H-SAF Soil Moisture Week 2019

Exercise - Application for analyzing spatial patterns

In this exercise we will

- Get ASCAT Data Record, ERA5 time-series and RZSM time-series
- Move the data into the correct location
- Read time-series data
- Visualize time-series and map data
- Extract data for a given time step
- Performing some analysis using soil moisture, SWI index and rainfall data

All codes and data are freely available at [c-hydro_github_repository (https://github.com/c-hydro/fp-labs.git)](https://github.com/c-hydro/fp-labs.git) or at [eumetrain_hsaf_github_repository (https://github.com/H-SAF/eumetrain_sm_week_2019.git)](https://github.com/H-SAF/eumetrain_sm_week_2019.git).

Metop ASCAT CDR 12.5 km sampling (2007-2017) H113

1. sm -- soil moisture [%]
2. frozen_probability -- frozen soil probability H %]
3. snow_probability -- snow cover probability [%]
4. time -- time step [daily]

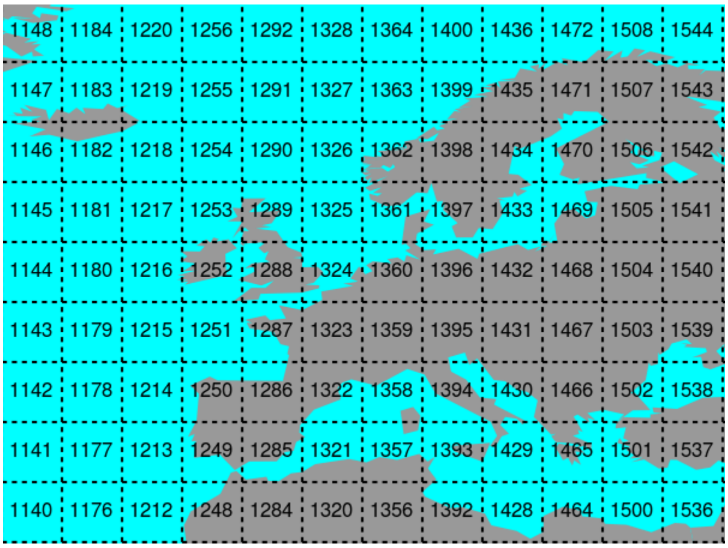
ECMWF ERA5 TimeSeries 30 km grid (2000-)

1. tp -- total precipitation [mm]
2. skt -- skin temperature [K]
3. time -- time step [hourly]

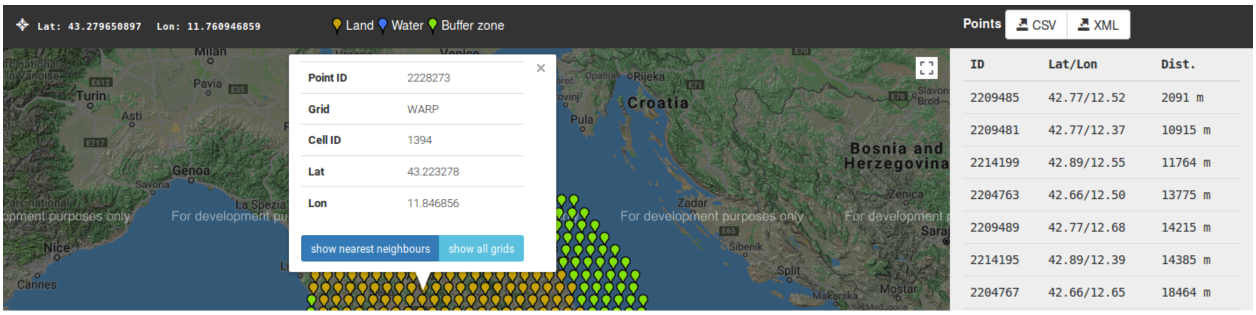
ECMWF RZSM DataRecord 16 km resolution (1992-2014) H27

1. var40 -- root zone soil moisture - level 1 - 0-7 cm
2. var41 -- root zone soil moisture - level 2 - 7-28 cm
3. var42 -- root zone soil moisture - level 3 - 28-100 cm
4. var43 -- root zone soil moisture - level 4 - 100-289 cm
5. time -- time step [daily]

All datasets are converted in time-series format following the WARP5 grid schematization. It stores the time series in 5x5 degree cells. This means there will be 2566 cell files (without reduction to land points) and a file called grid.nc which contains the information about which grid point is stored in which file.



Each cell contains gpis that are id locations identified by longitude and latitude coordinates. Using [grid_point_locator (http://rs.geo.tuwien.ac.at/dv/dgg/)](http://rs.geo.tuwien.ac.at/dv/dgg/) you can retrieve gpis information for selected domain.



Libraries

In [1]:
```
%matplotlib inline

# Libraries
import os
import calendar
import datetime
import warnings
import numpy as np
import pandas as pd

from tqdm import notebook
from os.path import join

from library.cima.domain_utils import get_grid, get_file_shp, get_file_json, create_points_shp
from library.cima.ts_utils import df_time_matching, df_temporal_matching, df_period_selection
from library.cima.ts_dset_reader import dset_init, dset_config, dset_period
from library.cima.map_utils import interpolate_point2map, create_map, create_image

from pytesmo.scaling import get_scaling_function, get_scaling_method_lut

from pytesmo.time_series.filters import exp_filter
from pytesmo.time_series import anomaly
import pytesmo.grid.resample as resample

from library.irpi.indices.drought import ssi, spi

from mpl_toolkits.axes_grid1.axes_divider import make_axes_locatable
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import matplotlib.colors as mc

# Info
print('Libraries loaded!')
# Filter warnings in notebook
warnings.filterwarnings("ignore")
```

```
Libraries loaded!
```

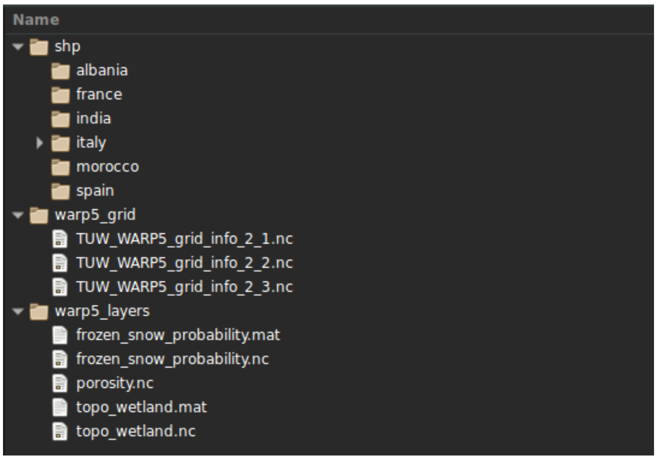
Exercise Configuration

In the configuration part:

- select your basin
- set correct paths of the data
- select time period of datasets
- set thresholds of snow and frozen conditions to filter ASCAT dataset

An example about how to organize static and dynamic data is reported.

- **structure of static data**: shapefile and grid files



- **structure of dynamic data**: ASCAT, ERA5 and RZSM datasets

In [2]:
```
# Domain
domain = 'italy'
exercize = 'ex_spatial_patterns'
file_shp_domain = 'tiber_basin.shp'

# Path(s)
root_path='/home/fabio/Desktop/PyCharm_Workspace/fp-labs/hsaf_event_week_2019/'

data_path_dyn = os.path.join(root_path,'test_data', 'dynamic')
data_path_static = os.path.join(root_path,'test_data', 'static')

tmp_path = os.path.join(root_path, 'test_outcome', 'tmp', exercize)
img_path = os.path.join(root_path, 'test_outcome', 'img', exercize)
ancillary_path = os.path.join(root_path, 'test_outcome', 'ancillary', exercize)

ascat_path_ts = os.path.join(data_path_dyn, 'ascat_dr', domain)
ascat_path_grid = os.path.join(data_path_static, 'warp5_grid')
ascat_path_layers = os.path.join(data_path_static, 'warp5_layers')
ascat_path_tmp = os.path.join(tmp_path, 'ascat')

era5_path_ts = os.path.join(data_path_dyn, 'era5_ts', domain)
era5_path_grid = os.path.join(data_path_dyn, 'era5_ts', domain)
era5_path_tmp = os.path.join(tmp_path, 'era5')

rzsm_path_ts = os.path.join(data_path_dyn, 'rzsm_ts', domain)
rzsm_path_grid = os.path.join(data_path_dyn, 'rzsm_ts', domain)
rzsm_path_tmp = os.path.join(tmp_path, 'rzsm')

domain_path_layer = os.path.join(data_path_static, 'shp', domain)
exercize_path_img = os.path.join(img_path)

# Parameter
ascat_mask_frozen_prob_threshold = 100 # if mask value is greater than threshold the value is discarded
ascat_mask_snow_prob_threshold = 100   # if mask value is greater than threshold the value is discarded

time_start = "2007-01-01" # format "%Y-%m-%d"
time_end = "2014-12-31"   # "format %Y-%m-%d"

temporal_matching = 24
temporal_drop_duplicates = False

max_dist = 35000

# Create img path
if not os.path.exists(img_path):
    os.makedirs(img_path)
# Create ancillary path
if not os.path.exists(ancillary_path):
    os.makedirs(ancillary_path)
# Create tmp path
if not os.path.exists(tmp_path):
    os.makedirs(tmp_path)
 # Create tmp path for ascat
if not os.path.exists(ascat_path_tmp):
    os.makedirs(ascat_path_tmp)
# Create tmp path for era5
if not os.path.exists(era5_path_tmp):
    os.makedirs(era5_path_tmp)
 # Create tmp path for rzsm
if not os.path.exists(rzsm_path_tmp):
    os.makedirs(rzsm_path_tmp)
```

Scaling methods

Available methods on pytesmo package are:

- **min-max correction** (min_max) - scales the input datasets so that they have the same minimum and maximum afterward
- **linear rescaling** (mean_std) - scales the input datasets so that they have the same mean and standard deviation afterwards
- **linear regression** (linreg) - scales the input datasets using linear regression
- **cdf matching** (cdf_match) - computes cumulative density functions of src and ref at their respective bin-edges by 5th order spline interpolation; then matches CDF of src to CDF of ref
- **linear cdf matching** (lin_cdf_match) - computes cumulative density functions of src and ref at their respective bin-edges by linear interpolation; then matches CDF of src to CDF of refs of src and ref at their respective bin-edges by linear interpolation; then matches CDF of src to CDF of ref

In [3]:
```
# Get scaling methods available on pytesmo
scaling_methods = get_scaling_method_lut()
# Print available methods
print(list(scaling_methods.keys()))
```

```
['linreg', 'mean_std', 'min_max', 'lin_cdf_match', 'cdf_match']
```

In [4]:
```
# Get scaling method
scaling_method_lr = get_scaling_function('linreg')
scaling_method_ms = get_scaling_function('mean_std')
```

Basin Configuration

The script loads the shapefile of the basin and creates a mask using the defined cell_size (degree) and boudary box buffer (bbox_ext in degree). After running the cell, results can be check using QGIS.

In [5]:
```
# Get basin information using a shapefile
basin_rows, basin_cols, basin_epsg, basin_transform, basin_meta_reference = get_file_shp(
    os.path.join(domain_path_layer, file_shp_domain),
    os.path.join(ancillary_path, 'basin_domain.tiff'),
    cell_size=0.005, bbox_ext=0)
# Print information about basin
print(basin_rows, basin_cols, basin_epsg, basin_transform)
```

```
232 201 EPSG:4326 | 0.01, 0.00, 11.91|
| 0.00,-0.01, 43.80|
| 0.00, 0.00, 1.00|
```

In [6]:
```
# Create basin grid using WARP5 reference system
basin_grid, basin_lons_2d, basin_lats_2d, basin_bbox = get_grid(
    os.path.join(ancillary_path, 'basin_domain.tiff'))
# Print information about basin
print(basin_bbox)
# Using QGIS to:
# 1) load basin shapefile
# 2) load basin tiff
# 3) check results
```

```
BoundingBox(left=11.911873170287814, bottom=42.638521266005235, right=12.916873170287815, top=43.7985212660052
3)
```

Datasets configuration

In this part ASCAT, ERA5 and RZSM datasets are configured using parameters and paths set previously.

