Dortmund Scaling - Midpoint Layer Derivation

This notebook identifies the source data for NYC scaling and explains pre-processing to output the four basic layers on which scenarios are built:

1. Simplified Land Cover
2. Simplified Land Use
3. Slope
4. Sunlight availability

To replicate this work, first open a blank project in QGIS. I have tested most of this on QGIS 3.18 and 3.24, though anything past 3.00 should suffice. When possible, I will include both a verbal explanation and a screenshot for the procedure described - this should make it easier to replicate the process in varying versions where the syntax or appearance may change slightly.

The goal of this notebook is to translate the input layers into the four basic layers described above. The input layers come from several open databases furnished by German municipal authorities.

Although we sometimes share more specific links in this procedure, it is worth pointing out that many of these datasets are available from the [NRW OpenData portal](https://www.geoportal.nrw/?activetab=map&openDownloadclient=true), as well as many other useful datasets. At the end of this analysis, we will have tranformed these inputs into four aligned rasters at 1m resolution. All functions will be conducted in EPSG 102003 because it works well with solar irradiance actions and is a relatively painless transition from the NYC projections used by the city. Based on those rasters, we will be able to develop a variety of scenarios that capture the possible areas of expansion for urban agriculture in NYC.

# Land Use and Ownership Layer Derivation This layer will describe the land use at a parcel level. These will be used primarily to sort the different types of gardens - e.g., it makes much more sense to assume an individual garden in a single-family backyard than a community garden. The final layer produced via this procedure will reflect the land use codebook available [here](https://docs.google.com/spreadsheets/d/176HUxNceFgc3QG77QbuSUSxrKWQ1HY6M?rtpof=true&authuser=jkhawes%40umich.edu&usp=drive_fs).

Overview: We can download Land Use data (“Natzung”) data that just need to be simplified and prepared for use in our coding scheme.

## 

## Download and Translate Natzung layer

The Land Use data for a variety of municipalities can be downloaded from here: <https://www.opengeodata.nrw.de/produkte/geobasis/lk/akt/gru_xml/> - Dortmund is of course just labeled as Dortmund. Additional details on data availability, including land use but also including other data, can be found here: <https://open.nrw/dataset/407373a2-422c-469c-a7e9-06a62b4d7d9a>.

The data are downloaded as an NAS file and can be easily translated to any format you choose. We’ll turn it into a raster soon, so unless you strongly object to using the NAS file, you could just use it for the steps below.

Before doing anything else, we have to render the data legible for an English-speaking user. If we want to literally translate things, we can do it with Field Calculator, or we can just cross-reference the German names for land uses with what Google Translate spit out. I pulled all the unique entries in the Natzung field out and included them with their translations below - it may be worth doing this again if you are looking at this in 2024 or later, since they may update the names periodically (and Google Translate has likely gotten better):

| Bahnverkehr | Railway traffic |
| --- | --- |
| Flaeche Besonderer Funktionaler Praegung | Area of special functional character |
| Flaeche Gemischter Nutzung | Area of mixed use |
| Fliessgewaesser | Watercourses, running water |
| Flugverkehr | Air traffic |
| Friedhof | Cemetery |
| Gehoelz | Woody, coppice, undergrowth |
| Hafenbecken | Harbour basin |
| Halde | Heap, slag heap, landfill |
| Industrie Und Gewerbeflaeche | Industry and commercial area |
| Landwirtschaft | Agriculture |
| Moor | Moor or bog |
| Platz | Square |
| Schiffsverkehr | Shipping traffic |
| Sport Freizeit Und Erholungsflaeche | Sport, leisure and recreation area |
| Stehendes Gewaesser | Stalling water, stagnant water |
| Strassenverkehr | Road traffic |
| Sumpf | Swamp |
| Tagebau Grube Steinbruch | Open pit quarry, mine |
| Unland Vegetationslose Flaeche | Unland vegetation-free area, Unland Area without vegetation |
| Wald | Forest |
| Weg | Path, way, away |
| Wohnbauflaeche | Housing area, residential area |

With this in hand, we can convert this into our universal numeric system (reproduced in the [codebook](https://docs.google.com/spreadsheets/d/176HUxNceFgc3QG77QbuSUSxrKWQ1HY6M?rtpof=true&authuser=jkhawes%40umich.edu&usp=drive_fs) for ease of access):

if( "nutzart" = 'Bahnverkehr', 82,

if( "nutzart" = 'Flaeche Besonderer Funktionaler Praegung' , 32,

if( "nutzart" = 'Flaeche Gemischter Nutzung' , 21,

if( "nutzart" = 'Fliessgewaesser' , 83,

if( "nutzart" = 'Flugverkehr' , 42,

if( "nutzart" = 'Friedhof' , 32,

if( "nutzart" = 'Gehoelz' , 33,

if( "nutzart" = 'Hafenbecken' , 83,

if( "nutzart" = 'Halde' , 43,

if( "nutzart" = 'Industrie Und Gewerbeflaeche' , 23,

if( "nutzart" = 'Landwirtschaft' , 90,

if( "nutzart" = 'Moor' , 33,

if( "nutzart" = 'Platz' , 32,

if( "nutzart" = 'Schiffsverkehr' , 42,

if( "nutzart" = 'Sport Freizeit Und Erholungsflaeche' , 26,

if( "nutzart" = 'Stehendes Gewaesser' , 83,

if( "nutzart" = 'Strassenverkehr' , 80,

if( "nutzart" = 'Sumpf' , 32,

if( "nutzart" = 'Tagebau Grube Steinbruch' , 23,

if( "nutzart" = 'Unland Vegetationslose Flaeche' , 70,

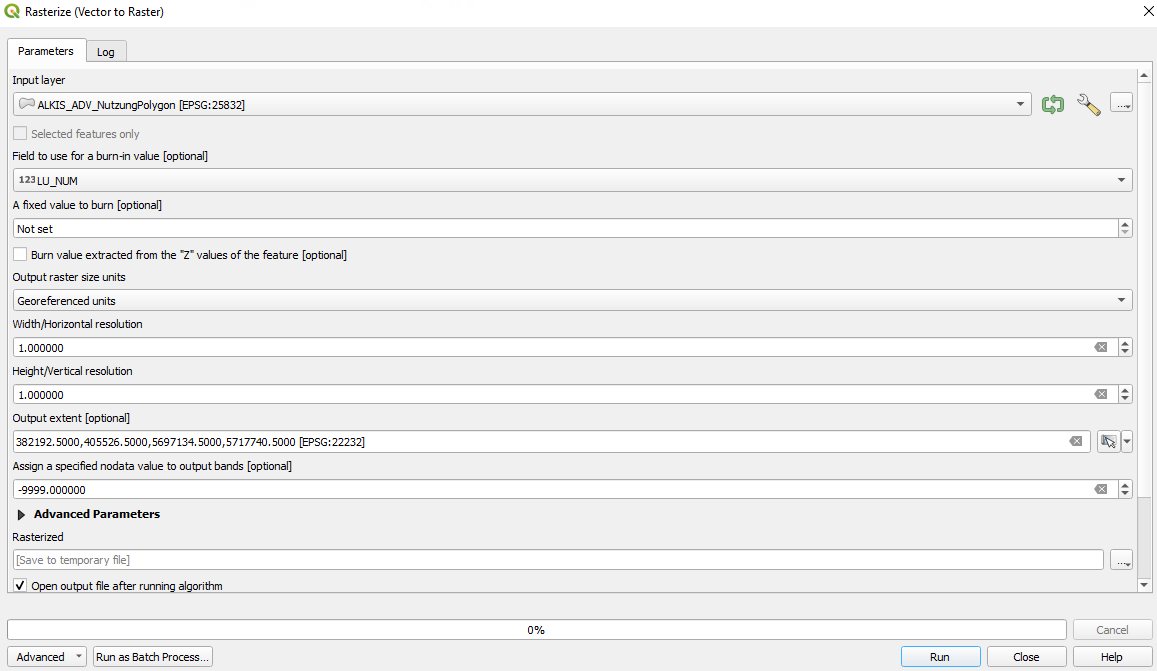
if( "nutzart" = 'Wald' , 33,

if( "nutzart" = 'Weg' , 81,

if( "nutzart" = 'Wohnbauflaeche' , 10, 0)))))))))))))))))))))))

## Rasterize

Once we have the layer numeric and in the correct projection (using 25832 for now, so we don’t need to reproject), we can move on to rasterizing it. If you want to make your life easier, complete the first couple steps of the [binary slope layer derivation](#_a4js6pqfyk55) before doing this so you can use the merged DTM as the output extent - this will save you from aligning things later. If LU\_NUM doesn’t show up as a field, it’s because the change hasn’t saved to the file yet. Make sure to save the change and stop editing the original land use shapefile. If this still doesn’t work, remove the shapefile and reload it. Sometimes the field will show up even if you haven’t saved it. In this case, it will pitch an error early on that will remind you to save the file. Overall, this should only take a few minutes.



Once this is complete, You may need to run r.null or a clip to the Dortmund extent to get the no-data set up. You can create a layer to clip with by dissolving the original land use vector into one shape and then clipping with that shape.

# Binary Slope Layer Derivation

This layer will describe the slope across the city. The final layer produced via this procedure will have the following codes:

* Ineligible, over 15% grade-- 0
* Flat ground – 1
* Flat roof – 2

Overview: We can use the LiDAR-derived DTM and DSM to identify flat ground and flat roofs throughout the city. We have to do these two things separately, since the DSM captures trees as well as buildings.

We will use the following data sets:

1. LiDAR - Bildbasiertes Digitales Oberflächenmodell 50 (LAS) - Paketierung: Einzelkacheln
2. DTM - Digitales Geländemodell - Gitterweite 1 m (XYZ) - Paketierung: Einzelkacheln

We will begin by preparing city-wide DTM and DSM layers, then we will proceed with the flat ground analysis. We will then identify flat roofs. Finally, we will mask buildings from the ground layer and add in the buildings results. This takes a fair amount of time, since we have to derive the DSM from LiDAR.

## Download data

Downloading the digital surface and terrain data is actually pretty straightforward. The first step is to take the Land Use data and use the dissolve tool to create one large shape for all of Dortmund. Then you go to [this website](https://www.opengeodata.nrw.de/produkte/geobasis/hm/) to learn about the terrain datasets. From that website, there is a link to the open data download client for Nordrhein-Westfalen (NRW, the state). If you proceed to that site, the download portal should show up as a hovering box over the basemap. You can actually also use this site to explore a variety of other data, but for now we just need this download client. Zip the shapefile you made of the Dortmund boundary and upload that to the website with the tool (step numbered 1). Then you can proceed to Select products, the step numbered 2. Chrome actually plays nice with this website and translates the file names fairly well for me, but it you don’t want to do that or if yours won’t translate, here they are in German:

* Bildbasiertes Digitales Oberflächenmodell 50 (LAS) - Paketierung: Einzelkacheln
* Digitales Geländemodell - Gitterweite 1 m (XYZ) - Paketierung: Einzelkacheln

On first import, the xyz.gz files for the DTM don’t seem to have a projection. I believe the projection for all the German govt data is going to be 25832 ETRS89 / UTM zone 32N.

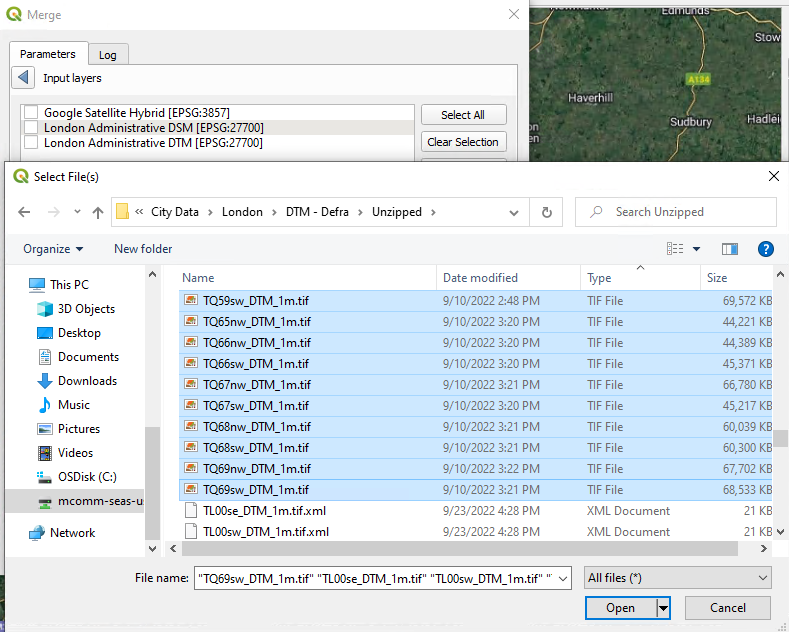
## Use lasTools to build a DSM

The DSM is provided as las files. It is necessary to convert these to a DSM using lasTools. Pretty much the [same procedure](https://docs.google.com/document/d/1bWgMKp40i6dT949iprVoX0UrYW69Ixmz6f5D3drVk5s/edit#heading=h.tpnbb4r3uqme) as Detroit should do the trick. The major difference is that the files are somewhat smaller, so I was able to run the whole city in one go. Should only take a few hours even if you’re multi-tasking - each command ran in about 10 minutes for me. It’s also worth noting that there’s no reason to use the feet and vertical\_feet command, since things are measured in meters here. Oddly enough, it doesn’t seem to make a serious difference - I tried both ways and 99% of cells end up within 1m, though I will say a visual inspection seems to suggest that the version without the feet command is cleaner.

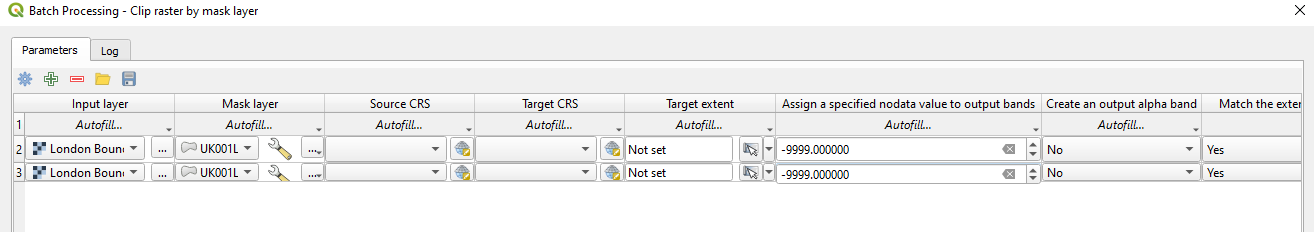
## Merge DSM and DTM files and clip to administrative boundary

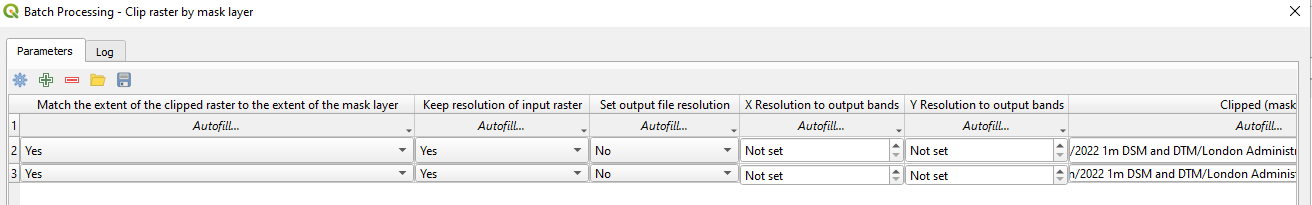
Since these files are downloaded as individual tiles (and the DSM is computed in tiles), they will need to be merged to be useful in city-wide analysis. Having them all in one folder like shown above is particularly helpful for this stage, since the merge tool can pull an entire directory. Under Raster >> Miscellaneous, the Merge tool will take a bunch of input layers and create on merged raster layer. You do not have to load the files into QGIS to merge them. Since they’re all in one folder, you can go in to the Merge tool, click Add Layer, then click Add Files. Go to the directory where you unzipped everything to and sort by file type so you don’t have to worry about the tfw files. Take all the tif files, select them, and add them to the merge. Click “OK” otherwise the list of inputs does not always save. Save the file as something like “Dortmund Bounding Box DTM” so you know this is the whole merged file before being clipped. Run the program.

