Observations Practical

An “observational tour” from raw GTS data to Bator (no Oulan)”

Eoin Whelan

Irish Meteorological Service

Practical Exercises

- GTS (ecCodes)
- BUFR (ecCodes, Metview)
- Preparation for NWP (Local tools)

Preparation (ecgate)

```
#
# on ecgate: add DA Training PATH, environment variables and modules
#
. /home/ms/spsehlam/hlam/daTraining/user_env.sh
```

GTS 1: Get the data

```
#
# on ecgate
#
$ cd $PERM
$ cp –r /hpc/perm/ms/spsehlam/hlam/daTraining/Day_1 .
$ cd Day_1
$ ls
 data    filters
$
```

GTS: List data

```
$ ls
 data    filters
#
# gts_ls: list GTS file printing header information
#
$ gts_ls data/gts/eidb201901_16_17_wmo.gts | more
#
# gts_ls: use –p option to select keys to print
#
$ gts_ls –p TT=IS data/gts/eidb201901_16_17_wmo.gts | more
```

GTS: exercise 1

```
#
# Print a count of GTS location indicators using gts_ls
# and UNIX commands
#
```

Hint 1: use the –p option

Hint 2: pipe gts_ls output through UNIX commands:

```
   gts_ls … … … | sort | uniq –c | sort -n
```

GTS: exercise 1

```
#
# Print a count of GTS location indicators using gts_ls
# and UNIX commands
#
```

Hint 1: use the –p option

Hint 2: pipe gts_ls output through UNIX commands:

```
   gts_ls … … … | sort | uniq –c | sort –n
```

Solution:

```
~$ gts_ls –p CCCC data/gts/eidb_sample_wmo.gts | sort | uniq –c | sort –n
```

GTS: exercise 2

```
#
# Print a count of GTS reports from your country using gts_ls
# and UNIX commands
#
```

Hint 1: use the –w and -p option

Hint 2: pipe gts_ls output through UNIX commands:

```
   gts_ls … … … | sort | uniq –c | sort -n
```

GTS: exercise 2

```
#
# Print a count of GTS reports from your country using gts_ls
# and UNIX commands
#
```

Hint 1: use the –w and -p option

Hint 2: pipe gts_ls output through UNIX commands:

```
   gts_ls … … … | sort | uniq –c | sort –n
```

Solution:

```
~$ gts_ls –p CCCC –w CCCC=EIDB data/gts/eidb_sample_wmo.gts | uniq –c
```

GTS: exercise 3

```
#
# List the days and hours of GTS reports in the file
# YY – Day of month, GG – Hour of day
#
```

Hint 1: use the -p option

Hint 2: pipe gts_ls output through UNIX commands:

```
   gts_ls … … … | sort -n | uniq –c
```

GTS: exercise 3

```
#
# List the days and hours of GTS reports in the file
# YY – Day of month, GG – Hour of day
#
```

Hint 1: use the -p option

Hint 2: pipe gts_ls output through UNIX commands:

```
  gts_ls … … … | sort -n | uniq –c
```

Solution:

```
~$ gts_ls –p YY,GG data/gts/eidb_sample_wmo.gts | sort –n | uniq
```

GTS: exercise 4

```
#
# Extract all binary GTS reports between 1030 and 1330 on the 17th
# using gts_filter
#
```

Hint 1: List all TT messages beginning with “I” using gts_ls

Hint 2: Refer to bufr_filter examples to draft appropriate filter

- https://confluence.ecmwf.int/display/ECC/bufr_filter
- Compare strings with ***var is “VAL”***

GTS: exercise 4

```
#
# Extract all binary GTS reports between 1030 and 1330 on the 17th
# using gts_filter
#
```

Hint 1: List all TT messages beginning with “I” using gts_ls

```
~$ gts_ls –p TT data/gts/eidb_sample_wmo.gts | sort | uniq | grep I[A-Z]
```

Hint 2: Refer to bufr_filter examples to draft appropriate filter

- https://confluence.ecmwf.int/display/ECC/bufr_filter
- Compare strings with ***var is “VAL”***

```
~$ gts_filter –o ob2019011712.gts gts_bufr.filter data/gts/eidb_sample_wmo.gts
```

GTS: exercise 4

```
#
# Extract all binary GTS reports between 1030 and 1330 on the 17th
# using gts_filter
#
          if ( TT is “IO” || TT is “IS” || TT is “IU” ){
            if ( YY == 17 && GG == 10 && gg > 29 ){
              write;
            }
            if ( YY == 17 && GG == 11 ){
              write;
            }
            if ( YY == 17 && GG == 12 ){
              write;
            }
            if ( YY == 17 && GG == 13 && gg < 31 ){
              write;
            }
          }
~$ gts_filter –o ob2019011712.gts gts_bufr.filter data/gts/eidb_sample_wmo.gts
```

BUFR: exercise 1

```
#
# We should all now have a “GTS” file containing only reports with
# BUFR encoded data
# Use bufr_ls to inspect the contents and list all the (WMO) BUFR
# data categories
#
```

Hint 1: Use the –p option (again!)

