Data Science Computing Project

Homework 2: Lyman-break Galaxies in R

Please review the background links we discussed in class at
`http://pages.stat.wisc.edu/~jgillett/DSCP/data/lyman`.

This homework is a first step in our search for an undiscovered, gravitationally lensed, high-redshift Lyman-break galaxy. It requires you to search through 100 spectra for those that look like cB58, the target Lyman-break galaxy. Later in the semester, you will use distributed computing tools to extend your search to a much larger collection of spectra.

Over the course of this homework, you will

- become familiar with our first computing tools, Linux and emacs, a powerful editor suitable for remote computing. If you are tempted to solve this homework without these tools, note that we will need them soon when we do remote large-scale computing, so please take this opportunity to learn them.
- practice creating, selecting and applying statistical methods to a research problem for which there isn't a known right answer and for which trial-and-error is required. This is preparation for the project phase of the course in which students do original work.
- become familiar with the Lyman-break galaxy data from the Sloan Digital Sky Survey (SDSS), in which we will do a large search later, doing a small-scale search in this homework.

1 Overview

Here are high-level instructions. Details are in the sections below.

1. Our high-level goal is to find 3 SDSS spectra closest to cB58 by a distance measure of your choice. We will work together to develop measures.
2. When we show our best spectra to Christy (the astronomer), here are some ways she will decide whether it is a Lyman-break galaxy. Look at `http://pages.stat.wisc.edu/~jgillett/DSCP/2/exampleMatch.pdf`, which shows the low-noise target spectrum of cB58 in red and the SDSS noisy spectrum of the same cB58 in blue. Note the presense of:
    - the Lyman-alpha absorption trough, around index 800
    - the Lyman-alpha emission peak just right of the trough, around 900
    - metal abosorption lines right of the peak, in red (and blue) from about 950 to 2600
3. We will consider integer redshifts (shift 0, shift 1, shift 2, ...) of wavelengths in attempting to match cB58.

4. For each of the 3 closest spectra, make a graph showing that spectrum aligned with cB58. If graphing only one spectrum, we would use x=wavelength, y=flux. However, in light of red shifting and the need to graph two aligned spectra at a time, we can abandon wavelength and instead use x = shifted wavelength, or, more simply, just x = index (that is, 1 to the length of the flux vector). Each of your graphs should show the alignment, or lack of alignment, as well as exampleMatch.pdf shows its alignment. Different graph designs are ok if they make deciding "Is this a nice alignment?" easy.

5. Report on what you tried, what worked, and what didn't.

2 Lyman-breaks data: Details

Here are detailed instructions. Please make sure you have read and understood the links at the beginning of this handout before proceeding.

First, we need to get the data.

1. Login to one of our Linux computers and use emacs to create a new directory, `~/2`.

2. From the emacs shell buffer's command line:

   (a) Get to the right directory: `cd ~/2`

   (b) Download the spectra data (21 MB):
   `wget http://www.stat.wisc.edu/~jgillett/DSCP/2/data.tar`

   (c) Unpack the data by running: `tar xvf data.tar` (tar="tape archive", xvf="extract verbose the file ..."). It will extract 100 `.fits` files in a new subdirectory, `data`. Each file contains a noisy SDSS spectrum of an astronomical object.

3. Download `cB58_Lyman_break.fit` (34 KB), a high-quality spectrum from a bigger telescope of the known Lyman-break galaxy, cB58:

   ```
   wget http://www.stat.wisc.edu/~jgillett/DSCP/2/cB58_Lyman_break.fit
   ```

4. Start an R buffer in emacs.

5. (Students using public02/03/04 do not need to do this step.) To download software for reading the spectra, run
   `install.packages("FITSio")` (reply "y" to the personal library questions; select any of the CRAN mirrors).

6. Run `require("FITSio")` to load FITSio into the current R session. (Or use `library("FITSio")` if you prefer that interface for loading a package.) Now you can use the function `readFrameFromFITS()` to read a `.fits` file (`https://en.wikipedia.org/wiki/FITS`) containing a spectrum into a data frame. For example:

- Run `readFrameFromFITS("cB58_Lyman_break.fit")` to see the high-quality, low-noise target data for cB58. This data frame has only two columns, `FLUX` and `LOGLAM`. See the descriptions for `flux` and `loglam` below to understand these columns.
- Run `readFrameFromFITS("data/spec-1353-53083-0579.fits")` to see the noisy SDSS data for cB58.