\*\*Ignore the occasional screenshots from NYC and London - the commands are the same - just different file names



You’ll need to do this with the DTM and DSM separately. Once they’re ready, you can clip those files to the administrative boundary to save some computational time - batch processing works for this. Using the Dortmund Boundary file found [here](https://data.london.gov.uk/dataset/statistical-gis-boundary-files-london), you can use the Clip Raster by Mask Layer tool and make everything outside the administrative boundary null.



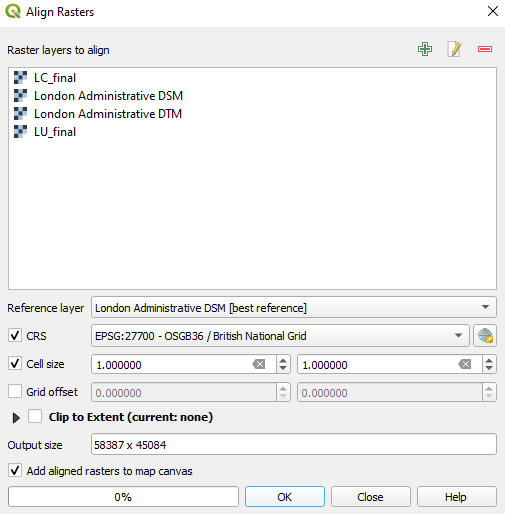


The output should look something like this:

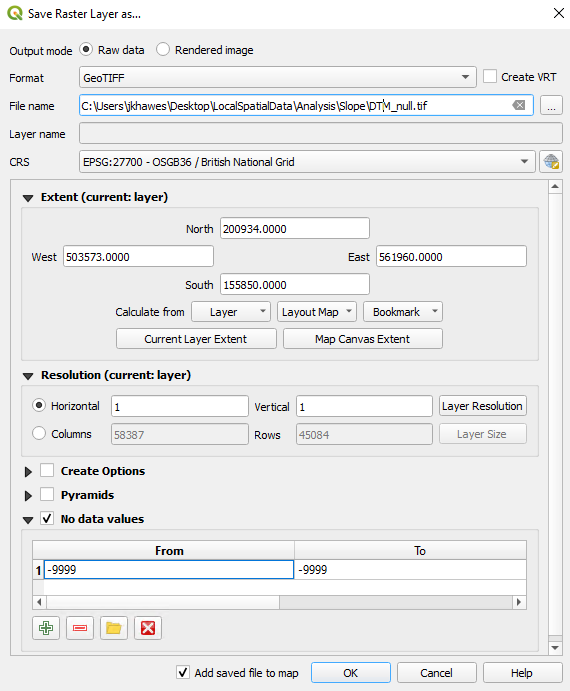


## Align with Land Use final layers

Next we need to make sure our slope layers align with the layers we created previously. Since the land use and land cover are already aligned, we don’t strictly need to align them. But the DSM is actually the best alignment reference - probably because it started out as a raster - so it works best just to run all current layers and get our aligned layers at this point.



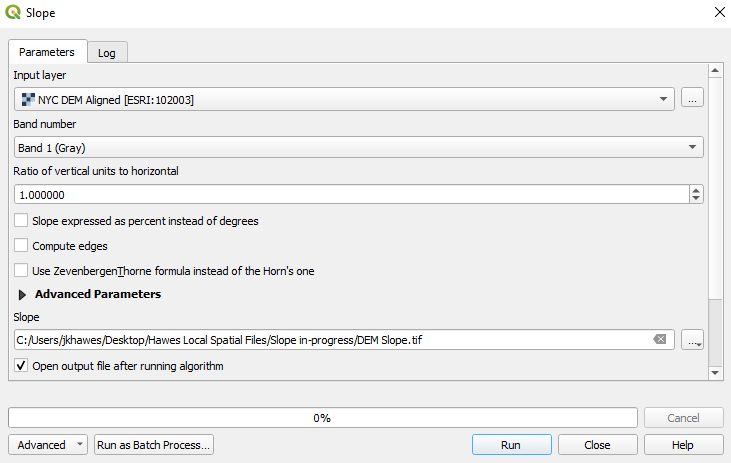
After aligning the files, you will likely have to remind the computer that -9999 is a null value and not an actual value (this would obviously throw off the slope pretty dramatically at the city boundary). At first glance, this is actually kind of a pain because the files are too large for GRASS and you can’t use the BIGTIFF option in QGIS, so we can’t use r.null. HOWEVER, the good news is that there is a way to accomplish this with a built-in QGIS function. Just right-click on the layer, then export it. We’ll use mostly default settings, but we’ll check the box for No data values at the bottom. When we do this, it allows us to input -9999 as a no data value, and the system keeps track of it from here on. Don’t forget to run this for both the DTM and DSM.



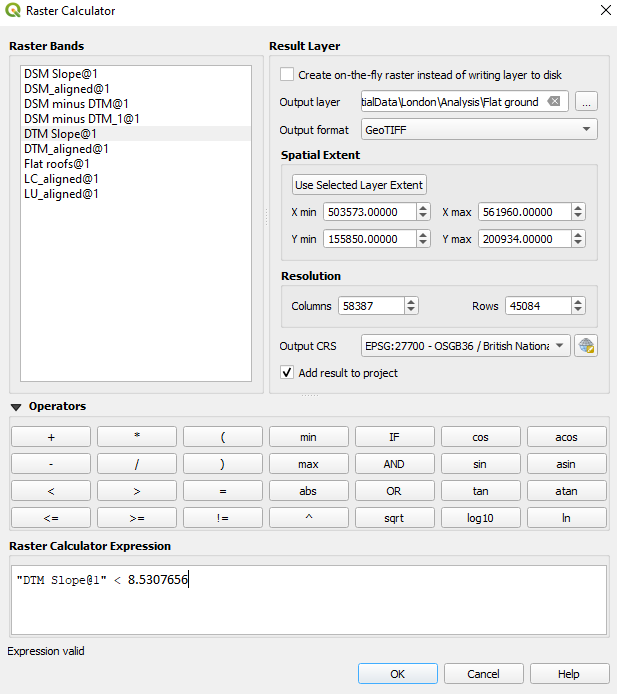
## Calculate Slope from the DTM

Now that we have clean layers to work with, we can derive slope on the ground. We can simply use the ***Slope*** function under Raster > Analysis on the DTM we just aligned. This will yield a slope layer for all ground cells in the city. We’re interested in places where the slope is less than 15%. Unfortunately, the “Slope expressed as percent instead of degrees” function seems to return absolutely outrageous values, so I don’t recommend using that. Instead, it seems better just to convert the 15% to degrees and use that in the raster calculator in the next step.

If you want to save yourself time later, run this as a batch process with the DSM as well.

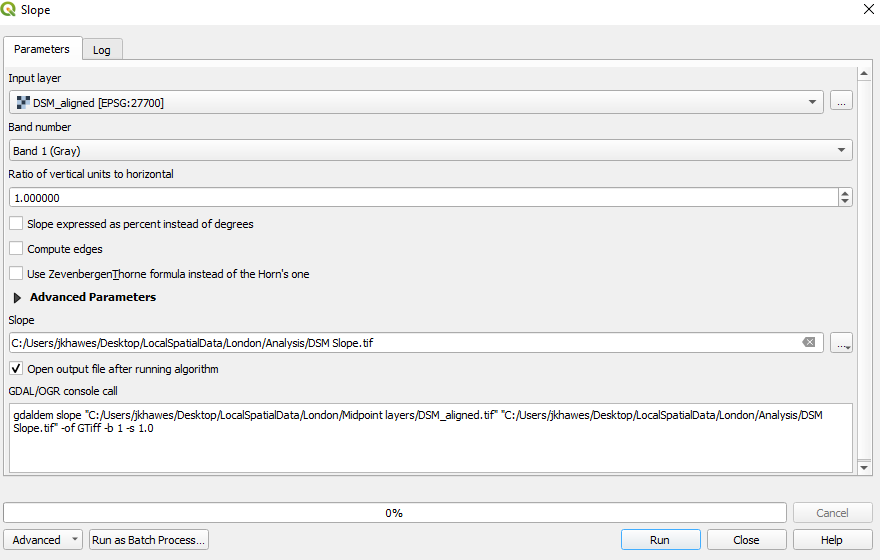


Once we have this Slope file, we can do some simple raster algebra to determine where this is greater than and less than 15%. Expressed as degrees, a 15% slope is arc-tangent of 0.15, which is 8.5307656. So we want to find places where the slope layer is less than that.

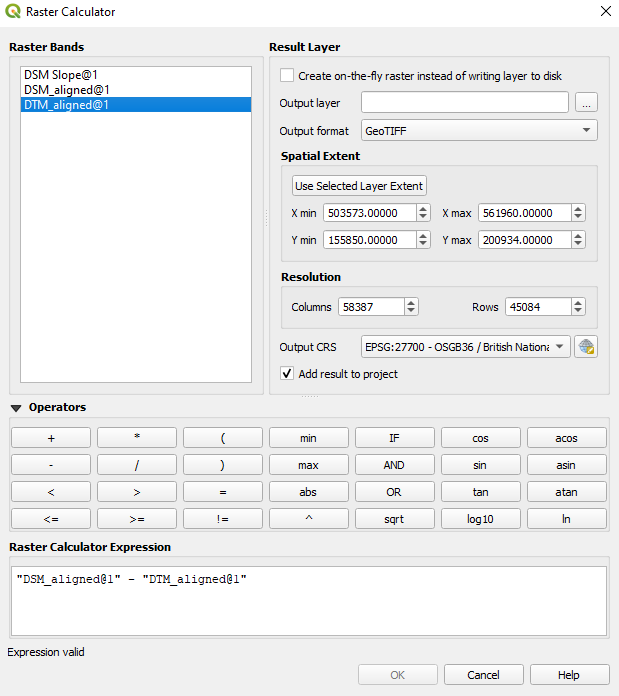


## Identify flat roofs from the DSM

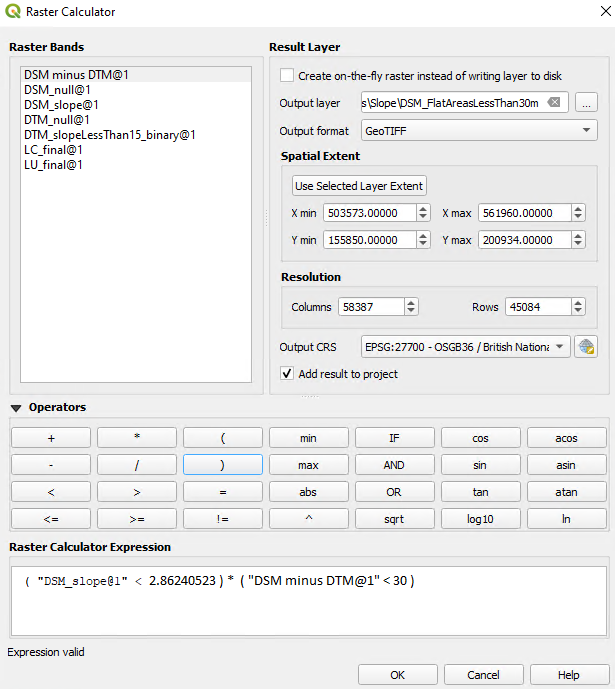
Next, we need to calculate the roof slopes - this is a bit trickier and requires several steps. First, we can run the Slope function under Raster > Analysis on the DSM.



Next, when we need to subtract the DEM from the DSM to make sure the ground level is zero all over the map.

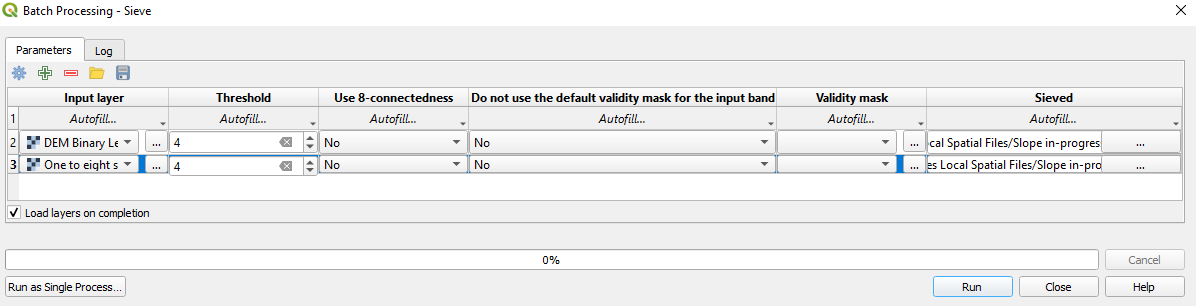


We can then run some simple raster calculations, finding the places where the Slope layer is less than 5% (less than arctangent of 0.05 = 2.86240523) and the Height of the DSM-DEM layer is less than 30m (8 stories or less) - note that this height is why we need to do the subtraction. If not, we can’t use 30 or something as a roof height, because ground level differs, so some roofs are below ground level in other places in the city. In some older versions of this, I also used 3.5 m as a minimum, but this really isn’t necessary since we use the building land cover classification below. Just need to limit the max so we’re not working on top of the Empire State building.



## Clean up layers with the sieve tool

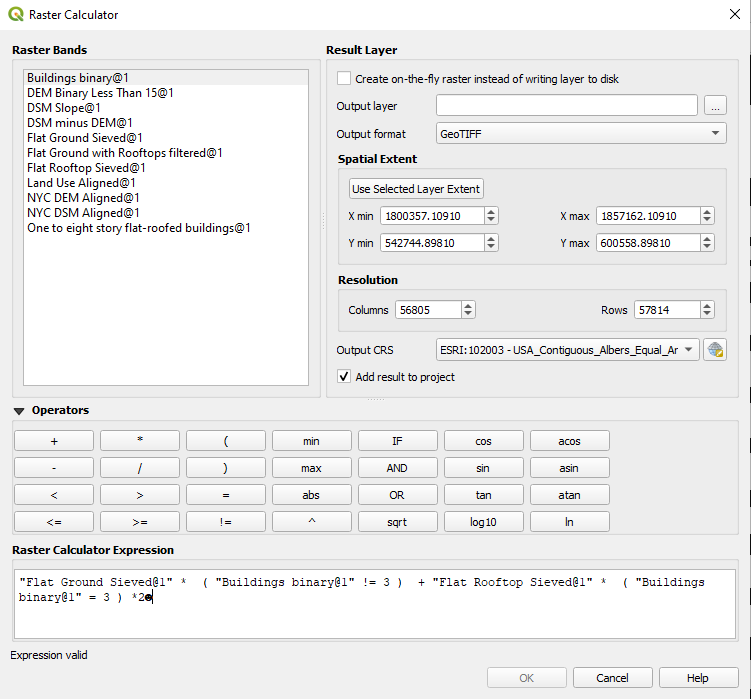
Once we have useful results for both ground and rooftops, we can use the sieve tool to clean it up a bit. I used 4 as a threshold, since anything smaller than this wouldn’t simultaneously hold a raised bed and allow for management. We’ll also re-run this sieve with the final classification, but this is a good clean-up middle step. Make sure to fix the symbology to 0-1 after running the function.