- Step 1 -- Create settings dictionary to summarize information about datasets

In [7]:
```
# Create ASCAT, ERA5 and RZSM settings
settings = {
    "ascat_path_ts": ascat_path_ts,
    "ascat_path_grid": ascat_path_grid,
    "ascat_path_layer": ascat_path_layers,
    "ascat_path_tmp": ascat_path_tmp,
    "ascat_mask_frozen_prob_threshold": ascat_mask_frozen_prob_threshold,
    "ascat_mask_snow_prob_threshold": ascat_mask_snow_prob_threshold,
    "era5_path_ts": era5_path_ts,
    "era5_path_grid": era5_path_grid,
    "era5_path_tmp": era5_path_tmp,
    "rzsm_path_ts": rzsm_path_ts,
    "rzsm_path_grid": rzsm_path_grid,
    "rzsm_path_tmp": rzsm_path_tmp,
    "domain_path_layer": domain_path_layer,
    "time_start": time_start,
    "time_end": time_end,
    "temporal_matching": temporal_matching,
    "temporal_drop_duplicates": temporal_drop_duplicates,
    "max_dist": max_dist
}
# Print information about ASCAT and ERA5 settings
for key, value in settings.items():
    print(str(key) + ": " + str(settings[key]))
```

```
ascat_path_ts: /home/fabio/Desktop/PyCharm_Workspace/fp-labs/hsaf_event_week_2019/test_data/dynamic/ascat_dr/i
taly
ascat_path_grid: /home/fabio/Desktop/PyCharm_Workspace/fp-labs/hsaf_event_week_2019/test_data/static/warp5_gri
d
ascat_path_layer: /home/fabio/Desktop/PyCharm_Workspace/fp-labs/hsaf_event_week_2019/test_data/static/warp5_la
yers
ascat_path_tmp: /home/fabio/Desktop/PyCharm_Workspace/fp-labs/hsaf_event_week_2019/test_outcome/tmp/ex_spatial
_patterns/ascat
ascat_mask_frozen_prob_threshold: 100
ascat_mask_snow_prob_threshold: 100
era5_path_ts: /home/fabio/Desktop/PyCharm_Workspace/fp-labs/hsaf_event_week_2019/test_data/dynamic/era5_ts/ita
ly
era5_path_grid: /home/fabio/Desktop/PyCharm_Workspace/fp-labs/hsaf_event_week_2019/test_data/dynamic/era5_ts/i
taly
era5_path_tmp: /home/fabio/Desktop/PyCharm_Workspace/fp-labs/hsaf_event_week_2019/test_outcome/tmp/ex_spatial_
patterns/era5
rzsm_path_ts: /home/fabio/Desktop/PyCharm_Workspace/fp-labs/hsaf_event_week_2019/test_data/dynamic/rzsm_ts/ita
ly
rzsm_path_grid: /home/fabio/Desktop/PyCharm_Workspace/fp-labs/hsaf_event_week_2019/test_data/dynamic/rzsm_ts/i
taly
rzsm_path_tmp: /home/fabio/Desktop/PyCharm_Workspace/fp-labs/hsaf_event_week_2019/test_outcome/tmp/ex_spatial_
patterns/rzsm
domain_path_layer: /home/fabio/Desktop/PyCharm_Workspace/fp-labs/hsaf_event_week_2019/test_data/static/shp/ita
ly
time_start: 2007-01-01
time_end: 2014-12-31
temporal_matching: 24
temporal_drop_duplicates: False
max_dist: 35000
```

- Step 2 -- Initialize and configure reader objects for ASCAT, ERA5 and RZSM datasets

In [8]:
```
# Initialize ASCAT, ERA5, RZSM datasets
reader_ascat, reader_era5, reader_rzsm = dset_init(settings)
datasets = dset_config(reader_ascat, reader_era5, reader_rzsm, settings)
# Print information about ASCAT and ERA5 datasets
print("ASCAT dataset settings: " + str(datasets["ASCAT"]))
print("ERA5 dataset settings: " + str(datasets["ERA5"]))
print("RZSM dataset settings: " + str(datasets["RZSM"]))
```

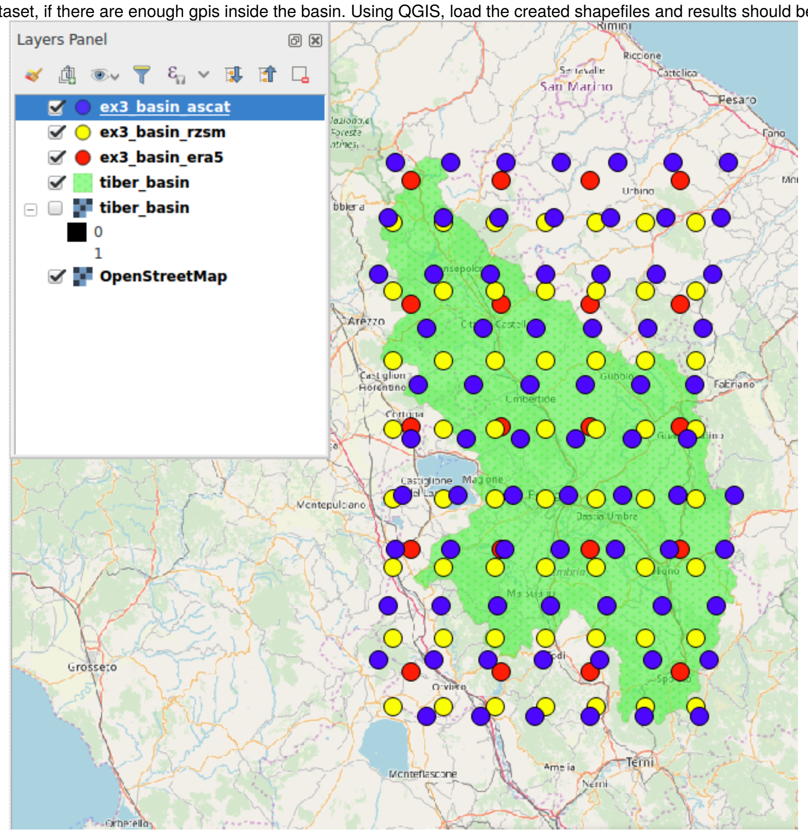
```
ASCAT dataset settings: {'class': <library.cima.ts_dset_driver.ASCAT_Dataset_DR object at 0x7f0f367c4b00>, 'co
lumns': ['sm'], 'type': 'reference', 'args': [], 'kwargs': {'mask_frozen_prob': 100, 'mask_snow_prob': 100}}
ERA5 dataset settings: {'class': <library.cima.ts_dset_driver.ERA5_Dataset_TS object at 0x7f0f367c4710>, 'colu
mns': ['tp', 'tsk'], 'type': 'other', 'grids_compatible': False, 'use_lut': True, 'lut_max_dist': 35000}
RZSM dataset settings: {'class': <library.cima.ts_dset_driver.RZSM_Dataset_TS object at 0x7f0f367c4fd0>, 'colu
mns': ['var40', 'var41', 'var42', 'var43'], 'type': 'other', 'grids_compatible': False, 'use_lut': True, 'lut_
max_dist': 35000}
```

- Step 3 -- Find GPIS of ASCAT, ERA5 and RZSM datasets using basin reference

In [9]:
```
# Create ASCAT, ERA5 and RZSM grid(s) using basin information
# Get ascat gpi(s)
gpis_ascat, lats_ascat, lons_ascat = reader_ascat.grid.get_bbox_grid_points(
    latmin=basin_bbox.bottom, latmax=basin_bbox.top, lonmin=basin_bbox.left,
    lonmax=basin_bbox.right, both=True)
gpis_ascat_n = gpis_ascat.__len__()
# Get era5 gpi(s)
gpis_era5, lats_era5, lons_era5 = reader_era5.grid.get_bbox_grid_points(
    latmin=basin_bbox.bottom, latmax=basin_bbox.top, lonmin=basin_bbox.left,
    lonmax=basin_bbox.right, both=True)
gpis_era5_n = gpis_era5.__len__()
# Get rzsm gpi(s)
gpis_rzsm, lats_rzsm, lons_rzsm = reader_rzsm.grid.get_bbox_grid_points(
    latmin=basin_bbox.bottom, latmax=basin_bbox.top, lonmin=basin_bbox.left,
    lonmax=basin_bbox.right, both=True)
gpis_rzsm_n = gpis_rzsm.__len__()

# Print information about ASCAT, ERA5 and RZSM gpi(s) numerosity
print("ASCAT GPIS N: " + str(gpis_ascat_n))
print("ERA5 GPIS N: " + str(gpis_era5_n))
print("RZSM GPIS N: " + str(gpis_rzsm_n))
```

```
ASCAT GPIS N: 73
ERA5 GPIS N: 20
RZSM GPIS N: 56
```

- Step 4 -- Verify, for each dataset, if there are enough gpis inside the basin. Using QGIS, load the created shapefiles and results should be as follows.



In [10]:
```
# Find ASCAT gpi(s) over basin (using a maximum distance parameter)
gpis_basin_ascat = basin_grid.calc_lut(reader_ascat.grid, max_dist=settings['max_dist'])
gpis_basin_ascat = np.unique(gpis_basin_ascat)
lons_basin_ascat, lats_basin_ascat = reader_ascat.grid.gpi2lonlat(gpis_basin_ascat)
# Create shapefile of ASCAT gpi(s) over basin
create_points_shp(gpis_basin_ascat, lons_basin_ascat, lats_basin_ascat,
                  file_name_shp=os.path.join(ancillary_path, 'basin_ascat.shp'))
```

In [11]:
```
# Find ERA5 gpi(s) over basin (using a maximum distance parameter)
gpis_basin_era5 = basin_grid.calc_lut(reader_era5.grid, max_dist=settings['max_dist'])
gpis_basin_era5 = np.unique(gpis_basin_era5)
lons_basin_era5, lats_basin_era5 = basin_grid.gpi2lonlat(gpis_basin_era5)
# Create shapefile of ERA5 gpi(s) over basin
create_points_shp(gpis_basin_era5, lons_basin_era5, lats_basin_era5,
                  file_name_shp=os.path.join(ancillary_path, 'basin_era5.shp'))
```

In [12]:
```
# Find RZSM gpi(s) over basin (using a maximum distance parameter)
gpis_basin_rzsm = basin_grid.calc_lut(reader_rzsm.grid, max_dist=settings['max_dist'])
gpis_basin_rzsm = np.unique(gpis_basin_rzsm)
lons_basin_rzsm, lats_basin_rzsm = basin_grid.gpi2lonlat(gpis_basin_rzsm)
# Create shapefile of RZSM gpi(s) over basin
create_points_shp(gpis_basin_rzsm, lons_basin_rzsm, lats_basin_rzsm,
                  file_name_shp=os.path.join(ancillary_path, 'basin_rzsm.shp'))
```

- Step 5 -- Find gpis of ERA5 and RZSM using ASCAT as reference dataset

In [13]:
```
# Define ASCAT, ERA5 and RZSM common gpis
gpis_ascat_ws = reader_ascat.grid.find_nearest_gpi(lons_ascat, lats_ascat, max_dist=settings['max_dist'])
gpis_ascat = gpis_ascat_ws[0]; dist_ascat = gpis_ascat_ws[1];
lons_ascat, lats_ascat = reader_ascat.grid.gpi2lonlat(gpis_ascat)

gpis_era5_ws = reader_era5.grid.find_nearest_gpi(lons_ascat, lats_ascat, max_dist=settings['max_dist'])
gpis_era5 = gpis_era5_ws[0]; dist_era5 = gpis_era5_ws[1];
lons_era5, lats_era5 = reader_era5.grid.gpi2lonlat(gpis_era5)

gpis_rzsm_ws = reader_rzsm.grid.find_nearest_gpi(lons_ascat, lats_ascat, max_dist=settings['max_dist'])
gpis_rzsm = gpis_rzsm_ws[0]; dist_rzsm = gpis_rzsm_ws[1];
lons_rzsm, lats_rzsm = reader_rzsm.grid.gpi2lonlat(gpis_rzsm)
```

A. Extract ASCAT, ERA5 and RZSM datasets

Once datasets are prepared, for all gpis, the time-series of ASCAT and ERA5 are extracted, SWI created and all data are stored in a common dataframe

In [14]:
```
# Download time-series using ASCAT gpi(s) as a reference
ts_ascat_ws = pd.DataFrame()
gpis_ws = zip(gpis_basin_ascat, lons_basin_ascat, lats_basin_ascat)

print(' => Download ASCAT time-series ... ')
for id, (gpi_ascat, lon_ascat, lat_ascat) in notebook.tqdm(enumerate(gpis_ws),
                                                           total=gpis_basin_ascat.__len__(),
                                                           desc=' ==== Download time-series progress'):

    # Select gpi for RZSM and ERA5
    gpi_rzsm = reader_rzsm.grid.find_nearest_gpi(lon_ascat, lat_ascat, max_dist=settings['max_dist'])[0]
    lon_rzsm, lat_rzsm = reader_rzsm.grid.gpi2lonlat(gpi_rzsm)

    gpi_era5 = reader_era5.grid.find_nearest_gpi(lon_ascat, lat_ascat, max_dist=settings['max_dist'])[0]
    lon_era5, lat_era5 = reader_era5.grid.gpi2lonlat(gpi_era5)

    # Info
    ts_sm = 'sm_' + str(gpi_ascat)
    ts_sp = 'sp_' + str(gpi_ascat)
    ts_fp = 'fp_' + str(gpi_ascat)
    ts_swi_t1 = 'swi_t1_' + str(gpi_ascat)
    ts_swi_t5 = 'swi_t5_' + str(gpi_ascat)
    ts_swi_t10 = 'swi_t10_' + str(gpi_ascat)
    ts_swi_t50 = 'swi_t50_' + str(gpi_ascat)
    ts_var40 = 'var40_' + str(gpi_ascat)

    # Get ASCAT data
    #print(' ==> Step: ' + str(id+1) + '/' + str(gpis_basin_ascat.shape[0]))
    #print(' ==> Get gpi: ' + str(gpi_ascat) + ' ... ')
    ts_ascat = reader_ascat.read_ts(gpi_ascat)
    ts_ascat = ts_ascat.loc[settings['time_start']:settings['time_end']]

    ts_rzsm = reader_rzsm.read_ts(gpi_rzsm)
    ts_rzsm = ts_rzsm.loc[settings['time_start']:settings['time_end']]
    #print(' ==> Get gpi: ' + str(gpi_ascat) + ' ... DONE')

    # Resample
    ts_ascat_ws[ts_sm] = ts_ascat.sm.resample('D').mean().dropna()
    ts_ascat_ws[ts_sp] = ts_ascat.snow_prob.resample('D').mean().dropna()
    ts_ascat_ws[ts_fp] = ts_ascat.frozen_prob.resample('D').mean().dropna()
    ts_ascat_ws[ts_var40] = ts_rzsm.var40.resample('D').mean().dropna()

    # Get julian dates of time series 1
    jd = ts_ascat_ws[ts_sm].index.to_julian_date().get_values()

    # Calculate filter SWI T=1,5,10,50
    ts_ascat_ws[ts_swi_t1]  = exp_filter(ts_ascat_ws[ts_sm].values, jd, ctime=1)
    ts_ascat_ws[ts_swi_t5]  = exp_filter(ts_ascat_ws[ts_sm].values, jd, ctime=5)
    ts_ascat_ws[ts_swi_t10]  = exp_filter(ts_ascat_ws[ts_sm].values, jd, ctime=10)
    ts_ascat_ws[ts_swi_t50]  = exp_filter(ts_ascat_ws[ts_sm].values, jd, ctime=50)


print(' => Download ASCAT time-series ... DONE!')
```

```
 => Download ASCAT time-series ...


 => Download ASCAT time-series ... DONE!
```

