for this section
    2. cd /Documents/CoreMS-[current date or version]/
    3. podman machine start
    4. podman-compose up -d
    5. podman container list
       1. This is just to confirm everything is running – you should see a single entry including “corems” in the name
    6. **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!
       1. List the volumes using: podman volume ls
       2. Remove the volume using the identifier: podman volume rm [volume name]
11. Test your CoreMS installation using the provided Python script and test data
    1. In your terminal, make sure that you have started your CoreMS environment
       1. mamba activate corems-[current date or version]
    2. Run the “CoreMS\_Runner.py” script on the provided example data
       1. python CoreMS\_Runner.py /Documents/CoreMS-[current date or version]/tests/tests\_data/ftms/srfa\_neg\_xml\_example.xml /Documents/Output
       2. CoreMS\_Runner.py has two options: input file and output folder
    3. If you see an output in your chosen output folder, everything worked correctly!

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