GDAL CLI Tutorial Workshop

Even Rouault and Seth Girvin

03 juil. 2026

CONTENTS

CHAPTER

ONE

INTRODUCTION

GDAL 3.11 introduced a new command line interface (CLI), simply called "gdal", supplementing the traditional well-known GDAL utilities (gdal_translate, ogr2ogr, etc.), to provide users with a more uniform, predictable and user-friendly experience. This workshop will give the opportunity to participants to get a hands-on experience to discover the capabilities of the new CLI through a series of exercises, including how to leverage them from Python.

1.1 Target audience

Suitable for those new to GDAL, as well as those already experienced with the traditional utilities and wishing to get up to speed with the new CLI. Some familiarity with geospatial raster and vector data and coordinate systems is assumed but not strictly required. Participants should not be afraid of command line use! Some SQL and Python knowledge will be useful for advanced exercises.

1.2 Content

We will explore the general principles of the CLI and apply them to the various algorithms it offers:

- exploring the contents and metadata of raster and vector datasets,
- performing file format transformations,
- subsetting, resampling, reprojection
- mosaicing and tiling raster datasets
- merging and partitioning vector datasets
- pixel operations
- DEM processing
- building virtual rasters and mosaics
- querying vector layers
- multidimensional dataset management
- Virtual System Interface (VSI) commands to list and copy files

We will also cover more advanced topics, such as basic processing pipelines, replayable pipelines (.gdalg.json files), nested pipelines, tee operator, and explore how to use the new capabilities from Python.

1.3 Copyright and License

CHAPTER

TWO

PRE-REQUISITES: GDAL INSTALLATION AND WORKSHOP SAMPLE DATASET DOWNLOAD

This workshop requires GDAL 3.13.1, released June 2026.

The suggested installation procedure is to use GDAL Conda builds. Conda is a system package management system that works on all major desktop operating system (Linux, Windows, MacOS X). It is mainly aimed at the Python ecosystem, but with a strong focus on tackling correctly the issue of software with native dependencies such as GDAL.

2.1 Linux

You have the choice between:

- using a Docker image
- or using Conda in your host environment to install GDAL and additional packages, and download test datasets.

The former is easier if you have Docker/Podman already installed on your system. The later involves more manual steps but is better integrated with your system.

2.1.1 Docker image containing binaries and workshop sample datasets

Assuming you have Docker or Podman already installed.

```
$ docker pull ghcr.io/rouault/gdal-cli-workshop
```

Allow X client from inside Docker to connect to the X server with:

```
$ xhost +local:root

non-network local connections being added to access control list
```

Now run the image with:

```
$ mkdir -p $HOME/gdal-cli-workshop
$ docker run -it --name gdal-cli-workshop \
        -v /tmp/.X11-unix:/tmp/.X11-unix \
        -v $HOME/gdal-cli-workshop:/data/gdal_cli_workshop_data-master/host \
        ghcr.io/rouault/gdal-cli-workshop
```

Inside the container:

```
(base) root@XXXXXXXXXXXX:/# gdal --version
```

```
GDAL 3.13.1 "Iowa City", released 2026/06/01
```

The datasets are there:

```
(base) root@XXXXXXXXXXXX:/# cd /data/gdal_cli_workshop_data-master/
(base) root@XXXXXXXXXXXX:/# ls -al
```

```
total 91296
drwxr-xr-x 5 root root     4096 May 22 01:37 .
drwxr-xr-x 3 root root     4096 May 22 04:12 ..
-rw-r--r-- 1 root root 18925846 May 22 01:37 20260519_00_tmp2m.nc
-rw-r--r-- 1 root root 18925846 May 22 01:37 20260519_06_tmp2m.nc
-rw-r--r-- 1 root root     1756 May 22 01:37 README.md
drwxr-xr-x 6 root root     4096 May 22 01:37 S2B_MSIL2A_20260423T094029_N0512_R036_
↪T34TDR_20260423T115714.SAFE
drwxr-xr-x 6 root root     4096 May 22 01:37 S2B_MSIL2A_20260423T094029_N0512_R036_
↪T34TER_20260423T115714.SAFE
drwxr-xr-x 6 root root     4096 May 22 01:37 S2B_MSIL2A_20260423T094029_N0512_R036_
↪T34TES_20260423T115714.SAFE
-rw-r--r-- 1 root root 35131910 May 22 01:37 dem.tif
-rw-r--r-- 1 root root 14909524 May 22 01:37 ne_10m_admin_1_states_provinces.zip
-rw-r--r-- 1 root root     5493 May 22 01:37 osm_conf_amenity.ini
-rw-r--r-- 1 root root  5528366 May 22 01:37 timisoara.osm.pbf
drwxrwxr-x 2 1000 1000     4096 May 22 05:06 host
```

Make sure that QGIS also starts:

```
(base) root@XXXXXXXXXXXX:/# qgis
```

You are done for the installation and can skip everything else in this page!

2.1.2 Conda installation

If you already have a Conda installation, skip this paragraph.

Download the Miniforge3 installer:

```
$ curl -LO https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-
↪Linux-x86_64.sh
```

Install it:

```
$ sh Miniforge3-Linux-x86_64.sh -b
```

will output something like:

```
PREFIX=/home/even/miniforge3
Unpacking bootstrapper...
Unpacking payload...
Extracting ca-certificates-2026.4.22-hbd8a1cb_0.conda
Extracting libgomp-15.2.0-he0feb66_19.conda
Extracting libzlib-1.3.2-h25fd6f3_2.conda
Extracting nlohmann_json-abi-3.12.0-h0f90c79_1.conda
[ ... snip ... ]
```

(continues on next page)

(continued from previous page)

```
Linking conda-package-handling-2.4.0-pyh7900ff3_2
Linking conda-26.3.2-py313h78bf25f_1

Transaction finished

installation finished.
```

Activate Miniforge3 for the current shell

```
~/miniforge3/bin/conda init
```

will output something like:

```
no change     /home/even/miniforge3/condabin/conda
no change     /home/even/miniforge3/bin/conda
no change     /home/even/miniforge3/bin/activate
no change     /home/even/miniforge3/bin/deactivate
no change     /home/even/miniforge3/etc/profile.d/conda.sh
no change     /home/even/miniforge3/etc/fish/conf.d/conda.fish
no change     /home/even/miniforge3/shell/condabin/Conda.psm1
no change     /home/even/miniforge3/shell/condabin/conda-hook.ps1
no change     /home/even/miniforge3/lib/python3.13/site-packages/xontrib/conda.xsh
no change     /home/even/miniforge3/etc/profile.d/conda.csh
modified      /home/even/.bashrc

==> For changes to take effect, close and re-open your current shell. <==
```

2.1.3 GDAL installation in a dedicated conda environment

First, we will create a Conda "environment" for the purpose of this workshop, and will call it "gdal-cli-workshop". A Conda environment is a kind of workspace where you can install a set of packages that will not interfere with the ones of other environments. We use the "conda-forge" channel to get up-to-date official releases from the conda community (if using the Miniforge3 installer, this is not needed).

```
$ conda create -y --name gdal-cli-workshop -c conda-forge
```

```
Retrieving notices: done
Channels:
 - conda-forge
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/even/miniforge3/envs/gdal-cli-workshop




Downloading and Extracting Packages:
```

(continues on next page)

(continued from previous page)

```
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate gdal-cli-workshop
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

As suggested, you need to activate the newly created environment with:

```
$ conda activate gdal-cli-workshop
```

When an environment is activated, new lines in the shell are prefixed with "(name_of_environment)".

Now, we can finally install GDAL ! We ask to install both the library and command line utilities with the `libgdal` meta package (that installs `libgdal-core` and most optional driver plugins such as `libgdal-jp2openjpeg`, etc), and the `gdal` package with the Python bindings and scripts.

```
(gdal) $ conda install -y -c conda-forge libgdal gdal
```

```
Channels:
 - conda-forge
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/even/miniforge3/envs/gdal-cli-workshop

  added / updated specs:
    - gdal
    - libgdal


The following packages will be downloaded:

[ ... snip ... ]

Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Now let's check we got the GDAL we expected:

```
$ gdal --version
```

```
GDAL 3.13.1 "Iowa City", released 2026/06/01
```

2.2 MacOS X

Note

The instructions for MacOS X are a bit succinct, due to lack of access to that platform.

2.2.1 Conda installation

If you already have a Conda installation, skip that paragraph. Otherwise follow the instructions at https://conda-forge.org/download/ and download the installer corresponding to your CPU architecture.

2.2.2 GDAL installation in a dedicated conda environment

Please follow *Linux instructions* which should apply to MacOS X as well.

2.3 Windows

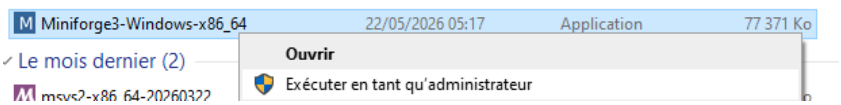
(tested on Windows 10, hopefully valid for Windows 11)

2.3.1 Conda installation

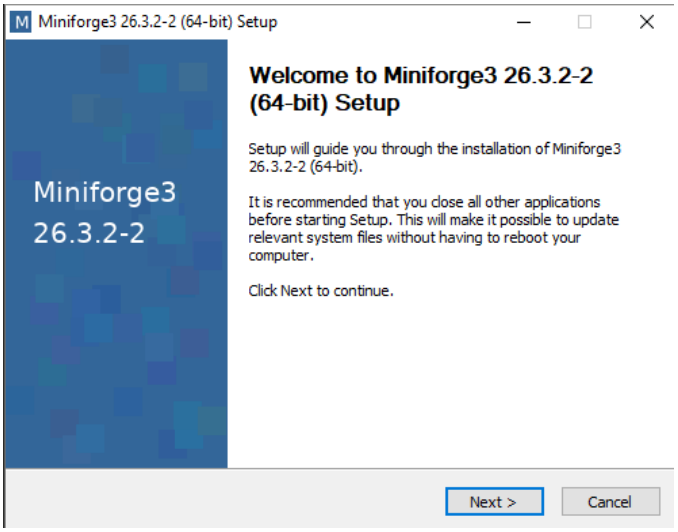
If you already have a Conda installation, you can skip this paragraph. However, installing a separate Miniforge3 instance for this workshop is recommended to avoid any interference with your existing Conda installation. It will also allow you to use the launcher script described in the next section without modification.

Download the Miniforge3 installer at https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-Windows-x86_64.exe

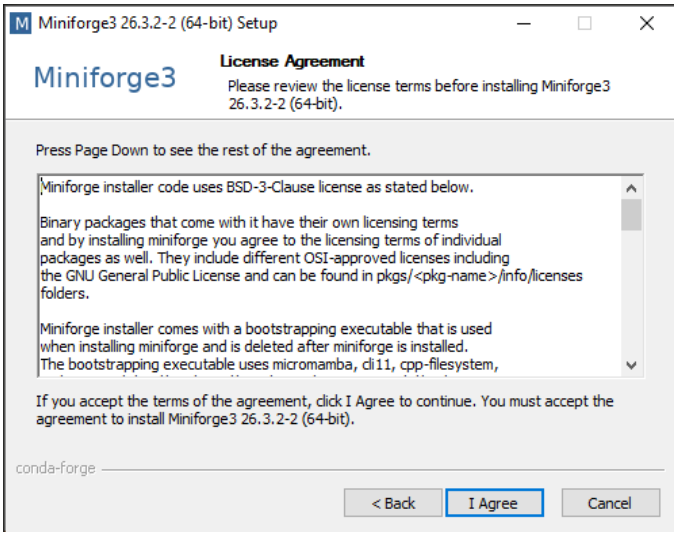
And execute it as an administrator, typically by right-clicking on the executable file and select "Run as administrator". Running as administrator is to make sure that you have permissions to install into `c:\gdal` so that later instructions can be applied without modification.



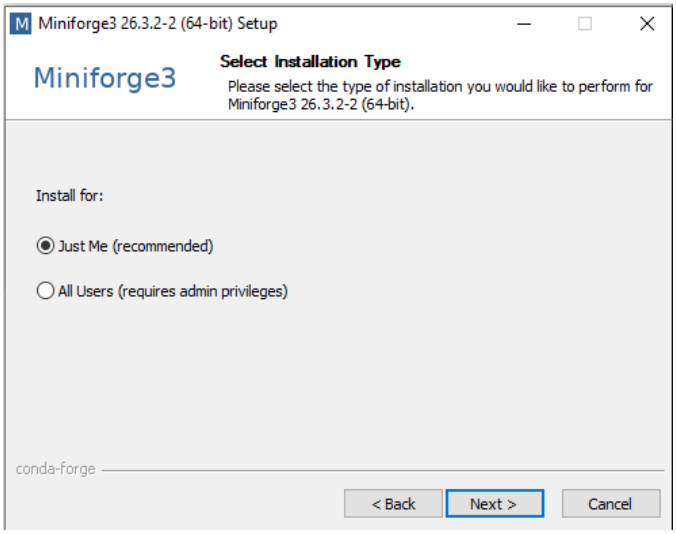
Validate the Welcome dialog:



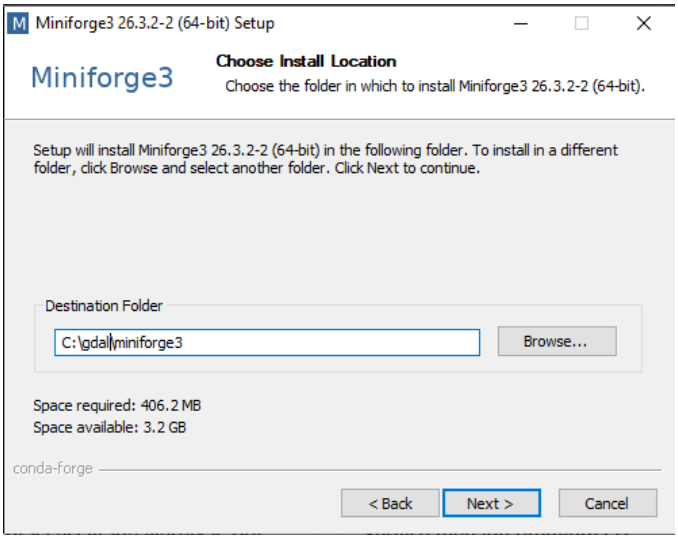
Accept the license agreement:



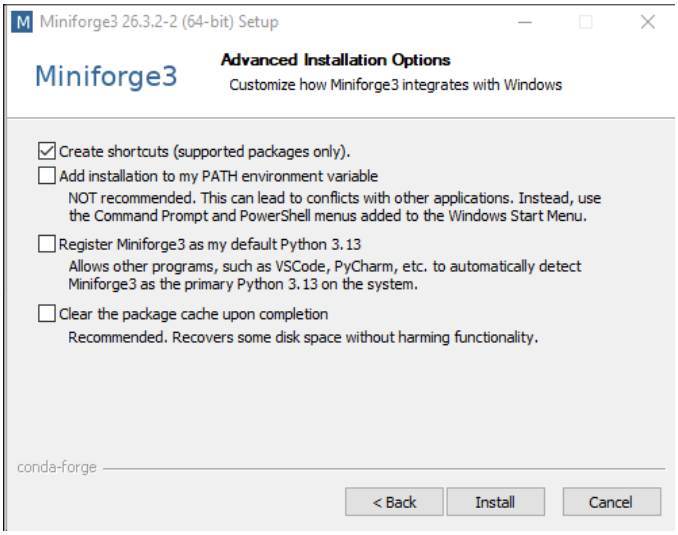
Select "Just Me" and validate:



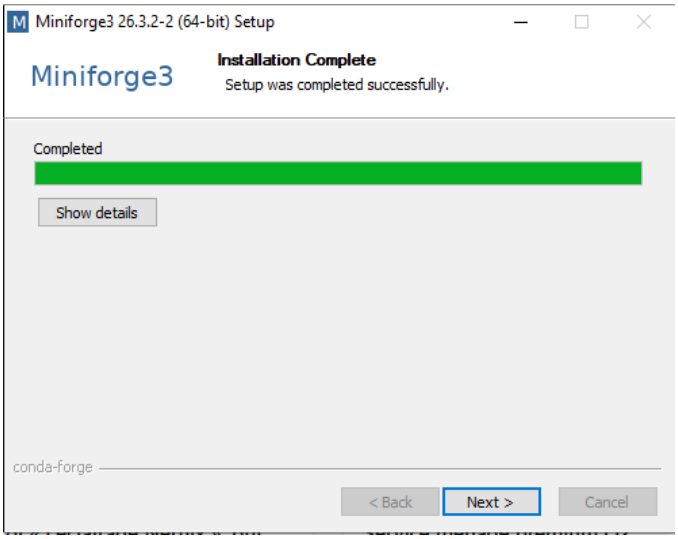
Modify the install path to `c:\gdal\miniforge3` and validate:



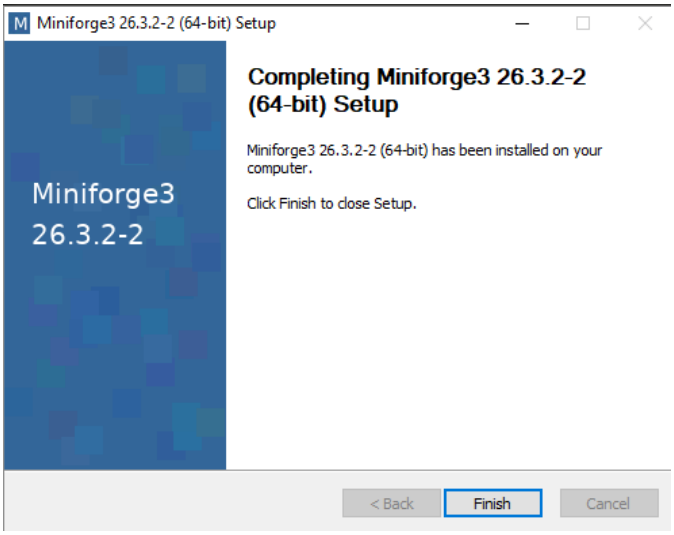
Only select "Create shortcuts" and click Install:



Once installation has completed, click Next:



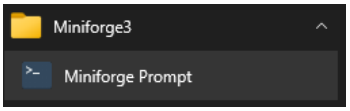
And finally click Finish:



2.3.2 GDAL installation in a dedicated conda environment

First, let's start a Conda enabled command line.

From the Start Menu, select "Anaconda Prompt"



which will open a `cmd` console with Conda executables available in the PATH.

Then we will create a Conda "environment" for the purpose of this workshop, and will call it "gdal", and install it into `c:\gdal\condaenv\gdal`. A Conda environment is a kind of workspace where you can install a set of packages that will not interfere with the ones of other environments. We use the "conda-forge" channel to get up-to-date official releases from the conda community.

```
(base) C:\Users\my_user_name>conda create -y --prefix c:/gdal/condaenv/gdal -c conda-
↪forge
```

Output:

```
Retrieving notices: done
Channels:
 - conda-forge
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: c:\gdal\condaenv\gdal


Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate c:\gdal\condaenv\gdal
```

(continues on next page)

(continued from previous page)

```
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

As suggested, you need to activate the newly created environment with:

```
(base) C:\Users\my_user_name>conda activate c:\gdal\condaenv\gdal
```

When an environment is activated, new lines in the shell are prefixed with "(name_of_environment)".

Now, we can finally install GDAL ! We ask to install both the library and command line utilities with the `libgdal` meta package (that installs `libgdal-core` and most optional driver plugins such as `libgdal-jp2openjpeg`, etc), and the `gdal` package with the Python bindings and scripts.

```
(c:\gdal\condaenv\gdal) C:\Users\my_user_name>conda install -y -c conda-forge gdal␣
↪libgdal
```

```
Channels:
 - conda-forge
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: c:\gdal\condaenv\gdal

  added / updated specs:
    - gdal
    - libgdal

The following packages will be downloaded:

    package                    |            build
    [ ... snip ... ]
    ------------------------------------------------------------
                                           Total:       180.9 MB

The following NEW packages will be INSTALLED:

    [ ... snip ... ]

[ ... displaying packages in download ... ]

Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Now let's check we got the GDAL we expected:

```
(c:\gdal\condaenv\gdal) C:\Users\my_user_name>gdal --version
```

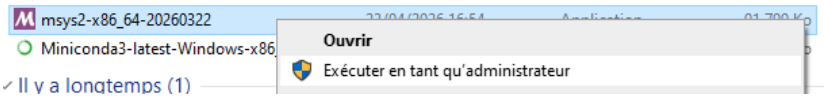
```
GDAL 3.13.1 "Iowa City", released 2026/06/01
```

2.3.3 Install a Bash shell

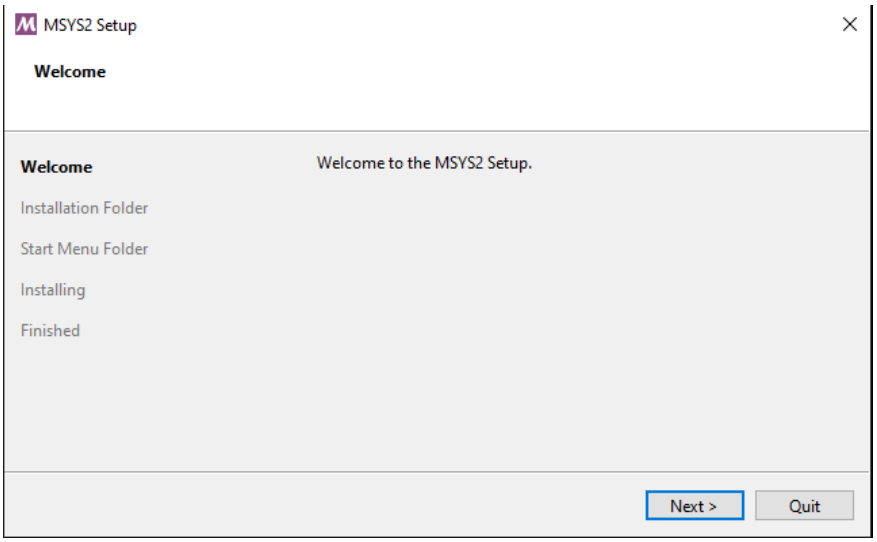
The GDAL CLI comes with a powerful auto-completion mechanism, but this requires it to be used from a Bash-compatible shell. In this paragraph, we will proceed to installing such shell.