## Final raster calculations

Lastly, we need to combine these layers into something intelligible - we will keep flat roofs and flat ground separate for now just for the sake of preserving information. We can always reclassify later. So our goal is: 0 = > 15% slope, 1 = flat ground, 2 = flat roof. How do we get there?

First, we’ll need to make all the building footprints zero in the ground slope layer and convert everything outside of buildings to zero in the roof slope layer. To keep the information discrete,, we can do one more raster calculation - RoofsLayer \* 2 + GroundLayer. In the end, we can do this all in one step. For the building layer conversion, we can import the building layer used in the previous steps. Since everything has been aligned, we should be able to use it directly. The raster calculation is exactly the same for the two layers, but inverted. See below.



After this calculation is complete, don’t move all the Slope files off the hard drive just yet - make sure to keep the aligned DSM - we will use it as our first input.

# Land Cover Layer Derivation

This layer will identify open ground areas and rooftops. These open areas and rooftops will then be filtered by other qualifications from the other layers (e.g. slope). The final layer produced via this procedure will have the following codes:

* Impervious - 1
* Low vegetation, grass or dirt - 2
* Roof - 3
* Trees - 4
* Otherwise occupied - 0 (e.g., monument, water, railroad, road)

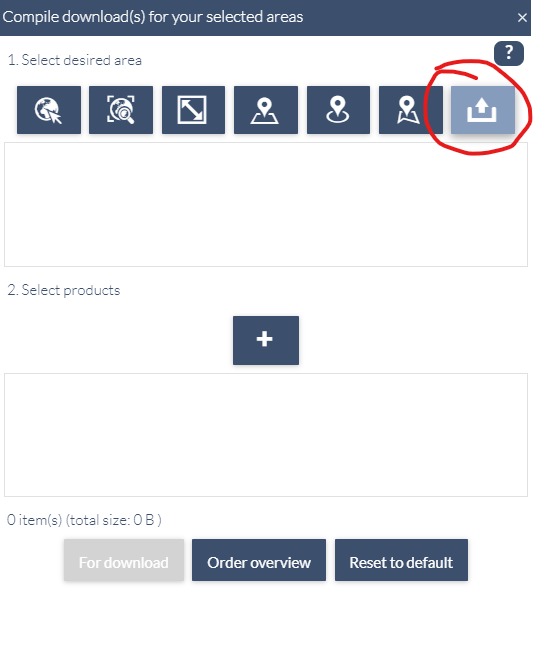
Overview: We do not have access to a land cover dataset in Dortmund, so we use a variety of available data and conduct machine learning to derive the land cover data. While most portions of the methods used in this paper are fairly straightforward transformations or format conversions, this section is really just a guide for the user to understand the basics of the machine learning method employed. The suggestions here are not certain, and they should be treated as an overall framework, not necessarily a step-by-step process for land cover derivation. Users are encouraged to refine and improve this process.

Note: You can skip this machine learning process if you prefer to simply use the land cover dataset we derived in June 2023: link.

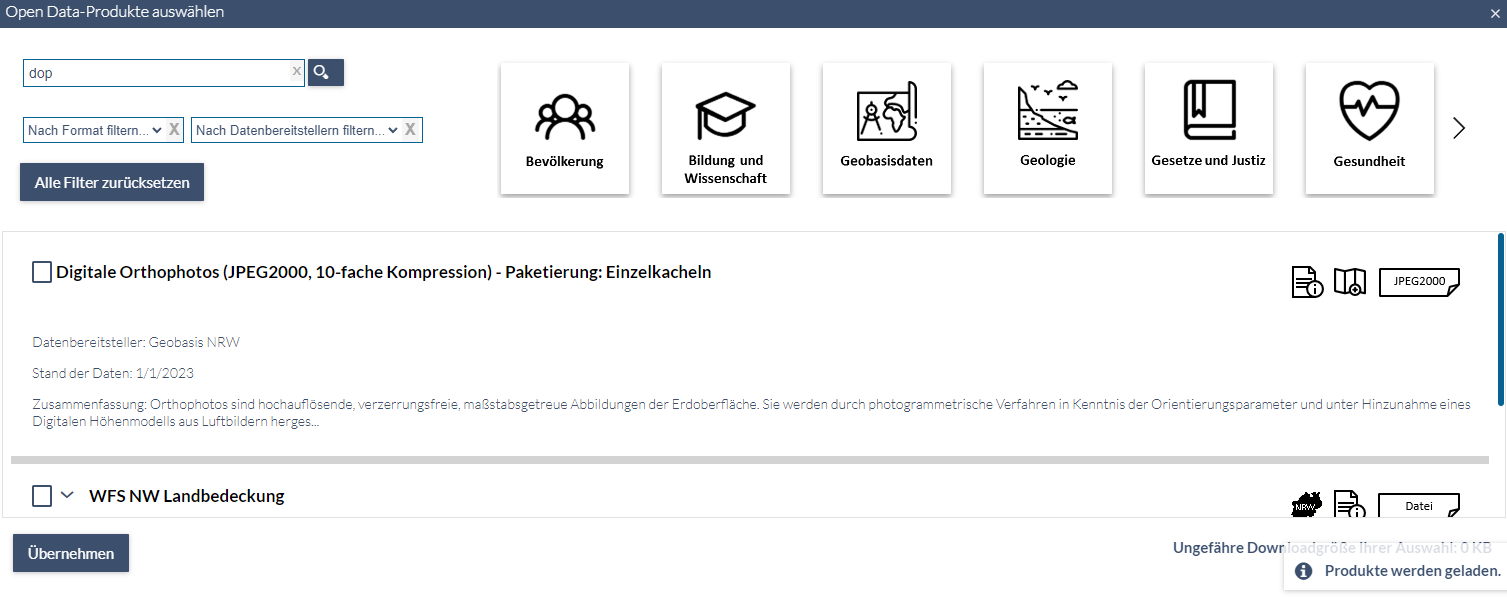
## Download and prepare orthoimagery for machine learning

Since there is no high-resolution land cover data available in Dortmund, we need to build our own dataset. The good news is that there is a nice set of orthoimagery available in the [NRW geoportal](https://www.geoportal.nrw/?activetab=map#/datasets/iso/4c8955f8-7c72-498e-bb3a-55b1680a4ec6). There are actually two geoportals, and I have an older version of these instructions saved [here](#_a6m30adrcmd6) that catalogs a more manually intensive way of getting these same data. Once you proceed to the Geoportal, you can use the Translate option in Google Chrome to actually render most of it in English. Click download in the top left. A menu should appear on the right.

For Step 1, I recommend uploading a zip file of the Dortmund boundary. In an earlier iteration of this, I used the [Land Use data](https://geoweb1.digistadtdo.de/doris_gdi/opengeodata/alkis/Nutzung.zip) (see below for more detail) to generate a boundary file for Dortmund with the dissolve function, but a more reliable solution would be to get administrative boundary data from [here](https://www.opengeodata.nrw.de/produkte/geobasis/vkg/dvg/dvg1/) and extract Dortmund from the gem (Gemeinden - municipality) or krs (Kreis - district) layers. However you get it, upload that shapefile as a zip file and you should be all set to get only the tiles relevant to Dortmund.



In Step 2, use the Search feature to find the orthoimagery or just scroll through. In 2023, they were saved as “dop”, so you could search with that and it would come up. Google Chrome translate will work well on this screen too, so if you do end up just scrolling it should be accessible.



Check the box and click proceed at the bottom to return to the download interface. You should be able to request it to prep the download and download all the tiles as a zip file.

For more information about this and other imagery files that are available, check out the online metadata [pages](https://www.bezreg-koeln.nrw.de/brk_internet/geobasis/luftbildinformationen/index.html).

Once we have the orthoimagery downloaded, we need to resample it to 1m. I have had mixed luck doing this to the original file, so it is sometimes better to split the 4 bands first. We can use raster calculator to do this. Identify the bands (as of 2023, Red = Ortho 1, Green = Ortho 2, Blue = Ortho 3, IR = Ortho 4) and multiply them by 1 to render the new file split into constituent layers. Make sure to name the layers clearly.

If you’re experienced with working with satellite imagery (on NDVI, etc.), you might be familiar with and expecting a procedure to recalculate the color values to top of atmosphere before moving towards an imagery-based remote sensing procedure - there is even a plugin in GRASS to do this. We don’t need to worry about this. First, we’re not classifying specific changes in vegetation, and second, they don’t embed the necessary metadata about location, timing, etc.

Now that you have the layers in separate files, you have dozens of options for how to resample them to 1m resolution. Two simple ones:

1. Use right-click>>Export>>Save As and change the resolution to be 1x1. Since the projection we’re using (EPSG 25832) is in meters, this does the job.
2. You can also use the GRASS command r.resamp.stats. Both options rely on GDAL commands, but the GRASS option is a bit more transparent. You can use average, median, or mode as the aggregation statistic. Average makes a smoother image but makes edges a bit harder to distinguish.

Once you have four layers at 1m resolution, your orthoimagery is ready to be used in the remote sensing process.

## Create other layers for ML

As mentioned above, machine learning is an extremely iterative process. It takes hours to learn but years to master - so the guidance included here is just my recommendation for *some* useful layers in the data context of Dortmund. This is not an exhaustive list, and you do not have to agree with me about which ones to include and exclude. The goal of this process is to inform the model as much as possible and create a land cover layer that accomplishes the basic goal - to differentiate between dirt, grass, roads/railroads, trees, water, and buildings.

### Orthoimagery neighbors

For this process, we use a pixel-by-pixel random forest algorithm instead of any kind of object classification. One of the major advantages of object classification is that it takes into account the context of a pixel when classifying it. In order to capture some of this strength while retaining the flexibility of a pixel-by-pixel algorithm, we incorporate a series of “neighbors” data points. Basically, we capture the RGB-I values of the north, east, south, and west neighbor pixels and use those 16 values as additional layers. To derive these 16 layers, we can use the r.neighbors function with a text input to identify the values of the neighbors. For quicker derivation, use batch processing (see below).

1. Step 1: create a text file for each of the neighbors you want. Text file looks like this for the west (left) neighbor:   
   0 0 0  
   1 0 0  
   0 0 0
2. Step 2: save the R, G, B, and I layers individually. You can run this as a batch process by using a configuration like this - link. Your output will be 16 rasters - Red west, Red north, Red east, Red south; Blue west, Blue north, etc.

### RGB-I Band Combinations

A number of combination of LandSat bands have been developed that help to distinguish different land cover types. While the RGB-I orthoimagery available in Dortmund does not have all the layers available in LandSat, combinations of red, green, and IR provide useful insights into vegetation, built space, and water. For details on other band combinations, see [here](https://web.pdx.edu/~nauna/resources/10_BandCombinations.htm).

1. The RGB-I bands are different for me than LandSat. The equivalency:
   1. Red – LS 3, Ortho 1
   2. Green - LS 2, Ortho 2
   3. Blue - LS 1, Ortho 3
   4. IR - LS 4, Ortho 4
2. Red over IR - Ortho 1 / Ortho 4 – TM3/TM4: This ratio has defined barren lands and urban area uniquely. But it could not define water body, forests and croplands.
3. IR over Red - Ortho 4 / Ortho 1 – TM4/TM3: This ratio distinguished vegetation, water and croplands. It has enhanced forests, barren lands. Because forests or vegetation exhibits higher reflectance in near IR region (0.76 -0.90u m) and strong absorption in red region (0.63-0.69u m) region. This ratio uniquely defines the distribution of vegetation. The lighter the tone, the greater the amount of vegetation present.
4. Green over Red - Ortho 2 / Ortho 1 – TM2/TM3: this ratio has distinguished croplands, barren lands sharply. But it hasn’t separated croplands, forests and water body. Both forests and water body has appeared as lighter tone and barren land appeared has dark tone. It did not enhance urban area. Chlorophyll has strong reflectance in the band 2 (0.52 -0.60u m) region and strong absorption in the band 3(0.63 -0.69u m) region, vegetation has appeared as higher tone.
5. Red over Green - Ortho 1 / Ortho 2 – TM3/TM2: This ratio has separated forests and croplands. Because band 3 (0.63-0.69m m) is the red chlorophyll absorption band of healthy green vegetation and band 2 (0.52-0.69m m) is the reflectance band from leaf surfaces. This ratio can be useful to discriminate broad classes of vegetation. Croplands have appeared as lighter (brighter) tone and forests appeared as dark tone.

There are hundreds of different vegetation indices. As long as you have a multi-spectral image, you can calculate them, and they’ll tell the model something different than just RGB-I. In addition to the ones above, i.vi will calculate any vegetation index you want (if you don’t want to do it manually with raster calculator).

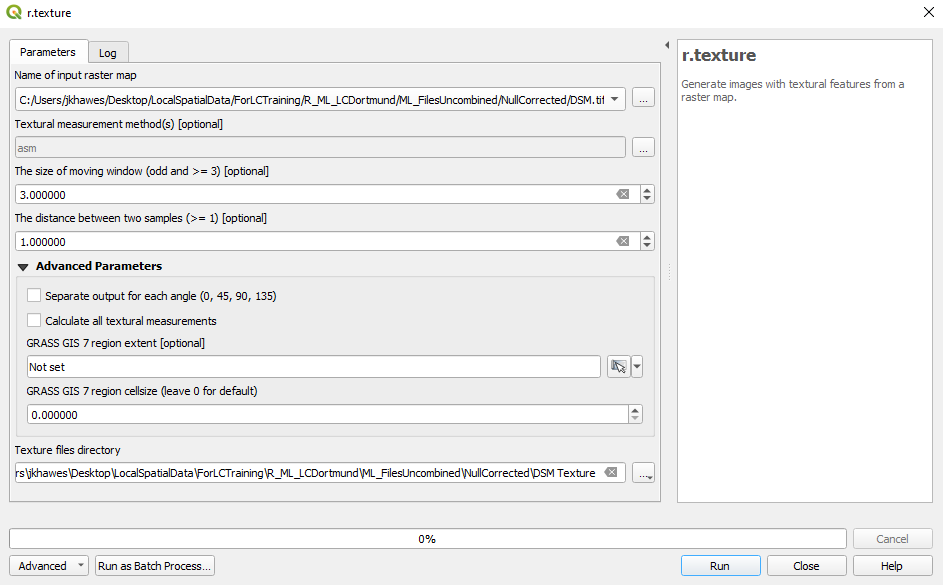
### DSM and DTM derivations

In my experience the DSM and DTM are not particularly helpful layers in their raw form. The DTM is obviously missing the raised features by its nature, while the ground level varies too much for the DSM to be insightful across the entire city. On the other hand, there are several derivations that have been helpful for me. Before deriving these layers, align the DSM and DTM to the orthoimagery. You could also align the final slope and land use while you’re at it to save time.

