

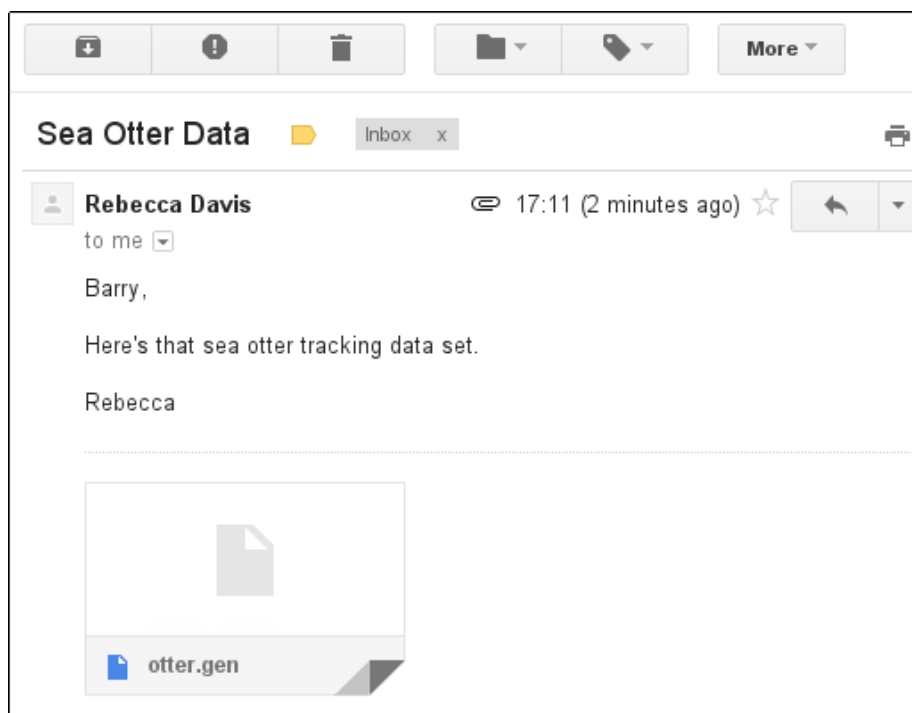
Geostat Summer School



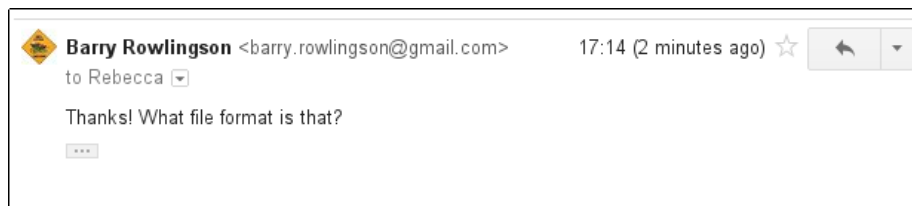
Unix Shell Programming

Your toolkit to an easier life...

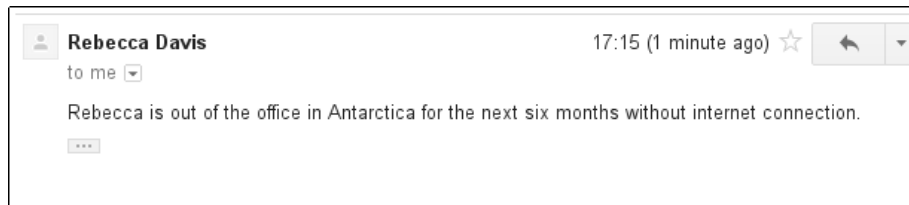
A little story...



A little story...



A little story...



What To Do

Now what?

What's the first thing you do when you get an unknown file?

- Double-click it and hope?
- Load it into Excel?
- Load it into Word?
- Search for the extension on the internet?

basic file ops

- Run file on it

```
$ file otter.gen
otter.gen: ASCII text
```

- Check its size

```
$ ls -l otter.gen
-rw-r--r-- 1 rowlings rowlings 2341 Apr 24 17:24 otter.gen
```

- Human-readable size...

```
$ ls -lh otter.gen
-rw-r--r-- 1 rowlings rowlings 2.3K Apr 24 17:24 otter.gen
```

- Word, line, character count

```
$ wc otter.gen
 130  251 2341 otter.gen
```

See the contents...

