

FILE ARCHIVE KIT (FARK)^{MET.NO, 2018}

INTRODUCTION

The process of generating **verification results** by co-locating **observation** and **model data**, typically requires 200 pieces of information. The FARK system provides a web-interface <http://fark.met.no> for the user to specify this information and organize the regular production of verification result.

Use FARK if you need **regular production** of **verification results** and don't want to spend your time writing scripts.

FARK colocates **NetCDF** model files and **BUFR** observation files, and generates **ASCII** table data and **splus** graphical files (jpg).

The important stuff you need know about the NetCDF and BUFR file formats, is explained in the *appendix*.

THE FARK PROCESS

FARK first generates **indexed lists** of the **NetCDF** model files and **BUFR** observation files, sorted by time.

FARK will then loop over the model files, and for each model file identify the relevant observation files. Model fields are interpolated to relevant observation locations and time. This co-located data is written to a **table file** according to the specifications in a **plotting script**.









Finally, the **plotting script** reads the **table file** and produces the **verification plots**.

Verification results are found under:
`/lustre/storeA/project/nwp/fark`

WEB INTERFACE

All information necessary to generate verification results can be put into the FARK web interface. The web interface contains buttons designed to make it easier for the user to provide this information.

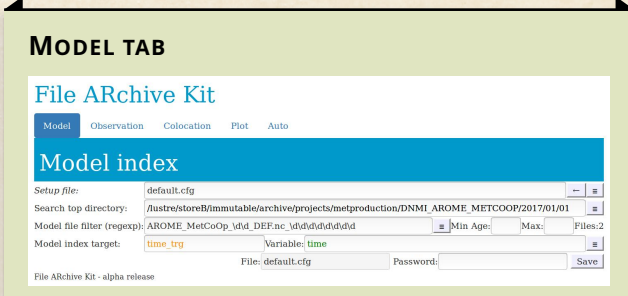
BUTTONS

Button	Description
	show alternatives
	move information
	delete table entry
	add table entry
	test process
	start process
	stop process
	save setup to server

The web interface is further divided into the following tabs: Model, Observation, Colocation, Plot and Auto.

Press the "tab title" to reload data from server.

MODEL TAB



In this tab you specify which NetCDF model files to index. The user must also specify which variable to use to sort the index. When a model file is added to the index, the index variable is pre-scanned to find the range of the index variable. Files overlap if their index variable range overlap.

Use the epoch-time as your index variable.

The index variable must have a small array size. FARK will also pre-scan the range of other variables with small array size (<1000).

The index variable is given a target name. A target name is a 'short name' used to represent the model variable or observation parameter.

OBSERVATION TAB

OBSERVATION TAB

File Archive Kit

Model Observation Colocation Plot Auto

Observation index

Setup file: default.cfg

Search top directory: /faststore/immutable/archive/projects/metproduction/DNM_OBS_BUFR/

Obs file filter (regex): syne_dddhhmmss.ddd.bufr\$

BUFR table path: /metfark/bufr/tables/

Data filter: BUFR type: 0 Sub type: 0 Info:

Observation targets:

Observation target	Position	Description	Info
yy	5	4001 YEAR	-
mm	6	4002 MONTH	-
dd	7	4003 DAY	-
hh	8	4004 HOUR	-
mi	9	4005 MINUTE	-

Observation index target: **obs_time** Expression: sec1970(yy,mm,dd,hh,mi)

File: default.cfg Password: Save

Here you specify which BUFR observation files to index. The user must specify an *index expression*, which must be assigned a **target** name. The user must also define the **observation targets** that are used in the index expression. The observation targets point to specific positions in the BUFR sequence.

COLOCATION TAB

This is where you specify how the model and observation data should be matched. The colocation tab contains several tables.