Download the MSYS2 installer at https://github.com/msys2/msys2-installer/releases/download/2026-03-22/msys2-x86_64-20260322.exe

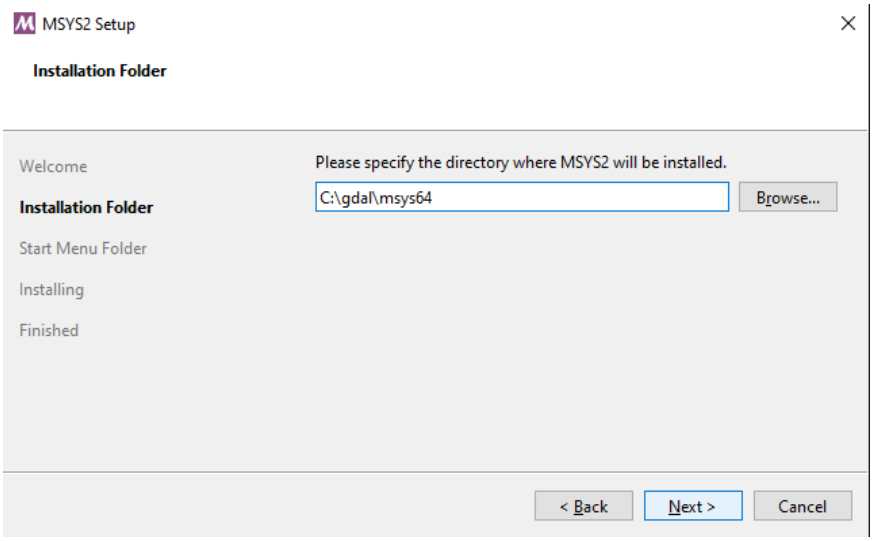
And execute it as an administrator, typically by right-clicking on the executable file and select "Run as administrator". Running as administrator is to make sure that you have permissions to install into `c:\gdal` so that later instructions can be applied without modification.



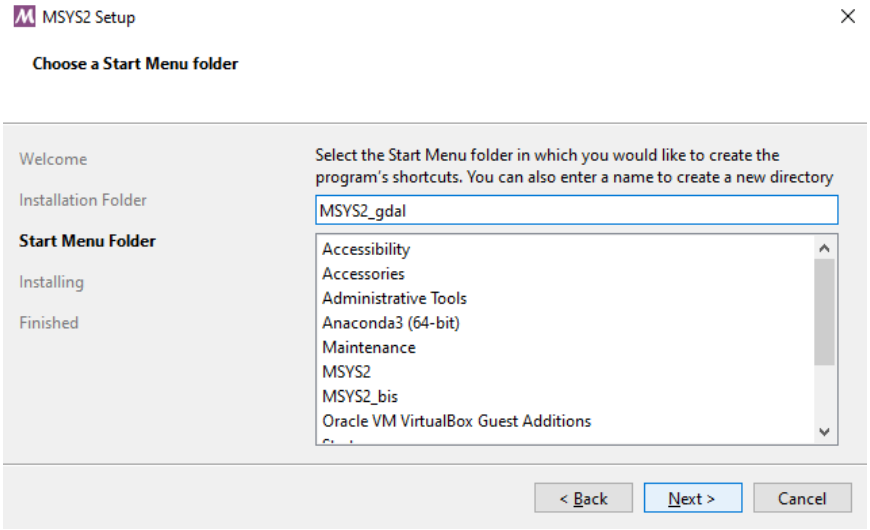
Click on Next:



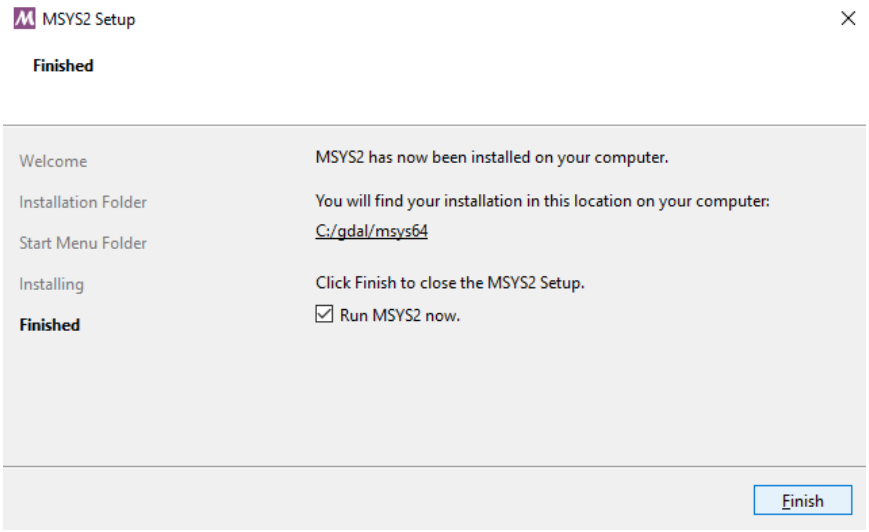
Specify `c:\gdal\msys64` as the installation folder and click on Next:



Specify "msys2_gdal" as the Start Menu folder and click on Next:



Click on Finish:



2.3.4 Create a launcher script

Download the script at https://raw.githubusercontent.com/rouault/gdal_cli_workshop/refs/heads/master/gdal.bat and save it as `c:\gdal\gdal.bat`. This script will launch a Bash shell with all the necessary environment to run GDAL, including the command line completion.

Note

The launcher script will only work if you have installed Miniforge3 and the Conda environment in the default locations as described in the previous sections. If you have installed them in different locations, you will need to edit the script to adjust the paths.

Launch `c:\gdal\gdal.bat` from the Explorer or a shortcut you may have created.

Type (<TAB> means to press the TAB key):

```
gdal --<TAB><TAB>
```

And you should see the following options to be proposed:

```
--config      --drivers     --help        --json-usage  --version
```

2.4 Getting datasets used in the workshop

Download https://github.com/rouault/gdal_cli_workshop_data/archive/refs/heads/master.zip (more than 1 GB) and unzip its content in a directory of your choice.

For example, on Linux/MacOSX:

```
$ mkdir $HOME/gdal_cli_workshop_data
$ cd $HOME/gdal_cli_workshop_data
$ curl -O https://github.com/rouault/gdal_cli_workshop_data/archive/refs/heads/master.zip
$ unzip master.zip
```

On Windows, download and unzip the file. You will need to set the working directory to the unzipped folder to run the exercises. For example if you unzipped the file in `C:\gdal\gdal_cli_workshop_data-master`, then in the

command line started by the launcher script, set the active directory to that location with:

```
(gdal) /c/gdal$ cd gdal_cli_workshop_data-master
```

Note

To meet GitHub file size constraints (max 100 MB / file), a few files have been removed from original Sentinel 2 datasets and file S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_T34TDR_A has been reprocessed to a reduced precision using `gdal raster convert` with `--of JP2OpenJPEG --creation-option QUALITY=45 --co BLOCKXSIZE=1024 --co BLOCKYSIZE=1024` settings to fit under 100 MB.

2.5 Installing `jq` utility (JSON processing)

In a Conda enabled shell,

```
conda install -y -c conda-forge jq
```

2.6 Strongly recommended: installing QGIS for visualisation

Warning

QGIS Conda builds for MacOS-X are unfortunately not currently available through Conda. Go to https://www.qgis.org/download/ for QGIS installers for MacOS-X.

For Linux and Windows, from a Conda enabled shell:

```
conda install -y -c conda-forge qgis
```

(1.3 GB extra download size, and presumably ~2 GB of extra package size once uncompressed)

2.7 Text editor

Have a simple text editor of your choice available to be able to prepare and copy and paste command lines.

2.8 Using GDAL development builds (for advanced / risk-tolerant users)

Users that want to test the latest state of the main GDAL development branch (called "master") can use the `gdal-master` Conda channel to get such builds, as explained in https://gdal.org/en/stable/download.html#gdal-master-conda-builds

CHAPTER

THREE

GETTING INFORMATION FROM RASTER DATASETS

From now, we assume you have a GDAL-enabled shell, and the current working directory is the one where you have downloaded the sample datasets, as indicated in the *Pre-requisites: GDAL installation and workshop sample dataset download* section.

3.1 Utilities demonstrated

- gdal dataset identify
- gdal raster info

3.2 Scanning a folder for GDAL datasets

First, make sure you are in the directory containing the workshop datasets (the exact location depends on your setup):

```
$ cd gdal_cli_workshop_data-master
# Windows
$ cd /c/gdal/gdal_cli_workshop_data-master
```

Now let's use gdal dataset identify in recursive mode in the current directory:

```
$ gdal dataset identify -r .
```

Output:

```
./S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/MTD_MSIL2A.xml:␣
↪SENTINEL2
./S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/INSPIRE.xml: GML
./S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_T34TDR_
↪A047681_20260423T094113/QI_DATA/MSK_DETFOO_B12.jp2: JP2OpenJPEG
./S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_T34TDR_
↪A047681_20260423T094113/QI_DATA/MSK_DETFOO_B05.jp2: JP2OpenJPEG
./S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_T34TDR_
↪A047681_20260423T094113/QI_DATA/MSK_QUALIT_B8A.jp2: JP2OpenJPEG
./S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_T34TDR_
↪A047681_20260423T094113/QI_DATA/MSK_QUALIT_B05.jp2: JP2OpenJPEG
./S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_T34TDR_
↪A047681_20260423T094113/QI_DATA/MSK_DETFOO_B07.jp2: JP2OpenJPEG
./S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_T34TDR_
↪A047681_20260423T094113/QI_DATA/MSK_QUALIT_B08.jp2: JP2OpenJPEG
./S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_T34TDR_
```

(continues on next page)

(continued from previous page)

```
↪A047681_20260423T094113/QI_DATA/MSK_DETFOO_B02.jp2: JP2OpenJPEG
./S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_T34TDR_
↪A047681_20260423T094113/QI_DATA/MSK_DETFOO_B09.jp2: JP2OpenJPEG
[ ... snip ... ]
./S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/HTML/banner_2.png:␣
↪PNG
./S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/HTML/star_bg.jpg:␣
↪JPEG
./S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/HTML/banner_1.png:␣
↪PNG
./S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/HTML/banner_3.png:␣
↪PNG
./S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/manifest.safe: GML
[ ... snip ... ]
```

We see 3 main Sentinel 2 products, the JPEG-2000 files with the data, and a bunch of auxiliary files, most of them being "noise".

3.3 Raster dataset with subdatasets

Let's get information on one of the Sentinel 2 datasets

```
$ gdal raster info S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/MTD_
↪MSIL2A.xml
```

Output:

Driver short name and long name. It is generally a good idea to consult the driver documentation page to learn about the peculiarities of each format. Here we are using Sentinel-2.

```
Driver: SENTINEL2/Sentinel 2
```

Metadata that applies at the dataset level

```
Metadata:
  AOT_QUANTIFICATION_VALUE=1000.0
  AOT_QUANTIFICATION_VALUE_UNIT=none
  AOT_RETRIEVAL_ACCURACY=0.0
  AOT_RETRIEVAL_METHOD=SEN2COR_DDV
  BOA_QUANTIFICATION_VALUE=10000
  BOA_QUANTIFICATION_VALUE_UNIT=none
  CAST_SHADOW_PERCENTAGE=0.020978
  CLOUDY_PIXEL_OVER_LAND_PERCENTAGE=13.652931
  CLOUD_COVERAGE_ASSESSMENT=13.622445
  CLOUD_SHADOW_PERCENTAGE=4.0E-6
  DATATAKE_1_DATATAKE_SENSING_START=2026-04-23T09:40:29.024Z
  DATATAKE_1_DATATAKE_TYPE=INS-NOBS
  DATATAKE_1_ID=GS2B_20260423T094029_047681_N05.12
  DATATAKE_1_SENSING_ORBIT_DIRECTION=DESCENDING
  DATATAKE_1_SENSING_ORBIT_NUMBER=36
  DATATAKE_1_SPACECRAFT_NAME=Sentinel-2B
  DEGRADED_ANC_DATA_PERCENTAGE=0.0
```

(continues on next page)

(continued from previous page)

```
  DEGRADED_MSI_DATA_PERCENTAGE=0
  FOOTPRINT=POLYGON((20.815513868354657 45.0636441557092, 20.738388385926616 45.
↪0825860720989, 20.7383279976147 45.08243722604747, 20.73829168579047 45.08244616120928,
↪ 20.737841954045535 45.081338607396354, 20.723616416649268 45.08479331137292, 20.
↪723749295718594 45.08512101907749, 20.420208979129363 45.15414241241193, 20.
↪420325912036265 45.15443445656253, 20.4201440484806 45.154474815657785, 20.
↪42019397077344 45.15459941811133, 20.41993843797668 45.15465628545975, 20.
↪420062047711312 45.154964985110986, 20.141998695588608 45.217031549439376, 20.
↪141951168063247 45.21691119363821, 20.140959170987895 45.21713316759646, 20.
↪14095913434592 45.217133074779504, 20.140956302668084 45.21713370924307, 20.
↪14068749523566 45.21645339695486, 19.828665356777663 45.28183133788763, 19.
↪82877859125412 45.282121770659984, 19.82873392014264 45.282130804492965, 19.
↪828808772594694 45.28232269102338, 19.828661421501813 45.28235252832233, 19.
↪828817354085736 45.282752449475936, 19.723826624134375 45.30404156688893, 19.
↪70695108472353 46.0462596089832, 21.126167237224053 46.05350473573164, 21.
↪124259208669503 45.193517377663945, 21.08029382801478 45.086817339340286, 21.
↪071227534895563 45.06492790336747, 20.815513868354657 45.0636441557092))
  FORMAT_CORRECTNESS=PASSED
  GENERAL_QUALITY=PASSED
  GENERATION_TIME=2026-04-23T11:57:14.000000Z
  GEOMETRIC_QUALITY=PASSED
  GRANULE_MEAN_AOT=0.06598
  GRANULE_MEAN_WV=0.75046
  HIGH_PROBA_CLOUDS_PERCENTAGE=0.001836
  L2A_QUALITY=PASSED
  MEDIUM_PROBA_CLOUDS_PERCENTAGE=0.009661
  NODATA_PIXEL_PERCENTAGE=9.990737
  NOT_VEGETATED_PERCENTAGE=28.988278
  OZONE_SOURCE=AUX_ECMWFT
  OZONE_VALUE=416.167104
  PREVIEW_GEO_INFO=Not applicable
  PREVIEW_IMAGE_URL=Not applicable
  PROCESSING_BASELINE=05.12
  PROCESSING_LEVEL=Level-2A
  PRODUCT_DOI=https://doi.org/10.5270/S2_-znk9xsj
  PRODUCT_START_TIME=2026-04-23T09:40:29.024Z
  PRODUCT_STOP_TIME=2026-04-23T09:40:29.024Z
  PRODUCT_TYPE=S2MSI2A
  PRODUCT_URI=S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE
  RADIATIVE_TRANSFER_ACCURACY=0.0
  RADIOMETRIC_QUALITY=PASSED
  REFERENCE_BAND=B4
  REFLECTANCE_CONVERSION_U=0.991700831171221
  SATURATED_DEFECTIVE_PIXEL_PERCENTAGE=0.0
  SENSOR_QUALITY=PASSED
  SNOW_ICE_PERCENTAGE=0.0
  SPECIAL_VALUE_NODATA=0
  SPECIAL_VALUE_SATURATED=65535
  THIN_CIRRUS_PERCENTAGE=13.610949
  UNCLASSIFIED_PERCENTAGE=0.243549
  VEGETATION_PERCENTAGE=56.460464
  WATER_PERCENTAGE=0.664281
```

(continues on next page)

(continued from previous page)

```
  WATER_VAPOUR_RETRIEVAL_ACCURACY=0.0
  WVP_QUANTIFICATION_VALUE=1000.0
  WVP_QUANTIFICATION_VALUE_UNIT=cm
```

And "subdatasets"

```
Subdatasets:
  SUBDATASET_1_NAME=SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_
↪20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634
  SUBDATASET_1_DESC=Bands B2, B3, B4, B8, AOT, WVP with 10m resolution, UTM 34N
  SUBDATASET_2_NAME=SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_
↪20260423T115714.SAFE/MTD_MSIL2A.xml:20m:EPSG_32634
  SUBDATASET_2_DESC=Bands B5, B6, B7, B8A, B11, B12, AOT, CLD, SCL, SNW, WVP with 20m␣
↪resolution, UTM 34N
  SUBDATASET_3_NAME=SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_
↪20260423T115714.SAFE/MTD_MSIL2A.xml:60m:EPSG_32634
  SUBDATASET_3_DESC=Bands B1, B9, AOT, CLD, SCL, SNW, WVP with 60m resolution, UTM 34N
  SUBDATASET_4_NAME=SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_
↪20260423T115714.SAFE/MTD_MSIL2A.xml:TCI:EPSG_32634
  SUBDATASET_4_DESC=True color image, UTM 34N
```

Subdatasets can be thought as sub-products or raster layers of a container file. The value of a `SUBDATASET_xxx_NAME` key can be used as valid GDAL dataset name.

3.4 Text information on a raster dataset

So for example let's open the 10m resolution bands with:

```
$ gdal raster info SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_
↪20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634
```

Output:

File(s) composing the dataset:

```
Files: S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/MTD_MSIL2A.xml
       S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_
↪T34TDR_A047681_20260423T094113/MTD_TL.xml
       S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_
↪T34TDR_A047681_20260423T094113/IMG_DATA/R10m/T34TDR_20260423T094029_B04_10m.jp2
       S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_
↪T34TDR_A047681_20260423T094113/IMG_DATA/R10m/T34TDR_20260423T094029_B03_10m.jp2
       S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_
↪T34TDR_A047681_20260423T094113/IMG_DATA/R10m/T34TDR_20260423T094029_B02_10m.jp2
       S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_
↪T34TDR_A047681_20260423T094113/IMG_DATA/R10m/T34TDR_20260423T094029_AOT_10m.jp2
       S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_
↪T34TDR_A047681_20260423T094113/IMG_DATA/R10m/T34TDR_20260423T094029_WVP_10m.jp2
```

Dimensions in pixel: first value is the width (number of columns), second value is the height (number of rows):

```
Size is 10980, 10980
```

Information about the Coordinate Reference System (CRS), also often informally called "projection":

```
Coordinate Reference System:
  - name: WGS 84 / UTM zone 34N
  - ID: EPSG:32634
  - type: Projected
  - projection type: UTM zone 34N, Transverse Mercator
  - units: metre
```

Information how the axis of the dataset maps to the axis of the coordinate reference system. Here the first axis of the data, which is always the X/width dimension for GDAL rasters maps to the first axis of the CRS which is Eastings, and the second axis of the raster, which is always the Y/height dimension maps to the second axis of the CRS which is Northings.

```
Data axis to CRS axis mapping: 1,2
```

Coordinates of the top-left corner of the raster, X value first, Y value second:

```
Origin = (399960.000000000000000,5100000.000000000000000)
```

Size of a pixel in CRS units (here meters), with X value first, Y value second:

```
Pixel Size = (10.000000000000000,-10.000000000000000)
```

Image Structure Metadata is format dependent and gives information on the compression method, internal organization with the `INTERLEAVING` keyword (pixel versus band interleaving. cf https://gdal.org/en/stable/user/raster_data_model.html#multiband-pixel-organization-interleave-metadata-item), etc.

```
Image Structure Metadata:
  COMPRESSION=JPEG2000
```

Coordinates of the corner and center. First tuple of values is expressed in CRS, so here as eastings, northings as this is a projected CRS. Second tuple is their corresponding value in the geographic CRS underlying the projected CRS, here WGS 84 as longitude, latitude in degree, minute and decimal seconds.

```
Corner Coordinates:
Upper Left  (  399960.000, 5100000.000) ( 19d42'25.02"E, 46d 2'46.53"N)
Lower Left  (  399960.000, 4990200.000) ( 19d43'45.90"E, 45d 3'29.49"N)
Upper Right (  509760.000, 5100000.000) ( 21d 7'34.20"E, 46d 3'12.62"N)
Lower Right (  509760.000, 4990200.000) ( 21d 7'26.31"E, 45d 3'54.69"N)
Center      (  454860.000, 5045100.000) ( 20d25'17.86"E, 45d33'28.71"N)
```

Followed by band specific information

```
Band 1 Block=128x128 Type=UInt16, ColorInterp=Red
  Description = B4, central wavelength 665 nm
  Overviews: 5490x5490, 2745x2745, 1373x1373
  Metadata:
    BANDNAME=B4
    BANDWIDTH=30
    BANDWIDTH_UNIT=nm
    WAVELENGTH=665
    WAVELENGTH_UNIT=nm
    SOLAR_IRRADIANCE=1512.79
    SOLAR_IRRADIANCE_UNIT=W/m2/um
    BOA_ADD_OFFSET=-1000
```

(continues on next page)

(continued from previous page)

```
  Image Structure Metadata:
    NBITS=15
  Imagery:
    CENTRAL_WAVELENGTH_UM=0.665
    FWHM_UM=0.030

[ ... snip ...]
```

Block corresponds to the smallest unit GDAL can access pixel values. It is typically either a whole line or a set of few lines called "strip", or a rectangular (almost always a square) called a "tile".

The data type `UInt16` is a 16-bit unsigned integer value, so for values between 0 and 65535, here actually restricted to 0-32767 given the `NBITS=15` metadata.

Overviews correspond to image pyramids, i.e. reduced resolution versions of the full resolution raster. They are generally automatically used by GDAL for processing occurring at a reduced resolution.

The Imagery metadata domain contains a few metadata items whose meaning is normalized across drivers, whereas the main (default) domain is driver specific and may contain anything.

`CENTRAL_WAVELENGTH_UM` corresponds to the Central Wavelength in micrometers and `FWHM_UM` to the Full-width half-maximum (FWHM) value in micrometers.

There are various options that can be used to customize the default output of `gdal raster info`. You can get them by asking for auto-completion suggestions by adding dash dash and pressing the TAB key twice.

```
$ gdal raster info SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_
↪20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634 --<TAB><TAB>
```

Output:

```
--approx-stats     --hist             --list-mdd         --no-ct            --no-mask  ␣
↪      --open-option      --subdataset
--checksum         --input            --metadata-domain  --no-fl            --no-md    ␣
↪      --output-format
--crs-format       --input-format     --min-max          --no-gcp           --no-nodata ␣
↪      --stats
```

For example statistics:

```
$ gdal raster info SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_
↪20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634 --stats
```

Answer "y" and validate.

