# File Parsing & Regular Expressions

Luís Pedro Coelho

Programming for Scientists

March 17, 2009

University of Pittsburgh

Carnegie Mellon

# File Represention

## What's a File?

A sequence of bytes.

(and meta-data).

# File Format Examples: FASTA format

```
> qqq
ACTTTGTTATATATACTATCTGTATTTTC
CTGGGTGAGAGAGTGGTTGAGAGGGGGAA
CCCCCAACCACATTTCCCCACACCCCCTG
ACTTTCCTATATGTCCATTTTTATAATC
> ppp
TTTTTGGTATCTATTTTCCACTCATTCTTTAT
TACCCCAGTCATCACAAAACACACACACAACC
ATTATCTCTAATATATAATTTTACCTTT
```

# Parsing FASTA

```
sequences = []
curseq = ''
for line in file('input.fsa'):
    if line[0] == '>':
        sequences.append(curseq)
    else:
        curseq += line.strip()
sequences.append(curseq)
```

# File Format Examples (II): GenBank

```
LOCUS       SCU49845     5028 bp    DNA                 PL
DEFINITION  Saccharomyces cerevisiae TCP1-beta gene, p
            (AXL2) and Rev7p (REV7) genes, complete cd
ACCESSION   U49845
VERSION     U49845.1  GI:1293613
KEYWORDS    .
SOURCE      Saccharomyces cerevisiae (baker's yeast)
  ORGANISM  Saccharomyces cerevisiae
            Eukaryota; Fungi; Ascomycota; Saccharomycc
            Saccharomycetales; Saccharomycetaceae; Sac
...
ORIGIN
        1 gatcctccat atacaacggt atctccacct caggtttaga
       61 ccgacatgag acagttaggt atcgtcgaga gttacaagct
      121 ctgcatctga agccgctgaa gttctactaa gggtggataa
      181 gaaccgccaa tagacaacat atgtaacata tttaggatat
      241 ggagagtgtg attattataa ttagaaagag aagggaaaaa
```

# Representing Text

## ASCII

- 65: A
- 66: B
- ...
- 48: 0
- 49: 1
- ...
- ...

127 code points taken.

# Two File Format Classes

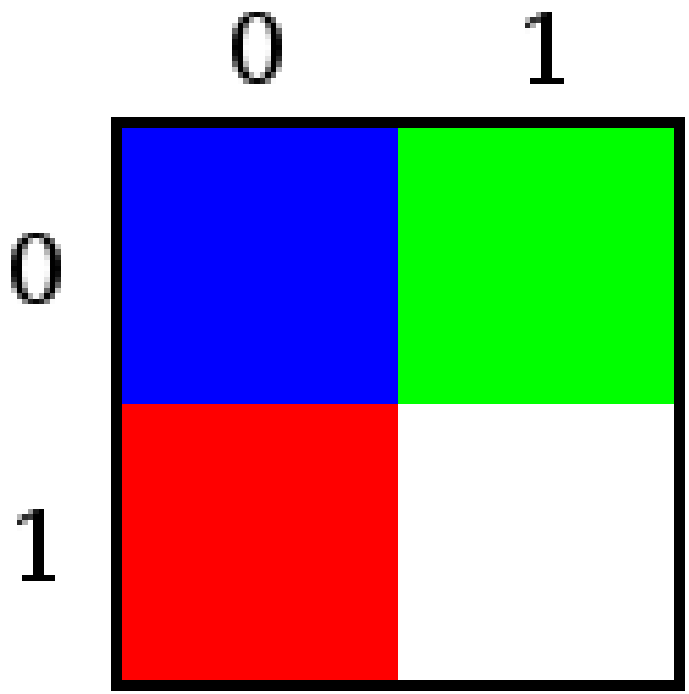
- Text files
- Non-text files (binary files)

| Size | Hex Value | Value | Meaning |
|---|---|---|---|
| 2 | 42 4D | ”BM” | Magic Number (66, 77) |
| 4 | 46 00 00 00 | 70 Bytes | Size of Bitmap |
| 2 | 00 00 | Unused | Application Specific |
| 2 | 00 00 | Unused | Application Specific |
| 4 | 36 00 00 00 | 54 bytes | The offset of data. |
| 4 | 28 00 00 00 | 40 bytes | Size of header. |
| 4 | 02 00 00 00 | 2 pixels | The width in pixels |
| 4 | 02 00 00 00 | 2 pixels | The height in pixels |
| 2 | 01 00 | 1 plane | Number of color planes. |
| 2 | 18 00 | 24 bits | The bits/pixel. |
| 4 | 00 00 00 00 | 0 | No compression used |
| 4 | 10 00 00 00 | 16 bytes | The size of the raw BMP data |
| 4 | 13 0B 00 00 | 2,835 pixels/m | The horizontal resolution |
| 4 | 13 0B 00 00 | 2,835 pixels/m | The vertical resolution |
| 4 | 00 00 00 00 | 0 | Number of colors in the palette |
| 4 | 00 00 00 00 | 0 | Means all colors are important |

| Size | Hex Value | Value | Meaning |
|---|---|---|---|
| 3 | 00 00 FF | 0 0 255 | Red, Pixel (0,1) |
| 3 | FF FF FF | 255 255 255 | White, Pixel (1,1) |
| 2 | 00 00 | 0 | Padding for 4 bytes/row |
| 3 | FF 00 00 | 255 0 0 | Blue, Pixel (0,0) |
| 3 | 00 FF 00 | 0 255 0 | Green, Pixel (1,0) |
| 2 | 00 00 | 0 | Padding for 4 bytes/row |

(Wikipedia)

# Bitmap



# Text vs. Binary

When possible, prefer text formats.
They are simpler.