Evaluation of time-series flags

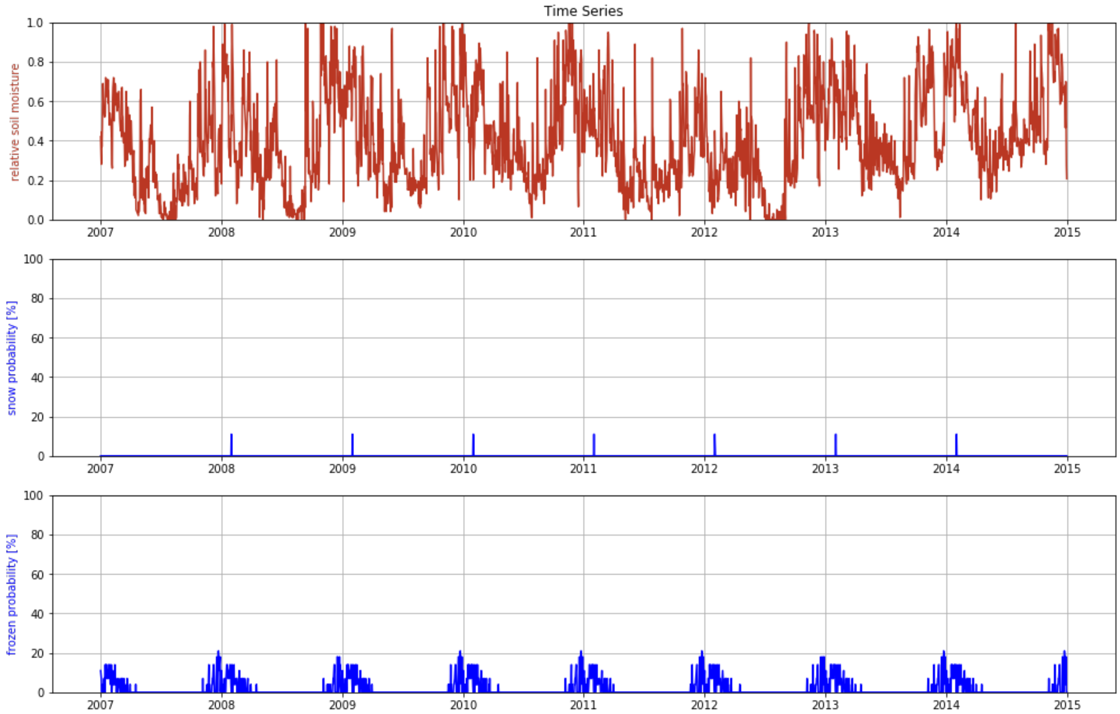
1. Select a ASCAT gpi to consider the effect of changing threshold of frozen or snow probability and view the differences between "flagged" and "complete" time-series

In [15]:
```
# Get time-series for a gpi
gpi_id = 0
gpi_basin_ascat = gpis_basin_ascat[gpi_id]

ts_sm = 'sm_' + str(gpi_basin_ascat)
ts_sp = 'sp_' + str(gpi_basin_ascat)
ts_fp = 'fp_' + str(gpi_basin_ascat)
ts_swi_t1 = 'swi_t1_' + str(gpi_basin_ascat)
ts_swi_t5 = 'swi_t5_' + str(gpi_basin_ascat)
ts_swi_t10 = 'swi_t10_' + str(gpi_basin_ascat)
ts_swi_t50 = 'swi_t50_' + str(gpi_basin_ascat)
ts_var40 = 'var40_' + str(gpi_ascat)

ts_ascat_id = pd.DataFrame()
ts_ascat_id['sm'] = ts_ascat_ws[ts_sm]
ts_ascat_id['snow_prob'] = ts_ascat_ws[ts_sp]
ts_ascat_id['frozen_prob'] = ts_ascat_ws[ts_fp]
# Print time-series for one gpi
print(ts_ascat_id.head())
```

```
               sm  snow_prob  frozen_prob
2007-01-02  0.420        0.0         11.0
2007-01-04  0.280        0.0          7.0
2007-01-05  0.445        0.0          4.0
2007-01-06  0.370        0.0          0.0
2007-01-07  0.470        0.0          4.0
```

2. Plot time-series of soil moisture, frozen probability and snow probability for evaluating the impact and distribution of flags

In [16]:
```
# Plot ASCAT variable(s) (soil moisture, snow probability and frozen probability)
fig, axs = plt.subplots(3, 1, figsize=(17, 11))
axs[0].plot(ts_ascat_id['sm'], color='#BA3723')
axs[1].plot(ts_ascat_id['snow_prob'], color='#0000FF')
axs[2].plot(ts_ascat_id['frozen_prob'], color='#0000FF')

axs[0].set_title('Time Series')
axs[0].set_ylabel('relative soil moisture', color='#BA3723')
axs[0].set_ylim(0, 1)
axs[0].grid(b=True)
axs[1].set_ylabel('snow probability [%]', color='#0000FF')
axs[1].set_ylim(0, 100)
axs[1].grid(b=True)
axs[2].set_ylabel('frozen probability [%]', color='#0000FF')
axs[2].set_ylim(0, 100)
axs[2].grid(b=True)

filename = os.path.join(img_path, "ex_ts_sm_flags.tiff")
fig.savefig(filename, dpi=120)
```



3. Filter time-series using a threshold for frozen probability field

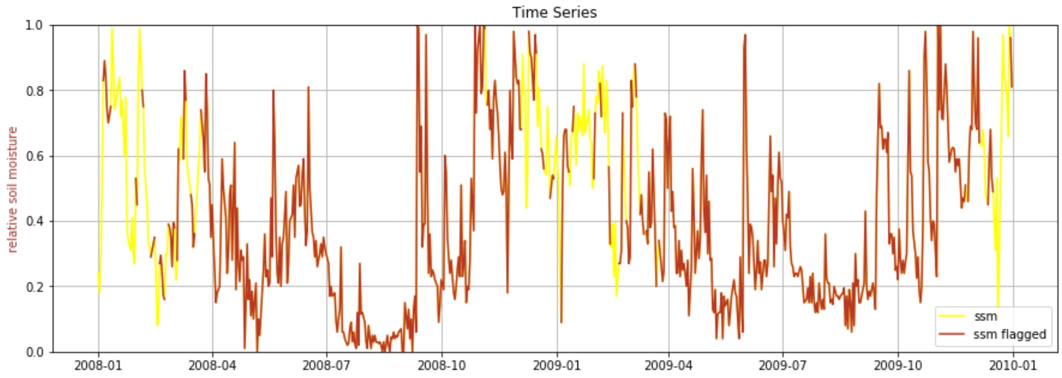
```
* frozen threshold can be modified according to basin climatology
* if snow is an important condition modify/add the cell using snow_prob field and declaring a snow threshold
```

In [17]:
```
# Filter time-series using a threshold for frozen probability field
frozen_thr = 7
ts_ascat_filter = ts_ascat_id.copy()
ts_ascat_filter.loc[(ts_ascat_id['frozen_prob'] >= frozen_thr), 'sm'] = np.nan
# Print time-series for one gpi
print(ts_ascat_filter.head())
```

```
               sm  snow_prob  frozen_prob
2007-01-02    NaN        0.0         11.0
2007-01-04    NaN        0.0          7.0
2007-01-05  0.445        0.0          4.0
2007-01-06  0.370        0.0          0.0
2007-01-07  0.470        0.0          4.0
```

4. Plot flagged and not-flagged time-series to check what are the differences between them

In [18]:
```
# Plot ASCAT variable(s) flagged
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(ts_ascat_id['2008':'2009']['sm'], color='#FFFF00', label='ssm')
ax.plot(ts_ascat_filter['2008':'2009']['sm'], color='#BA3723', label='ssm flagged')

ax.set_title('Time Series')
ax.set_ylabel('relative soil moisture', color='#BA3723')
ax.set_ylim(0, 1)
ax.grid(b=True)
plt.legend()

filename = os.path.join(img_path, "ex_ts_sm_filtered.tiff")
fig.savefig(filename, dpi=120)
```



5. Analyze the flagged and not-flagged time-series, plot a map for a day that can be investigated spatially

```
* modify time_analysis
* modify time_window (if needed)
```

In [19]:
```
# Time analysis [YYYY-MM-DD HH:MM]
time_analysis = "2012-09-01 00:00"
time_window = 24 # hours

# Select data time-series using a time reference step
pnt_ascat_ws = pd.DataFrame(
    columns=['lon', 'lat', 'gpi', 'sm', 'snow_prob', 'frozen_prob', 'var40',
             'swi_t1', 'swi_t5', 'swi_t10', 'swi_t50', 'time'])

gpis_ws = zip(gpis_basin_ascat, lons_basin_ascat, lats_basin_ascat)

print(' => Analyze ASCAT from time-series to maps ... ')
for id, (gpi_ascat, lon_ascat, lat_ascat) in notebook.tqdm(enumerate(gpis_ws),
                                                           total=gpis_basin_ascat.__len__(),
                                                           desc=' ==== Analyze time-series progress'):

    # Info
    ts_sm = 'sm_' + str(gpi_ascat)
    ts_sp = 'sp_' + str(gpi_ascat)
    ts_fp = 'fp_' + str(gpi_ascat)
    ts_swi_t1 = 'swi_t1_' + str(gpi_ascat)
    ts_swi_t5 = 'swi_t5_' + str(gpi_ascat)
    ts_swi_t10 = 'swi_t10_' + str(gpi_ascat)
    ts_swi_t50 = 'swi_t50_' + str(gpi_ascat)
    ts_var40 = 'var40_' + str(gpi_ascat)

    pnt_ascat_sm = df_time_matching(ts_ascat_ws[ts_sm],
                                    time_analysis, window=time_window)[0]
    pnt_ascat_sp = df_time_matching(ts_ascat_ws[ts_sp],
                                    time_analysis, window=time_window)[0]
    pnt_ascat_fp = df_time_matching(ts_ascat_ws[ts_fp],
                                    time_analysis, window=time_window)[0]
    pnt_ascat_swi1 = df_time_matching(ts_ascat_ws[ts_swi_t1],
                                    time_analysis, window=time_window)[0]
    pnt_ascat_swi5 = df_time_matching(ts_ascat_ws[ts_swi_t5],
                                    time_analysis, window=time_window)[0]
    pnt_ascat_swi10 = df_time_matching(ts_ascat_ws[ts_swi_t10],
                                    time_analysis, window=time_window)[0]
    pnt_ascat_swi50 = df_time_matching(ts_ascat_ws[ts_swi_t50],
                                    time_analysis, window=time_window)[0]
    pnt_ascat_var40 = df_time_matching(ts_ascat_ws[ts_var40],
                                    time_analysis, window=time_window)[0]

    pnt_ascat_ws = pnt_ascat_ws.append(
        {'lon': lon_ascat, 'lat': lat_ascat, 'gpi': gpi_ascat,
         'sm': pnt_ascat_sm, 'snow_prob': pnt_ascat_sp, 'frozen_prob': pnt_ascat_fp,
         'var40':pnt_ascat_var40,
         'swi_t1': pnt_ascat_swi1, 'swi_t5': pnt_ascat_swi5,
         'swi_t10': pnt_ascat_swi10, 'swi_t50': pnt_ascat_swi50,
         'time': time_analysis}, ignore_index=True)

# Remove NaN
pnt_ascat_ws = pnt_ascat_ws.dropna()
print(' => Analyze ASCAT from time-series to maps ... DONE')
```

```
 => Analyze ASCAT from time-series to maps ...

 => Analyze ASCAT from time-series to maps ... DONE
```

In [20]:
```
# Print point(s)
print(pnt_ascat_ws.head(n=3)); print(pnt_ascat_ws.tail(n=3));
```

```
        lon        lat      gpi    sm  snow_prob  frozen_prob     var40  \
0  11.891937  42.660675  2204747  0.03        0.0          0.0  0.397095
1  12.044397  42.660675  2204751  0.00        0.0          0.0  0.401367
2  12.196858  42.660675  2204755  0.00        0.0          0.0  0.406677

     swi_t1    swi_t5   swi_t10   swi_t50              time
0  0.146132  0.112522  0.076698  0.095108  2012-09-01 00:00
1  0.120343  0.093348  0.062862  0.088953  2012-09-01 00:00
2  0.074914  0.059046  0.038468  0.070813  2012-09-01 00:00
         lon        lat      gpi    sm  snow_prob  frozen_prob     var40  \
79  12.578540  43.785828  2251599  0.39        0.0          0.0  0.487030
80  12.733830  43.785828  2251603  0.40      127.0          0.0  0.487030
81  12.889121  43.785828  2251607  0.46      127.0          0.0  0.456787

      swi_t1    swi_t5   swi_t10   swi_t50              time
79  0.474895  0.317268  0.222073  0.134040  2012-09-01 00:00
80  0.400415  0.258917  0.189628  0.122803  2012-09-01 00:00
81  0.367853  0.227488  0.177101  0.130750  2012-09-01 00:00
```