Output:

```
Band 1 Block=128x128 Type=UInt16, ColorInterp=Red
  Description = B4, central wavelength 665 nm
  Min=0.000 Max=17143.000
  Minimum=0.000, Maximum=17143.000, Mean=1439.120, StdDev=582.568
  [... snip ... ]
  Metadata:
      [... snip ... ]
      STATISTICS_MINIMUM=0
      STATISTICS_MAXIMUM=17143
```

(continues on next page)

(continued from previous page)

```
    STATISTICS_MEAN=1439.1200572908
    STATISTICS_STDDEV=582.56825507515
    STATISTICS_VALID_PERCENT=100
```

Here it would seem that all pixels are valid, but the Sentinel 2 driver does not report a NoData / missing value for the dataset, hence if you open the dataset with `qgis S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/MTD_MSIL2A.xml` you will see that values at 0 actually correspond to a NoData value. Something to keep in mind and take into account in further processing.

3.5 Hidden feature: symbolic links and subdatasets

Warning

Only works as a true symbolic link for Linux and MacOSX, sorry...

But for Windows, you can have almost the same behavior by doing

```
gdal raster convert <input> <shortcut.vrt>
```

As those Sentinel 2 filenames are quite long, you can of course create symbolic links to them with:

```
$ ln -s S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/MTD_MSIL2A.xml␣
↪s2_TDR.xml
```

But even better you can also use them to point to subdatasets, which are not actual files:

```
$ ln -s SENTINEL2_L2A:s2_TDR.xml:10m:EPSG_32634 s2_TDR_10m.xml

(or on Windows)

$ gdal raster convert --of VRT SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_
↪T34TDR_20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634 s2_TDR_10m.xml
```

Check that this works with GDAL

```
$ gdal raster info s2_TDR_10m.xml
```

And that also works with QGIS

```
$ qgis s2_TDR_10m.xml
```

Let's repeat that with other tiles:

```
$ ln -s SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TER_20260423T115714.SAFE/
↪MTD_MSIL2A.xml:10m:EPSG_32634 s2_TER_10m.xml
$ ln -s SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDS_20260423T115714.SAFE/
↪MTD_MSIL2A.xml:10m:EPSG_32634 s2_TDS_10m.xml

(or on Windows)
```

(continues on next page)

(continued from previous page)

```
$ gdal raster convert --of VRT SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_
↪T34TER_20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634 s2_TER_10m.xml
$ gdal raster convert --of VRT SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_
↪T34TDS_20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634 s2_TDS_10m.xml
```

3.6 Short version

Do you find `gdal raster info` too long ?

You can use `gdal info` or even `gdal` !

Beware those short versions only accept the `--format` option, which is common between `gdal raster info` and `gdal vector info`

3.7 Getting information as JSON

Text output is friendly for humans, but no so much for machine processing.

```
$ gdal raster info SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_
↪20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634 --format=json
```


```
{
  "description":"SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_
↪20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634",
  "driverShortName":"SENTINEL2",
  "driverLongName":"Sentinel 2",
  "files":[
    "S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/MTD_MSIL2A.xml",
    "S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_
↪T34TDR_A047681_20260423T094113/MTD_TL.xml",
    "S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_
↪T34TDR_A047681_20260423T094113/IMG_DATA/R10m/T34TDR_20260423T094029_B04_10m.jp2",
    "S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_
↪T34TDR_A047681_20260423T094113/IMG_DATA/R10m/T34TDR_20260423T094029_B03_10m.jp2",
    "S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_
↪T34TDR_A047681_20260423T094113/IMG_DATA/R10m/T34TDR_20260423T094029_B02_10m.jp2",
    "S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_
↪T34TDR_A047681_20260423T094113/IMG_DATA/R10m/T34TDR_20260423T094029_B08_10m.jp2",
    "S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_
↪T34TDR_A047681_20260423T094113/IMG_DATA/R10m/T34TDR_20260423T094029_AOT_10m.jp2",
    "S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/GRANULE/L2A_
↪T34TDR_A047681_20260423T094113/IMG_DATA/R10m/T34TDR_20260423T094029_WVP_10m.jp2"
  ],
  "size":[
    10980,
    10980
  ],
  "coordinateSystem":{
    "wkt":"PROJCRS[\"WGS 84 / UTM zone 34N\",\n    BASEGEOGCRS[\"WGS 84\",\n        
↪DATUM[\"World Geodetic System 1984\",\n            ELLIPSOID[\"WGS 84\",6378137,298.
↪257223563,\n                LENGTHUNIT[\"metre\",1]]],\n        PRIMEM[\"Greenwich\",0,
```


(continues on next page)

(continued from previous page)


```
↪\n            ANGLEUNIT[\"degree\",0.0174532925199433]],\n        ID[\"EPSG\",4326]],\
↪n    CONVERSION[\"UTM zone 34N\",\n        METHOD[\"Transverse Mercator\",\n          ␣
↪ ID[\"EPSG\",9807]],\n        PARAMETER[\"Latitude of natural origin\",0,\n          ␣
↪ANGLEUNIT[\"degree\",0.0174532925199433],\n            ID[\"EPSG\",8801]],\n       ␣
↪PARAMETER[\"Longitude of natural origin\",21,\n            ANGLEUNIT[\"degree\",0.
↪0174532925199433],\n            ID[\"EPSG\",8802]],\n        PARAMETER[\"Scale factor␣
↪at natural origin\",0.9996,\n            SCALEUNIT[\"unity\",1],\n            ID[\
↪"EPSG\",8805]],\n        PARAMETER[\"False easting\",500000,\n            LENGTHUNIT[\
↪"metre\",1],\n            ID[\"EPSG\",8806]],\n        PARAMETER[\"False northing\",0,\
↪n            LENGTHUNIT[\"metre\",1],\n            ID[\"EPSG\",8807]]],\n   ␣
↪CS[Cartesian,2],\n        AXIS[\"easting\",east,\n            ORDER[1],\n          ␣
↪LENGTHUNIT[\"metre\",1]],\n        AXIS[\"northing\",north,\n            ORDER[2],\n  ␣
↪         LENGTHUNIT[\"metre\",1]],\n    ID[\"EPSG\",32634]]",
    "dataAxisToSRSAxisMapping":[
      1,
      2
    ]
  },
  "geoTransform":[
    399960.0,
    10.0,
    0.0,
    5100000.0,
    0.0,
    -10.0
  ],
  "metadata":{
    "":{
      "AOT_QUANTIFICATION_VALUE":"1000.0",
      "AOT_QUANTIFICATION_VALUE_UNIT":"none",
      "AOT_RETRIEVAL_ACCURACY":"0.0",
      "AOT_RETRIEVAL_METHOD":"SEN2COR_DDV",
      "BOA_QUANTIFICATION_VALUE":"10000",
      "BOA_QUANTIFICATION_VALUE_UNIT":"none",
      "CAST_SHADOW_PERCENTAGE":"0.020978",
      "CLOUDY_PIXEL_OVER_LAND_PERCENTAGE":"13.652931",
      "CLOUD_COVERAGE_ASSESSMENT":"13.622445",
      "CLOUD_SHADOW_PERCENTAGE":"4.0E-6",
      "DATATAKE_1_DATATAKE_SENSING_START":"2026-04-23T09:40:29.024Z",
      "DATATAKE_1_DATATAKE_TYPE":"INS-NOBS",
      "DATATAKE_1_ID":"GS2B_20260423T094029_047681_N05.12",
      "DATATAKE_1_SENSING_ORBIT_DIRECTION":"DESCENDING",
      "DATATAKE_1_SENSING_ORBIT_NUMBER":"36",
      "DATATAKE_1_SPACECRAFT_NAME":"Sentinel-2B",
      "DEGRADED_ANC_DATA_PERCENTAGE":"0.0",
      "DEGRADED_MSI_DATA_PERCENTAGE":"0",
      "FORMAT_CORRECTNESS":"PASSED",
      "GENERAL_QUALITY":"PASSED",
      "GENERATION_TIME":"2026-04-23T11:57:14.000000Z",
      "GEOMETRIC_QUALITY":"PASSED",
      "GRANULE_MEAN_AOT":"0.06598",
      "GRANULE_MEAN_WV":"0.75046",
```


(continues on next page)

(continued from previous page)


```
      "HIGH_PROBA_CLOUDS_PERCENTAGE":"0.001836",
      "L2A_QUALITY":"PASSED",
      "MEDIUM_PROBA_CLOUDS_PERCENTAGE":"0.009661",
      "NODATA_PIXEL_PERCENTAGE":"9.990737",
      "NOT_VEGETATED_PERCENTAGE":"28.988278",
      "OZONE_SOURCE":"AUX_ECMWFT",
      "OZONE_VALUE":"416.167104",
      "PREVIEW_GEO_INFO":"Not applicable",
      "PREVIEW_IMAGE_URL":"Not applicable",
      "PROCESSING_BASELINE":"05.12",
      "PROCESSING_LEVEL":"Level-2A",
      "PRODUCT_DOI":"https://doi.org/10.5270/S2_-znk9xsj",
      "PRODUCT_START_TIME":"2026-04-23T09:40:29.024Z",
      "PRODUCT_STOP_TIME":"2026-04-23T09:40:29.024Z",
      "PRODUCT_TYPE":"S2MSI2A",
      "PRODUCT_URI":"S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE",
      "RADIATIVE_TRANSFER_ACCURACY":"0.0",
      "RADIOMETRIC_QUALITY":"PASSED",
      "REFERENCE_BAND":"B4",
      "REFLECTANCE_CONVERSION_U":"0.991700831171221",
      "SATURATED_DEFECTIVE_PIXEL_PERCENTAGE":"0.0",
      "SENSOR_QUALITY":"PASSED",
      "SNOW_ICE_PERCENTAGE":"0.0",
      "SPECIAL_VALUE_NODATA":"0",
      "SPECIAL_VALUE_SATURATED":"65535",
      "THIN_CIRRUS_PERCENTAGE":"13.610949",
      "UNCLASSIFIED_PERCENTAGE":"0.243549",
      "VEGETATION_PERCENTAGE":"56.460464",
      "WATER_PERCENTAGE":"0.664281",
      "WATER_VAPOUR_RETRIEVAL_ACCURACY":"0.0",
      "WVP_QUANTIFICATION_VALUE":"1000.0",
      "WVP_QUANTIFICATION_VALUE_UNIT":"cm"
    },
    "IMAGE_STRUCTURE":{
      "COMPRESSION":"JPEG2000"
    }
  },
  "cornerCoordinates":{
    "upperLeft":[
      399960.0,
      5100000.0
    ],
    "lowerLeft":[
      399960.0,
      4990200.0
    ],
    "lowerRight":[
      509760.0,
      4990200.0
    ],
    "upperRight":[
      509760.0,
```


(continues on next page)

(continued from previous page)


```
      5100000.0
    ],
    "center":[
      454860.0,
      5045100.0
    ]
  },
  "wgs84Extent":{
    "type":"Polygon",
    "coordinates":[
      [
        [
          19.7069511,
          46.0462596
        ],
        [
          19.7294164,
          45.0581917
        ],
        [
          21.1239745,
          45.0651927
        ],
        [
          21.1261672,
          46.0535047
        ],
        [
          19.7069511,
          46.0462596
        ]
      ]
    ]
  },
  "bands":[
    {
      "band":1,
      "block":[
        128,
        128
      ],
      "type":"UInt16",
      "colorInterpretation":"Red",
      "description":"B4, central wavelength 665 nm",
      "min":0.0,
      "max":17143.0,
      "overviews":[
        {
          "size":[
            5490,
            5490
          ]
```


(continues on next page)

(continued from previous page)


```
        },
        {
          "size":[
            2745,
            2745
          ]
        },
        {
          "size":[
            1373,
            1373
          ]
        }
      ],
      "metadata":{
        "":{
          "BANDNAME":"B4",
          "BANDWIDTH":"30",
          "BANDWIDTH_UNIT":"nm",
          "WAVELENGTH":"665",
          "WAVELENGTH_UNIT":"nm",
          "SOLAR_IRRADIANCE":"1512.79",
          "SOLAR_IRRADIANCE_UNIT":"W/m2/um",
          "BOA_ADD_OFFSET":"-1000"
        },
        "IMAGE_STRUCTURE":{
          "NBITS":"15"
        },
        "IMAGERY":{
          "CENTRAL_WAVELENGTH_UM":"0.665",
          "FWHM_UM":"0.030"
        }
      }
    }
  ],
  "stac":{
    "proj:shape":[
      10980,
      10980
    ],
    "proj:wkt2":"PROJCRS[\"WGS 84 / UTM zone 34N\",\n    BASEGEOGCRS[\"WGS 84\",\n        DATUM[\"World Geodetic System 1984\",\n            ELLIPSOID[\"WGS 84\",6378137,298.257223563,\n                LENGTHUNIT[\"metre\",1]]],\n        PRIMEM[\"Greenwich\",0,\n            ANGLEUNIT[\"degree\",0.0174532925199433]],\n        ID[\"EPSG\",4326]],\n    CONVERSION[\"UTM zone 34N\",\n        METHOD[\"Transverse Mercator\",\n            ID[\"EPSG\",9807]],\n        PARAMETER[\"Latitude of natural origin\",0,\n            ANGLEUNIT[\"degree\",0.0174532925199433],\n            ID[\"EPSG\",8801]],\n        PARAMETER[\"Longitude of natural origin\",21,\n            ANGLEUNIT[\"degree\",0.0174532925199433],\n            ID[\"EPSG\",8802]],\n        PARAMETER[\"Scale factor at natural origin\",0.9996,\n            SCALEUNIT[\"unity\",1],\n            ID[\"EPSG\",8805]],\n        PARAMETER[\"False easting\",500000,\n            LENGTHUNIT[\"metre\",1],\n            ID[\"EPSG\",8806]],\n        PARAMETER[\"False northing\",0,\
```


(continues on next page)

(continued from previous page)


```
↪n            LENGTHUNIT[\"metre\",1],\n            ID[\"EPSG\",8807]]],\n
↪CS[Cartesian,2],\n        AXIS[\"easting\",east,\n            ORDER[1],\n
↪LENGTHUNIT[\"metre\",1]],\n        AXIS[\"northing\",north,\n            ORDER[2],\n
↪          LENGTHUNIT[\"metre\",1]],\n    ID[\"EPSG\",32634]]",
    "proj:epsg":32634,
    "proj:projjson":{
      "$schema":"https://proj.org/schemas/v0.7/projjson.schema.json",
      "type":"ProjectedCRS",
      "name":"WGS 84 / UTM zone 34N",
      "base_crs":{
        "type":"GeographicCRS",
        "name":"WGS 84",
        "datum":{
          "type":"GeodeticReferenceFrame",
          "name":"World Geodetic System 1984",
          "ellipsoid":{
            "name":"WGS 84",
            "semi_major_axis":6378137,
            "inverse_flattening":298.257223563
          }
        },
        "coordinate_system":{
          "subtype":"ellipsoidal",
          "axis":[
            {
              "name":"Geodetic latitude",
              "abbreviation":"Lat",
              "direction":"north",
              "unit":"degree"
            },
            {
              "name":"Geodetic longitude",
              "abbreviation":"Lon",
              "direction":"east",
              "unit":"degree"
            }
          ]
        },
        "id":{
          "authority":"EPSG",
          "code":4326
        }
      },
      "conversion":{
        "name":"UTM zone 34N",
        "method":{
          "name":"Transverse Mercator",
          "id":{
            "authority":"EPSG",
            "code":9807
          }
        },
```


(continues on next page)

(continued from previous page)


```
        "parameters":[
          {
            "name":"Latitude of natural origin",
            "value":0,
            "unit":"degree",
            "id":{
              "authority":"EPSG",
              "code":8801
            }
          },
          {
            "name":"Longitude of natural origin",
            "value":21,
            "unit":"degree",
            "id":{
              "authority":"EPSG",
              "code":8802
            }
          },
          {
            "name":"Scale factor at natural origin",
            "value":0.9996,
            "unit":"unity",
            "id":{
              "authority":"EPSG",
              "code":8805
            }
          },
          {
            "name":"False easting",
            "value":500000,
            "unit":"metre",
            "id":{
              "authority":"EPSG",
              "code":8806
            }
          },
          {
            "name":"False northing",
            "value":0,
            "unit":"metre",
            "id":{
              "authority":"EPSG",
              "code":8807
            }
          }
        ]
      },
      "coordinate_system":{
        "subtype":"Cartesian",
        "axis":[
          {
```


(continues on next page)

(continued from previous page)


```
            "name":"Easting",
            "abbreviation":"",
            "direction":"east",
            "unit":"metre"
          },
          {
            "name":"Northing",
            "abbreviation":"",
            "direction":"north",
            "unit":"metre"
          }
        ]
      },
      "id":{
        "authority":"EPSG",
        "code":32634
      }
    },
    "proj:transform":[
      10.0,
      0.0,
      399960.0,
      0.0,
      -10.0,
      5100000.0
    ],
    "raster:bands":[
      {
        "data_type":"uint16"
      },
      {
        "data_type":"uint16"
      },
      {
        "data_type":"uint16"
      },
      {
        "data_type":"uint16"
      },
      {
        "data_type":"uint16"
      },
      {
        "data_type":"uint16"
      }
    ],
    "eo:bands":[
      {
        "name":"b1",
        "description":"B4, central wavelength 665 nm",
        "common_name":"red"
      },
```


(continues on next page)

(continued from previous page)


```
      {
        "name":"b2",
        "description":"B3, central wavelength 560 nm",
        "common_name":"green"
      },
      {
        "name":"b3",
        "description":"B2, central wavelength 490 nm",
        "common_name":"blue"
      },
      {
        "name":"b4",
        "description":"B8, central wavelength 842 nm",
        "common_name":"nir"
      },
      {
        "name":"b5",
        "description":"AOT, Aerosol Optical Thickness map (at 550nm)"
      },
      {
        "name":"b6",
        "description":"WVP, Scene-average Water Vapour map"
      }
    ]
  }
}
```


We can for example extract only the colour interpretation of the first band by combining with the very powerful jq JSON command line processing utility

```
$ gdal raster info SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_
↪20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634 --format=json  | jq ".bands[0].
↪colorInterpretation"
```

Note

`jq` uses 0-based array indexing.

Output:

```
"Red"
```

3.8 Exercise

Extract the CRS code using `jq`.

Hint

Look at STAC related metadata in the JSON output.

==> *Exercise solution for raster info*.

3.9 Invoking Algorithms from Python

See How to use gdal CLI algorithms from Python.

Start Python:

```
$ python
```

And run:

```
from osgeo import gdal
help(gdal.alg)
```

```
NAME
    gdal.alg

SUBMODULES
    dataset
    driver
    mdim
    raster
    vector
    vsi
```

```
help(gdal.alg.raster.info)
```

```
Help on function info:

info(
    input_format: Optional[Union[List[str], dict, str]] = None,
    open_option: Optional[Union[List[str], dict, str]] = None,
    input: Optional[Union[List[gdal.Dataset], List[str], List[os.PathLike[str]]]] = None,
    output_format: Optional[str] = None,
    min_max: Optional[bool] = None,
    stats: Optional[bool] = None,
    approx_stats: Optional[bool] = None,
    hist: Optional[bool] = None,
```

(continues on next page)

(continued from previous page)

```
    no_gcp: Optional[bool] = None,
    no_md: Optional[bool] = None,
    no_ct: Optional[bool] = None,
    no_fl: Optional[bool] = None,
    checksum: Optional[bool] = None,
    list_mdd: Optional[bool] = None,
    metadata_domain: Optional[str] = None,
    no_nodata: Optional[bool] = None,
    no_mask: Optional[bool] = None,
    subdataset: Optional[int] = None,
    crs_format: Optional[str] = None,
    progress: Optional[Callable[[float, str, object], bool]] = None,
    **kwargs
)
    Return information on a raster dataset.

    Consult https://gdal.org/programs/gdal_raster_info.html for more details.

    Parameters
    ----------
    input_format: Optional[Union[List[str], dict, str]]=None
        Input formats
    open_option: Optional[Union[List[str], dict, str]]=None
        Open options
    input: Union[List[gdal.Dataset], List[str], List[os.PathLike[str]]]
        Aliases: dataset
        Input raster dataset
    output_format: Optional[str]=None
        Aliases: format
        Output format
    min_max: Optional[bool]=None
        Compute minimum and maximum value
    stats: Optional[bool]=None
        Retrieve or compute statistics, using all pixels
    approx_stats: Optional[bool]=None
        Retrieve or compute statistics, using a subset of pixels
    hist: Optional[bool]=None
        Retrieve or compute histogram
    no_gcp: Optional[bool]=None
        Suppress ground control points list printing
    no_md: Optional[bool]=None
        Suppress metadata printing
    no_ct: Optional[bool]=None
        Suppress color table printing
    no_fl: Optional[bool]=None
        Suppress file list printing
    checksum: Optional[bool]=None
        Compute pixel checksum
    list_mdd: Optional[bool]=None
        Aliases: list_metadata_domains
        List all metadata domains available for the dataset
    metadata_domain: Optional[str]=None
```

(continues on next page)

(continued from previous page)

```
       Report metadata for the specified domain. 'all' can be used to report metadata␣
↪in all domains

  Output parameters
  -----------------
  output_string: str
      Output string, in which the result is placed
```

Let's try it:

```
>>> gdal.alg.raster.info(input='SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_
↪T34TDR_20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634')

<osgeo.gdal.Algorithm; proxy of <Swig Object of type 'GDALAlgorithmHS *' at␣
↪0x7feb83e38cb0> >
```

So that's return an Algorithm object. Let's extract its output:

```
from osgeo import gdal
with gdal.alg.raster.info(input='SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_
↪T34TDR_20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634', min_max=True) as alg:
    print(alg.Output()['bands'][0])
```

After a few seconds:

```
{'band': 1, 'block': [128, 128], 'type': 'UInt16', 'colorInterpretation': 'Red',
↪'description': 'B4, central wavelength 665 nm', 'computedMin': 0.0, 'computedMax':␣
↪17143.0, 'overviews': [{'size': [5490, 5490]}, {'size': [2745, 2745]}, {'size': [1373,␣
↪1373]}], 'metadata': {'': {'BANDNAME': 'B4', 'BANDWIDTH': '30', 'BANDWIDTH_UNIT': 'nm',
↪ 'WAVELENGTH': '665', 'WAVELENGTH_UNIT': 'nm', 'SOLAR_IRRADIANCE': '1512.79', 'SOLAR_
↪IRRADIANCE_UNIT': 'W/m2/um', 'BOA_ADD_OFFSET': '-1000'}, 'IMAGE_STRUCTURE': {'NBITS':
↪'15'}, 'IMAGERY': {'CENTRAL_WAVELENGTH_UM': '0.665', 'FWHM_UM': '0.030'}}}
```

To exit Python and return to the shell, type:

```
exit()
```

3.10 Getting pixel value

Use gdal raster pixel-info

```
gdal raster pixel-info <dataset> <X> <Y>
```

By default, the X and Y positional arguments are in column,row coordinate space (`--position-crs=pixel`). It is also possible to specify them in the CRS of the dataset (`--position-crs=dataset`), or an explicit CRS.

