HW2: Lyman-break Galaxy Search

Erin Bogenschutz (ebogenschutz@wisc.edu (mailto:ebogenschutz@wisc.edu))

```
library(FITSio)
```

Introduction

In this assignment, I searched for spectra that resemble the known Lyman-break galaxy cB58.
I used standardized Euclidean distance with sliding windows to account for redshift differences.

What worked:
I standardized the flux of each spectrum (subtract mean, divide by SD) and then computing the Euclidean distance between the target galaxy cB58 and each spectrum. To handle noisy data, I applied a small smoothing window using a moving average before computing the distances in order to find the closest matches. My working code for calculating the distance was using the Euclidean distance:

```
#dist_val <- sum((cb_flux - segment)^2)

#    if (dist_val < best_dist) {
#      best_dist <- dist_val
#      best_shift <- i
#    }
```

To reduce the impact of noise in the spectra, I applied a moving average smoothing to each standardized flux array before performing the sliding-window comparison. The moving average replaces each flux value with the average of its neighboring points within a small window, which suppresses random fluctuations while preserving the broad spectrum.

```
#smooth_flux <- function(x, k = 3) {
#  filter <- rep(1/k, k)
#  stats::filter(x, filter, sides = 2)
#}
```

So then I put it together to loop over every spectrum and apply the smoothing, and the caluculating the Eculdian distance, which then compares the spectrum and eventually will match the spectrum we're looking for as number 1 match.

```
#for (f in spec_files) {
#  spec <- readFrameFromFITS(f)
#  flux <- (spec$flux - mean(spec$flux)) / sd(spec$flux)

# Smooth the flux lightly
#  flux <- as.numeric(smooth_flux(flux, k = 5))

# best_dist <- Inf
#  best_shift <- NA
#  max_shift <- length(flux) - n_cb

#  for (i in 1:max_shift) {
#    segment <- flux[i:(i + n_cb - 1)]

    # Skip if NA due to smoothing at edges
#    if (any(is.na(segment))) next

    # Euclidean distance on full flux
#    dist_val <- sum((cb_flux - segment)^2)

#    if (dist_val < best_dist) {
#      best_dist <- dist_val
#      best_shift <- i
#    }
# }
```

Challenges:
- Looping through all 100 spectra with all possible shifts is computationally slow.
- Choosing an appropriate distance measure was important to make sure alignment works. - For a while I had CB58 landing on in the 4th spot, which then I decided to slightly adjust the way that I dealt with the large peaks and valleys. I then decicided to smooth the data and take the average distances as it went along to find the best shift which resulted in -0579 being in my top 3, and being the best match.

What didn't work:

```
#corr_val <- cor(cb_flux, segment)
#score <- 1 - corr_val   # convert to "distance" (lower is better)

#if (score < best_dist) {
#  best_dist <- score
#  best_shift <- i
```

I also experimented with using correlation instead of Euclidean distance so that matches were based more on spectral shape than absolute differences. While this approach did identify some reasonable alignments(-0579 was number 10), it was less stable in practice and did not consistently rank the expected spectrum among the top matches, so I retained the Euclidean distance method.I then stopped working though this idea and applied a smoothing window.

