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 production

FARK first generates **indexed lists** of the **NetCDF** model files and **BUFR** observation files that will be used in the verification.

A basic description of the NetCDF and BUFR file formats is available in the *appendix*.

Next, 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** 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

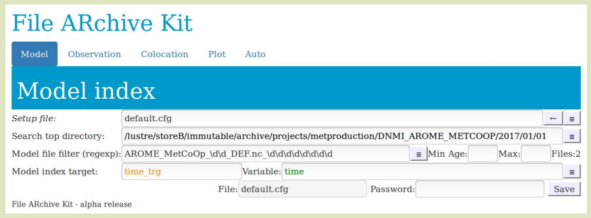
Button	Description
≡	show alternatives
←	move information
-	delete table entry
+	add table entry
▷	test process
▶	start process
■	stop process
Save	save setup to server

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

Model tab



Model index

Setup file: default.cfg
Search top directory: /lustre/storeB/immutable/archive/projects/metproduction/DNMI_AROME_METCOOP/2017/01/01
Model file filter (regexp): AROME_MetCoOp_\d\d_DEF.nc_\d\d\d\d\d\d\d\d Min Age: Max: Files:2
Model index target: time_trg Variable: time
File: default.cfg Password:
File ARchive Kit - alpha release

In this tab you specify which NetCDF model files to index. The user must also specify which variable to use for the index. The index variable is pre-scanned to find the range of the index. Files overlap if their index range overlaps.

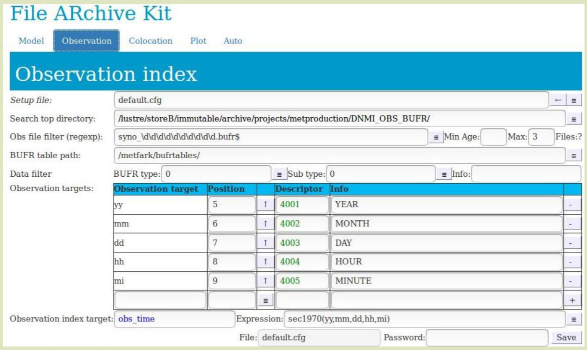
Use the epoch-time as your index-variable.

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

When a model file is being indexed, FARK will also pre-scan the range of other variables with small array size (<1000).

Observation tab

Observation tab



Setup file: default.cfg

Search top directory: /lustre/storeB/immutable/archive/projects/metproduction/DNMI_OBS_BUFR/

Obs file filter (regexp): syno_\d\d\d\d\d\d\d\d\d\d.bufr$ Min Age: Max: 3 Files:?

BUFR table path: /metfark/bufrtables/

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

Observation targets:

Observation target	Position		Descriptor	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

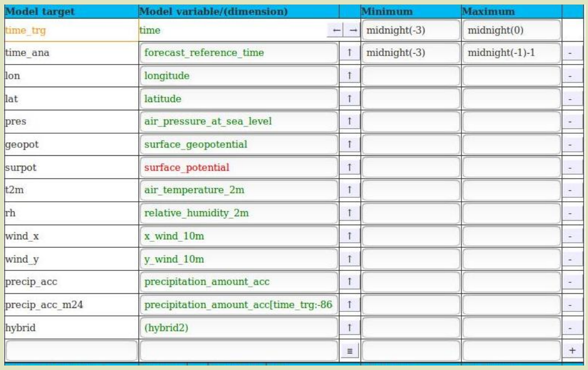
Here you specify which BUFR observation files to index. The user must also specify an *expression* for the index. The index expression is assigned a **target** name. The user must also define the **observation targets** that are used in the index expression. The observation targets point to specific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



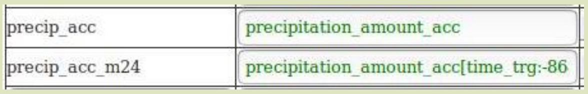
Model target	Model variable/(dimension)		Minimum	Maximum
time_trg	time	← →	midnight(-3)	midnight(0)
time_ana	forecast_reference_time	↑	midnight(-3)	midnight(-1)-1
lon	longitude	↑		
lat	latitude	↑		
pres	air_pressure_at_sea_level	↑		
geopot	surface_geopotential	↑		
surpot	surface_potential	↑		
t2m	air_temperature_2m	↑		
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.