- Run head on it

```
$ head otter.gen
1
-145.8953 60.63951
-145.8936 60.63902
-145.8931 60.63774
-145.8941 60.63768
-145.8922 60.63676
-145.8941 60.63651
-145.8976 60.63645
-145.8972 60.63480
-145.8955 60.63450
```

- Run tail on it

```
$ tail -5 otter.gen
-145.8901 60.62465
-145.8901 60.62465
END
END
```

What are those END lines doing there?

Paging Mr File

more (or less)

```
$ more otter.gen
1
-145.8953 60.63951
-145.8936 60.63902
-145.8931 60.63774
...
-145.8962 60.63878
-145.8965 60.64031
-145.8926 60.64104
END
2
-145.9652 60.61914
-145.9634 60.61841
```

```
-145.9599 60.61731
-145.9587 60.61780
-145.9587 60.61878
-145.9599 60.61902
-145.9587 60.61988
-145.9565 60.61939
-145.9577 60.61865
-145.9560 60.61792
-145.9595 60.61731
--More--(34%)
```

Then hit `space` for next page, `q` to quit, `/` to find, `h` for help.

`less` is an alternative pager to `more` with more features.

Search

The oddly-named `grep` command

```
$ grep END otter.gen
END
END
END
END
```

With matching line numbers

```
$ grep -n END otter.gen
33:END
89:END
128:END
129:END
```

Filter output to input

```
$ grep END otter.gen | wc -l
4
```

Stream Filtering

`grep` is great for filtering HUGE files

- Match a pattern:

```
$ grep ^LA1 postcodes.txt > LA1-codes.txt
```

- Chain filters in a pipe:

```
$ grep ^LA1 cases.txt | tr '[a-z]' '[A-Z]' | sed 's/FEMAIL/FEMALE/' | grep FEMALE > cl
```

- Sort and count unique lines:

```
$ sort cases.txt | uniq -c
  1 LA1 Femail
  2 LA1 Female
  1 LA1 Male
  1 LA2 Femail
  1 LA2 Female
  1 LA2 Maile
```

Database joins

Add postcode data to individual data

- Postcode data file:

```
LA1 34243,22310
LA2 77394,12848
LA3 66100,26230
```

- Sorted individual data file:

```
LA1 Femail
LA1 Female
LA1 Female
LA1 Male
LA2 Femail
LA2 Female
LA2 Maile
```

- Use join:

```
$ join sort-cases.txt LA1-codes.txt
LA1 Femail 34243,22310
LA1 Female 34243,22310
LA1 Female 34243,22310
LA1 Male 34243,22310
LA2 Femail 77394,12848
LA2 Female 77394,12848
LA2 Maile 77394,12848
```

Format hypothesis

For each otter track:

- ID number
- lat-long pair
- END mark

Then an END mark

awk

- `awk` is a very useful text-file stream processing language
- read a line at a time, split line into fields
- each line of the language is `pattern {action}`
- if the pattern expression is true, the action is run
- the default pattern is always true, the default action is print this line

Print all lines that don't have two fields:

```
$ awk 'NF!=2' otter.gen
1
END
2
END
3
END
END
```

Print message where second field is greater than some value:

```
$ awk '$2 > 60.64 {print NR," large Y ",$0}' otter.gen
31 large Y -145.8965 60.64031
32 large Y -145.8926 60.64104
98 large Y -145.9250 60.64104
99 large Y -145.9233 60.64153
```

More awk

How many points in each line?

Use the result of `grep -n END otter.gen` to get the line numbers:

```
$ grep -n END otter.gen
33:END
89:END
128:END
129:END
```

Use two awk rules:

- BEGIN {n=0} runs before the first line, and sets n to 0.
- {print \$1 - n; n=\$1 } runs on every line and prints the difference between n and the first field. It then updates n to that value:

```
$ grep -n END otter.gen | awk -F: 'BEGIN {n=0} ; {print $1 - n; n=$1 } '
```

```
33
56
39
1
```

Path Length

Compute the path length for each otter track...

```
$0=="END" && id != "END" {print id,dist}
NF==1 {dist=0; first=1; id=$1}
NF==2 && first==1 {
    x1=$1
    y1=$2
    first = 0
}
NF==2 {x2=$1
    y2=$2
    dist = dist + sqrt((x2-x1)^2 + (y2-y1)^2)
    y1 = y2
    x1 = x2
}
```

Then run:

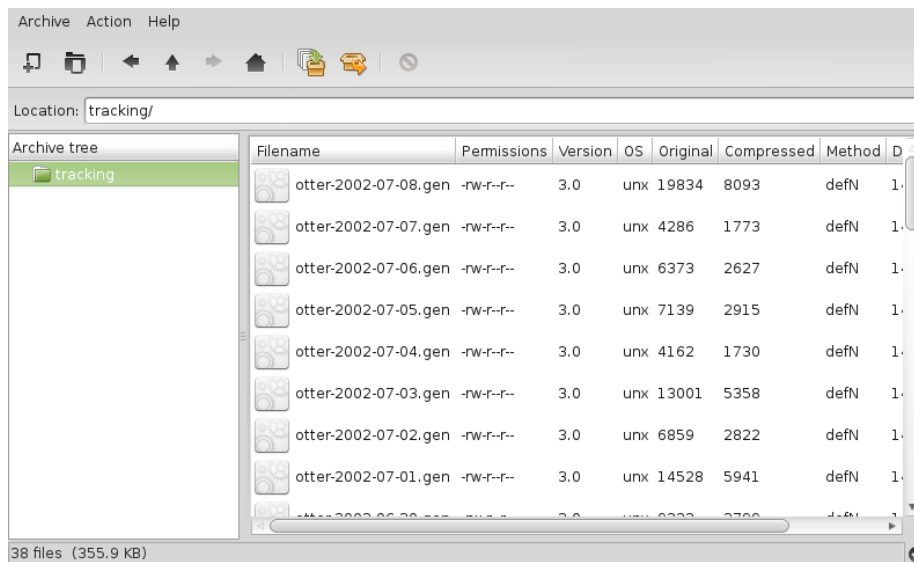
```
$ awk -f pathlength.awk otter.gen
1 0.070923
2 0.102671
3 0.110943
```


You've got mail!

From: rebecca davis

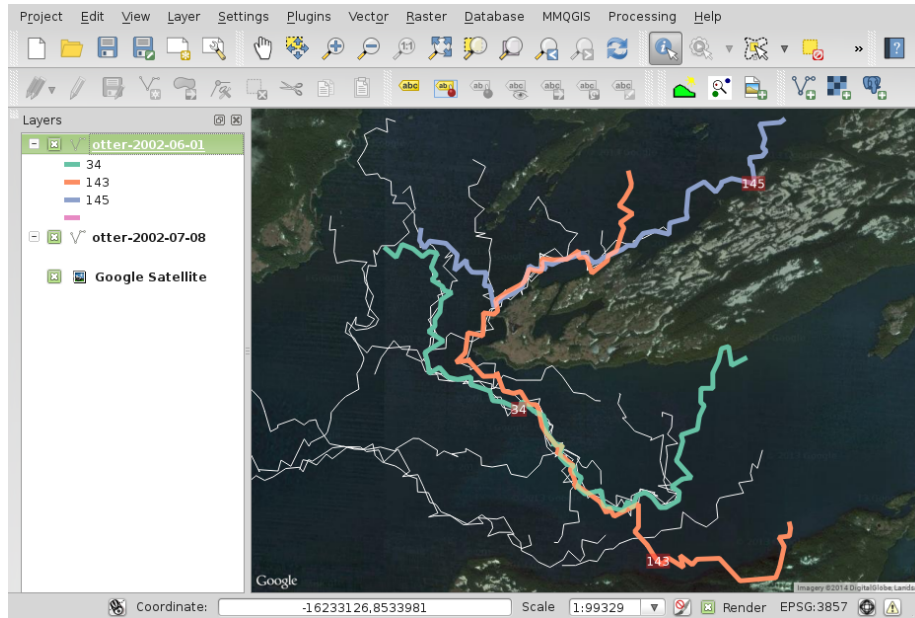
Subject: otter data

Just checking email before getting on plane to South Pole. Data is in Arc/Info Ungenerate format. Here's the full set of 38 track files.



GDAL/OGR

Load straight into QGIS...