DSM minus DTM provides a useful layer identifying raised objects like trees and buildings. The raster calculator can be used to produce this layer from those developed in the previous step. I also found the slope of this difference raster to be useful. By leveling the ground, this slope raster really highlights trees as compared to other raised objects like buildings. The Slope function under Raster >> Analysis should do the job here.

### Texture

Texture has been used in a number of cases to support remote sensing applications. It is particularly useful when differentiating things like tree plantations and natural forests, where patterns over space are one of the key differentiators. In cities, it is not as useful, but it does offer some help, especially for specifying buildings, fields, and other smooth spaces that are harder to distinguish with just color. I use the r.texture command with the default parameters to derive the texture of both the DSM and the IR layers.



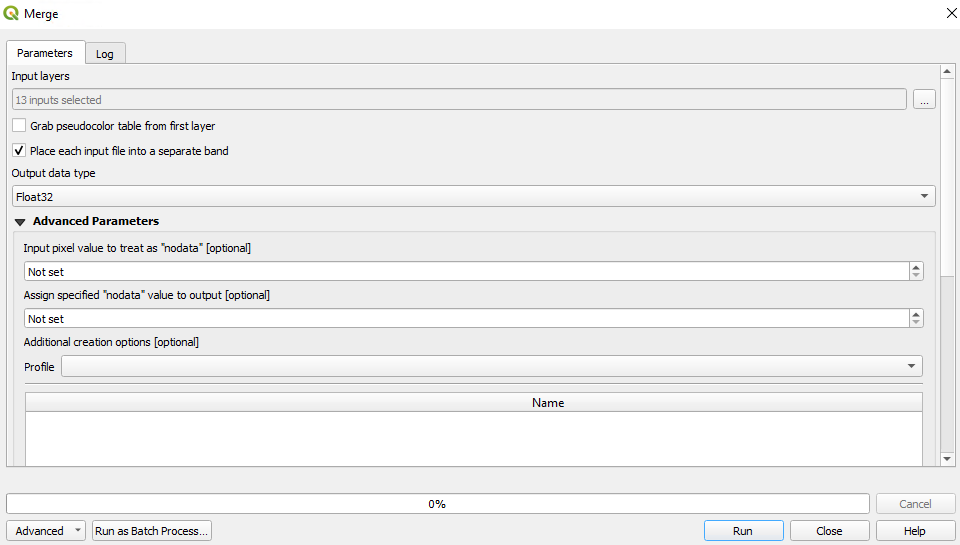
### PCA

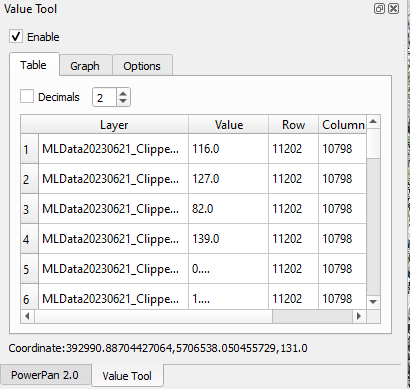
The last way that I’ve tried to derive other useful layers is with the i.pca command and running PCAs on the orthoimagery layers. I have actually not had good luck with this, but it’s a pretty standard practice in remote sensing, so I figured I’d point it out even though I don’t use it. If you’re unfamiliar with principal component analysis, the idea is basically to identify the key axes of variation in your data. Take multivariate data and simplify it into a few principal components - often you’ll just take the first layer it recommends and add it to your raster stack. You can find additional PCA guidance on the Internet [here](https://gisgeography.com/principal-component-analysis-gis-redundant-data/) and additional info on the tool [here](https://grass.osgeo.org/grass82/manuals/i.pca.html).

## Merge layers and tile raster

Merging the layers should be done in QGIS before moving to R. This is necessary because QGIS provides simple, quick sampling tools that also allow us to keep track of which layer is which.

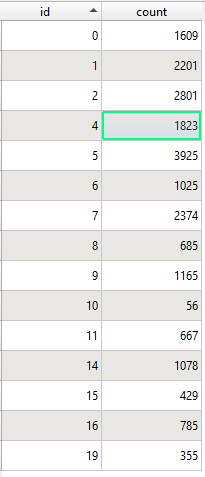
1. Align any layers that were created separate from the land cover process. Since most of these layers are derivations of one another, this shouldn’t be necessary, but it’s possible you’ve added something later or forgotten to align during creation at one point, and this is fairly easily solved at this point.
2. Before merging, it’s also helpful to re-clip the rasters to the Dortmund boundary. This should produce no data outside the city, but if it doesn’t you can also use r.null to make sure of that. Both of these things will help everything run faster.
3. Use the Merge tool to merge the aligned raster layers. I recommend merging directly from files instead of loading all the layers and merging from layers. Unless you have a much stronger computer than mine, QGIS will get very grumpy after a couple dozen 1m-resolution files of this size. If you select the layers one at a time (click the three dots by Input layers -> click Add File(s) -> select one layer -> hit ok -> repeat), the layers should appear in the order they’re selected. Otherwise it’s usually alphabetical ordering, though I can’t promise this because I’m not entirely sure of the background algorithm. You can also select all your layers at once and then manually reorder them, but I’ve found this to be a painstaking process and not particularly reliable (things inexplicably move around sometimes).



1. Once the layer is merged, use the Value Tool plugin to make sure you know the order of bands (so we can rename them in R). You can also track down the mergeInputFiles.txt file, which is reported in the Merge output window. My final order of bands was:
   1. "Red", "Green", "Blue", "IR", Red\_over\_IR", "IR\_over\_Red", "Red\_over\_Green", "Green\_over\_Red", DSM\_minus\_DTM", "DiffSlope", "DSMTexture", "IRTexture", RedNorth", "GreenNorth", "BlueNorth", "IRNorth", RedEast", "GreenEast", "BlueEast", "IREast", RedSouth", "GreenSouth", "BlueSouth", "IRSouth", RedWest", "GreenWest", "BlueWest", "IRWest"
   2. Value tool example:   
      
2. Unfortunately, we can’t classify the whole raster at the same time - it’s too big for R to handle as a dataframe. So instead we can tile the raster and run it through a for loop.
   1. The easiest way to do this is with the Save As function for a raster.
      1. Right-click on the full raster in the layers. Go to Export > Save as.
      2. Check “Create VRT” in the top-right, which should open a new options box that will let you set the maximum size as 1000x1000 (1km x 1km). Identify the output folder, and let it run. The tiles should be numbered sequentially.
   2. Another way to do this involves the Easy Raster Splitter plugin. So if the above isn’t working for you, download that.
      1. Before using the plugin, you need a vector to split the raster with. You can make it with the create grid tool under Vector > Research. I usually do 1km x 1km rectangles just for ease, but technically you can do whatever size as long as the rasters are small enough to run. I imagine you could do different shapes as well, though I’m not sure why you’d want to. Points obviously won’t work.
      2. Once the plugin is ready to use, you can set a new folder for the new rasters and split the file with the grid you just created. Conveniently, the grid comes with a built-in sequentially numbered id field for you to use with the output name.

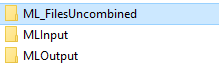
## Develop training layer

The most important ingredient in a machine learning process is the training data. In this case, this is a point layer where you manually identify several thousand points and classify them as you desire. You then tell the computer to learn about what kinds of raster cells match each classification, and the computer uses what it learned to classify a bunch of raster cells it doesn’t have training for. So when it comes to machine learning training, as the old saying goes, garbage in = garbage out. There is no end-all, be-all recipe for developing a training dataset, but the basics steps are:

1. Create a coding scheme that satisfies the information you need. For example, I use the following:
   1. Dirt = 0
   2. Shaded dirt = 10
   3. Low vegetation= 1
   4. Shaded low vegetation = 11
   5. Tall vegetation/dense trees = 2
   6. Pavement - street, sidewalk, or parking lot = 4
   7. Shaded pavement - street, sidewalk, or parking lot = 14
   8. Black/grey/white roofs = 5
   9. Shaded black/grey/white roof = 15
   10. Red/brown/other roofs = 6
   11. Shaded red/brown/other roofs = 16
   12. Water = 7
   13. Shaded water = 17
   14. Container, car, or other temporary object = 8
   15. Railway = 9
   16. Shaded railway = 19
2. Many of these categories are technically superfluous, since I don’t actually need to distinguish between different kinds of roofs or rail vs. road. However, it helps the computer to more reliably distinguish dirt from roads if I also code railroads separately.
3. Once you’ve established a classification, create an empty shapefile layer with one field: id.
4. In your empty shapefile, start adding points on exemplary cases. You want lots of good examples. If you create a cluster of points, you can copy-paste them with ctrl-c and ctrl-v and just use the move tool (on the Advanced Digitizing Toolbar near the left) to move them where you want them (since they’ll by default show up on the original spot).
   1. In the case of Dortmund, I deal with shade separately since the sun is at a pretty serious angle in most of the imagery. You can always merge them later.
   2. Can use high-res data for the visual inspection (could even use Google if you don’t have that). But then you extract the values from the layer you’re going to model. Just using the high-res stuff so you’re sure what you’re looking at if you’re zoomed in.
   3. Add an “other” category for things like shipping containers on construction sites
   4. You need a balance between the different land use codes for training. You can overtrain certain classes – but also, some things need more attention - e.g., roofs and dirt look more diverse than grass.
   5. On my first round of training, I tried to have more than 1000 training points for important classes (e.g., grey roofs) and more than 500 for most of the others. After many rounds of refining (see below), these were my final counts:
      1. 
      2. Note: I specifically eliminated 17 (shaded water) during one of my revisions because it was confusing shaded street with shaded water and led to a lot of error. Another option would’ve been to keep iterating the training, but since we have a water overlay in the land cover layer we use below, I didn’t see the point.
5. Once you have selected thousands of training points, you can use the Point Sampling Plugin to construct a layer with all the different layers. Obviously save this separate from your original point layer so you can always go back and edit the original points.

## Random forest algorithm

From here, we move to RStudio. I won’t walk you through this code, since it’s written and commented pretty extensively here. But here are the basics. We rely on the usual spatial packages (raster, etc.), but we add caret and randomforest. We use caret to specifically analyze how our training is going by creating a training and testing dataset. Then at the end we switch back to randomforest to train the final model and predict the images with this. Here’s the trick: you’re probably going to have to do this a couple dozen times at least before you’re satisfied with the image. You’ll add and remove training points (see above), you’ll add and remove raster layers (see above), you might even change the coding structure if you decide mine is unwieldy (see above - I did this at least twice). So what you’re going to do is create a file structure that allows you to easily edit the input rasters and shapefiles. Mine looked like this:



I would merge the current layer stack (multi-band raster) for machine learning, then create the tiles in MLInput. I’d always name it something similar so I wouldn’t have to edit that part of the code much - I saved old inputs in a totally different directory after I was done with them just to have them out of the way. If I really disliked the input, I just deleted them. I told the R code to create my tiled outputs in MLOutput. And I saved my shapefile in the parent directory, renaming it each time I changed the points or the layers.

The basic structure of the R file is:

1. Import training data. Detect outliers and delete them.
2. Use caret::train to create a model and produce some evaluation metrics for you to look at. Do this with both the data segmented and the full training dataset. You can either stop here and edit the training now or run the full model below before redoing it.
3. Use randomforest to create a final model and a for loop to predict the tiled images.

A couple rules of thumb for the random forest algorithm:

* Set the random seed so it’s a replicable process. You can always change this if you want to compare.
* Mtry should start out as the square root of your number of variables - just a rule of thumb. This is usually good enough.
* Number of trees should probably not exceed 500. Just excessive and overtrained.
* You can use tuneRF to help you identify the optimal value of these parameters.
* I recommend training about 10-15 tiles at a time while you’re iterating - enough space to see a lot of variation but small enough to run on the order of 5-10 minutes depending on the computer. This is commented out in the current code - just changes i to range between 240 and 250 instead of the full range.

Once you have some tiles you want to look at, go back to QGIS. Merge those tiles together (don’t put them in separate layers this time), and reclassify them so they’re more intelligible (e.g., combine all roofs, etc.). I created a text file with my reclass rules in it so I could do this quickly and iterate - see here. Once it’s been reclassified, you can change the color scheme to something intelligible. Again, I created my own and saved it for ease of access - see here. If you want to have a better feel for what the final layer is going to look like, you can do two things:

1. Some sort of sieving will clean up the layer. While the Sieve tool is very easy, the two-step r.neighbors procedure is better. First, go to r.reclass.area. Set the area size much lower than 1 hectare - more like 0.0004 hectares (4 sq meters). Run the algorithm, which will produce a layer with just the isolated pixels. Next, run r.neighbors with the merged/reclassified layer as the Input and the reclass.area layer as the raster layer to select the cells which should be processed. Make sure to change the neighborhood operation to median or mode to retain the classification structure, and you should be able to use the default settings for most of the rest. Run it, and a slightly cleaner sieve should be the output.
2. You can also use your overlays as visual overlays to help you ignore things that are going to get cleaned up in post-processing. For example, if you algorithm is convinced that that car in the middle of the road is a tree, you don’t really need to worry about it, since we have a roads overlay we can use that covers most pavement in the city (not all, which is another good reason to have it open so you can check).

As you see fit, iterate this process, the training layer process, and the merging layers process - always try to keep in mind the needs of the project. For example, if you have errors right on the edge of a building, that’s often probably ok, since it’ll be really shaded anyway. Don’t spend months on this. Above 90% accuracy is a good goal, and the metrics produced by caret are helpful for monitoring improvement, but the most effective judgment is your visual inspection. In terms of visual inspection, remember to look at the whole map before calling it a day - there’s a chance your sample of tiles is a little biased and you might need to make a couple tweaks that you wouldn’t have noticed in just the sample that you were working with to start.

Once this is complete, you can merge the unnecessary categories to get the final scheme. In my case, this meant:

* 1 = Impervious - Recode 4, 8, 14, 18 as 1
* 2 = Low vegetation, grass or dirt - Recode 0, 1, 3, 10, 11, 13 as 2
* 3 = Roof - Recode 5, 15, 6, 16 as 3
* 4 = Trees - Recode 2 as 4
* 0 = Otherwise occupied - Recode others (7, 9, 17, 19) as otherwise occupied

## Synthesize Land Cover Layer

Finally, we can use additional layers to supplement our remote sensing with what little detailed land cover data we do have. To do this, we’ll create two overlays. We have the landscape model which characterizes some land cover types we care about, as well as the land use map we used earlier, which identifies railroads and roads, among other things.