Variables in red are not in the scanned model file.

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

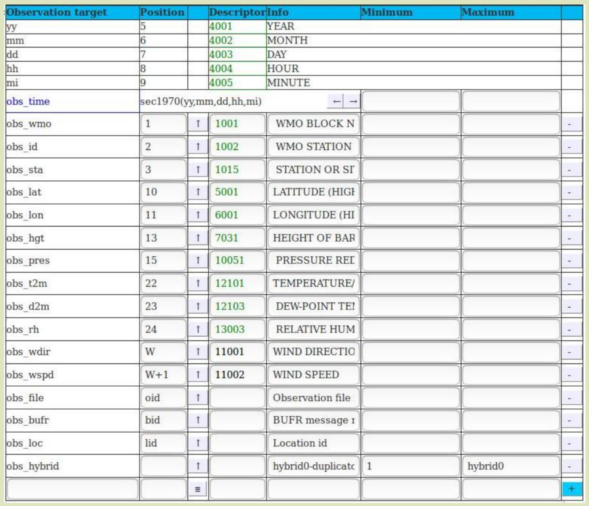


precip_acc	precipitation_amount_acc
precip_acc_m24	precipitation_amount_acc[time_trg:-86

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 offset model targets .

Observation targets table

Observation targets table



Observation target	Position		Descriptor	Info	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 SI		
obs_lat	10	↑	5001	LATITUDE (HIGI		
obs_lon	11	↑	6001	LONGITUDE (HI		
obs_hgt	13	↑	7031	HEIGHT OF BAR		
obs_pres	15	↑	10051	PRESSURE REI		
obs_t2m	22	↑	12101	TEMPERATURE/		
obs_d2m	23	↑	12103	DEW-POINT TEI		
obs_rh	24	↑	13003	RELATIVE HUM		
obs_wdir	W	↑	11001	WIND DIRECTIO		
obs_wspd	W+1	↑	11002	WIND SPEED		
obs_file	oid	↑		Observation file		
obs_bufr	bid	↑		BUFR message ı		
obs_loc	lid	↑		Location id		
obs_hybrid		↑		hybrid0-duplicatc	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. Additional observation targets needed to match the observation with the model fields are added here.

Position variables

In the example above `obs_wdir` uses a **position variable**, `W` instead of a 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).

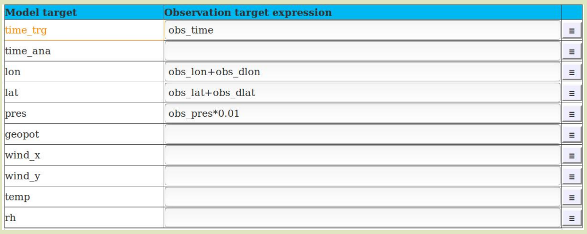
> Duplicate locations if you need to process multiple **model ensemble members**.

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_trg	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		≡

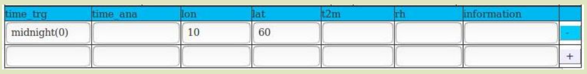
Model targets with blank observation target expressions are not used for matching. If insufficient matching rules are specified so that FARK can not determine how to interpolate a dimension used by a model target, FARK will average over that dimension (if it is small).

> 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 default values.

Default table

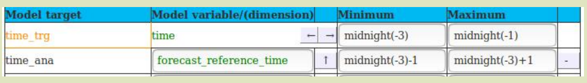


time_trg	time_ana	lon	lat	t2m	rh	information	
midnight(0)		10	60				-
							+

Filters

Co-location takes a lot of computer resources, and it is therefore 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.

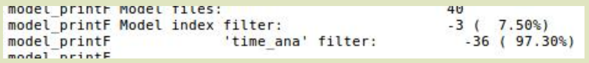
Model target filter



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 immediately be rejected. 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 ( 97.30%)
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*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.

