Specter: linear deconvolution as a new paradigm for targeted analysis of data-independent acquisition mass spectrometry

User Guide

Ryan Peckner

April 9, 2018

Specter is an algorithm for the targeted analysis of data-independent acquisition mass spectrometry experiments. It can analyze data from any instrument type and window acquisition scheme. The required user inputs are a DIA data file in centroided mzML format, a spectral library in blib format, and a mass accuracy parameter, specified in parts-per-million.

The raw output of Specter is a .csv file describing the total ion intensities (= sum of fragment ion intensities) of each precursor in the spectral library at each retention time point of the experiment. A snippet of the typical 'SpecterCoeffs' raw output file looks like this:

Scan index	Retention time (s)	Precursor sequence	Precursor charge	Total ion intensity
10032	268.3763	ETLDASLPSDYLK	2	1,569,034
10032	268.3763	NPAADAGSNNASKK	2	3,112,580
10033	268.4273	IVLVDDSIVR	2	722,175

while the 'SpecterQuants' file contains the identifications and quantifications based on these scan-by-scan total ion intensities:

Precursor sequence	Precursor charge	Quant
ETLDASLPSDYLK	2	148,110,338
NPAADAGSNNASKK	2	32,234,856
IVLVDDSIVR	2	11,768,772

System requirements

Specter requires a computing cluster with Apache Spark v. >= 1.6 (with the PySpark API), Anaconda v. >= 4.2, and Python version >= 2.7.9 (Specter hasn't been tested with Python 3). At least 100 GB of cluster RAM is recommended. R version >=3.2.3 is also required, with the packages 'kza','pracma' and 'moments' installed.

Setup

Download the contents of the GitHub repository to a location accessible to the Spark cluster you'll be using to run Specter. From the command line of the Spark cluster, `cd` to this directory. Now create a conda environment called "SpecterEnv" with the commands

```
$ conda env create -f SpecterEnv.yml
$ source activate SpecterEnv
```

Due to a bug in the package `pymzml`, which Specter uses to parse the mzML mass spec data files, you will need to download the most recent MS bioontology file from `http://data.bioontology.org/ontologies/MS/submissions/116/download?apikey=8b5b7825-538d-40e0-9e9e-5ab9274a9aeb`, rename it as "psi-ms-4.0.1.obo", and manually move it to the `pymzml` obo repository located in `/usr/anaconda/envs/SpecterEnv/lib/python2.7/site-packages/pymzml/obo` (this path may need to be altered depending on where your Anaconda install lives). This issue has been flagged with the `pymzml` developers but has yet to be resolved.

Next, the Spark executor nodes' Python path must be pointed to SpecterEnv so that Spark can access the various packages needed for Specter:

```
$ sudo export PYSPARK_DRIVER_PYTHON=/usr/anaconda/envs/SpecterEnv/bin/python
$ sudo export PYSPARK_PYTHON=/usr/anaconda/envs/SpecterEnv/bin/python
```

The path `/usr/anaconda/envs/SpecterEnv` may vary depending on your settings for Anaconda; the appropriate value can be found by running `conda info -e` at the command line to locate SpecterEnv.

Mass spectrometry data and spectral libraries

All mass spec data has to be provided in centroided mzML format. ProteoWizard's msconvert application is recommended to convert raw mass spec data to mzML and perform centroiding, including for AB SCIEX TripleTOF data in .wiff format. Spectral libraries must be provided in .blib format. Skyline can create a .blib spectral library from most DDA search engine output formats; for a tutorial on how to do this from a pep.xml file, see `http://dia-swath-course.ethz.ch/tutorials2014/Tutorial-3_Library.pdf`. The list of supported search engine output formats for conversion to blib by Skyline can be found at `https://skyline.ms/wiki/home/software/BiblioSpec/page.view?name=BlibBuild`.

Warning: *When building a spectral library from phosphoproteomics data, make sure to filter your search engine output so that only unambiguously localized phosphosites are included (using a localization score such as the A-score, PTM score, or variable modification location score from Spectrum Mill). This is necessary to avoid conflation of spectra from positional isomers.*

Running a Specter job

Once the SpecterEnv conda environment has been activated, the syntax for running Specter is

```
$ ./Specter.sh <mem> <mzMLpath> <blibPath> <index> <StartOrEnd> <numPartitions>
                                                                <instrumentType> <tol>
```

where the bracketed arguments are as follows:

- `mem`: The amount of memory to be provisioned to the Spark driver node.
- `mzMLname`: The full path to the mzML file containing the DIA data to be analyzed, without the mzML extension.
- `blibName`: The name of the blib file containing the spectral library, without the blib extension.
- `index`: The first or last index of the subset of MS2 spectra to be analyzed (see the next argument).
- `StartOrEnd`: Should index be interpreted as the first (StartOrEnd = "start") or last (StartOrEnd = "end") index of the spectra to be analyzed? This is useful for breaking jobs into smaller pieces to respect cluster memory constraints.
- `numPartitions`: The number of partitions Spark will use to parallelize the MS2 spectra. A reasonable starting choice is five times the number of cluster CPUs.
- `instrumentType`: This can be one of 'orbitrap','tof', or 'other'. Use of this argument in the first two cases helps avoid certain known issues with mzMLs coming from data converted from these instrument types.
- `tol`: The instrument mass accuracy, in parts-per-million.

For example, the command

```
$ ./Specter.sh 15g /rpeckner/data/20170501_PhosphoDIARun1 /rpeckner/libs/HumanPhosphoLib
                                                        100000 end 200 orbitrap 10
```

would tell Specter to analyze the first 100,000 MS2 spectra in the Orbitrap DIA experiment file `PhosphoDIARun1.mzML` located in `/rpeckner/data/20170501` using the spectral library `HumanPhosphoLib.blib` located in `/rpeckner/libs/`. The remaining parameters are the mass tolerance of 10 p.p.m., 200 partitions of the associated Spark RDD (the elements of this RDD correspond to the individual MS2 spectra), and 15GB RAM available to the driver node.

Specter will create a subdirectory 'SpecterResults' of the directory containing the mzML, into which the output files will be written. These are the following csv files with the common prefix '<mzMLname>_<blibName>':

- `header`: The header of the mzML file, containing the scan indices, retention times, and precursor m/z window centers.
- `SpecterCoeffs`: The raw Specter coefficients for every library precursor and MS/MS scan.
- `SpecterCoeffsDecoys`: The raw Specter coefficients for both library and decoy precursors and all MS/MS scans.
- `SpecterQuants`: The final identifications and quantifications of library precursors filtered to a 1% false discovery rate.

You can visualize the calculated chromatograms of individual library precursors in R using ggplot:

```
library(ggplot2)
SpecterChromatograms <- read.csv('/rpeckner/data/SpecterResults/20170501_PhosphoDIARun1_
                                                        SpecterCoeffs.csv', header=FALSE)
names(SpecterChromatograms) <- c("Coefficient","Scan","PeptideSeq", "PrecursorCharge",
                                                    "WindowMZCenter","RetentionTime")
ggplot(subset(SpecterChromatograms,
        PeptideSeq == "AEILLTYS[+80.0]K" & PrecursorCharge == 2),
                aes(x=RetentionTime,y=Coefficient)) + geom_point() + geom_line()
```