Ungenerate Format

Arc/Info Ungenerate Format

- Can be read by OGR/GDAL utilities
- Basic info:

```
$ ogrinfo otter.gen
Had to open data source read-only.
INFO: Open of `otter.gen'
      using driver `ARCGEN' successful.
1: otter (Line String)
```

- Summary info, all layers:

```
$ ogrinfo -so -al otter.gen
Had to open data source read-only.
INFO: Open of `otter.gen'
```

```
using driver `ARCGEN' successful.
```

```
Layer name: otter  
Geometry: Line String  
Feature Count: 3  
Extent: (-145.965700, 60.615840) - (-145.890100, 60.641530)  
Layer SRS WKT:  
(unknown)  
ID: Integer (0.0)
```

ogrinfo

ogrinfo

```
$ ogrinfo -al otter.gen  
Had to open data source read-only.  
INFO: Open of `otter.gen'  
using driver `ARCGEN' successful.
```

```
Layer name: otter  
Geometry: Line String  
Feature Count: 3  
Extent: (-145.965700, 60.615840) - (-145.890100, 60.641530)  
Layer SRS WKT:  
(unknown)  
ID: Integer (0.0)  
OGRFeature(otter):0  
  ID (Integer) = 1  
  LINESTRING (-145.8953 60.63951,-145.8936 60.63902,...1,-145.8926 60.64104)  
OGRFeature(otter):1  
  ID (Integer) = 2  
  LINESTRING (-145.9652 60.61914,-145.9634 60.61841,...  
...
```

ogr2ogr conversion

Command-line geodata conversion

```
$ ogr2ogr -s_srs "+init=epsg:4326" -t_srs "+init=epsg:4326" otter.shp otter.gen
```

```
$ ogrinfo -so -al otter.shp
```

```
INFO: Open of `otter.shp'  
      using driver `ESRI Shapefile' successful.
```

```
Layer name: otter  
Geometry: Line String  
Feature Count: 3  
Extent: (-145.965700, 60.615840) - (-145.890100, 60.641530)  
Layer SRS WKT:  
GEOGCS["GCS_WGS_1984",  
      DATUM["WGS_1984",  
            SPHEROID["WGS_84",6378137,298.257223563]],  
      PRIMEM["Greenwich",0],  
      UNIT["Degree",0.017453292519943295]]  
ID: Integer (10.0)
```

- Want to do:

```
$ ogr2ogr -s_srs "+init=epsg:4326" -t_srs "+init=epsg:4326" otter.shp otter.gen
```

for each “generate” file...

Loops and variables

Command-line loops

Use `echo` to print things:

```
$ echo "Hello World"  
Hello World
```

```
$ echo $HOME  
/home/rowlings
```

Loop over files and print name:

```
$ for f in *.gen ; do echo $f ; done  
otter-2002-06-01.gen  
otter-2002-06-02.gen  
otter-2002-06-03.gen  
otter-2002-06-04.gen  
otter-2002-06-05.gen  
...
```

How do we create the output shapefile name?

Variables and substitutions

Substitutions

```
$ for f in *.gen ; do echo ${f%.gen}.shp ${f:6:10} ; done
otter-2002-06-01.shp 2002-06-01
otter-2002-06-02.shp 2002-06-02
otter-2002-06-03.shp 2002-06-03
otter-2002-06-04.shp 2002-06-04
otter-2002-06-05.shp 2002-06-05
otter-2002-06-06.shp 2002-06-06
otter-2002-06-07.shp 2002-06-07
...
```

Batch Converting

Batch Conversion

- Want to do:

```
$ ogr2ogr -s_srs "+init=epsg:4326" -t_srs "+init=epsg:4326" otter.shp otter.gen
```

for each “generate” file...

- So the loop is:

```
$ for f in *.gen ; do
>   ogr2ogr -s_srs "+init=epsg:4326" -t_srs "+init=epsg:4326" ${f%.gen}.shp $f
>   done
```

- Now we have a shapefile for each input file.

Adding Attributes

We want to:

- Merge all the data into one shapefile
- Add the date to each feature



ogrinfo tricks

- Add a new field:

```
$ ogrinfo otter-2002-06-01.shp -sql "ALTER TABLE otter-2002-06-01 ADD COLUMN day char(15.0)"
```

- Add date to field:

```
$ ogrinfo otter-2002-06-01.shp -dialect SQLite -sql "UPDATE 'otter-2002-06-01' SET day=day"
```

- We now have:

```
$ ogrinfo -geom=NO -al otter-2002-06-01.shp
INFO: Open of `otter-2002-06-01.shp'
      using driver `ESRI Shapefile' successful.
```

```
Layer name: otter-2002-06-01
Geometry: Line String
Feature Count: 3
[etc]
ID: Integer (10.0)
day: String (15.0)
```

```
OGRFeature(otter-2002-06-01):0
  ID (Integer) = 34
  day (String) = 2002-06-01

OGRFeature(otter-2002-06-01):1
  ID (Integer) = 143
  day (String) = 2002-06-01

OGRFeature(otter-2002-06-01):2
  ID (Integer) = 145
  day (String) = 2002-06-01
```