6. Mask of undefined values of snow_prob and frozen_prob (when values are equal = 127)

In [21]:
```
# Mask undefined values (snow_prob = 127 or/and frozen_prob = 127)
pnt_ascat_ws.loc[pnt_ascat_ws['snow_prob'] == 127, 'snow_prob'] = np.nan
pnt_ascat_ws.loc[pnt_ascat_ws['frozen_prob'] == 127, 'frozen_prob'] = np.nan
# Print point(s)
print(pnt_ascat_ws.head(n=3)); print(pnt_ascat_ws.tail(n=3))
```

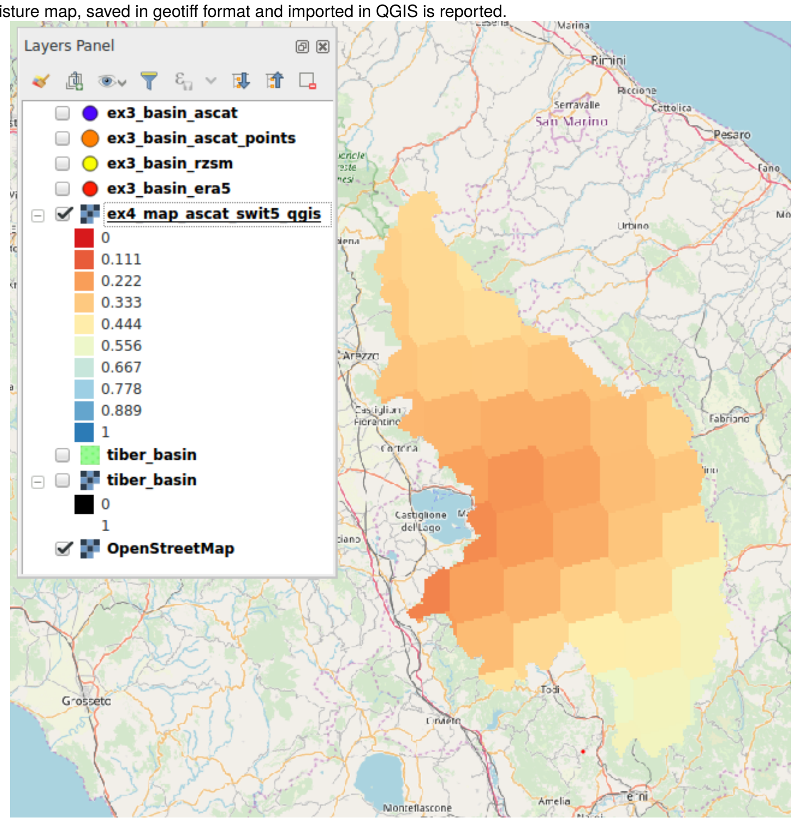
```
         lon        lat      gpi    sm  snow_prob  frozen_prob     var40  \
0  11.891937  42.660675  2204747  0.03        0.0          0.0  0.397095
1  12.044397  42.660675  2204751  0.00        0.0          0.0  0.401367
2  12.196858  42.660675  2204755  0.00        0.0          0.0  0.406677

     swi_t1    swi_t5   swi_t10   swi_t50             time
0  0.146132  0.112522  0.076698  0.095108  2012-09-01 00:00
1  0.120343  0.093348  0.062862  0.088953  2012-09-01 00:00
2  0.074914  0.059046  0.038468  0.070813  2012-09-01 00:00
          lon        lat      gpi    sm  snow_prob  frozen_prob     var40  \
79  12.578540  43.785828  2251599  0.39        0.0          0.0  0.487030
80  12.733830  43.785828  2251603  0.40        NaN          0.0  0.487030
81  12.889121  43.785828  2251607  0.46        NaN          0.0  0.456787

      swi_t1    swi_t5   swi_t10   swi_t50             time
79  0.474895  0.317268  0.222073  0.134040  2012-09-01 00:00
80  0.400415  0.258917  0.189628  0.122803  2012-09-01 00:00
81  0.367853  0.227488  0.177101  0.130750  2012-09-01 00:00
```

7. Interpolation of ASCAT values for selected date (nearest neighbour method)

- all variables (sm, SWI, snow prob, frozen prob and var40) are interpolated over basin domain
- figures are created and maps are saved as geotiff

In [22]:
```
# Interpolate ASCAT points over domain
map_ascat_sm = interpolate_point2map(pnt_ascat_ws['lon'].values, pnt_ascat_ws['lat'].values,
                                     pnt_ascat_ws['sm'].values, basin_lons_2d, basin_lats_2d)
map_ascat_swi_t5 = interpolate_point2map(pnt_ascat_ws['lon'].values, pnt_ascat_ws['lat'].values,
                                         pnt_ascat_ws['swi_t5'].values, basin_lons_2d, basin_lats_2d)
map_ascat_swi_t10 = interpolate_point2map(pnt_ascat_ws['lon'].values, pnt_ascat_ws['lat'].values,
                                          pnt_ascat_ws['swi_t10'].values, basin_lons_2d, basin_lats_2d)
map_ascat_swi_t50 = interpolate_point2map(pnt_ascat_ws['lon'].values, pnt_ascat_ws['lat'].values,
                                          pnt_ascat_ws['swi_t50'].values, basin_lons_2d, basin_lats_2d)
map_ascat_snow_prob = interpolate_point2map(pnt_ascat_ws['lon'].values, pnt_ascat_ws['lat'].values,
                                     pnt_ascat_ws['snow_prob'].values, basin_lons_2d, basin_lats_2d)
map_ascat_frozen_prob = interpolate_point2map(pnt_ascat_ws['lon'].values, pnt_ascat_ws['lat'].values,
                                         pnt_ascat_ws['frozen_prob'].values, basin_lons_2d, basin_lats_2d)
map_ascat_var40 = interpolate_point2map(pnt_ascat_ws['lon'].values, pnt_ascat_ws['lat'].values,
                                         pnt_ascat_ws['var40'].values, basin_lons_2d, basin_lats_2d)
# Save map(s) in a Tiff format (to plot file using QGIS)
create_map(map_ascat_sm, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_ascat_sm_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
create_map(map_ascat_swi_t5, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_ascat_swit5_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
create_map(map_ascat_swi_t10, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_ascat_swit10_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
create_map(map_ascat_swi_t50, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_ascat_swit50_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
create_map(map_ascat_snow_prob, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_ascat_snow_prob_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
create_map(map_ascat_frozen_prob, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_ascat_frozen_prob_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
create_map(map_ascat_var40, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_rszm_var40_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
```

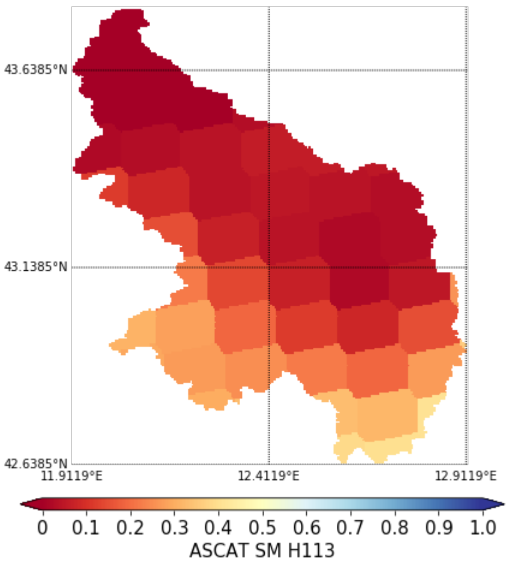
An example of ASCAT soil moisture map, saved in geotiff format and imported in QGIS is reported.



Plot maps

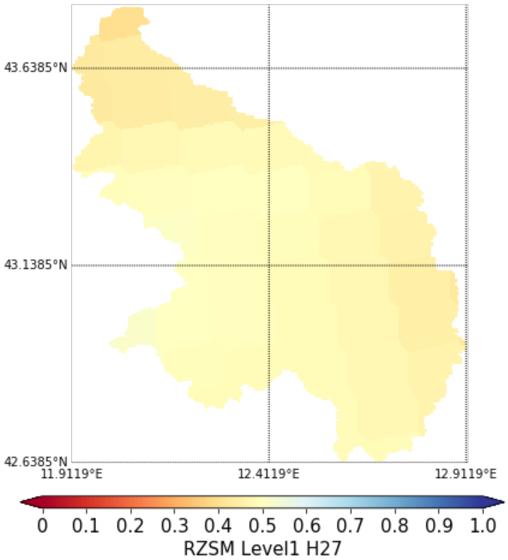
1. Plot map of ASCAT soil moisture (H113)

In [23]:
```
# Plot map of ASCAT SM
data = map_ascat_sm
lons = basin_lons_2d
lats = np.flipud(basin_lats_2d)

vmin = 0
vmax = 1
cb_label='ASCAT SM H113'

cmap_name='RdYlBu'

minlat = basin_bbox.bottom
maxlat = basin_bbox.top
minlon = basin_bbox.left
maxlon = basin_bbox.right

cb_xticklabels = ['0', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7', '0.8', '0.9', '1.0']
cb_tickloc = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

norm = mc.Normalize(vmin=vmin, vmax=vmax)

fig, ax = plt.subplots(figsize=(7, 7))
basemap = Basemap(llcrnrlat=minlat, urcrnrlat=maxlat, llcrnrlon=minlon, urcrnrlon=maxlon, resolution='i')
basemap.drawmapboundary(linewidth=0)
#oBaseMap.fillcontinents(color='gray', zorder=2)
basemap.drawparallels(np.arange(minlat, maxlat, 0.5), labels=[True, False, False, False], fontsize=10)
basemap.drawmeridians(np.arange(minlon, maxlon, 0.5), labels=[False, False, False, True], fontsize=10)

basedata = basemap.pcolormesh(lons, lats,
                                data, shading='flat', norm=norm, cmap=cmap_name)

cb_ax = fig.add_axes([0.1, 0.05, 0.8, 0.02])
cb = fig.colorbar(basedata, orientation='horizontal', cax=cb_ax, extend='both')
cb.set_label(cb_label, fontsize=15)
cb.ax.tick_params(labelsize=15)
cb.set_ticks(cb_tickloc)
cb.ax.set_xticklabels(cb_xticklabels)

filename = os.path.join(img_path, "ex_sp_ascat_sm.tiff")
fig.savefig(filename, dpi=120)
```



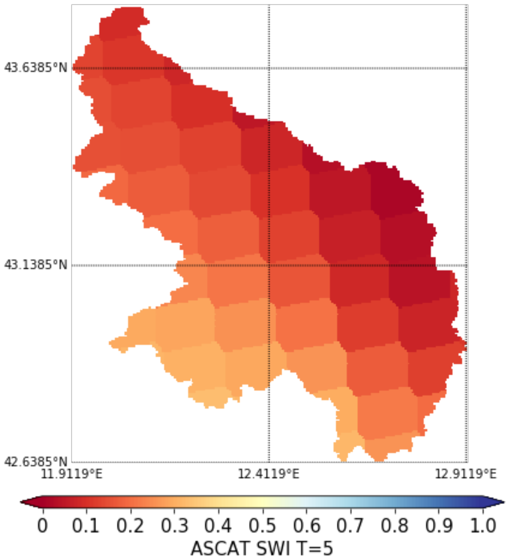
2. Plot map of RZSM level 1 (H27)

In [24]:
```
# Plot map of H27
data = map_ascat_var40
lons = basin_lons_2d
lats = np.flipud(basin_lats_2d)

vmin = 0
vmax = 1
cb_label='RZSM Level1 H27'

cmap_name='RdYlBu'

minlat = basin_bbox.bottom
maxlat = basin_bbox.top
minlon = basin_bbox.left
maxlon = basin_bbox.right

cb_xticklabels = ['0', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7', '0.8', '0.9', '1.0']
cb_tickloc = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

norm = mc.Normalize(vmin=vmin, vmax=vmax)

fig, ax = plt.subplots(figsize=(7, 7))
basemap = Basemap(llcrnrlat=minlat, urcrnrlat=maxlat, llcrnrlon=minlon, urcrnrlon=maxlon, resolution='i')
basemap.drawmapboundary(linewidth=0)
#oBaseMap.fillcontinents(color='gray', zorder=2)
basemap.drawparallels(np.arange(minlat, maxlat, 0.5), labels=[True, False, False, False], fontsize=10)
basemap.drawmeridians(np.arange(minlon, maxlon, 0.5), labels=[False, False, False, True], fontsize=10)

basedata = basemap.pcolormesh(lons, lats,
                                data, shading='flat', norm=norm, cmap=cmap_name)

cb_ax = fig.add_axes([0.1, 0.05, 0.8, 0.02])
cb = fig.colorbar(basedata, orientation='horizontal', cax=cb_ax, extend='both')
cb.set_label(cb_label, fontsize=15)
cb.ax.tick_params(labelsize=15)
cb.set_ticks(cb_tickloc)
cb.ax.set_xticklabels(cb_xticklabels)

filename = os.path.join(img_path, "ex_sp_rzsm_var40.tiff")
fig.savefig(filename, dpi=120)
```



