Programming QGIS with Custom Python Expression Functions

Prof. Stefan Keller
Simran Khare

Geometa Lab
HSR Hochschule für Technik Rapperswil

1. Introduction

Expressions in QGIS

```
✓ Select all features where “feature_name” = ‘Zürichsee’ OR feature_name = ‘Walensee’
✓ Select all features where “feature_name” LIKE ‘%see’
```

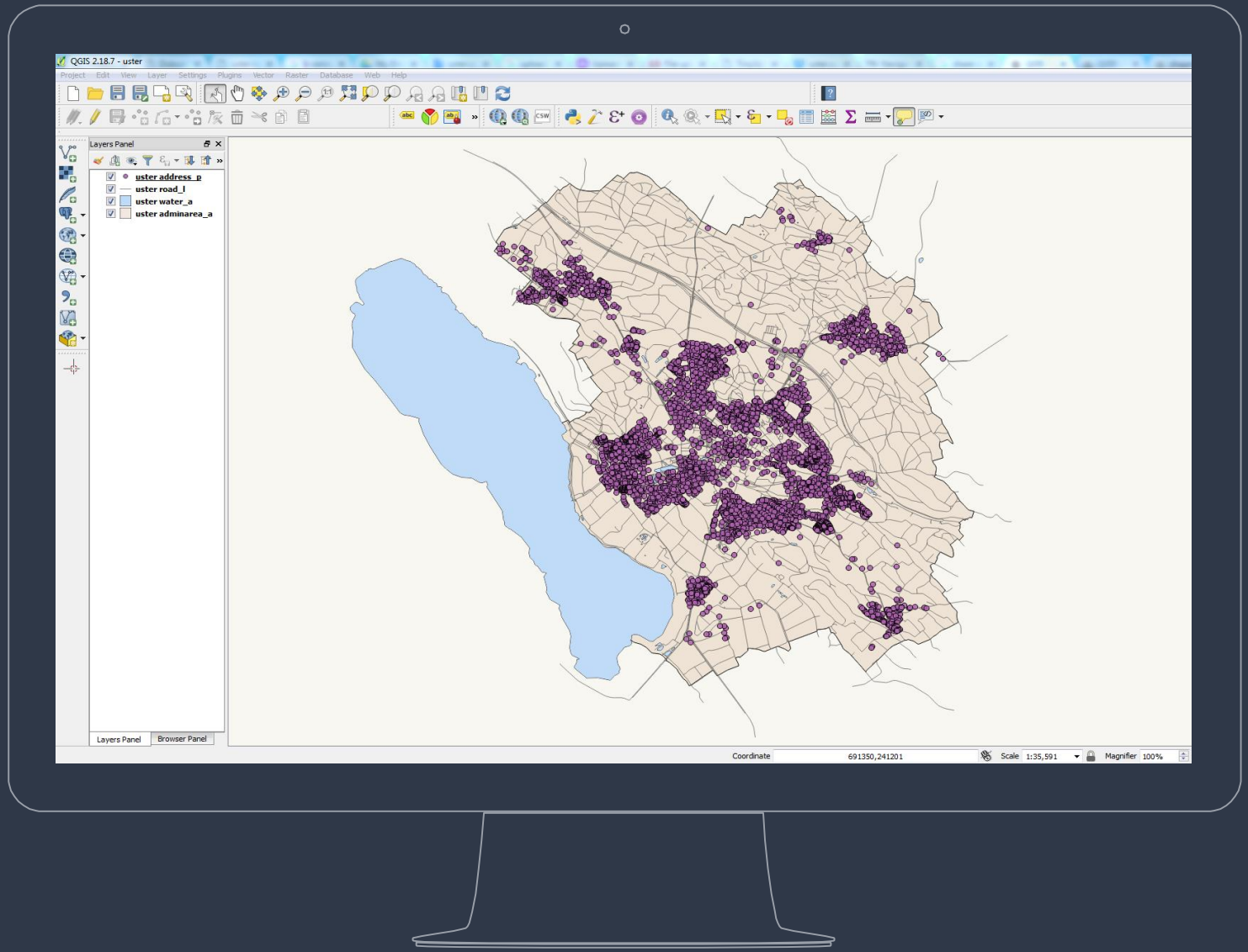
- QGIS supports expression based selections, filters, labels, field calculations, and much more.
- Look out for the expression symbol: 