All Together Now

Scripted

```
for genfile in *.gen ; do

  echo "Processing " $genfile

  layer="{genfile%.gen}"
  shapefile="{layer}.shp"
  day="{genfile:6:10}"

  ogr2ogr -s_srs "+init=epsg:4326" -t_srs "+init=epsg:4326" $shapefile $genfile

  ogrinfo $shapefile -sql "ALTER TABLE $layer ADD COLUMN day character(15)"
  ogrinfo $shapefile -dialect SQLite -sql "UPDATE '$layer' SET day='$day'"

done
```

- Now we have all the shapefiles, with the date as a field in each
- Next step: merge all the shapefiles

Merge

Start with a first shapefile, update and append:

```
ogr2ogr alltracks.shp part-1.shp
ogr2ogr -update -append alltracks.shp part-2.shp -nlno alltracks
ogr2ogr -update -append alltracks.shp part-3.shp -nlno alltracks
ogr2ogr -update -append alltracks.shp part-4.shp -nlno alltracks
...
```

Automate this...

Merge

Use append and update

```
for f in *.shp ; do
    if [ -f alltracks.shp ] ; then
        echo merging $f
        ogr2ogr -update -append alltracks.shp $f -nln alltracks
    else
        echo starting with $f
        ogr2ogr alltracks.shp $f
    fi
done
```

Final output

So we get...

```
$ ogrinfo -geom=NO -al alltracks.shp | more
INFO: Open of `alltracks.shp'
      using driver `ESRI Shapefile' successful.

Layer name: alltracks
Geometry: Line String
Feature Count: 214
Extent: (-145.981367, 60.586132) - (-145.817827, 60.649236)
Layer SRS WKT:
GEOGCS["GCS_WGS_1984",
  DATUM["WGS_1984",
    SPHEROID["WGS_84",6378137,298.257223563]],
  PRIMEM["Greenwich",0],
  UNIT["Degree",0.017453292519943295]]
ID: Integer (10.0)
day: String (15.0)
OGRFeature(alltracks):0
  ID (Integer) = 34
  day (String) = 2002-06-01

OGRFeature(alltracks):1
  ID (Integer) = 143
```



```

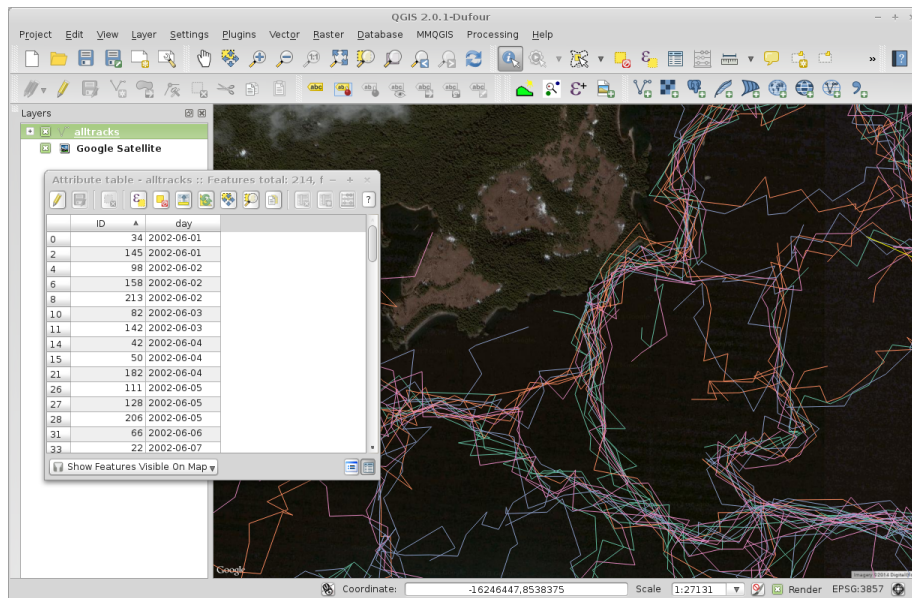
day (String) = 2002-06-01

OGRFeature(alltracks):2
  ID (Integer) = 145
  day (String) = 2002-06-01
...

```

Mapped

Tracks and table



Summary

Summary

- Use the shell
- Get file info
- Summarise file properties
- Extract/Transform
- Command-Line Geospatial Tools