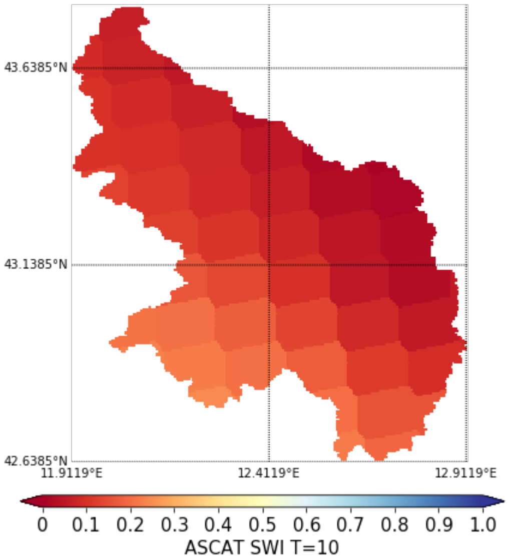
3. Plot map of ASCAT SWI T=5

In [25]:
```
# Plot map of ASCAT SWI T=5
data = map_ascat_swi_t5
lons = basin_lons_2d
lats = np.flipud(basin_lats_2d)

vmin = 0
vmax = 1
cb_label='ASCAT SWI T=5'

cmap_name='RdYlBu'

minlat = basin_bbox.bottom
maxlat = basin_bbox.top
minlon = basin_bbox.left
maxlon = basin_bbox.right

cb_xticklabels = ['0', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7', '0.8', '0.9', '1.0']
cb_tickloc = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

norm = mc.Normalize(vmin=vmin, vmax=vmax)

fig, ax = plt.subplots(figsize=(7, 7))
basemap = Basemap(llcrnrlat=minlat, urcrnrlat=maxlat, llcrnrlon=minlon, urcrnrlon=maxlon, resolution='i')
basemap.drawmapboundary(linewidth=0)
#oBaseMap.fillcontinents(color='gray', zorder=2)
basemap.drawparallels(np.arange(minlat, maxlat, 0.5), labels=[True, False, False, False], fontsize=10)
basemap.drawmeridians(np.arange(minlon, maxlon, 0.5), labels=[False, False, False, True], fontsize=10)

basedata = basemap.pcolormesh(lons, lats,
                              data, shading='flat', norm=norm, cmap=cmap_name)

cb_ax = fig.add_axes([0.1, 0.05, 0.8, 0.02])
cb = fig.colorbar(basedata, orientation='horizontal', cax=cb_ax, extend='both')
cb.set_label(cb_label, fontsize=15)
cb.ax.tick_params(labelsize=15)
cb.set_ticks(cb_tickloc)
cb.ax.set_xticklabels(cb_xticklabels)

filename = os.path.join(img_path, "ex_sp_ascat_swit5.tiff")
fig.savefig(filename, dpi=120)
```



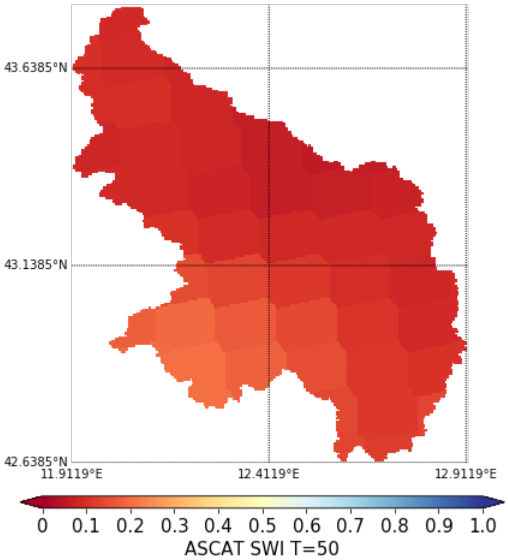
4. Plot map of ASCAT SWI T=10

In [26]:
```
# Plot map of ASCAT SWI T=10
data = map_ascat_swi_t10
lons = basin_lons_2d
lats = np.flipud(basin_lats_2d)

vmin = 0
vmax = 1
cb_label='ASCAT SWI T=10'

cmap_name='RdYlBu'

minlat = basin_bbox.bottom
maxlat = basin_bbox.top
minlon = basin_bbox.left
maxlon = basin_bbox.right

cb_xticklabels = ['0', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7', '0.8', '0.9', '1.0']
cb_tickloc = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

norm = mc.Normalize(vmin=vmin, vmax=vmax)

fig, ax = plt.subplots(figsize=(7, 7))
basemap = Basemap(llcrnrlat=minlat, urcrnrlat=maxlat, llcrnrlon=minlon, urcrnrlon=maxlon, resolution='i')
basemap.drawmapboundary(linewidth=0)
#oBaseMap.fillcontinents(color='gray', zorder=2)
basemap.drawparallels(np.arange(minlat, maxlat, 0.5), labels=[True, False, False, False], fontsize=10)
basemap.drawmeridians(np.arange(minlon, maxlon, 0.5), labels=[False, False, False, True], fontsize=10)

basedata = basemap.pcolormesh(lons, lats,
                              data, shading='flat', norm=norm, cmap=cmap_name)

cb_ax = fig.add_axes([0.1, 0.05, 0.8, 0.02])
cb = fig.colorbar(basedata, orientation='horizontal', cax=cb_ax, extend='both')
cb.set_label(cb_label, fontsize=15)
cb.ax.tick_params(labelsize=15)
cb.set_ticks(cb_tickloc)
cb.ax.set_xticklabels(cb_xticklabels)

filename = os.path.join(img_path, "ex_sp_ascat_swit10.tiff")
fig.savefig(filename, dpi=120)
```



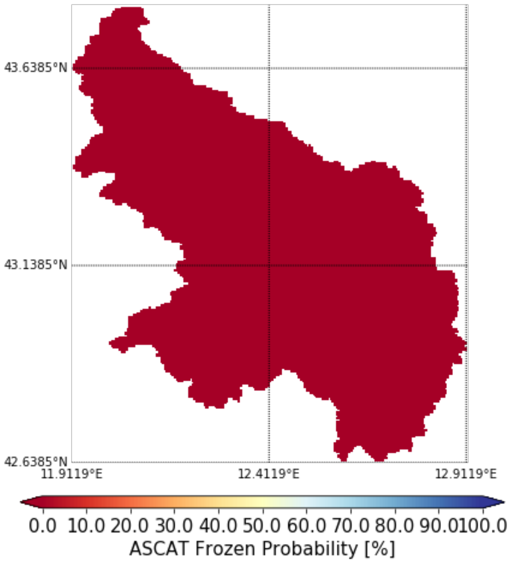
5. Plot map of ASCAT SWI T=50

In [27]:
```
# Plot map of ASCAT SWI T=50
data = map_ascat_swi_t50
lons = basin_lons_2d
lats = np.flipud(basin_lats_2d)

vmin = 0
vmax = 1
cb_label='ASCAT SWI T=50'

cmap_name='RdYlBu'

minlat = basin_bbox.bottom
maxlat = basin_bbox.top
minlon = basin_bbox.left
maxlon = basin_bbox.right

cb_xticklabels = ['0', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7', '0.8', '0.9', '1.0']
cb_tickloc = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

norm = mc.Normalize(vmin=vmin, vmax=vmax)

fig, ax = plt.subplots(figsize=(7, 7))
basemap = Basemap(llcrnrlat=minlat, urcrnrlat=maxlat, llcrnrlon=minlon, urcrnrlon=maxlon, resolution='i')
basemap.drawmapboundary(linewidth=0)
#oBaseMap.fillcontinents(color='gray', zorder=2)
basemap.drawparallels(np.arange(minlat, maxlat, 0.5), labels=[True, False, False, False], fontsize=10)
basemap.drawmeridians(np.arange(minlon, maxlon, 0.5), labels=[False, False, False, True], fontsize=10)

basedata = basemap.pcolormesh(lons, lats,
                              data, shading='flat', norm=norm, cmap=cmap_name)

cb_ax = fig.add_axes([0.1, 0.05, 0.8, 0.02])
cb = fig.colorbar(basedata, orientation='horizontal', cax=cb_ax, extend='both')
cb.set_label(cb_label, fontsize=15)
cb.ax.tick_params(labelsize=15)
cb.set_ticks(cb_tickloc)
cb.ax.set_xticklabels(cb_xticklabels)

filename = os.path.join(img_path, "ex_sp_ascat_swit50.tiff")
fig.savefig(filename, dpi=120)
```



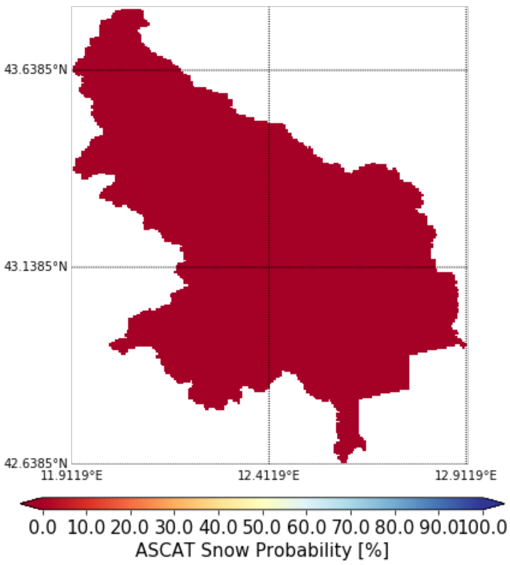
6. Plot map of ASCAT frozen soil probability

In [28]:
```
# Plot map of ASCAT frozen probability
data = map_ascat_frozen_prob
lons = basin_lons_2d
lats = np.flipud(basin_lats_2d)

vmin = 0
vmax = 100
cb_label='ASCAT Frozen Probability [%]'

cmap_name='RdYlBu'

minlat = basin_bbox.bottom
maxlat = basin_bbox.top
minlon = basin_bbox.left
maxlon = basin_bbox.right

cb_xticklabels = ['0.0', '10.0', '20.0', '30.0', '40.0', '50.0', '60.0', '70.0', '80.0', '90.0', '100.0']
cb_tickloc = [0, 10.0, 20.0, 30.0, 40.0, 50.0, 60.0, 70.0, 80.0, 90.0, 100.0]

norm = mc.Normalize(vmin=vmin, vmax=vmax)

fig, ax = plt.subplots(figsize=(7, 7))
basemap = Basemap(llcrnrlat=minlat, urcrnrlat=maxlat, llcrnrlon=minlon, urcrnrlon=maxlon, resolution='i')
basemap.drawmapboundary(linewidth=0)
#oBaseMap.fillcontinents(color='gray', zorder=2)
basemap.drawparallels(np.arange(minlat, maxlat, 0.5), labels=[True, False, False, False], fontsize=10)
basemap.drawmeridians(np.arange(minlon, maxlon, 0.5), labels=[False, False, False, True], fontsize=10)

basedata = basemap.pcolormesh(lons, lats,
                              data, shading='flat', norm=norm, cmap=cmap_name)

cb_ax = fig.add_axes([0.1, 0.05, 0.8, 0.02])
cb = fig.colorbar(basedata, orientation='horizontal', cax=cb_ax, extend='both')
cb.set_label(cb_label, fontsize=15)
cb.ax.tick_params(labelsize=15)
cb.set_ticks(cb_tickloc)
cb.ax.set_xticklabels(cb_xticklabels)

filename = os.path.join(img_path, "ex_sp_ascat_frozen_prob.tiff")
fig.savefig(filename, dpi=120)
```



7. Plot map of ASCAT snow cover probability

In [29]:
```
# Plot map of ASCAT snow probability
data = map_ascat_snow_prob
lons = basin_lons_2d
lats = np.flipud(basin_lats_2d)

vmin = 0
vmax = 100
cb_label='ASCAT Snow Probability [%]'

cmap_name='RdYlBu'

minlat = basin_bbox.bottom
maxlat = basin_bbox.top
minlon = basin_bbox.left
maxlon = basin_bbox.right

cb_xticklabels = ['0.0', '10.0', '20.0', '30.0', '40.0', '50.0', '60.0', '70.0', '80.0', '90.0', '100.0']
cb_tickloc = [0, 10.0, 20.0, 30.0, 40.0, 50.0, 60.0, 70.0, 80.0, 90.0, 100.0]

norm = mc.Normalize(vmin=vmin, vmax=vmax)

fig, ax = plt.subplots(figsize=(7, 7))
basemap = Basemap(llcrnrlat=minlat, urcrnrlat=maxlat, llcrnrlon=minlon, urcrnrlon=maxlon, resolution='i')
basemap.drawmapboundary(linewidth=0)
#oBaseMap.fillcontinents(color='gray', zorder=2)
basemap.drawparallels(np.arange(minlat, maxlat, 0.5), labels=[True, False, False, False], fontsize=10)
basemap.drawmeridians(np.arange(minlon, maxlon, 0.5), labels=[False, False, False, True], fontsize=10)

basedata = basemap.pcolormesh(lons, lats,
                              data, shading='flat', norm=norm, cmap=cmap_name)

cb_ax = fig.add_axes([0.1, 0.05, 0.8, 0.02])
cb = fig.colorbar(basedata, orientation='horizontal', cax=cb_ax, extend='both')
cb.set_label(cb_label, fontsize=15)
cb.ax.tick_params(labelsize=15)
cb.set_ticks(cb_tickloc)
cb.ax.set_xticklabels(cb_xticklabels)

filename = os.path.join(img_path, "ex_sp_ascat_snow_prob.tiff")
fig.savefig(filename, dpi=120)
```