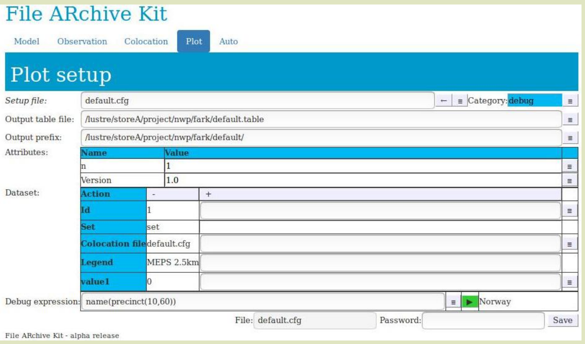
The **location filter** expression is applied when all relevant model fields have been interpolated to the observation location. This is the most expensive filter in terms of computer resources.

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



Setup file:	default.cfg ← ≡ Category: debug ≡
Output table file:	/lustre/storeA/project/nwp/fark/default.table ≡
Output prefix:	/lustre/storeA/project/nwp/fark/default/ ≡

Attributes:

Name	Value
n	1
Version	1.0

Dataset:

Action	-	+
Id	1	
Set	set	
Colocation file	default.cfg	
Legend	MEPS 2.5km	
value1	0	

Debug expression: name(precinct(10,60)) ≡ ▶ Norway

File: default.cfg Password: Save

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.

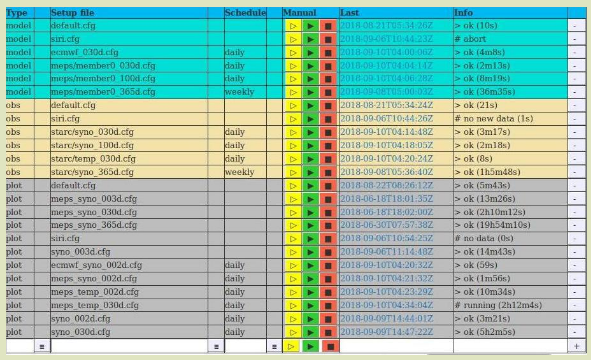
Verification results are placed on disk according to the `Output table file` and `Output prefix` paths. Use `YY MM DD HH MI` as wildcards in the output paths.

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 other attributes and columns.

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		▷ ▶ ■	2018-08-21T05:34:26Z	> ok (10s)	-
model	siri.cfg		▷ ▶ ■	2018-09-06T10:44:23Z	# abort	-
model	ecmwf_030d.cfg	daily	▷ ▶ ■	2018-09-10T04:00:06Z	> ok (4m8s)	-
model	meps/member0_030d.cfg	daily	▷ ▶ ■	2018-09-10T04:04:14Z	> ok (2m13s)	-
model	meps/member0_100d.cfg	daily	▷ ▶ ■	2018-09-10T04:06:28Z	> ok (8m19s)	-
model	meps/member0_365d.cfg	weekly	▷ ▶ ■	2018-09-08T05:00:03Z	> ok (36m35s)	-
obs	default.cfg		▷ ▶ ■	2018-08-21T05:34:24Z	> ok (21s)	-
obs	siri.cfg		▷ ▶ ■	2018-09-06T10:44:26Z	# no new data (1s)	-
obs	starc/syno_030d.cfg	daily	▷ ▶ ■	2018-09-10T04:14:48Z	> ok (3m17s)	-
obs	starc/syno_100d.cfg	daily	▷ ▶ ■	2018-09-10T04:18:05Z	> ok (2m18s)	-
obs	starc/temp_030d.cfg	daily	▷ ▶ ■	2018-09-10T04:20:24Z	> ok (8s)	-
obs	starc/syno_365d.cfg	weekly	▷ ▶ ■	2018-09-08T05:36:40Z	> ok (1h5m48s)	-
plot	default.cfg		▷ ▶ ■	2018-08-22T08:26:12Z	> ok (5m43s)	-
plot	meps_syno_003d.cfg		▷ ▶ ■	2018-06-18T18:01:35Z	> ok (13m26s)	-
plot	meps_syno_030d.cfg		▷ ▶ ■	2018-06-18T18:02:00Z	> ok (2h10m12s)	-
plot	meps_syno_365d.cfg		▷ ▶ ■	2018-06-30T07:57:38Z	> ok (19h54m10s)	-
plot	siri.cfg		▷ ▶ ■	2018-09-06T10:54:25Z	# no data (0s)	-
plot	syno_003d.cfg		▷ ▶ ■	2018-09-06T11:14:48Z	> ok (14m43s)	-
plot	ecmwf_syno_002d.cfg	daily	▷ ▶ ■	2018-09-10T04:20:32Z	> ok (59s)	-
plot	meps_syno_002d.cfg	daily	▷ ▶ ■	2018-09-10T04:21:32Z	> ok (1m56s)	-
plot	meps_temp_002d.cfg	daily	▷ ▶ ■	2018-09-10T04:23:29Z	> ok (10m34s)	-
plot	meps_temp_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	*debugging for advanced users,*
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...