Expression Functions (EFn)

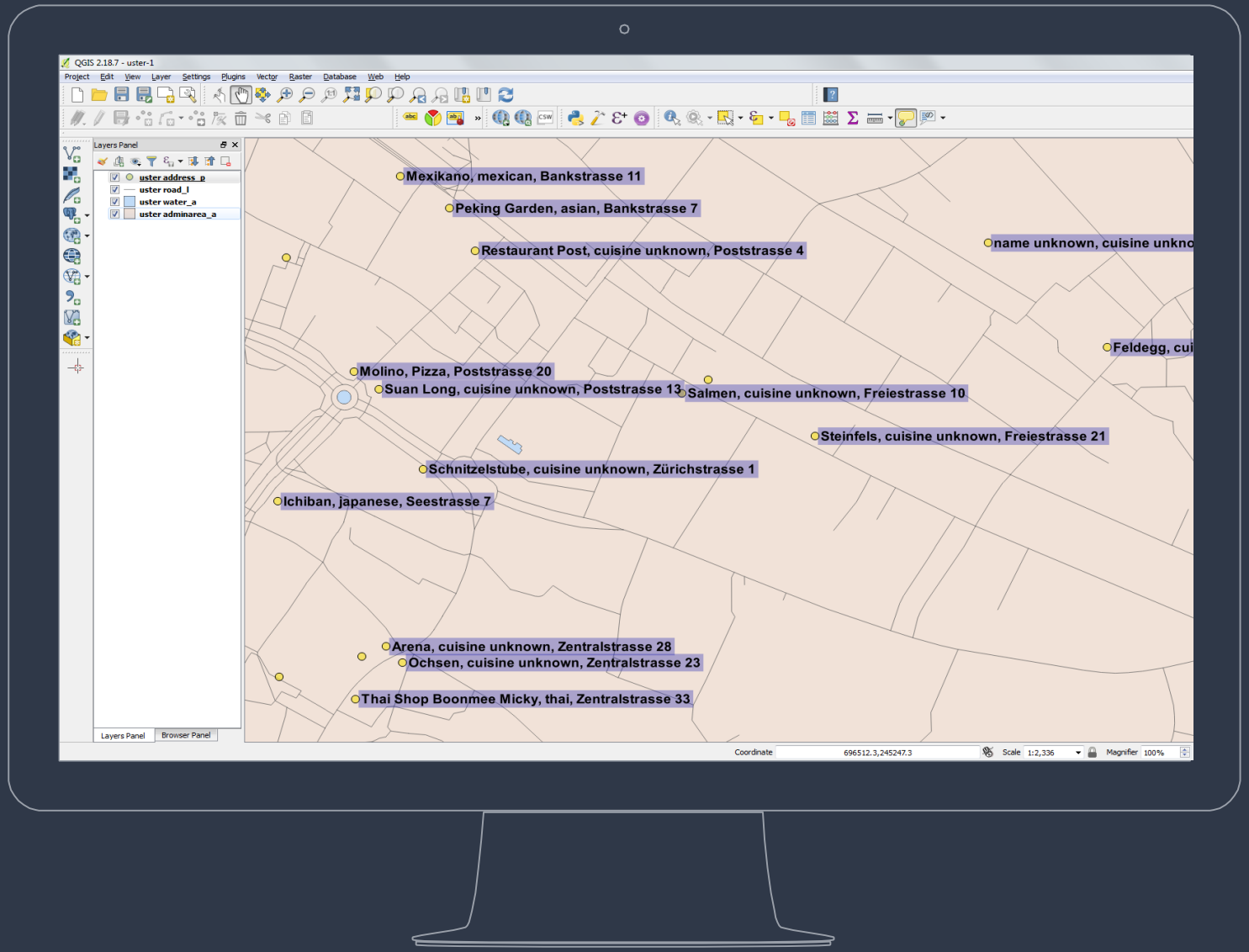
- Built in functions that can be used in QGIS expressions.
- Examples:

```
✓ Get the maximum population: maximum(“population”)
✓ Determine the hemisphere: CASE WHEN “latitude” > 0 THEN ‘North’ ELSE ‘South’
```

Q. What if we want to do something more interesting?



A. Write a Custom Python Expression Function for it



In this workshop

Using Custom Expression Functions for:

1. Feature Selection

2. Feature Labeling

3. Field Calculation

4. Recap with OSM Data

Syntax for Custom Python Expression Functions

```
@qgsfunction(args='auto', group='Custom', referenced_columns=['column_name'])
    def function_name(input_value, feature, parent):
        # Statements to be executed
        return return_value
```

✓ Every custom EFn must receive *feature* and *parent* as its last two parameters.

✓ The function must be preceded by the Python decorator: @qgsfunction.

- args: the number of arguments the function receives, excluding *feature* and *parent.* Defaults to ‘auto’.
- group: the group in which the function would be placed in the expression builder UI.
- referenced_columns: any columns or attributes that would be accessed within the function

2. Hands on Exercises

Getting Started

Prerequisite

Working installation of QGIS

- Latest: 2.18 (The future LTR)
- LTR: 2.14

- The code has been tested with QGIS 2.18, but should also work with QGIS 2.14.
- More information about the difference between versions 2.18 and 2.14 is available on Github.

Link: https://github.com/simran001/GeoPython-Workshop

Getting the code and sample data

- The code and workshop plan on Github:

  https://github.com/simran001/GeoPython-Workshop

  Short URL: https://goo.gl/ZaDxs6

- Dataset:

  Populated Places Simple from Natural Earth

  http://www.naturalearthdata.com/downloads/50m-cultural-vectors/50m-populated-places/

- Downloading the code:
  - Navigate to /python/expressions in the folder where QGIS is installed.
    - Linux and Mac: ~/.qgis2/python/expressions
    - Windows: %userprofile%/.qgis2/python/expressions

Task 1

Feature Selection with Custom EFn

Task 1.1. Select all capital cities with population greater than a user defined number.

- Dataset used: Populated Places Simple
- Function: is_populous_capital()

```
@qgsfunction(args='auto', group='Custom', referenced_columns=['featurecla', 'pop_max'])
    def is_populous_capital(input_pop, feature, parent):
        is_capital = feature['featurecla'] == 'Admin-0 capital'
        is_populous = feature['pop_max'] > 'input_pop'
        return is_capital and is_populous
```

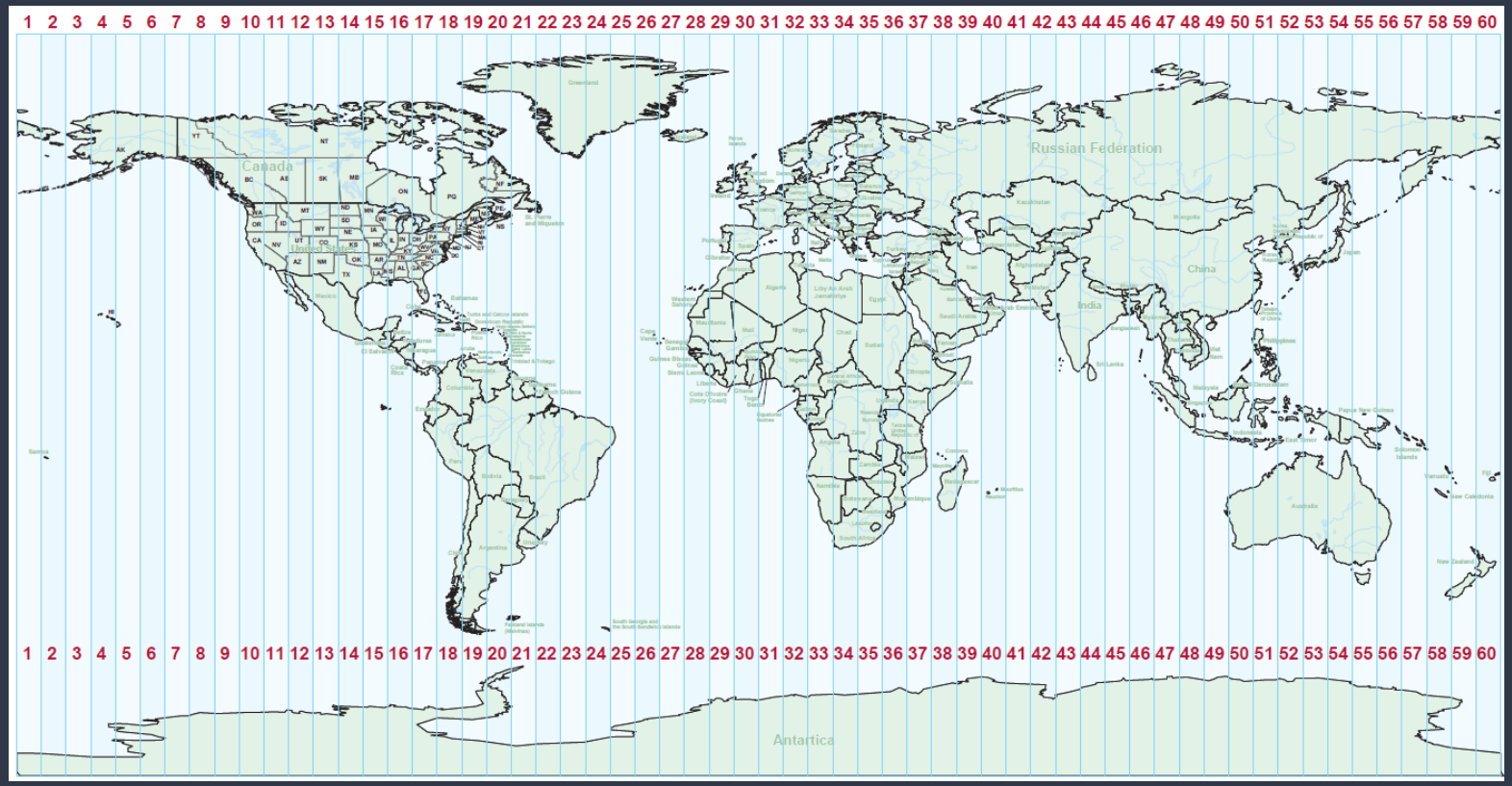
In the Expression tab, we can call the function as below to select all capital cities with a population greater than 2 million.

```
is_populous_capital(2000000)
```

Task 1.2. Select features based on the value of their calculated UTM Zone.

- Dataset used: Populated Places Simple
- Function: get_utm_zone()

UTM Zones In The World:



Task 2

Feature Labeling with Custom EFn

Task 2.1. Label all points as ‘City_Name: Population_Rank’.

- Dataset used: Populated Places Simple
- Function: get_population_rank()

Task 2.2. Display Map Tips as ‘City_Name: UTM_Zone: Population_Rank’.

- Dataset used: Populated Places Simple
- Function: get_utm_zone() and get_population_rank()

Task 3

Field Calculation with Custom EFn

Task 3. Writing an expression function to calculate a new ‘address’ field using reverse geocoding

Reverse geocoding generates an address from a given latitude and longitude. We will be using Nominatim’s reverse geocoding API, which is the search engine used for Openstreetmap data.

Parameters:

http://nominatim.openstreetmap.org/reverse?<query>

lat = <value> & lon = <value>

format = [xml | json]

The address will be found in the ‘display_name’.

Recap with OSM Data