MODEL TARGETS TABLE

Model target	Model variable (dimension)	Minimum	Maximum
time_trg	time	midnight(-3)	midnight(0)
time_ana	forecast_reference_time	midnight(-3)	midnight(-1)-1
lon	longitude		
lat	latitude		
pres	air_pressure_at_sea_level		
geopot	surface_geopotential		
surpot	surface_potential		
t2m	air_temperature_2m		
rh	relative_humidity_2m		
wind_x	x_wind_10m		
wind_y	y_wind_10m		
precip_acc	precipitation_amount_acc		
precip_acc_m24	precipitation_amount_acc[time_trg-86]		
hybrid	(hybrid2)		

The **model targets** table lists model variables and their target names. The (saved) **model index** target name is listed first. Model targets needed for matching and verification, are added here. The **model target** can also be a **dimension**. In this case enter the dimension name surrounded by brackets instead of a variable name, for instance

(ensemble_member).

MODEL VARIABLE OFFSET

precip_acc	precipitation_amount_acc
precip_acc_m24	precipitation_amount_acc[time_trg-86]

If you want a model variable, say 24 hours earlier than the observation time, you can use an **offset**. The square brackets added after the variable name should contain the name of the model target that should be offset and the offset amount (separated by colon). In this example model target `precip_acc_m24` contains the accumulated precipitation 24 hours before the target `precip_acc`. *Matching rules* should not be defined for model targets that use an offset.

OBSERVATION TARGETS TABLE

Observation target	Position	Description	Minimum	Maximum
yy	5	4001 YEAR		
mm	6	4002 MONTH		
dd	7	4003 DAY		
hh	8	4004 HOUR		
mi	9	4005 MINUTE		
obs_time		sec1970(yy,mm,dd,hh,mi)		
obs_wmo	1	1001 WMO BLOCK N		
obs_id	2	1002 WMO STATION		
obs_sta	3	1015 STATION OR ST		
obs_lat	10	5001 LATITUDE (HIG)		
obs_lon	11	6001 LONGITUDE (HIG)		
obs_hgt	13	7031 HEIGHT OF BAR		
obs_pres	15	10051 PRESSURE REL		
obs_t2m	22	12101 TEMPERATURE/		
obs_d2m	23	12103 DEW-POINT TE/		
obs_rh	24	13003 RELATIVE HUM		
obs_wdir	W	11001 WIND DIRECTIO		
obs_wspsd	W+1	11002 WIND SPEED		
obs_file	oid	Observation file		
obs_bufr	bid	BUFR message i		
obs_loc	lid	Location id		
obs_hybrid		hybrid0-duplicat	1	hybrid0

In the **observation targets** table the user can specify observation targets in the BUFR sequence. The targets already defined in the (saved) **observation index** are listed first. Observation targets needed for matching and verification, are added here.

POSITION VARIABLES

In the example above `obs_wdir` uses a **position variable**, W, instead of a fixed number. The reason for this is that the wind speed happens to appear after a *delayed replicator* in the BUFR sequence. The FARK system will search the BUFR sequence for the specified descriptor,

11001, and assign the corresponding position to the **position variable**, **W**. The **position variable** can be used in the position expressions of later observation targets, as we see in the example with `obs_wspd` with the position expression `W+1`. FARK will repeat the search for position variables until the end of the BUFR message, giving a new location for every match found.

Use **position variables** when you process radiosonde TEMP BUFR messages.

If only the descriptor is specified, the system will search the BUFR sequence for the next entry with the given descriptor.

INTERNAL VARIABLES

Internal variables are indicated as position variables in the position field, without any descriptor.

Position	Description
mid	model file index position
oid	observation file index position
bid	BUFR message number
sid	observation number in message
lid	location number in message

A location is identified using **oid**, **bid** and **lid**.

DUPLICATE LOCATION

If you want to compare the same observation location to several model fields, for instance different ensemble members, you need to duplicate the observation location. In this case you specify the min and max values and not the position nor descriptor (the max value may be a model dimension).

Duplicate locations if you need to process multiple **model ensemble members**.