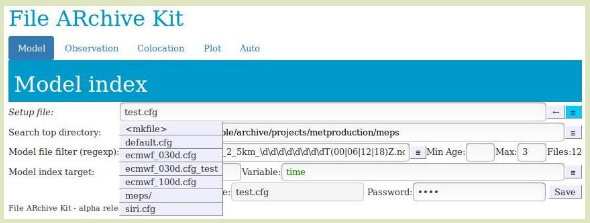
...CREATE A MODEL INDEX

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.

Model tab



File ARchive Kit
Model | Observation | Colocation | Plot | Auto
Model index
Setup file: test.cfg
Search top directory: ble/archive/projects/metproduction/meps
Model file filter (regexp): _2_5km_\d\d\d\d\d\d\d\dT(00|06|12|18)Z.nc Min Age: Max: 3 Files:12
Model index target: Variable: time
e: test.cfg Password: ••••
<mkfile>
default.cfg
ecmwf_030d.cfg
ecmwf_030d.cfg_test
ecmwf_100d.cfg
meps/
siri.cfg
File ARchive Kit - alpha rele

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. 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 provided when changing or deleting the new index. Press Save to create the index setup on the server.

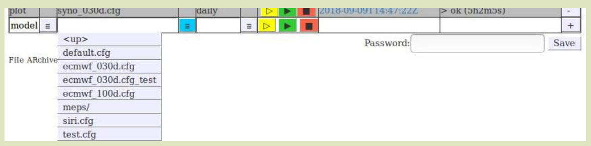
4. Edit the new index setup

Edit the index setup fields. You may press ≡ next to `Model file filter(regexp)` to select which model file to scan. Variables in the scanned file appear as relevant options to other fields (when pressing ≡), for instance `Variable:`. When you are satisfied, press Save 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`.

Auto tab



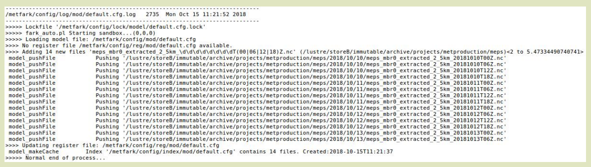
plot | syno_030d.cfg | daily | 2018-09-09T14:47:22Z | > ok (5h2m5s)
model
<up>
default.cfg
ecmwf_030d.cfg
ecmwf_030d.cfg_test
ecmwf_100d.cfg
meps/
siri.cfg
test.cfg
Password:
File ARchive

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.