For example, getting pixel values at Timișoara center (45.7558° N, 21.2322° E):

```
$ gdal raster pixel-info \
   SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TER_20260423T115714.SAFE/MTD_
↪MSIL2A.xml:10m:EPSG_32634 \
   21.2322 45.7558 --position-crs EPSG:4326
```

```
{
  "type":"FeatureCollection",
  "crs":{
    "type":"name",
    "properties":{
      "name":"urn:ogc:def:crs:EPSG::32634"
    }
  },
  "features":[
    {
      "type":"Feature",
      "properties":{
        "input_coordinate":[
          21.232199999999999,
          45.755800000000001
        ],
        "column":1807.8714361003731,
        "line":3305.8006131011061,
        "bands":[
          {
            "band_number":1,
            "raw_value":2490,
            "unscaled_value":2490.0,
            "files":[
              "S2B_MSIL2A_20260423T094029_N0512_R036_T34TER_20260423T115714.SAFE\/
↪GRANULE\/L2A_T34TER_A047681_20260423T094113\/IMG_DATA\/R10m\/T34TER_20260423T094029_
↪B04_10m.jp2"
            ]
          },
          "[... snip ...]",
          {
            "band_number":6,
            "raw_value":927,
            "unscaled_value":927.0,
            "files":[
              "S2B_MSIL2A_20260423T094029_N0512_R036_T34TER_20260423T115714.SAFE\/
↪GRANULE\/L2A_T34TER_A047681_20260423T094113\/IMG_DATA\/R10m\/T34TER_20260423T094029_
↪WVP_10m.jp2"
            ]
          }
        ]
      },
      "geometry":{
        "type":"Point",
        "coordinates":[
          518058.71436100372,
          5066941.9938689889
        ]
      }
    }
  ]
}
```

CHAPTER

FOUR

GETTING INFORMATION FROM VECTOR DATASETS

4.1 Utility

gdal vector info

4.2 Getting vector layer names

```
# run from the workshop data directory

$ gdal vector info timisoara.osm.pbf --summary
```

Output:

```
INFO: Open of `timisoara.osm.pbf'
      using driver `OSM' successful.
1: points (Point)
2: lines (Line String)
3: multilinestrings (Multi Line String)
4: multipolygons (Multi Polygon)
5: other_relations (Geometry Collection)
```

4.3 Getting structure of vector layers

```
$ gdal vector info timisoara.osm.pbf
```

Output:

```
INFO: Open of `timisoara.osm.pbf'
      using driver `OSM' successful.

Layer name: points
Geometry: Point
Feature Count: -1
Layer Coordinate Reference System:
  - name: WGS 84
  - ID: EPSG:4326
  - type: Geographic 2D
Data axis to CRS axis mapping: 2,1
osm_id: String (0.0)
```

(continues on next page)

(continued from previous page)

```
name: String (0.0)
barrier: String (0.0)
highway: String (0.0)
ref: String (0.0)
address: String (0.0)
is_in: String (0.0)
place: String (0.0)
man_made: String (0.0)
other_tags: String (0.0)

Layer name: lines
Geometry: Line String
Feature Count: -1
Layer Coordinate Reference System:
  - name: WGS 84
  - ID: EPSG:4326
  - type: Geographic 2D
Data axis to CRS axis mapping: 2,1
osm_id: String (0.0)
name: String (0.0)
highway: String (0.0)
waterway: String (0.0)
aerialway: String (0.0)
barrier: String (0.0)
man_made: String (0.0)
railway: String (0.0)
z_order: Integer (0.0)
other_tags: String (0.0)

Layer name: multilinestrings
Geometry: Multi Line String
Feature Count: -1
Layer Coordinate Reference System:
  - name: WGS 84
  - ID: EPSG:4326
  - type: Geographic 2D
Data axis to CRS axis mapping: 2,1
osm_id: String (0.0)
name: String (0.0)
type: String (0.0)
other_tags: String (0.0)

Layer name: multipolygons
Geometry: Multi Polygon
Feature Count: -1
Layer Coordinate Reference System:
  - name: WGS 84
  - ID: EPSG:4326
  - type: Geographic 2D
Data axis to CRS axis mapping: 2,1
osm_id: String (0.0)
osm_way_id: String (0.0)
```

(continues on next page)

(continued from previous page)

```
name: String (0.0)
type: String (0.0)
aeroway: String (0.0)
amenity: String (0.0)
admin_level: String (0.0)
barrier: String (0.0)
boundary: String (0.0)
building: String (0.0)
craft: String (0.0)
geological: String (0.0)
historic: String (0.0)
land_area: String (0.0)
landuse: String (0.0)
leisure: String (0.0)
man_made: String (0.0)
military: String (0.0)
natural: String (0.0)
office: String (0.0)
place: String (0.0)
shop: String (0.0)
sport: String (0.0)
tourism: String (0.0)
other_tags: String (0.0)

Layer name: other_relations
Geometry: Geometry Collection
Feature Count: -1
Layer Coordinate Reference System:
  - name: WGS 84
  - ID: EPSG:4326
  - type: Geographic 2D
Data axis to CRS axis mapping: 2,1
osm_id: String (0.0)
name: String (0.0)
type: String (0.0)
other_tags: String (0.0)
```

4.4 Listing features

Listing all features from the `points` layer

```
$ gdal vector info timisoara.osm.pbf --input-layer points --features
```

Output:

```
INFO: Open of `timisoara.osm.pbf'
      using driver `OSM' successful.

Layer name: points
Geometry: Point
Feature Count: -1
```

(continues on next page)

(continued from previous page)

```
Layer Coordinate Reference System:
  - name: WGS 84
  - ID: EPSG:4326
  - type: Geographic 2D
Data axis to CRS axis mapping: 2,1
osm_id: String (0.0)
name: String (0.0)
barrier: String (0.0)
highway: String (0.0)
ref: String (0.0)
address: String (0.0)
is_in: String (0.0)
place: String (0.0)
man_made: String (0.0)
other_tags: String (0.0)
OGRFeature(points):25478311
  osm_id (String) = 25478311
  highway (String) = traffic_signals
  POINT (21.2694902 45.7290217)

OGRFeature(points):25478391
  osm_id (String) = 25478391
  name (String) = Elbromplast
  other_tags (String) = "bus"=>"yes","public_transport"=>"stop_position"
  POINT (21.2698833 45.7331868)

[ ... snip ... ]
```

4.5 Applying an attribute filter, and limiting the output

You can apply an attribute filter by specifying a SQL WHERE filtering clause with the `--where` option. By default, this uses the OGR SQL dialect which is a subset of the full SQL-92 standard.

```
$ gdal vector info timisoara.osm.pbf --input-layer points --features --where "other_tags␣
↪LIKE '%restaurant%'" --limit 5
```

Output:

```
INFO: Open of `timisoara.osm.pbf'
      using driver `OSM' successful.

Layer name: points
Geometry: Point
Feature Count: -1
Layer Coordinate Reference System:
  - name: WGS 84
  - ID: EPSG:4326
  - type: Geographic 2D
Data axis to CRS axis mapping: 2,1
osm_id: String (0.0)
name: String (0.0)
```

(continues on next page)

(continued from previous page)

```
barrier: String (0.0)
highway: String (0.0)
ref: String (0.0)
address: String (0.0)
is_in: String (0.0)
place: String (0.0)
man_made: String (0.0)
other_tags: String (0.0)
OGRFeature(points):304555464
  osm_id (String) = 304555464
  name (String) = Stejarul
  other_tags (String) = "amenity"=>"restaurant"
  POINT (21.4248647 45.7521308)

OGRFeature(points):380526548
  osm_id (String) = 380526548
  name (String) = Restaurant Tinecz
  other_tags (String) = "addr:city"=>"Timișoara","addr:housenumber"=>"51","addr:street"=>
↪"Calea Aradului","amenity"=>"restaurant","cuisine"=>"regional","opening_hours"=>"Mo-Su␣
↪11:00-23:00","phone"=>"0736 660 787","website"=>"https://www.restauranttinecz.ro/"
  POINT (21.2221875 45.7710081)

OGRFeature(points):380526549
  osm_id (String) = 380526549
  name (String) = Iris
  other_tags (String) = "amenity"=>"restaurant","cuisine"=>"regional"
  POINT (21.2147065 45.7746345)

OGRFeature(points):477200233
  osm_id (String) = 477200233
  name (String) = Pizzeria Rebecca
  other_tags (String) = "amenity"=>"restaurant"
  POINT (21.2240089 45.7754256)

OGRFeature(points):477234572
  osm_id (String) = 477234572
  name (String) = Pizzeria Amedeea
  other_tags (String) = "amenity"=>"restaurant"
  POINT (21.2310551 45.7757755)
```

4.6 JSON output / open options

A number of GDAL raster and vector drivers have "open options", which control how the driver behaves. They are documented in each driver's documentation page. For OSM, at https://gdal.org/en/stable/drivers/vector/osm.html#open-options

The `other_tags` field is exposed using the PostgreSQL HSTORE key/value data type which predates the introduction of JSON. But we may query it to be exposed as JSON with `--open-option TAGS_FORMAT=JSON`, and also request features to be exposed as GeoJSON with `--format json`.

```
$ gdal vector info timisoara.osm.pbf --input-layer points --features --where "other_tags␣
↪LIKE '%restaurant%'" --limit 1 --format json --open-option TAGS_FORMAT=JSON
```

```
{
  "description":"timisoara.osm.pbf",
  "driverShortName":"OSM",
  "driverLongName":"OpenStreetMap XML and PBF",
  "layers":[
    {
      "name":"points",
      "metadata":{},
      "geometryFields":[
        {
          "name":"",
          "type":"Point",
          "nullable":true,
          "coordinateSystem":{
            "wkt":"GEOGCRS[\"WGS 84\",\n    DATUM[\"World Geodetic System 1984\",\n
↪  ELLIPSOID[\"WGS 84\",6378137,298.257223563,\n            LENGTHUNIT[\"metre\",1]]],\
↪n    PRIMEM[\"Greenwich\",0,\n        ANGLEUNIT[\"degree\",0.0174532925199433]],\n
↪CS[ellipsoidal,2],\n        AXIS[\"geodetic latitude (Lat)\",north,\n
↪ORDER[1],\n            ANGLEUNIT[\"degree\",0.0174532925199433]],\n        AXIS[\
↪"geodetic longitude (Lon)\",east,\n            ORDER[2],\n            ANGLEUNIT[\
↪"degree\",0.0174532925199433]],\n    ID[\"EPSG\",4326]]",
            "projjson":{
              "$schema":"https://proj.org/schemas/v0.7/projjson.schema.json",
              "type":"GeographicCRS",
              "name":"WGS 84",
              "datum":{
                "type":"GeodeticReferenceFrame",
                "name":"World Geodetic System 1984",
                "ellipsoid":{
                  "name":"WGS 84",
                  "semi_major_axis":6378137,
                  "inverse_flattening":298.257223563
                }
              },
              "coordinate_system":{
                "subtype":"ellipsoidal",
                "axis":[
                  {
                    "name":"Geodetic latitude",
                    "abbreviation":"Lat",
                    "direction":"north",
                    "unit":"degree"
                  },
                  {
                    "name":"Geodetic longitude",
                    "abbreviation":"Lon",
                    "direction":"east",
                    "unit":"degree"
                  }
                ]
              },
              "id":{
                "authority":"EPSG",
```


(continues on next page)

(continued from previous page)


```
            "code":4326
          }
        },
        "dataAxisToSRSAxisMapping":[
          2,
          1
        ]
      }
    }
  ],
  "featureCount":-1,
  "fields":[
    {
      "name":"osm_id",
      "type":"String",
      "nullable":true,
      "uniqueConstraint":false
    },
    {
      "name":"name",
      "type":"String",
      "nullable":true,
      "uniqueConstraint":false
    },
    {
      "name":"barrier",
      "type":"String",
      "nullable":true,
      "uniqueConstraint":false
    },
    {
      "name":"highway",
      "type":"String",
      "nullable":true,
      "uniqueConstraint":false
    },
    {
      "name":"ref",
      "type":"String",
      "nullable":true,
      "uniqueConstraint":false
    },
    {
      "name":"address",
      "type":"String",
      "nullable":true,
      "uniqueConstraint":false
    },
    {
      "name":"is_in",
      "type":"String",
      "nullable":true,
```


(continues on next page)

(continued from previous page)


```
          "uniqueConstraint":false
        },
        {
          "name":"place",
          "type":"String",
          "nullable":true,
          "uniqueConstraint":false
        },
        {
          "name":"man_made",
          "type":"String",
          "nullable":true,
          "uniqueConstraint":false
        },
        {
          "name":"other_tags",
          "type":"String",
          "subType":"JSON",
          "nullable":true,
          "uniqueConstraint":false
        }
      ],
      "features":[
        {
          "type":"Feature",
          "properties":{
            "osm_id":"304555464",
            "name":"Stejarul",
            "other_tags":{
              "amenity":"restaurant"
            }
          },
          "fid":304555464,
          "geometry":{
            "type":"Point",
            "coordinates":[
              21.4248647,
              45.7521308
            ]
          }
        }
      ]
    }
  ],
  "metadata":{},
  "domains":{}
}
```

4.7 Exercise

Find the 5 closest bars from Timișoara center (45.7558° N, 21.2322° E)

Hint

- use option `--sql`
- use SQLite dialect (`--dialect SQLite`)
- the name of the geometry column when using a non-RDBMS data set, such as OSM, and with SQLite dialect, is `geometry`
- use Spatialite ST_Point function
- use Spatialite ST_Distance function

==> *Exercise solution for vector info*.

CHAPTER

FIVE

GENERAL PRINCIPLES OF THE NEW GDAL CLI

5.1 Why a new CLI ?

Webinar given on June 3, 2025 about GDAL Command Line Interface Modernization:

- PDF slide deck
- recording of the video.

Reasons:

- **Program naming inconsistencies:**
  - `gdal_translate` vs `gdalwarp`
  - `gdal_merge` vs `ogrmerge`
- **Inconsistent argument naming:**
  - `gdal_translate -projwin ulx uly lrx lry`
  - `gdalwarp -te llx lly urx ury`
- **Dataset order inconsistency:**
  - `gdal_translate in.tif out.tif`
  - `gdalwarp out.tif in.tif`
- Some utilities silently overwriting output file (`gdal_translate`), while others silently update it (`gdalwarp`).
- "Jumbo" programs with tens of arguments like `gdal_translate`, `gdalwarp` or `ogr2ogr`.

5.2 Principles of new CLI

5.2.1 `git` style hierarchical commands

```
gdal
|
+-- raster              For all commands that accept a raster input (some may␣
↪output vector)
    |
    +-- info
    +-- convert
    +-- reproject
    +-- calc
    +-- clip
```

(continues on next page)

(continued from previous page)

```
    +-- contour         (outputs vector)
    +-- overview
    +-- mosaic
    +-- tile
    +-- index
    +-- ...
|
+-- vector              For all commands that accept a vector input (some may␣
↪output raster)
    |
    +-- info
    +-- convert
    +-- reproject
    +-- clip
    +-- rasterize       (outputs raster)
    +-- mosaic
    +-- index
    +-- ...
+-- mdim
    |
    +-- info
    +-- convert
    +-- mosaic
|
+-- pipeline
|
+-- dataset
    |
    +-- check
    +-- copy
    +-- delete
    +-- calc
    +-- identify
    +-- rename
|
+-- driver
    |
    +-- cog
    +-- gpkg
    +-- gti
    +-- openfilegdb
    +-- parquet
    +-- pdf
    +-- rpftoc
|
+-- vsi
    |
    +-- copy
    +-- delete
    +-- list
    +-- move
    +-- sozip
```

(continues on next page)

(continued from previous page)

```
    +-- sync
|
+-- info
|
+-- convert
```

5.2.2 Smaller scope programs

- `gdal_translate` ==>
  - gdal raster convert
  - gdal raster clip
  - gdal raster edit
- `gdalwarp` ==>
  - gdal raster reproject
  - gdal raster update
- `ogr2ogr` ==>
  - gdal vector convert
  - gdal vector clip
  - gdal vector edit
  - gdal vector reproject

5.2.3 Program syntax

- Positional arguments: 90% of positional arguments are for input dataset(s) and output dataset. Always in that order

```
$ gdal raster convert input.png output.tif
```

- Positional arguments can also be specified as named arguments:
  - Long version

    ```
    $ gdal raster convert --input input.png --output output.tif
    ```

  - Short version

    ```
    $ gdal raster convert -i input.png -o output.tif
    ```

- Setting the value of a named argument:
  - space character separator as above:
  - or equal character separator:

    ```
    $ gdal raster convert --input=input.png --output=output.tif
    ```

- No more silent overwriting
  - `--overwrite` will be required and suggested if needed

```
$ gdal raster convert in.png out.tif
$ gdal raster convert in2.png out.tif
ERROR 1: convert: Dataset 'out.tif' already exists. You may specify the -
↪-overwrite/--append option.
```

Warning

Overwrite destroys and recreates the whole dataset.

Some programs with vector output can offer a more limited scope using `--overwrite-layer`.

- `--update` or `--append` may also be available

5.2.4 I need help !!!

```
$ gdal raster convert --help
```

```
Usage: gdal raster convert [OPTIONS] <INPUT> <OUTPUT>

Convert a raster dataset.

Positional arguments:
  -i, --input <INPUT>                                  Input raster datasets [required]␣
↪[not available in pipelines]
  -o, --output <OUTPUT>                                Output raster dataset [required]␣
↪[not available in pipelines]

Common Options:
  -h, --help                                           Display help message and exit
  --json-usage                                         Display usage as JSON document␣
↪and exit
  --config <KEY>=<VALUE>                               Configuration option [may be␣
↪repeated]
  -q, --quiet                                          Quiet mode (no progress bar or␣
↪warning message) [not available in pipelines]

Options:
  -f, --of, --format, --output-format <OUTPUT-FORMAT>  Output format ("GDALG" allowed)␣
↪[not available in pipelines]
  --co, --creation-option <KEY>=<VALUE>                Creation option [may be repeated]␣
↪[not available in pipelines]
  --overwrite                                          Whether overwriting existing␣
↪output dataset is allowed [not available in pipelines]
                                                       Mutually exclusive with --append
  --append                                             Append as a subdataset to␣
↪existing output [not available in pipelines]
                                                       Mutually exclusive with --
↪overwrite

Advanced Options:
```

(continues on next page)

(continued from previous page)

```
  --if, --input-format <INPUT-FORMAT>                    Input formats [may be repeated]␣
↪[not available in pipelines]
  --oo, --open-option <KEY>=<VALUE>                      Open options [may be repeated]␣
↪[not available in pipelines]

For more details, consult https://gdal.org/programs/gdal_raster_convert.html

WARNING: the gdal command is provisionally provided as an alternative interface to GDAL␣
↪and OGR command line utilities.
The project reserves the right to modify, rename, reorganize, and change the behavior of␣
↪the utility
until it is officially frozen in a future feature release of GDAL.
```

5.2.5 Smart auto-completion

Provided you use a Bash-compatible shell, suggestion of program name, options and arguments are available by pressing <TAB> <TAB>

Discover sub-programs

```
$ gdal raster <TAB><TAB>
```

```
as-features      clip             create           index            overview        ␣
↪proximity         roughness          slope            unscale
aspect           color-map        edit             info             pansharpen      ␣
↪reclassify        scale              stack            update
blend            compare          fill-nodata      mosaic           pipeline        ␣
↪reproject         select             tile             viewshed
calc             contour          footprint        neighbors        pixel-info      ␣
↪resize            set-type           tpi              zonal-stats
clean-collar     convert          hillshade        nodata-to-alpha  polygonize       rgb-
↪to-palette    sieve            tri
```

Discover named arguments

```
$ gdal raster convert --<TAB><TAB>
```

```
--append           --creation-option  --input            --input-format     --open-
↪option      --output           --output-format    --overwrite        --quiet
```

Discover values of arguments

Which output formats are available?

```
gdal raster convert --format=<TAB><TAB>
```

```
GTiff            PNG              PCRaster         ISIS3            WMS              PDF␣
↪              LCP              S102             HF2              NGW
COG              JPEG             ILWIS            PDS4             RST             ␣
↪MBTiles           GTX              S104             ZMap             MiraMonRaster
```

(continues on next page)

(continued from previous page)

```
VRT              MEM              SRTMHGT          VICAR            GSAG             ␣
↪CALS              KRO              S111             SIGDEM             ENVI
NITF             GIF              Leveller         ERS              GSBG             ␣
↪WMTS              ROI_PAC            NWT_GRD            JPEGXL             EHdr
HFA              FITS             Terragen         JP2OpenJPEG      GS7BG              MRF␣
↪              RRASTER          PostGISRaster    TileDB           ISCE
AAIGrid            BMP              netCDF           GRIB             KMLSUPEROVERLAY  PNM␣
↪              KEA              SAGA             GPKG             Zarr
DTED               PCIDSK             HDF4Image        RMF              WEBP             BT ␣
↪              BAG              XYZ              OpenFileGDB      GDALG
```

Discover contextual values of arguments

Which creation options are available for COG output?

```
gdal raster convert --format=COG --creation-option <TAB><TAB>
```

```
COMPRESS=              QUALITY=               JXL_EFFORT=            BLOCKSIZE=          ␣
↪    WARP_RESAMPLING=       ZOOM_LEVEL_STRATEGY=   ADD_ALPHA=
OVERVIEW_COMPRESS=     OVERVIEW_QUALITY=      JXL_DISTANCE=          INTERLEAVE=         ␣
↪    OVERVIEWS=             TARGET_SRS=            GEOTIFF_VERSION=
LEVEL=                 MAX_Z_ERROR=           JXL_ALPHA_DISTANCE=    BIGTIFF=            ␣
↪    OVERVIEW_COUNT=        RES=                   SPARSE_OK=
PREDICTOR=             MAX_Z_ERROR_OVERVIEW=  NUM_THREADS=           RESAMPLING=         ␣
↪    TILING_SCHEME=         EXTENT=                STATISTICS=
OVERVIEW_PREDICTOR=    JXL_LOSSLESS=          NBITS=                 OVERVIEW_
↪RESAMPLING=   ZOOM_LEVEL=            ALIGNED_LEVELS=
```

Which compression methods are available for COG output?

```
gdal raster convert --format=COG --creation-option COMPRESS=<TAB><TAB>
```

```
NONE          LZW           JPEG          DEFLATE       LZMA          ZSTD          WEBP␣
↪         LERC          LERC_DEFLATE  LERC_ZSTD     JXL
```

CHAPTER

SIX

FORMAT CONVERSION

6.1 Raster

Let's try converting a Sentinel 2 dataset to Cloud-optimized GeoTIFF (COG) using gdal raster convert

```
$ gdal raster convert S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/
↪MTD_MSIL2A.xml TDR.tif --format COG
```

```
ERROR 6: COG driver does not support 0-band source raster
```

OK, we need to select one of the subdatasets:

```
$ gdal raster convert SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_
↪20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634 TDR.tif --format COG
```

Wait ~ 1 minute.