B. Extract ASCAT, ERA5 and RZSM datasets

- Get data of ASCAT and RZSM
- resample the time-series
- generate SWI at different T
- rescale SWI using RZSM variables

In [30]:
```
# Rescale ASCAT time-series
ts_scale_ws = pd.DataFrame()
gpis_ws = zip(gpis_basin_ascat, lons_basin_ascat, lats_basin_ascat)

print(' => Rescale ASCAT time-series ... ')
for id, (gpi_ascat, lon_ascat, lat_ascat) in notebook.tqdm(enumerate(gpis_ws),
                                                          total=gpis_basin_ascat.__len__(),
                                                          desc=' ==== Rescale time-series progress'):

    # Select gpi for RZSM and ERA5
    gpi_rzsm = reader_rzsm.grid.find_nearest_gpi(lon_ascat, lat_ascat, max_dist=settings['max_dist'])[0]
    lon_rzsm, lat_rzsm = reader_rzsm.grid.gpi2lonlat(gpi_rzsm)

    gpi_era5 = reader_era5.grid.find_nearest_gpi(lon_ascat, lat_ascat, max_dist=settings['max_dist'])[0]
    lon_era5, lat_era5 = reader_era5.grid.gpi2lonlat(gpi_era5)

    # Info
    ts_sm = 'sm_' + str(gpi_ascat)
    ts_var40 = 'var40_' + str(gpi_ascat)
    ts_var41 = 'var41_' + str(gpi_ascat)
    ts_var42 = 'var42_' + str(gpi_ascat)
    ts_swi_t5 = 'swi_t5_' + str(gpi_ascat)
    ts_swi_t10 = 'swi_t10_' + str(gpi_ascat)
    ts_swi_t50 = 'swi_t50_' + str(gpi_ascat)
    ts_swi_t5_scaled_ms = 'swi_t5_scaled_ms_' + str(gpi_ascat)
    ts_swi_t10_scaled_ms = 'swi_t10_scaled_ms_' + str(gpi_ascat)
    ts_swi_t50_scaled_ms = 'swi_t50_scaled_ms_' + str(gpi_ascat)

    # Get data
    #print(' ==> Point: ' + str(id+1) + '/' + str(gpis_basin_ascat.shape[0]))
    ts_ascat = reader_ascat.read_ts(gpi_ascat)
    ts_ascat = ts_ascat.loc[settings['time_start']:settings['time_end']]
    ts_rzsm = reader_rzsm.read_ts(gpi_rzsm)
    ts_rzsm = ts_rzsm.loc[settings['time_start']:settings['time_end']]

    # Scale time-series
    ts_scale_ws[ts_sm] = ts_ascat.sm.resample('D').mean().dropna()
    ts_scale_ws[ts_var40] = ts_rzsm.var40.resample('D').mean().dropna()
    ts_scale_ws[ts_var41] = ts_rzsm.var41.resample('D').mean().dropna()
    ts_scale_ws[ts_var42] = ts_rzsm.var42.resample('D').mean().dropna()

    # Get julian dates of time series 1
    jd = ts_ascat_ws[ts_sm].index.to_julian_date().get_values()

    ts_scale_ws[ts_swi_t5]  = exp_filter(ts_scale_ws[ts_sm].values, jd, ctime=5)
    ts_scale_ws[ts_swi_t10]  = exp_filter(ts_scale_ws[ts_sm].values, jd, ctime=10)
    ts_scale_ws[ts_swi_t50]  = exp_filter(ts_scale_ws[ts_sm].values, jd, ctime=50)

    ts_scale_ws[ts_swi_t5_scaled_ms] = scaling_method_ms(ts_scale_ws[ts_swi_t5], ts_scale_ws[ts_var40])
    ts_scale_ws[ts_swi_t10_scaled_ms] = scaling_method_ms(ts_scale_ws[ts_swi_t10], ts_scale_ws[ts_var41])
    ts_scale_ws[ts_swi_t50_scaled_ms] = scaling_method_ms(ts_scale_ws[ts_swi_t50], ts_scale_ws[ts_var42])

print(' => Rescale ASCAT time-series ... DONE')
```

```
 => Rescale ASCAT time-series ...

 => Rescale ASCAT time-series ... DONE
```

Evaluation of SWI time-series

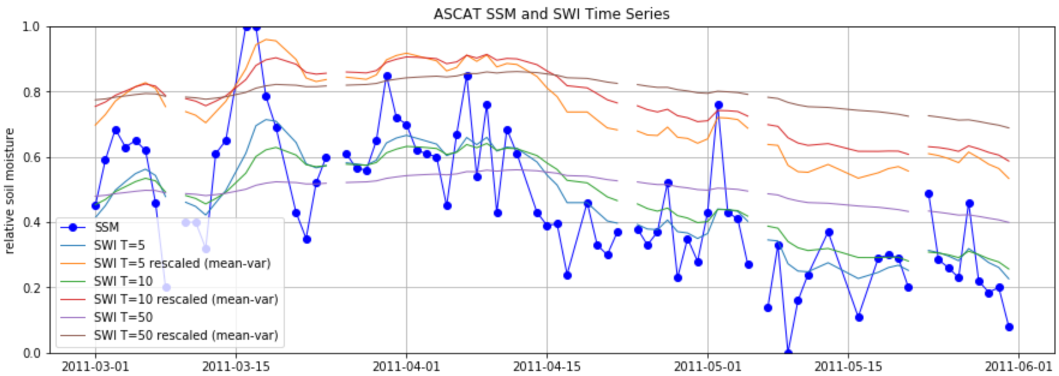
1. Select a ASCAT gpi to check the results of soil water index

In [31]:
```
# Get time-series for a gpi
gpi_id = 7
gpi_basin_ascat = gpis_basin_ascat[gpi_id]

# Info
ts_sm = 'sm_' + str(gpi_basin_ascat)
ts_var40 = 'var40_' + str(gpi_basin_ascat)
ts_var41 = 'var41_' + str(gpi_basin_ascat)
ts_var42 = 'var42_' + str(gpi_basin_ascat)
ts_swi_t5 = 'swi_t5_' + str(gpi_basin_ascat)
ts_swi_t10 = 'swi_t10_' + str(gpi_basin_ascat)
ts_swi_t50 = 'swi_t50_' + str(gpi_basin_ascat)
ts_swi_t5_scaled_ms = 'swi_t5_scaled_ms_' + str(gpi_basin_ascat)
ts_swi_t10_scaled_ms = 'swi_t10_scaled_ms_' + str(gpi_basin_ascat)
ts_swi_t50_scaled_ms = 'swi_t50_scaled_ms_' + str(gpi_basin_ascat)

ts_scale_id = pd.DataFrame()
ts_scale_id['sm'] = ts_scale_ws[ts_sm]
ts_scale_id['var40'] = ts_scale_ws[ts_var40]
ts_scale_id['var41'] = ts_scale_ws[ts_var41]
ts_scale_id['var42'] = ts_scale_ws[ts_var42]
ts_scale_id['swi_t5'] = ts_scale_ws[ts_swi_t5]
ts_scale_id['swi_t10'] = ts_scale_ws[ts_swi_t10]
ts_scale_id['swi_t50'] = ts_scale_ws[ts_swi_t50]
ts_scale_id['swi_t5_scaled_ms'] = ts_scale_ws[ts_swi_t5_scaled_ms]
ts_scale_id['swi_t10_scaled_ms'] = ts_scale_ws[ts_swi_t10_scaled_ms]
ts_scale_id['swi_t50_scaled_ms'] = ts_scale_ws[ts_swi_t50_scaled_ms]

# Print time-series for one gpi
print(ts_scale_id.head())
```

```
              sm     var40     var41     var42    swi_t5   swi_t10   swi_t50  \
2007-01-02  0.23  0.773956  0.751526  0.645782  0.230000  0.230000  0.230000
2007-01-04  0.10  0.486969  0.752014  0.647919  0.152171  0.158522  0.163700
2007-01-05  0.32  0.579102  0.719788  0.648743  0.223058  0.219557  0.217192
2007-01-06  0.20  0.418823  0.717682  0.649933  0.215211  0.213795  0.212743
2007-01-07  0.47  0.532043  0.697601  0.650909  0.290023  0.276730  0.266477

            swi_t5_scaled_ms  swi_t10_scaled_ms  swi_t50_scaled_ms
2007-01-02          0.537061           0.564703           0.507226
2007-01-04          0.469124           0.503970           0.436159
2007-01-05          0.531001           0.555830           0.493497
2007-01-06          0.524151           0.550934           0.488728
2007-01-07          0.589454           0.604408           0.546326
```

2. Plot SWI time-series to analyze the filtering method

In [32]:
```
# Plot ASCAT SSM, SWI-T5, SWI-T10, SWI-T50 variable(s)
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(ts_scale_id['2011-03':'2011-05']['sm'], lw=1, color='#0000FF', label='SSM', marker='o')
ax.plot(ts_scale_id['2011-03':'2011-05']['swi_t5'], lw=1, label='SWI T=5')
ax.plot(ts_scale_id['2011-03':'2011-05']['swi_t5_scaled_ms'], lw=1, label='SWI T=5 rescaled (mean-var)')
ax.plot(ts_scale_id['2011-03':'2011-05']['swi_t10'], lw=1, label='SWI T=10')
ax.plot(ts_scale_id['2011-03':'2011-05']['swi_t10_scaled_ms'], lw=1, label='SWI T=10 rescaled (mean-var)')
ax.plot(ts_scale_id['2011-03':'2011-05']['swi_t50'], lw=1, label='SWI T=50')
ax.plot(ts_scale_id['2011-03':'2011-05']['swi_t50_scaled_ms'], lw=1, label='SWI T=50 rescaled (mean-var)')

ax.set_ylim(0, 1)
ax.set_title('ASCAT SSM and SWI Time Series')
ax.set_ylabel('relative soil moisture')
ax.grid(b=True)
plt.legend()

filename = os.path.join(img_path, "ex_ts_ascat_swi_scaled_period.tiff")
fig.savefig(filename, dpi=120)
```



3. Analyze the time-series, plot the maps (sm, SWI and SWI scaled) for a day that can be investigated spatially, compare different SWI maps

- modify time_analysis
- modify time_window (if needed)

In [33]:
```
# Time analysis [YYYY-MM-DD HH:MM]
time_analysis = "2012-09-15 00:00"
time_window = 24 # hours

# Select data time-series using a time refernce step
pnt_scale_ws = pd.DataFrame(
    columns=['lon', 'lat', 'gpi', 'sm', 'var40', 'var41', 'var42',
             'swi_t5', 'swi_t10', 'swi_t50',
             'swi_t5_scaled_ms', 'swi_t10_scaled_ms', 'swi_t50_scaled_ms',
             'time'])
gpis_ws = zip(gpis_basin_ascat, lons_basin_ascat, lats_basin_ascat)

print(' => Analyze ASCAT from time-series to maps ... ')
for id, (gpi_ascat, lon_ascat, lat_ascat) in notebook.tqdm(enumerate(gpis_ws),
                                                           total=gpis_basin_ascat.__len__(),
                                                           desc=' ==== Analyze time-series progress'):

    # Info
    ts_sm = 'sm_' + str(gpi_ascat)
    ts_var40 = 'var40_' + str(gpi_ascat)
    ts_var41 = 'var41_' + str(gpi_ascat)
    ts_var42 = 'var42_' + str(gpi_ascat)
    ts_swi_t5 = 'swi_t5_' + str(gpi_ascat)
    ts_swi_t10 = 'swi_t10_' + str(gpi_ascat)
    ts_swi_t50 = 'swi_t50_' + str(gpi_ascat)
    ts_swi_t5_scaled_ms = 'swi_t5_scaled_ms_' + str(gpi_ascat)
    ts_swi_t10_scaled_ms = 'swi_t10_scaled_ms_' + str(gpi_ascat)
    ts_swi_t50_scaled_ms = 'swi_t50_scaled_ms_' + str(gpi_ascat)

    pnt_scale_sm = df_time_matching(ts_scale_ws[ts_sm],
                                    time_analysis, window=time_window)[0]
    pnt_scale_var40 = df_time_matching(ts_scale_ws[ts_var40],
                                       time_analysis, window=time_window)[0]
    pnt_scale_var41 = df_time_matching(ts_scale_ws[ts_var41],
                                       time_analysis, window=time_window)[0]
    pnt_scale_var42 = df_time_matching(ts_scale_ws[ts_var42],
                                       time_analysis, window=time_window)[0]

    pnt_scale_swi_t5 = df_time_matching(ts_scale_ws[ts_swi_t5],
                                           time_analysis, window=time_window)[0]
    pnt_scale_swi_t10 = df_time_matching(ts_scale_ws[ts_swi_t10],
                                            time_analysis, window=time_window)[0]
    pnt_scale_swi_t50 = df_time_matching(ts_scale_ws[ts_swi_t50],
                                            time_analysis, window=time_window)[0]

    pnt_scale_swi_t5_scaled_ms = df_time_matching(ts_scale_ws[ts_swi_t5_scaled_ms],
                                                  time_analysis, window=time_window)[0]
    pnt_scale_swi_t10_scaled_ms = df_time_matching(ts_scale_ws[ts_swi_t10_scaled_ms],
                                                   time_analysis, window=time_window)[0]
    pnt_scale_swi_t50_scaled_ms = df_time_matching(ts_scale_ws[ts_swi_t50_scaled_ms],
                                                   time_analysis, window=time_window)[0]

    pnt_scale_ws = pnt_scale_ws.append(
        {'lon': lon_ascat, 'lat': lat_ascat, 'gpi': gpi_ascat,
         'sm': pnt_scale_sm,
         'var40': pnt_scale_var40, 'var41': pnt_scale_var41, 'var42': pnt_scale_var42,
         'swi_t5': pnt_scale_swi_t5, 'swi_t10': pnt_scale_swi_t10, 'swi_t50': pnt_scale_swi_t50,
         'swi_t5_scaled_ms': pnt_scale_swi_t5_scaled_ms,
         'swi_t10_scaled_ms': pnt_scale_swi_t10_scaled_ms,
         'swi_t50_scaled_ms': pnt_scale_swi_t50_scaled_ms,
         'time': time_analysis}, ignore_index=True)

# Remove NaN
pnt_scale_ws = pnt_scale_ws.dropna()

print(' => Analyze ASCAT from time-series to maps ... DONE')
```

```
 => Analyze ASCAT from time-series to maps ...

 => Analyze ASCAT from time-series to maps ... DONE
```