# Line Endings

- Unix: LF (line feed)
- Windows: CRLF (carriage return, line feed)
- (Old Mac OS: CR)

The extra carriage returns will often show up as ^M in Unix. Some unix text files will show up as a single ultra-long line on Windows.

# What About International Characters?

- Such as á or ç?
- Or $\mu$?
- Or Asian characters?
- Or —?

It's a mess!

# International Character Sets

- Traditional (latin-1,latin-9,latin-15,...)
- Unicode (16-bits, or 32-bits)
- UTF-8

# Unicode

## Unicode

Use 16 bits for (almost) all possible possible characters.

Use 32 bits for all possible characters.

## Byte Order

If you have a 2 byte number, which byte do you write first?

# UTF-8

Emerging standard (at some levels).

# Parsing Files

Let's say you had to parse the following "file":

```
Job id          Name             User      Time Use S Queue
--------------- ---------------- --------- -------- - -----
318695.c0-32    q1025-X.sh       jieyuel    169:29:2 R workq
320137.c0-32    q29199-X.sh      lcoelho    113:10:0 R workq
320139.c0-32    q29212-X.sh      lcoelho    113:29:0 R workq
320141.c0-32    q29226-X.sh      lcoelho    113:24:3 R workq
320143.c0-32    q29240-X.sh      lcoelho    113:09:3 R workq
320145.c0-32    q29254-X.sh      lcoelho    113:58:5 R workq
```

# First Try

```
for line in file('input.txt'):
    if line[0] not in '0123456789':
        continue
    jobid,name,user,time,s,queue = line.split()
    if user != 'lcoelho':
        continue
    jobid = jobid[:-len('.c0-32')] # 318695.c0-32
    runid = name[1:name.find('-')] # q1025-X.sh
    runid = int(runid)
    print "%-12s %-12s %-12%s' % (jobid,runid,time)
```

# Regular Expressions

- You have already seen regular expressions: `cat *.py`
- A regular expression is a “pattern”.

# Basic Regular Expressions

- 'A', 'B', ... match themselves
- '.' matches anything
- '*' means repeat the previous any number of times (including zero)
- '+' means repeat the previous at least once
- '?' means one or zero of the previous
- '[abc]' mean either 'a' or 'b' or 'c'
- ...

# Example

```
import re
line = '320143.c0-32 q29240-X.sh      lcoelho          113:0
if re.match('[0-9]+.c0-32 q[0-9]+-X.sh +lcoelho +[0-9:]+.*'
    print 'matches!'
```

# Get Information Out

```
import re
line = '320143.c0-32 q29240-X.sh      lcoelho          113:0
match = re.match('([0-9]+).c0-32 q([0-9]+)-X.sh +lcoelho +(
if match:
    jobid,runid,time = match.groups()
    print "%-12s %-12s %-12%s' % (jobid,runid,time)
```

# Example

Let's say your files look like this:

- experiment0_t0_0.txt
- experiment0_t0_1.txt
- experiment0_t0_2.txt
- ...
- experiment0_t0_14.txt
- ...
- experiment0_t1_0.txt
- experiment0_t1_1.txt
- ...
- experiment1_t0_0.txt
- ...
- experiment123_t124_125.txt

You could match this with:

```
import re
for file in os.listdir('.'):
    if re.match(r'experiment([0-9]+)_t([0-9]+)_([0-9]+)\.tx
```

# Compiling Patterns

```
import re
for file in os.listdir('.'):
    if re.match(r'experiment([0-9]+)_t([0-9]+)_([0-9]+)\.tx
```

```
import re
pat = re.compile(r'experiment([0-9]+)_t([0-9]+)_([0-9]+)\.t
for file in os.listdir('.'):
    if pat.match(line)
```

# Closing Quote

"Some people, when confronted with a problem, think *I know, I'll use regular expressions*. Now they have two problems."

# Example

## Apache Log

```
128.237.247.73 - - [17/Mar/2009:17:39:27 -0400] "GET /
        6997 "http://coupland.cbi.cmu.edu/pfs/" "Mozil
        Windows NT 6.0; SLCC1; .NET CLR 2.0.50727; Med
128.237.247.73 - - [17/Mar/2009:17:39:42 -0400] "GET /
        - "http://coupland.cbi.cmu.edu/pfs/homeworks.h
        MSIE 7.0; Windows NT 6.0; SLCC1; .NET CLR 2.0.
128.237.247.73 - - [17/Mar/2009:17:39:45 -0400] "GET /
        4998 "http://coupland.cbi.cmu.edu/pfs/index.ht
        MSIE 7.0; Windows NT 6.0; SLCC1; .NET CLR 2.0.
```