Let's check:

```
$ gdal info TDR.tif
```

```
[ ... snip ... ]
Image Structure Metadata:
  LAYOUT=COG
  COMPRESSION=LZW
  INTERLEAVE=PIXEL
[ ... snip ... ]
Band 1 Block=512x512 Type=UInt16, ColorInterp=Red
  Description = B4, central wavelength 665 nm
  Overviews: 5490x5490, 2745x2745, 1373x1373
[ ... snip ... ]
```

6.2 Exercise

1. Play with the different compression methods and observe effects on image size.
2. Try creating a 8-bit JPEG-compressed tiled GeoTIFF file with the red, green and blue bands.

> **Hint**
>
> Use gdal raster scale

3. Improve the visual result by reducing the effect of extreme pixel values (outliers) using statistical clamping.

> **Hint**
>
> Use gdal raster select
>
> You can improve the visual result by clamping the input range to 2 standard deviations of the mean.

==> *Exercise solution for raster format conversion*.

6.3 Vector

Convert the `timisoara.osm.pbf` to GeoPackage, with the `other_tags` field as JSON using gdal vector convert

```
$ gdal vector convert timisoara.osm.pbf timisoara.gpkg --open-option TAGS_FORMAT=JSON
```

Let's check:

```
$ gdal info timisoara.gpkg
```

```
INFO: Open of `timisoara.gpkg'
  using driver `GPKG' successful.

Layer name: points
Geometry: Point
Feature Count: 20415
Extent: (20.167133, 45.346126) - (22.818334, 46.318563)
Layer Coordinate Reference System:
  - name: WGS 84
  - ID: EPSG:4326
  - type: Geographic 2D
  - area of use: World, west -180.00, south -90.00, east 180.00, north 90.00
Data axis to CRS axis mapping: 2,1
FID Column = fid
Geometry Column = geom
osm_id: String (0.0)
name: String (0.0)
barrier: String (0.0)
highway: String (0.0)
ref: String (0.0)
address: String (0.0)
is_in: String (0.0)
place: String (0.0)
man_made: String (0.0)
other_tags: String(JSON) (0.0)

Layer name: lines
[ ... snip ... ]

Layer name: multilinestrings
[ ... snip ... ]
```

(continues on next page)

(continued from previous page)

```
Layer name: multipolygons
[ ... snip ... ]

Layer name: other_relations
[ ... snip ... ]
```

6.4 Exercise

1. Convert `timisoara.osm.pbf` to GeoJSON

   Hint

   Use the `--input-layer` argument to generate one output file per input layer.

2. If you open the above `timisoara.gpkg` in QGIS, you can see that the `other_relations` layer cannot be displayed, because it contains geometry collection features that QGIS does not support.

   Try "exploding" those collections in single primitives (points, lines, polygons).

   Hint

   Use gdal vector explode-collections

3. Create a layer for each geometry type in the `other_collections` layer.

   Hint

   Use gdal vector filter and Spatialite ST_GeometryType function

==> *Exercise solution for vector format conversion*.

CHAPTER

SEVEN

SUBSETTING, RESAMPLING, REPROJECTION

7.1 Raster resampling/resizing

Use gdal raster resize

```
# run from the workshop data directory

$ gdal raster resize \
    SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TER_20260423T115714.SAFE/MTD_
↪MSIL2A.xml:10m:EPSG_32634 \
    10m_bands_to_20m.tif \
    --resolution 20,20 \
    --resampling cubic
```

or

```
$ gdal raster resize \
    SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TER_20260423T115714.SAFE/MTD_
↪MSIL2A.xml:10m:EPSG_32634 \
    10m_bands_half_size.tif \
    --size 50%,50% \
    --resampling cubic
```

Let's compare them with gdal raster compare

```
$ gdal raster compare 10m_bands_to_20m.tif 10m_bands_half_size.tif
```

==> no output, meaning they are bit-to-bit identical

7.2 Let's do that in Python

First,open a Python interpreter in the directory containing the workshop datasets:

```
$ python
```

Next, paste the following code snippet into the interpreter to perform the same operation as in the previous command-line example:

```
from osgeo import gdal
filename = "SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TER_20260423T115714.
↪SAFE/MTD_MSIL2A.xml:10m:EPSG_32634"
```

(continues on next page)

(continued from previous page)

```
with gdal.alg.raster.resize(input=filename, output_format="MEM", output="", size=["50%",
↪"50%"]) as alg:
    output_dataset = alg.Output()
    print(output_dataset.ReadAsArray().shape)
```

```
(6, 5490, 5490)
```

Alternatively:

```
from osgeo import gdal
filename = "SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TER_20260423T115714.
↪SAFE/MTD_MSIL2A.xml:10m:EPSG_32634"
alg = gdal.Algorithm("raster", "resize")
alg["input"] = filename
alg["output_format"] = "MEM"
alg["output"] = ""
alg["size"]=["50%","50%"]
alg.Run()
output_dataset = alg.Output()
print(output_dataset.ReadAsArray().shape)
output_dataset.Close()  # needed when the output is a "real" file, to make sure it is␣
↪closed
```

7.3 Clipping

Use gdal raster clip and gdal vector clip

```
$ gdal raster clip \
    SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TER_20260423T115714.SAFE/MTD_
↪MSIL2A.xml:10m:EPSG_32634 \
    clip.tif \
    --bbox 21.06809,45.64922,21.43590,45.86361 \
    --bbox-crs EPSG:4326
```

```
$ gdal vector clip timisoara.gpkg --layer points \
    timisoara_points_clipped.gpkg \
    --like clip.tif
```

7.4 Exercise

Clip `SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TER_20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634` with a circle centered on Timișoara center (45.7558° N, 21.2322° E) with a radius of 1 km.

Hint

1. Create a GeoJSON file with a single point with the center.

```
{"type":"Point", "coordinates":[<X>,<Y>]}
```

2. With `gdal vector info`, note the name of the layer
3. With `gdal vector sql`, create a circle centered around that geometry using options `--sql` and `--dialect Spatialite`, knowing that the geometry column name will be `geometry` and using SQL functions `ST_Transform` to reproject the coordinate to the EPSG code of the raster layer, and `ST_Buffer` to create the circle.

==> *Exercise solution for clipping*.

7.5 Reprojection

Use gdal raster reproject and gdal vector reproject

```
$ gdal raster reproject s2_TDR_10m.xml --output-crs <TAB><TAB>
```

```
EPSG:      ESRI:      IAU_2015:  IGNF:      NKG:       OGC:       PROJ:
```

```
$ gdal raster reproject s2_TDR_10m.xml --output-crs EPSG:<TAB><TAB>
```

```
10659 -- ETRF2000 + EOMA 1980 height                    10596 -- WGS 84 / GLANCE Europe␣
↪                          7902 -- ITRF90 (geographic 3D)
10660 -- HD72 / EOV + EOMA 1980 height                  27704 -- WGS 84 / Equi7 Europe ␣
↪                          7903 -- ITRF91 (geographic 3D)
23700 -- HD72 / EOV                                     4230 -- ED50 (geographic 2D)   ␣
↪                          7904 -- ITRF92 (geographic 3D)
3819 -- HD1909 (geographic 2D)                          3034 -- ETRS89-extended / LCC␣
↪Europe                      7905 -- ITRF93 (geographic 3D)
4237 -- HD72 (geographic 2D)                            3035 -- ETRS89-extended / LAEA␣
↪Europe                     7906 -- ITRF94 (geographic 3D)
4075 -- SREF98 (geographic 2D)                          32234 -- WGS 72 / UTM zone 34N ␣
↪                          7907 -- ITRF96 (geographic 3D)
4074 -- SREF98 (geographic 3D)                          32434 -- WGS 72BE / UTM zone␣
↪34N                         7908 -- ITRF97 (geographic 3D)
8682 -- SRB_ETRS89 / UTM zone 34N                       32634 -- WGS 84 / UTM zone 34N ␣
↪                          8997 -- ITRF2000 (geographic 2D)
8685 -- SRB_ETRS89 (geographic 2D)                      3571 -- WGS 84 / North Pole␣
↪LAEA Bering Sea             7909 -- ITRF2000 (geographic 3D)
8684 -- SRB_ETRS89 (geographic 3D)                      3572 -- WGS 84 / North Pole␣
↪LAEA Alaska                 7910 -- ITRF2005 (geographic 3D)
6316 -- MGI 1901 / Balkans zone 7                       3573 -- WGS 84 / North Pole␣
↪LAEA Canada                 7911 -- ITRF2008 (geographic 3D)
3836 -- Pulkovo 1942(83) / Gauss-Kruger zone 4          3574 -- WGS 84 / North Pole␣
↪LAEA Atlantic               7912 -- ITRF2014 (geographic 3D)
31600 -- Dealul Piscului 1930 / Stereo 33               3575 -- WGS 84 / North Pole␣
↪LAEA Europe                 8857 -- WGS 84 / Equal Earth Greenwich
4316 -- Dealul Piscului 1930 (geographic 2D)            3576 -- WGS 84 / North Pole␣
↪LAEA Russia                 8888 -- WGS 84 (Transit) (geographic 2D)
3844 -- Pulkovo 1942(58) / Stereo70                     10598 -- WGS 84 / GLANCE North␣
↪America                   8988 -- ITRF88 (geographic 2D)
3906 -- MGI 1901 (geographic 2D)                        27705 -- WGS 84 / Equi7 North␣
↪America                   8989 -- ITRF89 (geographic 2D)
```

(continues on next page)

(continued from previous page)

```
4805 -- MGI (Ferro) (geographic 2D)                        32600 -- WGS 84 / UTM grid␣
↪system (northern hemisphere)  8990 -- ITRF90 (geographic 2D)
4178 -- Pulkovo 1942(83) (geographic 2D)                  3408 -- NSIDC EASE-Grid North  ␣
↪                          8991 -- ITRF91 (geographic 2D)
3331 -- Pulkovo 1942(58) / 3-degree Gauss-Kruger zone 7  6931 -- WGS 84 / NSIDC EASE-
↪Grid 2.0 North               8992 -- ITRF92 (geographic 2D)
3334 -- Pulkovo 1942(58) / Gauss-Kruger zone 4           3395 -- WGS 84 / World Mercator␣
↪                          8993 -- ITRF93 (geographic 2D)
4179 -- Pulkovo 1942(58) (geographic 2D)                  3857 -- WGS 84 / Pseudo-
↪Mercator                          8994 -- ITRF94 (geographic 2D)
7409 -- ETRS89 + EVRF2000 height                          3410 -- NSIDC EASE-Grid Global ␣
↪                          8995 -- ITRF96 (geographic 2D)
7423 -- ETRS89 + EVRF2007 height                          6933 -- WGS 84 / NSIDC EASE-
↪Grid 2.0 Global              8996 -- ITRF97 (geographic 2D)
9422 -- ETRS89 + EVRF2019 height                          10178 -- IGS20 (geographic 2D) ␣
↪                          8998 -- ITRF2005 (geographic 2D)
9423 -- ETRS89 + EVRF2019 mean-tide height                10177 -- IGS20 (geographic 3D) ␣
↪                          8999 -- ITRF2008 (geographic 2D)
25834 -- ETRS89 / UTM zone 34N                            10345 -- Hughes 1980␣
↪(geographic 2D)                      9000 -- ITRF2014 (geographic 2D)
3046 -- ETRS89 / UTM zone 34N (N-E)                       10346 -- NSIDC Authalic Sphere␣
↪(geographic 2D)           9003 -- IGS97 (geographic 2D)
23034 -- ED50 / UTM zone 34N                              10606 -- WGS 84 (G2296)␣
↪(geographic 2D)                   9002 -- IGS97 (geographic 3D)
4231 -- ED87 (geographic 2D)                              10605 -- WGS 84 (G2296)␣
↪(geographic 3D)                   9006 -- IGS00 (geographic 2D)
4668 -- ED79 (geographic 2D)                              10781 -- ITRF2020-u2023␣
↪(geographic 2D)                   9005 -- IGS00 (geographic 3D)
10571 -- ETRF2020 (geographic 2D)                         10780 -- ITRF2020-u2023␣
↪(geographic 3D)                   9009 -- IGb00 (geographic 2D)
10570 -- ETRF2020 (geographic 3D)                         10785 -- IGb20 (geographic 2D) ␣
↪                          9008 -- IGb00 (geographic 3D)
4258 -- ETRS89 (geographic 2D)                            10784 -- IGb20 (geographic 3D) ␣
↪                          9012 -- IGS05 (geographic 2D)
4937 -- ETRS89 (geographic 3D)                            4087 -- WGS 84 / World␣
↪Equidistant Cylindrical           9011 -- IGS05 (geographic 3D)
9059 -- ETRF89 (geographic 2D)                            4276 -- NSWC 9Z-2 (geographic␣
↪2D)                         9014 -- IGS08 (geographic 2D)
7915 -- ETRF89 (geographic 3D)                            4322 -- WGS 72 (geographic 2D) ␣
↪                          9013 -- IGS08 (geographic 3D)
7917 -- ETRF90 (geographic 3D)                            4324 -- WGS 72BE (geographic␣
↪2D)                            9017 -- IGb08 (geographic 2D)
7919 -- ETRF91 (geographic 3D)                            4326 -- WGS 84 (geographic 2D) ␣
↪                          9016 -- IGb08 (geographic 3D)
7921 -- ETRF92 (geographic 3D)                            4740 -- PZ-90 (geographic 2D)  ␣
↪                          9019 -- IGS14 (geographic 2D)
7923 -- ETRF93 (geographic 3D)                            4760 -- WGS 66 (geographic 2D) ␣
↪                          9018 -- IGS14 (geographic 3D)
7925 -- ETRF94 (geographic 3D)                            4891 -- WGS 66 (geographic 3D) ␣
↪                          9053 -- WGS 84 (G730) (geographic 2D)
7927 -- ETRF96 (geographic 3D)                            4923 -- PZ-90 (geographic 3D)  ␣
↪                          9054 -- WGS 84 (G873) (geographic 2D)
```

(continues on next page)

(continued from previous page)

```
7929 -- ETRF97 (geographic 3D)                          4979 -- WGS 84 (geographic 3D) ␣
↪                        9055 -- WGS 84 (G1150) (geographic 2D)
8399 -- ETRF2005 (geographic 3D)                        4985 -- WGS 72 (geographic 3D) ␣
↪                        9056 -- WGS 84 (G1674) (geographic 2D)
8403 -- ETRF2014 (geographic 3D)                        4987 -- WGS 72BE (geographic␣
↪3D)                        9057 -- WGS 84 (G1762) (geographic 2D)
9060 -- ETRF90 (geographic 2D)                          6893 -- WGS 84 / World Mercator␣
↪+ EGM2008 height         9380 -- IGb14 (geographic 2D)
9061 -- ETRF91 (geographic 2D)                          7657 -- WGS 84 (G730)␣
↪(geographic 3D)                     9379 -- IGb14 (geographic 3D)
9062 -- ETRF92 (geographic 2D)                          7659 -- WGS 84 (G873)␣
↪(geographic 3D)                     9474 -- PZ-90.02 (geographic 2D)
9063 -- ETRF93 (geographic 2D)                          7661 -- WGS 84 (G1150)␣
↪(geographic 3D)                    9475 -- PZ-90.11 (geographic 2D)
9064 -- ETRF94 (geographic 2D)                          7663 -- WGS 84 (G1674)␣
↪(geographic 3D)                    9518 -- WGS 84 + EGM2008 height
9065 -- ETRF96 (geographic 2D)                          7665 -- WGS 84 (G1762)␣
↪(geographic 3D)                    9705 -- WGS 84 + MSL height
9066 -- ETRF97 (geographic 2D)                          7678 -- PZ-90.02 (geographic␣
↪3D)                        9707 -- WGS 84 + EGM96 height
9067 -- ETRF2000 (geographic 2D)                        7680 -- PZ-90.11 (geographic␣
↪3D)                        9755 -- WGS 84 (G2139) (geographic 2D)
7931 -- ETRF2000 (geographic 3D)                        7816 -- WGS 84 (Transit)␣
↪(geographic 3D)                   9754 -- WGS 84 (G2139) (geographic 3D)
9068 -- ETRF2005 (geographic 2D)                        7900 -- ITRF88 (geographic 3D) ␣
↪                        9990 -- ITRF2020 (geographic 2D)
9069 -- ETRF2014 (geographic 2D)                        7901 -- ITRF89 (geographic 3D) ␣
↪                        9989 -- ITRF2020 (geographic 3D)
```

Or use CRS explorer at spatialreference.org

and let's write as a replayable `.gdalg.json` file: https://gdal.org/en/stable/drivers/raster/gdalg.html

```
$ gdal raster scale TDR_rgb.tif TDR_rgb_byte_clamped.gdalg.json  \
        --input-min 400 \
        --input-max 2400 \
        --output-data-type uint8
```

```
$ cat TDR_rgb_byte_clamped.gdalg.json
```

```
{
  "type":"gdal_streamed_alg",
  "command_line":"gdal raster scale --input TDR_rgb.tif --output-data-type UInt8 --input-
↪min 400 --input-max 2400 --output-format stream --output streamed_dataset",
  "gdal_version":"3130000"
}
```

```
$ gdal info TDR_rgb_byte_clamped.gdalg.json
```

```
Driver: GDALG/GDAL Streamed Algorithm driver
Files: TDR_rgb_byte_clamped.gdalg.json
Size is 10980, 10980
```

(continues on next page)

(continued from previous page)

```
Coordinate Reference System:
  - name: WGS 84 / UTM zone 34N
  - ID: EPSG:32634
  - type: Projected
  - projection type: UTM zone 34N, Transverse Mercator
  - units: metre
  - area of use: Between 18°E and 24°E..., west 18.00, south 0.00, east 24.00, north 84.
↪00
Data axis to CRS axis mapping: 1,2
Origin = (399960.000000000000000,5100000.000000000000000)
Pixel Size = (10.000000000000000,-10.000000000000000)
Metadata:
  AOT_QUANTIFICATION_VALUE=1000.0
  AOT_QUANTIFICATION_VALUE_UNIT=none
  AOT_RETRIEVAL_ACCURACY=0.0
  AOT_RETRIEVAL_METHOD=SEN2COR_DDV
  AREA_OR_POINT=Area
  BOA_QUANTIFICATION_VALUE=10000
  BOA_QUANTIFICATION_VALUE_UNIT=none
  CAST_SHADOW_PERCENTAGE=0.020978
  CLOUDY_PIXEL_OVER_LAND_PERCENTAGE=13.652931
  CLOUD_COVERAGE_ASSESSMENT=13.622445
  CLOUD_SHADOW_PERCENTAGE=4.0E-6
  DATATAKE_1_DATATAKE_SENSING_START=2026-04-23T09:40:29.024Z
  DATATAKE_1_DATATAKE_TYPE=INS-NOBS
  DATATAKE_1_ID=GS2B_20260423T094029_047681_N05.12
  DATATAKE_1_SENSING_ORBIT_DIRECTION=DESCENDING
  DATATAKE_1_SENSING_ORBIT_NUMBER=36
  DATATAKE_1_SPACECRAFT_NAME=Sentinel-2B
  DEGRADED_ANC_DATA_PERCENTAGE=0.0
  DEGRADED_MSI_DATA_PERCENTAGE=0
  FORMAT_CORRECTNESS=PASSED
  GENERAL_QUALITY=PASSED
  GENERATION_TIME=2026-04-23T11:57:14.000000Z
  GEOMETRIC_QUALITY=PASSED
  GRANULE_MEAN_AOT=0.06598
  GRANULE_MEAN_WV=0.75046
  HIGH_PROBA_CLOUDS_PERCENTAGE=0.001836
  L2A_QUALITY=PASSED
  MEDIUM_PROBA_CLOUDS_PERCENTAGE=0.009661
  NODATA_PIXEL_PERCENTAGE=9.990737
  NOT_VEGETATED_PERCENTAGE=28.988278
  OZONE_SOURCE=AUX_ECMWFT
  OZONE_VALUE=416.167104
  PREVIEW_GEO_INFO=Not applicable
  PREVIEW_IMAGE_URL=Not applicable
  PROCESSING_BASELINE=05.12
  PROCESSING_LEVEL=Level-2A
  PRODUCT_DOI=https://doi.org/10.5270/S2_-znk9xsj
  PRODUCT_START_TIME=2026-04-23T09:40:29.024Z
  PRODUCT_STOP_TIME=2026-04-23T09:40:29.024Z
  PRODUCT_TYPE=S2MSI2A
```

(continues on next page)

(continued from previous page)

```
  PRODUCT_URI=S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE
  RADIATIVE_TRANSFER_ACCURACY=0.0
  RADIOMETRIC_QUALITY=PASSED
  REFERENCE_BAND=B4
  REFLECTANCE_CONVERSION_U=0.991700831171221
  SATURATED_DEFECTIVE_PIXEL_PERCENTAGE=0.0
  SENSOR_QUALITY=PASSED
  SNOW_ICE_PERCENTAGE=0.0
  SPECIAL_VALUE_NODATA=0
  SPECIAL_VALUE_SATURATED=65535
  THIN_CIRRUS_PERCENTAGE=13.610949
  UNCLASSIFIED_PERCENTAGE=0.243549
  VEGETATION_PERCENTAGE=56.460464
  WATER_PERCENTAGE=0.664281
  WATER_VAPOUR_RETRIEVAL_ACCURACY=0.0
  WVP_QUANTIFICATION_VALUE=1000.0
  WVP_QUANTIFICATION_VALUE_UNIT=cm
Image Structure Metadata:
  INTERLEAVE=PIXEL
Corner Coordinates:
Upper Left  (  399960.000, 5100000.000) ( 19d42'25.02"E, 46d 2'46.53"N)
Lower Left  (  399960.000, 4990200.000) ( 19d43'45.90"E, 45d 3'29.49"N)
Upper Right (  509760.000, 5100000.000) ( 21d 7'34.20"E, 46d 3'12.62"N)
Lower Right (  509760.000, 4990200.000) ( 21d 7'26.31"E, 45d 3'54.69"N)
Center      (  454860.000, 5045100.000) ( 20d25'17.86"E, 45d33'28.71"N)
Band 1 Block=256x256 Type=Byte, ColorInterp=Red
  Metadata:
    BANDNAME=B4
    BANDWIDTH=30
    BANDWIDTH_UNIT=nm
    BOA_ADD_OFFSET=-1000
    SOLAR_IRRADIANCE=1512.79
    SOLAR_IRRADIANCE_UNIT=W/m2/um
    WAVELENGTH=665
    WAVELENGTH_UNIT=nm
Band 2 Block=256x256 Type=Byte, ColorInterp=Green
  Metadata:
    BANDNAME=B3
    BANDWIDTH=35
    BANDWIDTH_UNIT=nm
    BOA_ADD_OFFSET=-1000
    SOLAR_IRRADIANCE=1824.93
    SOLAR_IRRADIANCE_UNIT=W/m2/um
    WAVELENGTH=560
    WAVELENGTH_UNIT=nm
Band 3 Block=256x256 Type=Byte, ColorInterp=Blue
  Metadata:
    BANDNAME=B2
    BANDWIDTH=65
    BANDWIDTH_UNIT=nm
    BOA_ADD_OFFSET=-1000
    SOLAR_IRRADIANCE=1959.75
```

(continues on next page)

(continued from previous page)

```
    SOLAR_IRRADIANCE_UNIT=W/m2/um
    WAVELENGTH=490
    WAVELENGTH_UNIT=nm
```

```
$ gdal raster reproject TDR_rgb_byte_clamped.gdalg.json TDR_3035.tif \
    --output-crs EPSG:3035 -r cubic --creation-option TILED=YES --creation-option␣
↪COMPRESS=JPEG
```

Let's look at it in QGIS.