In [34]:
```
# Print point(s)
print(pnt_scale_ws.head(n=3)); print(pnt_scale_ws.tail(n=3));
```

```
         lon        lat      gpi    sm     var40     var41     var42  \
0  11.891937  42.660675  2204747  0.30  0.720612  0.563324  0.447662
1  12.044397  42.660675  2204751  0.35  0.761475  0.592438  0.463318
2  12.196858  42.660675  2204755  0.34  0.799377  0.627869  0.472931

     swi_t5   swi_t10   swi_t50  swi_t5_scaled_ms  swi_t10_scaled_ms  \
0  0.372612  0.315529  0.172028          0.622359           0.603571
1  0.385289  0.305126  0.157972          0.658326           0.615121
2  0.394478  0.306784  0.146899          0.703365           0.653041

   swi_t50_scaled_ms              time
0           0.454850  2012-09-15 00:00
1           0.465326  2012-09-15 00:00
2           0.496357  2012-09-15 00:00
          lon        lat      gpi    sm     var40     var41     var42  \
79  12.578540  43.785828  2251599  0.56  0.867096  0.798920  0.345734
80  12.733830  43.785828  2251603  0.55  0.867096  0.798920  0.345734
81  12.889121  43.785828  2251607  0.55  0.875061  0.796356  0.351379

      swi_t5   swi_t10   swi_t50  swi_t5_scaled_ms  swi_t10_scaled_ms  \
79  0.509733  0.446390  0.232699          0.811752           0.757496
80  0.522510  0.446703  0.222525          0.820745           0.756870
81  0.552937  0.464752  0.234582          0.817100           0.744239

    swi_t50_scaled_ms              time
79           0.463988  2012-09-15 00:00
80           0.457413  2012-09-15 00:00
81           0.464769  2012-09-15 00:00
```

4. Interpolation of ASCAT values for selected date (nearest neighbour method)

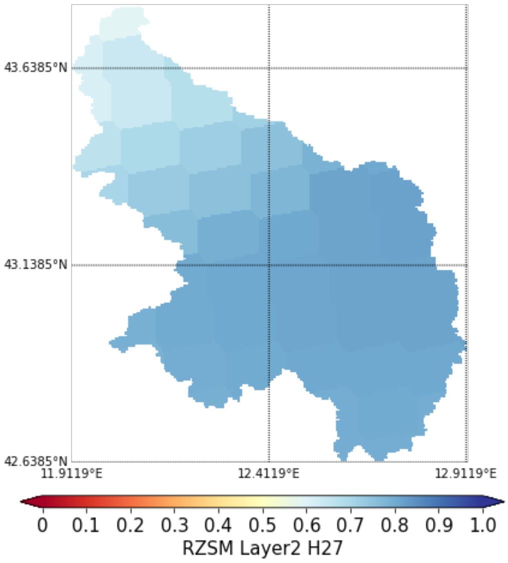
- All variables (sm, SWI, SWI scaled) are interpolated over basin domain
- figures are created and maps are saved as geotiff

In [35]:
```
# Interpolate values over domain
map_scale_swi_t5 = interpolate_point2map(
    pnt_scale_ws['lon'].values, pnt_scale_ws['lat'].values,
                            pnt_scale_ws['swi_t5'].values, basin_lons_2d, basin_lats_2d)
map_scale_swi_t10 = interpolate_point2map(
    pnt_scale_ws['lon'].values, pnt_scale_ws['lat'].values,
                            pnt_scale_ws['swi_t10'].values, basin_lons_2d, basin_lats_2d)
map_scale_swi_t50 = interpolate_point2map(
    pnt_scale_ws['lon'].values, pnt_scale_ws['lat'].values,
                            pnt_scale_ws['swi_t50'].values, basin_lons_2d, basin_lats_2d)
map_scale_swi_t5_scaled_ms = interpolate_point2map(
    pnt_scale_ws['lon'].values, pnt_scale_ws['lat'].values,
                            pnt_scale_ws['swi_t5_scaled_ms'].values, basin_lons_2d, basin_lats_2d)
map_scale_swi_t10_scaled_ms = interpolate_point2map(
    pnt_scale_ws['lon'].values, pnt_scale_ws['lat'].values,
                            pnt_scale_ws['swi_t10_scaled_ms'].values, basin_lons_2d, basin_lats_2d)
map_scale_swi_t50_scaled_ms = interpolate_point2map(
    pnt_scale_ws['lon'].values, pnt_scale_ws['lat'].values,
                            pnt_scale_ws['swi_t50_scaled_ms'].values, basin_lons_2d, basin_lats_2d)
map_scale_var40 = interpolate_point2map(
    pnt_scale_ws['lon'].values, pnt_scale_ws['lat'].values,
                            pnt_scale_ws['var40'].values, basin_lons_2d, basin_lats_2d)
map_scale_var41 = interpolate_point2map(
    pnt_scale_ws['lon'].values, pnt_scale_ws['lat'].values,
                            pnt_scale_ws['var41'].values, basin_lons_2d, basin_lats_2d)
map_scale_var42 = interpolate_point2map(
    pnt_scale_ws['lon'].values, pnt_scale_ws['lat'].values,
                            pnt_scale_ws['var42'].values, basin_lons_2d, basin_lats_2d)

# Save map(s) in a Tiff format (to plot file using QGIS)
create_map(map_scale_swi_t5, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_ascat_swit5_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
create_map(map_scale_swi_t10, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_ascat_swit10_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
create_map(map_scale_swi_t50, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_ascat_swit50_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
create_map(map_scale_swi_t5_scaled_ms, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_ascat_swit5_scaled_ms_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
create_map(map_scale_swi_t10_scaled_ms, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_ascat_swit10_scaled_ms_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
create_map(map_scale_swi_t50_scaled_ms, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_ascat_swit50_scaled_ms_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
create_map(map_scale_var40, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_rzsm_var40_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
create_map(map_scale_var41, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_rzsm_var41_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
create_map(map_scale_var42, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_rzsm_var42_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
```

Plot maps

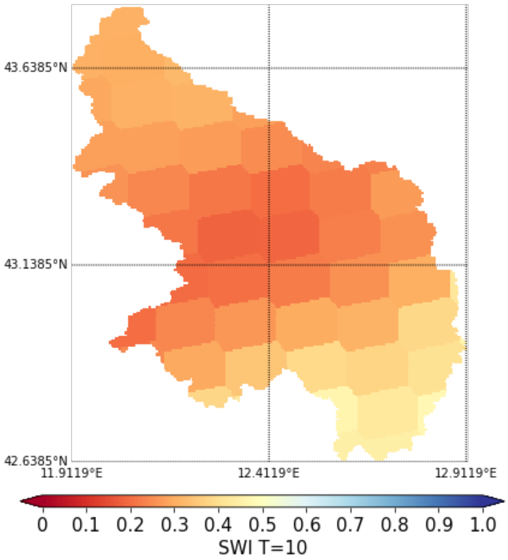
1. Plot map of RZSM var41 (H27)

In [36]:
```
# Plot map of H27
data = map_scale_var41
lons = basin_lons_2d
lats = np.flipud(basin_lats_2d)

vmin = 0
vmax = 1
cb_label='RZSM Layer2 H27'

cmap_name='RdYlBu'

minlat = basin_bbox.bottom
maxlat = basin_bbox.top
minlon = basin_bbox.left
maxlon = basin_bbox.right

cb_xticklabels = ['0', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7', '0.8', '0.9', '1.0']
cb_tickloc = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

norm = mc.Normalize(vmin=vmin, vmax=vmax)

fig, ax = plt.subplots(figsize=(7, 7))
basemap = Basemap(llcrnrlat=minlat, urcrnrlat=maxlat, llcrnrlon=minlon, urcrnrlon=maxlon, resolution='i')
basemap.drawmapboundary(linewidth=0)
#oBaseMap.fillcontinents(color='gray', zorder=2)
basemap.drawparallels(np.arange(minlat, maxlat, 0.5), labels=[True, False, False, False], fontsize=10)
basemap.drawmeridians(np.arange(minlon, maxlon, 0.5), labels=[False, False, False, True], fontsize=10)

basedata = basemap.pcolormesh(lons, lats,
                              data, shading='flat', norm=norm, cmap=cmap_name)

cb_ax = fig.add_axes([0.1, 0.05, 0.8, 0.02])
cb = fig.colorbar(basedata, orientation='horizontal', cax=cb_ax, extend='both')
cb.set_label(cb_label, fontsize=15)
cb.ax.tick_params(labelsize=15)
cb.set_ticks(cb_tickloc)
cb.ax.set_xticklabels(cb_xticklabels)

filename = os.path.join(img_path, "ex_sp_rzsm_var41.tiff")
fig.savefig(filename, dpi=120)
```



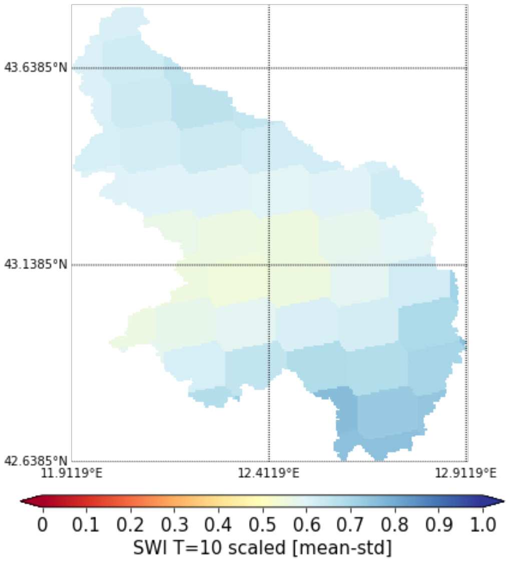
2. Plot map of ASCAT SWI T=10

In [37]:
```
# Plot map of ASCAT SWI_10
data = map_scale_swi_t10
lons = basin_lons_2d
lats = np.flipud(basin_lats_2d)

vmin = 0
vmax = 1
cb_label='SWI T=10'

cmap_name='RdYlBu'

minlat = basin_bbox.bottom
maxlat = basin_bbox.top
minlon = basin_bbox.left
maxlon = basin_bbox.right

cb_xticklabels = ['0', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7', '0.8', '0.9', '1.0']
cb_tickloc = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

norm = mc.Normalize(vmin=vmin, vmax=vmax)

fig, ax = plt.subplots(figsize=(7, 7))
basemap = Basemap(llcrnrlat=minlat, urcrnrlat=maxlat, llcrnrlon=minlon, urcrnrlon=maxlon, resolution='i')
basemap.drawmapboundary(linewidth=0)
#oBaseMap.fillcontinents(color='gray', zorder=2)
basemap.drawparallels(np.arange(minlat, maxlat, 0.5), labels=[True, False, False, False], fontsize=10)
basemap.drawmeridians(np.arange(minlon, maxlon, 0.5), labels=[False, False, False, True], fontsize=10)

basedata = basemap.pcolormesh(lons, lats,
                              data, shading='flat', norm=norm, cmap=cmap_name)

cb_ax = fig.add_axes([0.1, 0.05, 0.8, 0.02])
cb = fig.colorbar(basedata, orientation='horizontal', cax=cb_ax, extend='both')
cb.set_label(cb_label, fontsize=15)
cb.ax.tick_params(labelsize=15)
cb.set_ticks(cb_tickloc)
cb.ax.set_xticklabels(cb_xticklabels)

filename = os.path.join(img_path, "ex_sp_ascat_swit10.tiff")
fig.savefig(filename, dpi=120)
```