The data frame for each of the noisy spectra has these columns:

(a) `flux` is light intensity at a given wavelength. It is theoretically nonnegative, but with noise it can be negative.

(b) `loglam` is `log(x=wavelength, base=10)`, so `wavelength = 10^loglam`.

(c) `ivar` ("inverse variance") is $\frac{1}{s_i^2}$, where s_i^2 is an estimated variance of the i^{th} flux. (Each spectrum is constructed from several observations and each `flux[i]` value is an average of several underlying measurements having sample variance s_i^2.)

(d) `and_mask` is 0 for a good observation. We could exclude data with nonzero `and_mask`. (Masks flag problems in the data. `and_mask` indicates whether there is bad data at this (wavelength, flux) pixel in all observations.)

(e) `or_mask` is 0 for a good observation. We can ignore it. (It says there is bad data at this pixel in at least one observation.)

(f) `wdisp` is the resolution of the spectrograph at that wavelength. We can ignore it.

(g) `sky` is the spectrum of the sky, mostly already subtracted out from `flux`. We can ignore it. (Some search algorithms may need to mask out +/-5 pixels around 5577 Angstroms to avoid spurious matches to a strong emission line due to Earth's atmosphere.)

(h) `model` is SDSS's hypothesis of the true spectrum, shifted to the redshift of the object. We can ignore it. (Well, one possible search criterion is to seek each spectrum that is closer to cB58 than to the spectrum's model.)

3 Developing distance measures

To explore and develop ideas for a distance measure, you must participate in the Canvas Discussion at `https://canvas.wisc.edu/courses/450226/discussion_topics/2018753`. Make at least three posts between 2/4/25 and 2/12/25:

- one post proposing or developing an idea
- one asking a helpful question about an idea
- one giving a constructive criticism of an idea

We want to simulate, via Canvas, the brainstorming that goes on in a research-and-development meeting of a motivated team.

We are exploring! Some ideas may work, some may fail. Please don't worry that your idea might get shot down: if you articulate a reasonable thought, probably ten other students (and teachers)

share it, and getting it out advances the conversation. If it gets shot down, fine! Then we can shoot back (kindly) or move to the next idea. Teachers will award a bonus point or two to a few of the best posts, and mistaken ideas may be among these best posts.

Here are examples of ideas from past semesters:

- Find the correlation, r, between cB58 and a spectrum, for all spectra and all redshifts; then use largest r.
- For a target $y_1, ..., y_n$ and spectrum $s_1, ..., s_n$, find difference vectors, $d_i = y_{i+1} - y_i$ and $e_i = s_{i+1} - s_i$, and then find spectra with minimum difference in difference vectors d and e.
- Use a chi-squared statistic to get the distance from cB58.

Implement a search from the Canvas Discussion. A successful search should find at least one persuasive match in its top 3 spectra, namely `data/spec-1353-53083-0579.fits`, since that is the the SDSS noisy spectrum of cB58. Some hints:

- Consider the `and_mask` and `ivar` variables for some distance measures. Note that cB58 has about 2000 points, but the other spectra have about 5000 each. Consider all possible alignments, from 1-2000, 2-2001, 3-2002, through 3001-5000.
- (If your search is successful, please ignore this bullet point.) If you cannot get an idea from the Discussion to work, briefly report on the failure, including your failed distance code. Then you can implement the following simple idea. Mostly forget about astronomy and recognize that the x-coordinate wavelengths are not relevant (because we'll redshift them) and the y-coordinates (flux) are relevant. So we have one vector of y-coordinates for cB58, the known Lyman break galaxy. We have another 100 vectors of y-coordinates (in the data directory). Which of the 100 vectors is most like the cB58 vector?

  For two vectors of y-coordinates A and B, a simple measure is the Euclidean distance$(A, B) = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2}$. (Using absolute values instead of squaring seems equally simple.)