The observation target `obs_hybrid` in the example above is an example of location duplication. The target `obs_hybrid` takes the value of the duplication index, i.e. `1, 2, 3, ..., hybrid0`.

MATCH RULES TABLE

The **match rules** table specifies how the model targets should match the observation targets.

MATCH RULES TABLE

Model target	Observation target expression	
time_try	obs_time	==
time_ana		==
lon	obs_lon+obs_dlon	==
lat	obs_lat+obs_dlat	==
pres	obs_pres*0.01	==
geopot		==
wind_x		==
wind_y		==
temp		==
rh		==

Model targets with blank observation target expressions are not used for matching. If insufficient matching rules are specified so that FARK can not determine how to interpolate a dimension used by a model target, FARK will average over that dimension (if it is small).

A location is discarded if a match rule has a target that is undefined.

DEFAULT TABLE

The **default** table is only visible if the observation index file has been set to `<none>`. The default table specifies how the model targets should match a set of fixed values.

DEFAULT TABLE

time_try	time_ana	lon	lat	z2m	rh	information	
midnight(0)		10	60				
							+

FILTERS

Although FARK has all the features necessary for a full verification of operational forecast ensemble data using radiosonde data, this is option is probably not realistic from a computer resource perspective.

It is anyway a good strategy to filter out unwanted data as early as possible in the data processing. The plot log in the `Auto` tab contains summaries of how different filters and the quality control removed data.

MODEL TARGET FILTER

Model target	Model variable/(dimension)	Minimum	Maximum
time_trg	time	midnight(-3)	midnight(-1)
time_ana	forecast_reference_time	midnight(-3)-1	midnight(-3)+1

If a model target has **min** and **max** limits that are outside the pre-scanned range, the model file will be skipped. Below is an example of the resulting plot log summary.

PLOT LOG (MODEL FILE FILTER)

```
model_printF Model files: 40
model_printF Model index filter: -3 ( 7.50%)
model_printF 'time_ana' filter: -36 ( 97.38%)
model_printF
```

The observation **min** and **max** filters are applied as the BUFR messages are read from the observation file.

The **observation filter** expression is applied to all locations in a BUFR message at once, and has functions like

```
msgclosest (obs_pres*0.01,1000,500)
```

that selects the locations with the expression, $obs_pres \times 0.01$, closest to the given list, 1000, 500.

The **observation filter** expression can only be based on observation targets.

The model **min** and **max** filters are re-applied immediately after the corresponding model field has been interpolated to the observation location.

The **location filter** expression is applied when all relevant model fields have been interpolated to the observation location. This is the most expensive filter in terms of computer resources.

There is a debug option available for testing the built in filter functions. The debug expression can not accept any targets. Note that “blank” returns zero.

A location is rejected if a filter expression returns 0.

PLOT TAB

PLOT TAB

File Archive Kit

Model Observation Colocation Plot Auto

Plot setup

Setup file: default.cfg

Output table file: /usr/share/Project/mwp/fark/default/table

Output prefix: /usr/share/Project/mwp/fark/default/

Attributes:

Name	Value
n	1
Version	1.0
Action	+
Id	1
Set	set
Colocation file	default.cfg
Legend	MEPS 2.5km
Value1	0

Dataset:


Debug expression: name(precinct(10,60))

File: default.cfg Password: Norway

File Archive Kit - alpha release

This is where you provide information requested by the plotting script. The user chooses plotting script in the Category field.

Verification results are placed on disk according to the Output table file and Output prefix paths. Use YY MM DD HH MI as wildcards in the output paths.

There are two tables in the plot tab. The **Attributes** table allows the user to specify attributes that apply for all the data, for instance titles, units and labels. Some attributes can only have fixed values, indicated by an action button . Special attributes can be used to enumerate columns and other attributes, for instance increasing n to 2 would add another column, value2, in the example above.

A plotting script can compare different colocation datasets. The **Dataset** table assigns every dataset an Id, Colocation file and legend, along with the columns requested by the plotting script.

