# **Regular Expressions and grep**

Regular expressions are defined patterns that often can match many strings. They are similar to the wildcards that we have seen at in previous sections. Grep can use extended regular expressions that extend its pattern matching capabilities. To use these, one uses grep -E or egrep.

Make two files.

dates.txt

11/15/16

1-15-12

3/19/05

3-19-05

3.19.05

4.20.17

12.3.16

12/4/15

superheroes.txt

Batman Bruce Wayne Hero DC

Catwoman Selina Kyle Villain DC

Invisible Woman Susan Storm Richards Hero Marvel

Supergirl Linda Danvers Hero DC

Superman Clark Kent Hero DC

Wonder Woman Diana Prince Hero DC

# [**Basic Metacharacters**](https://ryanstutorials.net/regular-expressions-tutorial/regular-expressions-basics.php#dot)

. (dot) Any character.

[ ] Range. Match any of these characters at a site.

[^ ]Range. Match any character which is not one of these.

\ (backslash) Escape. Remove or add special meaning to a character.

Let’s say we want any month between 3 and 9 followed by a period. The dot in a regular expression means "match anything" except line breaks. If it is in a character class, it means "match a dot."

How does:

egrep '^[3-9][.]' dates

differ from...

egrep '^[3-9].' dates

Try:

egrep '^(1)?[1-9].' dates

egrep '^(1)?[1-9]\.' dates

egrep '3[.]19[.]15' dates #reports 1 date

grep -E '3.19.05' dates #this reports all the 3...dates.

How about using our superheroes file?

egrep '[BC]at' superheroes.txt

'[BC1234567]at' #matches these digits, B, and C then at

You can add to the range

echo '234235' | egrep '[0-9]'

echo '234235' | egrep -i '[a-z]'

[0-9a-zA-Z]

Or...

[0-9a-zA-Z!.\_]

Note the dot is a metacharacter outside the character class.

The dash is a metacharacter in the character class. Otherwise, just a dash.

-[0-9a-zA-Z!.\_]

# [**Multipliers**](https://ryanstutorials.net/regular-expressions-tutorial/regular-expressions-basics.php#multipliers)

**\*** Match zero or more times.

**+** Match one or more times.

**?** Match zero or one times.

**{x}** Match exactly x times.

**{x,y}** Match between x and y times.

**{x,}** Match at least x times.

? and \* always succeed; + must find at least one.

In the dates file

egrep '^1?[1-9].' dates

Imagine you have 10 digits. You could match: [0-9][0-9] etc...

Easier: append {10}. Your regular expression becomes:

[0-9]{10}

echo 5198244120 | egrep [0-9]{10}

echo 5198244120**7** | egrep [0-9]{10} #will this work?

echo 519824412 | egrep [0-9]{10} #will this work?

A{10}

.{10} #matches any 10 character string.

GREAT!{5} matches GREAT!!!!!

Note the quantifier applies to the preceding character.

We can place parentheses if we want to match more than one character.

echo 'hellohellohello' | egrep 'hello{3}'

echo 'hellohellohello' | egrep '(hello){3}'

The number in the squiggly brackets can give a minimum and maximum.

A{10,} 10 or more

A{10,12} #means 10-12

Phone number example.

5551234567

555 1234567

(555) 123 4567

(555) 123-4567

555-123-4567

Tackle easiest first: a number with no spaces

egrep '[0-9][0-9] etc...'

egrep '[0-9]{10}'

'^[0-9]{3}[ -]?[0-9]{3}[ -]?[0-9]{4}$'

Remember \* means none or more.... these all match.

echo 'hello' | egrep 'A\*'; #this matches.

echo 'hello123' | egrep '[a-z]+ \*[1-9]+'

echo 'hello,123' | egrep '[a-z]+.\*[1-9]+'

Matching a metacharacter such as . (dot)

\. "an escaped dot"

e.g. www. uoguelph.ca

www\.uoguelph\.ca

**More UNIX metacharacters**

\w Any word character [a-zA-Z0-9\_]

\s Any whitespace [\ \t\n\r\f]

\W Any non-word [^\w]

\S Any non-whitespace [^\s]

\d A digit (ie. 0 - 9)

\DAnything which is not a digit.

