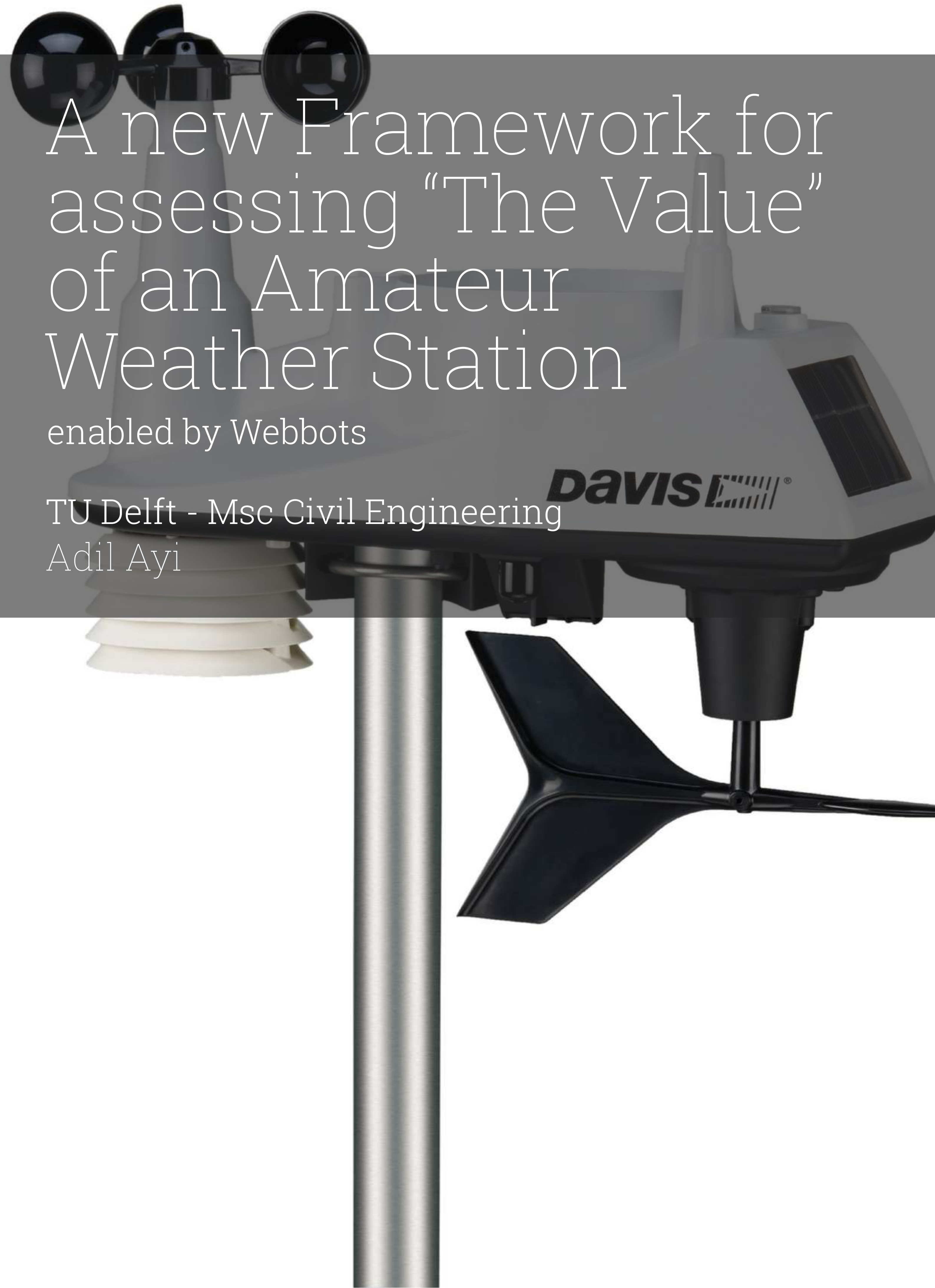


A new Framework for assessing "The Value" of an Amateur Weather Station

enabled by Webbots

TU Delft - Msc Civil Engineering
Adil Ayi

A new Framework for assessing "The Value" of an Amateur Weather Station

enabled by Webbots

by

Adil Ayi

Student Name	Student Number
Adil Ayi	4720458

Chair and main supervisor:	Dr. M.A. Schleiss
Committee member:	Dr. T.A. Bogaard
Committee member:	Dr. R.C. Lindenbergh
Project duration:	November, 2022 - July, 2023
Faculty:	Faculty of Civil Engineering and Geosciences, Delft

Cover: Davis Vatage Vue by Davis Instruments under CC BY-NC 2.0

TUDelft

Preface

The journey of this thesis began with a fascination for Personal Weather Stations (PWS) and their potential to contribute valuable data to weather and climate observations. Personal weather stations, owned and operated by individuals or organizations, have emerged as a decentralized network of weather measurement instruments. However, the assessment of the value and reliability of these stations has been a challenging task due to various factors such as metadata organization, station uniqueness, device setups, network availability, and credibility.

As I embarked on this research endeavor, my aim was to conduct the first-ever national assessment of the Dutch PWS network, specifically focusing on the WOW-NL network and its 1031 weather stations. I was driven by a desire to unravel the complexities surrounding the evaluation of PWS networks and to explore their untapped potential for enhancing spatial resolution in weather and climate observations.

Throughout this thesis, I delve into the intricacies of metadata organization and its impact on assessing the 'Value' of a weather station. I examine the uniqueness of stations in terms of their spatial distribution and location characteristics, shedding light on the concentration of PWS stations in urban regions. I also delve into the analysis of device setups and their significance in ensuring high-quality station output.

Furthermore, I explore the availability of the network over time, uncovering the challenges associated with obtaining consistent and reliable data. The importance of unbiased evaluation and credibility assessment is emphasized, particularly in stations managed by esteemed organizations.

The research presented in this thesis would not have been possible without the support and guidance of my advisors and the valuable contributions of researchers in the field. I am grateful for the opportunities I have had as a Master Student at the TU Delft to work with Dr. M.A. (Marc) Schleiss, whose involvement has provided further insights into the complexities of PWS networks.

As I reflect on the journey, I am filled with gratitude for the advancements in technology that have made it possible to access and analyze vast amounts of data. Web bots, in particular, have played a significant role in extracting and generating data, providing invaluable resources for geospatial analysis and assessing the value of PWS stations.

It is my hope that this thesis will contribute to the growing body of knowledge in the field of weather citizen science, stimulate further research and development, and inspire new approaches for leveraging citizen-generated data in weather and climate observations.

I extend my sincere thanks to all those who have supported me throughout this research endeavor, and I invite readers to embark on this journey with me as we explore the value of the Dutch PWS network and its potential for advancing our understanding of weather and climate phenomena.

Adil Ayi
Delft, July 2023

Abstract

This thesis explores the assessment of the value of Personal Weather Stations (PWS) in the Dutch context. A framework comprising five values—Device Setup, Uniqueness, Availability, Credibility, and Accuracy—was developed to quantify the value of PWS stations. The research involved analyzing data from 1031 Dutch weather stations as a first ever national analysis of a weather citizen science network. We overcame various obstacles, providing valuable insights for future developments to improve weather citizen science networks.

By utilizing a web bot we created called "the WOWbot", a national PWS network analysis was conducted, extracting information from the WOW-NL network. The analysis of the existing metadata structure revealed challenges and ambiguities that require attention. The current rating system is dependent on three variables (location and two sensor classifications) and was found to be unreliable due to the facts that $\sim 40\%$ of the time these variables are unknown for stations in the Netherlands. Also, when the data is known, subjective assessments of the classification in the metadata description makes it prone to misunderstandings and wrong inputs by the citizens, this means even if there is data we can't trust it fully. Recommendations were made to address these issues and move towards a more objective assessment of station value in chapter 5.

Based on relevant literature and findings from our analysis of the current metadata system, a new framwork to assess the value of a personal weather station was made. In chapter 6 the framework is immediatly used to assess the value of all personal weather stations in the Netherlands, showcasing the usefullness but also an example of how to utilize the framework. Except 'Accuracy' which has been reseached in depth in other papers, we have quantified each value category. For example the most "Unique" stations in the netherlands have been determined based on a Nearest Neighbor Analysis utilizing a cutoff value in the distribution. Therefore succesfully quanitfying which stations are statistically more spatially exclusive then others. For the more in depth description of how the framework has been used for assessing the value of the weather stations in the Netherlands we refer to chapter 6.

Overall, this thesis contributes to the advancement of the weather citizen science field by providing insights into assessing the value of personal weather stations. Continued research should focus on resolving identified issues, refining assessment criteria, and enhancing metadata quality to unlock the untapped potential of PWS networks for weather and climate observations. A list of improvements to actualize this change can be found in the conclusion in chapter 7.

Contents

Nomenclature

Abbreviations and Definitions used

Abbreviations

Abbreviation	**Definition**
AWS	Amateur Weather Station
PWS	Personal Weather Station
UI	User Interface
HTML	HyperText Markup Language (Coding Language for a Website)
WOW	Weather Observation Website
CWOP	Citizen Weather Observer Program
KNMI	Koninklijk Nederlands Meteorologisch Instituut
LGN	Landelijk Grondgebruik Nederland
API	Application programming interface
KDE	Kernel Density Estimation
UTC	Coordinated Universal Time
Metadata	a set of data that describes and gives information about the context of other data

1 Introduction

A personal weather station (PWS) is a set of weather measuring instruments that is owned and operated by an individual, club, association, or business [1]. These stations are often less advanced than traditional weather instruments and can include sensors to measure a variety of weather conditions, such as wind speed, wind direction, temperature, humidity, barometric pressure, and rainfall. The quality and placement of personal weather stations can vary, making it difficult to determine which stations provide accurate and comparable data.

Personal weather stations typically include a digital console that displays the collected data, and some may be connected to a personal computer for storage and sharing. Many personal weather station owners share their data with others by uploading it to websites or distributing it through amateur radio. The Citizen Weather Observer Program (CWOP) and Weather Underground are popular platforms for sharing personal weather station data. These services allow personal weather station owners to submit their data, which is then used by organizations such as the National Weather Service (NWS) to improve forecast models [22]. In Western Europe, the WOW network is a well known source of in situ weather data. The Dutch version of this network is called WOW-NL and is managed and maintained by the Royal Netherlands Meteorological Institute (KNMI).

With these networks in full effect and still growing [14], the potential to use PWS data for weather forecasting or research purposes has gained popularity in the scientific field for its potential to "fill data gaps" [21]. A "Spatial Quality Control" procedure before using the data has also been proposed in research by the American Meteorological Society [22] in the context of using the data to improve weather forecasts. One of the big findings is how "Quality Control sometimes fails to identify the reason for systematic errors caused by ill-chosen locations of citizen weather stations". For example, a station placed too close to a building can exhibit diurnally varying biases that are difficult to detect by analyzing the time series of observations alone. As metadata are rarely available, a black list of stations known to be poorly located cannot be created. What if we are able to generate this metadata based on the data that is now widely available via the networks on the internet?

Availability, from a researcher's perspective, the PWS networks are an invaluable source of information. But one of the limitations of these platforms is the export limitations. For example, in the KNMI's WOW-NL network only a single station can be looked at, at a time. The time series is available, but only for the time you specify in the portal. The value for a researcher is in looking at the full dataset, from different stations and analyze the spatial distribution to name an example.

An Internet bot, also known as a web robot or simply a bot, is a software program designed to automate certain tasks on the internet. These tasks can include imitating human activity, such as sending messages or interacting with websites [12]. Internet bots are able to carry out these tasks, which are often simple and repetitive, much faster than a human could. For example, querying different PWS's and gathering, and more importantly, generating data about it. The Bot operate in a client-server model, with the bot functioning as the client and web servers playing the role of the server.

Therefore also doing some logic, like looking at the temporal range and resolution telling something about the age of the weather station, is possible. The bot can therefore gather useful information about the stations in the network, which can be used within a Geospatial framework to assess the value of stations. Establishing this framework is what the thesis will be about.

The use of Bots to access public data has been a point of discussion for a long time. In 2019 the “Journal of business, Entrepreneurship and the Law” published an article where The inherit right to be able to access publicly available information has been highlighted [25]. With the growing datasets that become available and the rise of “Big Data”, data collection becomes more of an issue. The article stated: “As the number of available datasets and the hunger for new data insights grow, online platforms increasingly seek to protect their information from web scrapers. Unfortunately, these online platforms do not always successfully distinguish bad from good actors, thus risking stomping out new innovation and valid competition.”. This thesis is a manifestation of this, a geospatial analysis of PWS networks has not been done before but to get the data to do this is almost impossible. Therefore, hindering the growth of the field.

In this thesis, we create a framework to assess the “Value” of weather stations in a citizen network. We begin with collecting a national dataset of weather stations to analyze the current state of the Dutch network. After, we project our framework on all stations in the Netherlands, assessing the stations on value on a national scale. Next to that, we investigate the opportunity Web Bots gives researchers to analyze PWS networks. Similar things have been done in the UK by Simon Bell from Aston University in his PhD for a small area and with a web scraper. He stated there was value in “allowing us to summarize the volume of information available” [3].

2 Objectives

There are two objectives that we focus on in this thesis. Foremost, creating a robust definition of "Value" for a Personal Weather Station. Like Peter Ackroyd wrote in the detective novel Chatterton: "The value is always in the eye of the beholder. What is worthless to one person may be very important to someone else" [2]. From what perspective do we look at value in this paper? To determine the perspective, we first must understand the goal we want to reach. The framework will be aid into assessing the value of a personal weather station from a retrospective point of view. We look at the whole station, its context, the area it is active, what it is measuring and who is managing it. This wholistic approach is what can really determine the value of PWS. To be able to have this retrospective view we must, approach the problem from different angles (scientific, practical and user-centered).

The second objective is to understand the way bots can help researcher gather PWS data from the web. Currently, most websites featuring PWS data are run by Meteorological Organizations. The data in the network is publicly created, citizen data, but get unintentional privatized by organizations that manage them. This creates a big barrier for researchers to start using these networks since it's not fully open sourced. The problem now is that either a log-in is required to download the data or that the website does not support batch exportation, making it almost impossible to download regions of PWS data. By showing a way to create a bot to gather bypass this inconveniences, we can validate a way to remove these barriers in the research field. We are now able to investigate stations on a bigger spatial scale. With the scraping tool, one can merge data from different networks and have a national or even global database. Most publications so far were limited to single provinces and/or small study regions because of the dependency on the use of one network.

So essentially, there are two main reasons for using a bot. First, to automatically gather large amounts of PWS data that would otherwise be difficult or time-consuming to get. Second, to automatically gather and process metadata that can be used to assess the value of a station. Next to that, the bot is the only way to extract large databases from citizen data networks (at least for the public). It is sad that the data gathered by volunteers and enthusiasts is in the hand of companies/organizations which do not really make it easy for the public to actively use them.

Research Question: How to create a framework that can assess the "value" of an Amateur Weather Station in a Network

1. How do current PWS networks quantify the value of a PWS station? And how good or reliable is this rating system?
2. How valuable are the personal weather stations in the Netherlands based on our framework?

3 Background

3.0.1. WOW-NL

WOW-NL (`https://wow.knmi.nl/`) is a platform that allows individuals to participate in a large network of weather observers, where users can share, view, compare, and archive current weather observations. It was developed by KNMI in collaboration with the Met Office, and is part of the Weather Observations Website (WOW) concept. Over 10,000 stations in 220 countries are part of the network, including those of the KNMI. Anyone with an automatic weather station connected to the internet can participate in WOW-NL by registering their station on the WOW-UK website and transmitting data. The data are updated every 10 minutes and are available on WOW-NL for everyone to view. The potential benefits of WOW-NL include insights into extreme weather trends, knowledge about urban climates, and more detailed advice about the impact of weather [17].

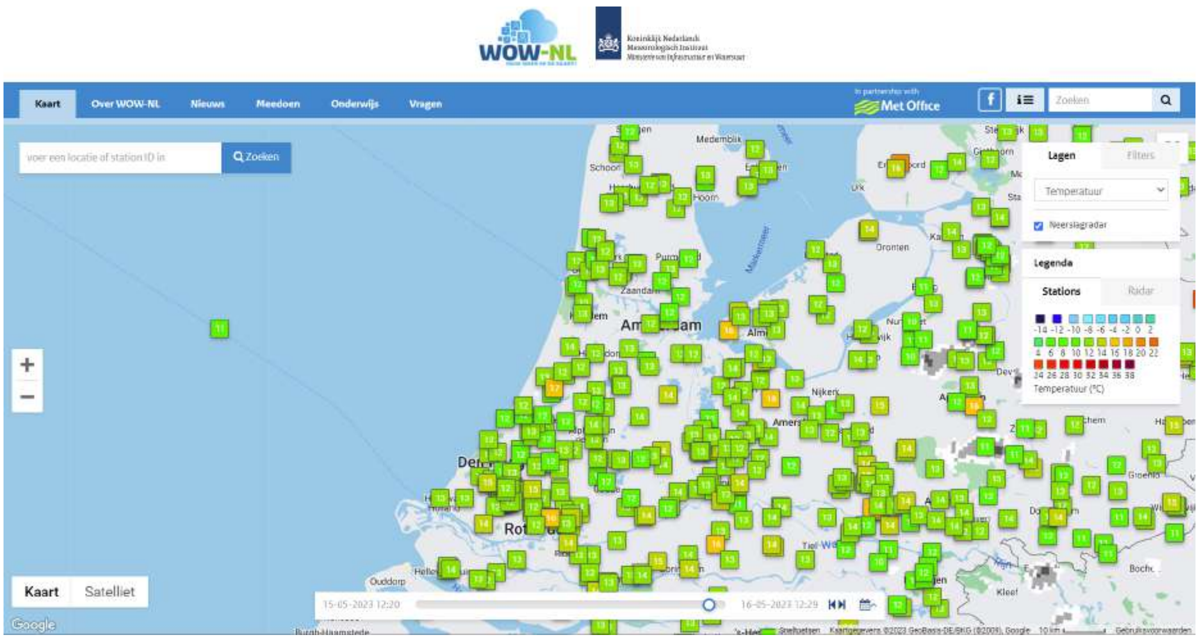


Figure 3.1: Main Page of the WOW-NL Website with a Map that shows all Online Weather Station at that time

3.0.2. WOW-NL metadata

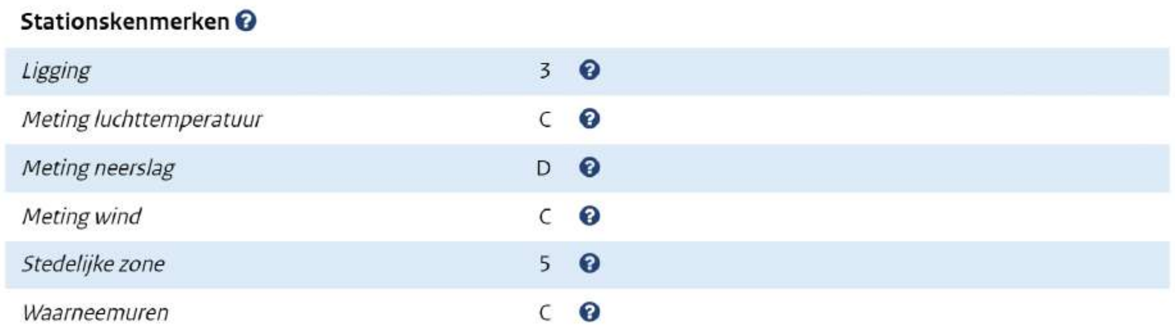
For each station in the WOW network, some additional information can be accessed by clicking the station on the website or going to the URL: "$https : //wow.knmi.nl/\#\{station - id\}$". In the station page one can look at general information about the station, its characteristics and outputs. All this additional information is referred to as the "metadata". Table 3.1 shows all the metadata fields that are available for a given station.

Field	**Definition**	**Datatype**
Station ID	Unique identification of the weather station	Integer
Position	Geographical coordinates of the weather station	Decimal (Lat, Lon)
Elevation	Height of the weather station above sea level	Integer (meters)
Time zone	Time zone of the weather station	String
Active station	Indicates whether the weather station is active	Boolean
Download?	Indicates whether downloading of data is allowed	Boolean
Website	URL of the website of the weather station	String (URL)
Station motivation	Reason why the weather station was set up	String
Official station	Indicates whether the weather station is officially recognized	Boolean
Organization	Name of the organization that manages the weather station	String
Location	Address information of the weather station	Categorical
Air temperature	Indicates whether air temperature is measured	Categorical
Precipitation	Indicates whether precipitation is measured	Categorical
Wind	Indicates whether wind speed and direction are measured	Categorical
Urban area	Indicates whether the weather station is located in an urban area	Categorical
Observation hours	Period during which the weather station makes observations	Categorical
Star rating	Rating of the weather station on a scale of 0 to 5 stars (more information in chapter 3.1)	Integer
Description	Description of the weather station	String (Free Text)
Additional	Additional information about the weather station	String (Free Text)

Table 3.1: Station Metadata Description and Datatype Extracted and Available from the WOW-NL Network

3.1. Metadata Definitions

In the metadata, most of the information is self-explanatory. There are, however, 6 entries that are based on a classification setup by the “Met Office” (A UK Metrological Organization that manages WOW). These classifications are about the location and setup of the PWS [18].



Stationskenmerken

Ligging	3
Meting luchttemperatuur	C
Meting neerslag	D
Meting wind	C
Stedelijke zone	5
Waarneemuren	C

Figure 3.2: Metadata Classification Data for A Station in the WOW-NL Network

3.1.1. Location

the classification system for station location based on various factors. The location can be classified as Very Open, Open, Standard, Limited, Sheltered, or Very Sheltered, depending on the number and proximity of obstacles and heated buildings to the temperature or precipitation instruments. 0 means there are a lot of obstacles around the station, and 5 means very open surrounding with almost no obstacles. Next to the integers, The classification system also includes categories for stations located on rooftops (R) or near traffic (T), and an “Unknown” (U) category.

Rating: The classification system ranges from 0-5, with 5 being the most open location and 0 being the most sheltered. Rooftop and traffic categories are designated by “R” and “T”, respectively, and the “Unknown” category is self-explanatory.

3.1.2. Sensor

The temperature, precipitation and windspeed instruments are categorized based on the location of the device and its calibration. Sunshine and wind speed instruments should be placed in the most open locations, which often means on rooftops or masts. The minimum distance between the instruments and obstacles is based on the height difference between the sensor and the obstacle, with a minimum distance of twice that height difference.

Temperature: the rating system for measuring air temperature. The rating system is based on the type of instrument used and the location classification of the station. Class A stations use standard instruments in a Stevenson screen that have been calibrated within the past 10 years, and require a minimum location classification of 3 (as defined in ’Location’ metadata). Class B stations use the same instruments as Class A, but can have a location classification of 2 or 3. Class C stations also use standard instruments, but can have a location classification of 1 or less. Class D stations use non-standard instruments or screens and can have a location classification of 1 or less.

Precipitation: the rating system for measuring precipitation. Class A stations use manual or calibrated tipping bucket rain gauges at a standard height of 30 cm above the ground, and require a minimum location classification of 3. Class B stations use the same instruments as Class A, but can have a location classification of 2 or 3. Class C stations use manual or calibrated tipping bucket rain gauges, but can have a location classification of 1 or less. Class D stations use non-standard instruments or screens and can have a location classification of 1 or less.

Wind: This classification categorizes the measurement of wind based on the location and characteristics of the wind meters used. It has four categories: A, B, C, and U, as well as a 0 category for locations where no wind measurements are taken. Categories A and B describe wind meters that are attached to a pole or mast, with A being more precise with a height of 10 meters and no obstacles within a 100-meter radius, and B being less precise with no obstacles within a 50-meter radius. Category C describes wind meters that are attached to a building or wall. Category U is for situations where there is no information available about the wind measuring instruments being used.

3.1.3. Urban Space

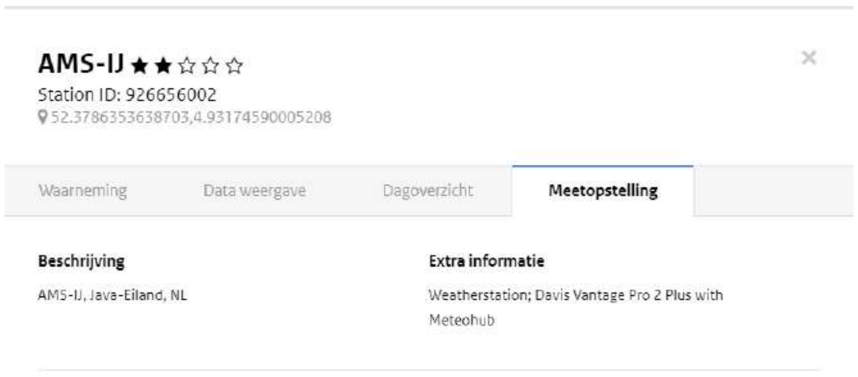
This rating system classifies areas based on their level of urban development, population density, and types of buildings. The system has seven categories, ranging from a very strong urban area with high-rise buildings to a semi-rural area with scattered houses in a natural or agricultural setting. The unknown category is for cases where there is no information available about the urban classification of an area. 1 means very urban and 7 is the most rural location. U again means Unknown and speaks for itself

3.1.4. Observation time

The observation time rating system refers to the time at which weather observations are made and reported. The system has four possible ratings ranging from A to D, plus an “Unknown” option. The first three ratings (A, B, and C) all provide a standard period of 24 hours for temperature and precipitation data. Rating A aims to provide a weather report at 9:00 UTC every day, while Rating B aims to provide a report between 6:00 and 9:00 UTC. Rating C sets the 24-hour period from midnight to midnight, which is the standard for most automated weather stations. Rating D is for situations where the closing time for temperature and precipitation data is different from A, B, or C, or where there are extremes that do not relate to a 24-hour period. The Unknown option is used when there is no information available about the reporting hours.

3.2. Current Star Rating

When a weather station is registered on WOW (the citizen science project), users can enter a number of “location attributes” that provide other users with a better understanding of the station’s measurement location. The location attributes are chosen based on methods used by organizations such as COL, WMO, and Met Office to classify station locations. Both the station’s location and the measured quantities are classified.


AMS-IJ ★★☆☆☆
Station ID: 926656002
52.3786353638703,4.93174590005208

Beschrijving
AMS-IJ, Java-Eiland, NL

Extra informatie
Weatherstation; Davis Vantage Pro 2 Plus with Meteohub

Figure 3.3: WOW -NL Station Page for a Random Station In the Netherlands with Rating Next to its Name

The current star ratings are only based on 3 of the metadata: Location, Temperature and Precipitation. This means that the station surrounding, the temperature, and precipitation sensor setup are the determining variables for the star rating. The way the star rating is give is as follows [18]:

$$5* : Location = 5 \ \cap \ Temperature = A \ \cap \ Precipitation = A$$
$$4* : Location \geq 3 \ \cap \ Temperature \in \{A, B, C\} \ \cap \ Precipitation \in \{A, B, C\}$$
$$3* : Location \geq 3$$
$$2* : Location \geq 1$$
$$1* : Location \in \{0, 1, R, U\}$$

Note how from an 3-star rating and under only the location metadata has influence on the station's rating. Next to that we can see how if metadata is Unknown (U) the rating is a 1 star by default, which is a harsh penalty for not delivering data.

3.3. Webbots

A web robot, also known as an internet bot or simply bot, is a type of software application designed to perform automated tasks on the internet [8]. Bots are programmed to imitate human activity on a large scale, such as messaging, and can perform simple and repetitive tasks faster than a person [11]. Bots typically act as clients in a client-server model, with web servers serving as the server. The primary use of bots is for web crawling, which involves automated scripts fetching, analyzing, and organizing information from web servers [24]. Bots are responsible for generating more than half of all web traffic [4].

Web servers may employ various measures to restrict bot behavior, such as providing a robots.txt file containing rules for bot activity [7]. Bots that fail to comply with these rules may be denied access or removed from the website in question. However, adherence to these rules is voluntary, and there is no way to ensure that bot creators or implementers read or acknowledge the rules. While some bots, such as search engine spiders, are beneficial, others may be used for malicious purposes, such as launching attacks on political campaigns [6].

3.4. Landelijk Grondgebruik Nederland (LGN)

The LGN2021 database, or Landelijk Grondgebruiksbestand Nederland 2021, is a geospatial dataset that provides information about land use and land cover in the Netherlands. It is a comprehensive and detailed representation of the country's land surface, categorizing different land cover types across the entire territory [16].

The database classifies land into various categories, such as built-up areas, agricultural land, forests, nature reserves, water bodies, and other land cover types. Each category represents a specific type of land use or land cover, allowing researchers, planners, and policymakers to understand the distribution and characteristics of different land features in the Netherlands.

The determination of the LGN (Landelijk Grondgebruiksbestand Nederland) database involves a comprehensive process of data collection, classification, and mapping to provide an accurate representation of land use and land cover in the Netherlands. Here is an overview of how the LGN database is determined:

1. Data sources: Multiple data sources are used to gather information about land use and land cover. These sources include aerial imagery, satellite imagery, topographic maps, administrative records, and other geospatial datasets. The data sources are selected based on their quality, resolution, and relevance to land cover classification.
2. Image acquisition and preprocessing: Aerial imagery or satellite imagery is acquired, covering the entire extent of the Netherlands. These images capture the land surface at various wavelengths, allowing the identification and differentiation of different land cover types. The acquired imagery undergoes preprocessing to correct for geometric distortions, atmospheric effects, and radiometric inconsistencies.

3. Image interpretation and classification: Trained image interpreters analyze the imagery and visually identify different land cover features, such as urban areas, agricultural fields, forests, water bodies, and natural vegetation. They use their expertise to classify each pixel or region in the imagery into predefined land cover classes based on visual cues, spectral characteristics, and contextual information.
4. Ground truth data and validation: To ensure the accuracy of the classification, ground truth data is collected. This involves field surveys, ground-based measurements, and validation points where land cover types are physically verified. The ground truth data serve as reference points to assess the classification accuracy and make necessary adjustments or corrections to the classification scheme.
5. Data integration and mapping: The classified land cover information is integrated into a geospatial database, where each land cover class is assigned a unique code or identifier. This database combines the classified imagery with other geospatial data layers, such as administrative boundaries or topographic features, to provide a comprehensive representation of land use and land cover across the country.
6. Database updates and maintenance: The LGN database is regularly updated to account for changes in land use and land cover over time. New imagery and data sources are incorporated, and classification algorithms are refined to ensure the database remains up-to-date and reflects the current state of land cover in the Netherlands.

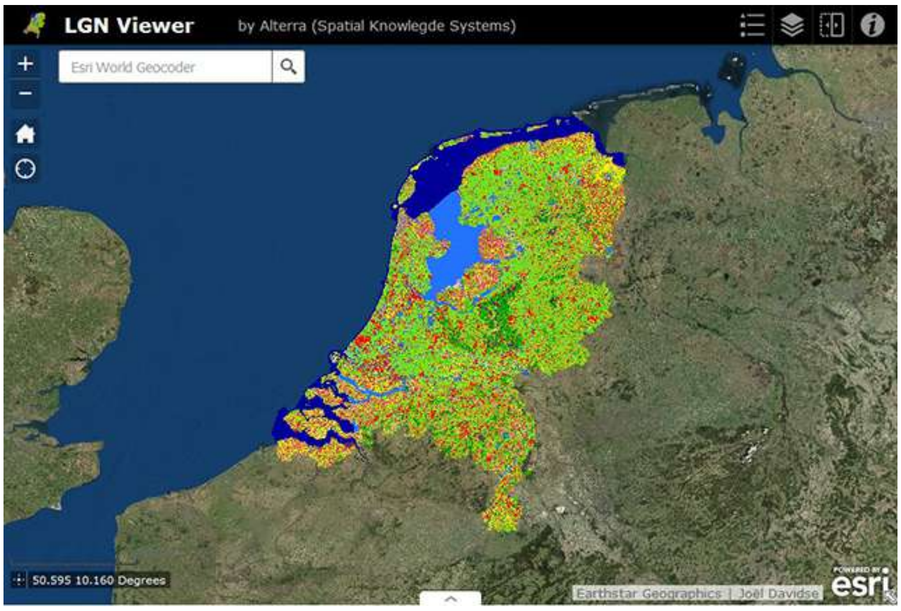

LGN Viewer
by Alterra (Spatial Knowlegde Systems)
Esri World Geocoder
50.595 10.160 Degrees
Earthstar Geographics | Joël Davidse
esri

Figure 3.4: LGN2021 Viewer with the Land types for a resolution of 25 m for The Netherlands

4 Methods

This thesis will be a step forward to be able to assess value of a PWS in the context of a citizen science network. A high value station is a station with a lot of insight or is relevant enough to include in a use case based on other criteria. In this thesis, we propose a framework to assess the value of a PWS.

To be able to get to this objective assessment of a PWS, we have a two-step approach. We will first analyze the network of weather stations on Weather Observation Website (WOW) for the Netherlands to get a better understanding of the current status of the PWS networks. This will be the first time a PWS network will be analyzed on a national scale. A lot of questions will be answered that give insight on how these networks are being used now. What data is available in the network? How do we bypass the availability issue of the network? What are structural problems in the dataset? With these insights, we are able to structurally show the problems and be able to suggest improvements.

In the second part, we have a practical framework which can be used in combination with data that is really available in the network. To demonstrate, we actually will assess all stations in the Netherlanders on Value conform the framework in the capacity we are able to do this with the resources we have to date. The result will be a new and improved station network in which the stations are assessed on different criteria for value/usefulness.

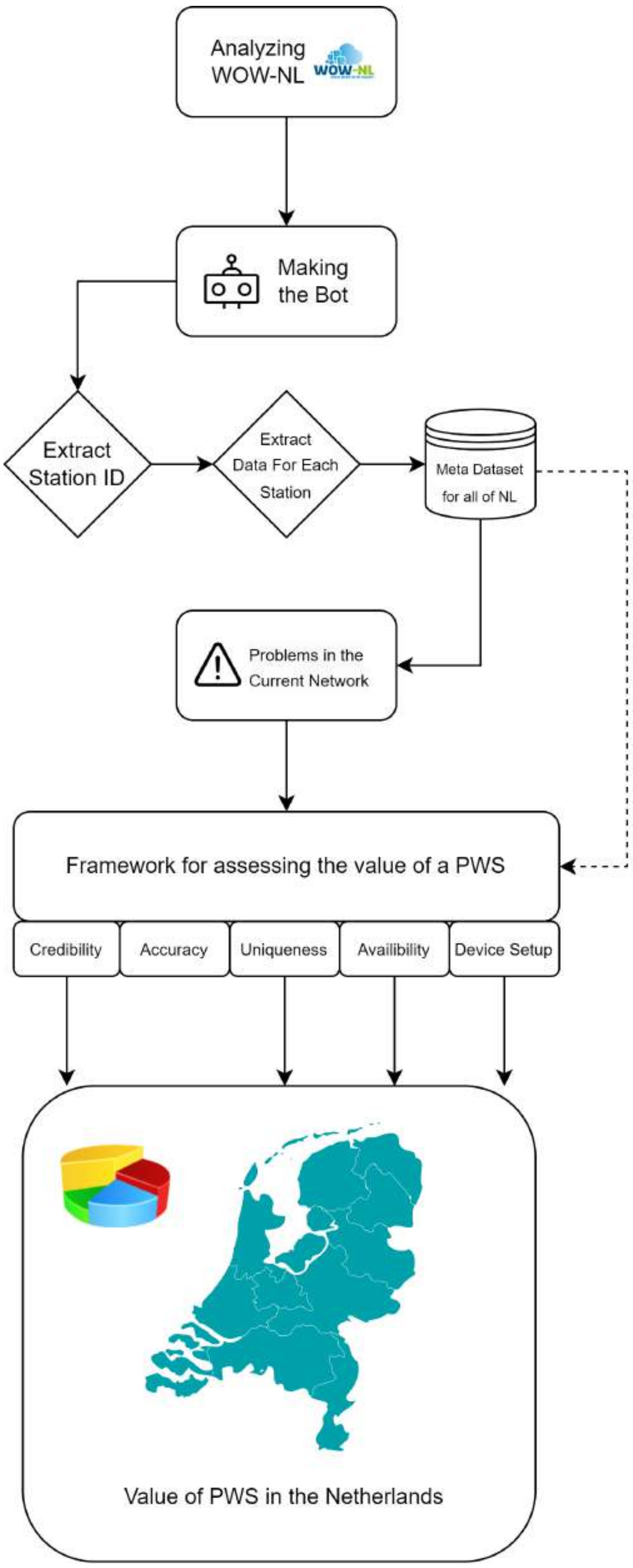

Analyzing WOW-NL
WOW-NL
Making the Bot
Extract Station ID
Extract Data For Each Station
Meta Dataset for all of NL
Problems in the Current Network
Framework for assessing the value of a PWS
Credibility
Accuracy
Uniqueness
Availibility
Device Setup
Value of PWS in the Netherlands

Figure 4.1: Flowchart of the Method of this Research

4.1. Values of a PWS

The need for a more objective framework that can tell us something about the value of a station is needed. Through looking at the metadata, and reading relevant papers on personal weather stations. We have set up a framework with 5 values that are important for a PWS. These 5 values are more to be seen as criteria on which we can quantify the value of a certain personal weather station. In figure 4.2, the values are visualized with its definition in the schematic in figure 4.2. In table 4.1, the definition of each value can be read.

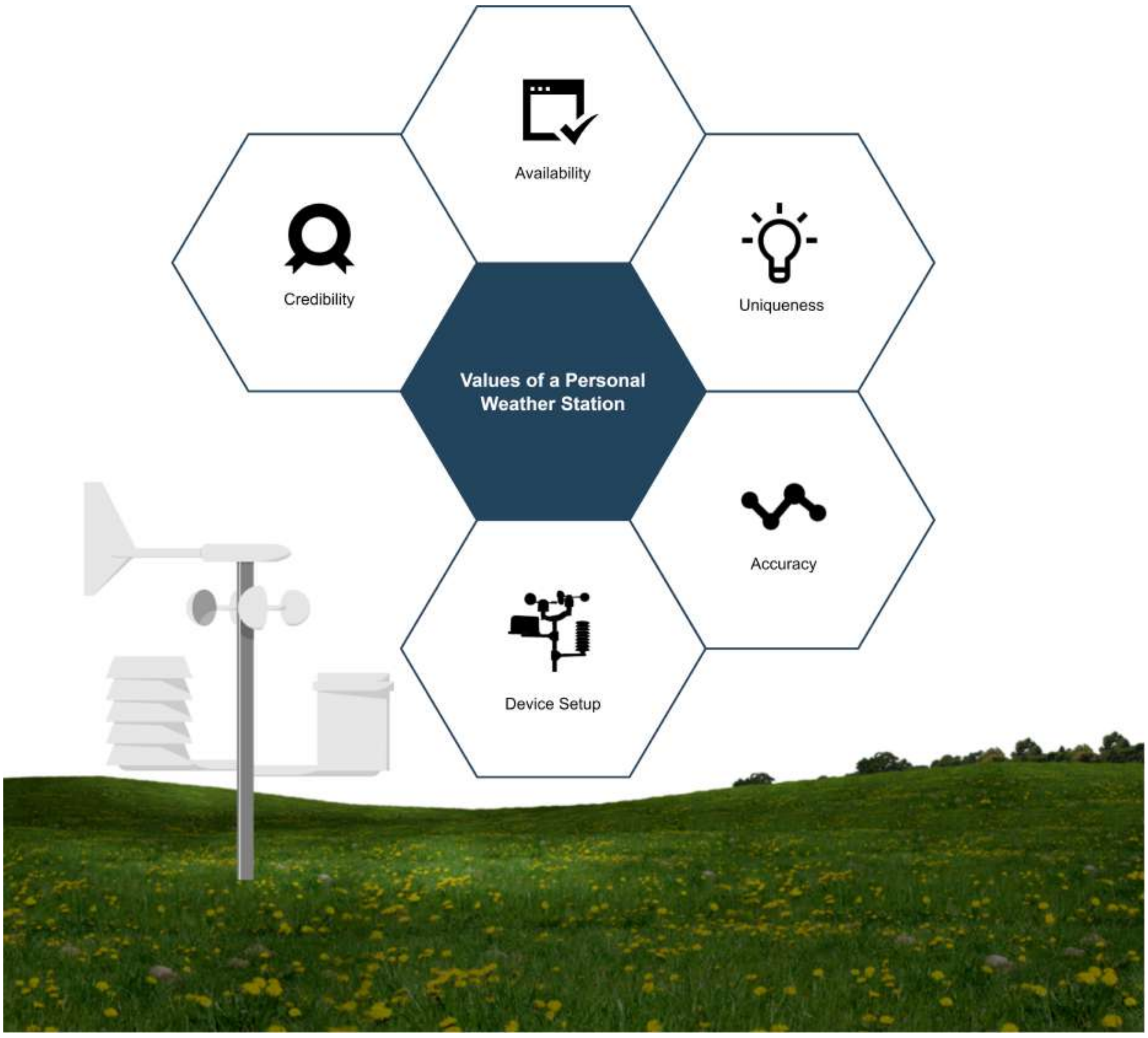


Figure 4.2: Framework to Assess the "Value" of a Personal Weather Station in a Citizen Network

While accuracy remains a fundamental aspect of a PWS value, this thesis aims to shift the focus towards the broader value that these weather monitoring systems can offer and be assessed on. By exploring the broader information about a PWS, this study highlights and tries to understand PWS from a higher level, hopefully giving new insights into the field. Other research done has focussed on Accuracy of PWS, for example work done by PhD Simon Bell from the University of Birmingham [3]. Therefore, the accuracy criteria will not be looked at in this thesis.

Value	Definition
Device Setup:	What kind of sensors are used, when were they installed, and how are they combined? A station with old/cheap or little variety in sensor type is less valuable. That the device setup effects the value of a station is well known and discussed in depth by the world metrological organization (WMO) in standards they write [15]
Credibility:	Credible owners tend to maintain the devices better, therefore increasing the value of a station.
Availability:	A station that steadily reports data is more valuable because it is fulfilling its main purpose of delivering. A station with a very good device setup where the data is not extractable is almost worthless. The availability of a station has been talked about as an important part of a weather station in the PhD research by Simon Bell [3]
Uniqueness:	A weather station that gives high insights because the measurement is exclusive or that the station is located in an interesting location. A spatially exclusive station is one that is unique in a sense that it is one of the few observations in an area. Location Exclusivity is a station that measures something that is unique, for example, a certain ground type.
Accuracy:	Error/Difference of the measurement time series from a Personal Weather Station through comparison with professional and surrounding PWS. Countless research have looked at accuracy as a way to analyze the value of a station. In most cases, it is the first thing one thinks about.

Table 4.1: Definition of the Values of a PWS in a Citizen Network

4.1.1. Use Case Example

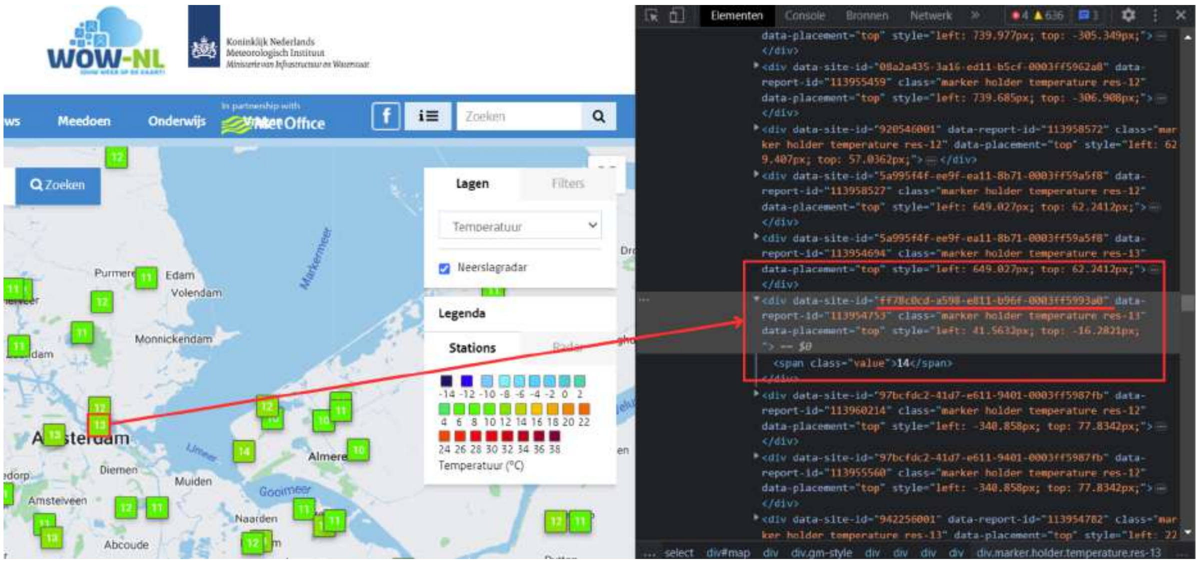
Notice how the framework would be useful in situations we want to make a choice between two station. The framework gives you the criteria to approach the analysis of making a choice.For example, if the two stations have the same owner the credibility is the same and therefore one of the 5 axes of value is the same. It's then up to the other 4 values to be assessed. Let's say the device is the same, it's equally unique because they are spatially next to each other, but the availability is worse of one. We now know the station with better availability is the more "valuable" station of the two.

4.2. The WOWBot

Based on the way the website has been programmed, we can develop a script that is able to extract the information we need. Python Selenium is a web automation and testing tool that enables programmatic control of web browsers [9]. It can be utilized to extract metadata from citizen science networks like the WOW-NL network, specifically for retrieving information on all stations in the Netherlands. Python Selenium is often the preferred choice for batch export in this scenario due to the absence of a direct API or data export feature. By automating web interactions, Selenium allows for automated navigation, search, and extraction of station metadata. However, it is important to note that Selenium-based scraping may require periodic updates to accommodate changes in the web interface. It is advisable to explore alternative options, such as APIs, before resorting to Selenium. Nevertheless, when no alternative is available (like in this case), Python Selenium proves to be a reliable tool for efficiently extracting metadata from the WOW-NL network. There are 2 steps to be taken to be able to extract all the stations in the WOW-NL network.

1. Extract all the station IDs of the network.
2. Extract data of each individual station though the station IDs gathered in (1).

For (1), we created a bot that is able to go to `https://wow.knmi.nl/`, remove all filters on the map to show all stations and zoom to the Netherlands in such a way that all stations are shown in the UI of the website. This way all stations correspond to an element on the website which can be inspected and extract from the HTML. These elements can be seen on the map in figure 4.3 as rectangles and carry with them the station-id in the description, which is the string we extract with the bot.



```
data-placement="top" style="left: 739.977px; top: -305.349px;">…
</div>
▸<div data-site-id="08a2a435-3a16-ed11-b5cf-0003ff5962a8" data-
report-id="113955459" class="marker holder temperature res-12"
data-placement="top" style="left: 739.685px; top: -306.908px;">…
</div>
▸<div data-site-id="920546001" data-report-id="113958572" class="mar
ker holder temperature res-12" data-placement="top" style="left: 62
9.407px; top: 57.0362px;">…</div>
▸<div data-site-id="5a995f4f-ee9f-ea11-8b71-0003ff59a5f8" data-
report-id="113958527" class="marker holder temperature res-12"
data-placement="top" style="left: 649.027px; top: 62.2412px;">…
</div>
▸<div data-site-id="5a995f4f-ee9f-ea11-8b71-0003ff59a5f8" data-
report-id="113954694" class="marker holder temperature res-13"
data-placement="top" style="left: 649.027px; top: 62.2412px;">…
</div>
▾<div data-site-id="ff78c0cd-a598-e811-b96f-0003ff5993a0" data-
report-id="113954753" class="marker holder temperature res-13"
data-placement="top" style="left: 41.5632px; top: -16.2821px;
"> == $0
   <span class="value">14</span>
</div>
▸<div data-site-id="97bcfdc2-41d7-e611-9401-0003ff5987fb" data-
report-id="113960214" class="marker holder temperature res-12"
data-placement="top" style="left: -340.858px; top: 77.8342px;">…
</div>
▸<div data-site-id="97bcfdc2-41d7-e611-9401-0003ff5987fb" data-
report-id="113955560" class="marker holder temperature res-12"
data-placement="top" style="left: -340.858px; top: 77.8342px;">…
</div>
▸<div data-site-id="942256001" data-report-id="113954782" class="mar
ker holder temperature res-13" data-placement="top" style="left: 22
```

Figure 4.3: WOW-NL Page Code Corresponding to Object on the Map

The result of IDs of all weather stations in the Netherlands. This list will be input for the next task the bot can execute. The bot will one for one go to the station's website: `https://wow.knmi.nl/#{station_ID}` This website will show the real time information about the station. The bot does the following to get the information:

1. Check if the station website exists. If it does not, the station is offline and is not anymore in the WOW database.
2. Extract the location coordinates of the station and check whether the station is inside the polygon of the Netherlands. If it is not, the station is skipped during the next step, meaning that no metadata are extracted from this station.
3. Go to the Meta-Data subpage and extract all information available on the station, including the description left by the maintainer.
4. Aggregate all the extracted data into the database of all stations in the Netherlands (formatting).

Algorithm 1 Extract PWS-id Bot

```
 1: Start Chrome WebDriver
 2: go to -> https://wow.knmi.nl/
 3: time.sleep(loading time)          ▷ Loading time is time until HTML stops generating
 4: Turn off all Filters on the viewer to reveal all stations
 5: time.sleep(loading time)
 6: Extract the HTML
 7:
 8: Active Station div-class= ”marker holder temperature res-i” with i=1,...,28
 9: Offline Station div-class= ”marker holder temperature no-data wow”
10:
11: if div-class == marker holder temperature res-i with i=1,...,28 then
12:     Save div-id to Online List
13: end if
14: if div-class == marker holder temperature no-data wow then
15:     Save div-id to Offline List
16: end if
```

Algorithm 2 Extract PWS Metadata, Appendix A

```
 1: Inputs:
        Output of algorithm: ”Extract PWS-id Bot”
 2: Initialize:
        Chrome Webdriver Already Initialized
 3: for all station-ids do
 4:     go to -> https://wow.knmi.nl/#{station-id}
 5:     if URL Exist then
 6:         time.sleep(loading time)      ▷ Loading time is time until HTML stops generating
 7:         Extract Station Coordinates (Latitude, Longitude)
 8:         if Station Coordinates inside the Netherlands then
 9:             click on -> subpage ”Meetopstelling” (Measurement setup)
10:             Extract the Metadata Available (As defined in table 3.1)
11:         end if
12:     end if
13: end for
```

With this an automated bot has been made that does 2 things, gets the names of all stations in the Netherlands and extract the data for that station, if it exists. The program has been coded and is available in the appendix of this thesis. The code has been developed in python and uses selenium, a python package. Note how information about the versions used is essential for recreating the bot and using it for own purposes, check the ”requirements.txt” file in appendix A.

4.3. Analysis of the Network

4.3.1. Nearest Neighbor

Nearest neighbor analysis is a method used to analyze spatial patterns in a dataset. The aim is to determine whether the points in the dataset are randomly distributed or whether there is a systematic pattern to their arrangement. This method is commonly used in fields such as ecology, geography, and urban planning.

The basic principle of nearest neighbor analysis is to calculate the distance between each point in the dataset and its nearest neighbor [13]. The pseudocode and algorithm to calculate the NN can be read in Algorithm 3.

Algorithm 3 Nearest Neighbor Analysis

```
1: Inputs:
       n weather stations $\mathbf{u}_1, ..., \mathbf{u}_n$ with $\mathbf{u}_i = (u_{ix}, u_{iy}) \in \mathbb{R}^2$
       m comparison weather stations $\mathbf{v}_1, ..., \mathbf{v}_m$ with $\mathbf{v}_i = (v_{ix}, v_{iy}) \in \mathbb{R}^2$
       distance function $d : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^+$
2: Initialize:
       $d_{u_i,v_i} = \sqrt{(v_{ix} - u_{ix})^2 + (v_{iy} - u_{iy})^2}$
       $\mathbf{u}_{nn(j)}$ the nearest neighbor of $\mathbf{v}_j$
3: for $\mathbf{u}_i$ in $\mathbf{u}_1, ..., \mathbf{u}_n$ do
4:     the input dataset $d(\mathbf{u}_{uu(j)}, \mathbf{v}_j) = \min_{1 \le i \le n} d(\mathbf{u}_i, \mathbf{v}_j)$
5: end for
6: Save $d(\mathbf{u}_{uu(j)}, \mathbf{v}_j)$ for all j
```

Note how this algorithm can also be used to calculate the NN-distance for a network. Not only to compare two networks. In our case, the algorithm will be used to calculate the nearest neighbor distance of all amateur stations against the amateur stations of the same network. In the second case, we use the algorithm to calculate the distance to the nearest professional station. The results is a distance metric for each station in the network telling us how far away another amateur station is for potential cross validation or comparison or how far a professional station is.

The distances can be compared to a theoretical distribution of distances that would be expected if the points were randomly distributed. This is called the NNI (nearest neighbor index). The formula for nearest neighbor analysis is:

$$Nearest\ Neighbor\ Index(NNI) = \frac{D}{D'} \tag{4.1}$$

$$D = Observed\ Mean\ Distance \tag{4.2}$$

$$D' = Expected\ Mean\ Distance \tag{4.3}$$

If the observed mean distance is less than the expected mean distance, the NNI will be less than 1, indicating that the points are clustered together more than would be expected by chance. If the observed mean distance is greater than the expected mean distance, the NNI will be greater than 1, indicating that the points are more evenly spaced than would be expected by chance. If the observed mean distance is equal to the expected mean distance, the NNI will be 1, indicating a random pattern.

Nearest neighbor analysis can be used to detect clustering or spatial patterns in a variety of datasets, such as tree locations in a forest, crime incidents in a city, or animal territories in a habitat. It is important to note that nearest neighbor analysis assumes a two-dimensional, Euclidean space and that other methods may be more appropriate for datasets with different spatial properties.

4.3.2. Language Processing

A word cloud is a visual representation of a collection of words, where the size of each word indicates its frequency or importance. Making a word cloud involves several steps [23]:

1. Choose a dataset: The first step is to choose the text that will be used to generate the word cloud. This could be a single document, a collection of documents, or even a website or social media feed.
2. Preprocess the text: Once the dataset has been selected, it needs to be preprocessed to remove stop words (e.g., ”and,” ”the,” ”a,” etc.), punctuation, and other irrelevant information. This can be done using text preprocessing libraries in Python such as NLTK or SpaCy.
3. Count the frequency of each word: After preprocessing, the frequency of each word in the dataset is counted using a word frequency counter. Python libraries such as Counter can be used for this purpose.
4. Choose a visualization tool: Several tools are available to create word clouds. Some popular options include WordCloud in Python, Tagxedo, and Wordle.
5. Generate the word cloud: Once the visualization tool has been selected, the word cloud can be generated. The tool will usually have several options for customizing the appearance of the word cloud, such as font size, color scheme, and layout. The most common approach is to use the frequency of each word to determine its size in the word cloud.
6. Refine the word cloud: After generating the word cloud, it is a good idea to refine it by adjusting the font size or color scheme, or by removing certain words that are not relevant to the analysis.

In summary, making a word cloud involves preprocessing the text, counting the frequency of each word, choosing a visualization tool, generating the word cloud, and refining the result. Word clouds can be a useful tool for visualizing and summarizing large datasets of text, and can help to identify key themes and topics.

5 Results: Analysis of the Current state of the Dutch PWS Network

5.1. Challenges in extracting information from the WOW-NL network using a bot

When making the Web bot, we learned a lot about the WOW-NL network and how a bot can assist in extracting information about personal weather stations. For example, it is important to understand the dynamic behaviors of the website. A single input with the bot can change the website in such a way that the HTML changes and the bot might not be able to perform its task. For example, there are some stations for which the station ID can be read, but the station website does not exist. During the development phase, a lot of errors and exceptions had to be considered. Over time, the bot become better at detecting edge cases and handling them with the use of try/except statements. When a try returns an error in the except statement, the station ID gets saved in a separate file for further investigation by humans. The list below gives an overview of the particularities found in the structure of the WOW-NL network:

1. Stations for which the ID is known but which do not exist when trying to access their data.
2. Metadata entries sometimes can have values outside the range of acceptable parameters defined in the WOW-NL documentation. For example, in some stations, the value 'U' which stands for 'Unknown', is entered as 'null', ' ' or even '0'. In this case, the bot simply overwrites the value and saves it as 'U'.
3. The website can suddenly become unavailable due to server maintenance
4. The default map does not show any stations, only aggregations of multiple stations as a rectangle. This way a user can click on for example on the rectangle around Amsterdam to then see the stations in the area of Amsterdam on the map. This way, the website minimizes loading for the user and server. For us, this means we have to zoom in twice to get to a resolution where we are able to extract individual stations.

In total, the bot found 981 personal weather stations in the Netherlands (1031 if we also include the professional stations). The time needed to extract all information using the bot was around 6 hours. The bot is coded in such a way the extraction can be done on the background from a console on a local or virtual machine.

5.2. Metadata analysis

After running the code, we have a dataset of all stations in the Netherlands and all their metadata entries. This is very interesting information because for the first time, we are able to analyze the spatial distribution and frequency of station characteristics in the network. For example, we can show how many three, four and five star rated stations there are, where these stations are located, and who operates them.

The stations that were found are visualized in figure 5.1. Here the spatial structure of the network can be seen for the first time. The green points are stations that are active, the red ones are not active based on the metadata of the network. As we can see, almost all stations are active (99.3% like in 5.5f). Figure 5.5 shows pie charts for all the metadata entries that we were able to extract from the WOW-NL network, including the activity shown in Figure 5.1.

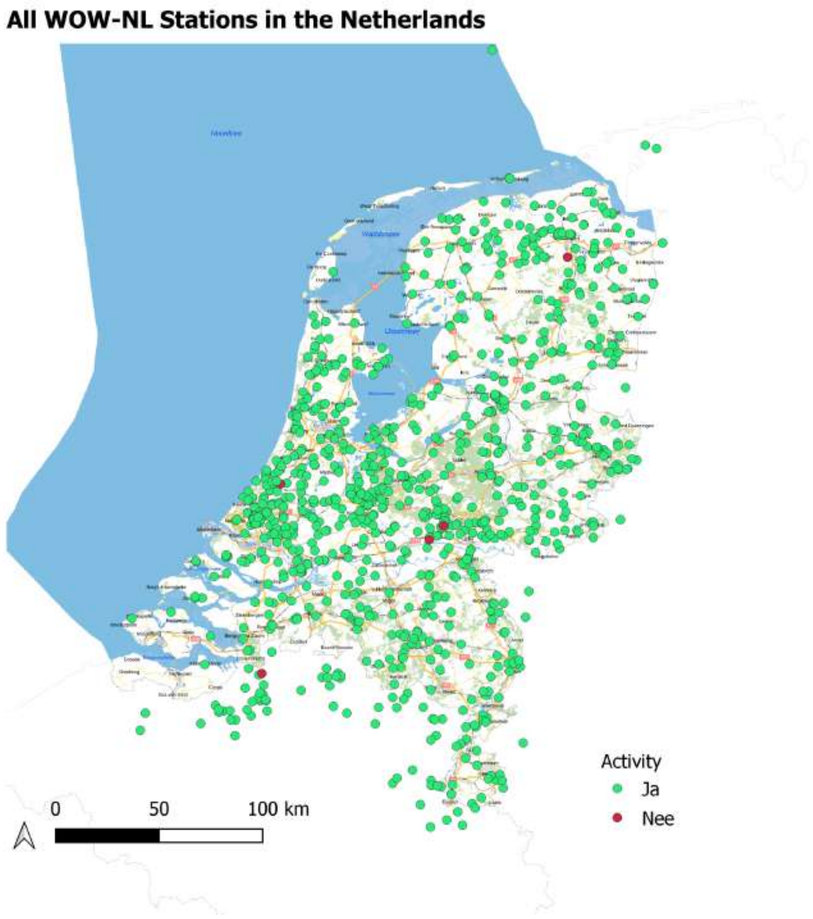


Figure 5.1: Map of all, 1031 stations in the WOW-NL network extracted using the web bot. Only stations located in the Netherlands are shown

Figure 5.2 until 5.4, is the information about what weather variables the station is measuring. The second row is the information about the motivation and location of the station. Lastly, the charts in the last row are some yes/no questions that the participants of the network had to fill in about the station. These results tell us a lot about the status of the network and what can be improved.

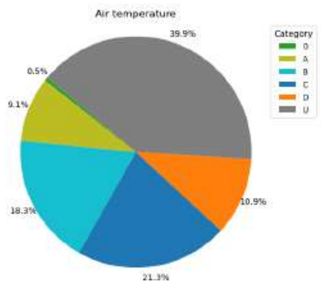


Figure 5.2: Temperature Setup Categories

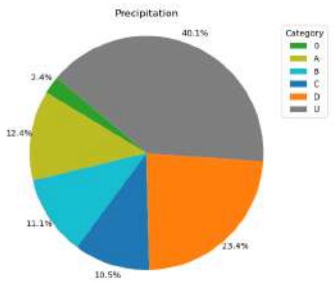


Figure 5.3: Precipitation Setup Categories

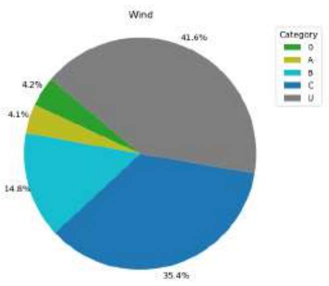
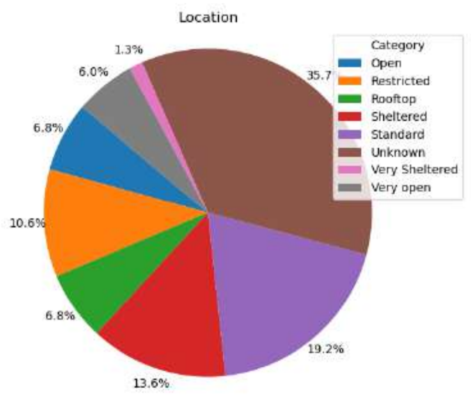
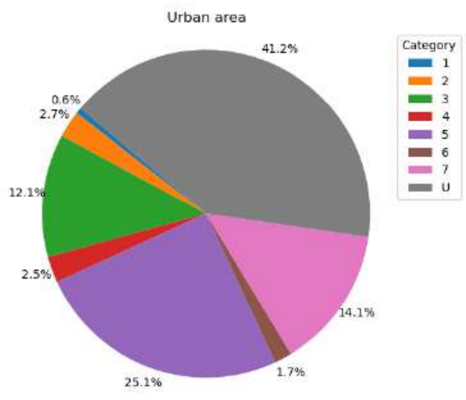


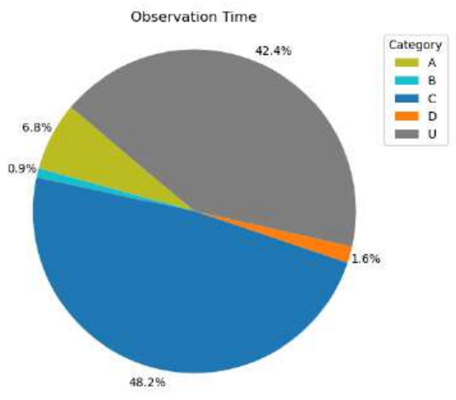
Figure 5.4: Wind Setup Categories



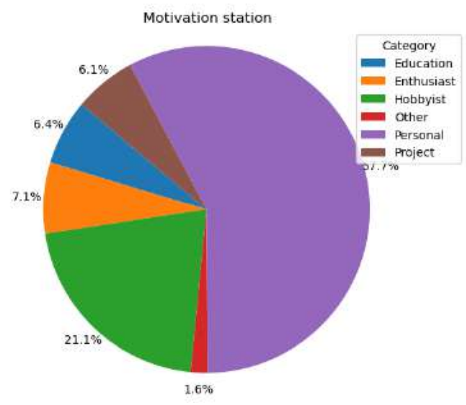
(a)



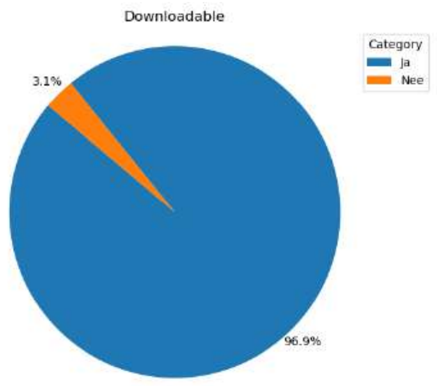
(b)



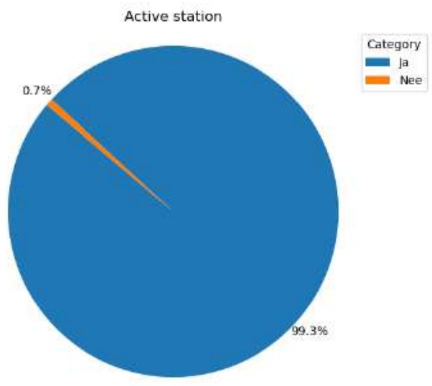
(c)



(d)



(e)



(f)

Figure 5.5: Surrounding and Reporting Metadata in a Pie Chart for all stations in the Netherlands

Here are some of the insights that can be seen from the pie charts above.

1. For almost 40% of all stations, the kind of measurement device(s) is unknown ('U', gray)
2. Almost 59% of the stations are in rural areas as we can see in the pie chart "Urban Area" (5>).
3. Most of the stations (57.7 %) in the network are for personal use.
4. The motivation is described in vague terms such as "enthusiast", "hobbyist" and "personal", which are overlapping terms.
5. 99.3% are active, and 96.9% of their data are downloadable.
6. Almost all stations are managed by citizens, only 0.2% are official stations
7. With the current star rating

5.2.1. KNMI rating

WOW (Weather Observations Website) station is automatically classified based on the information provided during registration. The classification results in a score between 1 and 5 stars. This score is based on the location of the station and the quality of temperature and precipitation device setup. The classification process is explained more in depth in section 3.2. This is an questionable approach where the amount of information used to form a rating is only 3 of the 6 main metadata entries. When we look at the rating system spread over the Netherlands, we can see the following map:

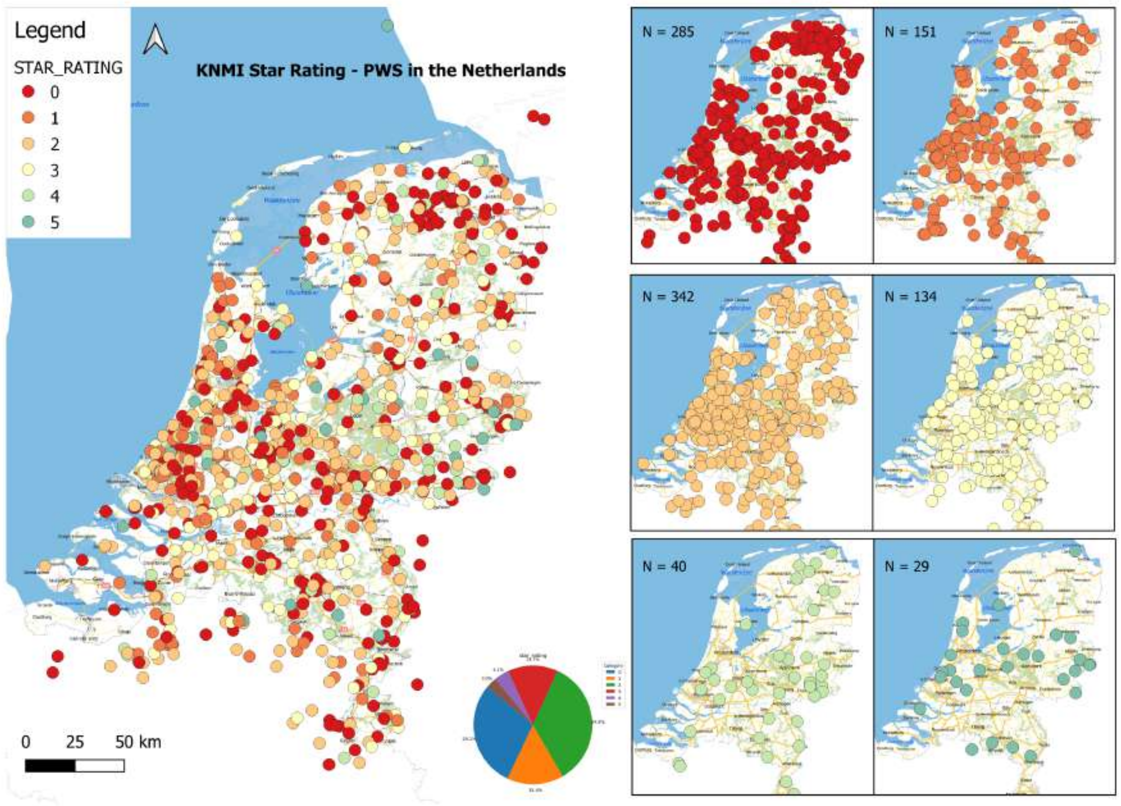


Figure 5.6: Spatial distribution and frequency of PWS star rating in the WOW-NL network, extracted by the web bot

We can see that 80% of the stations are rated a 2 or lower. Almost all of them receive a low rating because of the large numbers of unknown metadata values. For example, 393 stations have unknown “precipitation” classification (see Table 3.1). For 391 stations, the “air temperature” classification ((see Table 3.1)) is unknown. The “Location” classification (see Table 3.1) is not known for 350 stations.

These results tell us about a bigger systematic problem. The rating is based on 3 variables that 50% of the time are unknown. Moreover, the variables themselves are often vaguely defined, overlapping and rather subjectively assessed by the users themselves. The classification is a human input which is prone to misjudgment of the user. For example, the “motivation” entry has overlapping options like, hobby, enthusiast, or amateur. What is the difference between this? An amateur can still be an enthusiast and doing it for a hobby.

5.2.2. Language Processing

One of the other interesting data that a citizen provides the network is additional notes on the station in free text. In this free text, some interesting things can be highlighted by the citizen, like particularities about the area or the sensor that has been used. In the WOW-NL network there are 2 inputs of free text, 'additional information' and 'extra info'. These text boxes again show how there is no systematic registration of device information. The two text boxes seem to have the same function, and the exclusion of one would not negatively affect the metadata of a station. After going deeper in the registration process, an "instruction manual" for the registration of the station and filling in the metadata has been found [19]. After reading through, we found that 'extra info' text box is for explaining the "device site" in more depth (optional), while the 'additional information' is again an extra text box at the end of the form to give "information about the site or the equipment" as stated in the manual. This shows the inconsistent registration process. Therefore, we combined both datasets for our analysis. After aggregating the strings and removing stop words in Dutch and English, we were able to plot and create a word cloud for what citizens have to say about their PWS (figure 5.7).