Nice, but can we get rid of the black collar ?

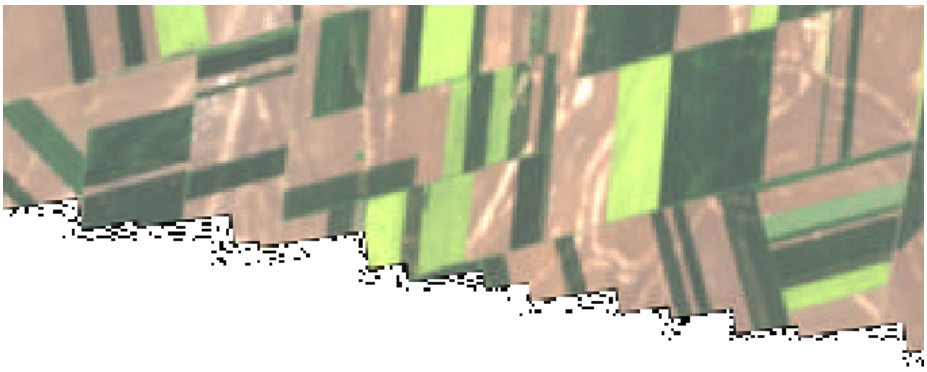
Sure let's say the nodata value (invalid value, sentinel value) to zero.

```
$ gdal raster edit TDR_3035.tif --nodata 0
```

Note

`gdal raster edit` is a bit of an exception in that it does not take an input and output dataset, but a single one. The "edit" wording should make it clear that this is edition in place.

Let's look again in QGIS. Zoom in on the south-western border. Hum "interesting" artifacts appear.



Let's clean them with gdal raster clean-collar

```
$ gdal raster clean-collar TDR_3035.tif TDR_3035_with_mask.tif \
    --add-mask --creation-option TILED=YES  --creation-option COMPRESS=JPEG
```

7.6 Exercise

Generate in a single step a tiled JPEG-compressed GeoTIFF reprojected image.

And make sure it includes overviews.

Hint

1. Use the `--input-nodata` and `--add-alpha` options
2. Use gdal raster overview add or use a format whose generation with GDAL automatically includes overviews.

==> *Exercise solution for raster reprojection*.

CHAPTER

EIGHT

TILING, MERGING

8.1 Raster mosaic

Let's create a virtual mosaic in VRT format using gdal raster mosaic

```
# run from the workshop data directory

$ gdal raster mosaic \
    --input-nodata 0 \
    SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/MTD_
↪MSIL2A.xml:10m:EPSG_32634 \
    SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TER_20260423T115714.SAFE/MTD_
↪MSIL2A.xml:10m:EPSG_32634 \
    SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TES_20260423T115714.SAFE/MTD_
↪MSIL2A.xml:10m:EPSG_32634 \
    s2.vrt
```

Let's add overviews:

```
$ gdal raster overview add -r cubic s2.vrt
```

This generates a `.vrt.ovr` file

Let's open `s2.vrt` in QGIS

8.2 Raster tiling

Using gdal raster tile

```
$ gdal raster tile s2.vrt s2_tiled --format JPEG
```

```
ERROR 6: tile: Only up to 4 bands supported for JPEG (with alpha ignored).
```

Preliminary operations:

```
$ gdal raster select s2.vrt s2_rgb.vrt --band 1,2,3
$ gdal raster scale s2_rgb.vrt s2_rgb_clamped.vrt \
        --input-min 400 \
        --input-max 2400 \
        --output-data-type uint8
```

Now let's generate the individual files:

```
$ gdal raster tile s2_rgb_clamped.vrt s2_tiled --format JPEG
```

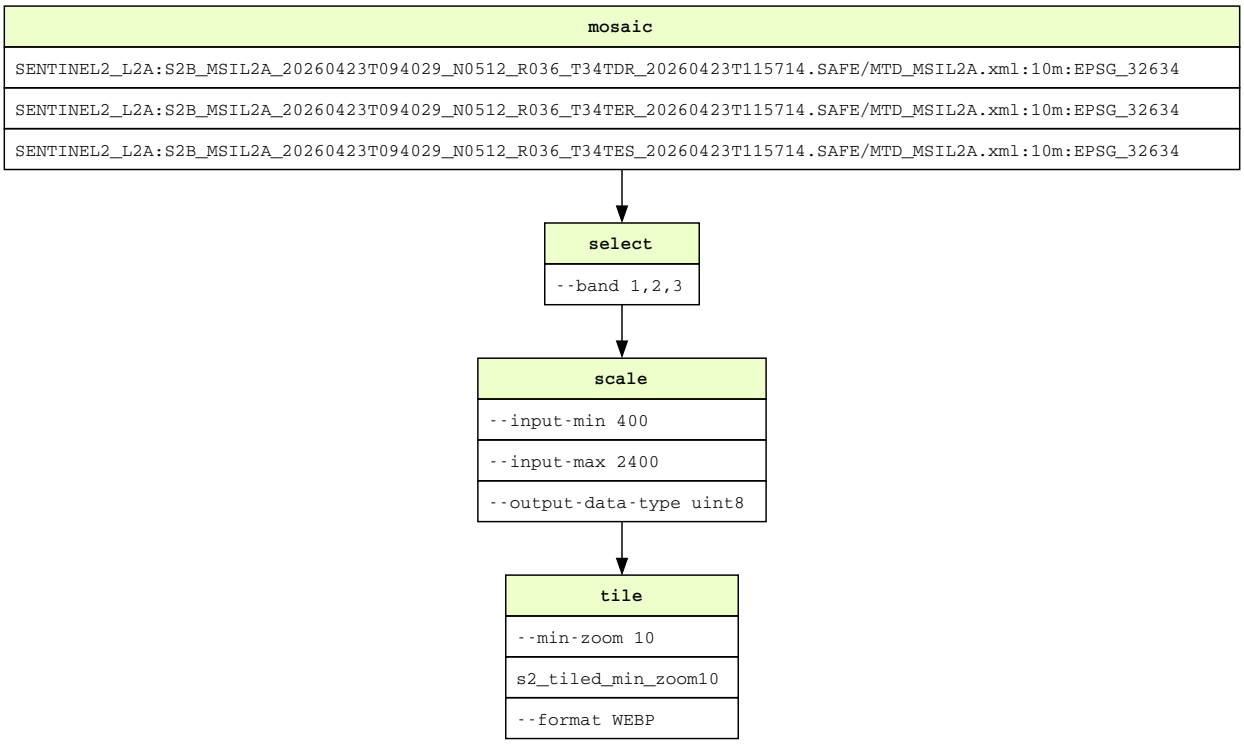
and check the result (if Firefox is not installed, manually open the file in the browser):

```
$ firefox s2_tiled/leaflet.html
```

8.3 Doing everything at the same time using a pipeline

Using gdal raster pipeline

```
$ gdal pipeline \
       mosaic SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_
↪20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634 \
       SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TER_20260423T115714.SAFE/
↪MTD_MSIL2A.xml:10m:EPSG_32634 \
       SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TES_20260423T115714.SAFE/
↪MTD_MSIL2A.xml:10m:EPSG_32634 \
       ! \
       select --band 1,2,3 \
       ! \
       scale --input-min 400 \
             --input-max 2400 \
             --output-data-type uint8 \
       ! \
       tile --min-zoom 10 s2_tiled_min_zoom10 --format WEBP
```



and check the result:

```
$ firefox s2_tiled_min_zoom10/openlayers.html
```

8.4 Exercise

Modify above pipeline to materialize the scaled dataset as a GeoTIFF before tiling.

Hint

Use gdal raster materialize

==> *Exercise solution for materializing pipeline intermediate result*.

8.5 Vector partitioning

Using gdal vector partition

First let's create a GeoPackage file from the original OSM data with a slightly customized import configuration with the OSM tag `amenity` being exposed as a top-level GDAL attribute:

```
$ gdal vector convert timisoara.osm.pbf --layer points --oo CONFIG_FILE=osm_conf_amenity.
↪ini amenity.gpkg
```

And let's create a partition around the values of that field:

```
$ gdal vector partition amenity.gpkg --field amenity amenity_partitioned

$ gdal vsi ls -lR amenity_partitioned
```

```
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=bar
-rw-r--r-- 1 unknown unknown       106496 2026-05-21 18:30 points/amenity=bar/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=place_of_
↪worship
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=place_of_
↪worship/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=nightclub
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=nightclub/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=loading_dock
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=loading_dock/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=events_venue
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=events_venue/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=shelter
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=shelter/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=payment_
↪terminal
```

(continues on next page)

(continued from previous page)

```
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=payment_
↪terminal/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=fast_food
-rw-r--r-- 1 unknown unknown       131072 2026-05-21 18:30 points/amenity=fast_food/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=veterinary
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=veterinary/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=childcare
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=childcare/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=bus_station
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=bus_station/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=pharmacy
-rw-r--r-- 1 unknown unknown       139264 2026-05-21 18:30 points/amenity=pharmacy/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=casino
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=casino/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=smoking_area
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=smoking_area/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=recycling
-rw-r--r-- 1 unknown unknown       270336 2026-05-21 18:30 points/amenity=recycling/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=telephone
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=telephone/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=cinema
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=cinema/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=waste_transfer_
↪station
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=waste_transfer_
↪station/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=waste_basket
-rw-r--r-- 1 unknown unknown       126976 2026-05-21 18:30 points/amenity=waste_basket/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=ice_cream
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=ice_cream/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=university
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=university/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=conference_
↪centre
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=conference_
↪centre/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=dancing_school
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=dancing_school/
↪part_0000000001.gpkg
```

(continues on next page)

(continued from previous page)

```
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=vehicle_
↪inspection
-rw-r--r-- 1 unknown unknown       98304 2026-05-21 18:30 points/amenity=vehicle_
↪inspection/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=parcel_locker
-rw-r--r-- 1 unknown unknown      135168 2026-05-21 18:30 points/amenity=parcel_locker/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=marketplace
-rw-r--r-- 1 unknown unknown       98304 2026-05-21 18:30 points/amenity=marketplace/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=dentist
-rw-r--r-- 1 unknown unknown      110592 2026-05-21 18:30 points/amenity=dentist/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=bank
-rw-r--r-- 1 unknown unknown      139264 2026-05-21 18:30 points/amenity=bank/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=clock
-rw-r--r-- 1 unknown unknown       98304 2026-05-21 18:30 points/amenity=clock/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=gambling
-rw-r--r-- 1 unknown unknown       98304 2026-05-21 18:30 points/amenity=gambling/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=studio
-rw-r--r-- 1 unknown unknown       98304 2026-05-21 18:30 points/amenity=studio/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=post_box
-rw-r--r-- 1 unknown unknown       98304 2026-05-21 18:30 points/amenity=post_box/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=kindergarten
-rw-r--r-- 1 unknown unknown       98304 2026-05-21 18:30 points/amenity=kindergarten/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=charging_
↪station
-rw-r--r-- 1 unknown unknown      110592 2026-05-21 18:30 points/amenity=charging_
↪station/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=school
-rw-r--r-- 1 unknown unknown       98304 2026-05-21 18:30 points/amenity=school/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=courthouse
-rw-r--r-- 1 unknown unknown       98304 2026-05-21 18:30 points/amenity=courthouse/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=language_school
-rw-r--r-- 1 unknown unknown       98304 2026-05-21 18:30 points/amenity=language_
↪school/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=pub
-rw-r--r-- 1 unknown unknown      106496 2026-05-21 18:30 points/amenity=pub/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=vending_machine
-rw-r--r-- 1 unknown unknown       98304 2026-05-21 18:30 points/amenity=vending_
↪machine/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown        4096 2026-05-21 18:30 points/amenity=arts_centre
-rw-r--r-- 1 unknown unknown       98304 2026-05-21 18:30 points/amenity=arts_centre/
```

(continues on next page)

(continued from previous page)

```
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=fuel
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=fuel/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=ferry_terminal
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=ferry_terminal/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=public_bookcase
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=public_
↪bookcase/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=clinic
-rw-r--r-- 1 unknown unknown       106496 2026-05-21 18:30 points/amenity=clinic/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=community_
↪centre
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=community_
↪centre/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=doctors
-rw-r--r-- 1 unknown unknown       106496 2026-05-21 18:30 points/amenity=doctors/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=social_facility
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=social_
↪facility/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=car_wash
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=car_wash/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=parking
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=parking/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=food_court
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=food_court/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=bicycle_parking
-rw-r--r-- 1 unknown unknown       110592 2026-05-21 18:30 points/amenity=bicycle_
↪parking/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=theatre
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=theatre/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=post_office
-rw-r--r-- 1 unknown unknown       106496 2026-05-21 18:30 points/amenity=post_office/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=funeral_hall
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=funeral_hall/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=car_rental
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=car_rental/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=toilets
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=toilets/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=waste_disposal
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=waste_disposal/
```

(continues on next page)

(continued from previous page)

```
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=payment_centre
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=payment_centre/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=bicycle_rental
-rw-r--r-- 1 unknown unknown       106496 2026-05-21 18:30 points/amenity=bicycle_rental/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=fountain
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=fountain/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=atm
-rw-r--r-- 1 unknown unknown       114688 2026-05-21 18:30 points/amenity=atm/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=__HIVE_DEFAULT_
↪PARTITION__
-rw-r--r-- 1 unknown unknown      3194880 2026-05-21 18:30 points/amenity=__HIVE_DEFAULT_
↪PARTITION__/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=biergarten
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=biergarten/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=taxi
-rw-r--r-- 1 unknown unknown       106496 2026-05-21 18:30 points/amenity=taxi/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=disused
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=disused/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=bureau_de_
↪change
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=bureau_de_
↪change/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=veterinary_
↪pharmacy
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=veterinary_
↪pharmacy/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=townhall
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=townhall/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=library
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=library/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=drinking_water
-rw-r--r-- 1 unknown unknown       118784 2026-05-21 18:30 points/amenity=drinking_water/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=police
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=police/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=hospital
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=hospital/part_
↪0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=bench
-rw-r--r-- 1 unknown unknown       180224 2026-05-21 18:30 points/amenity=bench/part_
↪0000000001.gpkg
```

(continues on next page)

(continued from previous page)

```
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=bicycle_repair_
↪station
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=bicycle_repair_
↪station/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=restaurant
-rw-r--r-- 1 unknown unknown       147456 2026-05-21 18:30 points/amenity=restaurant/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=driving_school
-rw-r--r-- 1 unknown unknown        98304 2026-05-21 18:30 points/amenity=driving_school/
↪part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=parking_
↪entrance
-rw-r--r-- 1 unknown unknown       106496 2026-05-21 18:30 points/amenity=parking_
↪entrance/part_0000000001.gpkg
drwxr-xr-x 1 unknown unknown         4096 2026-05-21 18:30 points/amenity=cafe
-rw-r--r-- 1 unknown unknown       126976 2026-05-21 18:30 points/amenity=cafe/part_
↪0000000001.gpkg
```

8.6 Vector concatenation

Using gdal vector concat

```
--mode <MODE>                                              Determine the strategy to␣
↪create output layers from source layers . MODE=merge-per-layer-name|stack|single␣
↪(default: merge-per-layer-name)
--output-layer <OUTPUT-LAYER>                              Name of the output vector␣
↪layer (single mode), or template to name the output vector layers (stack mode)
--source-layer-field-name <SOURCE-LAYER-FIELD-NAME>        Name of the new field to add␣
↪to contain identification of the source layer, with value determined from 'source-
↪layer-field-content'
--source-layer-field-content <SOURCE-LAYER-FIELD-CONTENT>  A string, possibly using
↪{AUTO_NAME}, {DS_NAME}, {DS_BASENAME}, {DS_INDEX}, {LAYER_NAME}, {LAYER_INDEX}
--field-strategy <FIELD-STRATEGY>                          How to determine target␣
↪fields from source fields. FIELD-STRATEGY=union|intersection (default: union)
-s, --input-crs <INPUT-CRS>                                Input CRS
-d, --output-crs <OUTPUT-CRS>                              Output CRS
```

Example:

```
$ gdal vector concat $(find amenity_partitioned -name "*.gpkg") concatenated.gpkg
```

CHAPTER

NINE

DEM, VSI AND NESTED PIPELINES

9.1 Failed attempt at getting a remote DEM

Let's find a DEM:

https://ec.europa.eu/eurostat/web/gisco/geodata/digital-elevation-model/eu-dem

```
$ gdal vsi ls -l /vsicurl/https://gisco-services.ec.europa.eu/dem/100k/EU_DEM_mosaic_
↪1000K.ZIP
```

```
---------- 1 unknown unknown  25648052226 1970-01-01 00:00 /vsicurl/https://gisco-
↪services.ec.europa.eu/dem/100k/EU_DEM_mosaic_1000K.ZIP
```

25 GB... hum

Note

If you are using the *MSYS2 on Windows environment*, you may see the error below:

```
ERROR 3: list: 'C:\gdal\msys64\vsicurl\https;\\gisco-services.ec.europa.eu\dem\100k\
↪EU_DEM_mosaic_1000K.ZIP' does not exist or cannot be accessed
```

This is caused by MSYS2 path conversion (see https://www.msys2.org/docs/filesystem-paths/), which rewrites the URL passed to GDAL and breaks */vsicurl/*. Fix it by disabling path conversion for the rest of this tutorial session with:

```
$ export MSYS_NO_PATHCONV=1
```

Let's have a look inside:

```
$ gdal vsi ls -l /vsizip/vsicurl/https://gisco-services.ec.europa.eu/dem/100k/EU_DEM_
↪mosaic_1000K.ZIP
```

```
---------- 1 unknown unknown  24145219947 2013-10-02 19:55 eudem_dem_3035_europe.tif
---------- 1 unknown unknown   2374678674 2013-10-02 16:06 eudem_dem_3035_europe.tif.ovr
---------- 1 unknown unknown       290568 2013-10-16 10:16 metadata_iso19139.pdf
---------- 1 unknown unknown        18269 2013-10-16 10:00 metadata_iso19139.xml
---------- 1 unknown unknown         2836 2013-11-04 11:27 readme.txt
```

Let's be brave and inspect that huge TIFF file:

```
$ gdal info /vsizip/vsicurl/https://gisco-services.ec.europa.eu/dem/100k/EU_DEM_mosaic_
↪1000K.ZIP/eudem_dem_3035_europe.tif
```

```
Driver: GTiff/GeoTIFF
[ ... takes forever ... ]
```

If that file had not been a compressed file (that is not using ZIP deflate compression), that would have been just fine, but nothing we can do about bad choices of data producers.