  For each of cB58 and the other spectrum, I first standardized y-coordinates by subtracting their mean and dividing by their standard deviation. I figured subtracting the mean allows aligning spectra of objects with different brightness (height on the y-axis).

  I divided by the standard deviation (that is, I multiplied by a constant) because I wanted to force graphical alignment of cB58 and `spec-1353-53083-0579.fits`. Christy, the astronomer, said two objects with the same luminosity at different distances would have brightness at each wavelength differ by the ratio of their distances squared, so multiplying by a constant allows aligning objects at different distances.

  This approach put `spec-1353-53083-0579.fits` in the top 3.

4 Submission Instructions

Please turn in these four files:

0. There's no file for this item (0), but don't forget to write the three Discussion posts mentioned above.

1. `hw2.R`, which implements your search

   - Please put this code as its first line: `rm(list=ls())` This will ensure that your script does not rely on variables defined in your current R session but not in your script.
   - Include your name and `primaryEmail@wisc.edu` email address.
   - Before turning in `hw2.R`, run `source("hw2.R")` to ensure that your script works. We will run this in a directory that contains `cB58_Lyman_break.fit` and the "data" directory when grading.
   - Your `hw2.R` should read `cB58_Lyman_break.fit` and the 100 `data/spec-....fits` files (from `data.tar`, and one at a time, in a loop). It should write a file called `hw2.csv` whose 100 lines have the format `distance,spectrumID,i`, where
     - `distance` is your measure of the distance from this spectrum to cB58
     - `spectrumID` is the spectrum ID, e.g., `spec-1353-53083-0579.fits`
     - `i` is the shift in the spectrum at which your alignment with cB58 begins

     These lines should be sorted by increasing distance. As an example, a possible line is `1032,spec-1353-53083-0579.fits,519`, which says "the object in spec-1353-53083-0579.fits has a distance 1032 from cB58 when red-shifted by 519."

2. `hw2.csv`, the record of your search and the output of running `source("hw2.R")`

3. `hw2.Rmd`, which is your report.

   - To compile `hw2.Rmd` to `hw2.html` on our linux computers, run R and then
     `require("knitr"); knit2html("hw2.Rmd")`
     or
     `require("markdown"); rmarkdown::render("hw2.Rmd")`
     This may require installing a missing package, e.g.
     `install.packages("knitr")` or
     `install.packages("markdown")`
     Or you may write and compile `hw2.Rmd` in RStudio on your local machine.
   - Include your name and `primaryEmail@wisc.edu` email address.
   - Include a brief introduction describing what you tried, what worked, and what didn't. In particular,
     - Describe the measure you chose and why you chose it.
     - Mention difficulties you had.
   - Your `hw2.Rmd` should read your `hw2.csv` file and make three graphs, showing cB58 aligned with each of the top three spectra from your search in `hw2.R`. Include a legend with each graph identifying cB58 and the other spectrum.
   - Knit your hw2.Rmd to make hw2.html.

4. `hw2.html`, the output of `hw2.Rmd`

To turn in these files only, please make a new directory, e.g., `~/turnIn/2`. Copy the four required files there. Go to its parent directory, `~/turnIn`. In the shell, run `tar cvf 2.tar 2` ("tape archive c=create, v=verbose, f=file to create `2.tar` from the 2 directory") to create the file `2.tar` from your `2` directory. It contains your four files.

Upload `2.tar` as Canvas's `HW2_Lyman100` assignment.

5 Additional Notes and Helpful Tips

Here are some notes regarding this assignment and some common errors:

- Do not use `setwd()`, as it will fail if we do not have your directory when we run `source("HW2.R")`; similarly, use relative paths, not absolute paths, in your code.
- Do not change file and directory names.
- Make sure variables are defined by including this as your first line: `rm(list=ls())`
- Make sure `FITSio` is loaded within your script. Please don't install and load packages we didn't indicate (unless you have permission).
- Here is a test:
  1. Download your submitted `.tar` file.
  2. Extract it in a new directory (`tar xvf 2.tar test_HW2`)
  3. Copy `cB58_Lyman_break.fit` and the `data` directory into your directory.
  4. Use `source("hw2.R")` in R, or `Rscript hw2.R` from the shell, to confirm that it works.