Model log



```
--------------------------------------------------------------------------------
/metfark/config/log/mod/default.cfg.log   2735  Mon Oct 15 11:21:52 2018
--------------------------------------------------------------------------------
>>>>> Lockfile '/metfark/config/lock/model/default.cfg.lock'
>>>>> fark_auto.pl Starting sandbox...(0,0,0)
>>>>> Loading model file: /metfark/config/mod/default.cfg
>>>> No register file /metfark/config/reg/mod/default.cfg available.
>>>> Adding 14 new files 'meps_mbr0_extracted_2_5km_\d\d\d\d\d\d\d\dT(00|06|12|18)Z.nc' (/lustre/storeB/immutable/archive/projects/metproduction/meps)<2 to 5.47334490740741>
 model_pushFile          Pushing '/lustre/storeB/immutable/archive/projects/metproduction/meps/2018/10/10/meps_mbr0_extracted_2_5km_20181010T00Z.nc'
 model_pushFile          Pushing '/lustre/storeB/immutable/archive/projects/metproduction/meps/2018/10/10/meps_mbr0_extracted_2_5km_20181010T06Z.nc'
 model_pushFile          Pushing '/lustre/storeB/immutable/archive/projects/metproduction/meps/2018/10/10/meps_mbr0_extracted_2_5km_20181010T12Z.nc'
 model_pushFile          Pushing '/lustre/storeB/immutable/archive/projects/metproduction/meps/2018/10/10/meps_mbr0_extracted_2_5km_20181010T18Z.nc'
 model_pushFile          Pushing '/lustre/storeB/immutable/archive/projects/metproduction/meps/2018/10/11/meps_mbr0_extracted_2_5km_20181011T00Z.nc'
 model_pushFile          Pushing '/lustre/storeB/immutable/archive/projects/metproduction/meps/2018/10/11/meps_mbr0_extracted_2_5km_20181011T06Z.nc'
 model_pushFile          Pushing '/lustre/storeB/immutable/archive/projects/metproduction/meps/2018/10/11/meps_mbr0_extracted_2_5km_20181011T12Z.nc'
 model_pushFile          Pushing '/lustre/storeB/immutable/archive/projects/metproduction/meps/2018/10/11/meps_mbr0_extracted_2_5km_20181011T18Z.nc'
 model_pushFile          Pushing '/lustre/storeB/immutable/archive/projects/metproduction/meps/2018/10/12/meps_mbr0_extracted_2_5km_20181012T00Z.nc'
 model_pushFile          Pushing '/lustre/storeB/immutable/archive/projects/metproduction/meps/2018/10/12/meps_mbr0_extracted_2_5km_20181012T06Z.nc'
 model_pushFile          Pushing '/lustre/storeB/immutable/archive/projects/metproduction/meps/2018/10/12/meps_mbr0_extracted_2_5km_20181012T12Z.nc'
 model_pushFile          Pushing '/lustre/storeB/immutable/archive/projects/metproduction/meps/2018/10/12/meps_mbr0_extracted_2_5km_20181012T18Z.nc'
 model_pushFile          Pushing '/lustre/storeB/immutable/archive/projects/metproduction/meps/2018/10/13/meps_mbr0_extracted_2_5km_20181013T00Z.nc'
 model_pushFile          Pushing '/lustre/storeB/immutable/archive/projects/metproduction/meps/2018/10/13/meps_mbr0_extracted_2_5km_20181013T06Z.nc'
>>>> Updating register file: /metfark/config/reg/mod/default.cfg
 model_makeCache       Index '/metfark/config/index/mod/default.cfg' contains 14 files. Created:2018-10-15T11:21:37
>>>>> Normal end of process...
```

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.

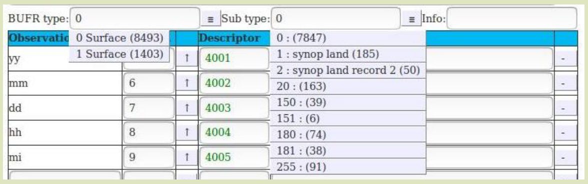
1. CREATE THE SETUP FILE

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, you may press ≡ next to `Obs file filter(regexp)` to select which observation file to scan.