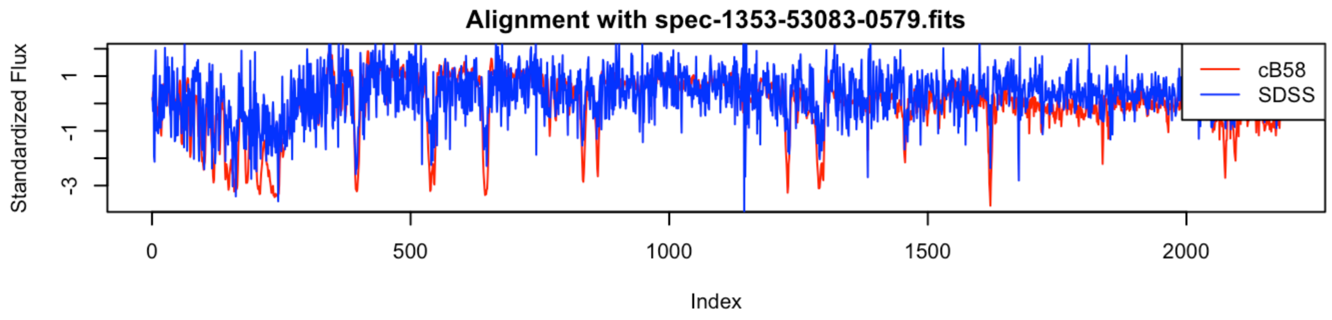
Load results

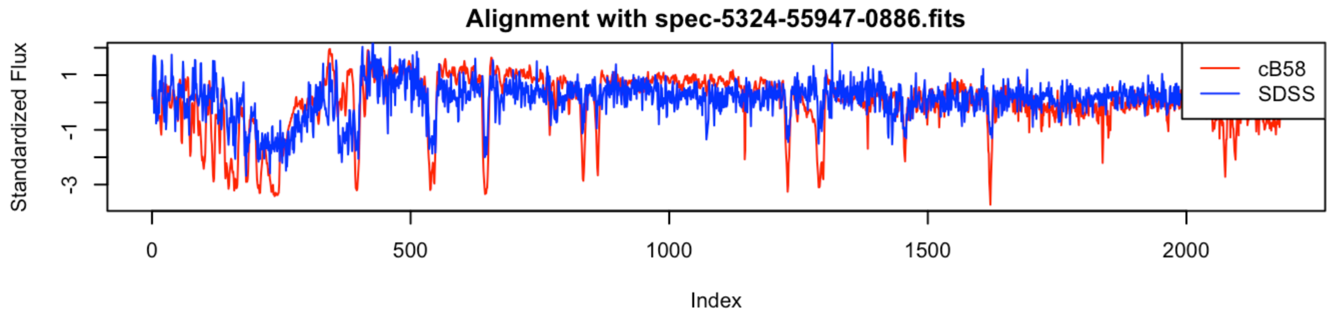
```
results <- read.csv("hw2.csv")
head(results, 3)
```

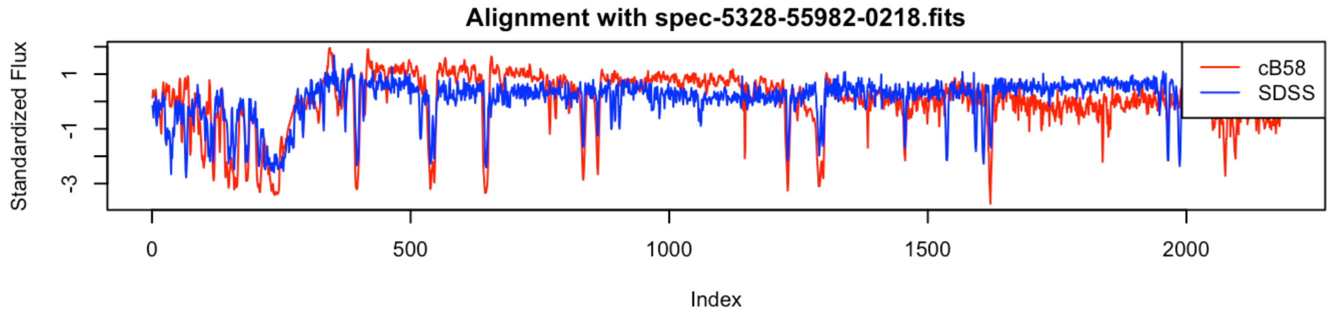
```
##   distance                spectrumID   i
## 1 1183.471 spec-1353-53083-0579.fits 519
## 2 1385.150 spec-5324-55947-0886.fits 828
## 3 1562.801 spec-5328-55982-0218.fits 906
```

Alignment Plots

```
cb58 <- readFrameFromFITS("cB58_Lyman_break.fit")
cb_flux <- (cb58$FLUX - mean(cb58$FLUX)) / sd(cb58$FLUX)
n_cb <- length(cb_flux)

top3 <- results[1:3, ]

par(mfrow=c(3,1), mar=c(4,4,2,1))

for (k in 1:3) {
  spec <- readFrameFromFITS(file.path("data", top3$spectrumID[k]))
  flux <- (spec$flux - mean(spec$flux)) / sd(spec$flux)

  shift <- top3$i[k]
  aligned <- flux[shift:(shift + n_cb - 1)]

  plot(cb_flux, type="l", col="red",
       main = paste("Alignment with", top3$spectrumID[k]),
       xlab="Index", ylab="Standardized Flux")
  lines(aligned, col="blue")
  legend("topright", legend=c("cB58","SDSS"),
         col=c("red","blue"), lty=1)
}
```


Alignment with spec-1353-53083-0579.fits
Standardized Flux
Index
cB58
SDSS
1
-1
-3
0
500
1000
1500
2000


Alignment with spec-5324-55947-0886.fits
Standardized Flux
Index
cB58
SDSS
1
-1
-3
0
500
1000
1500
2000


Alignment with spec-5328-55982-0218.fits
Standardized Flux
Index
cB58
SDSS
1
-1
-3
0
500
1000
1500
2000