Hint 2: Have a look in $ECCODES_DIR/share/definitions/bufr for
inspiration! (grep –i category *.def)

BUFR: exercise 1

```
#
# We should all now have a “GTS” file containing only reports with
# BUFR encoded data
# Use bufr_ls to inspect the contents and list all the (WMO) BUFR
# data categories
#
```

Hint 1: Use the –p option (again!)

Hint 2: Have a look in $ECCODES_DIR/share/definitions/bufr for inspiration! (grep –i category *.def)

```
~$ bufr_ls –p dataCategory ob2019011712.gts | sort –n | uniq
```

BUFR: exercise 2

```
#
# Let’s create a “pure” BUFR file with surface and upper-air data
# i.e no oceanographic (dataCategory=31)
# Use bufr_filter to create this file (from ob2019011712.gts)
#
```

Hint 1: You are BUFR experts now! No more hints!

BUFR: exercise 2

```
#
# Let’s create a “pure” BUFR file with surface and upper-air data
# i.e no oceanographic (dataCategory=31)
# Use bufr_filter to create this file (from ob2019011712.gts)
#
          if ( dataCategory != 31 ){
            write;
          }
```

```
~$ bufr_filter –o ob2019011712.bufr surf_and_ua.filter ob2019011712.gts
```

BUFR: exercise 3

```
#
# Filter BUFR from your favourite “centre” using bufr_filter
#
```

BUFR: exercise 3

```
#
# Filter BUFR from your favourite “centre” using bufr_filter
#
          if ( centre is “eidb” ){
            write;
          }

~$ bufr_filter –o eidb2019011712.bufr eidb_bufr.filter ob2019011712.bufr
```

BUFR: exercise 4

```
#
# Examine your favourite "centre" BUFR using metview
#

~$ metview –e BUFR eidb2019011712.bufr &
```

- Sort BUFR messages by dataCategory (Typ)
- Examine data using “Data Tree”
- View data locations using “Locations”

BUFR: exercise 5

```
#
# Split your “ob” BUFR in to files readable by Bator
# Examine the output using metview
#

synop: dataCategory 0 and internationalDataSubCategory [0-7]
ship:  dataCategory 1 and internationalDataSubCategory [0-7]
buoy:  dataCategory 1 and internationalDataSubCategory [20]
pilot: dataCategory 2 and internationalDataSubCategory [1-3]
temp:  dataCategory 2 and internationalDataSubCategory [4-7]
amdar:dataCategory 4 and internationalDataSubCategory [0]
gpsso: dataCategory 0 and internationalDataSubCategory [14]
```

BUFR: exercise 5

```
#
# Split your “ob” BUFR in to files readable by Bator
# Examine the output using metview
#
if (dataCategory == 0 ){
  if (internationalDataSubCategory >= 0 && internationalDataSubCategory <=7 ) {
    write “split/synop”;
  }
 if (internationalDataSubCategory == 14 ) {
    write “split/gpsso”;
  }
}
if (dataCategory == 1 ){
  if (internationalDataSubCategory >= 0 && internationalDataSubCategory <=7 ) {
    write “split/ship”;
  }
 if (internationalDataSubCategory == 20) {
    write “split/buoy”;
  }
}
```

Local tools: exercise 1

```
#
# Two (metview based) plotting scripts should be available to you
# plotWmoObsConv & plotEcmObsConv

~$ plotWmoObsConv –h

~$ plotWmoObsConv –i ob2019011712.bufr –d 2019011712 –w 90 –t surfland –a GLOB
~$ xv datacover.png

# Explore and enjoy!
```

Local tools: exercise 2

```
#
# The ShuffleBufr tool is used in ALADIN and HIRLAM systems to
# split BUFR (as in Ex. 5) to be read by BATOR for create
# ODB data to be read by the model

~$ ShuffleBufr

~$ mkdir sbSplit
~$ cp ob2019011712.bufr sbSplit
~$ cd sbSplit
~$ ShuffleBufr –i ob2019011712.bufr –s3 –e1

# Explore and enjoy!
```

Local tools: exercise 3

```
#
# The Guessparamcfg tool is available to construct the so-called
# param file required by BATOR

~$ Guessparamcfg

~$ cd sbSplit
~$ Guessparamcfg –i temp
~$ Guessparamcfg –i temp –n 10



# Explore and enjoy!
```