BUFR TYPE AND SUB TYPE



BUFR type: 0				Sub type: 0	Info:	
Observatio	0 Surface (8493)		Descriptor	0 : (7847)		
yy	1 Surface (1403)	↑	4001	1 : synop land (185)		-
				2 : synop land record 2 (50)		
mm	6	↑	4002	20 : (163)		-
				150 : (39)		
dd	7	↑	4003	151 : (6)		-
hh	8	↑	4004	180 : (74)		-
				181 : (38)		
mi	9	↑	4005	255 : (91)		-

The `BUFR type` and `Sub type` 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. When you are satisfied, press Save to save your settings.

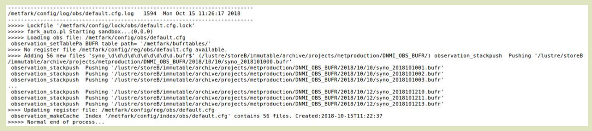
2. CREATE THE OBSERVATION INDEX

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.

OBSERVATION LOG



```
--------------------------------------------------------------------------
/metfark/config/log/obs/default.cfg.log   1594  Mon Oct 15 11:26:17 2018
--------------------------------------------------------------------------
>>>>> Lockfile '/metfark/config/lock/obs/default.cfg.lock'
>>>>> fark_auto.pl Starting sandbox...(0,0,0)
>>>>> Loading obs file: /metfark/config/obs/default.cfg
 observation_setTablePa BUFR table path= '/metfark/bufrtables/'
>>>> No register file /metfark/config/reg/obs/default.cfg available.
>>>> Adding 56 new files 'syno_\d\d\d\d\d\d\d\d\d\d.bufr$' (/lustre/storeB/immutable/archive/projects/metproduction/DNMI_OBS_BUFR/) observation_stackpush  Pushing '/lustre/storeB
/immutable/archive/projects/metproduction/DNMI_OBS_BUFR/2018/10/10/syno_2018101000.bufr'
 observation_stackpush  Pushing '/lustre/storeB/immutable/archive/projects/metproduction/DNMI_OBS_BUFR/2018/10/10/syno_2018101001.bufr'
 observation_stackpush  Pushing '/lustre/storeB/immutable/archive/projects/metproduction/DNMI_OBS_BUFR/2018/10/10/syno_2018101002.bufr'
 observation_stackpush  Pushing '/lustre/storeB/immutable/archive/projects/metproduction/DNMI_OBS_BUFR/2018/10/10/syno_2018101003.bufr'
...
 observation_stackpush  Pushing '/lustre/storeB/immutable/archive/projects/metproduction/DNMI_OBS_BUFR/2018/10/12/syno_2018101210.bufr'
 observation_stackpush  Pushing '/lustre/storeB/immutable/archive/projects/metproduction/DNMI_OBS_BUFR/2018/10/12/syno_2018101211.bufr'
 observation_stackpush  Pushing '/lustre/storeB/immutable/archive/projects/metproduction/DNMI_OBS_BUFR/2018/10/12/syno_2018101213.bufr'
>>>> Updating register file: /metfark/config/reg/obs/default.cfg
 observation_makeCache  Index '/metfark/config/index/obs/default.cfg' contains 56 files. Created:2018-10-15T11:22:37
>>>>> Normal end of process...
```

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. 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. SCAN FILES

Make sure you have scanned a model-file in the `Model tab`, and observation-file in the `Observation tab`, so that model variables and observation BUFR sequences can be listed where relevant.

3. EDIT THE SETUP

Add the model and observation targets needed to match an observation with the model, for instance `time`, `latitude` and `longitude`. Add targets for the verification parameters, for instance `pressure` and `temperature`. When you are satisfied, press Save to save your settings.

Finally define the matching expressions. Model targets with blank observation expressions are not used for matching.

4. 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*. 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. When you are satisfied, press Save to save your plot setup.

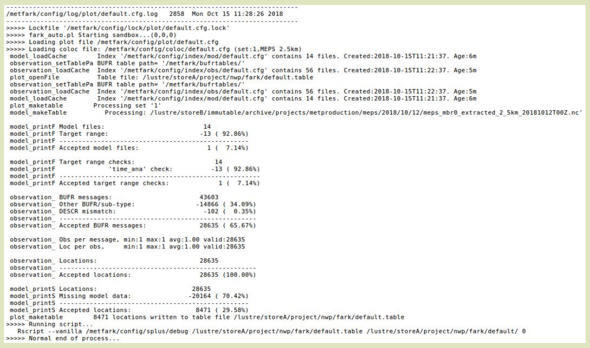
2. Create verification output

Switch to the `Auto tab`, add your plot-job to the table and press ▶ next to your plot job to run the job.