AUTO TAB

AUTO TAB

Type	Setup file	Schedule	Manual	Last	Info
model	default.cfg			2018-09-21T03:35:25Z	> ok (10s)
model	hurl.cfg			2018-09-09T19:44:23Z	# short
model	scenwf_030d.cfg	daily		2018-09-10T04:20:00Z	> ok (14m5s)
model	scenwf_syno_030d.cfg	daily		2018-09-10T04:04:14Z	> ok (2m13s)
model	scenwf_syno_100d.cfg	daily		2018-09-10T04:06:28Z	> ok (1m31s)
model	scenwf_syno_365d.cfg	weekly		2018-09-10T04:09:25Z	> ok (10m35s)
obs	default.cfg			2018-09-21T03:34:24Z	> ok (21s)
obs	hurl.cfg			2018-09-09T10:44:26Z	# no new data (1s)
obs	starchyno_030d.cfg	daily		2018-09-10T04:14:48Z	> ok (3m17s)
obs	starchyno_100d.cfg	daily		2018-09-10T04:18:05Z	> ok (2m18s)
obs	starchemp_030d.cfg	daily		2018-09-10T04:20:24Z	> ok (8s)
obs	starchyno_365d.cfg	weekly		2018-09-09T05:36:40Z	> ok (15m48s)
plot	default.cfg			2018-09-22T09:26:12Z	> ok (5m43s)
plot	scenwf_003d.cfg			2018-09-18T18:01:35Z	> ok (13m26s)
plot	scenwf_syno_030d.cfg			2018-09-18T18:02:00Z	> ok (2h10m12s)
plot	scenwf_syno_365d.cfg			2018-09-30T07:57:36Z	> ok (19m54m10s)
plot	hurl.cfg			2018-09-09T10:54:23Z	# no data (0s)
plot	syno_003d.cfg			2018-09-09T11:14:49Z	> ok (14m43s)
plot	scenwf_syno_002d.cfg	daily		2018-09-10T04:20:32Z	> ok (59s)
plot	scenwf_syno_002d.cfg	daily		2018-09-10T04:21:32Z	> ok (1m56s)
plot	scenwf_syno_002d.cfg	daily		2018-09-10T04:23:29Z	> ok (10m34s)
plot	scenwf_syno_030d.cfg	daily		2018-09-10T04:34:04Z	# running (2h12m4s)
plot	syno_002d.cfg	daily		2018-09-09T14:44:01Z	> ok (3m21s)
plot	syno_030d.cfg	daily		2018-09-09T14:47:22Z	> ok (5h2m5s)

This is where you tell the computer to actually do some work. Available types of jobs are:

Type	Description
model	maintain model index file,
obs	maintain observation index file,
coloc	<i>debugging for advanced users,</i>
plot	generate verification products.

Each job can be executed manually or according to a schedule.


The Last column has links to log files from the last **successful** and **un-successful** runs.


HOW TO...


...CREATE A MODEL INDEX

Change focus to the “Model” tab.


1. MAKE SURE INDEX DOES NOT EXIST

Enter the name of your new index in the setup file field, for instance `test.cfg`, and press  next to the field.




If your options include `<mkfile>` then the index does not exist. If the index already exists, the index setup will be loaded automatically. In this case press `<rmfile>` to delete the existing index from the server. Remember to use the correct password when changing or deleting an existing index. Finally, press  again and “forget” the index on your client by pressing `<fgfile>`.




2. FIND A SUITABLE INDEX TO COPY

Enter the name of the index you wish to copy in the setup file field. You may press  to navigate. The index setup will be loaded automatically when an existing index is specified.


3. CREATE NEW INDEX SETUP

Enter the name of the new and non-existing index you wish to create in the setup file field. Select the password that has to be provided when changing or deleting the new index. Press  to create the index setup on the server.


4. EDIT THE MODEL INDEX SETUP

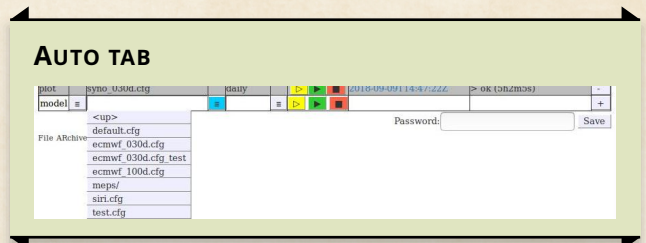
Set the regex file pattern to `.*` if you do not already know the file name pattern. If you press  next to Search top directory and the directory contains files that match your file pattern, FARK will list available file patterns and files in addition to any directories. When you have set the file pattern and search top directory, you may press  next to Model file filter (regex) to select which model file to scan for variables and dimensions. Variables in the scanned file appear as relevant options to other fields (when pressing ). Finally, set the model index variable so that it represents the epoch-time.



The index variable is used to sort the index. The index variable range is used to determine if files overlap.

When you are satisfied, press  to save your settings.

5. CREATE THE MODEL INDEX

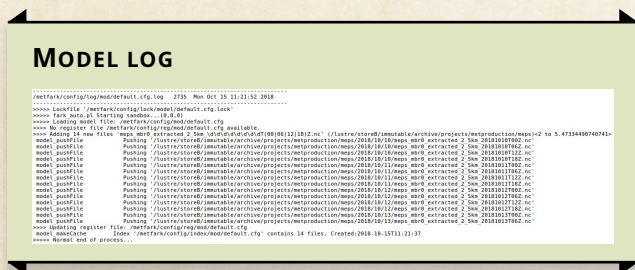
Switch to the Auto tab. The last row in the table allows you to enter new jobs. Select the model type, and press  next to Setup file.



Your new model setup `test.cfg` should now be available as an option, select it. Enter the password and press  to permanently add your job to the table. Finally press  to the right of your model setup file to create the model index itself.

6. CHECK THE LOG

When the job is finished, check the log. You may view the log by pressing the link in the Last column in the Auto tab table.



The log lists the model-files added to the index together with the size and modification date of the index.

The log of a successful run contains
Normal end of process....

Note that several processes flush output to the log, giving it a somewhat “shuffled” appearance.


...CREATE AN OBSERVATION INDEX

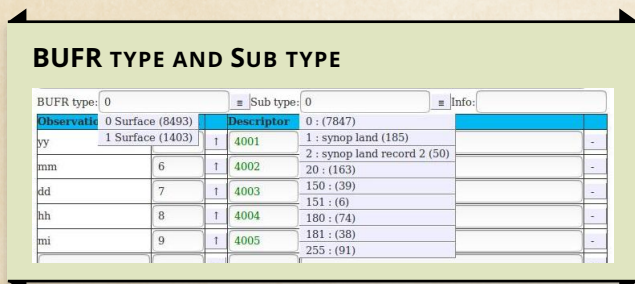
Change focus to the “Observation” tab.

1. CREATE THE SETUP FILE



Creating the observation index setup is similar to creating the model index setup as indicated above.

2. EDIT THE SETUP

After creating your new observation index setup, and set the file search parameters, you may press  next to `Obs file filter(regex)` to select which observation file to scan. Next, specify the `BUFR type` and `Sub type`.



The alternatives are extracted from the scanned file, along with their associated BUFR sequences. Define the targets that you need in your observation index expression

by entering values in the bottom row of the observation targets table, and by pressing the  to the right. You may remove a row by pressing  to the right of the respective row. The values of the removed row are put in the bottom row.

Targests in red are not found in the scanned observation file.

Finally, add an observation index expression that gives the epoch-time of the observation, that can be compared to the model index variable. When you are satisfied, press [Save](#) to save your settings.

2. CREATE THE OBSERVATION INDEX

Add the job in the `Auto` tab and press .