3. Plot map of ASCAT SWI T=10 Scaled

In [38]:
```
# Plot map of ASCAT SWI_10 rescaled
data = map_scale_swi_t10_scaled_ms
lons = basin_lons_2d
lats = np.flipud(basin_lats_2d)

vmin = 0
vmax = 1
cb_label='SWI T=10 scaled [mean-std]'

cmap_name='RdYlBu'

minlat = basin_bbox.bottom
maxlat = basin_bbox.top
minlon = basin_bbox.left
maxlon = basin_bbox.right

cb_xticklabels = ['0', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7', '0.8', '0.9', '1.0']
cb_tickloc = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

norm = mc.Normalize(vmin=vmin, vmax=vmax)

fig, ax = plt.subplots(figsize=(7, 7))
basemap = Basemap(llcrnrlat=minlat, urcrnrlat=maxlat, llcrnrlon=minlon, urcrnrlon=maxlon, resolution='i')
basemap.drawmapboundary(linewidth=0)
#oBaseMap.fillcontinents(color='gray', zorder=2)
basemap.drawparallels(np.arange(minlat, maxlat, 0.5), labels=[True, False, False, False], fontsize=10)
basemap.drawmeridians(np.arange(minlon, maxlon, 0.5), labels=[False, False, False, True], fontsize=10)

basedata = basemap.pcolormesh(lons, lats,
                              data, shading='flat', norm=norm, cmap=cmap_name)

cb_ax = fig.add_axes([0.1, 0.05, 0.8, 0.02])
cb = fig.colorbar(basedata, orientation='horizontal', cax=cb_ax, extend='both')
cb.set_label(cb_label, fontsize=15)
cb.ax.tick_params(labelsize=15)
cb.set_ticks(cb_tickloc)
cb.ax.set_xticklabels(cb_xticklabels)

filename = os.path.join(img_path, "ex_sp_ascat_swit10_scaled_ms.tiff")
fig.savefig(filename, dpi=120)
```



C. Extract ASCAT, ERA5 and RZSM datasets

- Get data of ASCAT and ERA5
- resample the time-series

In [39]:
```
# Get ASCAT and ERA5 time-series
ts_rain_ws = pd.DataFrame()
gpis_ws = zip(gpis_basin_ascat, lons_basin_ascat, lats_basin_ascat)

print(' => Get ASCAT and ERA5 time-series ... ')
for id, (gpi_ascat, lon_ascat, lat_ascat) in notebook.tqdm(enumerate(gpis_ws),
                                                           total=gpis_basin_ascat.__len__(),
                                                           desc=' ==== Get time-series progress'):

    # Select gpi for RZSM and ERA5
    gpi_rzsm = reader_rzsm.grid.find_nearest_gpi(lon_ascat, lat_ascat, max_dist=settings['max_dist'])[0]
    lon_rzsm, lat_rzsm = reader_rzsm.grid.gpi2lonlat(gpi_rzsm)

    gpi_era5 = reader_era5.grid.find_nearest_gpi(lon_ascat, lat_ascat, max_dist=settings['max_dist'])[0]
    lon_era5, lat_era5 = reader_era5.grid.gpi2lonlat(gpi_era5)

    # Info
    ts_sm = 'sm_' + str(gpi_ascat)
    ts_tp = 'tp_' + str(gpi_ascat)

    # Get data
    #print(' ==> Point: ' + str(id+1) + '/' + str(gpis_basin_ascat.shape[0]))
    ts_ascat = reader_ascat.read_ts(gpi_ascat)
    ts_ascat = ts_ascat.loc[settings['time_start']:settings['time_end']]
    ts_era5 = reader_era5.read_ts(gpi_era5)
    ts_era5 = ts_era5.loc[settings['time_start']:settings['time_end']]

    # Scale time-series
    ts_rain_ws[ts_sm] = ts_ascat.sm.resample('D').mean().dropna()
    ts_rain_ws[ts_tp] = ts_era5.tp.resample('D').sum().dropna()

print(' => Get ASCAT and ERA5 time-series ... DONE')
```

```
=> Get ASCAT and ERA5 time-series ...

=> Get ASCAT and ERA5 time-series ... DONE
```

1. Analyze the time-series, plot the maps (sm and total precipitation) for a day that can be investigated spatially

- modify time_analysis
- modify time_window (if needed)
- modify accum_window to vary accumulation period for rainfall

In [40]:
```
# Time analysis [YYYY-MM-DD HH:MM]
time_analysis = "2012-08-31 00:00"
time_window = 24 # hours
accum_window = 1 # days

idx = pd.date_range(end=time_analysis, periods=accum_window, freq='D')

# Select data time-series using a time refernce step
pnt_rain_ws = pd.DataFrame(columns=['lon', 'lat', 'gpi', 'sm', 'tp', 'time'])
gpis_ws = zip(gpis_basin_ascat, lons_basin_ascat, lats_basin_ascat)

print(' => Analyze ASCAT and ERA5 from time-series to maps ... ')
for id, (gpi_ascat, lon_ascat, lat_ascat) in notebook.tqdm(enumerate(gpis_ws),
                                                           total=gpis_basin_ascat.__len__(),
                                                           desc=' ==== Analyze time-series progress'):

    # Info
    ts_sm = 'sm_' + str(gpi_ascat)
    ts_tp = 'tp_' + str(gpi_ascat)

    pnt_rain_sm = df_time_matching(ts_rain_ws[ts_sm],
                                   time_analysis, window=time_window)[0]

    ts_rain_sel = pd.DataFrame(ts_rain_ws[ts_tp], index=idx)
    pnt_rain_tp = ts_rain_sel[ts_tp].sum()

    pnt_rain_ws = pnt_rain_ws.append(
        {'lon': lon_ascat, 'lat': lat_ascat, 'gpi': gpi_ascat,
         'sm': pnt_rain_sm, 'tp': pnt_rain_tp,
         'time': time_analysis}, ignore_index=True)

# Remove NaN
pnt_rain_ws = pnt_rain_ws.dropna()

print(' => Analyze ASCAT and ERA5 from time-series to maps ... DONE')
```

```
=> Analyze ASCAT and ERA5 from time-series to maps ...

=> Analyze ASCAT and ERA5 from time-series to maps ... DONE
```

In [41]:
```
# Print point(s)
print(pnt_rain_ws.head(n=3)); print(pnt_rain_ws.tail(n=3));
```

```
         lon        lat      gpi    sm         tp              time
0  11.891937  42.660675  2204747  0.51  15.676260  2012-08-31 00:00
1  12.044397  42.660675  2204751  0.49  15.676260  2012-08-31 00:00
2  12.196858  42.660675  2204755  0.30   8.305132  2012-08-31 00:00
          lon        lat      gpi    sm        tp              time
79  12.578540  43.785828  2251599  0.91  7.950485  2012-08-31 00:00
80  12.733830  43.785828  2251603  0.58  3.222227  2012-08-31 00:00
81  12.889121  43.785828  2251607  0.28  2.577543  2012-08-31 00:00
```

2. Interpolation of ASCAT and ERA5 values for selected date (nearest neighbour method)

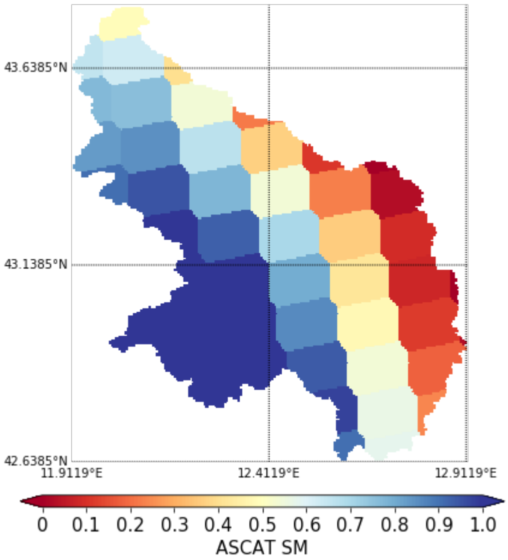
- all variables (sm and tp) are interpolated over basin domain
- figures are created and maps are saved as geotiff

In [42]:
```
# Interpolate values over domain
map_rain_sm = interpolate_point2map(
    pnt_rain_ws['lon'].values, pnt_rain_ws['lat'].values, pnt_rain_ws['sm'].values,
    basin_lons_2d, basin_lats_2d)
map_rain_tp = interpolate_point2map(
    pnt_rain_ws['lon'].values, pnt_rain_ws['lat'].values, pnt_rain_ws['tp'].values,
    basin_lons_2d, basin_lats_2d)

# Save map(s) in a Tiff format (to plot file using QGIS)
create_map(map_rain_tp, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_era5_sm_tp_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
create_map(map_rain_sm, basin_rows, basin_cols, basin_epsg, basin_transform,
           file_name_data=os.path.join(img_path, "ex_sp_ascat_sm_tp_qgis.tiff"),
           file_name_mask=os.path.join(ancillary_path, "basin_domain.tiff"))
```

Plot maps

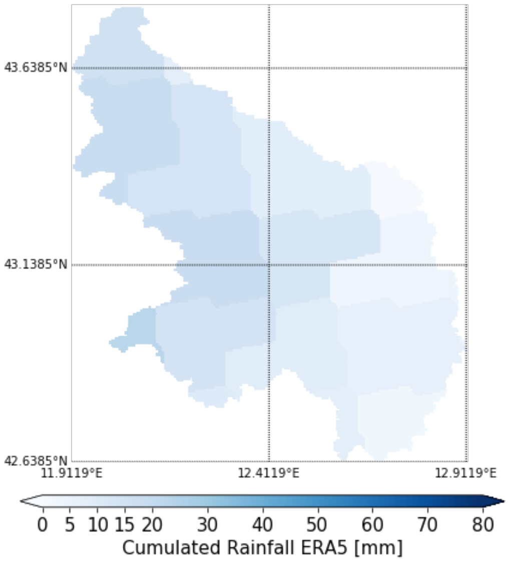
1. Plot ASCAT soil moisture H113

In [43]:
```
# Plot map of ASCAT SM
data = map_rain_sm
lons = basin_lons_2d
lats = np.flipud(basin_lats_2d)

vmin = 0
vmax = 1
cb_label='ASCAT SM'

cmap_name='RdYlBu'

minlat = basin_bbox.bottom
maxlat = basin_bbox.top
minlon = basin_bbox.left
maxlon = basin_bbox.right

cb_xticklabels = ['0', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7', '0.8', '0.9', '1.0']
cb_tickloc = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

norm = mc.Normalize(vmin=vmin, vmax=vmax)

fig, ax = plt.subplots(figsize=(7, 7))
basemap = Basemap(llcrnrlat=minlat, urcrnrlat=maxlat, llcrnrlon=minlon, urcrnrlon=maxlon, resolution='i')
basemap.drawmapboundary(linewidth=0)
#oBaseMap.fillcontinents(color='gray', zorder=2)
basemap.drawparallels(np.arange(minlat, maxlat, 0.5), labels=[True, False, False, False], fontsize=10)
basemap.drawmeridians(np.arange(minlon, maxlon, 0.5), labels=[False, False, False, True], fontsize=10)

basedata = basemap.pcolormesh(lons, lats,
                              data, shading='flat', norm=norm, cmap=cmap_name)

cb_ax = fig.add_axes([0.1, 0.05, 0.8, 0.02])
cb = fig.colorbar(basedata, orientation='horizontal', cax=cb_ax, extend='both')
cb.set_label(cb_label, fontsize=15)
cb.ax.tick_params(labelsize=15)
cb.set_ticks(cb_tickloc)
cb.ax.set_xticklabels(cb_xticklabels)

filename = os.path.join(img_path, "ex_sp_ascat_sm_tp.tiff")
fig.savefig(filename, dpi=120)
```



2. Plot ERA5 cumulated rainfall

In [44]:
```
# Plot map of cumulated rainfall
data = map_rain_tp
lons = basin_lons_2d
lats = np.flipud(basin_lats_2d)

vmin = 0
vmax = 80
cb_label='Cumulated Rainfall ERA5 [mm]'

cmap_name='Blues'

minlat = basin_bbox.bottom
maxlat = basin_bbox.top
minlon = basin_bbox.left
maxlon = basin_bbox.right

cb_xticklabels = ['0', '5', '10', '15', '20', '30', '40', '50', '60', '70', '80']
cb_tickloc = [0, 5.0, 10.0, 15.0, 20.0, 30.0, 40.0, 50.0, 60.0, 70.0, 80.0]

norm = mc.Normalize(vmin=vmin, vmax=vmax)

fig, ax = plt.subplots(figsize=(7, 7))
basemap = Basemap(llcrnrlat=minlat, urcrnrlat=maxlat, llcrnrlon=minlon, urcrnrlon=maxlon, resolution='i')
basemap.drawmapboundary(linewidth=0)
#oBaseMap.fillcontinents(color='gray', zorder=2)
basemap.drawparallels(np.arange(minlat, maxlat, 0.5), labels=[True, False, False, False], fontsize=10)
basemap.drawmeridians(np.arange(minlon, maxlon, 0.5), labels=[False, False, False, True], fontsize=10)

basedata = basemap.pcolormesh(lons, lats,
                              data, shading='flat', norm=norm, cmap=cmap_name)

cb_ax = fig.add_axes([0.1, 0.05, 0.8, 0.02])
cb = fig.colorbar(basedata, orientation='horizontal', cax=cb_ax, extend='both')
cb.set_label(cb_label, fontsize=15)
cb.ax.tick_params(labelsize=15)
cb.set_ticks(cb_tickloc)
cb.ax.set_xticklabels(cb_xticklabels)

filename = os.path.join(img_path, "ex_sp_era5_sm_tp.tiff")
fig.savefig(filename, dpi=120)
```



On-the-job Training:

- Visualization and comparison of soil moisture spatial maps
- Visualization of spatial quality flags (ASCAT)
- Analysis of the periods/regions (in time/space) to be masked out due to flag poor quality (if any) [ASCAT Product]
- Application of the Soil Water Index to ASCAT soil moisture product for different T-values
- Application of rescaling techniques to make soil moisture maps in the same range of values
- Comparison of cumulated precipitation and soil moisture maps

In [ ]: