Data integration memo, Case: Berry data from LUKE

by Tuomas Valtanen

The original berry dataset contains only few pieces of information: date of the observation of a berry, type of berry and its location. Some preliminary forest and tree type information also provided for categorical purposes (hue in seaborn). The location is in Finnish coordinate format, which is known as **ETRS89-TM35FIN (often called EUREF or just ETRS89 as well, since they are extremely similar in accuracy).** This can be a problem, since international systems use **WSG84** instead (Google Maps etc.), ***and they are not interchangeable.***

However, **it is possible to convert any ETRS89-TM35FIN –coordinate into a WSG84 –coordinate**. For example, this tool allows us to transform all the coordinates into international format, allowing us to integrate other data sources better:

https://fmnh-ws-prod.it.helsinki.fi/tipu-api/coordinates

This document contains integration ideas for your semester project while explaining also how the additional datasets that you can combine with your berry data have been built by your instructor.

Note: Programming the data download and manipulation logic for this use case from scratch is difficult and tedious in practice. Since this course is on the first year of your studies, it's not viable to do this by everyone in the course in order to proceed with the project. The programming skill level required for this operation is close to a professional developer with 3-5 years of work experience in the field of software development.

For what it's worth, this document also showcases perfectly why good programming skills are essential also in artificial intelligence: the world is full of data, but sometimes they can't be combined easily due to different accuracies, coordinate systems etc.

Long story short, the resulting csv-files are in Moodle for you to download. This document mainly explains how these new additional csv-files were built.

Idea 1: Integrate weather history information

By default, we have two approaches here: either use FMI data (Finnish Meteorological Institute), which is open and free to use, or try finding another weather data service that suits our project.

FMI data is extremely difficult to use. It's based on XML-data format, and it follows a certain Geographical data service standard called "INSPIRE". These kinds of data APIs are often used by GIS-software (advanced mapping software), which are more familiar to experts working with land surveillance or other geo-related data tasks.

Unless you are skilled with Python and XML, have ample amounts of time to go through FMI's data documentation and willing to "trial and error" with the data, using FMI's data on a tight schedule like this is not always advised. There also exists a Python module that allows an easier way to get FMI data, but it also needs learning in order to utilize it:

https://github.com/pnuu/fmiopendata

The second approach, is to use another service that provides easier JSON data for our needs. Most data APIs tend not to be free, and our purpose is not use commercial solutions that actually cost money.

Visual Crossing -service allows you to use the weather timeline API for free with limitations. The most essential limitation is that you can only have 1000 data queries per day. Our berry data has ~850 unique berry observations, so we can basically download all data we need within a day's limit. However, if we make a mistake, we have to wait another day to try our download code script again. See:

https://www.visualcrossing.com/resources/documentation/weather-api/timeline-weather-api/

The data in this API is very good and easy to understand, as far as it goes in the context of weather data. It also allows us to get historical data for a certain date and coordinate. Sample of the data on next page.

Data sample:


```
queryCost:          1
latitude:           63.84731663524114
longitude:          27.00001566231386
resolvedAddress:    "63.84731663524114, 27.00001566231386"
address:            "63.84731663524114, 27.00001566231386"
timezone:           "Europe/Helsinki"
tzoffset:           3
days:
  0:
    datetime:       "2020-07-22"
    datetimeEpoch:  1595365200
    tempmax:        15.5
    tempmin:        12.8
    temp:           14.2
    feelslikemax:   15.5
    feelslikemin:   12.8
    feelslike:      14.2
    dew:            14.2
    humidity:       99.5
    precip:         2.386
    precipprob:     100
    precipcover:    58.33
    preciptype:
      0:            "rain"
    snow:           0
    snowdepth:      0
    windgust:       38
    windspeed:      19.6
    winddir:        294.9
    pressure:       999.9
    cloudcover:     90.4
    visibility:     10.3
    solarradiation: 104.3
    solarenergy:    6.7
    uvindex:        3
    sunrise:        "03:51:38"
    sunriseEpoch:   1595379098
    sunset:         "22:43:00"
    sunsetEpoch:    1595446980
    moonphase:      0.06
    conditions:     "Rain, Overcast"
    description:    "Cloudy skies throughout the day with rain."
    icon:           "rain"
    stations:
```

Integration of this data required the following steps:

- Every berry observation in the original berry data needs also the WSG84 –coordinate (lat/long) based on the x and y –columns in the data (LUOMUS-service is helpful here, link in first page)

- After getting the converted lat/long –values, merge them with the original berry dataset

- Based on each berry observation, create the Visual Crossing Timeline API JSON Data URL based on **latitude, longitude** and **datetime** (year-month-day). This allows us to get weather information on each location on each timeframe in our data

- After the list of JSON Data URLs (~850 in total), query the API in a Python loop to download all the JSON data into separate files
    - Also approximately 850 data files after the download phase (matching the number of berry data rows)

- Combine the downloaded JSON files into a single JSON Array (collection), convert it into CSV-format, and save the result into a CSV-file
    - **Additional tricks used**: keep also the **datetime** and **name of berry** saved for each JSON file for easier row matching when merging DataFrames later

- In Jupyter Notebook, create two DataFrames: original berry data and the weather CSV data

- For both DataFrames, create a helper column called "latlon", which basically combines latitude and longitude into a single string, separated by a dash, for example:

  63.476942248701-25.193478894290

- Since the accuracy of the latlon –values are not identical in the berry data and weather API, some rounding down has to be done (into 12 decimals in this case) in order for them to match later when we combine DataFrames

- Finally, create a new DataFrame by combining the two DataFrames by using pandas merge and the common fields **berry, datetime** and **latlon**.

- **End result:** a single DataFrame that contains the original berry data + matching weather data for each berry observation

- **A copy of the cleaned weather data can be found in Moodle**

Idea 2: Integrate forest inventory data from LUKE

LUKE also has open data for forest inventory in this link:
https://opendata.luke.fi/dataset/national-forest-inventory-nfi-vmi12

The data is a huge Excel –file, but luckily it uses the **same coordination system as the original berry data.** There's just one problem: the accuracy is way more vague in the forest inventory data than in the original berry dataset. This means we have to once again round down the accuracy of the coordinates in order for the value to match. This, however, is not much of a problem, since we already have vague coordinates from LUKE in the berry dataset, so we don't lose much information even if we round down a little more.

Some of the column explanations (see full list at the end of this document):

A	B
Variables	Description
	Huom: kaikki tilavuustiedot ja biomassatiedot on laskettu koealakuviolle ilman korjausta kuvion osuudella koealasta
pk	ETRS-TM35FIN y-coordinate
ik	ETRS-TM35FIN x-coordinate, km
osite	sample area
lohko	sample area group index
koeala	sample number
kuvio	number of land plot
koko9	land plot portion in 9m sample area
koko564	land plot portion in 5.64m sample area
kunta	municipality number
maku	county
vv	measurement year
mluok	land class (forest, swamp etc.)
fra	land class by FAO
alar	growth location main type
kpty	growth location type
kptylm	growth location additional type
mlmuut	previous land type and land type additional information
ojtil	ditching condition
pt	wood production limitation, 1=no production, 2=limited production, 3=full production
maalaji	land type
jaksot	timber period count
khlk	development class
valli	dominant timber period's dominant tree type
ppa	timber area in the land plot
kasru	seedlings, amount of trees per hectar in land plot

Sample of actual data:

pk	ik	osite	lohko	koeala	kuvio	koko9	koko564	kunta	maku	vv	mluok	fra	alar	kpty	kptylm	mlmuut	ojtil	pt	maalaji	jaksot	khlk	valli	ppa	kasru	klpm
6654	102	0	1012	4	1	10	10	417	21	18	3	3	1	7	0	0	0	1	1	.	.		.	.	.
6672	118	0	2023	1	1	10	10	417	21	18	6	3	.	.		0	.	3	.	.	.		.	.	.
6672	118	0	2023	2	1	10	10	417	21	18	1	1	1	3	0	0	0	3	3	1	5	1	19	.	
6672	118	0	2023	3	1	10	10	417	21	18	1	1	1	3	0	0	0	3	3	1	5	1	17	.	
6672	119	0	2023	4	1	10	10	417	21	18	6	3	.	.		0	.	3	.	.	.		.	.	.
6673	120	0	3023	1	1	10	10	417	21	18	3	3	1	7	0	0	0	3	1	.	.		.	.	.
6673	120	0	3023	2	1	10	10	417	21	18	1	1	1	2	0	0	0	3	3	1	5	1	30	.	
6673	120	0	3023	3	1	10	10	417	21	18	5	3	.	.		0	.	3	.	.	.		.	.	.
6673	120	0	3023	4	1	10	10	417	21	18	1	1	1	2	0	0	0	3	4	1	1	0	0	0	.
6673	120	0	3023	5	1	10	10	417	21	18	1	1	1	4	0	0	0	3	3	1	4	1	17	.	
6673	107	0	4022	3	1	10	10	417	21	18	3	3	1	7	0	0	0	1	1	.	.		.	.	.
6673	107	0	4022	5	1	10	10	417	21	18	3	3	1	7	0	0	0	1	1	.	.		.	.	.
6674	110	0	5022	1	1	10	10	417	21	18	1	1	1	2	0	0	0	3	3	1	5	2	19	.	
6675	110	0	5022	5	2	1	0	417	21	18	3	3	1	7	.	0	.	3	.	.	.		.	.	.
6675	117	0	6023	1	1	9	10	417	21	18	1	1	3	5	0	0	4	3	0	1	4	1	14	.	
6675	117	0	6023	1	2	1	0	417	21	18	1	1	3	4	.	0	.	3	.	.	5	1	.	.	.
6675	117	0	6023	2	1	8	10	417	21	18	1	1	3	4	0	0	4	3	0	1	5	1	12	.	
6675	117	0	6023	2	2	2	0	417	21	18	1	1	3	3	0	0	4	3	0	1	5	4	14	.	
6675	117	0	6023	3	1	10	10	417	21	18	1	1	1	3	0	0	0	3	3	1	4	1	12	.	
6675	117	0	6023	4	1	10	10	417	21	18	1	1	1	3	0	0	0	3	3	1	4	3	14	.	
6675	117	0	6023	5	1	10	10	417	21	18	1	1	1	3	0	0	0	3	3	1	5	1	21	.	
6676	127	0	7023	2	2	4	3	438	21	18	3	3	1	7	.	0	.	1	.	.	.		.	.	.
6676	127	0	7023	3	1	5	5	438	21	18	1	1	1	3	0	0	0	1	3	1	5	1	14	.	
6676	127	0	7023	3	2	5	5	438	21	18	2	2	1	7	0	0	0	1	1	1	.	1	5	.	.
6676	127	0	7023	4	1	6	7	438	21	18	1	1	1	3	0	0	0	1	3	1	5	1	18	.	
6677	113	0	8022	1	1	10	10	417	21	18	2	1	1	7	0	0	0	2	1	1	.	1	5	.	.
6677	113	0	8022	2	1	10	10	417	21	18	1	1	1	4	0	0	0	3	3	1	5	1	30	.	
6677	113	0	8022	3	1	8	9	417	21	18	4	3	.	.		0	.	3	.	.	.		.	.	.
6677	113	0	8022	3	2	2	1	417	21	18	1	1	1	4	0	0	0	3	3	1	4	1	25	.	
6677	113	0	8022	4	1	10	10	417	21	18	1	1	1	3	0	0	0	3	3	1	5	1	20	.	
6677	113	0	8022	5	1	8	10	417	21	18	2	1	1	7	0	0	0	2	1	1	.	1	6	.	.
6677	113	0	8022	5	2	2	0	417	21	18	3	3	1	7	.	0	.	3	.	.	.		.	.	.
6680	110	0	9022	3	1	5	5	417	21	18	7	3	.	.		0	.	3	.	.	.		.	.	.
6680	110	0	9022	3	2	5	5	417	21	18	6	3	.	.		0	.	3	.	.	.		.	.	.
6680	110	0	9022	4	1	10	10	417	21	18	1	1	1	2	0	0	0	1	3	2	5	1	29	.	
6680	110	0	9022	5	1	10	10	417	21	18	1	1	1	3	0	0	0	3	3	1	6	1	28	.	
6679	108	0	10022	1	1	10	10	478	21	18	5	3	.	.		0	.	3	.	.	.		.	.	.
6680	108	0	10022	2	1	10	10	478	21	18	5	3	.	.		0	.	3	.	.	.		.	.	.
6680	108	0	10022	3	1	10	10	478	21	18	5	3	.	.		0	.	3	.	.	.		.	.	.

Integration of this data required the following steps:

- Convert the Excel-file's data sheet into CSV
    - Some changes needed to be made, like converting commas to periods for better decimals in pandas

- Load both berry data and forest inventory CSV data into two DataFrames
    - Add a combination column for x and y –values for both DataFrames called "xy", separated with a dash, for example:

      410000-7040000

        - Before this combination, round both x and y down a little bit, so that the berry data's **xy** matches with the **xy** in forest inventory data

- Group forest inventory data by xy and use mean() –function for all values to remove duplicates
    - This has resulted in some minor problems with data, for example wood production should be 1, 2 or 3, now it can be 1.5 because of using mean()

- Finally, create a new DataFrame by combining the DataFrames by using pandas merge and the common fields **berry, datetime** and **xy**.

- **End result:** a single DataFrame that contains the original berry data + matching forest inventory data for each berry observation

- **A copy of the cleaned forest inventory data can be found in Moodle**

Idea 3: Combine all datasets into one

This will result into a large dataset column-wise: 856 rows and 60-80 columns.

This of course makes analysis tasks quite difficult, since there is so much information to work with. You can decide which columns to keep or combine. The idea here is to have possibilities for your ideas; most of the columns most likely don't provide anything interesting to the analysis.

Integration can be done similarly with **location data, datetime** and **berry name data**. End result is one large DataFrame with many columns.

Code examples on how to combine these datasets in Moodle.

Inspect the data and feel free to ask questions from your instructor!

Idea 4: Other ideas

Any other ideas for new data that could be helpful, pollution or something similar?

Appendix: Full description for the forest inventory data

Variables	Description
pk	ETRS-TM35FIN y-coordinate
ik	ETRS-TM35FIN x-coordinate, km
osite	sample area
lohko	sample area group index
koeala	sample number
kuvio	number of land plot
koko9	land plot portion in 9m sample area
koko564	land plot portion in 5.64m sample area
kunta	municipality number
maku	county
vv	measurement year
mluok	land class (forest, swamp etc.)
fra	land class by FAO
alar	growth location main type
kpty	growth location type
kptylm	growth location additional type
mlmuut	previous land type and land type additional information
ojtil	ditching condition
pt	wood production limitation, 1=no production, 2=limited production, 3=full production
maalaji	land type
jaksot	timber period count
khlk	development class
valli	dominant timber period's dominant tree type
ppa	timber area in the land plot
kasru	seedlings, amount of trees per hectar in land plot
klpm	dominant timber period, average diameter of trees, cm, within land plot
kpituus	dominant timber period, average height of trees, dm, within land plot
ika	dominant timber period, average age of trees, within land plot
kokru	dominant timber period, total amount of trees, within land plot
laatu	wood quality
thakm	logging done or not
thakam	datetime of logging
vu	living trees, volume in sample area land plot, m3/hectar
vuma	pine trees, volume in sample area land plot, m3/hectar
vuku	spruce trees, volume in sample area land plot, m3/hectar
vuko	birch trees, volume in sample area land plot, m3/hectar
vuml	other leafy trees, volume in sample area land plot, m3/hectar
bm1	living trees, biomass for trunk, branches, needles: thousands/hectar
bm2	living trees, biomass for stump and roots over 1cm
bm1ma	pine trees, biomass for trunk, branches, needles: thousands/hectar
bm2ma	pine trees, biomass for stump and roots over 1cm
bm1ku	spruce trees, biomass for trunk, branches, needles: thousands/hectar
bm2ku	spruce trees, biomass for stump and roots over 1cm
bm1ko	birch trees, biomass for trunk, branches, needles: thousands/hectar
bm2ko	birch trees, biomass for stump and roots over 1cm
bm1ml	other leafy trees, biomass for trunk, branches, needles: thousands/hectar
bm2ml	other leafy trees, biomass for stump and roots over 1cm
n_l	sample area land plot, amount of measured living trees

You can rename the columns as you wish in pandas, they are abbreviations in Finnish mostly.