3. CHECK TO LOG

View the log-file by following the link in the Last column.



The log lists the observation-files added to the index together with the size and modification date of the index.

...CREATE A COLOCATION SETUP


Before you can make a colocation setup, you must have created a model and observation index setup, and scanned a corresponding model and observation file. Change focus to the “Colocation” tab.

1. CREATE THE SETUP FILE

Creating the colocation setup is similar to creating the model index setup as indicated above.

2. EDIT THE SETUP

Add the model and observation targets needed to match an observation with the model, for instance `time`, `latitude` and `longitude`.

Add additional targets for the verification parameters, for instance pressure and temperature. Finally define the matching expressions. Model targets with blank observation expressions are not used for matching. When you are satisfied, press  to save your settings.

3. AUTO TAB COLOCATION JOB

It is not recommended to run a colocation-job in the Auto tab since this is an extremely slow process that generates an enormous amount of output.


...CREATE VERIFICATION PLOTS

Before you can make verification plots, you must have created a colocation *setup*, updated the corresponding model and observation indexes and scanned a corresponding set of model and observation files. Change focus to the “Plot” tab.

1. CREATE THE SETUP FILE

Creating the plot setup is similar to creating the model index setup as indicated above. Remember to change the output-fields if you copied the plot setup. Set the attributes and add your colocation datasets. The table file will contain all the data from all the datasets.


The verification script will only verify against observations that are present in all datasets,

so all datasets must overlap before the verification script can produce any output. When you are satisfied, press  to save your plot setup.

2. CREATE VERIFICATION OUTPUT

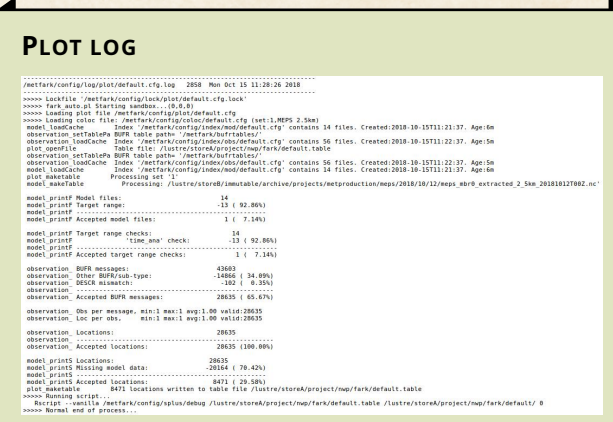
Switch to the Auto tab, add your plot-job to the table.

The model and observation indexes must be available before you run your plot job.

Press  next to your plot job, to run it.

3. CHECK TO LOG

View the log-file by following the link in the Last column.



```

PLOT LOG
-----
/metfark/config/plot/default.cfg.log 2018 Mon Oct 15 11:20:26 2018
.....
Lockfile /metfark/config/lock/plot/default.cfg.lock
..... Fark auto.pl Starting subshell... (0.5.0)
..... Loading plot file /metfark/config/plot/default.cfg
..... Loading color file /metfark/config/color/default.cfg (set:1,MEPS 2.5m)
..... Loading index file /metfark/config/index/default.cfg (contains 14 files, Created:2018-10-15T11:21:37, Age:0s)
model loadCache Index /metfark/config/index/default.cfg (contains 14 files, Created:2018-10-15T11:21:37, Age:0s)
observation testtable BUFR table path: /metfark/bufrtables/ contains 50 files, Created:2018-10-15T11:22:37, Age:0s
plot openFile Table file /lustre/store/project/imp/fark/default.table
observation testtable BUFR table path: /metfark/bufrtables/
observation loadCache Index /metfark/config/index/default.cfg (contains 14 files, Created:2018-10-15T11:21:37, Age:0s)
model loadCache Index /metfark/config/index/default.cfg (contains 14 files, Created:2018-10-15T11:21:37, Age:0s)
plot makeTable Processing set 1
..... Running script: /lustre/store/project/imp/fark/default.table
.....
model print# Model files: 14 ( 92.00%)
model print# Target range: 13 ( 92.00%)
model print# Accepted model files: 1 ( 7.14%)
model print# Target range check: 14
model print# 'time_and' check: 13 ( 92.00%)
model print# Accepted target range check: 1 ( 7.14%)
observation BUFR messages: 43603
observation Other BUFR sub-type: -14800 ( 34.00%)
observation DECC rawdata: -202 ( 0.35%)
observation Accepted BUFR messages: 28635 ( 65.63%)
observation Obs per message, min:1 max:1 avg:1.00 valid:28635
observation Loc per obs, min:1 max:1 avg:1.00 valid:28635
observation Locations: 28635
observation Accepted locations: 28635 (100.00%)
model print# Missing model data: -20144 ( 79.42%)
model print# Accepted locations: 8471 ( 29.58%)
plot makeTable 8471 locations written to table file /lustre/store/project/imp/fark/default.table
..... Running script:
.....
Runscript: /metfark/config/plot/default.cfg /lustre/store/project/imp/fark/default.table /lustre/store/project/imp/fark/default/0
..... Manual end of process.....
  
```

The plot log contains information on the indexes used, how many files they contained and when they were last modified. The log also lists the model files that were processed, and a summary of why locations were removed during the processing. The log is a good place to start looking for missing data.

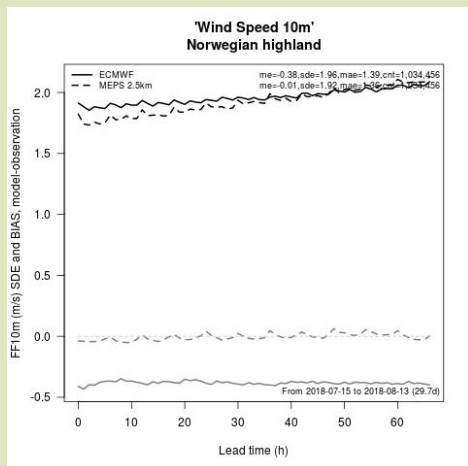
Check if your indexes are up-to-date if your time-dependent filters have removed all the data.

The script command (usually listed at the end of the log), contains the root output directory for the verification results, which you specified in the Plot tab. The next section shows some examples of verification plots that the “synop” verification script typically would place there.

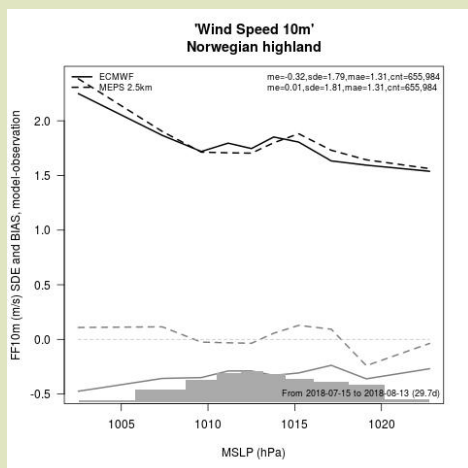
EXAMPLES

Here are some examples of verification plots.

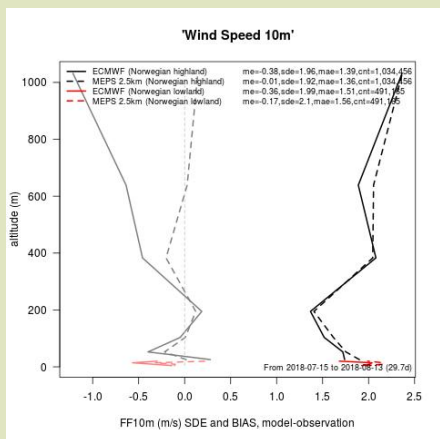
FF10M SCORE



FF10M vs MSLP



FF10M PROFILE



APPENDIX

NETCDF MODEL FILES

The NetCDF model files each contain a set of dimensions and variables, where each variable may have zero or more dimensions, for instance latitude(x,y) where x and y are dimensions. Note that some variables are accumulated, for instance precipitation_amount_acc(...,time). Rain rate is calculated by differentiating this variable with respect to time.

BUFR OBSERVATION FILES

A BUFR observation file may contain many BUFR messages with different BUFR type and sub-type. Each BUFR message may contain many observations of the same type, for instance SYNOP or TEMP. An observation may further contain many locations, for instance a radiosonde TEMP observation may contain data from many different heights in the atmosphere.

BUFR SEQUENCE

BUFR observations with the same BUFR type and sub-type use the same **BUFR sequence**.

BUFR SEQUENCE EXAMPLE

```
1 : 1001 WMO BLOCK NUMBER ~ 2
2 : 1002 WMO STATION NUMBER ~ 981
3 : 1015 STATION OR SITE NAME ~ 1020
4 : 2001 TYPE OF STATION ~ 0
5 : 4001 YEAR ~ 2018
6 : 4002 MONTH ~ 1
7 : 4003 DAY ~ 1
8 : 4004 HOUR ~ 0
9 : 4005 MINUTE ~ 0
10 : 5001 LATITUDE (HIGH ACCURACY) ~ 59.77909
11 : 6001 LONGITUDE (HIGH ACCURACY) ~ 21.37479
12 : 7030 HEIGHT OF STATION GROUND ABOVE MEAN SEA LEVEL (SEE NOTE 3) ~ 6
13 : 7031 HEIGHT OF BAROMETER ABOVE MEAN SEA LEVEL (SEE NOTE 4) ~ 8.3
14 : 10004 PRESSURE ~ 99530
15 : 10051 PRESSURE REDUCED TO MEAN SEA LEVEL ~ 99640
16 : 10061 3-HOUR PRESSURE CHANGE ~ -210
17 : 10063 CHARACTERISTIC OF PRESSURE TENDENCY ~ 6
18 : 10062 24-HOUR PRESSURE CHANGE
19 : 7004 PRESSURE
20 : 10009 GEOPOTENTIAL HEIGHT
21 : 7032 HEIGHT OF SENSOR ABOVE LOCAL GROUND (OR DECK OF MARINE PLATFORM) ~ 2
22 : 12101 TEMPERATURE/DRY-BULB TEMPERATURE ~ 274.45
23 : 12103 DEW-POINT TEMPERATURE ~ 274.05
24 : 13003 RELATIVE HUMIDITY ~ 97
```

The BUFR sequence contains a **position**, **descriptor** and **value** for each parameter in the observation. The **descriptor** is used to identify the observation parameter, for instance pressure is identified by the descriptor 7004.

DELAYED REPLICATOR

DELAYED REPLICATOR	
36 : 20012 CLOUD TYPE	
37 : 31001 DELAYED DESCRIPTOR REPLICATION FACTOR ~ 1	
38 : 8002 VERTICAL SIGNIFICANCE (SURFACE OBSERVATIONS)	
39 : 20011 CLOUD AMOUNT	
40 : 20012 CLOUD TYPE	
41 : 20013 HEIGHT OF BASE OF CLOUD	
42 : 31001 DELAYED DESCRIPTOR REPLICATION FACTOR ~ 0	
43 : 8002 VERTICAL SIGNIFICANCE (SURFACE OBSERVATIONS)	
44 : 20054 TRUE DIRECTION FROM WHICH CLOUDS ARE MOVING	
45 : 8002 VERTICAL SIGNIFICANCE (SURFACE OBSERVATIONS)	
46 : 20054 TRUE DIRECTION FROM WHICH CLOUDS ARE MOVING	

A BUFR sequence may contain a **delayed replicator** (descriptor 31001), which will duplicate a sub-section of the BUFR sequence the specified number of times.

BUFR sequence positions after a **delayed replicator** are not “fixed”.