9.2 Successful attempt

Fortunately https://github.com/OpenTopography/OT_BulkAccess_COGs/blob/main/OT_BulkAccessCOGs.ipynb provides a useful hint, with a VRT example:

```
$ gdal raster info --no-fl /vsicurl/https://opentopography.s3.sdsc.edu/raster/SRTM_GL1/
↪SRTM_GL1_srtm.vrt
```

```
Driver: VRT/Virtual Raster
Size is 1296001, 417601
Coordinate Reference System:
  - name: WGS 84
  - ID: EPSG:4326
  - type: Geographic 2D
Data axis to CRS axis mapping: 2,1
Origin = (-180.000138888888898,60.000138888888891)
Pixel Size = (0.000277777777778,-0.000277777777778)
Corner Coordinates:
Upper Left  (-180.0001389,  60.0001389) (180d 0' 0.50"W, 60d 0' 0.50"N)
Lower Left  (-180.0001389, -56.0001389) (180d 0' 0.50"W, 56d 0' 0.50"S)
Upper Right ( 180.0001389,  60.0001389) (180d 0' 0.50"E, 60d 0' 0.50"N)
Lower Right ( 180.0001389, -56.0001389) (180d 0' 0.50"E, 56d 0' 0.50"S)
Center      (   0.0000000,   2.0000000) (  0d 0' 0.00"E,  2d 0' 0.00"N)
Band 1 Block=128x128 Type=Int16, ColorInterp=Gray
  NoData Value=-32768
```

Now we can selectively download DEM covering our 3 Sentinel 2 tiles with:

```
# run from the workshop data directory

$ gdal raster clip \
    /vsicurl/https://opentopography.s3.sdsc.edu/raster/SRTM_GL1/SRTM_GL1_srtm.vrt \
    dem.tif \
    --like s2.vrt \
    --co COMPRESS=LZW --co TILED=YES --co PREDICTOR=2 --overwrite
```

9.3 Other successful attempt (using `/vsis3/`)

From https://hub.openeo.org/, Copernicus Data Space Ecosystem / Collections / COPERNICUS_30 / Items, one reaches https://stac.dataspace.copernicus.eu/v1/collections/cop-dem-glo-30-dged-cog/items which shows that one provider has a self-hosted (at least not under Amazon URL) S3 bucket at https://eodata.dataspace.copernicus.eu, which requires creating an account and getting credentials as indicated in https://documentation.dataspace.copernicus.eu/APIs/S3.html

Documentation of GDAL S3 related configuration options at: https://gdal.org/en/stable/user/virtual_file_systems.html#vsis3-aws-s3-files

Let's list the bucket content

```
$ gdal vsi ls -l /vsis3/eodata \
    --config AWS_S3_ENDPOINT=https://eodata.dataspace.copernicus.eu \
    --config AWS_ACCESS_KEY_ID=$EODATA_AWS_ACCESS_KEY_ID \
    --config AWS_SECRET_ACCESS_KEY=$EODATA_AWS_SECRET_ACCESS_KEY \
    --config AWS_VIRTUAL_HOSTING=NO
```

```
d--------- 1 unknown unknown            0 1970-01-01 00:00 C3S
d--------- 1 unknown unknown            0 1970-01-01 00:00 CAMS
d--------- 1 unknown unknown            0 1970-01-01 00:00 CEMS
d--------- 1 unknown unknown            0 1970-01-01 00:00 CLMS
d--------- 1 unknown unknown            0 1970-01-01 00:00 CLMS_archive
d--------- 1 unknown unknown            0 1970-01-01 00:00 Envisat
d--------- 1 unknown unknown            0 1970-01-01 00:00 Envisat-ASAR
d--------- 1 unknown unknown            0 1970-01-01 00:00 Global-Mosaics
d--------- 1 unknown unknown            0 1970-01-01 00:00 Jason-3
d--------- 1 unknown unknown            0 1970-01-01 00:00 Landsat-5
d--------- 1 unknown unknown            0 1970-01-01 00:00 Landsat-7
d--------- 1 unknown unknown            0 1970-01-01 00:00 Landsat-8-ESA
d--------- 1 unknown unknown            0 1970-01-01 00:00 SMOS
d--------- 1 unknown unknown            0 1970-01-01 00:00 SRTM
d--------- 1 unknown unknown            0 1970-01-01 00:00 Sentinel-1
d--------- 1 unknown unknown            0 1970-01-01 00:00 Sentinel-1-RTC
d--------- 1 unknown unknown            0 1970-01-01 00:00 Sentinel-2
d--------- 1 unknown unknown            0 1970-01-01 00:00 Sentinel-3
d--------- 1 unknown unknown            0 1970-01-01 00:00 Sentinel-5P
d--------- 1 unknown unknown            0 1970-01-01 00:00 Sentinel-6
d--------- 1 unknown unknown            0 1970-01-01 00:00 auxdata
```

Hint

You can streamline the access to such buckets by setting the credentials in a `$HOME/.gdal/gdalrc` file (or `$USERPROFILE/.gdal/gdalrc` on Windows and MYSYS2 e.g. `C:\Users\my_user_name\.gdal\gdalrc`)

```
[credentials]

[.eodata]
path=/vsis3/eodata
AWS_ACCESS_KEY_ID=<your-access-key-id-here>
AWS_SECRET_ACCESS_KEY=<your-secret-access-key-here>
AWS_S3_ENDPOINT=https://eodata.dataspace.copernicus.eu
AWS_VIRTUAL_HOSTING=NO
```

and them simply access a DEM 1 degree x 1 degree tile with:

```
$ gdal raster info \
    /vsis3/eodata/auxdata/CopDEM_COG/copernicus-dem-30m/Copernicus_DSM_COG_10_N45_00_
↪E021_00_DEM/Copernicus_DSM_COG_10_N45_00_E021_00_DEM.tif
```

```
Driver: GTiff/GeoTIFF
Files: /vsis3/eodata/auxdata/CopDEM_COG/copernicus-dem-30m/Copernicus_DSM_COG_10_N45_00_
↪E021_00_DEM/Copernicus_DSM_COG_10_N45_00_E021_00_DEM.tif
Size is 3600, 3600
Coordinate Reference System:
  - name: WGS 84
  - ID: EPSG:4326
  - type: Geographic 2D
  - area of use: World, west -180.00, south -90.00, east 180.00, north 90.00
Data axis to CRS axis mapping: 2,1
Origin = (20.999861111111112,46.000138888888891)
Pixel Size = (0.000277777777778,-0.000277777777778)
Metadata:
  AREA_OR_POINT=Point
Image Structure Metadata:
  LAYOUT=COG
  COMPRESSION=DEFLATE
  INTERLEAVE=BAND
  PREDICTOR=3
Corner Coordinates:
Upper Left  (  20.9998611,  46.0001389) ( 20d59'59.50"E, 46d 0' 0.50"N)
Lower Left  (  20.9998611,  45.0001389) ( 20d59'59.50"E, 45d 0' 0.50"N)
Upper Right (  21.9998611,  46.0001389) ( 21d59'59.50"E, 46d 0' 0.50"N)
Lower Right (  21.9998611,  45.0001389) ( 21d59'59.50"E, 45d 0' 0.50"N)
Center      (  21.4998611,  45.5001389) ( 21d29'59.50"E, 45d30' 0.50"N)
Band 1 Block=1024x1024 Type=Float32, ColorInterp=Gray
  Overviews: 1800x1800, 900x900, 450x450
```

9.4 Hypsometric rendering

With gdal raster color-map.

and following color ramp placed in `test.cpt` (Source: "elevation1" from https://hub.qgis.org/styles/28/ ported from XML)

```
0%     74,156,15,255
25%    255,250,104,255
50%    255,179,38,255
75%    146,100,30,255
100%   255,255,255,255
nv     0,0,0,0
```

Note

Above is optimized for high contrast for the analyzed area. Not appropriate for comparing different regions with different elevation ranges.

```
$ gdal raster color-map dem.tif --color-map test.cpt dem_colorized.tif
```

Result:

9.5 Shaded map

With gdal raster hillshade.

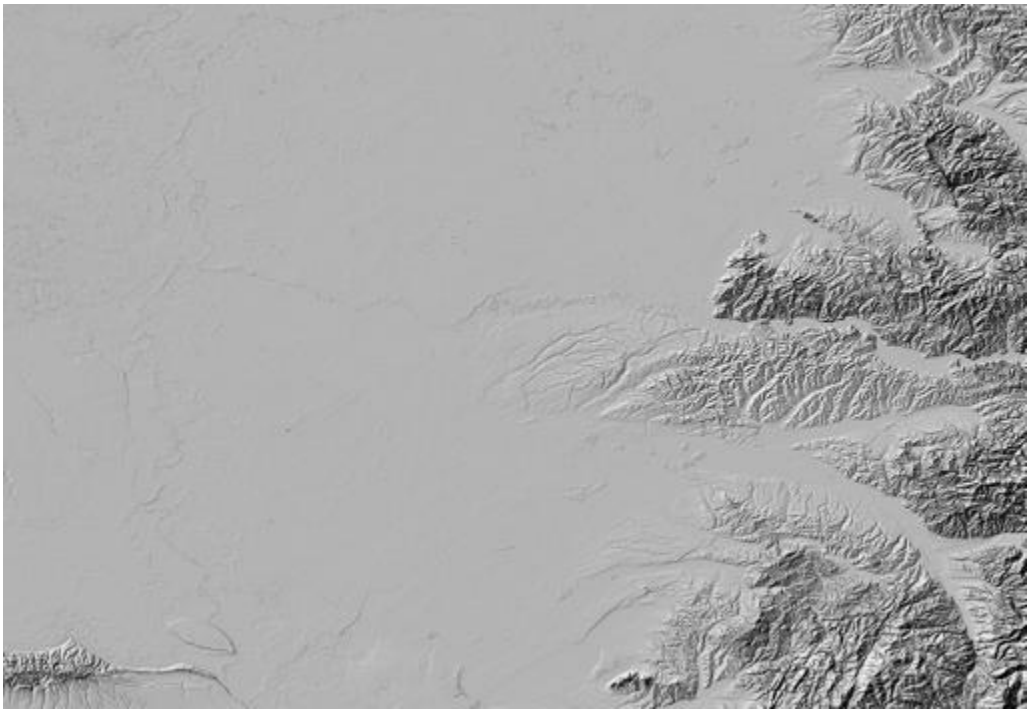
```
$ gdal raster hillshade dem.tif dem_hillshade.tif
```

Result:

You can exaggerate the effect of altitudes to accentuate slopes

```
$ gdal raster hillshade dem.tif dem_hillshade_z_5.tif --zfactor 5
```

Result:

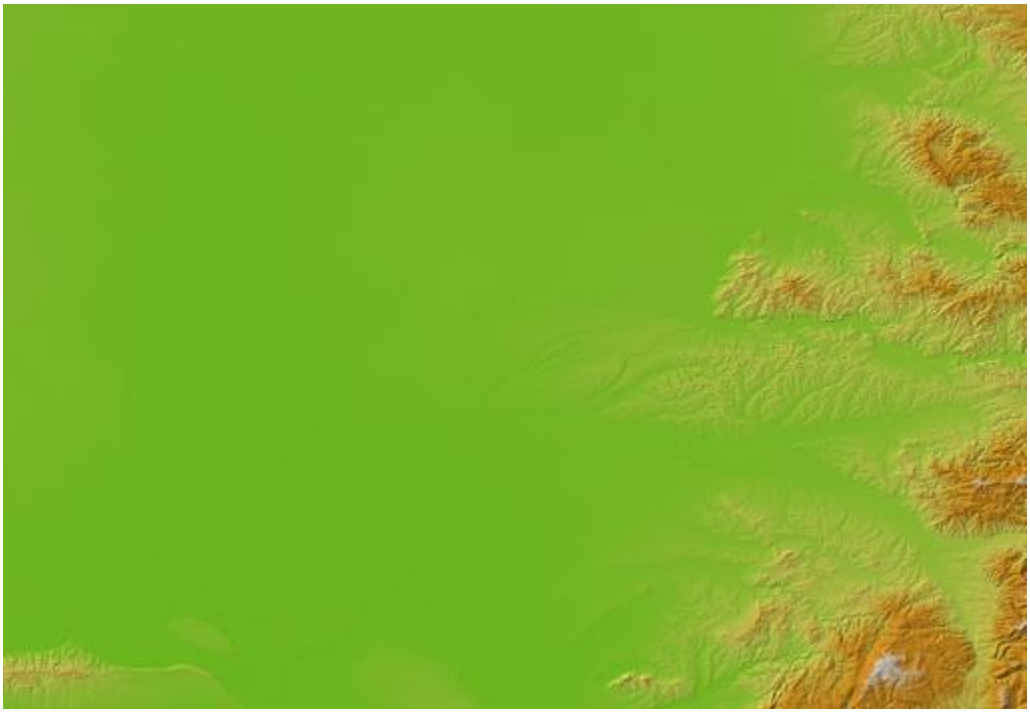


9.6 Combining hypsometric rendering and hillshade

With gdal raster blend.

```
$ gdal raster blend --color-input dem_colorized.tif \
       --overlay dem_hillshade.tif \
       --output dem_colorized_hillshade.tif \
       --operator hsv-value
```

Result:



You can adjust the opacity of the overlay layer:

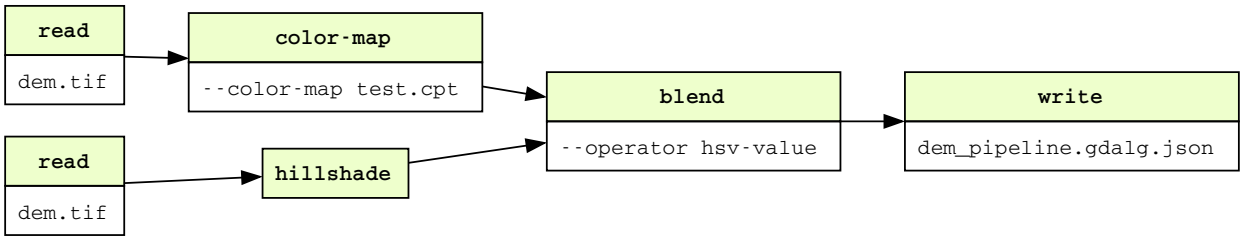
```
$ gdal raster blend --color-input dem_colorized.tif \
        --overlay dem_hillshade.tif \
        --output dem_colorized_hillshade_50pct.tif \
        --operator hsv-value --opacity 50
```

Result:

9.7 Doing it in one step: nested pipeline !

Using gdal raster pipeline, and write is a `.gdalg.json` file

```
$ gdal raster pipeline \
    read dem.tif ! \
    color-map --color-map test.cpt ! \
    blend [ read dem.tif ! hillshade ] --operator hsv-value ! \
    write dem_pipeline.gdalg.json
```



And let's open it in QGIS.

9.8 Exercise

Modify the above pipeline:

1. such that the hillshade stage appears first and the `color-map` one is done as a nested input pipeline.

Hint

Specify the hypsometric rendering with a nested input pipeline as the value of the `--input` argument of `blend` and use the special placeholder _PIPE_

2. to generate colorized and hillshaded maps as intermediate results

Hint

1. Use the tee pipeline operator
2. You may also use it inside a nested input pipeline

==> *Exercise solution for DEM pipeline*.

CHAPTER

TEN

PIXEL OPERATIONS

10.1 Pixel-wise computations

Using gdal raster calc

It performs pixel-wise calculations on one or more input GDAL datasets. This example uses the `s2_TER_10m.xml` dataset created in the *Hidden feature: symbolic links and subdatasets* section of the tutorial.

Let's compute a grayscale view using the well-known formula:

$$Y = 0.299 * Red + 0.587 * Green + 0.114 * Blue$$

```
$ gdal raster calc --input s2_TER_10m.xml  --output grayscale.tif --output-data-type␣
↪UInt16 --calc "0.299 * X[1] + 0.587 * X[2] + 0.114 * X[3]"
```

Let's use an aggregate function `avg` with `--flatten` to indicate we only want one single output band, so that avg(X) is expanded to `avg(X[1], X[2], X[3], X[4], X[5], X[6]`.

```
$  gdal raster calc --input s2_TER_10m.xml --output avg.tif --output-data-type UInt16 --
↪flatten --calc "avg(X)"
```

We can also doing the same using the `mean` builtin function;

```
$ gdal raster calc --input s2_TER_10m.xml --output mean.tif --output-data-type UInt16 --
↪flatten --calc "mean" --dialect builtin

$ gdal raster compare avg.tif mean.tif
```

10.2 Exercise

1. Compute the Normalized difference vegetation index (NDVI) using the well-know formula:

$$NDVI = \frac{NIR - Red}{NIR + Red}$$

2. Do the same but after having separated the NIR and Red bands into 2 separate files, and do not create any materialized (i.e. actual image) file in the process.
3. Create a .gdalg.json file with a simple pipeline computing the NDVI for one file, and replay that pipeline using substitutions to apply it to another Sentinel 2 tile

==> *Exercise solution for gdal raster calc*

10.3 Focal statistics

Using gdal raster neighbors

Compute the value of each pixel from its neighbors (focal statistics).

Let's attempt edge detection with the `edge1` kernel:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{10.1}$$

```
$ gdal raster neighbors s2_TER_10m.xml edge1.tif --kernel edge1 --overwrite
```

Result:

10.4 Zonal statistics

Using gdal raster zonal-stats

We are going to compute the average elevation in provinces around Timișoara.

First let find out the extent of our DEM:

```
$ gdal raster info dem.tif
```

```
Upper Left  (  19.6854167,  46.9537500) ( 19d41' 7.50"E, 46d57'13.50"N)
Lower Left  (  19.6854167,  45.0565278) ( 19d41' 7.50"E, 45d 3'23.50"N)
```

(continues on next page)

(continued from previous page)

```
Upper Right (  22.4426389,  46.9537500) ( 22d26'33.50"E, 46d57'13.50"N)
Lower Right (  22.4426389,  45.0565278) ( 22d26'33.50"E, 45d 3'23.50"N)
Center      (  21.0640278,  46.0051389) ( 21d 3'50.50"E, 46d 0'18.50"N)
```

Let's extract (but not clip) provinces that intersects that extent

```
$ gdal vector filter --bbox=19.6854167,45.0565278,22.4426389,46.9537500 \
    /vsizip/ne_10m_admin_1_states_provinces.zip admin_1_around_timis.gpkg
```

Note

If you are using the *MSYS2 on Windows environment*, and you set `MSYS_NO_PATHCONV=1` in the *VSI tutorial* you may see the error below:

```
ERROR 4: C:/gdal/msys64/vsizip/ne_10m_admin_1_states_provinces.zip: No such file or␣
↪directory
```

Switch path conversion back on with:

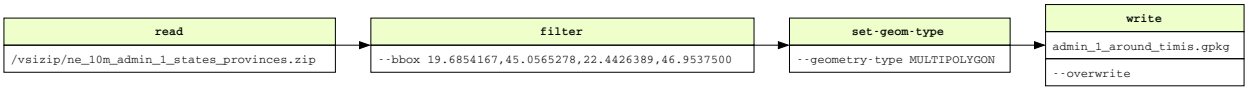
```
$ export MSYS_NO_PATHCONV=0
```

When run successfully the output will be:

```
Warning 1: A geometry of type MULTIPOLYGON is inserted into layer ne_10m_admin_1_states_
↪provinces of geometry type POLYGON, which is not normally allowed by the GeoPackage␣
↪specification, but the driver will however do it. To create a conformant GeoPackage,␣
↪if using ogr2ogr, the -nlt option can be used to override the layer geometry type.␣
↪This warning will no longer be emitted for this combination of layer and feature␣
↪geometry type.
```

Not critical, but we can fix it cleanly with:

```
$ gdal vector pipeline read /vsizip/ne_10m_admin_1_states_provinces.zip ! \
    filter --bbox=19.6854167,45.0565278,22.4426389,46.9537500 ! \
    set-geom-type --geometry-type MULTIPOLYGON ! \
    write admin_1_around_timis.gpkg --overwrite
```



And finally:

```
$ gdal raster zonal-stats dem.tif --stat mean --zones admin_1_around_timis.gpkg \
    dem_zonal_mean.gpkg --include-field admin,name --overwrite --include-geom
```

10.5 Exercise

Find the point of maximal elevation in each zone.

Bonus point for a pipeline avoiding the creation of the materialized `admin_1_around_timis.gpkg`.

Hint

Look at documented examples of gdal raster zonal-stats

==> *Exercise solution for gdal raster zonal-stats*

CHAPTER

ELEVEN

GDAL MULTIDIMENSIONAL TOOLS

11.1 Downloading and extracting information using GDAL multidimensional API

Using gdal mdim convert

Extracting NOAA Global Forecast System (GFS) data available from https://registry.opendata.aws/noaa-gfs-bdp-pds/

```
$ gdal mdim convert "/vsis3/noaa-gfs-bdp-pds/gdas.20260519/00/atmos/gdas.t00z.sfcanl.nc"␣
↪--config AWS_NO_SIGN_REQUEST=YES --array /tmp2m --output 20260519_00_tmp2m.nc

$ gdal mdim convert "/vsis3/noaa-gfs-bdp-pds/gdas.20260519/06/atmos/gdas.t06z.sfcanl.nc"␣
↪--config AWS_NO_SIGN_REQUEST=YES --array /tmp2m --output 20260519_06_tmp2m.nc
```

Warning

The above may only work properly on Linux due to limitations in the netCDF driver regarding working with files in `/vsi` virtual file systems, hence the converted files `20260519_00_tmp2m.nc` and `20260519_06_tmp2m.nc` are provided in the input datasets.

11.2 Inspecting

Using gdal mdim info

Let's have a look:

```
$ gdal mdim info 20260519_00_tmp2m.nc
```


```
"time": {
  "full_name": "/time",
  "datatype": "Float64",
  "dimensions": [
    "/time"
  ],
  "dimension_size": [
    1
  ],
  "attributes": {
    "calendar": "JULIAN",
```


(continues on next page)

(continued from previous page)


```
    "calendar_type": "JULIAN",
    "cartesian_axis": "T",
    "long_name": "time"
  },
  "unit": "hours since 2026-05-19 00:00:00"
}
```


```
$ gdal mdim info 20260519_06_tmp2m.nc
```


```
"time": {
  "full_name": "/time",
  "datatype": "Float64",
  "dimensions": [
    "/time"
  ],
  "dimension_size": [
    1
  ],
  "attributes": {
    "calendar": "JULIAN",
    "calendar_type": "JULIAN",
    "cartesian_axis": "T",
    "long_name": "time"
  },
  "unit": "hours since 2026-05-19 06:00:00"
}
```


We are going to modify the "time" (single) value of `20260519_06_tmp2m.nc` so it is relative to "hours since 2026-05-19 00:00:00" by first creating a multidimensional VRT

```
$ gdal mdim convert 20260519_06_tmp2m.nc 20260519_06_tmp2m.vrt
$ cp 20260519_06_tmp2m.vrt 20260519_06_tmp2m_mod.vrt
```

And modifying the time array as following:


```
<Array name="time">
  <DataType>Float64</DataType>
  <DimensionRef ref="time" />
  <Unit>hours since 2026-05-19 00:00:00</Unit>
  <InlineValuesWithValueElement><Value>6</Value></InlineValuesWithValueElement>
  <Attribute name="calendar">
    <DataType>String</DataType>
    <Value>JULIAN</Value>
  </Attribute>
  <Attribute name="calendar_type">
    <DataType>String</DataType>
    <Value>JULIAN</Value>
  </Attribute>
  <Attribute name="cartesian_axis">
    <DataType>String</DataType>
    <Value>T</Value>
  </Attribute>
```


(continues on next page)

(continued from previous page)

```
  <Attribute name="long_name">
    <DataType>String</DataType>
    <Value>time</Value>
  </Attribute>
</Array>
```

Now we can convert it back to netCDF:

```
$ gdal mdim convert 20260519_06_tmp2m_mod.vrt 20260519_06_tmp2m_mod.nc
```

11.3 Mosaicing / creating a 3D cube

Using gdal mdim mosaic

```
$ gdal mdim mosaic 20260519_00_tmp2m.nc 20260519_06_tmp2m_mod.nc 20260519_00_06_tmp2m.
↪zarr --format Zarr

$ gdal mdim info 20260519_00_06_tmp2m.zarr
```


```
{
  "type": "group",
  "driver": "Zarr",
  "name": "/",
  "dimensions": [
    {
      "name": "grid_xt",
      "full_name": "/grid_xt",
      "size": 3072,
      "indexing_variable": {
        "grid_xt": {
          "full_name": "/grid_xt",
          "datatype": "Float64",
          "dimensions": [
            "/grid_xt"
          ],
          "dimension_size": [
            3072
          ],
          "block_size": [
            3072
          ],
          "attributes": {
            "cartesian_axis": "X",
            "long_name": "T-cell longitude"
          }
        }
      }
    },
    {
      "name": "grid_yt",
      "full_name": "/grid_yt",
```


(continues on next page)

(continued from previous page)


```
      "size": 1536,
      "indexing_variable": {
        "grid_yt": {
          "full_name": "/grid_yt",
          "datatype": "Float64",
          "dimensions": [
            "/grid_yt"
          ],
          "dimension_size": [
            1536
          ],
          "block_size": [
            1536
          ],
          "attributes": {
            "cartesian_axis": "Y",
            "long_name": "T-cell latitude"
          }
        }
      }
    },
    {
      "name": "time",
      "full_name": "/time",
      "size": 2,
      "indexing_variable": {
        "time": {
          "full_name": "/time",
          "datatype": "Float64",
          "dimensions": [
            "/time"
          ],
          "dimension_size": [
            2
          ],
          "block_size": [
            2
          ],
          "attributes": {
            "calendar": "JULIAN",
            "calendar_type": "JULIAN",
            "cartesian_axis": "T",
            "long_name": "time"
          }
        }
      }
    }
  ],
  "arrays": {
    "grid_xt": {
      "full_name": "/grid_xt",
      "datatype": "Float64",
```


(continues on next page)

(continued from previous page)


```
      "dimensions": [
        "/grid_xt"
      ],
      "dimension_size": [
        3072
      ],
      "block_size": [
        3072
      ],
      "attributes": {
        "cartesian_axis": "X",
        "long_name": "T-cell longitude"
      }
    },
    "grid_yt": {
      "full_name": "/grid_yt",
      "datatype": "Float64",
      "dimensions": [
        "/grid_yt"
      ],
      "dimension_size": [
        1536
      ],
      "block_size": [
        1536
      ],
      "attributes": {
        "cartesian_axis": "Y",
        "long_name": "T-cell latitude"
      }
    },
    "time": {
      "full_name": "/time",
      "datatype": "Float64",
      "dimensions": [
        "/time"
      ],
      "dimension_size": [
        2
      ],
      "block_size": [
        2
      ],
      "attributes": {
        "calendar": "JULIAN",
        "calendar_type": "JULIAN",
        "cartesian_axis": "T",
        "long_name": "time"
      }
    },
    "tmp2m": {
      "full_name": "/tmp2m",
```


(continues on next page)

(continued from previous page)


```
      "datatype": "Float32",
      "dimensions": [
        "/time",
        "/grid_yt",
        "/grid_xt"
      ],
      "dimension_size": [
        2,
        1536,
        3072
      ],
      "block_size": [
        1,
        768,
        1536
      ],
      "attributes": {
        "cell_methods": "time: point",
        "long_name": "2m temperature",
        "missing": 9.99e+20,
        "output_file": "sfc"
      },
      "unit": "K"
    }
  }
}
```

CHAPTER

TWELVE

SURVEY

Please take a couple minutes answering this survey to help improving the workshop for future editions.

CHAPTER

THIRTEEN

EXERCISE SOLUTION FOR RASTER INFO

```
$ gdal raster info SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_
↪20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634 --format=json  | jq ".stac[\
↪"proj:epsg\"]"
```

Output:

```
32634
```

CHAPTER

FOURTEEN

EXERCISE SOLUTION FOR VECTOR INFO

```
$ gdal vector info timisoara.osm.pbf --sql "SELECT *, ST_Distance(geometry, ST_Point(21.
↪2322, 45.7558), true) distance_from_center FROM points WHERE other_tags LIKE '%bar%'␣
↪ORDER BY distance_from_center LIMIT 5" --dialect sqlite --features
```

Output:

```
INFO: Open of `timisoara.osm.pbf'
      using driver `OSM' successful.

Layer name: SELECT
Geometry: Point
Feature Count: 5
Extent: (21.228195, 45.755208) - (21.234416, 45.756671)
Layer Coordinate Reference System:
  - name: WGS 84
  - ID: EPSG:4326
  - type: Geographic 2D
Data axis to CRS axis mapping: 2,1
Geometry Column = GEOMETRY
osm_id: String (0.0)
name: String (0.0)
barrier: String (0.0)
highway: String (0.0)
ref: String (0.0)
address: String (0.0)
is_in: String (0.0)
place: String (0.0)
man_made: String (0.0)
other_tags: String (0.0)
distance_from_center: Real (0.0)
OGRFeature(SELECT):0
  osm_id (String) = 4214502952
  name (String) = MGM Bastion
  barrier (String) = (null)
  highway (String) = (null)
  ref (String) = (null)
  address (String) = (null)
  is_in (String) = (null)
  place (String) = (null)
  man_made (String) = (null)
```

(continues on next page)

(continued from previous page)

```
  other_tags (String) = "amenity"=>"bar","cuisine"=>"breakfast;burger;chicken;german;
↪grill;hot_dog;local","internet_access"=>"wlan","opening_hours"=>"24/7"
  distance_from_center (Real) = 90.4872767351965
  POINT (21.2326735 45.7565436)

OGRFeature(SELECT):1
  osm_id (String) = 7976845620
  name (String) = Barrio
  barrier (String) = (null)
  highway (String) = (null)
  ref (String) = (null)
  address (String) = (null)
  is_in (String) = (null)
  place (String) = (null)
  man_made (String) = (null)
  other_tags (String) = "addr:city"=>"Timișoara","addr:housenumber"=>"6","addr:street"=>
↪"Strada Eugeniu de Savoya","amenity"=>"bar"
  distance_from_center (Real) = 167.767604090299
  POINT (21.2304389 45.756671)

OGRFeature(SELECT):2
  osm_id (String) = 6930124470
  name (String) = The Post
  barrier (String) = (null)
  highway (String) = (null)
  ref (String) = (null)
  address (String) = (null)
  is_in (String) = (null)
  place (String) = (null)
  man_made (String) = (null)
  other_tags (String) = "amenity"=>"bar"
  distance_from_center (Real) = 184.50111354138
  POINT (21.2344156 45.7552083)

OGRFeature(SELECT):3
  osm_id (String) = 3755520724
  name (String) = Enoteca de Savoya
  barrier (String) = (null)
  highway (String) = (null)
  ref (String) = (null)
  address (String) = (null)
  is_in (String) = (null)
  place (String) = (null)
  man_made (String) = (null)
  other_tags (String) = "addr:city"=>"Timișoara","addr:housenumber"=>"11","addr:postcode
↪"=>"300085","addr:street"=>"Strada Eugeniu de Savoya","amenity"=>"bar","phone"=>"+40␣
↪256 433 644","website"=>"https://www.enotecadesavoya.ro/"
  distance_from_center (Real) = 315.856848810125
  POINT (21.2283292 45.7566568)

OGRFeature(SELECT):4
  osm_id (String) = 5224522189
```

(continues on next page)

(continued from previous page)

```
  name (String) = Storia
  barrier (String) = (null)
  highway (String) = (null)
  ref (String) = (null)
  address (String) = (null)
  is_in (String) = (null)
  place (String) = (null)
  man_made (String) = (null)
  other_tags (String) = "amenity"=>"bar","opening_hours"=>"Th-Sa 18:00-04:00; Tu 18:00-
↪02:00"
  distance_from_center (Real) = 326.302012684119
  POINT (21.2281946 45.7566704)
```

CHAPTER

FIFTEEN

EXERCISE SOLUTION FOR RASTER FORMAT CONVERSION

1. Selection of red, green, blue bands:

```
$ gdal raster select TDR.tif TDR_rgb.tif --band 1,2,3 --creation-option TILED=YES
```

or

```
$ gdal raster select TDR.tif TDR_rgb.tif --band red,green,blue --creation-option␣
↪TILED=YES
```

2. Scaling to 8-bit with JPEG compressed tiled GeoTIFF output

```
$ gdal raster scale TDR_rgb.tif TDR_rgb_byte_jpeg.tif \
    --output-data-type uint8  \
    --creation-option COMPRESS=JPEG \
    --creation-option TILED=YES
```

3. Improving the visual result by clamping the input range to 2 standard deviations of the mean.

   First let's collect statistics:

   ```
   $ gdal raster info --stats TDR_rgb.tif
   ```

   ```
   [ ... snip ... ]
   Band 1 Block=256x256 Type=UInt16, ColorInterp=Red
      Minimum=0.000, Maximum=17143.000, Mean=1439.120, StdDev=582.568
   [ ... snip ... ]
   Band 2 Block=256x256 Type=UInt16, ColorInterp=Green
      Minimum=0.000, Maximum=17660.000, Mean=1469.245, StdDev=543.003
   [ ... snip ... ]
   Band 3 Block=256x256 Type=UInt16, ColorInterp=Blue
      Minimum=0.000, Maximum=18510.000, Mean=1324.608, StdDev=482.921
   ```

   So roughly all bands have a mean around 1400 and a standard deviation of 500, so using [1400 - 2 * 500, 1400 + 2 * 500] should roughly remove the 5% outliers.

   ```
   $ gdal raster scale TDR_rgb.tif TDR_rgb_byte_jpeg_clamped.tif \
       --input-min 400 \
       --input-max 2400 \
       --output-data-type uint8  \
       --creation-option COMPRESS=JPEG \
       --creation-option TILED=YES
   ```

Note

We have not yet marked the nodata value as being 0, so above statistics are not fully correct. We'll do that later.

CHAPTER

SIXTEEN

EXERCISE SOLUTION FOR VECTOR FORMAT CONVERSION

1. Convert to GeoJSON

```
$ gdal vector convert timisoara.osm.pbf out_points.geojson --input-layer␣
↪points
$ gdal vector convert timisoara.osm.pbf out_lines.geojson --input-layer␣
↪lines
$ gdal vector convert timisoara.osm.pbf out_multilinestrings.geojson --
↪input-layer multilinestrings
$ gdal vector convert timisoara.osm.pbf out_multipolygons.geojson --input-
↪layer multipolygons
$ gdal vector convert timisoara.osm.pbf out_other_relations.geojson --input-
↪layer other_relations
```

2. "Exploding" collections in single primitives (points, lines, polygons).

```
$ gdal vector explode-collections timisoara.osm.pbf timisoara.gpkg --
↪overwrite
```

3. Create a layer for each geometry type

```
$ gdal vector filter timisoara.gpkg other_relations_points.gpkg \
    --input-layer other_relations  --where "ST_GeometryType(geom) = 'POINT'
↪" --overwrite

$ gdal vector filter timisoara.gpkg other_relations_lines.gpkg \
    --input-layer other_relations  --where "ST_GeometryType(geom) =
↪'LINESTRING'" --overwrite

$ gdal vector filter timisoara.gpkg other_relations_polygons.gpkg \
    --input-layer other_relations  --where "ST_GeometryType(geom) = 'POLYGON
↪'" --overwrite
```

CHAPTER

SEVENTEEN

EXERCISE SOLUTION FOR CLIPPING

```
gdal vector sql '{"type":"Point","coordinates":[21.2322,45.7558]}' \
    clipping_circle.gpkg --sql "select st_buffer(st_transform(geometry,32634), 1000)␣
↪from OGRGeoJSON" \
    --dialect sqlite --overwrite

 gdal raster clip \
     SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TER_20260423T115714.SAFE/MTD_
↪MSIL2A.xml:10m:EPSG_32634 \
     S2_TER_circle.tif \
     --like clipping_circle.gpkg
```

CHAPTER

EIGHTEEN

EXERCISE SOLUTION FOR RASTER REPROJECTION

```
$ gdal raster reproject TDR_rgb_byte_clamped.gdalg.json TDR_3035_with_ovr.tif --input-
↪nodata 0 --add-alpha --creation-option TILED=YES --creation-option COMPRESS=JPEG --
↪output-crs EPSG:3035 -r cubic

$ gdal raster overview add -r cubic  TDR_3035_with_ovr.tif
```

or:

```
$ gdal raster reproject TDR_rgb_byte_clamped.gdalg.json TDR_3035_cog.tif --input-nodata␣
↪0 --add-alpha --format COG --creation-option COMPRESS=JPEG --output-crs EPSG:3035 -r␣
↪cubic
```

CHAPTER

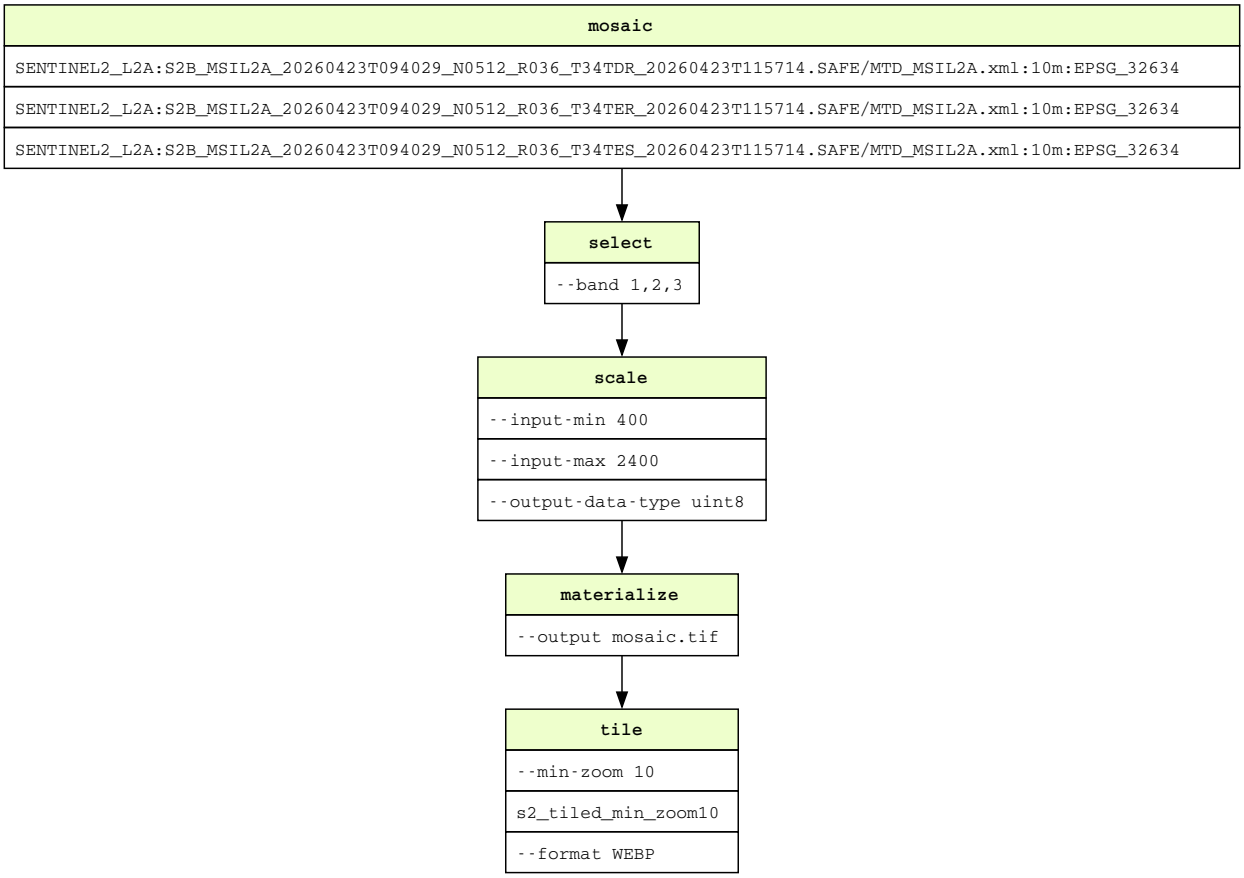
NINETEEN

EXERCISE SOLUTION FOR MATERIALIZING PIPELINE INTERMEDIATE RESULT

```
$ gdal raster pipeline \
        mosaic SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_
↪20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634 \
        SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TER_20260423T115714.SAFE/
↪MTD_MSIL2A.xml:10m:EPSG_32634 \
        SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TES_20260423T115714.SAFE/
↪MTD_MSIL2A.xml:10m:EPSG_32634 \
        ! \
        select --band 1,2,3 \
        ! \
        scale --input-min 400 \
              --input-max 2400 \
              --output-data-type uint8 \
        ! \
        materialize --output=mosaic.tif \
        ! \
        tile --min-zoom 10 s2_tiled_min_zoom10 --format WEBP
```

Note

Original idea was to use COG output instead of regular GeoTIFF, but fails currently because of https://github.com/OSGeo/gdal/issues/14730


mosaic
SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TDR_20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634
SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TER_20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634
SENTINEL2_L2A:S2B_MSIL2A_20260423T094029_N0512_R036_T34TES_20260423T115714.SAFE/MTD_MSIL2A.xml:10m:EPSG_32634
select
--band 1,2,3
scale
--input-min 400
--input-max 2400
--output-data-type uint8
materialize
--output mosaic.tif
tile
--min-zoom 10
s2_tiled_min_zoom10
--format WEBP

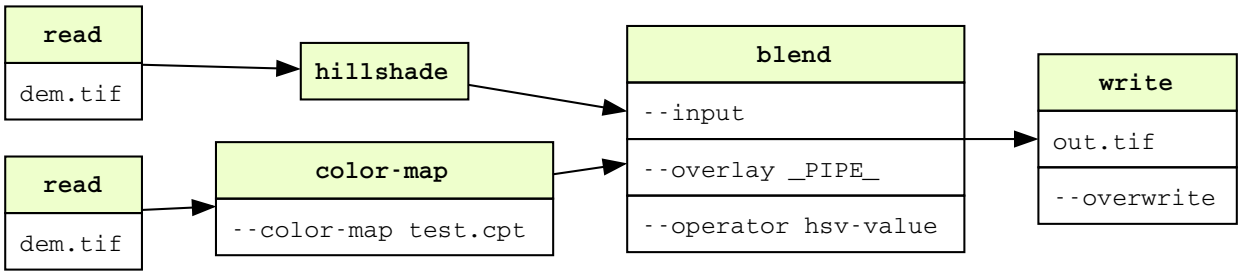
CHAPTER

TWENTY

EXERCISE SOLUTION FOR DEM PIPELINE

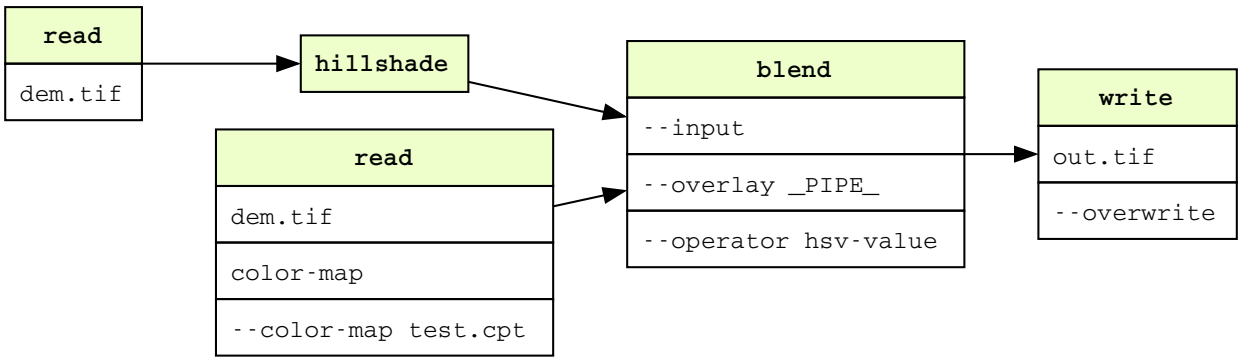
1. Modified pipeline to do hillshaded before hypsometric rendering

```
$ gdal raster pipeline \
    read dem.tif ! \
    hillshade ! \
    blend --input [ read dem.tif ! color-map --color-map test.cpt ] --overlay _PIPE_ --
↪operator hsv-value ! \
    write out.tif --overwrite
```



2. Generate colorized and hillshaded maps as intermediate results

```
$ gdal raster pipeline \
    read dem.tif ! \
    color-map --color-map test.cpt ! \
    tee [ write dem_colorized_tee.tif ] ! \
    blend [ read dem.tif ! hillshade ! tee [ write dem_hillshade.tif --overwrite ] ] --
↪operator hsv-value ! \
    write out.tif --overwrite
```



CHAPTER

TWENTYONE

EXERCISE SOLUTION FOR `GDAL RASTER CALC`

1.

```
$ gdal raster calc --input s2_TER_10m.xml --output ndvi.tif --calc "(X[4] - X[1]) /␣
↪(X[4] + X[1])"
```

2.

```
$ gdal raster select s2_TER_10m.xml red.gdalg.json --band red
$ gdal raster select s2_TER_10m.xml NIR.gdalg.json --band NIR
$ gdal raster calc --input red=red.gdalg.json --input NIR=NIR.gdalg.json --output ndvi.
↪gdalg.json  --calc "(NIR - red) / (NIR + red)"
```

3.

```
$ gdal pipeline calc --input s2_TER_10m.xml --calc "(X[4] - X[1]) / (X[4] + X[1])" !␣
↪write ndvi_generic.gdalg.json

$ gdal pipeline ndvi_generic.gdalg.json  --calc.input=s2_TES_10m.xml --write.output=NDVI_
↪TES.tif --format COG
```

CHAPTER

TWENTYTWO

EXERCISE SOLUTION FOR GDAL RASTER ZONAL-STATS

Using intermediate admin_1_around_timis.gpkg

```
$ gdal pipeline read  dem.tif ! \
    zonal-stats --stat max,max_center_x,max_center_y \
                --zones admin_1_around_timis.gpkg  \
                --include-field admin,name ! \
    make-point --x max_center_x --y max_center_y --output-crs EPSG:4326 ! \
    write dem_point_max_elev.gpkg
```

Starting from ne_10m_admin_1_states_provinces.zip

```
$ gdal pipeline read  dem.tif ! \
    zonal-stats --stat max,max_center_x,max_center_y \
                --zones [ read /vsizip/ne_10m_admin_1_states_provinces.zip ! \
                          filter --bbox=19.6854167,45.0565278,22.4426389,46.9537500 ]  \
                --include-field admin,name ! \
    make-point --x max_center_x --y max_center_y --output-crs EPSG:4326 ! \
    write dem_point_max_elev.gpkg --overwrite
```