3. Check to log

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

Plot log



```
-----------------------------------------------------------------------------
/metfark/config/log/plot/default.cfg.log   2858  Mon Oct 15 11:28:26 2018
-----------------------------------------------------------------------------
>>>>> Lockfile '/metfark/config/lock/plot/default.cfg.lock'
>>>>> fark_auto.pl Starting sandbox...(0,0,0)
>>>>> Loading plot file /metfark/config/plot/default.cfg
>>>>> Loading coloc file: /metfark/config/coloc/default.cfg (set:1,MEPS 2.5km)
 model_loadCache        Index '/metfark/config/index/mod/default.cfg' contains 14 files. Created:2018-10-15T11:21:37. Age:6m
 observation_setTablePa BUFR table path= '/metfark/bufrtables/'
 observation_loadCache  Index '/metfark/config/index/obs/default.cfg' contains 56 files. Created:2018-10-15T11:22:37. Age:5m
 plot_openFile          Table file: /lustre/storeA/project/nwp/fark/default.table
 observation_setTablePa BUFR table path= '/metfark/bufrtables/'
 observation_loadCache  Index '/metfark/config/index/obs/default.cfg' contains 56 files. Created:2018-10-15T11:22:37. Age:5m
 model_loadCache        Index '/metfark/config/index/mod/default.cfg' contains 14 files. Created:2018-10-15T11:21:37. Age:6m
 plot_maketable        Processing set '1'
 model_makeTable          Processing: /lustre/storeB/immutable/archive/projects/metproduction/meps/2018/10/12/meps_mbr0_extracted_2_5km_20181012T00Z.nc'

 model_printF Model files:                          14
 model_printF Target range:                        -13 ( 92.86%)
 model_printF ---------------------------------------------------
 model_printF Accepted model files:                  1 (  7.14%)

 model_printF Target range checks:                      14
 model_printF              'time_ana' check:           -13 ( 92.86%)
 model_printF ------------------------------------------------------
 model_printF Accepted target range checks:              1 (  7.14%)

 observation_ BUFR messages:                       43603
 observation_ Other BUFR/sub-type:                -14866 ( 34.09%)
 observation_ DESCR mismatch:                       -102 (  0.35%)
 observation_ ----------------------------------------------------
 observation_ Accepted BUFR messages:              28635 ( 65.67%)

