GEM 510: GIS for Forestry and Conservation

Paul D. Pickell

2023-08-21

Contents

Welcome

These are the course materials for GEM 510 in the Master of Geomatics for Environmental Management program (MGEM) at the University of British Columbia (UBC). These Open Educational Resources (OER) were developed to foster the Geomatics Community of Practice that is hosted by the Faculty of Forestry at UBC.

These materials are primarily lab assignments that students enrolled in GEM 510 will complete and submit for credit in the program. Note that much of the data referenced are either public datasets or otherwise only available to students enrolled in the course for credit. Deliverables for these assignments are submitted through the UBC learning management system and only students enrolled in the course may submit these assignments for credit.

How to use these resources

Each "chapter" is a standalone lab assignment designed to be completed over one or two weeks.

Students enrolled in GEM 510 will submit all deliverables through the course management system at UBC for credit and should consult the schedule and deadlines posted there. The casual user can still complete the tutorials step-by-step, but the data that are not alreadyh publicly available are not hosted on this website and therefore you will not have access to them.

How to get involved

Because this is an open project, we highly encourage contributions from the community. The content is hosted on our GitHub repository and from there

you can open an issue or start a discussion. Feel free to open an issue for any typos, factual discrepancies, bugs, or topics you want to see. We are always looking for great Canadian case studies to share! You can also fork our GitHub repository to explore the source code and take the content offline.

Chapter 1

Introduction to PostgreSQL and PostGIS

Written by Paul Pickell

Lab Overview

In this lab, you will be introduced to a Free and Open Source Software (FOSS) and widely-used database management system known as PostgreSQL. You will learn how to set up your own database, connect to an enterprise database, and write Structured Query Language (SQL) queries both from the command line and a popular postgres server admin graphical user interface. The software used in this lab includes PostgreSQL, psql, pgAdmin, and QGIS.

Learning Objectives

- Distinguish between PostgreSQL server, PostGIS database, and layer views
- Practice executing SQL queries on an enterprise PostGIS database from various software interfaces
- Export layer views from a PostGIS database in QGIS and ArcGIS Pro

---

Deliverables

Create a simple map of UBC Vancouver campus using at least five layers from the ubcv database (30 points)

Try querying a layer or two and showing only a subset of the total features. Symbolize each layer however you want in either QGIS or ArcGIS Pro. The goal here is to practice changing symbologies and creating a professional layout. At a minimum, your map should have:

- At least five layers symbolized from the ubcv database
- Informative title based on the layers you chose (do not use "UBC Vancouver Campus" as your title)
- North arrow
- Scale bar or other representation of scale
- Your name and date
- Projected in NAD83(CSRS) Canada Atlas Lambert (EPSG:3979)

Data

All data for this lab are accessible via the UBC PostgreSQL server. Instructions for connecting to the server are given in the tasks below.

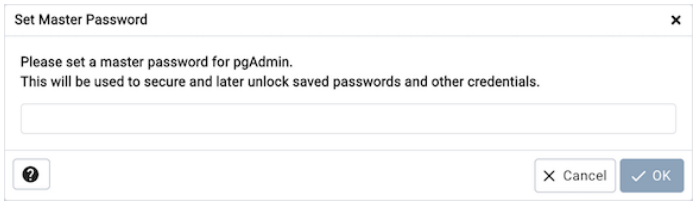
Task 1: Set up pgAdmin and Connect to the UBC PostgreSQL server

Step 1: Ensure that you are authenticated through UBC myVPN. If you are at UBC, use your dedicated ethernet port or connect via the ubcprivate wireless network. If you are away from campus, you will need to first connect to the UBC myVPN service using Cisco AnyConnect Secure Mobility Client. Only authenticated users with a Campus Wide Login (CWL) and connected to the UBC myVPN may access the UBC PostgreSQL server.

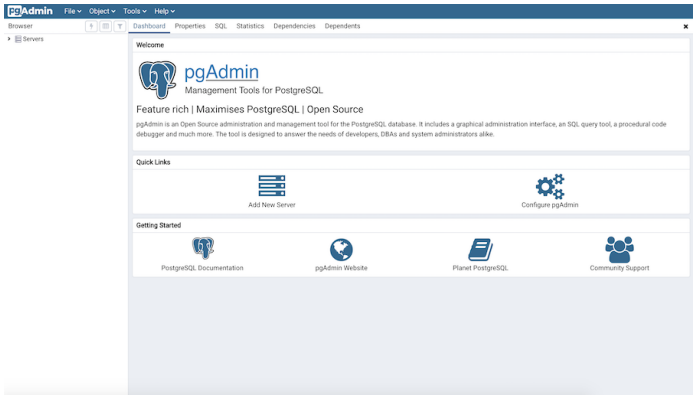
Step 2: Start pgAdmin 4. This software is a management tool for PostgreSQL databases, hence "pg".

Step 3: Upon starting pgAdmin, you will be prompted to set a master password. This is an important step as this master password will be used to encrypt

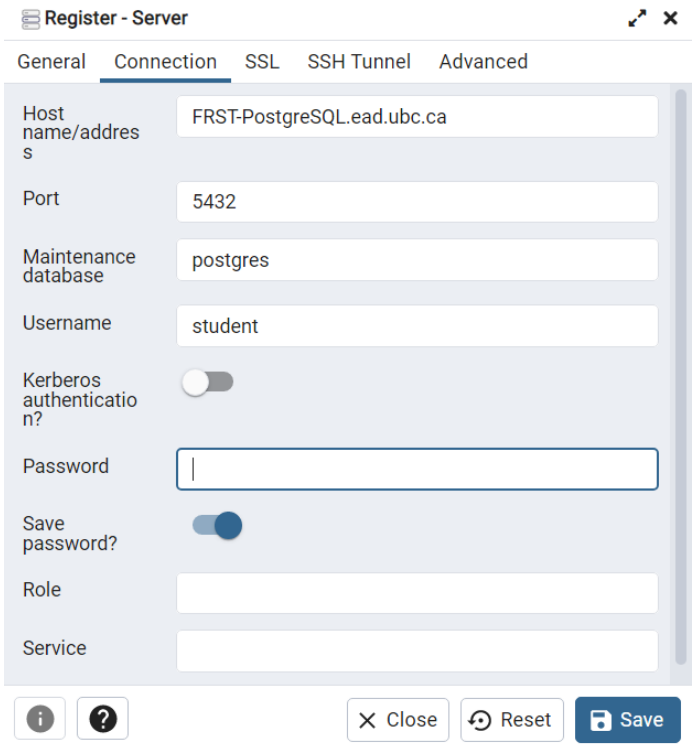
all other passwords you use to connect to various server databases. DO NOT USE THE SAME PASSWORD AS YOU WILL USE FOR ANY DATABASE!



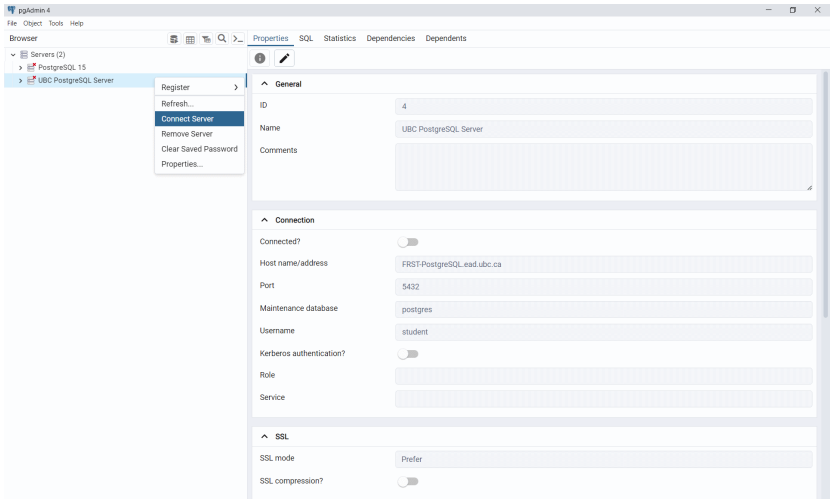
Step 4: From the pgAdmin dashboard, click "Add New Server".



Step 5: Name the connection "UBC PostgreSQL Server". Click the "Connection" tab, enter FRST-PostgreSQL.ead.ubc.ca for the Host Name, change the Username to "student", and then enter the password that has been shared with the class. Leave everything else as the default, but you can choose to save the password if you want. Click "Save". If the password is correct, then pgAdmin should automatically connect to the server and you will see it listed in the Browser, just expand "Servers".



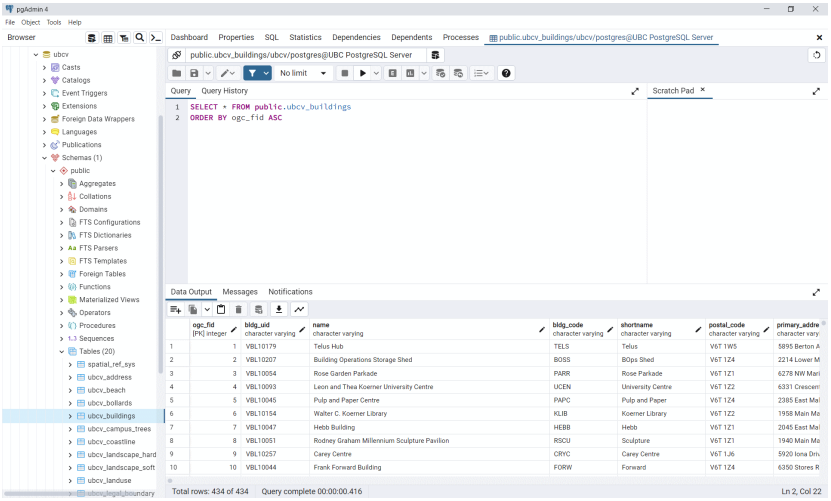
Although the connection was automatic this time, each time you start pgAdmin, you will need to reconnect to the server after entering your master password. This is as simple as double-clicking the server name from the server list or right-click and select "Connect Server".



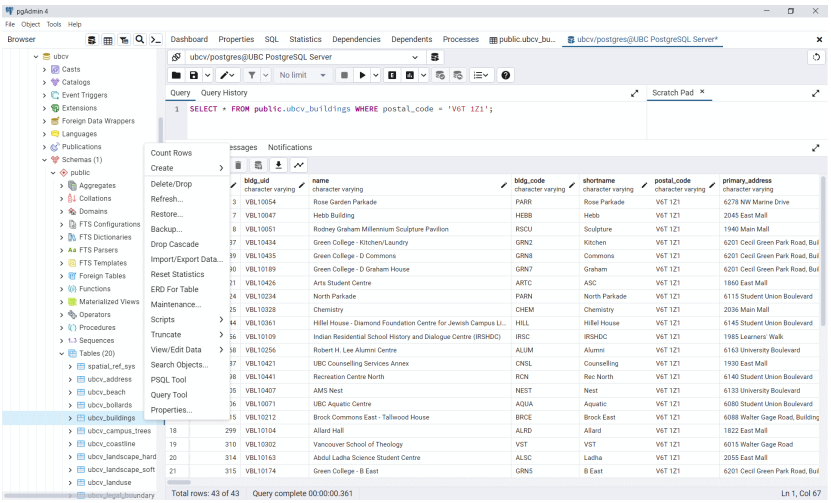
Step 6: From the Browser pane on the left, expand the ubcv connection, then

expand “Schemas”, then expand “public”, then expand “Tables” to view all the tables in the ubcv database.

Step 7: Right-click on a table and then select “View/Edit Data” > “All Rows”. pgAdmin will create an SQL query and then show the result. This is one way to view the attribute table from pgAdmin.

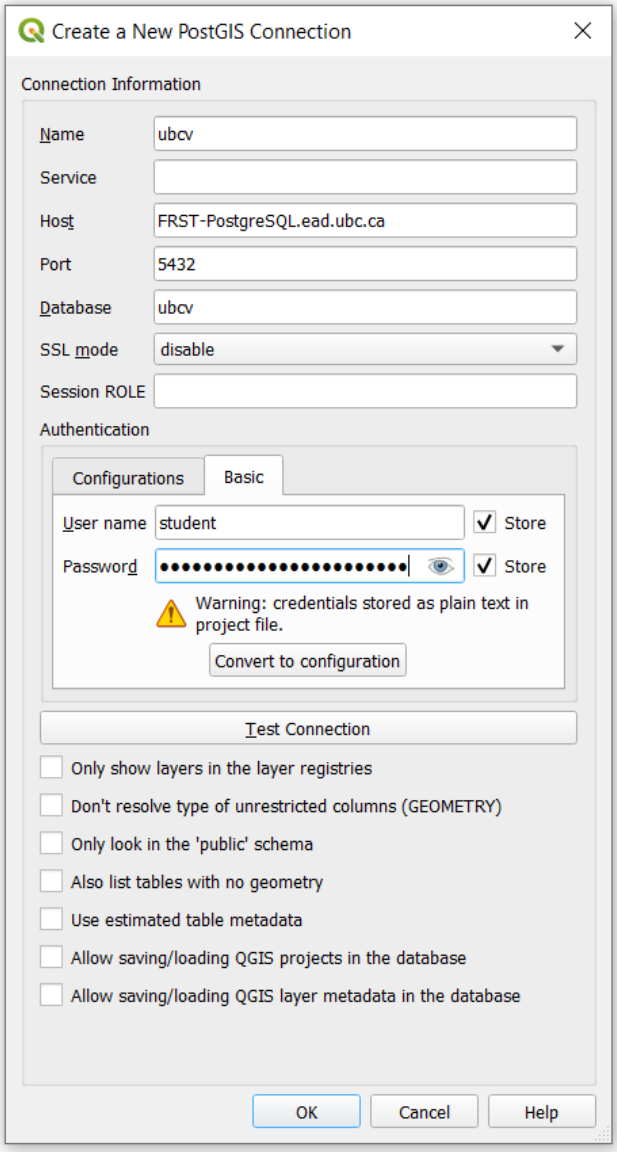


Step 8: You can also execute your own SQL query. Right-click on a table and select “Query Tool”. At the top, you can write any SQL query you want then click the Run button that looks like a triangle from the top ribbon. pgAdmin will return the resulting table below.



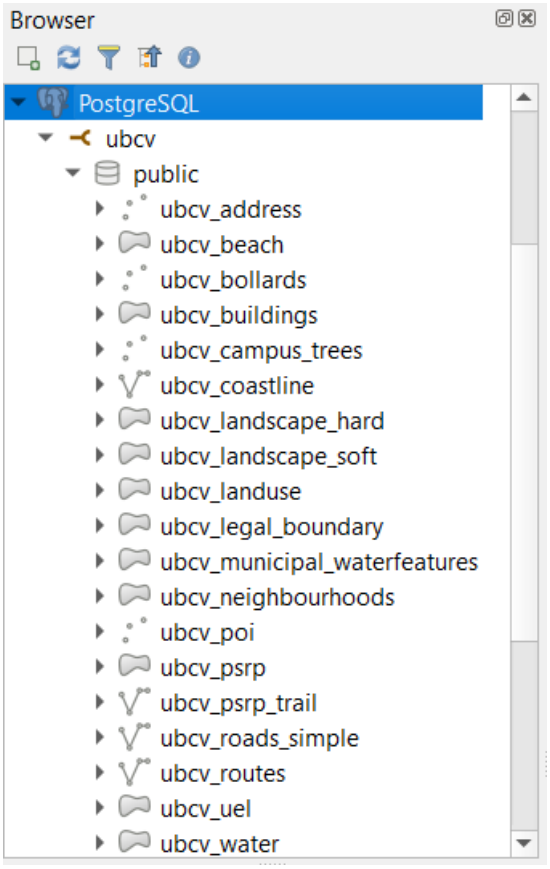
Task 2: Explore with QGIS

Step 1: Open QGIS. From the Browser pane on the left, right-click "PostgreSQL" and select "New Connection".



Step 2: In your Browser pane, you should now see the ubcv database connection. Expand it to see the available public schema, then expand that to see

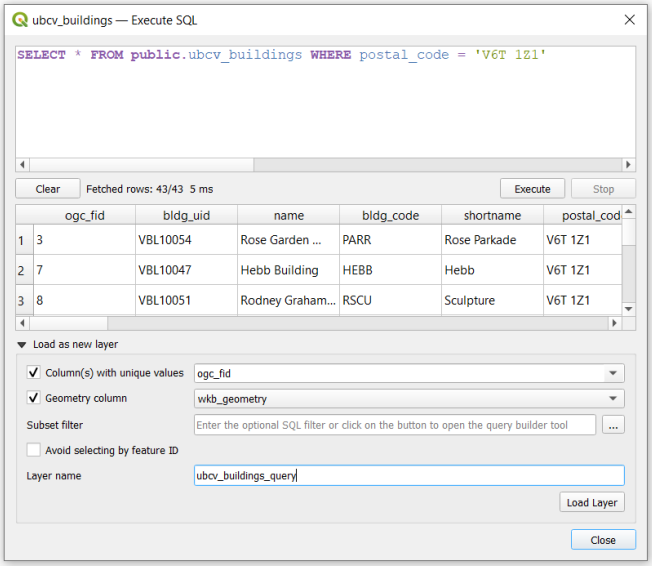
all available tables that have geometries. You can double-click any of these or click-and-drag into your empty project to view them.



Step 3: Spend some time exploring the data. Open the attribute tables. View the properties and metadata. What is the EPSG code for the Coordinate Reference System (CRS) for these data?

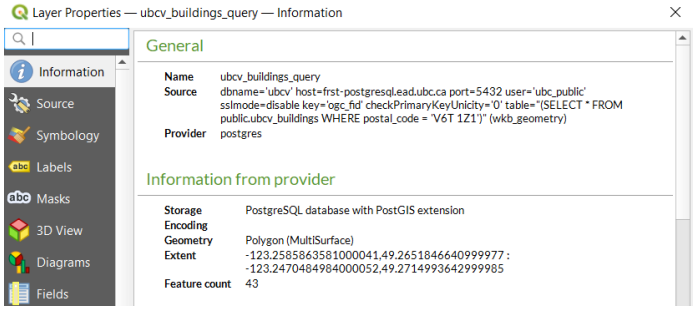
At this point, keep in mind that all the data still live on the postgres server. You may have already seen the other databases on the server. If you want to view the data in those databases, you will need to create a new database connection for each database that you want to connect to. Feel free to explore the data in the other databases, noting that what is available at any given time may be different from what you see in the screenshots in this lab. The UBC PostgreSQL server hosts many databases that are used for research and teaching!

Step 4: You can also execute SQL queries in QGIS directly from the PostGIS database. From the Browser pane, right-click any available table in the PostGIS database and then select "Execute SQL…". In the dialogue window that opens, you will see a default SQL query that will return the first 10 features (tuples) of the selected relation.



Step 5: Write an SQL query to return some subset of the data then click the "Execute" button. QGIS will return a preview of the attribute table for the query. From here, you can export this as a layer to view the result in the map. Expand "Load as a new layer" and then toggle on "Column(s) with unique values" and ensure the field is set to "ogc_fid", this is the primary key for the table in PostGIS. Toggle on "Geometry column" and ensure the field is set to "wbk_geometry", this is the "Well Known Binary" format used to represent features in PostGIS. Finally, change the "Layer name" to something to distinguish your query from others and then click "Load layer" and then "Close". The result of the SQL query should now appear as a layer in your QGIS map.

Note that the layer you just created does not actually exist outside your QGIS project. If you navigate into the properties of this layer, you will see the Source is simply a reference to the data that are still stored on the postgres server. This can be a really efficient way to handle data without unnecessarily copying it to a new file!

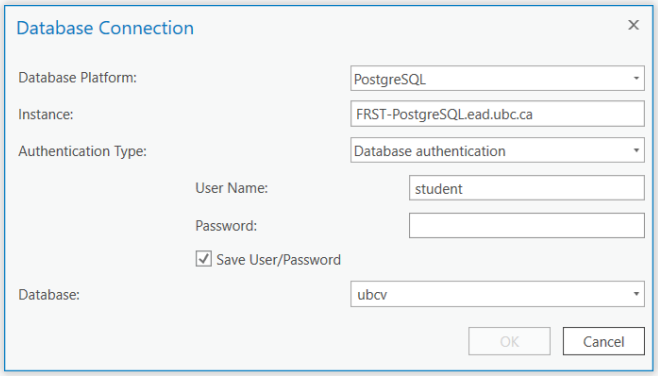


Step 5: You can export any table in the PostGIS database to your local computer by right-clicking the table name in the PostGIS database and then selecting "Export Layer" > "To File...". From the new dialogue window, you can configure the output file however you want. You can do the same for the query layer by right-clicking the query layer created in Step 5 and selecting "Export" > "Save Features As...".

Task 3: Explore with ArcGIS Pro

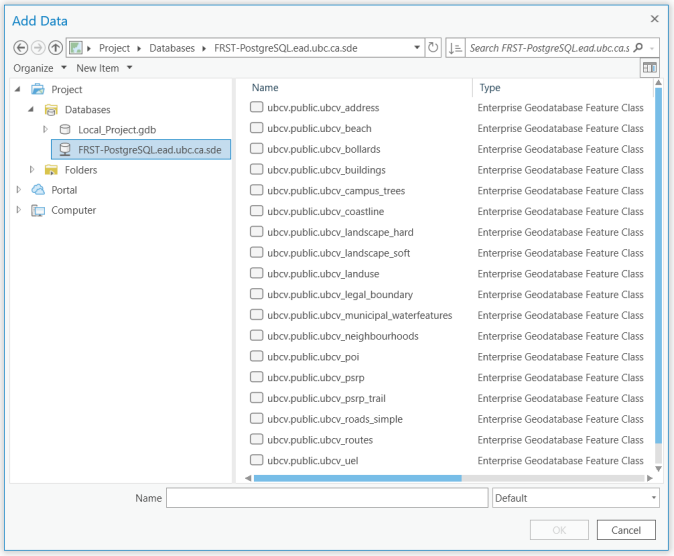
Step 1: Start ArcGIS Pro and create a new project.

Step 2: From the top ribbon, navigate to the "Insert" tab, then click "Connections", and from the drop-down menu, select "New Database Connection". Enter the database connection details in the same way that you did in the prior tasks. Once you have correctly entered the correct credentials, ArcGIS Pro will do a "soft" connection to the server to retrieve the available databases. From the "Database" drop-down menu, select "ubcv" and then click "OK".

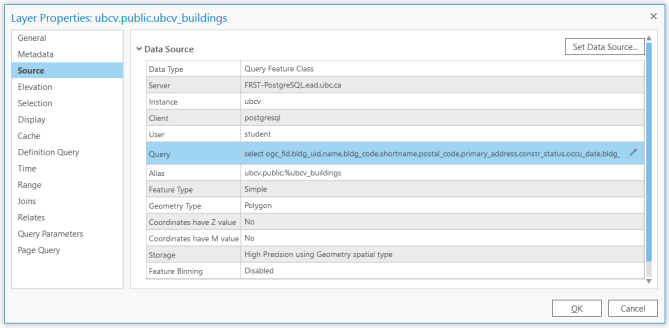


Step 3: From the top ribbon, navigate to the "Map" tab, then click "Add Data". From the dialogue window that opens, expand "Databases", and you will see

your local project geodatabase that was created when you initiated the project and the newly added postgres server. Click the postgres server connection to view the tables and then add a layer to your map.

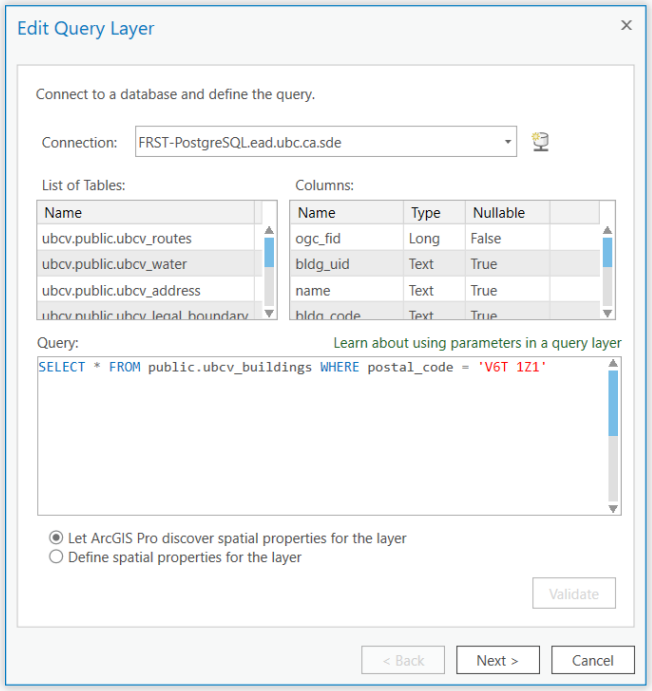


Step 4: Once you have added a layer to your map, right-click it in your Contents pane and open the properties. Again, like the query layer you created in QGIS, you will see this layer is a dynamic reference to the data on the PostGIS database.



Step 5: Under Data Source, there is a Query field that includes a generic SQL query statement. Click the pencil icon on the right to open the Edit Query Layer dialogue window. This window is similar to the qeury window you used in QGIS. Enter the same query you used in Task 2 then click “Validate”. If the query is valid, ArcGIS Pro will allow you to click “Next”. Leave the “Let ArcGIS Pro discover spatial properties for this layer” toggled on. The next window will allow you to select the Unique Identifier Field(s), just like in QGIS. This should automatically have selected “ogc_fid” and automatically identified

the geometry type, so you can leave everything as-is and click “Finish”.



You will be returned to the Properties dialogue of the layer you added in your map, but you should see that the Query field has now been updated. Click “OK” to exit the properties and the layer in the map should now only be showing the features that match your query.

Task 4: Explore with psql Shell

Step 1: From your Windows search bar, search for “psql” and open the SQL Shell (psql) application. This will open a command prompt and allow you to interact with the PostgreSQL server directly using mostly SQL statements.

The prompt should say `Server [localhost]:`. Type “FRST-PostgreSQL.ead.ubc.ca” and then hit “Enter” on your keyboard.

The prompt should then say `Database [postgres]:`. Type “ubcv” and then press “Enter” on your keyboard.

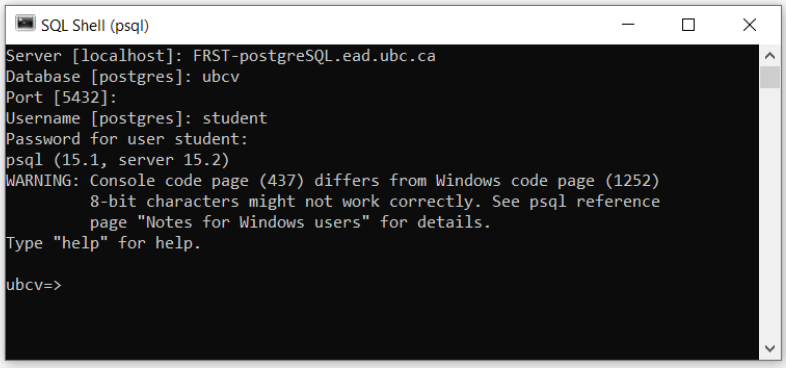
The prompt should then say `Port [5432]:`. This is the default port that the PostgreSQL server uses. Press “Enter” on your keyboard to accept the default

port.

The prompt should then say `Username [postgres]:`. Type "student" and then press "Enter" on your keyboard.

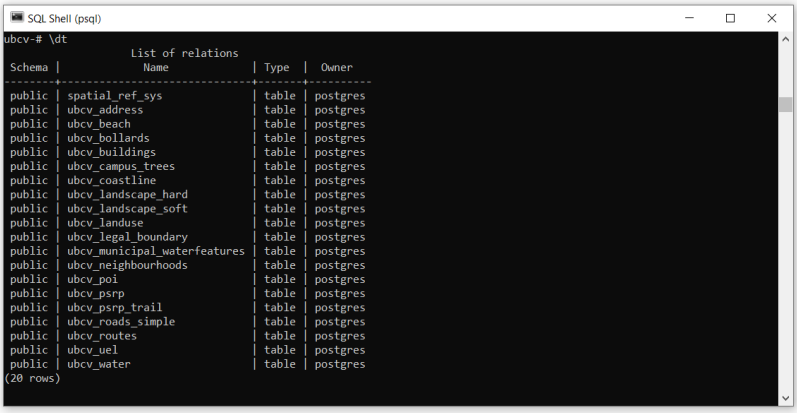
The prompt should then ask you for the password, `Password for user student:`. Type the password that was shared with the class and then press "Enter" on your keyboard.

If everything is successful, then you should see `ubcv=>`, which indicates you can now enter either psql commands or SQL statements. (See screenshot below)

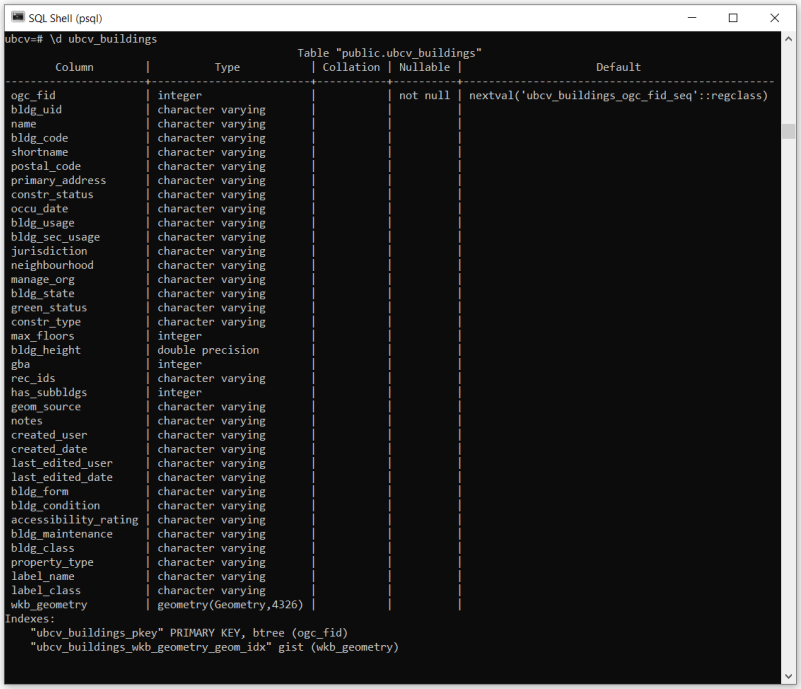


Step 2: List all the available databases on the postgres server with the `\l` command. If your psql window is too small, you might see `-- More --`. Just continue to press "Enter" to list more tables. You can change to any of these databases by using the `\c [database name]` command like `\c postgres`.

Step 3: List the tables in the ubcv database with the `\dt` command. You can view a full list of psql commands and their useage with the `\?` command.

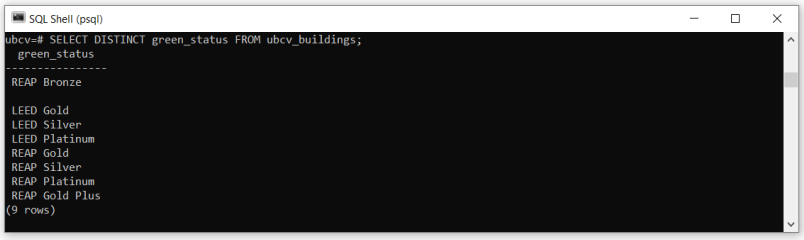


Step 4: List all the fields for any table with the `\d [table name]` command. For example, `\d ubcv_buildings`.

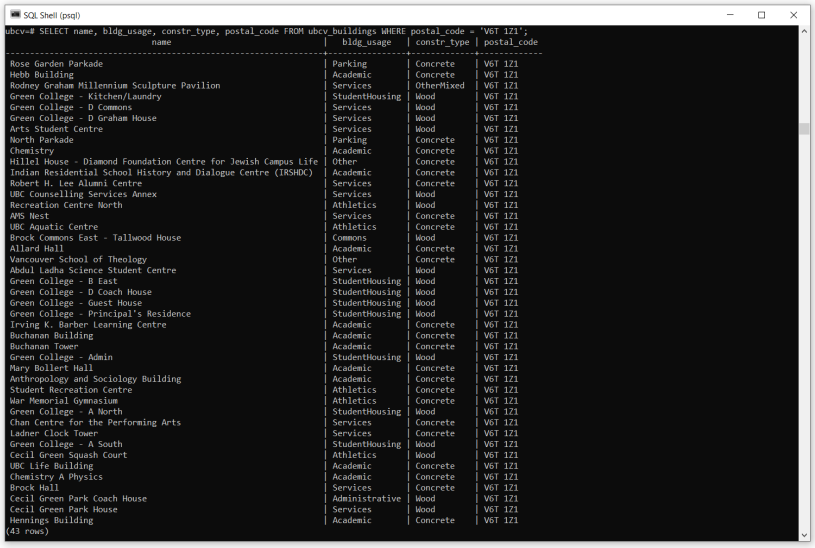


Step 5: In addition to using psql commands you can also execute SQL statements directly in the console. For example, type `SELECT COUNT(*) FROM ubcv_buildings;` to get a count of all the rows in a table. Notice that `SELECT` statements return tables, the table below has one row and one field.

Step 6: You can get the unique values of any field by using the `DISTINCT` keyword. For example, `SELECT DISTINCT green_status FROM ubcv_buildings;`. Note that in this returned table, the blank space below "REAP Bronze" is a valid empty value for this field.



Step 7: You can execute the SQL query like you did in the previous tasks, but the psql console is not ideal for wielding large tables with many fields and the formatting will not be very readable. So it is best to return only the fields that you actually need to inspect. For example, `SELECT name, bldg_usage, constr_type, postal_code FROM ubcv_buildings WHERE postal_code = 'V6T 1Z1';`.



Some useful psql commands:

- `\l` List all databases on the current PostgreSQL server
- `\c` [database name] Connect or switch to any database on the PostgreSQL server
- `\dt` List tables in the current database
- `\?` List all psql commands
- `\q` Quit the SQL Shell

Summary

You should now have a working knowledge of PostGIS databases, PostgreSQL servers, and how to connect and manage them from pgAdmin, QGIS, and ArcGIS Pro. You will continue to practice and extend these skills as nearly all data used in later labs will be accessed via the UBC PostgreSQL server.

Return to the **Deliverables** section to check off everything you need to submit for credit in the course management system.

Chapter 2

Analyze a Relational Database

Written by Paul Pickell

Lab Overview

In this lab you will learn how to create, manage, and analyze your own relational database. You will continue to practice with the tools that you learned in the prior lab: pgAdmin, QGIS, ArcGIS Pro, and psql. You will also be introduced to the Geospatial Data Abstraction Library (GDAL) and some of the handy programs that are available from this software library for managing data. You will learn more advanced Structured Query Language (SQL) statements that will allow you join tables, insert and update data, and analyze data in a relational database.

Learning Objectives

- Create and host your own PostgreSQL server with a PostGIS database
- Apply best practices for handling, organizing, and managing data
- Import geospatial data using the Geospatial Data Abstraction Library (GDAL)
- Practice joining and relating tables
- Practice inserting and updating data
- Analyze geospatial data in a relational database using SQL and PostGIS

Deliverables

Dump SQL file from Task 2 (15 points)

SQL statement (just the text) from Task 4 (15 points)

Data

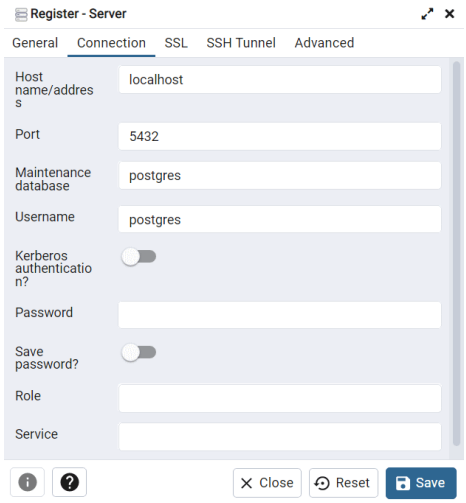
All data for this lab are accessible via the UBC PostgreSQL server. Instructions for connecting to the server are given in the tasks below. We will be using data from the `ubcv` database.

Task 1: Host your own PostgreSQL Server

Step 1: Open pgAdmin.

Step 2: From the **Browser** pane on the left, right-click **Servers** and then select **Register** then **Server...**. We are going to add another server, but this time, it will be hosted on your local machine.

Step 3: Name the server **localhost** then click the **Connection** tab and set the **Host Name/Address** also as **localhost**. Enter a password for the server and leave everything else as the default. When finished, click **Save**. The new localhost server should now appear in your Browser. You can expand it, navigate through **Databases**, **postgres**, **Schemas**, **public**, and **Tables** to find that it is indeed empty.



Step 4: Open the psql shell.

If you are already connected to the UBC PostgreSQL server, you can switch this connection to the localhost server you just made in pgAdmin with the following command: `\c "dbname=postgres host=localhost port=5432 user=postgres"`. You will then be prompted to enter your password.

Alternatively, you can open a new psql terminal window. The prompt should say `Server [localhost]:`. Press "Enter" on your keyboard to accept localhost as the server host name.

The prompt should then say `Database [postgres]:`. This is the default name of the database that was created when you hosted the server from pgAdmin. Press "Enter" on your keyboard to accept postgres as the database to connect to.

The prompt should then say `Port [5432]:`. This is the default port that the PostgreSQL server uses. Press "Enter" on your keyboard to accept the default port.

The prompt should then say `Username [postgres]:`. This is the default username that you created when you set up the PostgreSQL server in pgAdmin. Press "Enter" on your keyboard to accept the postgres username.

The prompt should then ask you for your password, `Password for user postgres:`. Type your password and then press "Enter" on your keyboard.

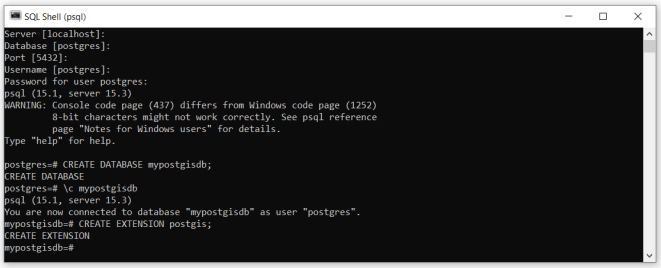
If everything is successful, then you should see `postgres=#`, which indicates you can now enter either psql commands or SQL statements. Note that the hashtag `#` indicates that you are connected to the current database with the "SUPERUSER" role, which is essentially the highest admin privilege.

Task 2: Create a PostGIS Database

Step 1: Create a new database called "mypostgisdb" with the following SQL command `CREATE DATABASE mypostgisdb;`. It is important to always escape SQL statements with a semi-colon otherwise psql will interpret your input as spanning multiple lines! If you forget the semi-colon, you can always just type it in the console and hit "Enter" on your keyboard and psql will interpret this as a two-line statement.

Step 2: Connect to your new database by using the psql command `\c mypostgisdb`. Note that this is NOT an SQL statement, so there is no need to escape the command with a semi-colon. Your console should now say `mypostgisdb=#`, which indicates you are now connected.

Step 3: Currently this is just a plain-vanilla PostgreSQL relational database that cannot handle spatial data. In order to convert this to a PostGIS database, we need to enable the PostGIS extension using the following SQL: `CREATE EXTENSION postgis;`.



Step 4: Listing the tables `\dt` will reveal the database has a table called "spatial_ref_sys". List the fields of this table with `\d spatial_ref_sys`.

In addition to the field names, you will see there is an index called "spatial_ref_sys_pkey", which is the dedicated field for the primary key for this table, and a constraint check to ensure that srid's are valid only between 1 and 998999. You can test this constraint by trying to insert a new srid value of 0: `INSERT INTO spatial_ref_sys (srid) VALUES (0);`. The returned message,

```
ERROR:  new row for relation "spatial_ref_sys" violates check constraint spatial_ref_sy
DETAIL:  Failing row contains (0, null, null, null, null).
```

DO NOT modify this table with any srids in the valid range otherwise you will need to recreate the PostGIS database!

Step 5: Return the first ten rows of the table for the fields "auth_name" and "auth_srid" with `SELECT auth_name,auth_srid FROM spatial_ref_sys LIMIT 10;`.

Table 2.1: Some PostgreSQL data types with example value ranges.

Name	Description	Values
int2	Signed 2-byte integer	-32,768 t
int4	Signed 4-byte integer	-2,147,48
int8	Signed 8-byte integer	-9,223,37
bigserial	Autoincrementing 8-byte integer	0 to 18,4
float4	Single prevision floating-point number (4 bytes)	$\pm 10^3$ wi
float8	Double precision floating-point number (8 bytes)	$\pm 10^3$ w
bool	Logical Boolean	True or I
char(n)	Fixed length character string where n is the number of permitted characters	'Hello wo
varchar(n)	Variable length character string where n is the maximum number of permitted characters	'Hello wo
date	Calendar date	31/12/20

This table contains EPSG codes, which are used to easily relate spheroids, datums, and measurement units to geospatial data. You will find these codes used everywhere when you look at metadata and the properties of different data layers. One very commonly used code is EPSG 4326, which references the WGS 1984 datum.

Step 6: Write an SQL query to return the proj4text of the WGS 1984 datum from the spatial_ref_sys table.

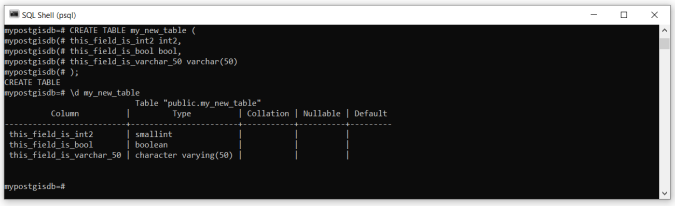
Now we will create some new data, just to show you how to use some useful SQL keywords. We will start with aspatial data first, then move on to spatial data. You can create a new table in your database with the syntax

```
CREATE TABLE table-name (
colunm-name-1 datatype,
colunm-name-2 datatype,
colunm-name-3 datatype,
...
);
```

Data types are very important and the shorthand notation for these in PostgreSQL are listed on this web page and some commonly used data types are reproduced below with examples.

Here is an example of creating a new table with some different data types:

```
CREATE TABLE my_new_table (
this_field_is_int2 int2,
this_field_is_bool bool,
this_field_is_varchar_50 varchar(50)
);
```



You can experiment with creating as many new tables as you want. If you need to delete a table, use the `DROP` keyword like this `DROP TABLE my_new_table;`.

It is good practice to assign a field (multiple fields) as a primary key when you create a table. This is done by simply adding `PRIMARY KEY` after the field definition when you make the table. For example:

```
CREATE TABLE my_new_table (
this_field_is_bigserial bigserial PRIMARY KEY,
this_field_is_int2 int2,
this_field_is_bool bool,
this_field_is_varchar_50 varchar(50)
);
```

The bigserial data type is especially useful for this purpose because it auto-increments as you add rows and can accommodate more than 18 *quintrillion* rows, that is more than 18 million trillions! You can also create composite primary keys that are comprised of two or more fields to uniquely identify all rows:

```
CREATE TABLE my_new_table (
this_field_is_int2 int2,
this_field_is_bool bool,
this_field_is_varchar_50 varchar(50),
PRIMARY KEY(this_field_is_int2, this_field_is_bool)
);
```

Step 7: Create a new table of assignments that are due next week. Include course code, assignment name, the percent weighting of the assignment on your final grade in that course, and the due date. Use the appropriate data types for each field and define a primary key.

Now that the table and fields are defined, we will insert some data into the table. Inserting data uses the `INSERT` keyword followed by `VALUES` and then a comma-separated list of the values you want to insert in parentheses. For example:

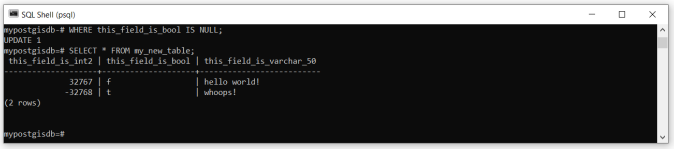
```
INSERT INTO my_new_table VALUES (32767, false, 'hello world!');
```

If you want to insert a value for specific fields, then specify the field name(s) after the table name:

```
INSERT INTO my_new_table (this_field_is_int2, this_field_is_varchar_50) VALUES (-32768,'hi again!
```

If you make a mistake or need to update a field later, then you use `UPDATE` and `SET` to identify the set of fields that should be updated. You can also test for NULL (empty values) using `IS NULL` or `IS NOT NULL`:

```
UPDATE my_new_table
SET this_field_is_bool = true, this_field_is_varchar_50 = 'whoops!'
WHERE this_field_is_bool IS NULL;
```



Dates require special handling because if you just try to insert them as strings or otherwise, they are treated as literals. For dates, we need to use the special `TO_DATE` function:

```
INSERT INTO my_new_table (this_field_is_date) VALUES (TO_DATE('31-12-1963', 'DD-MM-YYYY'));
```

Step 8: Now fill in your table of assignments by inserting and/or updating the values as needed.

Once you are satisfied with the state of your table, you will dump your entire PostGIS database to an output SQL file and this will be one of your deliverables for this lab. PostgreSQL features a utility program called `pg_dump` that will take any database and output an SQL statement in a file that can be used to re-create the database. In other words, `pg_dump` is a backup method. This is also

really handy if you want to create a local backup of a remote server! If you want more practice, try using it on the UBC PostgreSQL server.

Step 9: Open a Windows command prompt (search "command"). Take note of your current working directory, which is probably something like `C:\Users\[your username]`. This is where your database SQL file will be saved to. To dump your database, use the following command: `pg_dump -d mypostgisdb -U postgres -h localhost > mypostgisdb.sql`. You will be prompted to enter your password and then the new file `mypostgisdb.sql` will be saved in your working directory.

Step 10: Check to make sure that you can load the database back into PostgreSQL. In psql, create a new empty database named mypostgisdb_backup. Then, enter the following command from the Windows command prompt: `psql -U postgres -h localhost mypostgisdb_backup < mypostgisdb.sql`. You can now query the postgisdb_backup database to check that it is a copy of your other mypostgisdb database.

Task 3: Import Spatial Data Using GDAL

Now that you have some basic understanding of viewing and manipulating tables, we are going to look at some ways to import spatial data into your PostGIS database. To do this, we will be working with some utility programs in the Geospatial Data Abstraction Library (GDAL) pronounced "gee-dall". You will probably find the GDAL documentation pages very helpful for reference. GDAL is used in ArcGIS, GRASS, SAGA, QGIS, ENVI, Google Earth, and also has Python bindings and an R package with bindings. In short, GDAL is largely the workhorse behind most open source geospatial software packages.

Step 1: Open the OSGeo4W shell by searching for it in Windows. Enter the command `o-help` and inspect all the available programs.

You might notice `psql` can be run from the OSGeo4W shell and `pg_dump` is also there. Some others we will cover in later labs, including `pdal` (Point Data Abstraction Library) pronounced "poodle", which handles LiDAR data, and many of the raster and image manipulation programs. For this lab, we are going to focus on two commonly used vector programs: ogrinfo and ogr2ogr.

Step 2: Type `ogr2ogr --help` in the console and press "Enter" on your keyboard. Inspect all the flags - and arguments for this program.

ogr2ogr is the primary utility program for working with vector data and as you can see, there are a lot of options! To show you some of the power of this little application, we are going to export some data from the UBC PostgreSQL server

to our local machine, manipulate it, and then import it to our local PostGIS database.

Step 3: ogr2ogr is really useful for converting data between different formats and this is handy for exporting data out of a PostGIS database. Convert the ubcv_campus_trees table from the UBC PostgreSQL server with the following command:

```
ogr2ogr -f "ESRI Shapefile" ubcv_campus_trees.shp PG:"host=FRST-PostgreSQL.ead.ubc.ca user=studer
```

Replace `STUDENT_PASSWORD_HERE` with the password that has been shared with the class.

This command tells ogr2ogr to connect to the ubcv PostGIS database on the PostgreSQL server `PG:"host=FRST-PostgreSQL.ead.ubc.ca user=student dbname=ubcv password=STUDENT_PASSWORD_HERE"`, grab the table named `"ubcv_campus_trees"`, and then return it to the local machine as an ESRI Shapefile `-f "ESRI Shapefile"`. The file will be saved in your current working directory from the OSGeo4W shell.

Step 4: You can also specify an SQL query on the input data. For example, the following command will only return the Forest Sciences Centre polygon from ubcv_buildings:

```
ogr2ogr -where bldg_code='FSC' -f "ESRI Shapefile" ubcv_FSC.shp PG:"host=FRST-PostgreSQL.ead.ubc.
```

You can write more complex SQL statements with the `-sql` flag instead of `-where`.

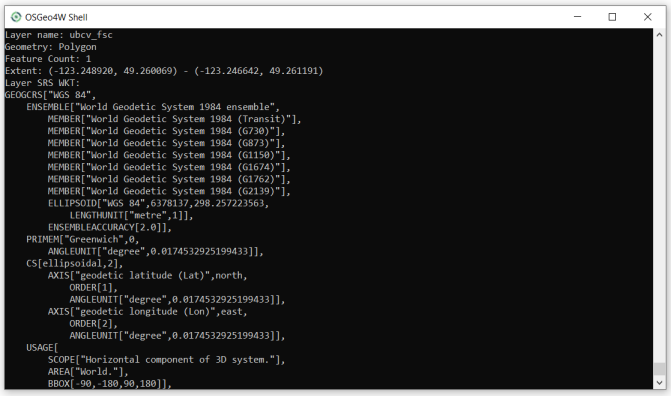
Step 5: Getting data into a PostGIS database is unremarkably similar as to taking it out. Import the FSC polygon into your **localhost PostGIS database** using the same command as Step 3, but change `"ESRI Shapefile"` to `"PostgreSQL"`, delete `ubcv_campus_trees.shp`, modify the connection parameters for your localhost server, change `"ubcv_campus_trees"` to `ubcv_FSC.shp`, and finally specify that we want these data reprojected into UTM Zone 10N coordinates `-t_srs EPSG:3157`. You will receive no feedback in the terminal if you were successful, but you can check in QGIS, ArcGIS Pro, or by using the ogrinfo program.

ogrinfo is a utility that is able to read a wide-variety of spatial data formats and return metadata for inspection. This program is useful for getting the geometry type of a file, the spatial extent of the features, and the coordinate system or the datum. You can also apply SQL queries and even inspect the attributes of specific features.

Step 6: Check that the FSC polygon was successfully added to your PostGIS database with the following command:

```
ogrinfo PG:"host=localhost user=postgres dbname=mypostgisdb password=YOUR_LOCALHOST_PAS
```

The screenshot below is a small snippet of what is returned in the OSGeo4W terminal window. As you can see, we have access to the feature count, spatial extent, Spatial Reference System (SRS), and even all the field names and data types if you scroll farther down. The `-so` flag gives us this "summary output".



Step 7: Repeat this process to import the ubcv_campus_trees.shp you created in Step 3 to your localhost PostGIS database.

Task 4: Analyze a PostGIS Database using SQL

Now let us have some fun and explore some of the cool spatial processing features of our PostGIS database. PostGIS supports an unfathomable number of spatial operations that you would ordinarily find in GIS software like QGIS and ArcGIS. This task is merely to give you a taste of what is possible with SQL and PostGIS.

Suppose we want to know what species of trees are planted next to the Forest Sciences Centre at UBC.

Step 1: Open psql and connect to your mypostgisdb database on your localhost PostgreSQL server.

Step 2: Buffer the ubcv_fsc polygon with the following SQL statement:

```
SELECT ST_Buffer(wkb_geometry, 15) INTO ubcv_fsc_buffer FROM ubcv_fsc;
```

This statement tells PostGIS to use the buffer function `ST_Buffer`, which requires a geometry field `wkb_geometry` and a buffer distance `15` (meters in our

case, because we re-projected the polygon to UTM Zone 10N in the last task), on the FSC polygon `FROM ubcv_fsc`, and then write the resulting buffer to a new table `INTO ubcv_fsc_buffer`.

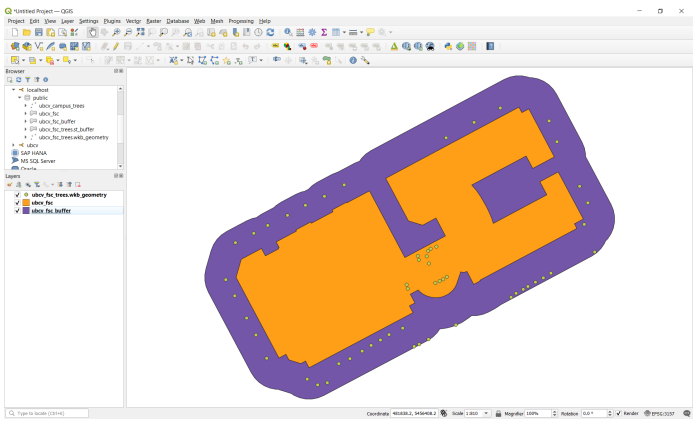
Step 3: Intersect the ubcv_campus_trees with the ubcv_fsc_buffer polygon you just made with the following SQL statement:

```
SELECT * INTO ubcv_fsc_trees FROM ubcv_campus_trees, ubcv_fsc_buffer WHERE ST_Intersects(wkb_geom
```

The `ST_Intersects` takes two geometry fields `wkb_geometry` (ubcv_campus_trees) and `st_buffer` (ubcv_fsc_buffer), computes the intersection, and returns the features from the first geometry argument (ubcv_campus_trees) to a new table `INTO ubcv_fsc_trees`.

If you get an error here like this, `ERROR: ST_Intersects: Operation on mixed SRID geometries (Point, 4326) != (Polygon, 3157)` then you know that you have not correctly changed the projection of the ubcv_campus_trees data when you imported it into your localhost postgres database. Go back to the last task and check Steps 5-7.

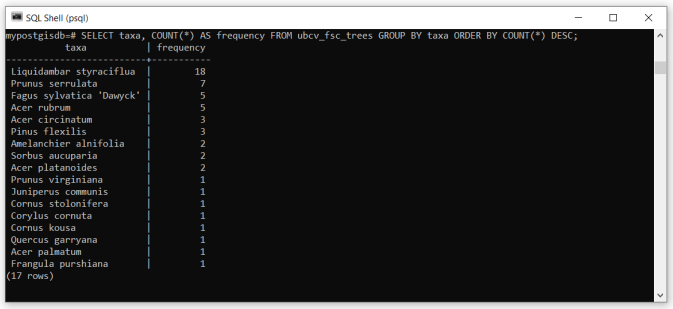
Step 4: Open QGIS and then add ubcv_fsc, ubc_fsc_buffer, and ubcv_fsc_trees.wkb_geometry to your map to inspect them.



Step 5: Now we will do a simple SQL query to determine the species that are planted around the Forest Sciences Building:

```
SELECT taxa, COUNT(*) AS frequency FROM ubcv_fsc_trees GROUP BY taxa ORDER BY COUNT(*) DESC;
```

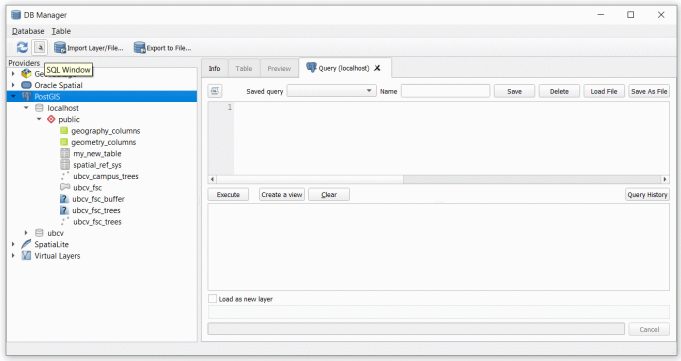
This will return the "taxa" column and create a new column called "frequency" that contains values of the `COUNT()` function. The `GROUP BY` statement is needed when we use an aggregation method like `COUNT()`. Finally, the relation is ordered by the aggregation result in descending `DESC` order.



Step 6: Modify the statement above to sort the taxa alphabetically.

These last few steps are just to illustrate that you can run the same SQL statements above directly in QGIS.

Step 7: Open QGIS. From the top menu toolbar, select **Database** then **DB Manager...** This will open a new window where you can do basically everything you did in this task and the prior task. You can import/export data to whatever format you want, inspect the database, tables, and even preview the spatial features.



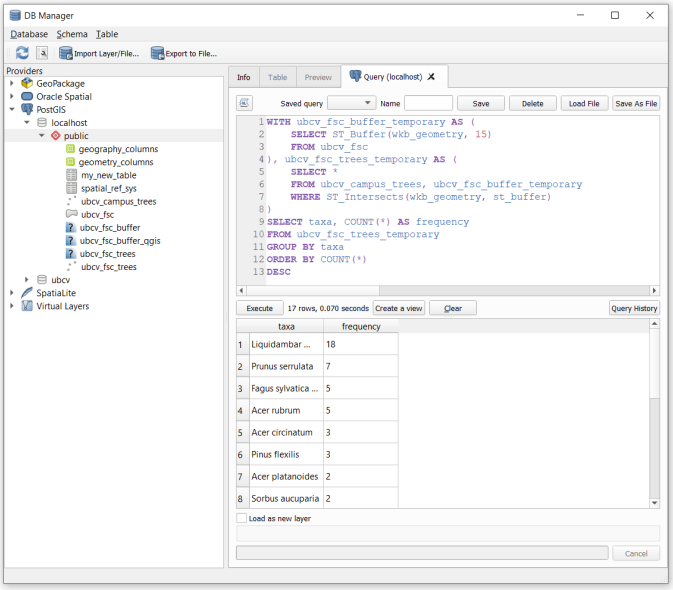
Step 8: At the top of the DB Manager window there is a small button to open an SQL tab. Click that and then enter the buffer statement again, but modify the output table name so that you do not create a conflict:

```
SELECT ST_Buffer(wkb_geometry, 15) INTO ubcv_fsc_buffer_qgis FROM ubcv_fsc;
```

You should now have a new table in the database now called **ubcv_fsc_buffer_qgis**. You can also save the query by giving it a name in the **Name** field and even output the SQL query to a file. This can be helpful for managing large complex queries. The **Create a view** button will display the layer in your map view.

Step 9: Using the `WITH` clause, combine all of our independent queries above into a single query in QGIS that buffers the FSC polygon by 15 m, intersects

the buffer with the campus trees, and then reports the number of unique taxa around the building. See the screenshot below.



The `WITH` clause allows us to create temporary relations that can be referenced by name in subsequent `SELECT` statements. In the case above, **ubcv_fsc_buffer_temporary** is the temporary result from the buffer and **ubcv_fsc_trees_temporary** is the temporary result from intersecting the buffer with the campus trees.

As you will see, this is slightly different than using the `INTO` method from our prior examples, which creates a new relation in our database. The benefit of `WITH` and `AS` is that the relation is ephemeral and only exists for the moment that the SQL entire statement is executed; no new relation is written into our database.

Step 10: Using what you have learned, select one of the other PostGIS overlay methods and then apply it to any two datasets in the `ubcv` database on the UBC PostgreSQL server. You must use the `WITH` clause and you must also use one of the SQL aggregation functions. It is not important that your analysis makes sense, but it is important that your entire SQL statement can be executed from either QGIS or psql. Once your are satisfied with your SQL statement, copy and paste it to the assignment submission page on the UBC course management system.

Hint: Check what the geometry column name is called for the layers that you choose to use and enter those exactly as you see them into your overlay function. Make sure that you are using compatible geometries for the overlay method you choose and read the linked documentation above if you have any doubts!

Summary

PostGIS has many powerful functions for handling geospatial data directly within a PostgreSQL database. These are not always practical for day-to-day use when compared with user-friendly applications like QGIS or ArcGIS Pro. However, you should now appreciate that you can remotely manipulate geospatial data on a server and this opens the possibility of very powerful web-based applications, cloud-based GIS solutions, and automated scripting that requires little visualization. If you are interested in learning more about what others have done, you can check out the Crunchy Data YouTube channel to see past "PostGIS Day" recordings.

Return to the **Deliverables** section to check off everything you need to submit for credit in the course management system.

Chapter 3

Suitability and Overlay Analysis

Written by Annie Mejaes

Lab Overview

Oftentimes, what we consider to be GIS analysis in the natural resources domain is related to suitability analysis. This analysis typically involves identifying areas/features that could support a given activity, use, or process and eliminating areas/features that would not be able to fulfill the required needs. Suitability analysis typically refers to an ordinal classification of the capable areas/features to denote the relative abilities of these areas/features to fulfill these needs. This number is mostly likely represented as an index that ranges from 0 to 1 where 1 = the most suitability possible for a given area/feature and 0 = no suitability. Typically, there will be no 0 values present in this index as all incapable areas should have already been eliminated in previous steps.

In order to perform a suitability analysis, geospatial operations, such as vector polygon overlay and database table intersection are not only the distinguishing functional characteristics of a geographic information system, but are also the most common aspects of this process. In order to gain an understanding of the potential of cartographic modeling using a GIS, this lab has been constructed to take you through an exercise that closely mirrors a prototypical GIS analysis related to conservation values.

Marine spatial planning is "a collaborative and transparent approach to managing ocean spaces that helps to balance the increased demand for human activities with the need to protect marine ecosystems. It takes into consideration

all activities and partners in an area to help make informed decisions about the management of our oceans in a more open and practical way.

Marine Spatial Planning is internationally recognized as an effective tool for transparent, inclusive and sustainable oceans planning and management. Approximately 65 countries are currently using this approach. Marine Spatial Plans are tailored to each unique area to help manage human activities and their impacts on our oceans. Depending on the area, these plans may include areas for potential resource development and areas that require special protection" (Fisheries and Oceans Canada).

The State of Hawai'i defines Marine Protected Areas as "a subset of MMAs, and focus on protection, enhancement, and conservation of habitat and ecosystems. Some MPAs have very few fishing restrictions and allow sustainable fishing, while others restrict all fishing and are "no take" areas. In Hawai‘i, forms of MPAs have been in use for over 40 years." -Division of Aquatic Resources

Within this lab, we will utilize existing and new geospatial tools to conduct a marine spatial planning exercise in Hawaii, USA. You will complete tasks based on a scenario related to the critical habitat of cetaceans in Hawaii and conflicting uses.

Considered one of the world's most important humpback whale habitats, the Hawaiian Islands Humpback Whale National Marine Sanctuary was established in 1992 to protect humpback whales (*Megaptera novaeangliae*) and their habitat in Hawai‘i. Humpback whales commonly give birth and raise their young in the state's warm and shallow waters. Yet, there is increasing conflict between the whales and vessels, including recreational and whale watching boats, as well as cargo ships. Thus, national and state government agencies would like to expand the marine sanctuary to reduce the whales strikes, further protecting the whales and their calves. The government agencies are searching for the largest suitable areas to include as part of the sanctuary.

Learning Objectives

- Practice good data management principles
- Distinguish overlay tools and recognize when to apply them
- Calculate a weighted suitability index
- Identify and map the most suitable locations for new marine sanctuaries

---

Deliverables

Answers to the questions posed throughout the lab (10 points)

Map of the final suitability analysis, requirements listed below (20 points)

- The map should show the proposed sanctuary locations
- Sanctuary locations should be symbolized by their suitability
- The most suitable sanctuary location should be clearly identified in the map (e.g., different colour, line weighting, symbolization, etc.)
- You can add additional layers and a basemap, but be sure to avoid making the map too cluttered
- All features on the map should appear in the legend
- Map should be 11"x17" either as a landscape or portrait layout
- You should export the map as a PDF document
- Your map should have a title, north arrow, scale bar, legend, your name, and date

---

Data

All data for this lab are accessible via the UBC PostgreSQL server. Instructions for connecting to the server are given in prior labs. We will be using data from the `whale` database. You will be expected to practice proper data management by copying the data into your own database and then managing the many new datasets that are produced.

Data Organization

You will find this lab much easier if you keep your data in a structure that makes sense for you – and others (as much as possible!) – by using meaningful names. As you progress through each step, we recommend that you also take a look at your data and use tools to delete unnecessary fields in your attribute tables so that they are not overpopulated and confusing.

Files to Create

Here is a table of files you will be creating:

File Name	Created in Task
Humpback_BIA	1
Humpback_Hawaii_BIA	1
BIA_Sanctuary_Erase	2
BIA_Sightings	3
BIA_Multipart	3
BIA_Sightings_Multipart	3
Boat_Uses	4
BIA_Sightings_Boat	4
Top_Five_Sanctuary	5
Sanctuary_Buffer	5

Task 1: Export Relevant Data from Biologically Important Areas Shapefile

A biologically important area (BIA) identifies where cetaceans concentrate for specific behaviours.

Step 1: Start a new project in ArcGIS Pro.

Step 2: Connect to the `whale` database on the UBC PostgreSQL server in ArcGIS Pro. Add **Cetaceans_BIA** to your map in ArcGIS Pro.

Step 3: Right-click on the layer and select 'Attribute Table'. Explore the attribute table and understand what field you will need to use to extract 'Humpback Whale'.

Step 4: Right-click on the layer again, select 'Data', and then select 'Export Features'. Write a SQL query to extract the BIA for the humpback whales from the other species and save the output to your local geodatabase in your ArcGIS Pro project. Ensure the file is added to your map in ArcGIS Pro and then explore the new layer and attribute table.

Q1. What was the SQL expression that you used to export only the humpback whale BIA? (2 points)

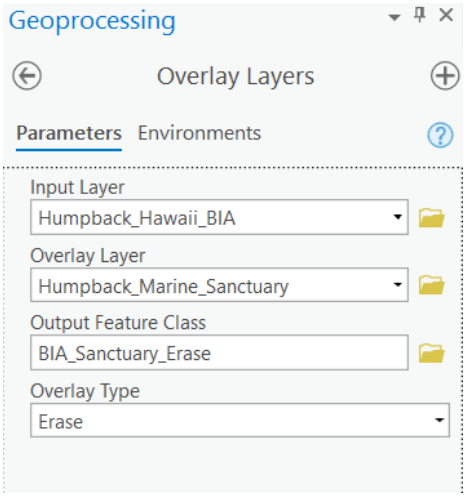
Q2. Apart from Hawaii, does the humpback whale have any other locations with a BIA for reproduction? (1 point) **Step 5:** Repeat Step 4, but this time only export the humpback whale BIA for Hawaii and name the file **Humpback_Hawaii_BIA**.

Task 2: Identify the BIA of the Humpback Whale that is not within the established Marine Sanctuary

Visually compare the Marine Sanctuary (**Humpback_Marine_Sanctuary**) to the BIA of the humpback whale population.

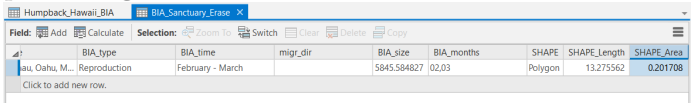
Step 1: Open the tool 'Overlay Layers'.

Step 2: Within the tool, the input layer should be the BIA, the overlay layer is the sanctuary. For 'Overlay Type', select Erase. Name the output **BIA_Sanctuary_Erase**.



Q3. What percentage of the BIA is outside of the marine sancuary?

(1 point)



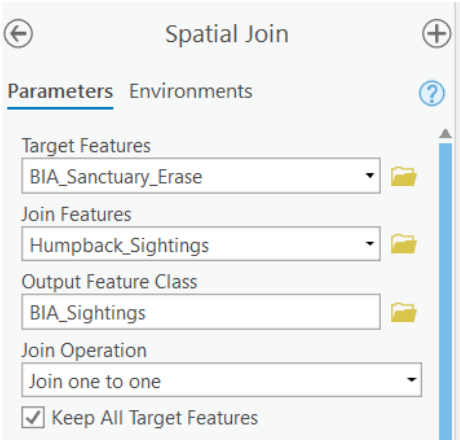
Task 3: Calculate the Density of Cetacean Sightings using the BIA for Humpback Whales

Next, we have a point layer named **Humpback_Sightings** that is curated based on where cetaceans have been located or identified in Hawaiian waters.

We want to understand the density of the sightings within the remaining BIA layer to understand which polygons should be prioritized.

Step 1: Open the tool 'Spatial Join'. Spatial Join is a valuable tool that joins attributes from one feature to another based on a spatial relationship. The target feature defines the spatial boundary, while the input feature is molded to the target feature.

Step 2: Within the tool, the target features are **BIA_Sanctuary_Erase**, the input feature is **Humpback_Sightings**, and the join operation is 'Join one to one'. The match option is 'Contains'. Name the output **BIA_Sightings**.



Step 3: Open the attribute table. What do you see went wrong with this analysis?

We need to introduce a new tool, 'Multipart to Singlepart' to fix this problem. After you have used the 'Overlay Layers' to erase features from the BIA for humpback whales, you will be left with several polygons that are still considered the same polygon by ArcGIS Pro, but are now spatially separated (refer to the help page for Multipart to Singlepart tool). Try finding one and selecting it. You will know it is multipart when a number of additional polygons are outlined when you click one polygon. You will need to separate those parts into unique polygons before continuing on to the next step. What this is allowing you to do is to recommend placement of the expanded marine sanctuary into a portion of a BIA.

Step 4: Open the 'Multipart to Singlepart' tool and select the input layer as the layer **BIA_Sanctuary_Erase**.

Q4. How many individual polygons now exist? (1 point) **Step 5**: Now re-do 'Spatial Join' from Steps 1-2. There are many polygons that have a minimal area, so, unselect 'Keep all Target Features', which will remove polygons that do not have any cetacean sightings within its boundaries.

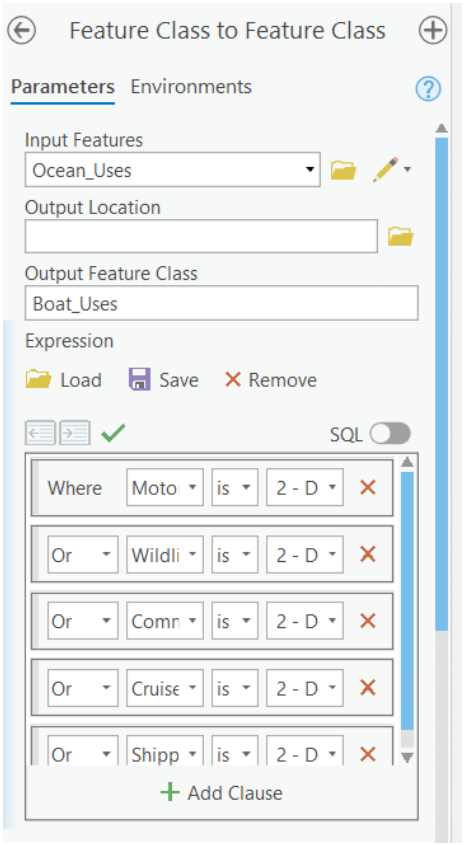
Step 6: Use symbology to display the BIA polygons by number of cetacean sightings.

Q5. What is the Target_FID of the polygon with the most cetacean sightings? (1 point)

Task 4: Extract the Boats from Ocean Uses and Identity the Vessels within the BIA Polygons

The **Ocean_Uses** layer contains a wide range of human activities that occur at the coast or in the ocean. We would like to extract 'Motorized Boating', 'Wildlife Viewing at Sea', 'Commercial Pelagic Fishing', 'Cruise Ships', and 'Shipping'. These human activities pose the most threat to humpback whales, and, so, we would like to expand the sanctuary to protect humpback whales where their current risk of collision with a vessel is the highest.

Step 1: Export features where Motorized Boating or any of the other human activities mentioned above is equal to 'Dominant Use'. Be sure to set the output location to your local ArcGIS Pro project geodatabase and name the output feature class **Boat_Uses**.



Now, we would like to bring the data on vessel usage and the BIA polygons together using the Overlay Layers tool. This is important to be able to see the spatial overlap between where humpback whales reproduce and rear their young and where vessels are labeled as a Dominant Use, indicating that there is a higher chance of boat strikes in these locations.

Step 2: Open the tool 'Overlay Layers'. The input layer should be **BIA_Sightings_Multipart** and the overlay layer is **Boat_Uses**. For 'Overlay Type', select Identity. Name the output **BIA_Sightings_Boat**.

Q6. Open the attribute table for the output. What does the 0, 1, and 2 represent in the swimming column? (1 point)

Task 5: Calculate Suitability and Identify Most Suitable Locations for the Marine Sanctuary

Now, we will determine where is the most suitable location to expand the marine sanctuary for humpback whales. The suitability value will be a new field between the value of 0 and 1 with 1 being the most suitable, and 0 being not suitable. It will be based on 3 characteristics (suitability factors):

- the larger the area the higher the suitability score
- the more cetacean sightings per area, within an area the higher the suitability score
- the more dominant ocean uses the higher the suitability score

Step 1: Before you calculate final suitability score, you will need to add three new fields to **BIA_Sightings_Boat** within the attribute table:

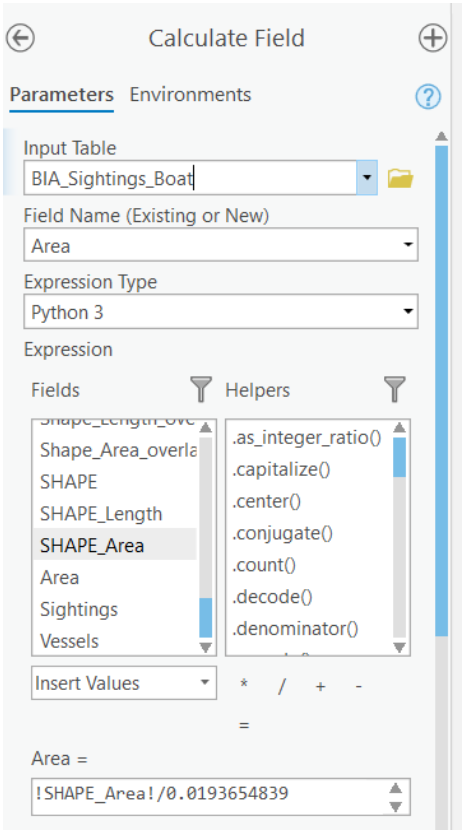
- Area (float)
- Sightings (float)
- Vessels (float)

Step 2: For each of the three fields you just added, you will need to calculate $[F/Fmax]$ for each record (row):

- F is the value of the attribute for that record
- $Fmax$ is the maximum record value for the field

HINT: Always add a decimal point to the Fmax so that the result is a floating point decimal number.

First, determine $Fmax$ for Area (use Shape Area field for $Fmax$), Sightings (use Join Count), and Vessels (calculate manually based on the number of columns). Then, Right-click on the field in the attribute table and view statistics (write down the number that appears beside 'maximum'). Finally, right-click on the field and select 'Calculate Field' and enter the expression using the correct field name and the maximum value for the field.



HINT: For Vessels, you will need to include more than one column for F.

Step 3: Finally, use a weighted suitability calculation by creating a new field, Suitability, and writing the following expression: [Area] * 0.4 + [Sightings] * 0.2 + [Vessels] * 0.4.

3.0.0.0.1 Q7. What is the maximum value for Area to four decimal places? (1 point)

3.0.0.0.2 Q8. What is the maximum value for Sightings to four decimal places? (1 point)

3.0.0.0.3 Q9. What is the maximum value for Vessels to four decimal places? (1 point)

To provide a buffer zone protecting the humpback whales within the newly formed sanctuary, we will select the five BIA areas with the highest suitability score and create a five kilometer buffer.

Step 4: Sort the Suitability field by right-clicking on the field title and selecting **Sort Descending**.

Step 5: Select and export the top five features to a new layer named **Top_Five_Sanctuary**. Clean up the attribute table, so that it only has relevant columns. This is the output table that is a deliverable for the lab.

Step 6: Open the tool **Buffer**. Input features will be **Top_Five_Sanctuary**. Name the output **Sanctuary_Buffer**. Set the distance as 5 kilometers. For dissolve type, select **Dissolve all output features into a single feature**.

You just conducted a suitability analysis and produced a map of how to potentially expand the humpback whale sanctuary!

Summary

Suitability modeling is a very common type of analysis that usually integrates multiple factors as geospatial layers using overlay and proximity tools. As you can imagine, there are many ways to structure this analysis and different weightings in the suitability calculation will yield entirely different results. Therefore, it is always important to maintain good justification and rationale for the weightings that you choose and any limitations in the analysis (e.g., missing data/information, incomplete or unavailable attributes, etc.) are clearly communicated in any recommendations that you make from your analysis. In this lab, you were also exposed to significant data manipulation and management. Using good data organization and naming conventions helps others understand your analysis and output data, including your future self if you ever need to return to your old work.

Return to the **Deliverables** section to check off everything you need to submit for credit in the course management system.

Chapter 4

Exploring QGIS and OpenStreetMap

Written by Paul Pickell

Lab Overview

In this lab you will learn some basic functionality of handling data with the QGIS software and learn how to extract and import an important source of Volunteered Geographic Information (VGI): OpenStreetMap (OSM). QGIS (Quantum GIS) is a Free and Open Source Software (FOSS) that is a flagship project maintained by the Open Source Geospatial Foundation (OSGeo). QGIS features much of the same functionality as other proprietary software like ESRI's ArcGIS and navigating the interface will be the primary purpose of this lab. OpenStreetMap is one of the most popular and comprehensive VGI datasets available for the planet and contains geographic information that is crowd-sourced. The spatial data model of OpenStreetMap data is significantly different from other vector models and therefore requires some special handling and consideration. At the end of this lab, you will produce a map of some OpenStreetMap data using QGIS.

Learning Objectives

- Import and export various types of geospatial data in QGIS
- Practice spatial and attribute queries

- Calculate fields with new values
- Install plugins and extensions for QGIS
- Create map layouts and modify layer symbologies

Deliverables

Answers to the questions posed throughout the lab (15 points)

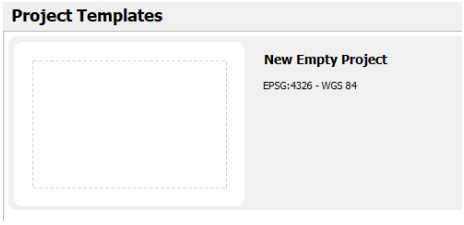
Your final symbolized map (15 points)

Data

Data for this lab will be downloaded directly from OpenStreetMap following the tasks below.

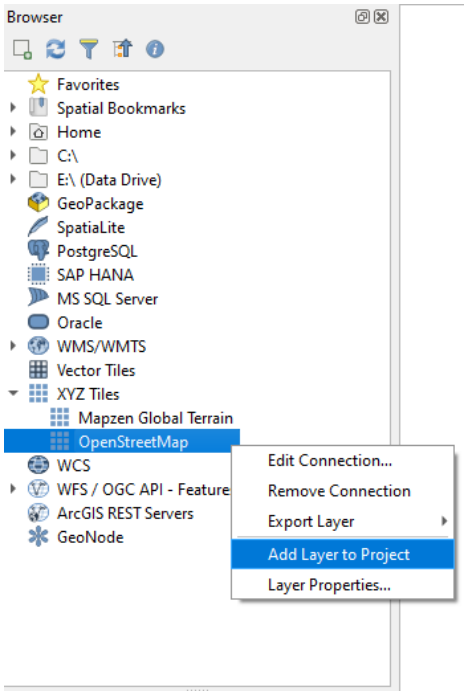
Task 1: Downloading OpenStreetMap data to QGIS

Step 1: Start QGIS and click **New Empty Project** from the Project Templates on your screen.



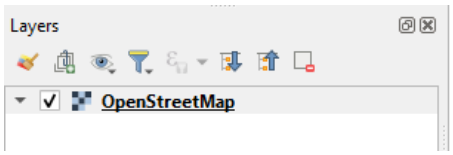
The Project Templates prompt will disappear to a white canvas, which is where you are able to view mapped data. On the left of your screen you will see two panes: **Browser** and **Layers**. These are where you can find data sources on your computer or network and the layers that are currently loaded into your map view. Currently, we do not have any data to view, so the Layers pane will be blank. But we do have many sources for data, so let us look at those.

Depending on your computer and networking connections, your Browser pane may look different than what is shown below.



Step 2: Save your project. In the top menu, select **Project** then click **Save As...** and navigate to a location on your computer where you want to store your project and related files. You should create a directory specifically for this purpose.

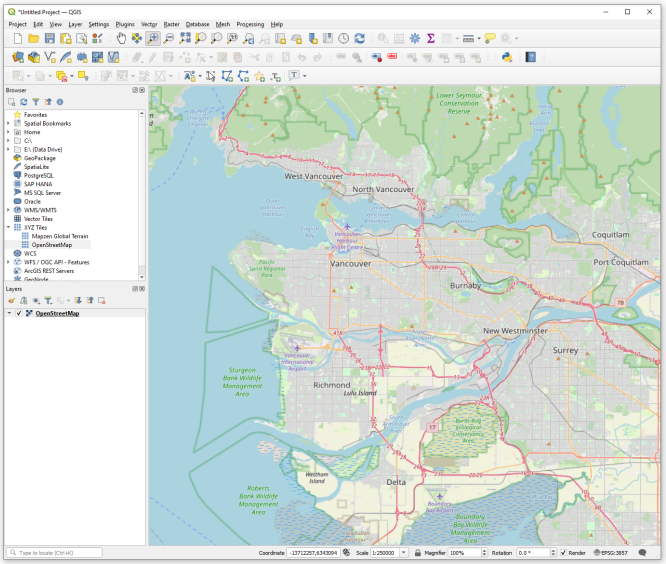
Step 3: Expand **XYZ Tiles** and then either double-click **OpenStreetMap** or right-click and select **Add Layer to Project**.



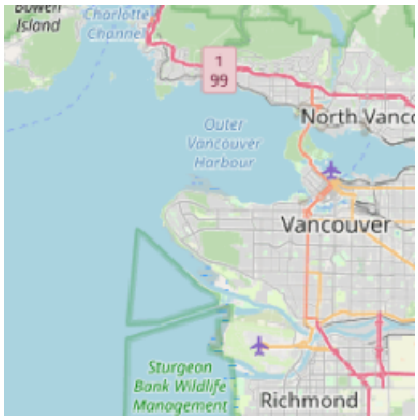
This will add the OpenStreetMap base map to your map view, so you should now see your first layer in the Layer pane and the map view should automatically display the new base map.

Step 4: Locate the Map Navigation Toolbar at the top. Use the magnifying glass to zoom into the area for Metro Vancouver. Once you find Metro Vancouver, change the scale of the map to 1:250000 at the bottom of your QGIS

interface.



This current view does not directly show you the OpenStreetMap data. Instead, you are looking at a tile service provided by OpenStreetMap, which symbolizes the OpenStreetMap data and then, depending on your zoom level and location, sends the necessary information to your computer in the form of a tile like the one pictured below showing a 256x256 pixel tile at zoom level 10 (zoom level 1 is the smallest scale world map).



In order to get the OpenStreetMap data, we need to install a plugin for QGIS. Plugins are written by a community of volunteer developers, some from organizations that pay for the plugin to be developed and then released as FOSS, and others just volunteer their time. Plugins are incredibly important for QGIS, because only basic features end up in the core GIS software. If there is ever

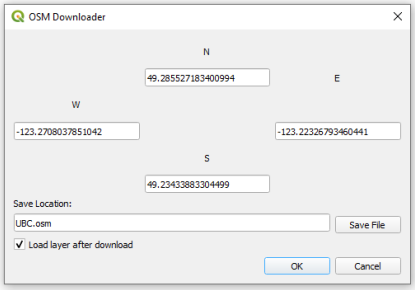
something you want to do that is not a basic feature of QGIS, you probably need to search for a plugin or write one yourself!

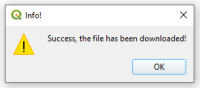
Step 5: From the menu bar at the top, click **Plugins** and then select **Manage and Install Plugins...** to open the **Plugin Manager**. Since these plugins are fetched from a server, you need to be connected to the internet to see plugins available for download. In the search bar, search for the **OSMDownloader** plugin. Click **Install Plugin**.

Once the plugin is installed, you can click **Close** to dismiss the Plugin Manager. Most plugins will automatically add an icon to your toolbar at the top, so search for and click the OSMDownloader icon that looks like this:

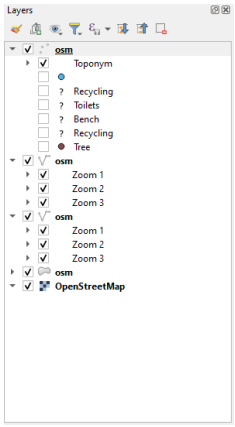
Your cursor should now appear as a crosshair on your map and now you can select a rectangular area to fetch the OpenStreetMap data from.

Step 6: Click and drag your cursor on the base map to select the UBC campus. A dialogue box will open showing you the extent of your selection. Save the file to the project folder your created earlier. Name the file **UBC.osm** and toggle on **Load layer after download**. Your coordinates need not be exactly the same as the image below, but you can modify your selection if you want by either manually editing the extent values (in decimal degrees) or clicking **Cancel** and then selecting again with your cursor.

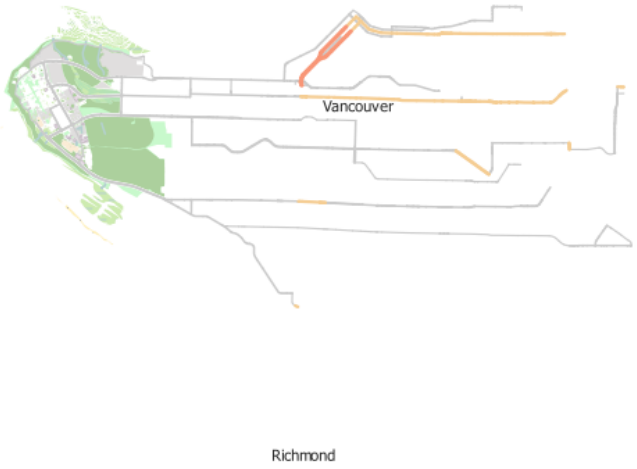




A progress bar will appear at the top and then a message will be displayed that the download was successful. You should now see an updated list of layers available in your Layers pane.



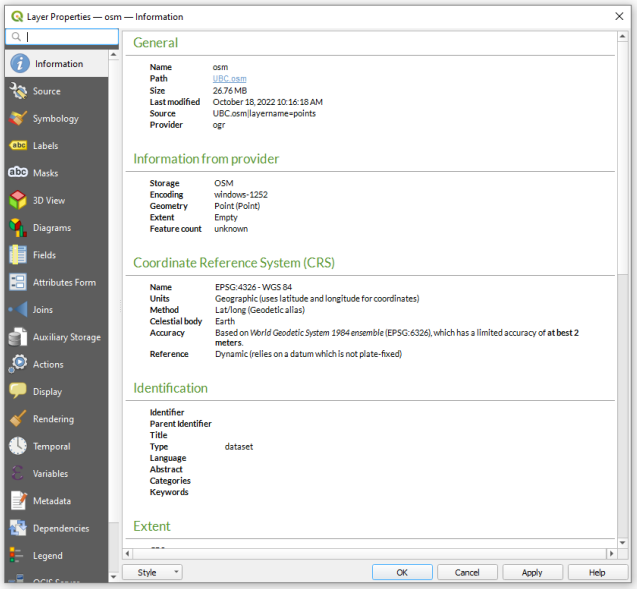
Step 7: Toggle off the OpenStreetMap base map and depending on how you made your selection in the last step, you should see something like this in your map view:



You can also download OpenStreetMap data directly from the OpenStreetMap website and then drag and drop the .osm file directly into the QGIS map view or the Layers pane.

Task 2: Inspecting and processing OpenStreetMap data

Step 1: Right-click the osm point layer in the Layers pane, select **Properties...** and then select the **Information** tab on the left navigation menu of the **Layer Properties** dialogue box.



You should notice under **Information from provider** the **Storage** type is **OSM** and the **Coordinate Reference System (CRS)** is **EPSG:4326 – WGS 84**. OpenStreetMap data will always be distributed with these characteristics. At this point, it is important to recognize how different the OSM spatial data model is from other data structures like an ESRI Shapefile.

OSM files are comprised of **nodes**, **ways**, and **relations**. These generally correspond to points, lines, and polygons, but not exactly. Each of these feature types are described by **tags** or attributes. Nodes are a pair of latitude/longitude coordinates and can either be a building block for another type of feature (e.g., a way or relation) or they can have attributes of their own or both. A way is a list of two or more nodes that connect and form a line. Ways can be open (lines) or closed (areas). It is important to recognize that a node can participate in multiple ways simultaneously, while also describing itself as a discrete point feature. Relations are just that: they allow us to model relationships between existing nodes, ways, and even other relations. For example, individual islands might be represented by a closed way, but an archipelago would be represented by a relation that contains the closed ways for all the islands. In this way, the OSM data model is flexible and eliminates redundancies of spatial information such that nodes and ways can be recycled into all kinds of relationships.

Q1. Interpret the OSM entity-relationship data model shown below. In a brief sentence, describe each of the seven connections between nodes, ways, relations, and tags. (1 point each for 7 points total)
Step 2: Dismiss the **Layer Properties** dialogue box and then right-click the

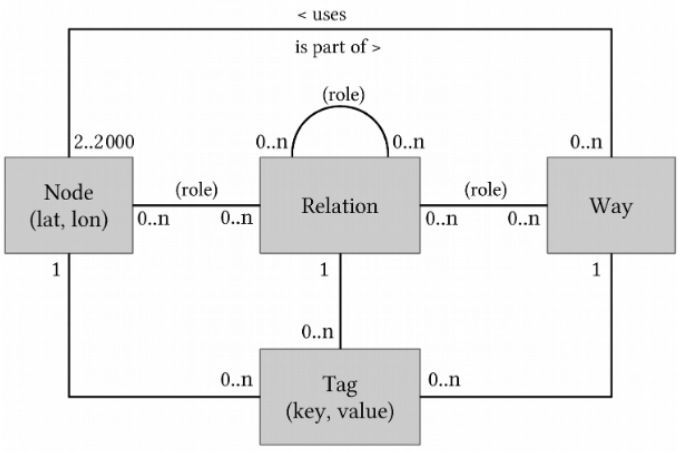
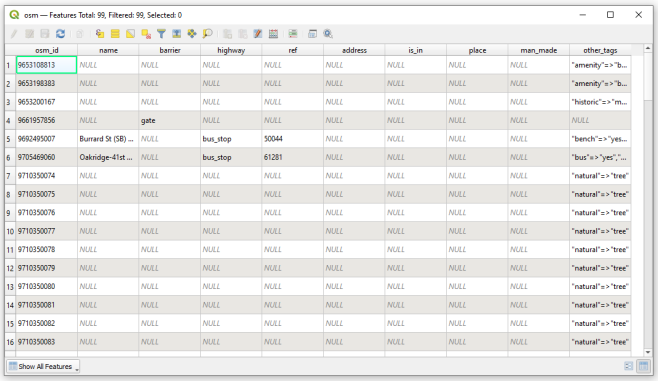


Figure 4.1: Figure Credit: Jochen Topf

OSM point layer again in the Layers pane, but this time select **Open Attribute Table**. Your table might have a different number of features and different values for the attributes than what is shown below.

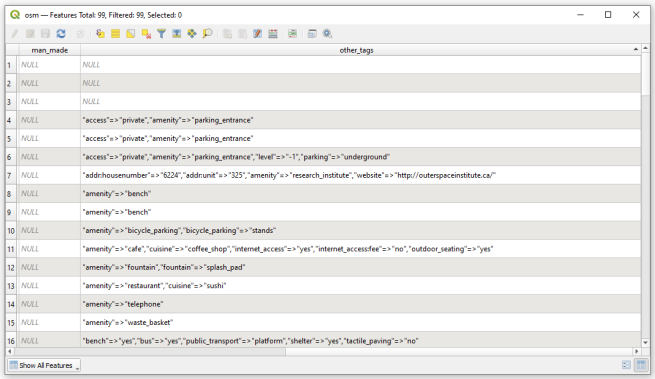


The first field **osm_id** is the primary key of the node for the data you have downloaded. This is the primary key of all nodes in the planetary database of OpenStreetMap data (not just your file), which is why it starts from a very large number, 9,653,108,813 in the image above.

The remaining attributes are the tags, which might look random, and that is because OpenStreetMap was originally designed to store street information, but then evolved into a larger participatory community mapping project. Thus, we see there are some initial keys that relate to streets: **barrier**, **highway**, **address**, etc. But not all OpenStreetMap nodes are related to streets, so over

time, the community has added many new tags, and most of these will not apply to every node, which is why we see so many **NULL** values.

Step 3: Click and drag the **other_tags** attribute column to expand it so that you can see the other tags for each node. You might need to also scroll to the right to view it or maximize the attribute table view.



	man_made	other_tags
1	NULL	NULL
2	NULL	NULL
3	NULL	NULL
4	NULL	"access"=>"private","amenity"=>"parking_entrance"
5	NULL	"access"=>"private","amenity"=>"parking_entrance"
6	NULL	"access"=>"private","amenity"=>"parking_entrance","level"=>"-1","parking"=>"underground"
7	NULL	"addr:housenumber"=>"6224","addr:unit"=>"325","amenity"=>"research_institute","website"=>"http://outerspaceinstitute.ca/"
8	NULL	"amenity"=>"bench"
9	NULL	"amenity"=>"bench"
10	NULL	"amenity"=>"bicycle_parking","bicycle_parking"=>"stands"
11	NULL	"amenity"=>"cafe","cuisine"=>"coffee_shop","internet_access"=>"yes","internet_access:fee"=>"no","outdoor_seating"=>"yes"
12	NULL	"amenity"=>"fountain","fountain"=>"splash_pad"
13	NULL	"amenity"=>"restaurant","cuisine"=>"sushi"
14	NULL	"amenity"=>"telephone"
15	NULL	"amenity"=>"waste_basket"
16	NULL	"bench"=>"yes","bus"=>"yes","public_transport"=>"platform","shelter"=>"yes","tactile_paving"=>"no"

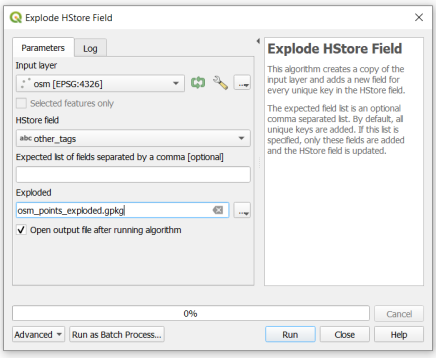
Notice how the **other_tags** are encoded as a comma-separated string list. These tags are structured as key-value pairs. For example, "*amenity*" => "*bench*" indicates the key "*amenity*" has a value "*bench*". All the keys and values are always strings, even numerical values. On the one hand, this data scheme allows for many tags to be stored for any given node and irrelevant tags or attributes need not be stored at all. This type of data encoding is known as **Hstore** and is an alternative way to store information and reduces the size of an attribute table.

Q2. How could an Hstore data encoding affect how you can interact with and query the data in a relational database? (3 points) In order to actually access any of the attributes in the other_tags field, we need to explode the Hstore. Exploding the Hstore means to parse all the tags for each tuple/node and create a new dataset that contains an attribute table with all the unique attribute fields in the way that we are normally accustomed to with the relational database model.

Step 3: From the top navigation menu, click **Processing**, then select **Toolbox**. This opens the **Processing Toolbox** pane for QGIS where you can find all available tools.

Step 4: In the search bar, search for the tool **Explode HStore field** and then double click the tool to open it. For the **Input Layer**, select the osm point layer from the drop-down menu and for **HStore field** select **other_tags**. For **Exploded**, enter **osm_points_exploded.gpkg** for the output file name geopackage and optionally the path to the directory where your project is lo-

cated. You can click the icon to the right of the field and select **Save to File...** to navigate to your project directory.



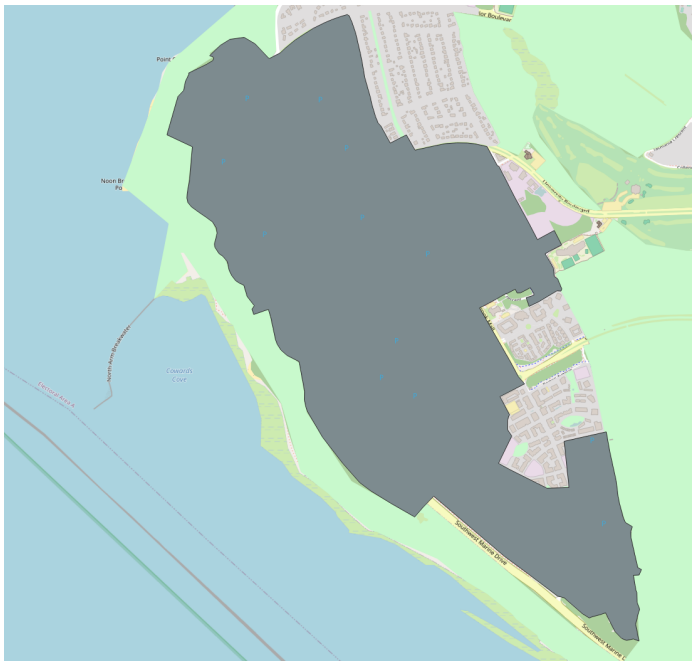
You should now find a new copy of your osm points in your **Layers** pane. It is also possible to achieve these results by using another popular plugin called **QuickOSM**, which allows you to refine your query based on available tags and it can also parse local .osm and .pbf files (the planetary OpenStreetMap dataset is distributed as a .pbf).

Step 5: Repeat the last step again for the other OpenStreetMap layers, ensuring that you name the outputs to correspond to the data type (e.g., polygon, LineString, MultiLineString). You may need to inspect the layer properties to distinguish the two line layers.

Step 6: Inspect the attribute table for the MultiLineString layer.

Q3. What do these features all have in common? (1 point) MultiLineString is a multipart line feature that is represented as a single tuple in the attribute table with multiple line parts. The line parts need not be connected or continuous, so this feature type allows for multiple line segments to be represented as a single tuple in the attribute table.

Step 7: Toggle on the osm_polygons_exploded layer in the map view and then use the **Identify tool** to explore the polygon data and find a key that identifies the UBC campus boundary. Select and export that polygon to a new geopackage file in your project called **osm_ubc_campus_boundary**. Make sure you toggle on **Save only selected features** when you do the export. There might be several options, so you will need to sort through them until you get the proper boundary shown below.



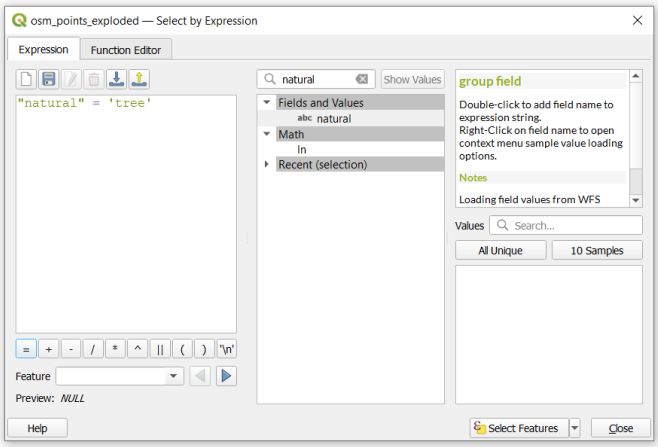
Step 8: Use the **Select by Location** tool to select the osm_points_exploded by the campus polygon you just exported. Toggle on Intersect for the selection. Export the selected points to a new geopackage in your project called **osm_points_exploded_campus**. Ensure that **Save only selected features** is toggled on when you do the export.

Once you are satisfied with the export, return to the attribute table and click the **Deselect all features** from the layer button at the top. This is an important step, because if you leave the selection, then all future geoprocessing may only occur on the features that you have selected!

Step 9: Open the attribute table for the **osm_points_exploded_campus** layer and click the **Select features using an expression** icon.

This displays the Select by Expression dialogue and you can enter in a query to subset the data. The expression goes into the blank left side of the dialogue, the center of the dialogue gives you options for constructing your query, and the right-side displays help information based on what is selected from the center. From the center, expand the **Fields and Values** item. There are a ton of tags!

Let us search for a tag instead. Using the search bar above, type **natural**, then under Fields and Values double-click **natural**. This will add the field name to your statement on the left. On the right, click the **All Unique** button to see what values are available for this tag. We will select all trees, so double-click **tree**. Add an equal sign to your statement so that it reads "*natural*" = '*tree*', as shown below. Notice how the key is in "double quotes" and the value is in 'single quotes'. Click **Select Features** to execute the statement then close the dialogue and inspect the result.



Step 10: With the selection active from the last step, right-click the osm_points_exploded_campus layer in the Layers pane and select **Export** and then **Save Selected Features As...** and name the output **osm_campus_trees**. Ensure that **Save only selected features** is toggled on then click **OK**. Inspect the output in the map. Remember to deselect the attribute table!

Notice that a lot of street trees are missing on campus? You can rectify this if you want, just sign up for an account at OpenStreetMap.org and start digitizing! You can also use any of a number of free apps on iOS and Android to start mapping from your phone or to record GPS tracks.

Step 11: Open the attribute table and use the Identify tool to find a key in the osm_polygons_exploded that you can use to identify buildings. Once you have identified the key, select all the buildings with an expression. Hint: statements involving $NULL$ values for keys need to use the IS and NOT operators, rather than = (equal to) and ! = (not equal to).

Step 12: We are now going to modify the selection from the last step to only include buildings that are located on campus. Open the **Select by Location** tool and change **Select features** from to the **osm_polygons_exploded** layer. Toggle on **are within** for the geometric predicate and then **By comparing to the features from** should be set to the osm_ubc_campus_boundary

polygon. Change **Modify current selection by** to **selecting within current selection**. This last part is important as we are basically taking what we select in the last step and selecting from that selection the buildings that are geographically within the UBC boundary. Click Run and then inspect the output to make sure only buildings within the UBC boundary are selected. When you are satisfied, export the selection to a new geopackage in your project called **osm_campus_buildings**. Again, make sure you toggle on **Save only selected features** and then deselect the layer when you are done.

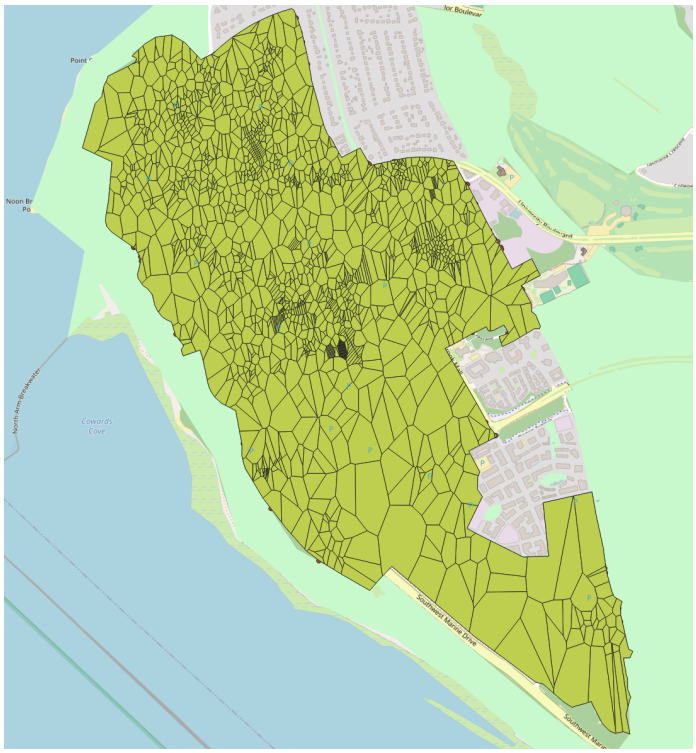
Q4. How could a value of "YES" for a key impact your analysis of that key instead of a more specific value? (1 point)

Task 3: Simple spatial analysis with QGIS

Step 1: From the top menu, select **Vector**, then **Geometry Tools**, then **Voronoi Polygons...** For the **Input layer**, select **osm_points_exploded_campus** and change the **Buffer region** to **100**.

Click the ellipse button next to the blank field for **Voronoi polygons** and select **Save to GeoPackage...** Name the output **osm_campus_voronoi** and then click **Run**. Inspect the output in the map. You should now have a Voronoi diagram that is a little larger than campus.

Step 2: From the main menu, click **Processing** and then **Toolbox** to open the **Processing toolbox** pane (if it is not already open). Search for the **Clip** tool and then clip the Voronoi polygons by the UBC campus boundary and name the output **osm_campus_voronoi_clip** then click **Run**. Inspect the output in the map. You should now see Voronoi polygons within the campus boundary as shown below.



Step 3: Open the attribute table for the clipped Voronoi polygons and click the **Open field calculator** button that looks like an abacus. This will display the **Field Calculator** dialogue. We are going to calculate the polygon areas and save those values to a new field.

Step 4: Ensure that **Create a new field** is toggled on. In **Output field name**, the new field is going to be called **Area**. Then type $*area* into the **Expression** box on the left and click **OK**. The units of this new Area field are square meters because your QGIS project properties default to the WGS 84 ellipsoid with square meters as the unit for area measurements. You can change this if you want by navigating to **Project** in the main menu, selecting **Properties**, and then changing the units and ellipsoid to whatever you want.

Step 5: Open the osm_campus_voronoi_clip layer properties. On the left, select **Symbology**. By default, all features are symbolized the same, called **Single Symbol**. At the top drop-down menu, select **Graduated**.

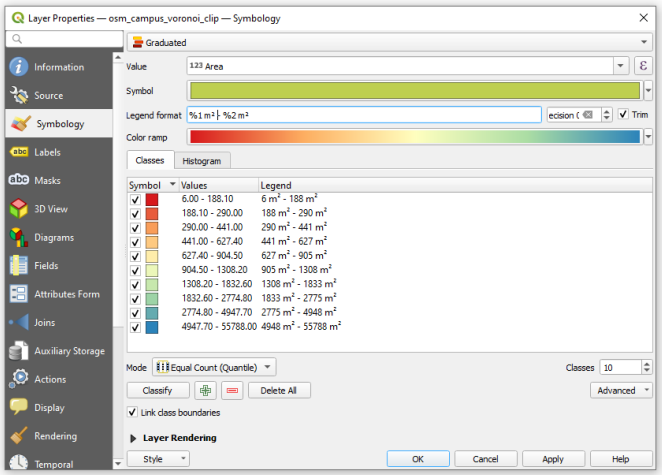
Step 6: For **Value**, select the **Area** field that you calculated earlier. The **Symbol** determines how the features will be filled and should be a solid color. Leave the Symbol as-is.

Step 7: The **Legend format** is used to programmatically modify the legend

text entry in the final map. Because we have selected a Graduated symbology, we will have values that range from %1 to the next value for that color %2. You can add additional text here to indicate units. For example, $\%1m^2 - \%2m^2$ will yield a legend entry that looks like "2425 m2 – 3425 m2". Add units of square meters to the legend format.

Step 8: The **Color Ramp** controls the range of hues (H) and their saturation (S) and value (V). If you click directly on the default color ramp icon, it will open the Select Color Ramp dialogue window where you can modify both discrete and continuous color ramps. You can even control opacity of these values to get some interesting effects on your map. If you opened the Select Color Ramp dialogue, close it. Another option, is to choose a pre-defined color ramp using the downward pointing triangle just to the right of the color ramp. Select **Spectral**.

Step 9: You should now see values and colors appear below in the **Classes** tab. If you do not see anything, then look for the **Mode** drop-down menu towards the bottom of the dialogue window and select **Equal Count (Quantile)**. Click **Apply** and inspect how the Voronoi polygons are now symbolized in your map. Try the other modes and increase or decrease the number of **Classes**, applying the changes each time so that you can inspect the map. Return the Mode to **Equal Count (Quantile)**.



Step 10: If you closed it, open the symbology of the Voronoi polygons again, and then expand the **Layer Rendering** at the bottom of the dialogue window. Change the **Opacity** of the layer to 50% then click **Apply**. Make sure you have other layers toggled off and the OpenStreetMap base map toggled on. Use the Voronoi diagram and the newly created symbology to interpret the density of OpenStreetMap nodes across campus.

Q5. Which areas of campus have a high density of nodes in OpenStreetMap on UBC Campus? What are some possible reasons why there might be low node density in the other areas? (3 points)

Summary

You learned how to handle the OpenStreetMap data model and perform some simple analysis from these data. Next, you might want to consider the implications for using Volunteered Geographic Information (VGI) generally and OpenStreetMap data specifically in future projects. What are the limitations of OpenStreetMap and where does this VGI perform well? Are there any similar, alternative data sources for your area of interest?

Return to the **Deliverables** section to check off everything you need to submit for credit in the course management system.

Chapter 5

Raster and Least Cost Path Analysis

Written by Paul Pickell

Lab Overview

In this lab, you will explore many of the different raster operations using the Geospatial Data Abstraction Library (GDAL). We will practice using these utilities directly from the command line and in QGIS. You will create a variety of different rasters from different sources and practice deriving new rasters through calculations. These efforts will culminate in a least cost path analysis in ArcGIS Pro. We will examine a critical conservation problem in western Alberta, Canada: Grizzly Bear (*Ursus arctos horribilis*) habitat and movement. Grizzly Bears are among the largest land predators in North America and have had much of their historical range reduced by human activity. The largest and most well-studied population now resides in the Rocky Mountain foothills of western Alberta.

Learning Objectives

- Describe the raster data model
- Convert between image coordinates and geographic coordinates
- Convert vector data to raster

- Distinguish data types in rasters and identify appropriate encoding of sampled values for different phenomena
- Mosaic rasters together
- Evaluate expressions on rasters using arithmetic and relational operators
- Combine rasters and calculate weighted overlays
- Calculate a least cost path of a landscape indicator species

---

Deliverables

Answers to the questions posed throughout the lab (20 points)

Create a map of your least cost path that supports your interpretations in Q8 (10 points)

Data

All data for this lab are accessible via the UBC PostgreSQL server. Instructions for connecting to the server are given in prior labs.

Task 1: Define a raster area of interest

Our first task is to define an Area of Interest (AOI) that will represent the extent of all of our raster processing.

Step 1: Open QGIS and create a new project named "GrizzlyBear".

Step 2: Connect to the `bear` database on the UBC PostgreSQL server and load the "grizzlybearmanagementareas" layer into your map. These polygons represent the Grizzly Bear population units or Bear Management Areas (BMA) in the Rocky Mountain foothills of Alberta (Alberta Open Government License). For this lab, we will be focusing on the Yellowhead BMA for our AOI.

Step 3: Write an SQL query to select the Yellowhead BMA and then export it to your local QGIS project folder as a shapefile named "yellowhead_bma.shp", and change the "CRS" to NAD83 / UTM zone 11N (EPSG:26911).

We are going to rasterize this layer and do the basic raster processing outside of QGIS/ArcGIS Pro for a couple of reasons. First, setting the options and raster processing environments in QGIS/ArcGIS Pro is overly complicated and prone to error and unexpected behaviour. Second, you will learn how to use some valuable raster processing utilities of GDAL that can be easily batched and run in chain sequence (i.e., you can batch process large research datasets from a command line).

Step 4: Open the OSGeo4W shell and navigate to your QGIS project folder using the change directory `cd` command. To do this, first copy the path of your project folder, usually something like `C:\Users\paul\Documents\QGIS\GrizzlyBear`. Then in the console type `cd C:\Users\paul\Documents\QGIS\GrizzlyBear` to change to that directory.

The first GDAL utility we are going to use is `gdal_rasterize` (link to documentation). This utility takes a vector input datasource and converts it to a raster in the output. The simplest usage is `gdal_rasterize <src_datasource> <dst_filename>`. All of the flags preceded by `-` are indicated as optional by the square brackets `[]`. If you do not use any of the flags when you run the utility, then the utility will use the default values/settings for those flags that are indicated in the documentation. The AOI raster we are going to create will specify a few conditions of all the other raster processing we will do:

- raster extent (number of rows/columns and geographic coordinates)
- pixel/cell resolution
- spatial reference system

Step 5: In the OSGeo4W shell, enter the following command `gdal_rasterize -burn 1 -tr 20 20 -ot Byte -a_nodata -tap yellowhead_bma_utm.shp yellowhead_bma_utm.tif`.

- `-burn` flag indicates we want to assign a value of `1` to the output raster where the features intersect the new raster grid.
- `-tr` flag indicates we want the output cell resolution to be 20 m (we know its meters because those are the units of the coordinate system) and this is specified twice `20 20`, once for the x-axis and again for the y-axis.
- `-ot` flag indicates that we want the output data type of the raster to be `"Byte"`, which translates to a storage cost of 8 bits per pixel (bits per pixel $\times$ rows $\times$ columns = size of raster on disk).
- `-a_nodata` flag indicates that we want to specifically set the nodata value as `nodata` (default is zero). This will come in handy later because nodata limits the number of cell calculations we need to perform while zero is a valid processing value.

- `-tap` flag (target aligned pixels) indicates that we want the new raster extent to be calculated using floor and ceiling functions:

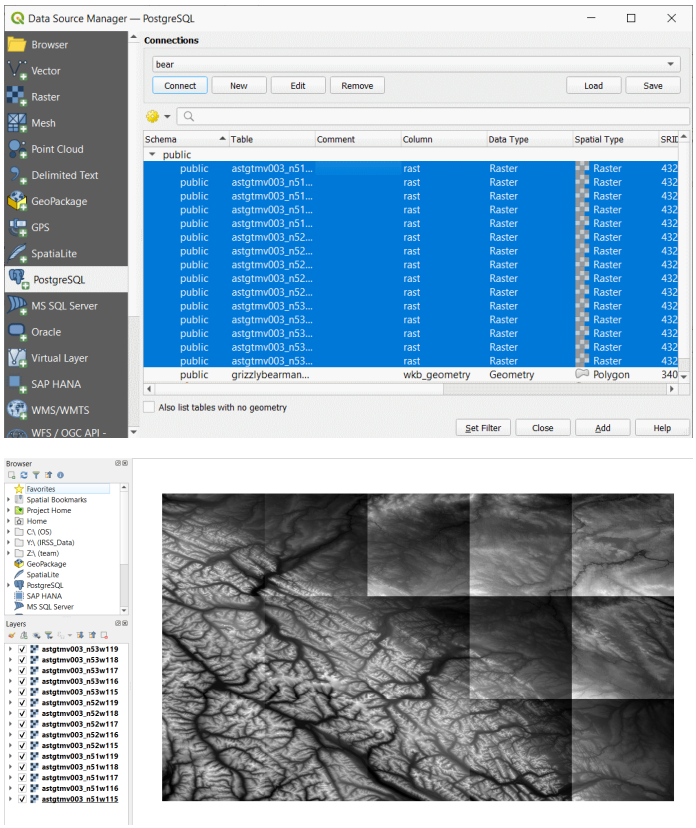
```
ymax (top) = y_resolution × ceiling(ymax / y_resolution) = 20 × ceiling(5,936,280.44760
ymin (bottom) = y_resolution × floor(ymin / y_resolution) = 20 × floor(5,753,454.57494
xmin (left) = x_resolution × floor(xmin / x_resolution) = 20 × floor(401,496.997149 / 2
xmax (right) = x_resolution × ceiling(xmax / x_resolution) = 20 × ceiling(661,663.19412
```

The above calculations are probably not immediately intuitive, but the idea is that we want to create a raster grid that is aligned to the ceiling and floor of our x- and y-resolution. The overall size of the raster is the same whether we do this alignment or not (same number of rows and columns), the only difference is that we are shifting the grid so that the extent coordinates are whole numbers (integers), which makes it easier for us to compute new grids. Bring "yellowhead_bma_utm.tif" into QGIS to inspect it. Check that the values, extent, cell size, and coordinate system are correct. You can also run the `gdalinfo` utility to check the metadata from the command line like `gdalinfo yellowhead_bma_utm.tif`.

Task 2: Seamless mosaicking

A common task in raster analysis is mosaicking multiple rasters together to perform analysis on a single file over a larger extent. We will practice doing this with some ASTER Digital Elevation Models (DEM) that are distributed as non-overlapping tiles in WGS 1984 (EPSG:4326). The source for these data can be found at NASA's Earth Science Data Systems portal, but we will be retrieving these data from the UBC PostgreSQL server instead.

Step 1: Open your QGIS project and from the "Data Source Manager" connect to the `bear` database of the UBC PostgreSQL server. You should see a listing of 15 rasters there, which have names prefixed with "astgtmv003". Select all 15 and then click "Add". It is very important that you add the rasters using this method rather than from the "Browser" pane, otherwise you might have unexpected errors.



Step 2: Open the "Layer Properties" for "astgtmv003_n51w115" and inspect the extent values and the pixel size.

Q1. What units are the extent and pixel size in? (1 point) If you are wondering why the y pixel size is negative, it is to facilitate the conversion between image coordinates and geographic/projected coordinates. Image coordinates are the unique combination of rows `i` and columns `j` representing each pixel in the image. By convention, the origin `(0,0)` for image coordinates is the upper left corner of the image. Image coordinates are always positive and increase as you move down the image to the lower right corner at `(n,m)`, where `n` is the number of columns in the image (think x-axis) and `m` is the number of rows in the image (think y-axis). Most geographic and projected coordinate systems decrease along the y-axis (as you move south), which is the inverse of image coordinates that are increasing. Thus, the y pixel size is usually stored as a negative number in order to convert image coordinates into geographic/projected coordinates.

For example, to convert pixel `(i,j)` from image coordinates to geographic coordinates where the image origin `(0,0)` corresponds to geographic coordi-

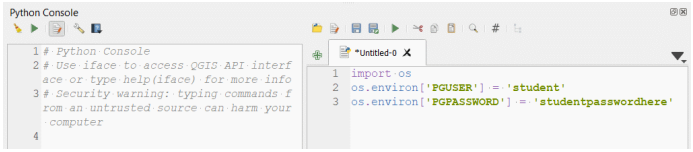
nates `(-115.0001388888889977,52.0001388888888982)` as is the case for "astgtmv003_n51w115" and the x pixel size is `0.000277778` and the y pixel size is `-0.000277778`:

```
Longitude(i) = -115.0001388888889977 + i × 0.000277778
Latitude(j) = 52.0001388888888982 + j × -0.000277778
```

Q2. Given the equations above, what is the exact latitude and longtiude of the pixel with image coordinates (1907,1918)? Show your work for full credit. (2 points) **Step 3:** Copy the coordinates from Q2 and paste them into the "Coordinate" field at the bottom of your QGIS screen (see image below) in the format `latitude, longitude` and press "Enter". Change the scale to `1:500` and press "Enter". You should now be able to distinguish individual pixels. Click the coordinate reference system at the bottom right of the screen and change it from the default value of `EPSG:4326` (WGS 1984) to `EPSG:26911` (NAD 1983 UTM Zone 11N).

Q3. Why do the pixels now appear to be elongated along the y-axis? (2 points) **Step 4:** Save your PostgreSQL credential to your QGIS environment variables by opening the Python console from your toolbar in QGIS, click the small "Show editor" button, and then paste the following into the editor and click the "Run" button:

```
import os
os.environ['PGUSER'] = 'student'
os.environ['PGPASSWORD'] = 'studentpasswordhere'
```

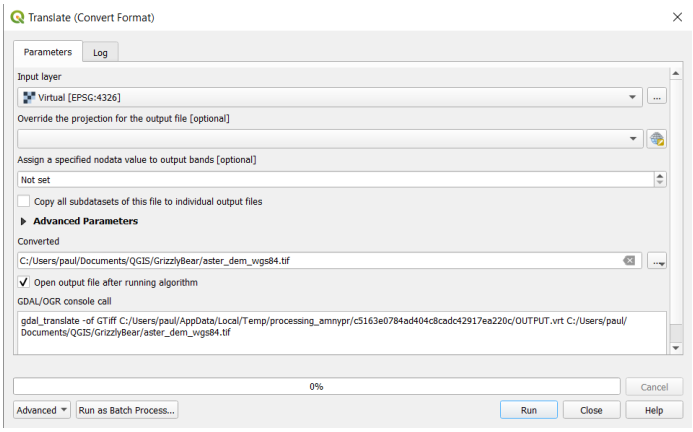


This step will automatically authenticate you each time you need to transact with the PostgreSQL server through the GDAL utilities. Your credential is only saved for the duration of your QGIS session, so once you close QGIS and re-open your project, you will need to come back to this step to save your environment variables. Close the python console and editor when you are done.

Step 5: From the QGIS processing toolbox, search for the "Build virtual raster" tool. This tool takes a set of rasters as input and essentially creates a catalog of those data for later processing. Click the ellipse "..." next to the "Input Layers" field and then click "Select All" to toggle on all the ASTER DEMs currently in

your project. Click "Run". The tool does not actually create a new raster file, but what you will see is that you can now manipulate this catalog of rasters as if it were a single file, which is very handy for mosaicking. Notice when you run these GDAL tools, the log shows you the exact GDAL command that you can run in the command line, like we did in Task 1.

Step 6: The GDAL utility that will actually make the mosaic is `gdal_translate`, which is really just a utility for converting between different raster formats. We are going to use it to read in the virtual raster and write out a GeoTiff. Search for the tool in the processing toolbox, "Translate", make sure you are using the tool under "Raster Conversion". Make sure the virtual raster is selected for "Input layer" and then click the ellipse "…" next to the "Converted" field, "Save to file…", and then name the output "aster_dem_wgs84.tif". Click "Run" and then inspect the output in QGIS. Notice that we did not download any of the raster data to our local machine until this very last step.



The mosaic still needs to be reprojected and clipped to the same extent as our AOI, which is what we will do in the next step using the `gdalwarp` utility. The QGIS tool, "Warp", does not expose all the options that we need, so we will do this next step in the OSGeo4W shell.

Step 7: Open the OSGeo4W shell and navigate to your project folder where you saved "aster_dem_wgs84.tif". Type the following command into the console and then hit "Enter": `gdalwarp -s_srs EPSG:4326 -t_srs EPSG:26911 -te 401480 5753440 661680 5936300 -tr 20 20 -r cubic aster_dem_wgs84.tif aster_dem_utm.tif`.

- `-s_srs` flag defines the input spatial reference system (coordinate system), which is WGS 1984 or `EPSG:4326`.
- `-t_srs` flag defines our target SRS, which is NAD 1983 UTM Zone 11N or `EPSG:26911`.

- `-te` flag defines our target extent `xmin ymin xmax ymax`, which is the calculation we did in Task 1 `401480 5753440 661680 5936300`.
- `-tr` flag defines the target resolution, again specified twice for both axes `10 10`.
- `-r` flag defines the re-sampling method. This is an important flag for `gdalwarp` because any time that you re-project a raster, you are resampling that raster. The term "resampling" comes from the fact that a raster is "sampled" to begin with. Each cell in a raster is a sample of some random variable over the extent of the raster, and the value of that sample is correct for the center of the cell. When we re-project a raster, the locations of the cells change, therefore we need to recalculate the samples or "resample" the raster. Our selected method of resampling in this case is `cubic`, which is a smoothing algorithm and well-suited for continuous data like elevation.
- `aster_dem_wgs84.tif` is the input raster and `aster_dem_utm.tif` is the output (resampled) raster. Inspect the output in QGIS.

Note that we did not use the `-ot` output data type flag, so `gdalwarp` used the `Int16` data type of our input raster. If you set `-ot Float32`, then you would see that the cubic resampling algorithm actually yields decimals instead of the integers that you see in the current output, but for our purposes that sub-meter level of precision is not needed.

Step 8: Search for the "Raster Calculator" tool. Double-click the "aster_dem_utm.tif" mosaic to add it to the expression, then click the multiply button "*", then double-click the"yellowhead_bma_utm.tif" raster to add it to the expression. Scroll down further and click the ellipse "..." next to the "Reference layer(s)" field, toggle on "yellowhead_bma_utm.tif", and then click "OK". Scroll down further and click the ellipse "..." next to the "Output" field to save the output as "aster_dem_utm_aoi.tif". Click "Run" and then inspect the output in your map.

Task 3: Calculating movement costs

In this task, we will calculate a few derivative rasters that are going to represent various costs for Grizzly Bears to move across the landscape. At the end of the task, we will produce a cost raster that will be one of the inputs to our least cost path model in the following task.

Step 1: Open your QGIS project and search for the "Slope" tool. Add "aster_dem_utm_aoi.tif" as the "Elevation layer" and save the output to "aster_slope_utm_aoi.tif". Inspect the output in your map.

Step 2: We are going to adjust the slope values to a range of 0 and 1. This is easily done by dividing all values by 90, which is the theoretical maximum slope in degrees. Open the "Raster Calculator" tool and type the expression `"aster_slope_utm_aoi@1" / 90`. Change the "Reference layer" to "yellowhead_bma_utm.tif", click "OK", and name the output "aster_slope_utm_aoi_cost.tif" then click "Run". Inspect the output in your map, you should now have values between 0 and 1.

Step 3: Open the "Data source manager", connect to the `bear` database on the UBC PostgreSQL server, and add the "ca_forest_vlce2_2019" raster to your map. This layer is a clipping of a national land cover dataset for Canada for the year 2019. The values of the raster refer to specific land covers. Open the properties for the layer. As you can see, it has a different coordinate system, resolution, and extent than our AOI. Export this raster as a GeoTiff to your local project folder and keep the output name the same.

Step 4: Open the OSGeo4W shell and navigate to your project folder. Run the `gdalwarp` utility again, parameterize it the same as before in Task 2, but change the `-s_srs` flag to `EPSG:3978`, the `-r` flag to `near`, the input file to `ca_forest_vlce2_2019.tif`, and the output file to `land_cover_utm.tif`. Inspect the output in QGIS.

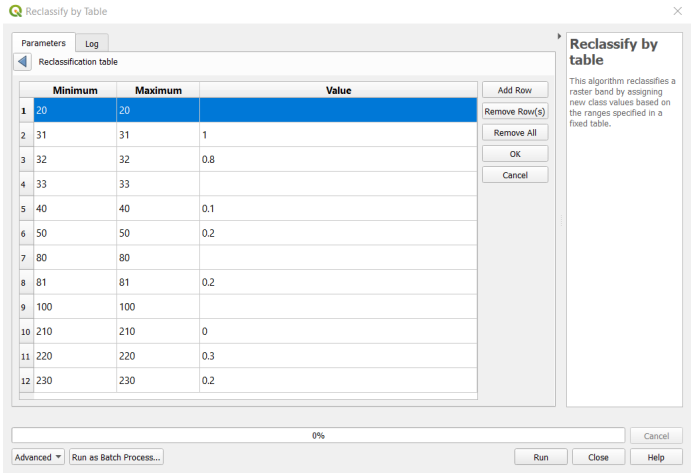
Q4. Why did we choose 'near' as the resampling method? (1 point) Step 5: Use the "Raster Calculator" tool to multiply "yellowhead_bma_utm.tif" by "land_cover_utm.tif" and write the output to "land_cover_utm_aoi.tif".

We want to use the land cover layer as an input to our cost surface, but the values of the land cover raster are codes, which are nominal data. These values cannot directly be compared as-is. For example, the code for water is 20 and the code for herbs is 100. These numbers do not imply that herbs is 5 times more costly than water, they are simply numerical placeholders for the concept of land cover. We need to define and then assign proper values for each land

cover that represents the cost or resistance for Grizzly Bears to traverse a cell of that land cover. We can do this by reclassifying the land cover raster. Below is a table indicating the land cover class codes and some proposed cost values scaled between 0 (no cost) and 1 (highest cost):

Code	Land Cover	Proposed Cost
20	Water	?
31	Snow/Ice	1
32	Rock/Rubble	0.8
33	Exposed Barren Land	?
40	Bryoids	0.1
50	Shrubs	0.2
80	Wetland	?
81	Treed Wetland	0.2
100	Herbs	?
210	Coniferous	0
220	Broadleaf	0.3
230	Mixedwood	0.2

Step 6: Open the "Reclassify by Table" tool. Select "land_cover_utm_aoi.tif" as the "Raster layer" and then click the ellipse "..." next to "Reclassification table". Populate the table with all the land cover codes. You will need to use some judgement and determine what would be reasonable values for the four missing codes in the table above. There is no right or wrong answer for these, but you will need to justify your choices. When you have filled in the table (see image below), change "Range boundaries" to "min <= value <= max", click "OK", and then name the output "land_cover_utm_aoi_cost.tif". Inspect the output in your map, you should have values ranging from 0 to 1.



Q5. What was your rationale for cost choices for classes 20, 33, 80, and 100? (4 points) The last layer we are going to add is distance to roads as a simple proxy for human-related mortality.

Step 7: Open "yellowhead_roads" in QGIS from the UBC PostgreSQL database, if you have not already. Open the "Reproject Layer" tool and change the projection for "yellowhead_roads" to EPSG:26911 and name the output "yellowhead_roads_utm.shp".

Step 8: In the OSGeo4W shell, enter the following command `gdal_rasterize -burn 1 -tr 20 20 -ot Byte -a_nodata nodata -te 401480 5753440 661680 5936300 yellowhead_roads_utm.shp yellowhead_roads_utm.tif`. You should recognize all these flags by now. Inspect the output in QGIS.

Step 9: Search for the "Proximity (Raster Distance)" tool. The input layer will be "yellowhead_roads_utm.shp". Name the output as "yellowhead_roads_distance_utm.tif" then click "Run" then inspect the output in the map.

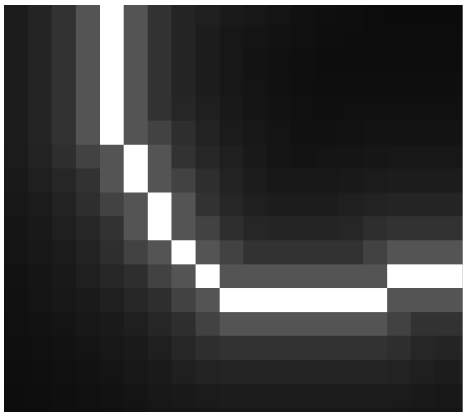
Step 10: Open the "Raster Calculator" tool and add the expression `(1 / ("yellowhead_roads_distance_utm@1" + 0.5)) / 2`. Let us break this one down:

- `1 / "yellowhead_roads_distance_utm@1"` would give us the inverse distances, however, cells that are roads have a distance of zero, and $1 \div 0 = undefined = nodata$. Therefore, we need to add a fraction to the denominator.

- `1 / ("yellowhead_roads_distance_utm@1" + 0.5)` avoids nodata values in the situation above, but now roads have a value of $1 \div 0.5 = 2$. We ultimately want these values to be between 0 and 1, with 1 representing roads.

- `(1 / ("yellowhead_roads_distance_utm@1" + 0.5)) / 2` dividing everything by 2 causes the 0.5 fraction to be removed from all cells, and road cells to be assigned a value of $2 \div 2 = 1$.

This way, the cost of traversing a roadway is 1 and decreases as you move away from the road. Change the "Reference layer" to "yellowhead_bma_utm.tif" and save the output as "yellowhead_roads_distance_utm_cost.tif". Inspect the output in your map and zoom in close to a road.



Now that we have three different cost layers, slope, land cover, and distance to roads, we simply need to decide how to combine them into a single cost raster. This is made easy by the fact that all layers are scaled between 0 and 1. The simplest approach would be to add them all together, but this assumes equal-weighting given to each cost factor. It is also common to weight layers differently and this can be achieved by multiplying a set of weights to the layers that add up to 1. For example, to weight slope 10%, land cover 30%, and road distance 60%, we would use the formula $totalcost = slope \times 0.1 + landcover \times 0.3 + road \times 0.6$. Note that $totalcost = slope \times 0.33 + landcover \times 0.33 + road \times 0.33$ is functionally the same as $totalcost = slope + landcover + road$. Differences in the percentage weightings are not absolute difference, but relative. For example, 30% weight to land cover implies that land cover is weighted 3 times more than slope at 10%: $0.3 \div 0.1 = 3 = 300\%$.

Step 11: Pick weightings for your three layers (the weights must sum to 1), then open the "Raster Calculator" tool and apply the formula above. Set the "Reference layer" to "yellowhead_bma_utm.tif" and save the output as "yellowhead_cost_raster.tif".

Q6. What was your rationale for selecting your weights? (2 points)

Q7. What additional or alternative layers could you have used for costs in this analysis? Give at least two examples and explain why they are pertinent. (4 points) In these last steps, we are going to create the point layers that will be used to trace the least cost path in the following task.

Step 12: Add the "grizzly_bear_core_access_management_area" to your map from the UBC PostgreSQL server. These represent the core habitat of Grizzly Bear across all regions. Select the Yellowhead unit and then search for the "Centroids" tool under "Vector Geometry", ensure "Input Layer" is set to "grizzly_bear_core_access_management_area", "Selected features only" is

toggled on, and save the output as “yellowhead_bma_centroid.shp”. Reproject this layer to EPSG:26911 and name the output “source.shp”.

Step 13: Now let us randomly choose a point in the secondary habitat area. Add the “grizzly_bear_secondary_access_management_area” and select the Yellowhead unit. Search for the “Random points inside polygons” tool, ensure “Input layer is set to”grizzly_bear_secondary_access_management_area”, “Selected features only” is toggled on, and save the output as “yellowhead_bma_target.shp”. Reproject this layer to EPSG:26911 and name the output “target.shp”. You can repeat this as many times as you want until you get a satisfactory location.

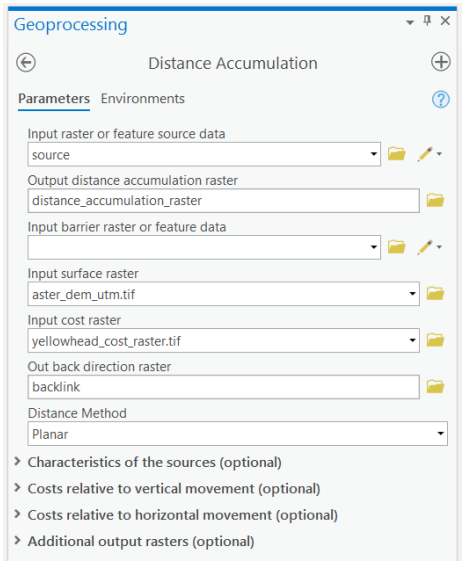
Task 4: Least cost path analysis in ArcGIS Pro

For this last task, we are going to perform the least cost path analysis in ArcGIS Pro. QGIS has a least cost path plugin, but it is not as robust as the ArcGIS Pro tools.

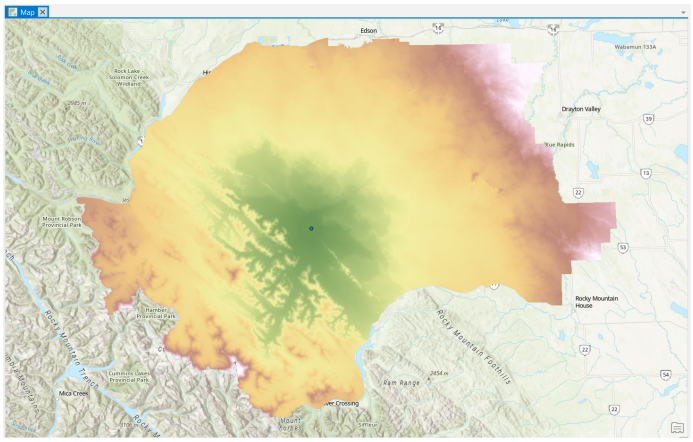
Step 1: Open a new ArcGIS Pro project and add the following layers to your map:

- Total cost raster: “yellowhead_cost_raster.tif”
- Aster DEM: “aster_dem_utm.tif”
- Slope raster: “aster_slope_utm_aoi.tif”
- Land cover raster: “land_cover_utm_aoi.tif”
- Road polylines: “yellowhead_roads_utm.shp”
- Yellowhead source: “source.shp”
- Yellowhead target: “target.shp”

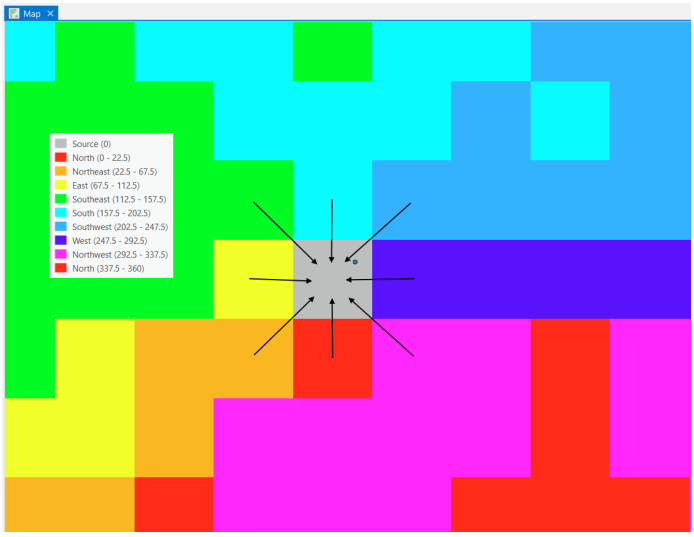
Step 2: Open the “Distance Accumulation” tool and parameterize it according to the image below. You can save the two output rasters (“distance_accumulation_raster” and “backlink”) in your project geodatabase or in your QGIS project folder as geotiffs, just be mindful that you cannot save a geotiff inside your geodatabase. This tool will take your source point and calculate the accumulative cost to travel to all other cells using a combination of the cost raster and the real geographic distance, both horizontal and vertical, which is why we supply the ASTER DEM.



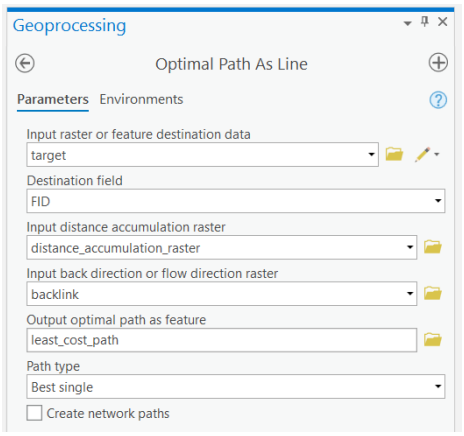
The "distance_accumulation_raster" shows us the accumulated cost-distance to reach any given cell in the AOI from the source point.



The "backlink" raster shows us the direction of the least costly path back to the source for any given cell. In other words, this raster is encoding the lowest directional movement cost for the 8-neighbours of each cell. Since the costs are accumulative relative to the source cell, this means that we can trace a path from any given pixel in the "backlink" raster back to the source cell. The image below shows a close-up for the source cell and illustrates how these values are encoded.



Step 3: Now we can trace the least cost path between our source and target locations. Open the "Optimal Path as Line" tool and parameterize like the image below then click "Run".



Step 4: Inspect the output. You should be able to trace the line along the "backlink" raster. Toggle on the roads. How many roads did you path cross? Toggle on the land cover. Were there any land covers that were avoided? Toggle on the slope. Did the path avoid any steep slopes or otherwise maneuver around the terrain?

Q8. Interpret the impact that your choice of weightings had on the final least cost path that you observed. (4 points) **Step 5:** Create a map of your least cost path that supports your interpretations in Q8. Export it as a pdf and upload to the course management system.

Summary

Raster analysis is comprised of manipulating base data, deriving new rasters, and weighting and combining rasters together in order to answer a spatial question. In this lab, you learned how to effectively set up a raster analysis through careful planning and choice of factors like raster extent, coordinate system, cell size, alignment, data type, and resampling algorithms. Rasters can lead to powerful spatial analyses like least cost path analysis due to the explicit encoding of information in the spatial neighborhood.

Return to the **Deliverables** section to check off everything you need to submit for credit in the course management system.

Chapter 6

Digitization and Editing with Kart

Written by Paul Pickell

Lab Overview

Version control is an important concept to software development that has recently been adopted and adapted for use in GIS. The ability to track data changes, manage versions, and collaborate on data editing are significant advances for managing geospatial data assets in an enterprise environment.

In this lab, you will learn about version control using both Git and Kart software. Git is a widely used distributed version control software system for tracking changes to files. Kart simply extends this distributed version control to geospatial vector data including points, lines, polygons, and point cloud datasets. You will also practice digitizing and editing datasets in QGIS and ArcGIS Pro.

Learning Objectives

- Practice using distributed version control software systems Git and Kart
- Digitize and edit geospatial features in QGIS and ArcGIS Pro
- Commit edits to a local repository
- Push edits to a local repository

- Manage merge conflicts and diffs between working copies of a repository

---

Deliverables

Answers to the questions posed throughout the lab (20 points)

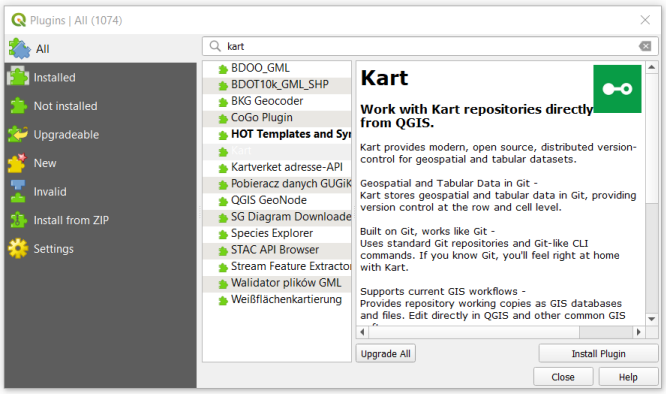
Map response to the question posed at the end of Task 4 (10 points)

Data

All data for this lab are accessible via the UBC PostgreSQL server, the public MGEM Data Store, and the public MGEM Geoserver. Instructions for connecting to the server and data store are given in the tasks below and prior labs.

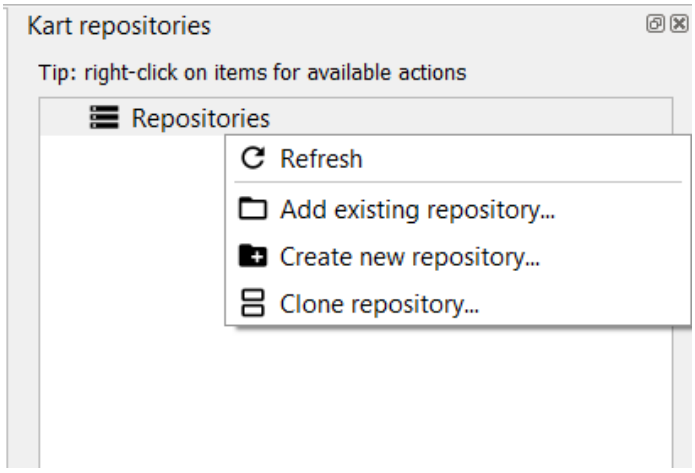
Task 1: Install Kart and create your first repository

Step 1: Open QGIS and install the Kart plugin using the Plugin Manager from the top toolbar. If you have not already installed the latest version of Kart on your computer, then you will be prompted to download and install the latest version.

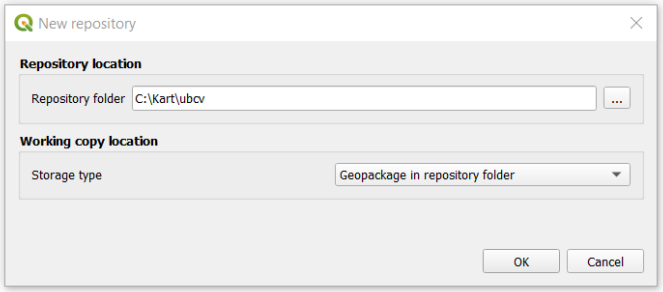


Step 2: Create a directory on your computer where you can store your first Kart repository, which we will call `ubcv`. For example, `C:\Kart\ubcv`.

Step 3: From the QGIS navigation bar at the top, select "Plugins" > "Kart" > "Repositories…". This opens the Kart repositories pane.

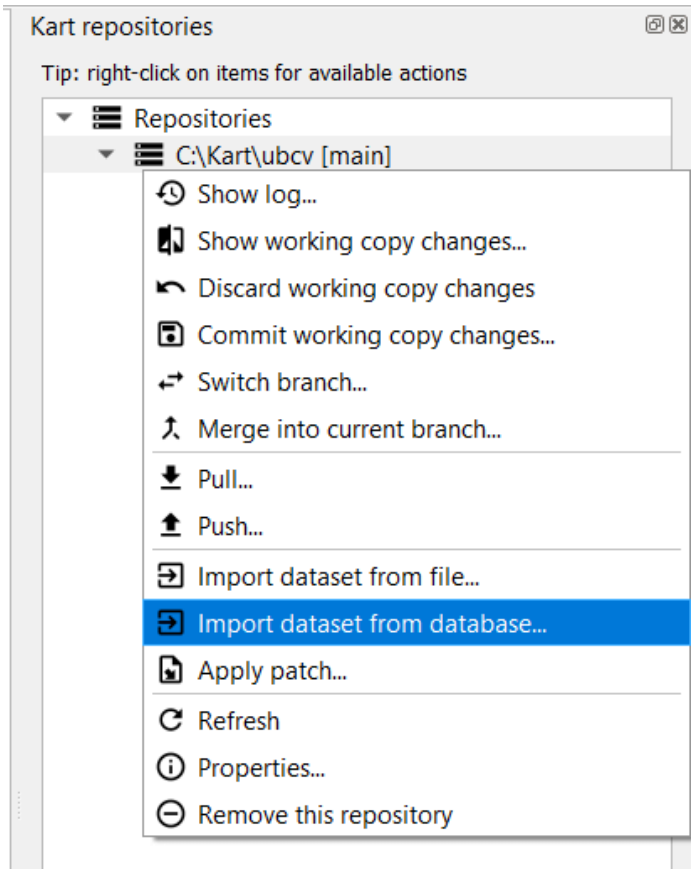


Step 4: From the Kart repositories pane, right-click "Repositories" and select "Create new repository…". Add the path to the directory you created in Step 2, ensure "Storage Type" is set to Geopackage and click "OK" to initiate the Kart repository.

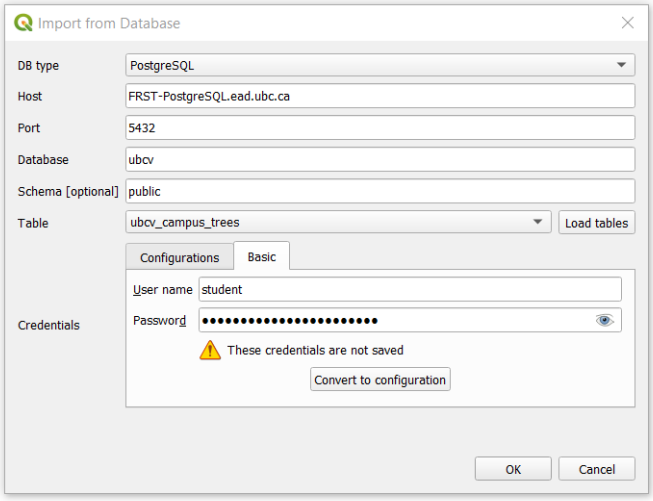


Expand your repositories list and you should see your new repostiory. If you expand "Datasets", you will see nothing there. What we have done is basically told Kart to watch this directory for any changes to the files. If you navigate to the folder in your file system, you will see some new files `KART_README.txt`, `.git`, and `.kart` that Kart has added, which enables version control in this new repository. Now let us add some data and visualize it in QGIS.

Step 5: Right-click the repository `C:\Kart\ubcv [main]` and then select "Import dataset from database…". We are going to connect to the ubcv database on the UBC PostgreSQL server.

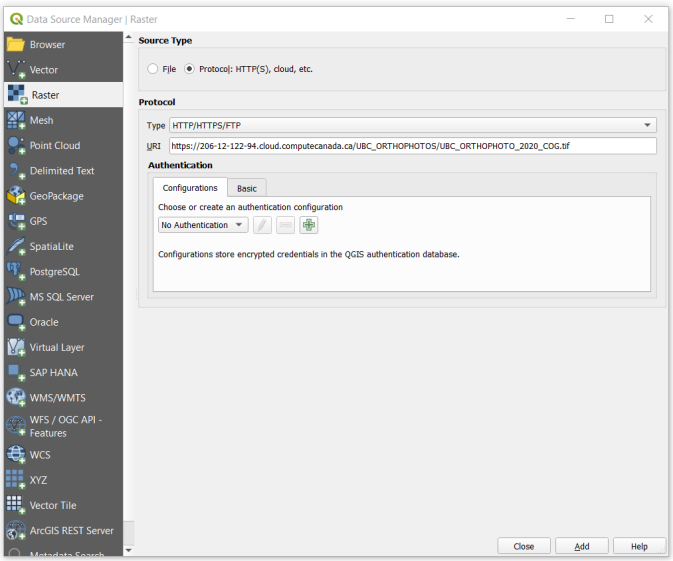


Step 6: Add the parameters to connect to the UBC PostgreSQL server. Be sure to switch to the "Basic" tab for credentials and add the student credential that was provided to the class. Once you have added the credential, click the "Load Tables" button to so a soft connection to the database. If the connection was successful, then select "ubcv_campus_trees" from the drop-down menu and then click "OK" to add the table to your Kart repository.



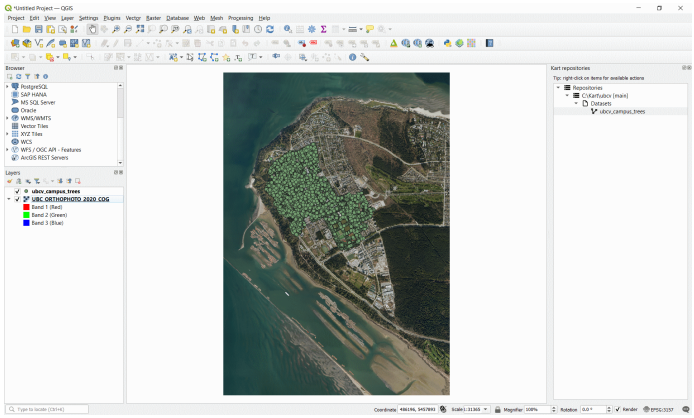
We will be comparing this point dataset to a recent orthophoto of UBC Vancouver campus and then making some edits.

Step 7: Add the most recent UBC orthophoto to your QGIS project. Navigate to the MGEM Orthophoto Data Store and then right-click the most recently dated "UBC_ORTHOPHOTO_YEAR_COG.tif" file listed there and then select "Copy Link". Return to your QGIS project, open the Data Source Manager, select "Raster" from the left navigation bar, change "Source Type" to "Protocol: HTTP(S), cloud, etc.", and then paste the link that you copied into the "URI" field and press "Add".



UBC orthophotos are publicly available datasets that are also encoded as Cloud Optimized Geotiffs (COG), which enables fast retrieval and zoom levels for cloud-hosted raster datasets. If you find that the file hosted on the server is not very responsive, you can also download the geotiff to your computer and add it as a regular file. Note that these orthophotos are large files (~7 GB) and ensure you have enough disk space in the location you want to save it.

Step 8: Drag and drop the "ubcv_campus_trees" dataset into your map from the Kart repository pane. Your QGIS project should look something like what is shown below.



Step 9: Zoom in to inspect the trees and the orthophoto.

Q1. What time of year do you think the orthophoto was collected? How do you know? (1 point)

Q2. Discuss three reasons why the campus trees do not perfectly match the locations in the orthophoto. (3 points)

Task 2: Edit a layer in QGIS with version control

Since the campus trees do not perfectly align with the locations in the orthophoto, we will practice editing some of them and then use the version control capabilities of Kart to inspect and manage the changes.

Step 1: From your QGIS toolbar, click the "Toggle Editing" icon to start an editing session.

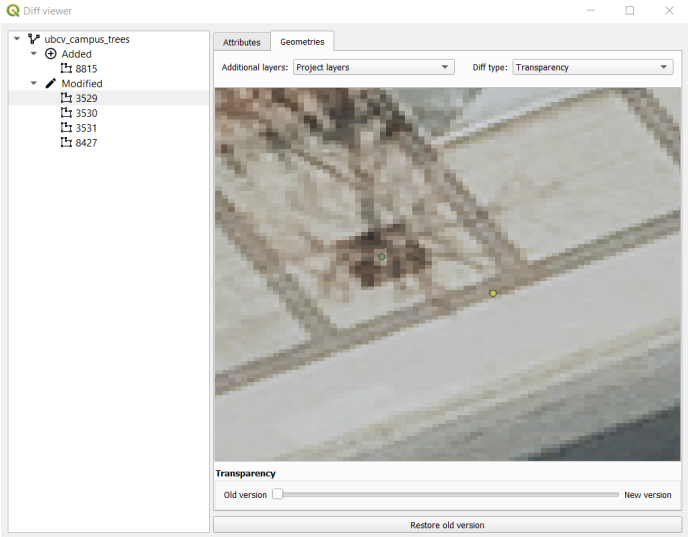
Step 2: Once an editing session is started, you will be given the option to add new points or edit the vertices of existing points. Click the "Vertex Tool" icon to edit the existing point dataset.

Step 3: Zoom into an area where you can easily see where some points are not perfectly intersecting with the bottom of the tree trunk in the orthophoto. In the example below, we are looking at some trees just outside the Forest Sciences Centre at UBC on Agronomy Road. Right-click on the point you want to edit to show its current coordinate and it will highlight red in the map. Then left-click the same point and your cursor will now have a red "x" to indicate you are ready to place the new coordinate. Left-click again to move the point to the location of your cursor at the base of the tree in the orthophoto. If you are not satisfied with the location, just left-click the point and then left-click again to place it where you want. Repeat this step for 3-5 more points.

Step 4: Add a missing tree by clicking the "Add Point Feature" icon. Your cursor will now have a cross-hair and you simply left-click again anywhere on the map canvas to add the new point.

Step 5: A dialogue window should appear that prompts you to add values for the attributes. Since we do not really know any of these values, just let Kart autopopulate most of the values. Scroll down to "notes" and add a short message to this field then click "OK". Add at least one point this way and then click the "Save Layer Edits" on the toolbar.

Step 6: Right-click the repository from the Kart repositories pane and select "Show working copy changes…". This will open the Diff viewer that allows you to inspect the changes you just made to the layer. In the screenshot below, you can see that we added one point and modified four others. You can click the "Geometries" tab to view the change you made to the geometry of the feature, which is really handy.



Step 7: Right-click the repository again and this time select "Commit working copy changes…". You will be prompted to enter a commit message. Commit messages help you and your collaborators understand what is happening in the changes that you just made. For this commit message, we are going to indicate that we edited four point locations and added one point.

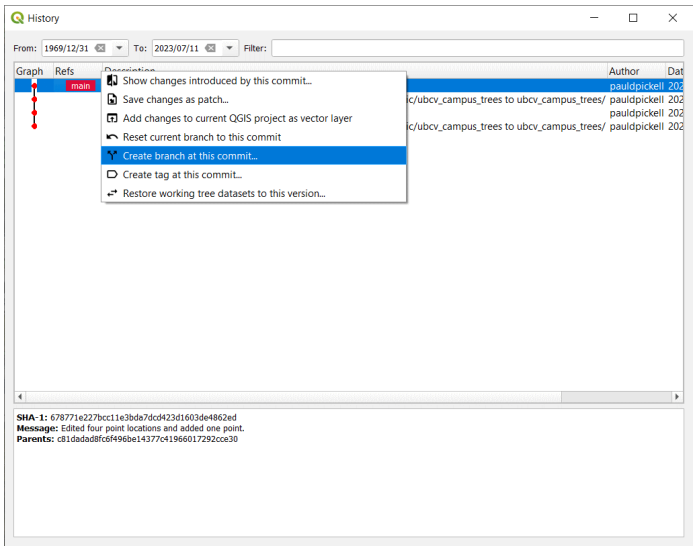
You can think of committing as a way of saving your work progress in your repository. Every commit can be inspected and/or reverted if needed, so it is good practice to commit changes that are related to each other so that the commit messages can be concise and informative. Avoid committing every single edit separately as this can make navigating and interpreting the commit history difficult.

Step 8: Right-click the repository again and this time select "Show log…". This will open the commit history for the repository and you should see the most recent commit you just made at the top along with the commit message you just entered in the last step. You can right-click any of the commits and select "Show changes introduced by this commit…", which is another way to inspect the changes to attributes and geometries of the features with the diff viewer.

Task 3: Working on and merging different branches

Step 1: Right-click the repository and select "Show log…" to display the commit history.

Step 2: Right-click the commit you made in Task 2 and select “Create branch at this commit…”. You will be prompted to enter a branch name, name it “mybranch”. You should now see both “main” and “mybranch” noted on the top commit.



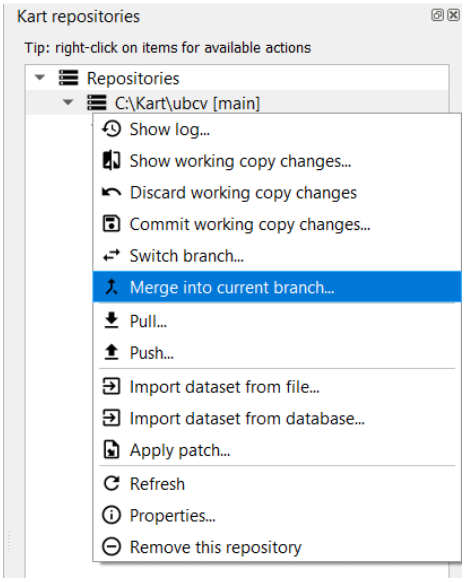
Step 3: Right-click the top commit again and select “Switch to branch ‘mybranch’…”.

Step 4: Now working on “mybranch”, start an editing session, pan around the orthophoto and add 5-10 more missing tree points. Do not change anything about the attributes, just click “OK” to dismiss after creating the point. Be sure to save your edits to the layer and then commit the change.

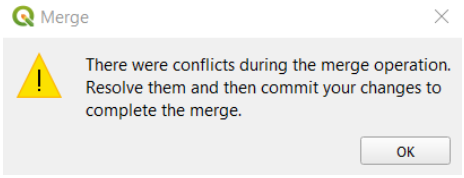
Step 5: Open the commit history of the repository. You will see now that “mybranch” is ahead of “main” by the commit you just made.

Step 6: Switch the branch back to “main” by right-clicking the commit tagged with “main” and then digitize 5-10 different tree points. Do not change anything about the attributes, just click “OK” to dismiss after creating the point. Commit your edits to the repository.

Step 7: Right-click the repository and select “Merge into current branch…”. In the dialogue window that appears, select “mybranch” from the drop-down menu for “Branch” and click “OK”. Since we are currently on the “main” branch, this will merge the edits/commits in “mybranch” into “main”.

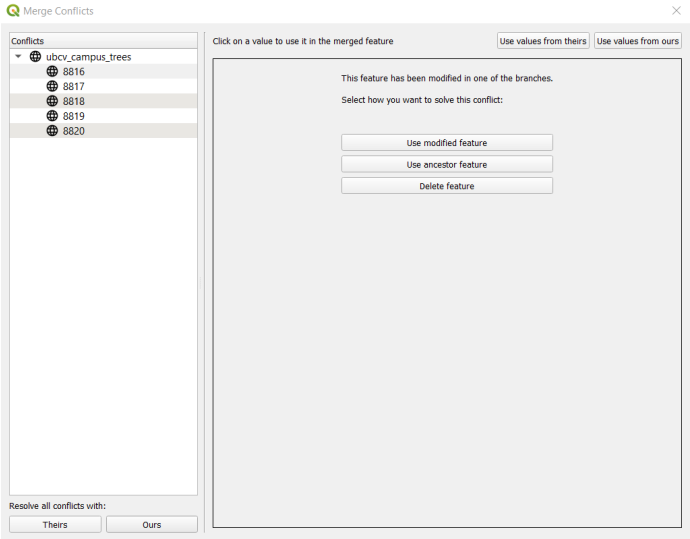


Step 8: You should see an error message appear warning you about a merge conflict between the two branches. Inspect the merge conflict by right-clicking the repository and selecting "Resolve conflicts...". Inspect the conflicting commits then answer the question below. You may need to close the Merge Conflicts window and inspect the ubcv_campus_trees attribute table to really appreciate what is going on here.

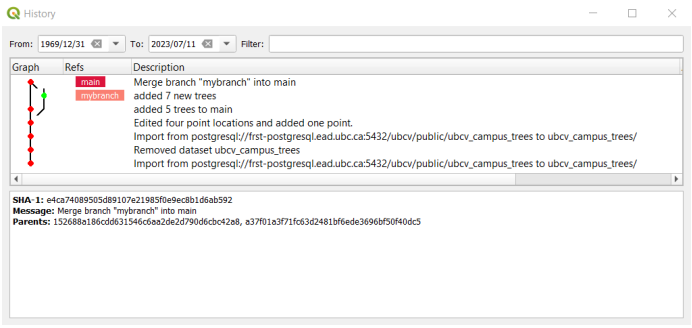


Q3. Describe why the merge conflict occurred. (3 points) **Step 9:** Once you have answered the question above, open the Merge Conflicts window again. Click each feature one-at-a-time listed in the left then click "Use modified feature". This is going to force Kart to recognize the most recent edits we made in "mybranch" and overwrite the conflicts in the "main" branch (our current branch). Once you have accepted all the modified feature edits, the Merge Conflict window will automatically close.

Q4. What is the difference between the "modified feature" and "ancestor feature"? (1 point)



Step 10: Open the commit history for the repository. You should now see the commit history between both branches, and we are now back to a single up-to-date branch ("main").



In this example, we simulated two different branches to explore these tools. We generally try to avoid merge conflicts rather than produce them intentionally! Branches are really helpful for managing collaborative work on different aspects of the same project.

For example, we can allocate specific work to a specific branch, like a "digitize-trees" branch might be for digitizing trees only and a "identifytrees" branch might be for changing the attribute values for the tree species of the points that are made from the other branch. Then that leaves the "main" branch as the merge point for all the work. In this way, it is possible to collaboratively edit a large dataset with multiple people without causing merge conflicts between branches.

Task 4: Digitize some buildings in ArcGIS Pro

Step 1: Start a new ArcGIS Pro project. This time, we are going to show you how to add the UBC orthophoto as a Web Map Service (WMS) from the MGEM Geoserver. A WMS is a protocol for exchanging map tiles over the internet and is especially good for serving image data like the UBC orthophoto. Both ArcGIS and QGIS support the WMS protocol, though we will illustrate how to do this with ArcGIS Pro in this section.

Step 2: Navigate to the MGEM Geoserver in your preferred web browser at the following URL: https://206-12-122-94.cloud.computecanada.ca/geoserver/web/. From the home page, select "Layer Preview" on the left panel under "Data". Navigate through the pages until you find the UBC orthophoto series listed. Find the most recently published orthophoto, then click the "OpenLayers" link to the right of it to show a preview of the data in your browser. You should be able to zoom in and out with exceptional speed because these are encoded as COGs.

Step 3: Return to the Geoserver homepage and on the right you will see a bunch of acronyms listed with different version numbers under "Service Capabilities". Find "WMS", right-click "1.3.0", and copy the link to your clip board.

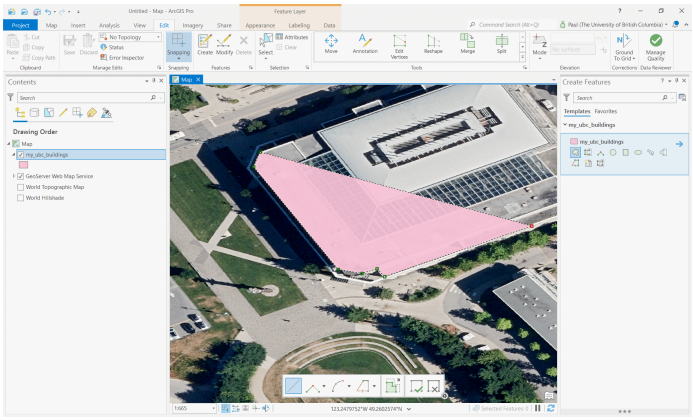
Step 4: Return to ArcGIS Pro, click the "Insert" tab, then click the "Connections" icon, and finally select "New WMS Server". In the dialogue window that appears, paste the link you just copied for the WMS into the "Server URL" field, leave everything else as default and click "OK".

Step 5: There are a few different ways to access the WMS resources in ArcGIS Pro. You can open a Catalog pane or window, expand "Servers", and continue expanding all the elements until you see the individual layers. Alternatively, you can click the "Add Data" icon on the "Map" tab, expand "Servers", and do the same. Add the most recent UBC orthophoto to your map as a WMS. If the orthophoto that you selected is unresponsive, try another year.

In the following steps, we are going to digitize a building on campus. To do so in ArcGIS Pro, we need to first create a new feature class that will hold the polygon we are going to digitize.

Step 6: In a Catalog window pane, expand "Databases" and then right-click the geodatabase for your project. Select "New" and then "Feature Class". Name it "my_ubc_buildings" and make sure the type is set to "Polygon". Click "Next" to view fields. Add two fields: "Name" (data type is text) and "Vertices" (data type is double). Click "Next" to view the spatial reference. Probably the default is set to WGS 1984. Under "Projected Coordinate System" find NAD 1983 UTM Zone 10N. You can continue clicking "Next" to view more properties that you can set, but we will leave these as the defaults, so you can click "Finish" when you are done. You should now have an empty polygon feature class in you map.

Step 7: Click the "Edit" tab and click the "Create" button to start creating new features. The "Create Features" pane will open. Click on "my_ubc_buildings" to highlight it and start the editing session. Now pick a building and zoom in so that you can clearly see its borders, then start left-clicking to add vertices along the building perimeter. Be sure to work either in a clockwise or counter-clockwise pattern otherwise you will criss-cross the boundary. Once you are happy, you can simply double left-click in the last position to finish the edit or press F2. Ta-da! You have now digitized your first building.



Step 8: At the top ribbon on the "Edit" tab, click "Attributes" to open the attribute editor for the selected feature. Change the name to the name of the building and click "Apply".

Step 9: Digitize four more buildings on campus and add the building name to the attribute table. Once you are done, be sure to click "Save" on the top ribbon under the "Edit" tab to save your work to the feature class in the geodatabase.

Next, we are going to validate and compare your digitized buildings against the official building dataset that is produced by UBC Campus + Community Planning.

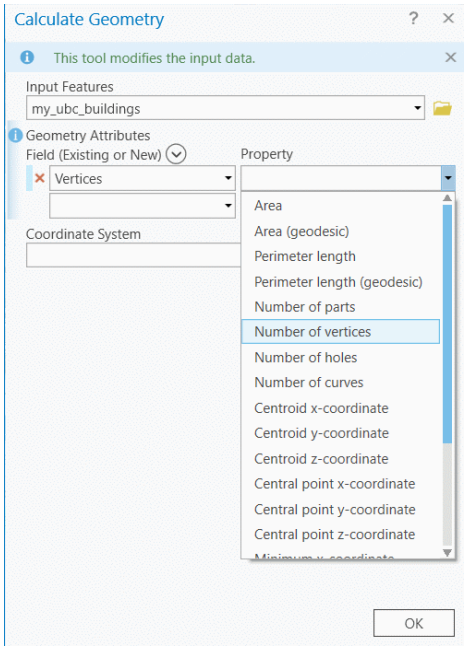
Step 10: Add the official "ubcv_buildings" data to your map from the `ubcv` database on the PostgreSQL server. For information about connecting to the server, refer to Lab 1. Since these data are read-only from the PostgreSQL server, we need to export the feature class to our local geodatabase so that we can make some calculations. We also need to change the coordinate system from WGS 1984 to our preferred projected coordinate system of NAD 1983 UTM Zone 10N. To do this, we can use a single tool "Project". Save the output with a name of "ubcv_buildings_utm" to your geodatabase.

Step 11: Add the local copy of "ubcv_buildings_utm" to your map and compare your digitization to the official record. Make some observations and answer the questions below.

Q5. Describe why your boundary does not align with the official dataset? Give three possible sources of error. (3 points)

Q6. What do you observe about the location of the official building dataset compared with the apparent location of the buildings in the orthophoto? (3 points)

Q7. How could relief displacement have impacted your digitization? (2 points) **Step 12:** Open the attribute table of the "ubcv_buildings_utm" layer and add a field called "Vertices" with double data type. Save the change then return to the attribute table, find the new field, right-click it, select "Calculate Geometry" and then under the "Property" drop-down, select "Number of vertices" and click "OK". Repeat this step for "my_ubc_buildings".



Step 13: Inspect the attribute tables of "ubcv_buildings_utm" and "my_ubc_buildings" and compare the "Shape_Area", "Shape_Length", and "Vertices" fields then answer the questions below.

Q8. Which building that you digitized had the largest deviation of Shape_Area compared with the official buildings dataset? Give the value difference in square meters and discuss why you think this building had the worst area error. (2 points)

Q9. Which building that you digitized had the largest deviation of Shape_Length compared with the official buildings dataset? Give the value difference in square meters and discuss why you think this building had the worst perimeter error. (2 points) Your last deliverable for the lab will be a mapped response to the following question: Did you digitize more or fewer vertices than the official UBC dataset? Discuss the implications of more or fewer vertices on positional accuracy, area accuracy, and perimeter accuracy. Your map should show a single building exemplar and should be annotated with supporting text and other elements (e.g., arrows, text boxes, etc.) to answer the question and illustrate your understanding of the relationship between digitization and accuracy measures.

Summary

By now, you should appreciate that digitization is an imperfect process. There are many sources of errors that can accumulate, especially when you are working on a crowd-sourced or collaborative editing project. It takes time to develop and practice good digitization skills. You might want to take a moment to reflect on the question, "which data set is right?" Invariably, the answer is the one with "authority".

Return to the **Deliverables** section to check off everything you need to submit for credit in the course management system.