### Land use layer as an overlay

We can derive several things from the land use model and its constituent layers. First and foremost, we can clear up any confusion between roads, railroads, and water. We can do this with a couple simple raster operations. First, we can reclassify the LU layer to have roads as 100, railroads as 200, and water as 300. This is a simple r.reclass operation:

80 = 100

81 = 100

82 = 200

83 = 300

\* = 0

Once we’ve reclassified this layer, we can use raster calculator to generate a LC layer with values in the 100s, 200s, and 300s (raster calculation LC + RRW). The resulting layer can then be reclassified:

0 = 0

1 = 1

2 = 2

3 = 3

4 = 4

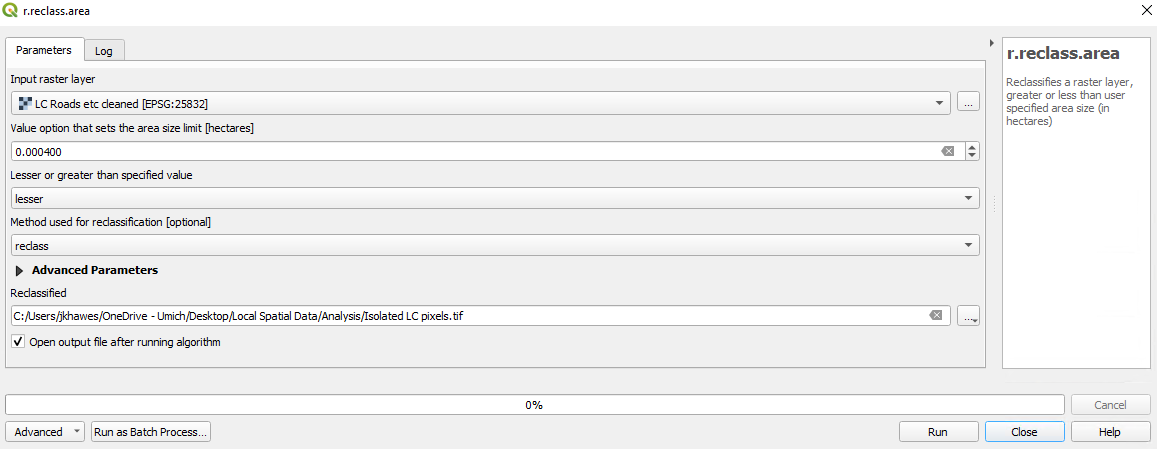
100 thru 104 = 1

200 thru 204 = 0

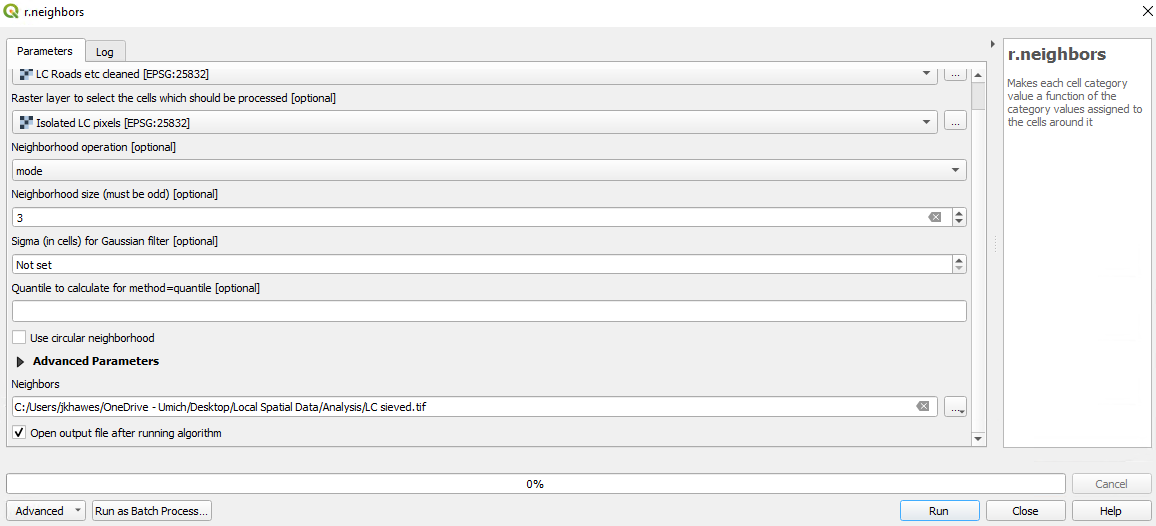
300 thru 304 = 0

### Sieve - r.reclass.area and

Again, some sort of sieving will clean up the layer - we did this while iterating the LC layer but have not done it yet with the final classification. While the Sieve tool is very easy, the two-step r.neighbors procedure is better. First, go to r.reclass.area. Set the area size much lower than 1 hectare - more like 0.0004 hectares (4 sq meters). Run the algorithm, which will produce a layer with just the isolated pixels. This might take quite some time to run if you’re using a standard computer. Upwards of an hour.



Next, run r.neighbors with the merged/reclassified layer as the Input and the reclass.area layer as the raster layer to select the cells which should be processed. Make sure to change the neighborhood operation to median or mode to retain the classification structure, and you should be able to use the default settings for most of the rest. Run it, and a slightly cleaner sieve should be the output.



The resulting file should have values 0 thru 4.

* 0= completely unusable space
* 1= impermeable
* 2=low vegetation, grass or dirt
* 3 = buildings
* 4 = trees

You should now have a usable land cover file.

# Sunlight Availability

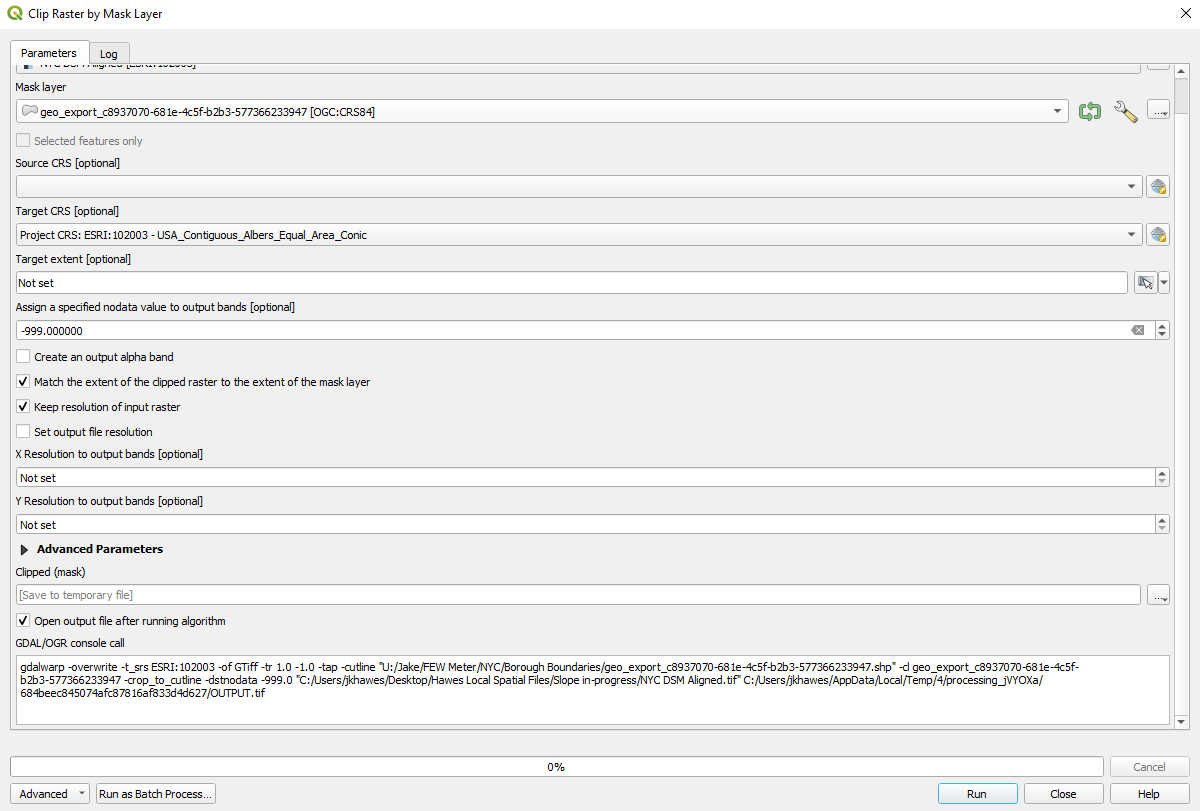
The final layer we will derive is a sunlight availability layer. This will take advantage of the r.sun package in GRASS, which takes the DSM and converts it to solar irradiance. We follow the same process originally deployed in NYC, replicated here. This one is fairly complex, so I originally derived my process from an [example done in Canada](https://www.sciencedirect.com/science/article/pii/S0038092X10000812?via%3Dihub#fn15). Based on that paper, we need a few inputs to make this work, including:

| Canadian layer/example | Dortmund layer/notes | Source link |
| --- | --- | --- |
| Digital Elevation Model (DEM) | DSM - include shading from buildings and trees |  |
| Slope/inclination | Derived from DSM |  |
| Aspect/orientation | Derived from DSM |  |
| Latitude | Not necessary if we use a proper projection (...) |  |
| Albedo: the ratio of diffusely reflected radiation on a surface to its incident radiation. | Albedo can probably be calculated for each city with this function, or we can use urban averages. For i.albedo, just need landsat imagery: https://grass.osgeo.org/grass78/manuals/i.albedo.html |  |
| Mean days and corresponding angular position of the sun. | Can use the same mean days if we do want to do the calculation for every month. “ Table 1.6.1 in Duffie and Beckman (1991) readily provides the day of month, day of year and δ (sun declination) values to input into the simulation -- J.A. Duffie, W.A. Beckman, Solar Engineering of Thermal processes (second ed.), John Wiley & Sons (1991)” |  |
| Linke turbidity: a convenient approximation to model the atmospheric absorption and scattering of the solar radiation under clear skies. | If all we want is very high level stuff, we can get that from the same place the example paper did. Resolution is about the scale of NYC. Have three different raster cells for whole city, all the same value. Able to make a raster with the resolution of our DEMs and DSMs by downsampling. | <http://www.soda-pro.com/help/general-knowledge/linke-turbidity-factor> |
| Ground-measured values of global horizontal irradiation (GHI). | Available from NASA SSE POWER program - GHI is the first value (ALLSKY\_SFC\_SW\_DWN CERES SYN1deg All Sky Surface Shortwave Downward Irradiance (kW-hr/m^2/day)) while GHI under Clear-Sky conditions is the second value (CLRSKY\_SFC\_SW\_DWN CERES SYN1deg Clear Sky Surface Shortwave Downward Irradiance (kW-hr/m^2/day)) | Available at a 1x1 degree resolution. It claims to be ½ by ½ but doesn’t seem to output that for 2019 at least. – <https://power.larc.nasa.gov/data-access-viewer/> |
| Clear sky index Kc: “Ratio of the global horizontal irradiance to the global horizontal irradiance under clear sky conditions. It is important not to confuse and hence misuse this definition with those for insolation clearness index and clear sky insolation clearness index.” | Available in the POWER suite of indicators as ALLSKY\_KT. We do not use the normalized value - this transforms the Kc with the latitude - Kc = shortwave direct horizontal (GHI) / shortwave direct top-of atmos -- I haven’t figured out the use for the normalized parameter yet. Maybe comparing different locations? See [here](https://www.star.nesdis.noaa.gov/smcd/emb/radiation/solar_resource_definitions.php) for simple definitions. See [here](https://www.star.nesdis.noaa.gov/smcd/emb/radiation/documents/SRDB_1.0_Parameter_Definitions.pdf) for other details.  Definitions (rather unhelpful, except it explicitly mentions GHI): <https://power.larc.nasa.gov/#resources> | The regional data access panel at the website above allows “NetCDF” export, which can be imported as a raster in Q: <https://ereefs.aims.gov.au/ereefs-aims/help/how-to-open-a-NetCDF-file-with-ArcMap-and-QGIS>  So basically we turn this into a raster of the appropriate resolution and multiply to get a final value from r.sun. |

With this set of inputs, we should be able to calculate the shading effects in essentially any city around the world. The DSM is the hardest thing to find, and we should be able to simulate this with building height data, which is more often available. In Dortmund, we’re good to go with our highest-hit DSM.

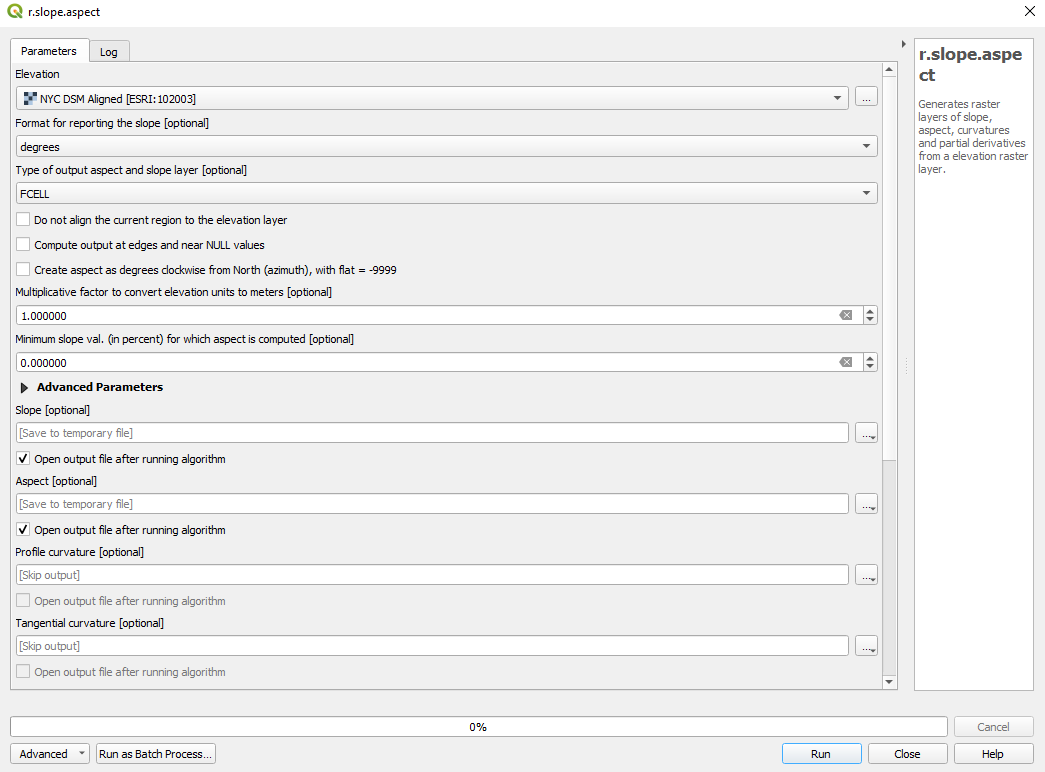
## Clip DSM to work better with GRASS

For some reason, of all the commands we use, the GDAL commands embedded in r.sun are the only ones that use the default maximum raster size, and they won’t save anything too big. So before moving forward, we have to clip the DSM to work with this limitation. This is best achieved by clipping to the borough boundaries. See the settings below.



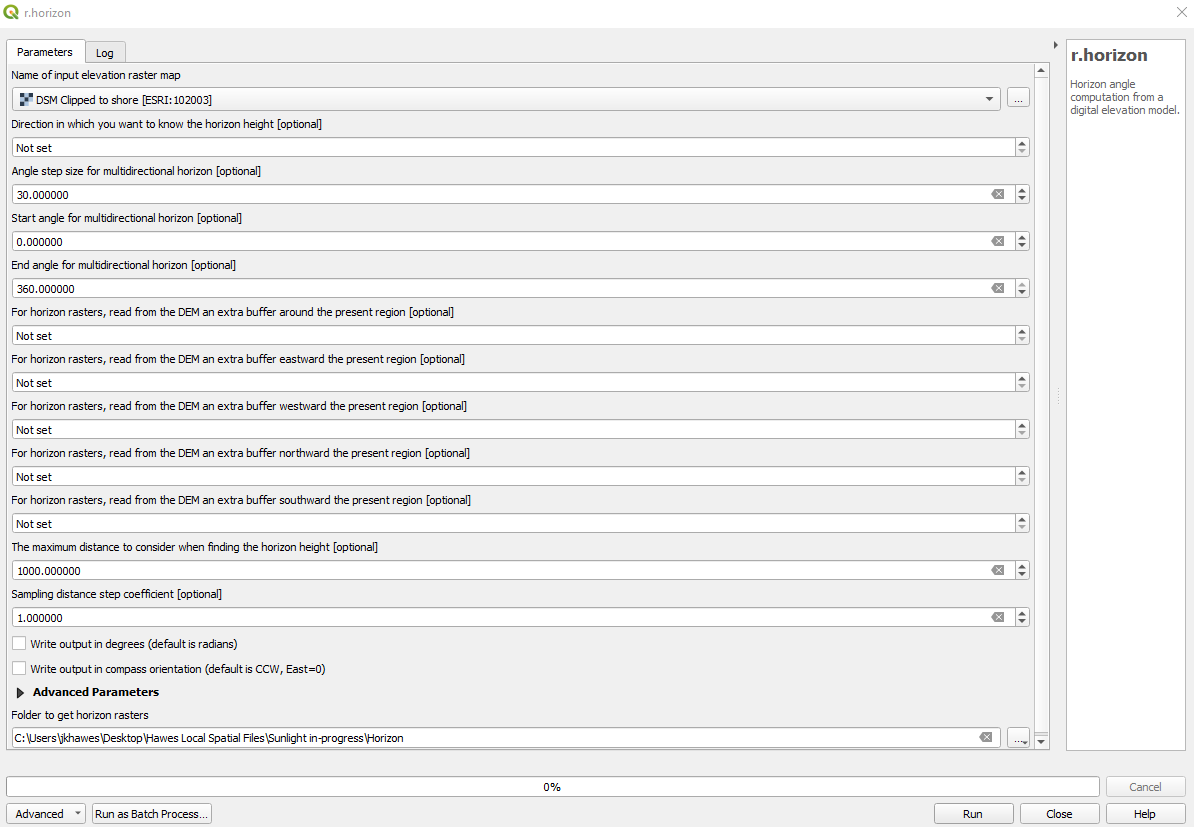
## r.slope.aspect

Once we have a useable DSM, we need to produce maps of the slope and the aspect based on our DSM. We could use the DSM slope map we already have, but it’s just as easy to just run it all within GRASS to make sure everything is formatted the way r.sun wants it to be. This command is fairly straightforward, only a couple things need to be customized. First, we need to uncheck the box that asks about aligning with the elevation region. We do want to align all of our calculations with that region. Second, we want to suppress the outputs other than slope and aspect. No reason to spend time calculating things we won’t use.

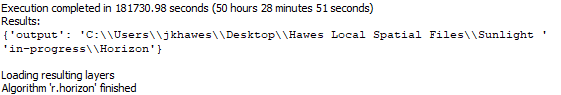


## Optional: r.horizon

If you are planning to use GRASS GIS more directly via commands, it can be very helpful to run r.horizon to determine the horizon height at all locations in the city. Unfortunately, if you plan to run the r.sun suite via QGIS, the interface does not play nice with loading an entire directory, which is the required format of the r.horizon output and the r.sun input. I’d recommend skipping this unless you really want to do it with programming - takes a very long time to run.

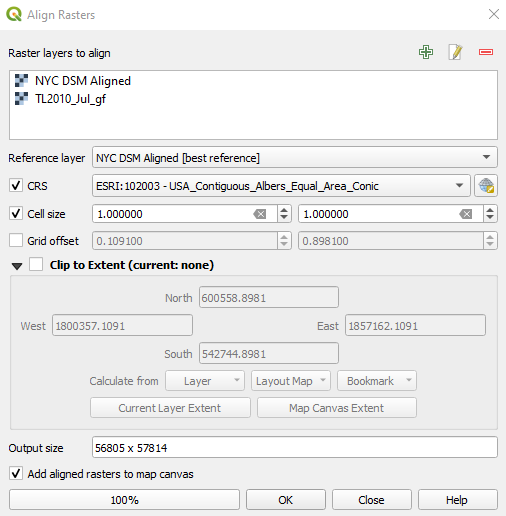


This will take a very long time to run - on the order of 50 hours on the Beast at 1m resolution. In fact, the last time I ran this, it ran for 50 hours and only saved 120-360, so you may have to run it twice to convince it to save everything correctly.



## Aligning Linke

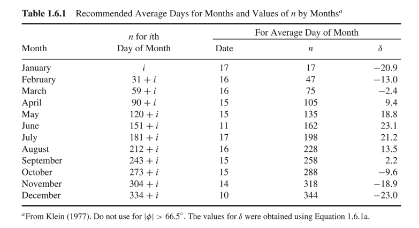
The last step before running the final command is to align the Linke values with the files we’ll be using. This will also clip the raster and will probably take 10-15 minutes (obviously align the Linke files with the Dortmund DSM, this is just an example from NYC).



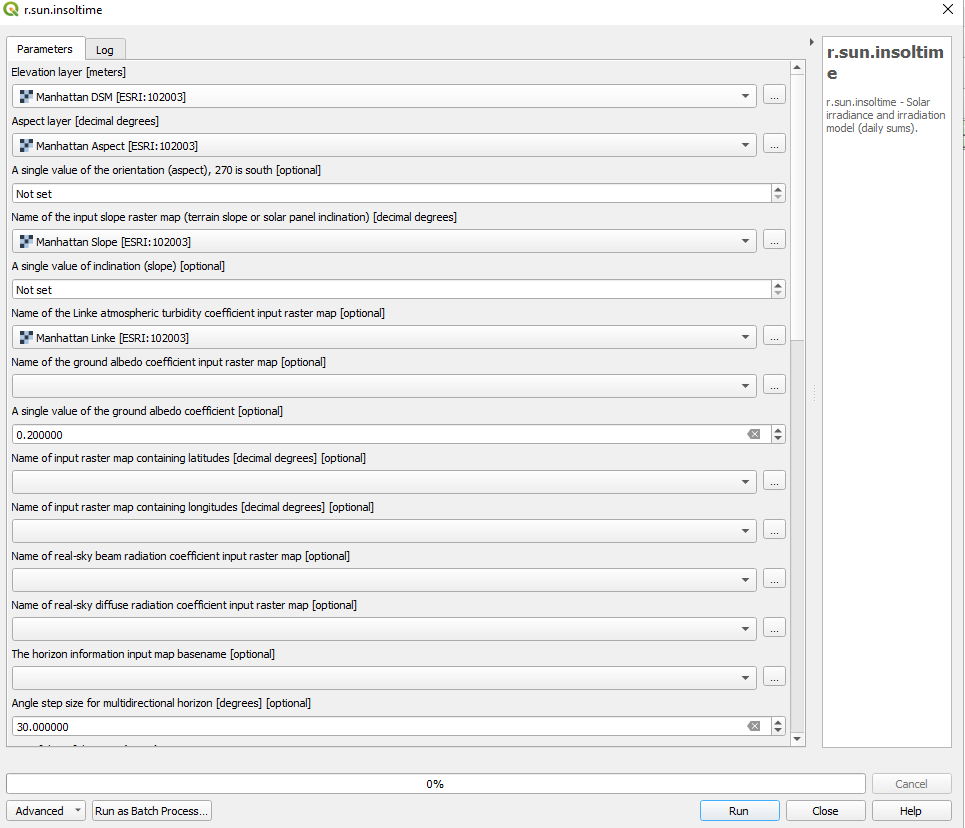
Once the Linke values have been converted to a 1x1m raster, we should be able to clip this to the borough boundaries, as we’ve done with the others.

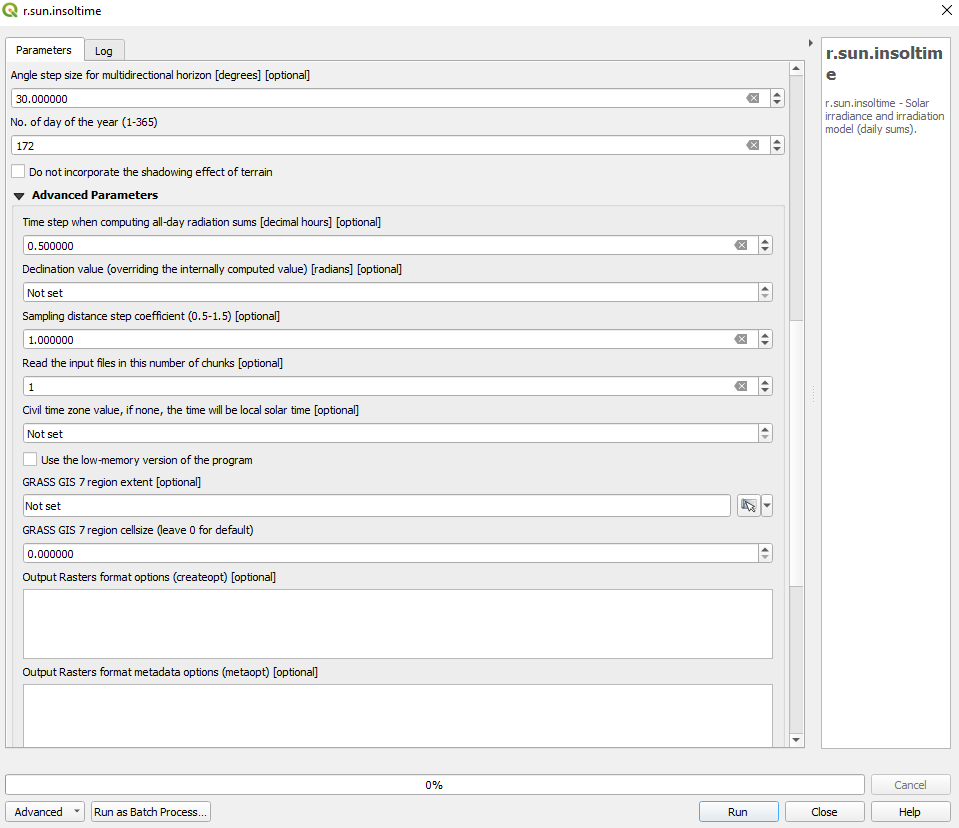
## r.sun.insoltime

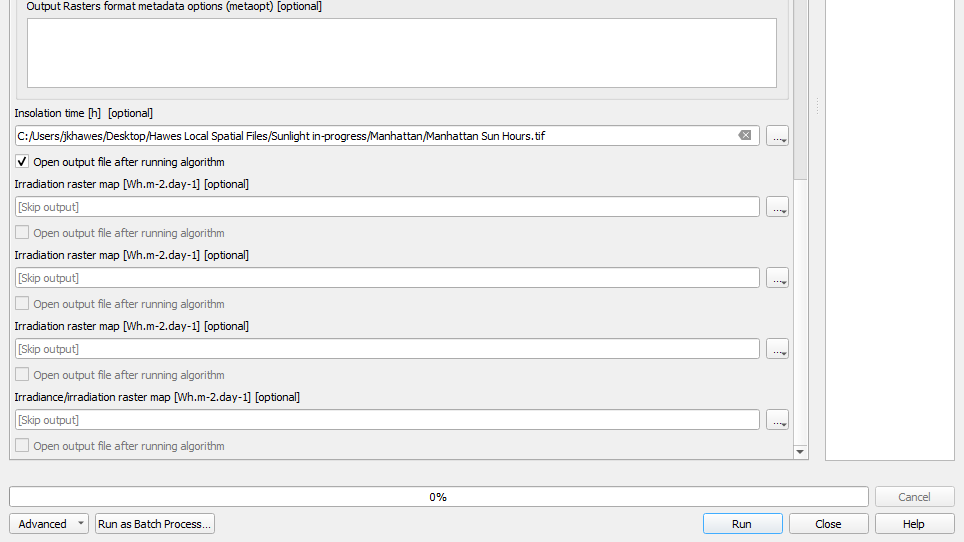
If we are strictly interested in the number of hours of sunlight, then we can simply ignore the more complex aspects like levels of radiation from the NASA data. We can retrieve average days from the book cited in the Canadian paper:



This means that the inputs look like this:







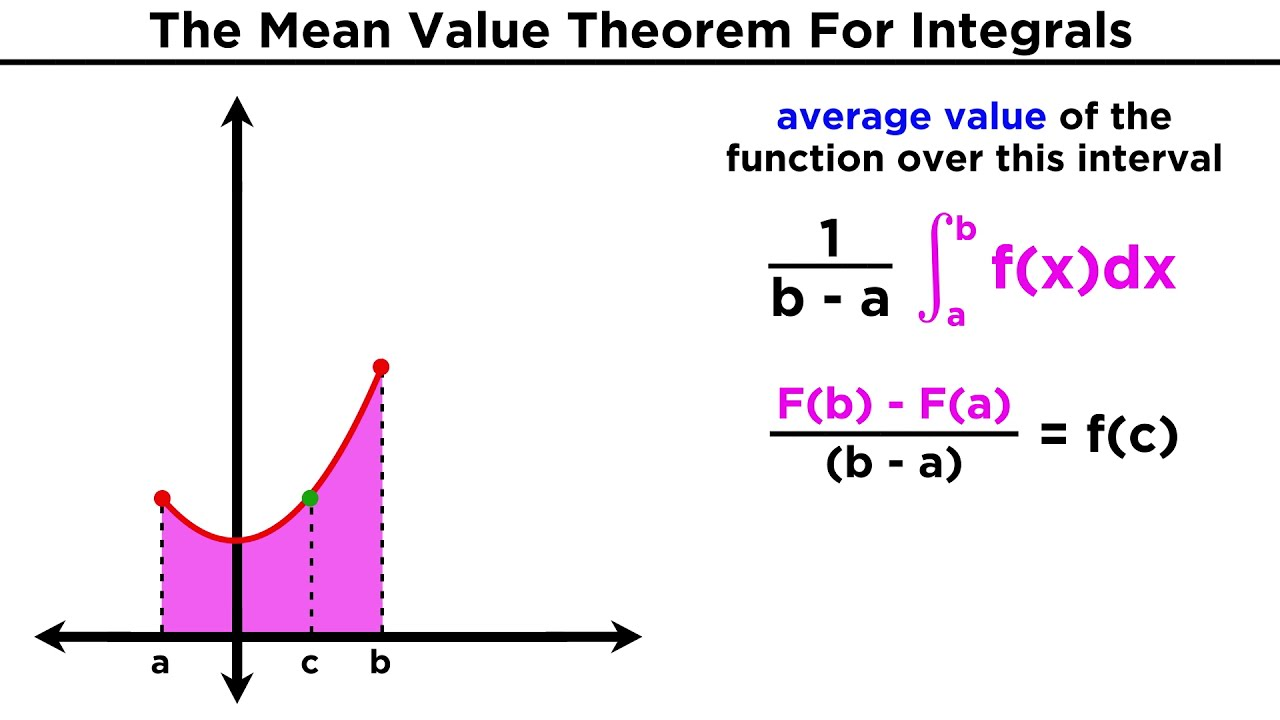
These commands run slowly, but stubbornly. It took about 1.5 days to run it for each month. Despite that extremely long calculation, it produces good results similar to this example from NYC.



## Merge results and calculate average

First, we obviously need to recombine these tiles. This can be run as a batch process and should take several hours to run.

Once we have full sunhours layers for each month, we can take the average value of the function over the desired interval (the time between April average and October average). We can either try to fit and upside down parabola or we can assume two piecewise linear functions. The second produces more reproducible math between locations and on each cell, so we live with that simplification. The rest of the calculus in this whole process has been hidden in algorithms in QGIS, but you can actually follow along as we derive this particular equation. The mean value theorem says that we can calculate the average value of a function over any particular interval on which it is continuous, which our piecewise function is. For piecewise functions, we calculate the mean value theorem of the constituent pieces and then take a weighted mean.



We derive our equation with the standard y = mx+b, assuming that slope is linear:

For April to July, this is:

For July to October, this is:

So once we conduct the integration, the y goes away and the x gets filled in, but that still leaves us with b1 and b2. So before we can jump to the mean value theorem, we have to calculate the value of b1 and b2. We can do this by simply plugging in values we have already:

If we plug July into April to July, we have:

Solving, we end up with:

If we plug July into July to October, we have:

Solving, we end up with:

We can run both of these as raster calculations and end up with b1 and b2 as rasters. Now we can go ahead and integrate.

This is obviously a more complicated bit of math. For April to July, we end up with:

Following this same math for July to October, we end up with:

At the end, this turns out to be five raster calculations. We calculate and first with separate raster calculations, then we can directly calculate and .

The final calculation is just one final raster calculation - .

For b1, the raster calculation looks like this: "July Sunhours@1" - ( 198\* ( ( "July Sunhours@1" - "April Sunhours@1" ) / 93 ) )

For b2, the calculation looks like this: "July Sunhours@1" - ( 198 \* ( ( "October Sunhours@1" - "July Sunhours@1" ) / 90 ) )

For MeanValue1, the calculation looks like this: ( 28179 / 17298 ) \* ( "July Sunhours@1" - "April Sunhours@1" ) + "b1@1"

For MeanValue2, the calculation looks like this: ( 43740 / 16200 ) \* ( "October Sunhours@1" - "July Sunhours@1" ) + "b2@1"

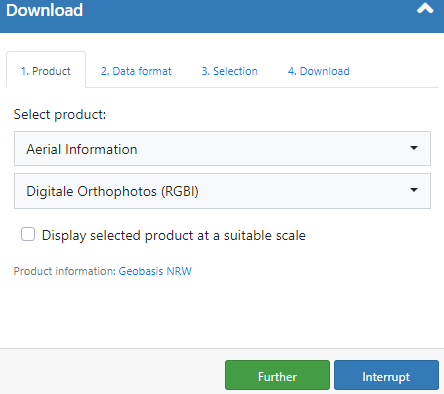
For MeanValueOverall, the calculation looks like this: ( ( 93 \* "MV1@1" ) + ( 90 \* "MV2@1" ) ) / 183

## 

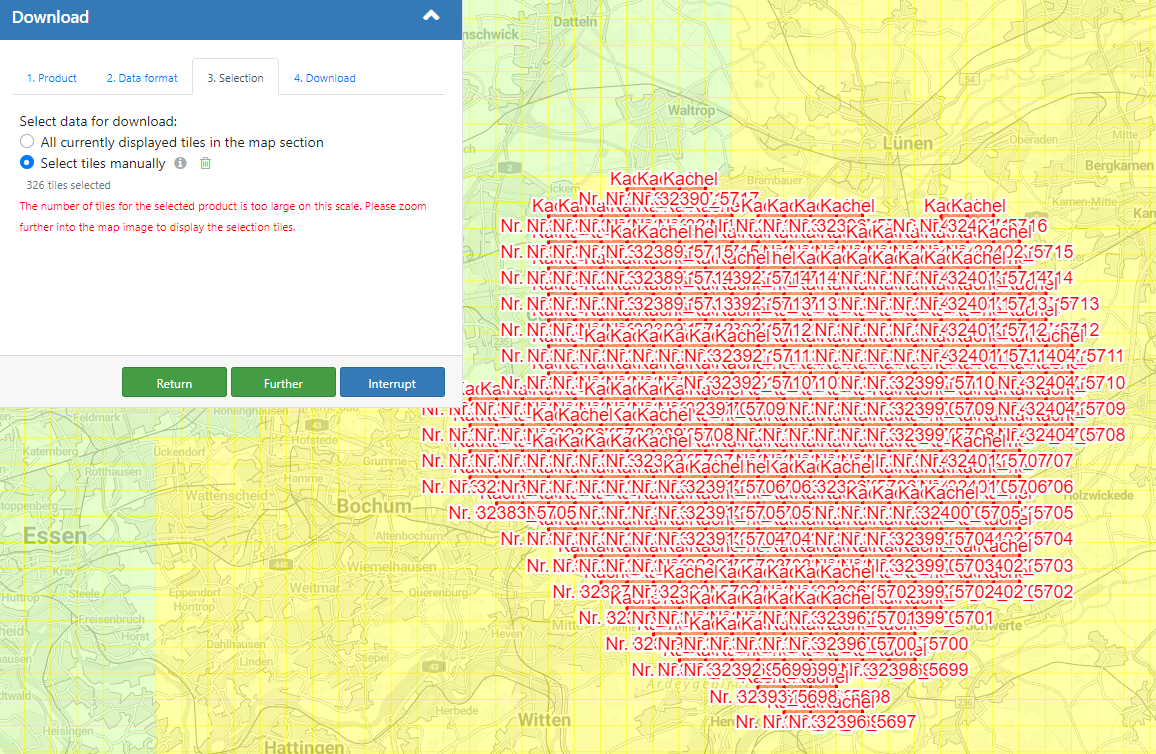
# Unused notes on data availability and other analysis techniques

## Other orthoimagery source

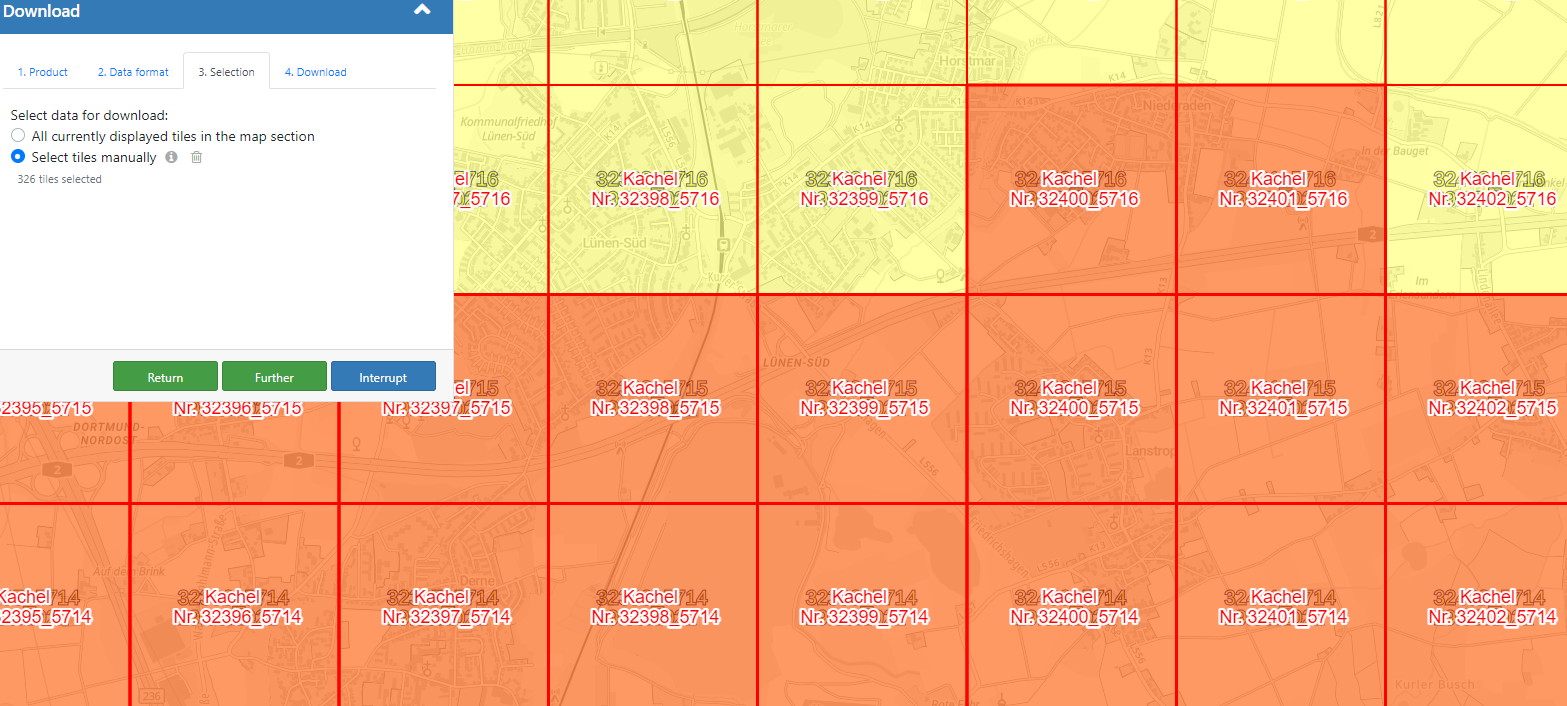
According to the [NRW website](https://www.bezreg-koeln.nrw.de/brk_internet/geobasis/luftbildinformationen/aktuell/digitale_orthophotos/index.html), we can download orthoimagery in RGB-I from a [data portal](https://www.tim-online.nrw.de/tim-online2/?bg=basemapDE_grau&scale=1400000&center=450000,5700000&wms=https://www.wms.nrw.de/geobasis/wms_nw_dop-uebersicht,nw_dop_uebersicht&opacity=70&legend=true) online - they also have a variety of other data, but this is the one that’s available online. You can use the translate feature in Google to get a pretty good translation of the different menus, and the result should look something like this:



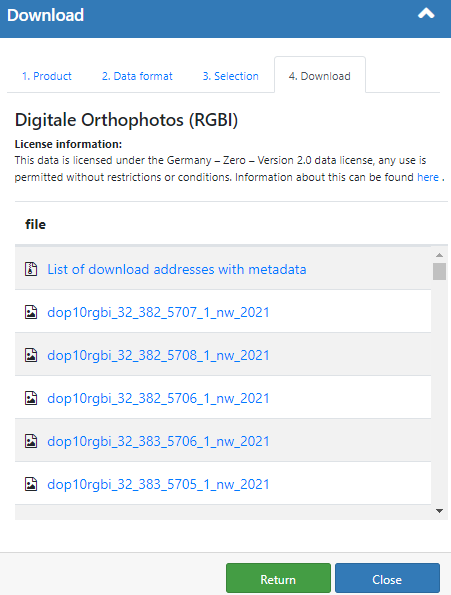
Click Further to continue and select the .tiff, which should be your only option. On screen 3 (Selection), I couldn’t get the auto-select option to work well, so I used the manual selection. Unfortunately, there is not a way to upload a shapefile or anything, so we have to manually select the tiles we’re interested in. For the first round of download, I just approximated the Dortmund boundary file by working my way around the perimeter and looking for landmarks. I created the boundary and then just filled it in by selecting all the interior tiles as well.



For example, on the NE side it looks like this:



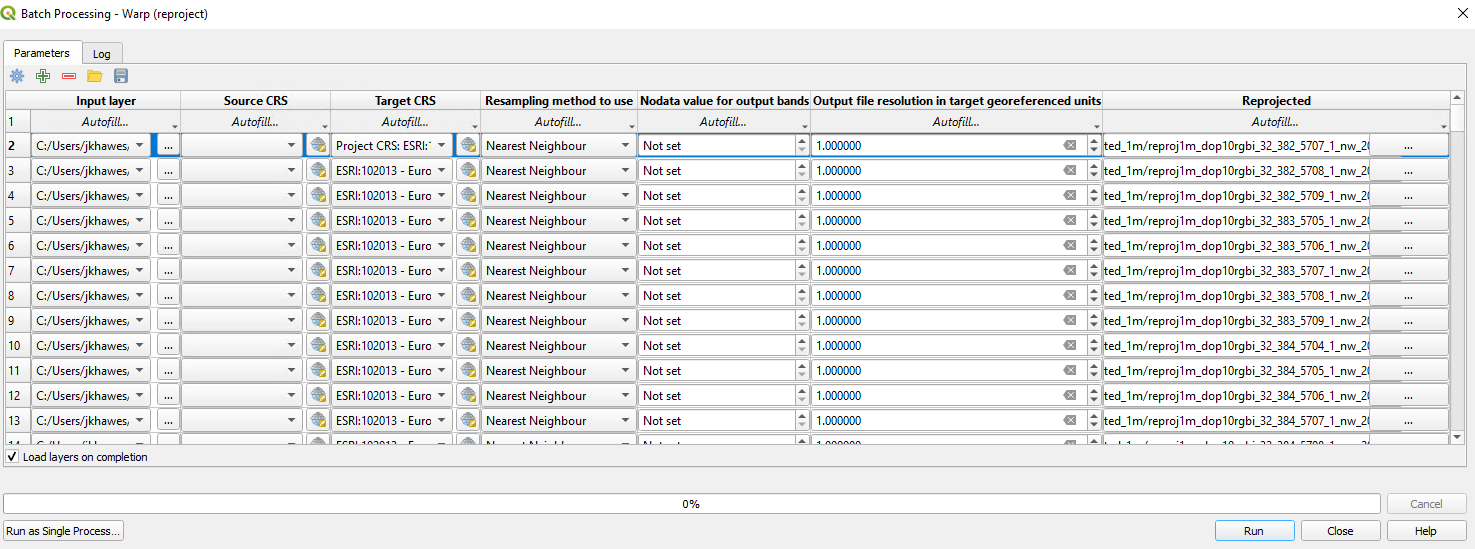
The good news is that you shouldn’t have to do this again, even if you need the data again. In addition to all the download buttons, the portal provides the download link for all the selected files as a txt file, which I’ve made available [here](https://drive.google.com/file/d/1LCo8WesUMvVMsei83etbVSZpkQm1iKBp/view?usp=sharing).



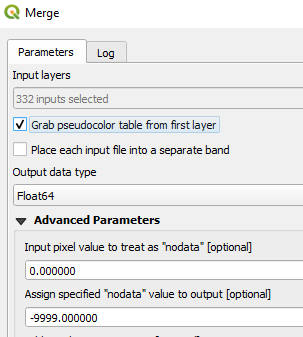
Really all that’s left to do once you have this list is just to go ahead and click through each file to download it.

## Messy conversion from 25832 to 102013

1. We can simultaneously warp the data and resample them to 1m. We can do this for all the orthoimagery at once, and it shouldn’t take a particularly long time to run (~20 minutes on the beast including loading the layers). If we use the Warp tool in Batch Processing mode, we can use the Autofill button under Input layer to import the entire orthoimagery directory. Once in a while this crashes the warp tool, but just try it again and it should work. From there, set the Target CRS to 102013 and the Output resolution to 1 (use autofill down). You can leave the Resampling method as Nearest Neighbor. I experimented with the different options, and in this case it makes no difference at all which one you select. Select the three dots on the top row under Reprojected to make a file and a file name prefix. I used reproj1m\_ and then used the autofill with parameter values setting to make that the prefix for the original file names.



1. We can go straight to a merge from here, but we have to make sure it’s filtering out the overlapping areas where there is no data. We also have to tell it to take the pseudocolor table from the first layer, or it will only merge the first band of the raster. Make sure to check the directory for the merge input text file when you merge so you don’t have to go figure out what each band is.



## Landscape Model as an overlay

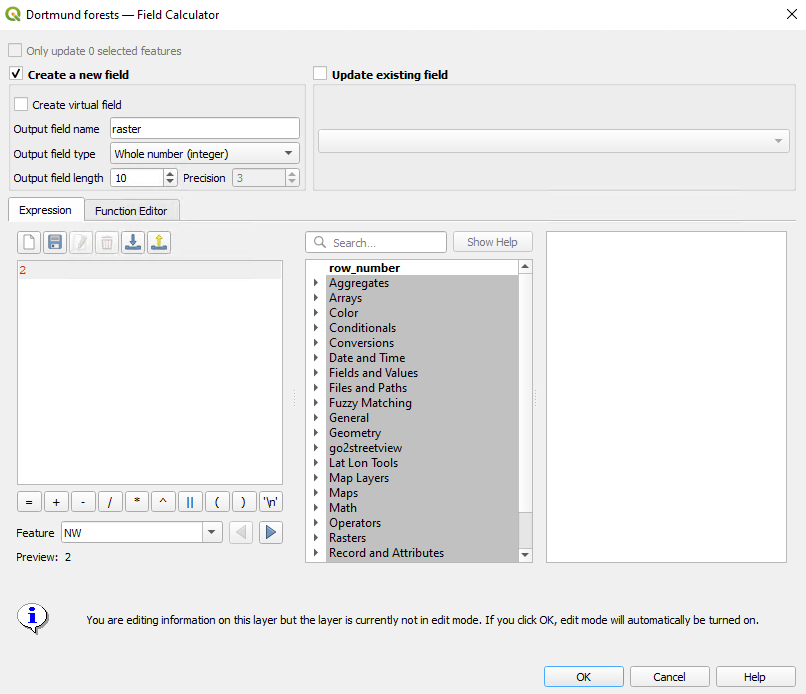
The so-called “Landscape Model” for NRW can be downloaded from [this link](https://www.opengeodata.nrw.de/produkte/geobasis/lm/akt/basis-dlm/) - I use the shapefile out of preference and habit. You can find more information about the dataset [here](https://www.geoportal.nrw/?activetab=map#/datasets/iso/4c8955f8-7c72-498e-bb3a-55b1680a4ec6) and [here](https://www.bezreg-koeln.nrw.de/brk_internet/geobasis/landschaftsmodelle/aktuelle_landschaftsmodelle/basis_dlm/index.html), though most of what you need is downloaded in the zip file with the data. Only a few of these data layers are useful for our land cover map generation, but it’s worth extracting them because it helps us clean up the final vegetation layers. The first step for English speakers of course is to translate the land cover data from German. According to the metadata embedded in the zip file, these are all of the code, which I use Google Translate to work through:

| Ebene | Objektart | Tranlastion | Objektartcode | Anzahl |
| --- | --- | --- | --- | --- |
| fdv01\_b |  | N/A | 44004 | 2753 |
| fdv01\_b |  | N/A | 51006 | 119 |
| fdv01\_b |  | N/A | 44006 | 3285 |
| fdv01\_b |  | N/A | 57003 | 2838 |
| fdv01\_b |  | N/A | 44005 | 218 |
| fdv01\_b |  | N/A | 57001 | 86 |
| fdv01\_b |  | N/A | 71006 | 4955 |
| fdv01\_b |  | N/A | 44002 | 2906 |
| fdv01\_b |  | N/A | 44003 | 84 |
| fdv01\_f |  | N/A | 44006 | 3285 |
| fdv01\_f |  | N/A | 71006 | 4955 |
| fdv01\_f |  | N/A | 51006 | 119 |
| fdv01\_f |  | N/A | 44005 | 218 |
| fdv01\_l |  | N/A | 44004 | 2753 |
| fdv01\_l |  | N/A | 57003 | 2838 |
| fdv01\_p |  | N/A | 57001 | 86 |
| geb01\_f | AX\_KommunalesGebiet | Municipal area | 75003 | 398 |
| geb01\_l | AX\_Gebietsgrenze | Territory boundary | 75009 | 1432 |
| geb01\_n | AX\_Bundesland | Federal state | 73002 | 1 |
| geb01\_n | AX\_Gemeinde | Community | 73005 | 396 |
| geb01\_n | AX\_KreisRegion | Circle region (??) | 73004 | 31 |
| geb01\_n | AX\_Regierungsbezirk | Administrative district | 73003 | 5 |
| geb02\_f | AX\_Insel | Island | 74004 | 91 |
| geb03\_f | AX\_NaturUmweltOderBodenschutzrecht | Nature, environment, or soil protection law | 71006 | 5033 |
| geb03\_f | AX\_Schutzzone | protection zone | 71012 | 3 |
| geb03\_f | AX\_SonstigesRecht | Miscellaneous Law | 71011 | 56 |
| geb03\_l | AX\_NaturUmweltOderBodenschutzrecht | Nature, environment, or soil protection law | 71006 | 760 |
| geb03\_p | AX\_Denkmalschutzrecht | monument protection law | 71009 | 2 |
| geb03\_p | AX\_NaturUmweltOderBodenschutzrecht | Nature, environment, or soil protection law | 71006 | 10962 |
| gew01\_f | AX\_Fliessgewaesser | running water | 44001 | 570 |
| gew01\_f | AX\_Hafenbecken | harbor basin | 44005 | 180 |
| gew01\_f | AX\_StehendesGewaesser | Stagnant water | 44006 | 19690 |
| gew01\_l | AX\_Gewaesserachse | water axis | 44004 | 369737 |
| gew02\_f | AX\_Gewaessermerkmal | water feature | 55001 | 62 |
| gew02\_p | AX\_Gewaessermerkmal | water feature | 55001 | 13786 |
| gew02\_p | AX\_Wasserspiegelhoehe | water level | 57001 | 3990 |
| gew03\_l | AX\_Gewaesserstationierungsachse | water stationing axis | 57003 | 6347 |
| hdu01\_b |  | N/A |  | 208167 |
| hdu01\_f |  | N/A |  | 15277 |
| hdu01\_l |  | N/A |  | 192664 |
| hdu01\_p |  | N/A |  | 226 |
| rel01\_f | AX\_FelsenFelsblockFelsnadel | rock boulder pinnacle | 61006 | 38 |
| rel01\_l | AX\_DammWallDeich | dam wall dike | 61003 | 1880 |
| rel01\_l | AX\_FelsenFelsblockFelsnadel | rock boulder pinnacle | 61006 | 14 |
| rel01\_p | AX\_FelsenFelsblockFelsnadel | rock boulder pinnacle | 61006 | 12 |
| rel01\_p | AX\_Hoehleneingang | cave entrance | 61005 | 132 |
| sie01\_f | AX\_Ortslage | location | 52001 | 6697 |
| sie02\_f | AX\_Bergbaubetrieb | mining operation | 41004 | 45 |
| sie02\_f | AX\_FlaecheBesondererFunktionalerPraegung | Area with a special functional character | 41007 | 17415 |
| sie02\_f | AX\_FlaecheGemischterNutzung | Mixed use area | 41006 | 113434 |
| sie02\_f | AX\_Friedhof | Cemetery | 41009 | 3326 |
| sie02\_f | AX\_Halde | stockpile | 41003 | 67 |
| sie02\_f | AX\_IndustrieUndGewerbeflaeche | Industry And Commercial Space | 41002 | 39323 |
| sie02\_f | AX\_SportFreizeitUndErholungsflaeche | Sports Leisure And Recreation Area | 41008 | 21806 |
| sie02\_f | AX\_TagebauGrubeSteinbruch | Open Pit Quarry | 41005 | 632 |
| sie02\_f | AX\_Wohnbauflaeche | residential area | 41001 | 200239 |
| sie03\_f | AX\_BauwerkOderAnlageFuerIndustrieUndGewerbe | Structure Or Plant For Industry And Trade | 51002 | 1869 |
| sie03\_f | AX\_BauwerkOderAnlageFuerSportFreizeitUndErholung | Structure Or Facility For Sport Leisure And Recreation | 51006 | 9181 |
| sie03\_f | AX\_HistorischesBauwerkOderHistorischeEinrichtung | Historical Building Or Establishment | 51007 | 54 |
| sie03\_f | AX\_SonstigesBauwerkOderSonstigeEinrichtung | Other structure or facility | 51009 | 2 |
| sie03\_f | AX\_VorratsbehaelterSpeicherbauwerk | reservoir storage building | 51003 | 1235 |
| sie03\_l | AX\_BauwerkOderAnlageFuerSportFreizeitUndErholung | Structure Or Facility For Sport Leisure And Recreation | 51006 | 127 |
| sie03\_l | AX\_HistorischesBauwerkOderHistorischeEinrichtung | Historical Building Or Establishment | 51007 | 1215 |
| sie03\_l | AX\_Leitung | Management | 51005 | 3513 |
| sie03\_l | AX\_SonstigesBauwerkOderSonstigeEinrichtung | Other structure or facility | 51009 | 3079 |
| sie03\_l | AX\_Transportanlage | transport system | 51004 | 616 |
| sie03\_p | AX\_BauwerkOderAnlageFuerIndustrieUndGewerbe | Structure Or Plant For Industry And Trade | 51002 | 37719 |
| sie03\_p | AX\_HistorischesBauwerkOderHistorischeEinrichtung | Historical Building Or Establishment | 51007 | 1704 |
| sie03\_p | AX\_SonstigesBauwerkOderSonstigeEinrichtung | Other structure or facility | 51009 | 16747 |
| sie03\_p | AX\_Transportanlage | transport system | 51004 | 42 |
| sie03\_p | AX\_VorratsbehaelterSpeicherbauwerk | reservoir storage building | 51003 | 1695 |
| sie04\_f | AX\_Hafen | Harbor | 52002 | 176 |
| sie04\_f | AX\_Schleuse | sluice | 52003 | 48 |
| sie05\_f | AX\_Turm | Tower | 51001 | 127 |
| sie05\_p | AX\_Turm | Tower | 51001 | 1309 |
| veg01\_f | AX\_Landwirtschaft | Agriculture | 43001 | 318334 |
| veg02\_f | AX\_Wald | Forest | 43002 | 216895 |
| veg03\_f | AX\_Gehoelz | Woody | 43003 | 90283 |
| veg03\_f | AX\_Heide | Moor or heather | 43004 | 1485 |
| veg03\_f | AX\_Moor | Moor or bog | 43005 | 179 |
| veg03\_f | AX\_Sumpf | Swamp | 43006 | 232 |
| veg03\_f | AX\_UnlandVegetationsloseFlaeche | Unland Area without vegetation | 43007 | 7607 |
| veg04\_f | AX\_Vegetationsmerkmal | Vegetation feature | 54001 | 7495 |
| veg04\_l | AX\_Vegetationsmerkmal | Vegetation feature | 54001 | 111494 |
| veg04\_p | AX\_Vegetationsmerkmal | Vegetation feature | 54001 | 11333 |
| ver01\_f | AX\_Platz | Place | 42009 | 7188 |
| ver01\_f | AX\_Strassenverkehr | Road traffic | 42001 | 41555 |
| ver01\_l | AX\_Fahrbahnachse | Lane axis | 42005 | 24461 |
| ver01\_l | AX\_Strassenachse | Road axis | 42003 | 844836 |
| ver02\_l | AX\_Fahrwegachse | Route axis | 42008 | 571520 |
| ver02\_l | AX\_WegPfadSteig | Way path slope | 53003 | 109695 |
| ver03\_f | AX\_Bahnverkehr | Rail traffic | 42010 | 9230 |
| ver03\_l | AX\_Bahnstrecke | Railway line | 42014 | 15376 |
| ver03\_l | AX\_SeilbahnSchwebebahn | Cable car suspension railway | 53005 | 116 |
| ver04\_f | AX\_Flugverkehr | Air traffic | 42015 | 228 |
| ver05\_f | AX\_Schiffsverkehr | Ship traffic | 42016 | 295 |
| ver05\_l | AX\_SchifffahrtslinieFaehrverkehr | Shipping line, ferry traffic | 57002 | 29 |
| ver06\_f | AX\_Bahnverkehrsanlage | Railway system | 53004 | 673 |
| ver06\_f | AX\_BauwerkImGewaesserbereich | Structure in the water area | 53009 | 190 |
| ver06\_f | AX\_BauwerkImVerkehrsbereich | Structure in the traffic area | 53001 | 3759 |
| ver06\_f | AX\_Flugverkehrsanlage | Air traffic facility | 53007 | 256 |
| ver06\_l | AX\_BauwerkImGewaesserbereich | Structure in the water area | 53009 | 148904 |
| ver06\_l | AX\_BauwerkImVerkehrsbereich | Structure in the traffic area | 53001 | 34771 |
| ver06\_l | AX\_Strassenverkehrsanlage | Road traffic installation | 53002 | 4 |
| ver06\_p | AX\_Bahnverkehrsanlage | Railway system | 53004 | 2632 |
| ver06\_p | AX\_BauwerkImVerkehrsbereich | Structure in the traffic area | 53001 | 39 |
| ver06\_p | AX\_EinrichtungenFuerDenSchiffsverkehr | Facilities for shipping | 53008 | 2580 |
| ver06\_p | AX\_Flugverkehrsanlage | Air traffic facility | 53007 | 305 |
| ver06\_p | AX\_Strassenverkehrsanlage | Road traffic installation | 53002 | 1969 |

The files we’re interested in are veg\_01, veg\_02, and veg\_03. The first step is to clip these shapefiles down to the Dortmund boundary with the Clip function. You can run this as a batch process, should go quickly. The easiest thing to do to combine these files is to field calculate a new field and then delete the rest so we can combine and rasterize them. Let’s follow this coding scheme:

Agriculture = 1, Forest = 2, Woody = 2, Moor or heather = 3, Unland Area without vegetation = 4

Moor or bog & Swamp do not appear in Dortmund so we don’t need to worry about them. Since Agriculture and Forest are their own files, we can use a really simple field calculation.



Since the Other file is more complicated, we can use this recoding scheme:

if("OBJART\_TXT" ='AX\_Gehoelz', 2,

if("OBJART\_TXT" = 'AX\_Heide', 3,

if("OBJART\_TXT" = 'AX\_UnlandVegetationsloseFlaeche', 4,0)))

Then use the attribute table to delete everything that isn’t “raster” (we only have to do this because the ag attribute table is different for some reason) and merge the shapefiles with the Merge Vector Layers tool. Finally, rasterize this later. Align it with the DTM so it works well for the next step.