 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_ ----------------------------------------------------
 observation_ Accepted locations:                  28635 (100.00%)

 model_printS Locations:                         28635
 model_printS Missing model data:               -20164 ( 70.42%)
 model_printS ---------------------------------------------------
 model_printS Accepted locations:                 8471 ( 29.58%)
 plot_maketable        8471 locations written to table file /lustre/storeA/project/nwp/fark/default.table
>>>>> Running script...
   Rscript --vanilla /metfark/config/splus/debug /lustre/storeA/project/nwp/fark/default.table /lustre/storeA/project/nwp/fark/default/ 0
>>>>> Normal 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 how data was removed during the processing. The log is a good place to start looking for missing data.

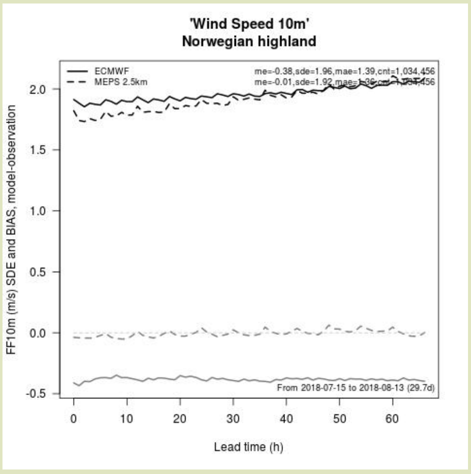
Check if your indexes are up-to-date if your time-dependent filters have removed all data.

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.

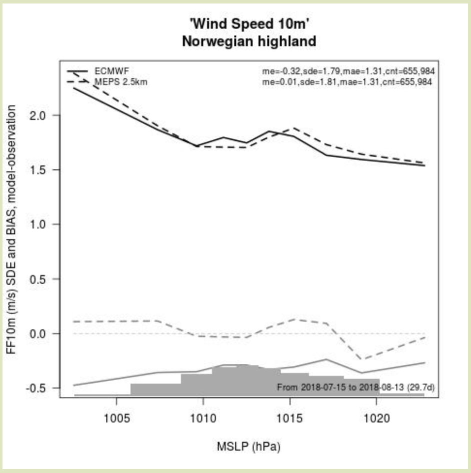
Examples

Here are some examples of verification plots.

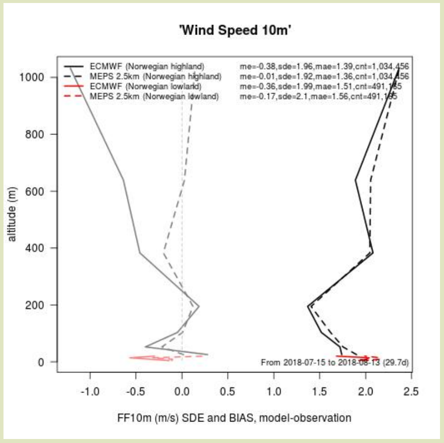
FF10m score



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.

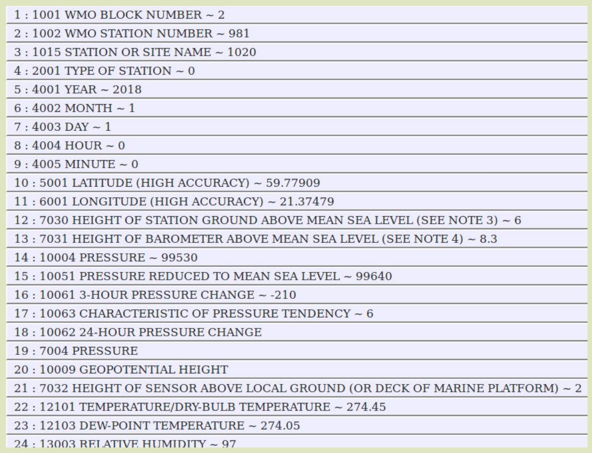
BUFR observation files

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

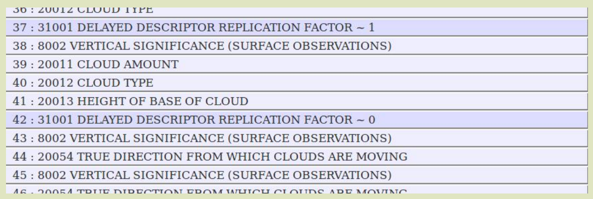


```
1 : 1001 WMO BLOCK NUMBER ~ 2
2 : 1002 WMO STATION NUMBER ~ 981
3 : 1015 STATION OR SITE NAME ~ 1020
4 : 2001 TYPE OF STATION ~ 0
5 : 4001 YEAR ~ 2018
6 : 4002 MONTH ~ 1
7 : 4003 DAY ~ 1
8 : 4004 HOUR ~ 0
9 : 4005 MINUTE ~ 0
10 : 5001 LATITUDE (HIGH ACCURACY) ~ 59.77909
11 : 6001 LONGITUDE (HIGH 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".

Performance

Co-locating large amounts of data takes a lot of resources, and it is quite easy to set up jobs that are not able to run successfully. Generating **table files** does not usually require too much memory but can take much processing time. However the **plotting script** will typically not be able to process a **table file** with more than 10 million entries without running out of memory (on Ares). This corresponds to 20 days of verification for northern Europe. 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.