\t A tab

echo 5198244120 | egrep '\w'

echo 'hello' | egrep '\w+'

echo ',,,' | egrep '\w+'

echo 'hello you' | egrep '\w+ +\w+'

hello you

echo 'helloyou' | egrep '\w+ +\w+'

The space meta character is very handy.

echo 'hello you' | egrep '\w+\s+\w+'

echo 'hello,you'| egrep '\w+\W+\w+'

echo ' hello'| egrep '^\S'

echo ' hello'| egrep '^\s'

# [**Anchors and Word Boundaries**](https://ryanstutorials.net/regular-expressions-tutorial/regular-expressions-intermediate.php#anchors)

**^** The beginning of the line.

$The end of the line.

^cat$ #means match beginning of line, c, a, t, end of line.

^$ #means beginning of line matched with end of line. This will match an empty line with nothing in it, even spaces.

^ #means match beginning. every line matches... even empty ones

# [**Alternation**](https://ryanstutorials.net/regular-expressions-tutorial/regular-expressions-advanced.php#alternation)

Match what is on the left, or, what is on the right of the pipe symbol.

'gr[a|e]y' #note here | is just a character within the list you define. It is a metacharacter outside of brackets.

egrep 'Super(man|girl)' superheroes.txt

(A|B){4} matches 4 As OR 4 Bs.

For the phone numbers:

'^([0-9]{3}|\([0-9]{3}]\))[ -]?[0-9]{3}[ -]?[0-9]{4}$'

echo Geoffrey | egrep '(Geo|Je)ff(rey|ery)'

… and others!

Jeffery|Jeffrey

Jeff(rey|ery)

Jeff(re|er)y

Geoffrey

(Geoff|Jeff)(rey|ery)

(Geo|Je)ff(rey|ery)

egrep '([BC]at|Two-Face)' superheroes.txt #this combines concepts.

**Negated character classes**

Sometimes we want to match anything that is not from a specific set of characters. You can do this with the carat character within the character class.

egrep 'S[^u]' superheroes.txt

**Compressing files in Linux**

Compressing means encoding information using fewer bits than the original representation. Compression is useful because it reduces resources required to store and transmit data.

The "**gzip**" command is a common way of compressing files within Linux.

By default, when you compress a file or folder using the "**gzip**" command it will have the same file name as it did before but now it will have the extension ".**gz**".

gzip Gmax.gene.gff3

If you have a file that has already been compressing, you can use the following commands to decompress it. (-d)

gzip –d Gmax.gene.gff3.gz

**Archiving files in Linux**

The **tar** program is used to create, maintain, modify, and extract files that are archived in the tar format. "tar" stands for tape archive. You can use the tar command to combine multiple files into a single archive file for easy storage and/or distribution. A tar archive file contains uncompressed byte streams of the files which it contains.

One often uses tar in conjunction with gzip, to create a compressed archive file. Typically, the compressed form of the archive receives a filename by appending the format-specific compressor suffix to the archive file name. For example, a tar archive *archive.tar*, is named *archive.tar.gz*, when it is compressed by gzip.

Key tar options are:

-c Create a new archive containing the specified items.

-v Tells tar to be verbose (report all files as they are added).

-f Read the archive from or write the archive to the specified file. You need to use the -f option, so tar knows you want to combine a number of files.

-x Extract to disk from the archive

-z Compress the resulting archive with gzip(1).

To combine multiple files into a single archive file (for example, my\_files.tar):

tar -cvf my\_files.tar file1 file2

To combine all the files in a directory into a single archive file (for example, my\_files.tar), use the following command (replace /path/to/my/directory with the absolute path to the directory containing the files you want to combine):

tar -cvf my\_files.tar /path/to/my/directory

To extract the contents of a tar archive file created by tar (for example, my\_files.tar):

tar -xvf my\_files.tar

In most systems, you can use tar in conjunction with the gzip compression utility to combine multiple files into a compressed archive file.

tar -cvzf my\_files.tar.gz file1 file2

tar -cvzf my\_files.tar.gz /path/to/my/directory

Extract the files from **gzip**ped archive **archive.tar.gz.** Here, the **z** tells **tar** that the archive will be compressed with **gzip**. **V** tells **tar** to operate verbosely; **f** tells **tar** that the next argument will be the name of the archive to operate on.

To extract the contents

tar -xzvf archive.tar.gz

**Downloading Files from the Git**

A lot code is available online. For example, GitHub. Github also has a version control system.

git clone <https://DTorkamaneh@bitbucket.org/jerlar73/fast-gbs.git>

git clone <https://bitbucket.org/jerlar73/fast-gbs/src/master/>

git clone <https://github.com/gaow/genetic-analysis-software.git>

git clone https://github.com/delvinso/zillow-nyc-housing-scrape-prediction.git