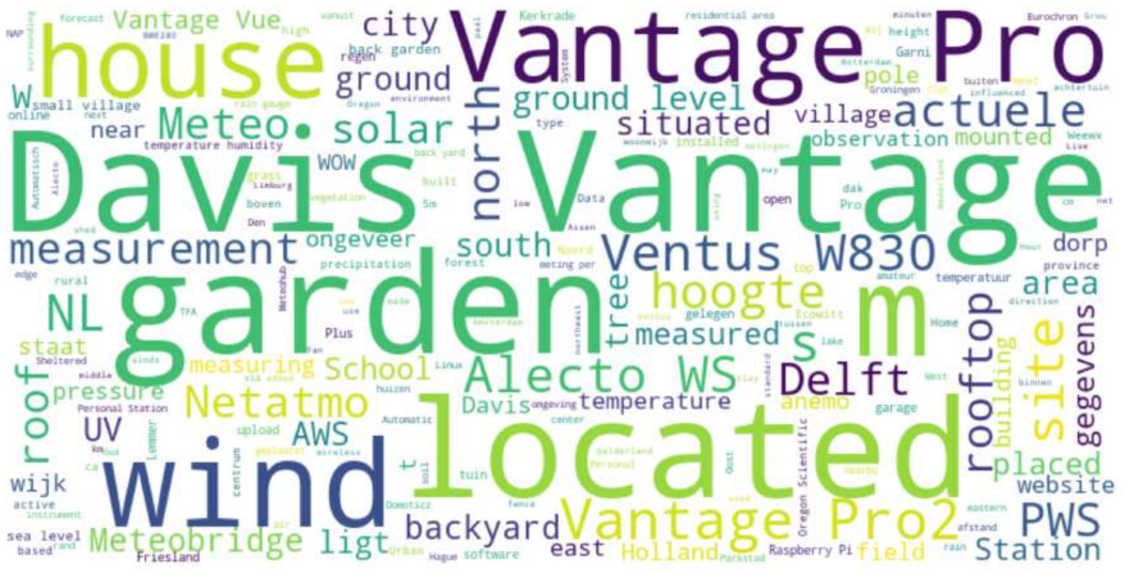


Figure 5.7: Word Cloud from the Free Text Box of All Dutch Weather Stations inside the WOW-NL Platform

In figure 5.7 we see the word cloud, the size of the words correspond to its frequency of occurrence of the terms. As can be seen in the figure above, the main signals are the device names. For example, "Davis Vantage", "Vantage Pro (2)", "Ventus W830", "Alecto WS" and 'Vantage Vue' are the main ones. Next to that, the location of device has been further specified a lot. The second-biggest frequency was "Garden", "House" and "Located". Interestingly, 'Delft' is a signal that has been caught on to as one of the few location names that has been mentioned in the network.

The word cloud in figure 5.7 not only tells us the information that the network users put in the text box. But also what attributes are missing from the network administration. The device name should be specified as an entry to the network. When linking this with the problems in the device categorization in table 3.1, we see that steps can be made in standardization of the documentation of device setup in a PWS.

6 Results: Assessing the Value of the Dutch Personal Weather Stations

in section 4.1 the framework for assessing a personal weather stations has been explained. Here we take the criteria and project it on the network we have extracted. The following is the value assessment of the Dutch PWS network for all criteria except 'Accuracy'.

6.1. Uniqueness

Uniqueness has been defined in two ways. Firstly, the station can be *"Spatially Exclusive"*, meaning the station is standalone in an area making interesting because no alternative are present. The second definition is *"location based interest"* where a station is measuring a unique location. An example would be a PWS on a Dune. Someone who is investigating sand transportation of the dunes might adhere a lot of value on wind measurements in this area. In this case, the user would use the framework to get the PWS in dunes for the Uniqueness criteria with a wind sensor in the device setup. With the national dataset of PWS's we finally are able to analyze the spatial structure of the network. We determined the *"Spatially Exclusivity"* of a station with two metrics:

1. Distance to Professional Stations
2. Distance to Amateur Stations

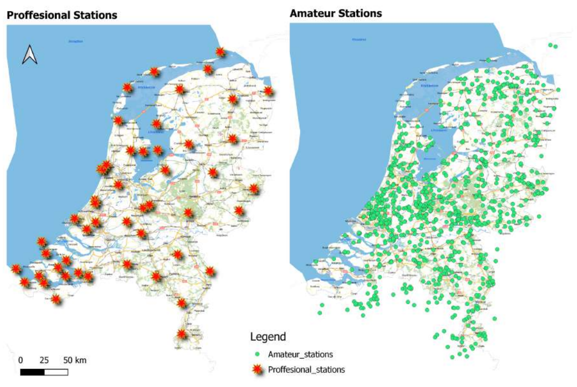


Figure 6.1: Weather Station in the Netherlands, appendix C for bigger image

"Uniqueness" in the framework can be assessed. As the definition states: A weather station that gives high insights because the measurement is exclusive or that the station is located in an interesting location. A spatially exclusive station is one that is unique in a sense that it is one of the few observations in an area. With the nearest neighbor analysis, we are able to calculate which stations are far from other stations, in other words "spatial exclusive" and thus Unique in our definition.

On the flip side, a station can be very close to a professional/amateur weather station, making it redundant because the other stations might have more insight because it might score better on one of the other criteria in the framework. In the case of a professional station, the accuracy is much higher because of more expensive device setup. While it is being maintained actively by the KNMI making it a more “credible” station. Lastly, availability does not differ, hence the use of the professional station has more “Value” as the framework proposes. In figure 6.1 the professional stations are spatially visualized. This is an example of where the framework makes sense of the assessment of PWS stations in a network.

In the same way, the framework is efficient in determining the value in a professional vs. amateur station comparison. The same holds for assessing 2 amateur stations. In this case, when we look at the stations that are far away from an amateur neighbor, we see this station as a spatially unique observation, which then again translates into having more value assuming the other variables are all the same for the stations that are being compared.

In figure 6.2 we calculated the Nearest Neighbors Distances and visualized it with a color ramp. Red means the station is close by, while Blue and Green Means it is far from its neighbor.

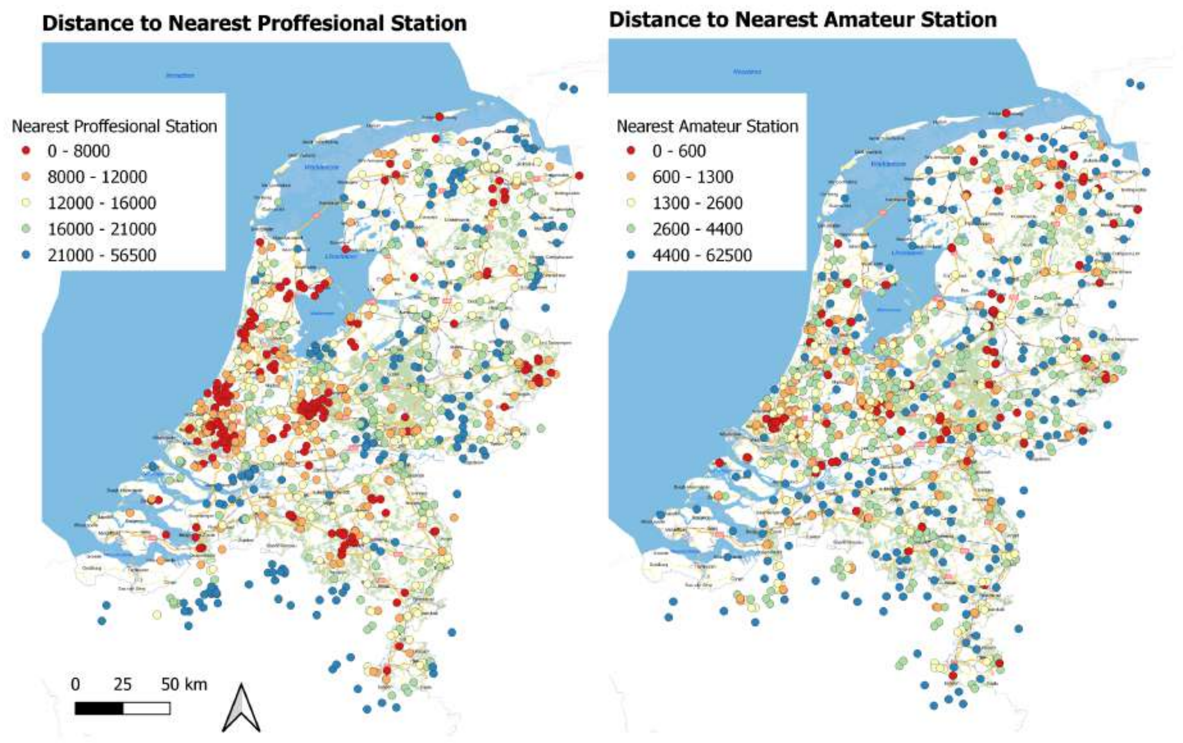


Figure 6.2: Nearest Neighbor Analysis For Amateur to Professional Distance and Amateur to Amateur Distance

When plotting all the distances of the PWS in the Netherlands in a histogram, we can see how the distribution of the distances differ for the two variants we calculated. In figure 6.3 and 6.4 the distributions can be seen. For the distribution of the professional stations, we can see how there is a peak frequency at the 10-20 km. Interestingly, being very close to a professional station is also a quite statistical unique feature, which has relatively low frequency of happening. In both the professional and amateur distances, there is a long tail. There are a lot of incidental outliers. For the professional station this can be seen at that after 40 km there are still some stations. In the amateur distance, it is clearer to see that there are more outliers already after 10 km. The amateur station distances peak around the 0 m and seem to have a log normal or exponential distribution from the looks of the KDE. The Professional station KDE looks (from visual inspection) like a bimodal Gamma distribution, with a short rise to the peak and a long decay to 0 frequency after the peak.

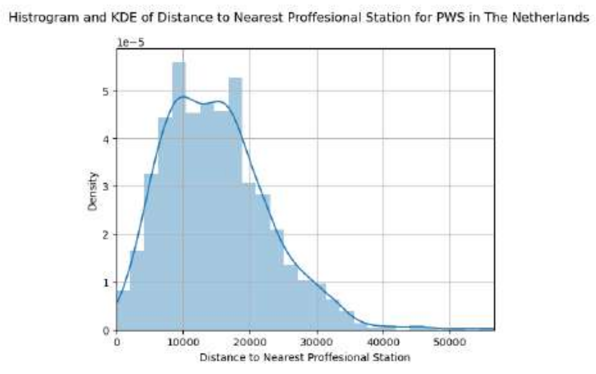


Figure 6.3: Histogram of Distance to Proffesional stations

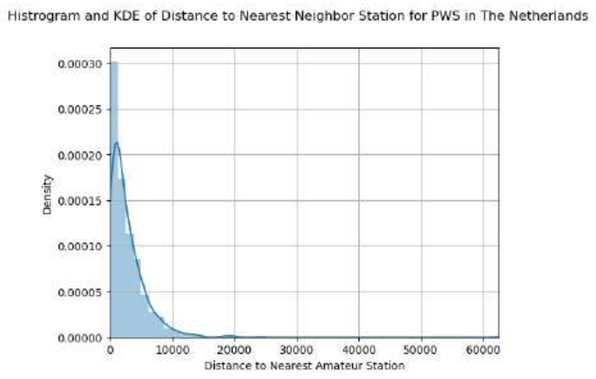


Figure 6.4: Histogram of Distance to Amateur stations

Looking at the distribution, we can take a cutoff values and plot the stations that are after this value. Spatial uniqueness can be approached in two ways, we can take a cutoff value based on the distribution and everything after this value is seen as statistical outliers/Unique. But the distribution can also be used to compare stations on a continuum. We are able to link the distance metric to how frequent this distance would occur, the station with a lower frequency metric is the one who is “More” Unique.

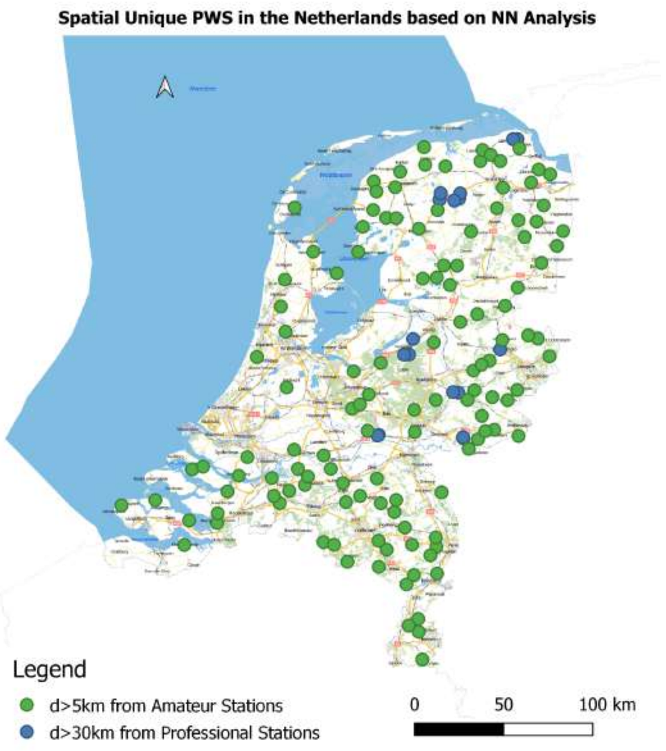

Spatial Unique PWS in the Netherlands based on NN Analysis
Legend
d>5km from Amateur Stations
d>30km from Professional Stations
0
50
100 km

Figure 6.5: Spatial Unique Personal Weather Stations in the Netherlands

From inspection, we chose a cutoff value of 30 km for the professional stations metric and 5 km for the amateur station distances. After plotting, this results in the spatial unique stations in the Netherlanders. There were 2 stations in the Netherlands that meet both criteria, these stations are in “Ter Apel” and “Vorden”, Both in North East of the Netherlands. In figure 6.5 all the stations that meet one of the criteria are plotted over the map of the Netherlands:

6.1.1. Location

The “Landelijk Grondgebruiksbestand Nederland” (LGN) is a geospatial dataset that provides information about land use in the Netherlands. It classifies land into categories such as built-up areas, agricultural land, forests, nature reserves, and water bodies.

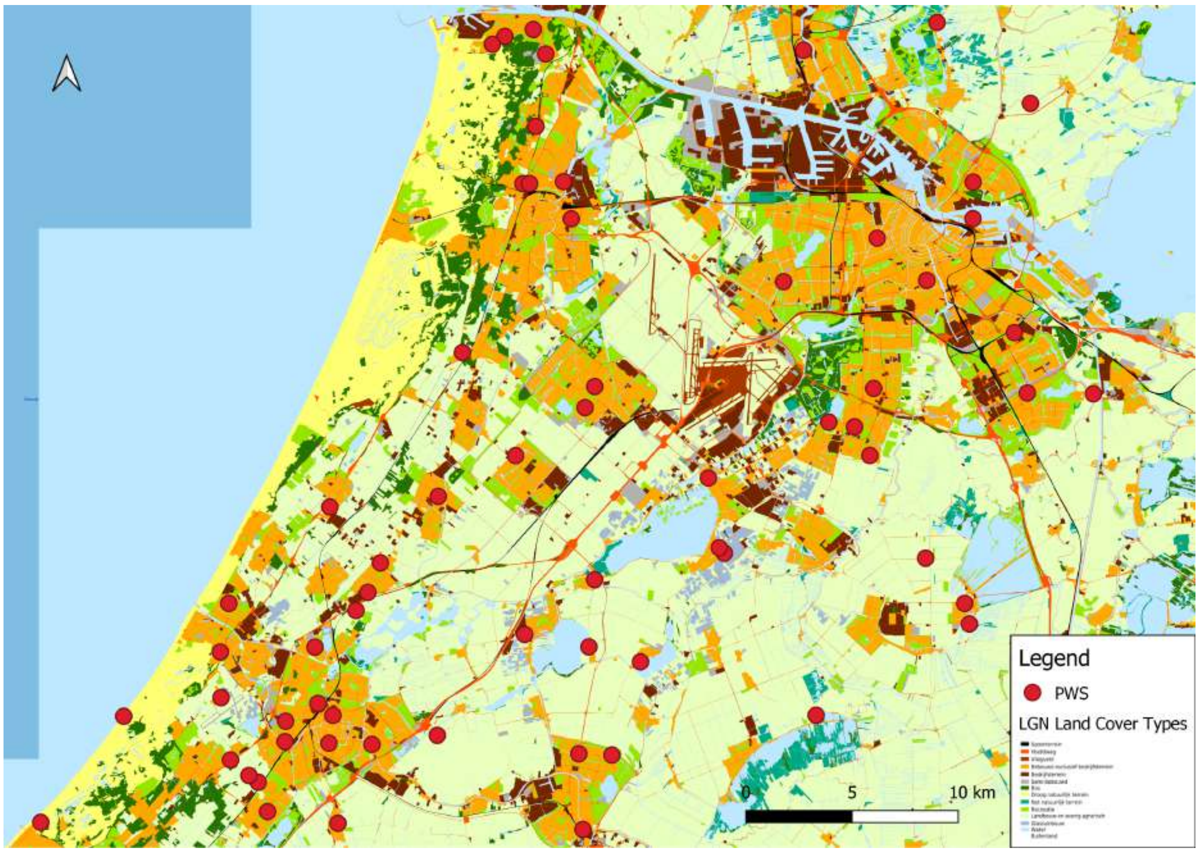


Figure 6.6: Land cover Type From the LGN2021 Dataset and the PWS Locations from the WOW-NL Platform in the Randstad

To determine which personal weather station (PWS) is in a specific land cover type using the LGN, we can overlay the PWS location data with the LGN dataset. By comparing the coordinates of the PWS with the spatial boundaries and classifications of land cover in the LGN, we can identify the land cover type in which each PWS is located. In figure 6.6 the land cover types of south of Holland are used as an example. as we can see, the stations can be linked to a land cover type.

This analysis can provide insights into the relationship between weather observations from PWS and the surrounding land cover characteristics. For example, we can examine how weather patterns differ between urban PWS (located in built-up areas) and rural PWS (in agricultural or natural areas). This can bring value to the user which might be interested in some type of land cover.

After doing the raster analysis, we got the results. From the 52 categories in the LGN2021 database, the stations only span over 22 of the categories. The distribution over the 22 categories has been plotted in figure 6.7. The colors correspond to what has been standardized by the WUR in the LGN database for each category. In the legend, the percentages are bold for each category. As we can see, there are two obvious anomalies, land type 18 and 23. Land type 18 (Red) correspond to “buildings in primarily built-up areas” and Land type 23 (Green) to “Grass in primarily built-up areas”. Interestingly, this means that 72.4 % of the weather stations in the Netherlands are in “primary build-up areas” which is the same as the urban space class like defined in the KNMI metadata. This does make sense since most weather stations are managed by amateurs that likely want to set up the device at home or close to an area they work or live. The LGN also makes a separation between urban space by also having ’secondary build-up areas’. 6.29% of the stations are located in this area.

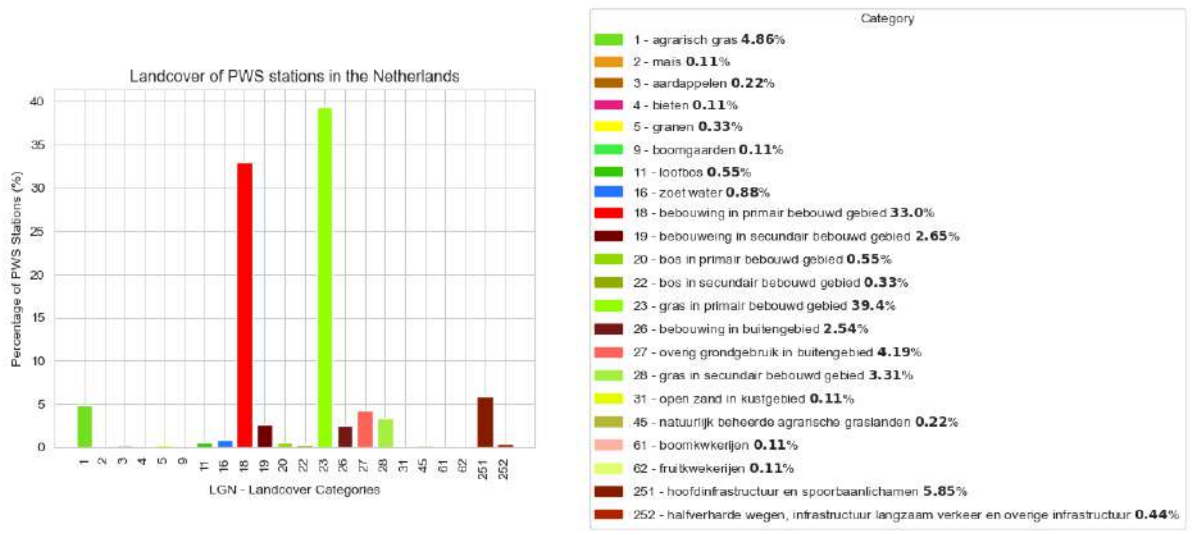


Figure 6.7: Land cover Type based on LGN2021 for all PWS in the Netherlands

The third and fourth-biggest land cover types are land type 251 (Brown) which is 'main infrastructure and railway bodies' at 5.85% and land type 1 (Green) which is 'agricultural grass' at 4.86%. In the rest, there are small amounts of stations in interesting land cover type. For example, 'open sand in coastal areas' or the classification based on crop production on the land cover. These all can be interesting from a value point of view for use cases where these locations are areas of interest.

6.1.2. Discussion

As seen in the results, a way to determine the uniqueness category for the value of a PWS has been performed. We are able to statistically determine the spatial uniqueness and location based uniqueness solely based on the location coordinates of the stations. Firstly, Statistical Spatial Unique stations were identified through Nearest Neighbor Analysis, employing cutoff values derived from the metric's distribution. This method allowed for the identification of weather stations that exhibit distinctive spatial characteristics compared to their neighboring stations.Secondly, Location Unique stations were determined based on the land cover types using the LGN2021 database from Wageningen University Research (WUR). By mapping the stations to their respective land cover types, unique stations were identified based on their specific geographical locations.

There is however a problem with the dataset. When reading in the device registration form provided by the WOW-NL, the coordinates of the station can be manually filled in. KNMI gives the advice to auto-locate, but when not choosing this option, one can be based on a postal code or address to determine the PWS location. The idea is then to use the postal code as a starting point and then change the pointer to the exact location of the station. When the station is located in a rural area, the device location solely is based on the user's input, which can be less reliable [19]. When an auto locates option is used in all stations, the procedure we propose needs no other human input and can objectively determine one of the 5 components of Value for a Personal Weather Station.

The analysis revealed a concentration of Personal Weather Stations (PWS) in primary built-up areas, with approximately 72.4% of the entire weather station network situated within urban areas. This finding highlights the preference for locating PWS in densely populated regions, potentially influenced by factors such as accessibility, user density, and the availability of infrastructure. Understanding the spatial distribution and uniqueness of weather stations is crucial for various applications, including spatial planning, climate studies, and data quality assessments. These insights can aid in optimizing the selection and placement of weather stations for specific research objectives, ensuring representative coverage and minimizing data redundancies.

6.2. Device Setup

In the current metadata system, the information about the device is structured in a classification. the classification is done based on multiple variables. By looking at the classification and isolating the variables, we are able to create an understanding of some aspects of the device setup.

6.2.1. Radiation Screen:

A radiation screen, also known as a radiation shield or radiation shield enclosure, is a critical component of a Personal Weather Station (PWS) that houses temperature and humidity sensors. Its primary purpose is to protect these sensors from direct solar radiation and other sources of unwanted heat, ensuring accurate and representative measurements of air temperature and humidity.

The design of a radiation screen aims to minimize the impact of external factors that can distort temp erature readings. It allows air to circulate freely around the sensors while shielding them from direct exposure to sunlight, radiant heat from surrounding surfaces, and reflected radiation. In figure 6.8 an example of a radiation screen on a Davis Weather Station can be seen. The radiation screen helps maintain a more stable and reliable thermal environment for the sensors to operate in. Common types of radiation screens used in PWS include the Stevenson screen and the naturally ventilated radiation shield.

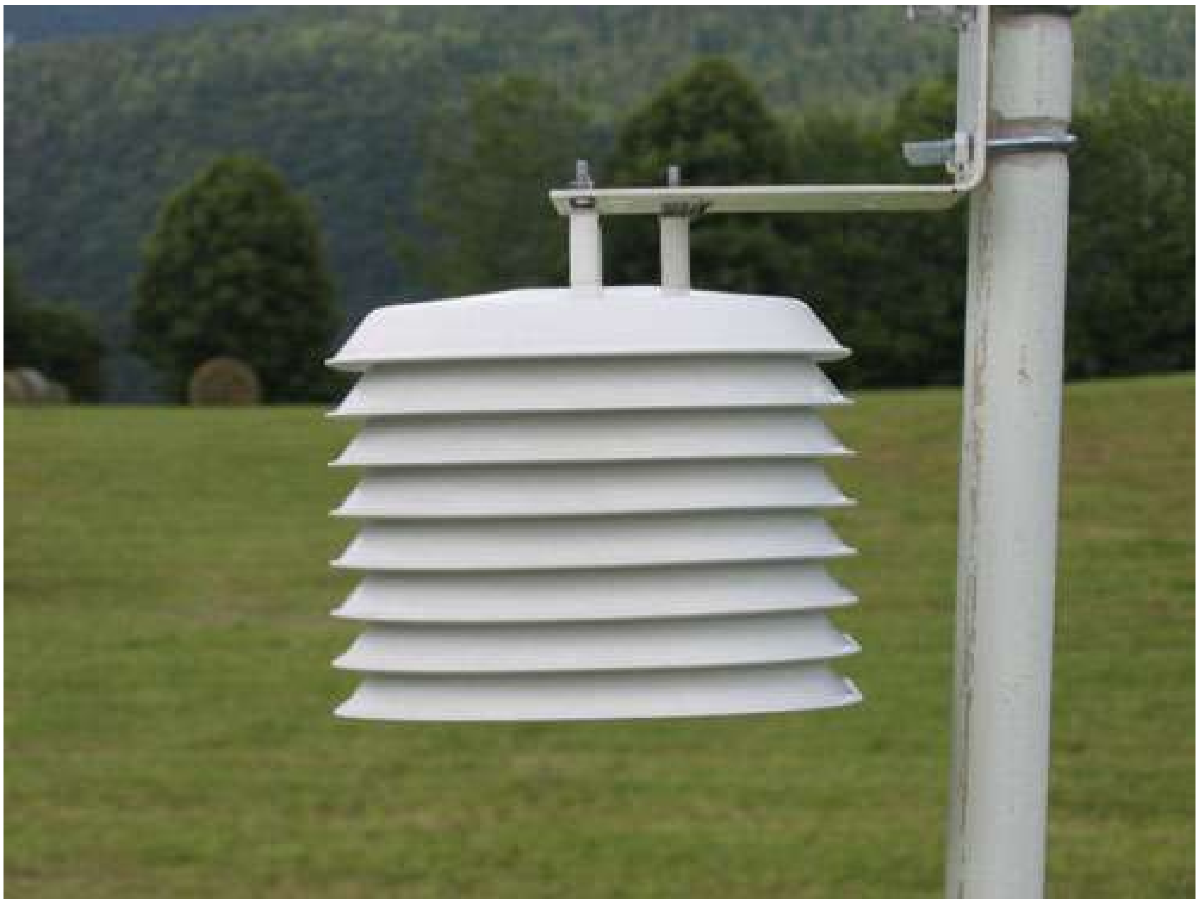


Figure 6.8: Example of a Radiation Screen in a Davis Weather Station [10]

The Stevenson screen is a double-louvered enclosure made of wood or another suitable material. It consists of a box-like structure with multiple horizontal and vertical slats. These slats create spaces or gaps that allow air to pass through while blocking direct sunlight. The design of the Stevenson screen ensures proper ventilation and prevents the accumulation of heat inside, providing an accurate representation of the ambient air temperature [20].

On the other hand, a naturally ventilated radiation shield (As can be seen in figure 6.8) is typically constructed with multiple layers of horizontal plates or vanes. These plates or vanes are angled to allow air to flow freely, promoting natural convection and heat dissipation. The design of this shield enhances the exchange of air while reducing the impact of external heat sources.

Both types of radiation screens are strategically positioned and mounted at an appropriate height to ensure accurate temperature and humidity measurements. They are often installed away from obstructions, such as buildings or trees, to minimize potential interference with air circulation and radiation exposure. The location and what type of screen is vaguely defined in the categorization. We are able to extract that a radiation screen is used, but not if it is a Stevenson screen.

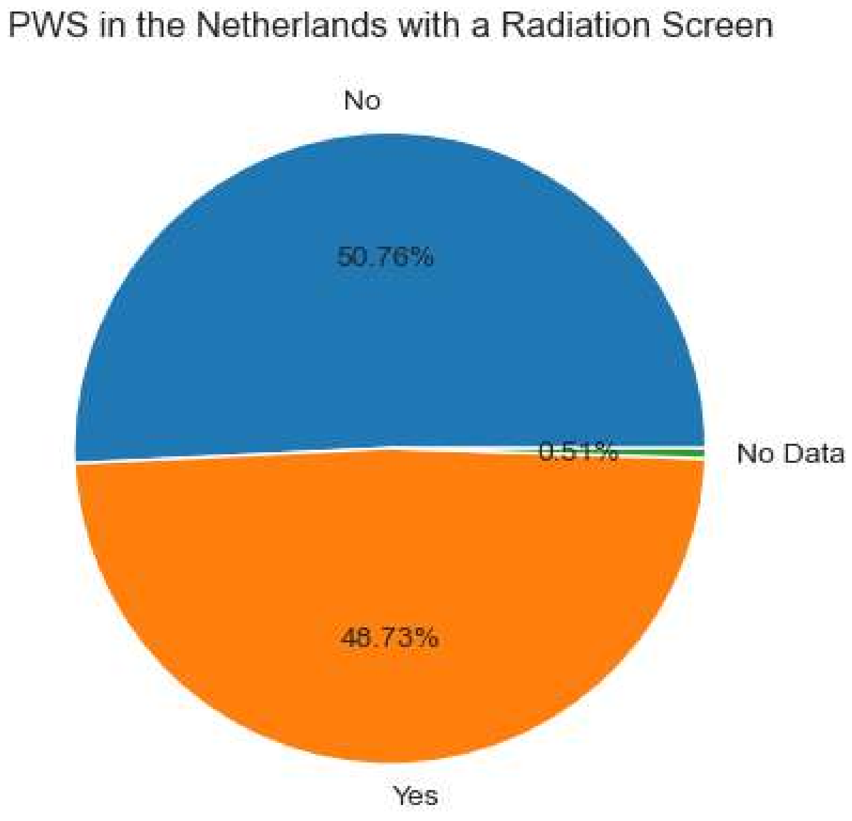


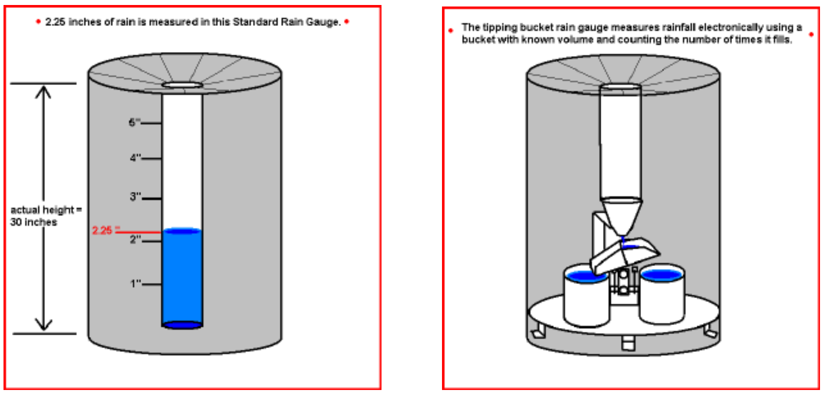
Figure 6.9: Percentage of Stations with and without a Radiation Screen in the Netherlands

By utilizing a radiation screen, PWS owners can enhance the reliability and accuracy of temperature and humidity measurements, providing more representative data for weather monitoring and analysis. Thus, also the value of the device setup. From extraction of information of the WOW-NL Network, we can determine that 48.2% of the PWS have a radiation screen of some sorts (including Stevenson Screens). A station with a radiation screen can be seen as a station that has more value because of more reliable device setup.

6.2.2. Precipitation

Rain gauges are devices used to measure the amount of precipitation, specifically rainfall, at a particular location. To ensure accurate measurements, rain gauges can be standardized and calibrated using a tipping bucket mechanism in a Personal Weather Station (PWS).

A Standard rain gauge is made out of a tube with a known, a consistent diameter over the height. By measuring the length of how filled the tube is, we get a sense of the volume that has fallen. The device must meet certain standards in terms of its dimensions, collection area, and measurement accuracy. An example of a 'Standard Rain Gauge' can be seen in figure 6.10a.


2.25 inches of rain is measured in this Standard Rain Gauge.
actual height = 30 inches
6"
4"
3"
2.25
2"
1"
The tipping bucket rain gauge measures rainfall electronically using a bucket with known volume and counting the number of times it fills.

(a) Standard Rain Gauge **(b)** Tipping Bucket Rain Gauge

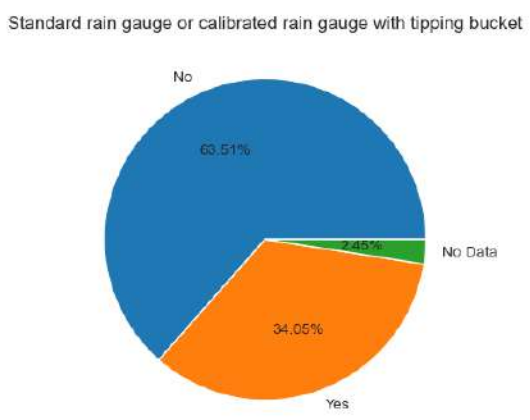
Figure 6.10: Both types of rain gauges[5]

Calibration, on the other hand, involves verifying and adjusting the rain gauge's measurement accuracy using a known reference or benchmark. In the case of a PWS, a commonly used method for calibration is the tipping bucket mechanism. A tipping bucket rain gauge consists of a container or bucket with a pivoting mechanism. As rainwater enters the bucket, it fills up and reaches a specific threshold, causing the bucket to tilt and discharge its contents. The tipping action is recorded electronically or mechanically, allowing for accurate measurement of rainfall. An illustration can be seen in figure 6.10b.

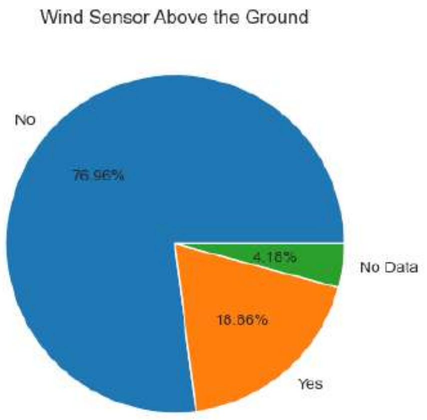
To calibrate a rain gauge with a tipping bucket mechanism, a known quantity of water, typically a standardized volume, is poured into the gauge. The bucket's tipping and discharge are observed and recorded. Any discrepancies in the measured quantity compared to the known reference can then be used to adjust and calibrate the rain gauge.

Calibration may involve making adjustments to the calibration factor or calibration coefficient of the rain gauge. This factor relates the recorded bucket tilts or discharge volumes to the actual amount of rainfall. Regular calibration is important to maintain the accuracy of rain gauges in PWS. Factors such as wear and tear, environmental conditions, and debris accumulation can affect their measurement precision over time. By periodically calibrating the rain gauge with a tipping bucket mechanism, PWS owners can ensure that their precipitation data remains reliable and consistent. The amount of devices in the Netherlands with this device setup can be seen in figure 6.11. As we can see 34.05% of the stations have either a standard rain gauge or a calibrated one with a tipping bucket. We are not able to assume these stations to be of higher value because the station without these two types of sensors might have a better one like a weighing gauge.

It's worth noting that the calibration process may vary depending on the specific design and manufacturer's instructions of the rain gauge used in the PWS. Following the manufacturer's guidelines and best practices for calibration is essential to achieve accurate rainfall measurements. Therefore, again important for determining the value of a PWS if these mechanisms are included in the device setup.



(a) Devices with standard or calibrated rain gauge in the Netherlands



(b) Wind sensors above the ground in the Netherlands

Figure 6.11: Sensor Information for the PWS in the Netherlands

6.2.3. Wind

A wind sensor in a Personal Weather Station (PWS) should be positioned on a pole or above the ground to ensure accurate and representative measurements of wind speed and direction. Here are the reasons why:

1. Unobstructed airflow: Placing the wind sensor on a pole or above the ground allows it to capture wind patterns and velocities without obstruction. Obstructions such as nearby buildings, trees, or other structures can create turbulence and interfere with the flow of wind, leading to inaccurate readings. By elevating the sensor, it is more likely to receive undisturbed and representative airflow, providing more reliable measurements.
2. Reduction of ground effects: The proximity of the wind sensor to the ground can introduce ground effects that affect wind measurements. Close to the ground, friction with the surface can slow down the wind speed and alter its direction. These ground effects are more prominent in low-level atmospheric layers and can cause significant discrepancies between the actual wind conditions and the measured values. Mounting the wind sensor higher above the ground minimizes these ground effects and improves the accuracy of the wind data.
3. Mitigation of local influences: Placing the wind sensor on a pole or elevated position helps reduce the influence of local features on wind measurements. Local topography, such as hills, valleys, or buildings, can create microclimates that differ from the broader atmospheric conditions. By elevating the sensor, it is less affected by these localized influences, allowing for measurements that better represent the prevailing wind patterns in the area.
4. Enhanced representativeness: A PWS aims to provide weather information that is representative of the surrounding area. Placing the wind sensor on a pole or above the ground helps ensure that the measured wind data represents the broader atmospheric conditions in the vicinity of the station. This is particularly important for applications such as wind resource assessments, microclimate studies, or weather forecasting, where accurate and representative wind information is crucial.

It is important to note that the height and specific installation requirements of the wind sensor can vary depending on the manufacturer's recommendations and the intended purpose of the PWS. Following the manufacturer's guidelines and best practices for mounting the wind sensor will help ensure accurate and reliable wind measurements in the PWS. Therefore, it is an important way to determine the value of the device setup. In the case of the Netherlands, 18.66% of the stations are above the ground or on a pole. The other 76.96% is defined in the classification as being located on a building or wall (more information about the classification can be read in the methods chapter).Station where the wind sensors are situated above the ground or even better on a pole have a higher value.

6.2.4. Discussion

As can be read in the previous results, the device setup is a complex system. There are a lot of parts to the setup like the sensor type or the accessories that can be added to the setup that can improve or worsen the output. A more systematic way to provide information about the device is needed. As we can also see and is discussed in section 5.2.3 in figure 6.2. The use of the free text box mostly has something to do with the device setup. For example, the device name is something which is one of the biggest signals. This means that the user does not have a place to put this information. The device setup can be very particular for each case, but standardization by being able to give up the device and model name is a good start. In the future, a more systematic device registration for a PWS network would enlighten us with a lot more information to base the 'value' of a personal weather station on. The basis should be to ask information that is not subjective, leading to comparability across stations. For example, by getting the device name and all the product names that are added to the setup, the network can automatically map the device to having radiation screens or what kind of sensor there are present with what precision based on the manufactures manuals.

6.3. Availability

In the metadata of the WOW-NL, the only information about availability is in the “observation hours” category. How this categorization works can be read in the methods. The information given is about when the stations report data to the network. As we can see in figure 6.12, most of the stations (48.2%) report at midnight. While the other 42.4% is unknown. This is already almost all stations. From 6.8% we know that the reporting moment is at 9:00 UTC. 0.9% reports between 6:00 and 9:00 UTC. Lastly, 1.6% we know have irregular reporting times.

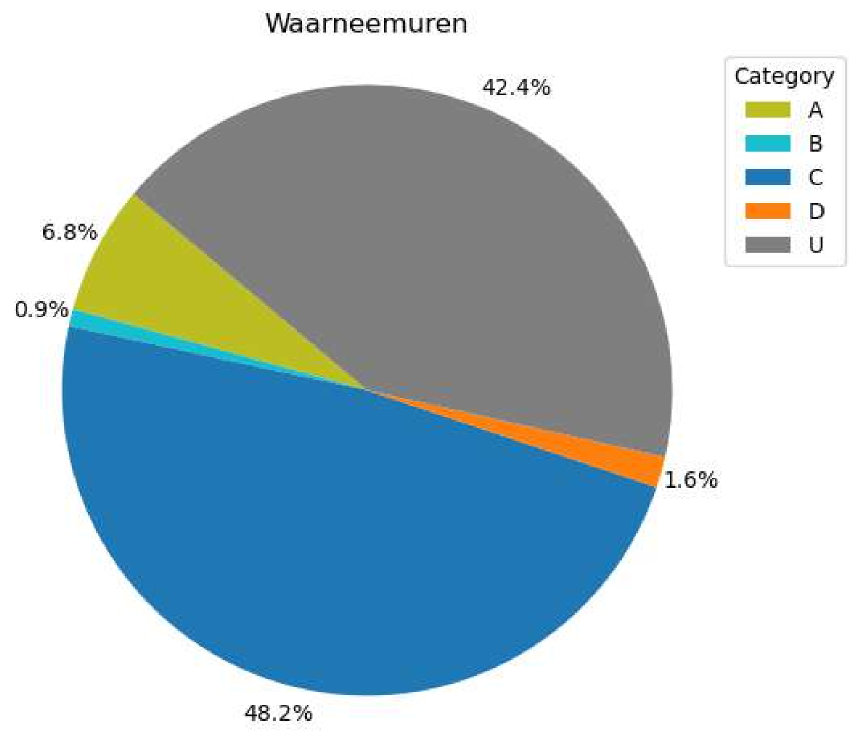


Figure 6.12: Pie Chart for the “Observation Hours” Metadata from WOW-NL

As we can see of the data a lot is unknown and even when we know when the reporting time is, it does not make sense why it is being asked for and not being calculated from the dataset or connection to the server. A lot of the station perhaps have a manual reporting option, making the metadata input only an ideal for the station and not the truth. This tries back to the device setup category in the previous results. We do not know enough about the context of the station to even base this conclusion to. This makes it hard for it to be a dataset to base the 'Availability' to in the Value Framework.

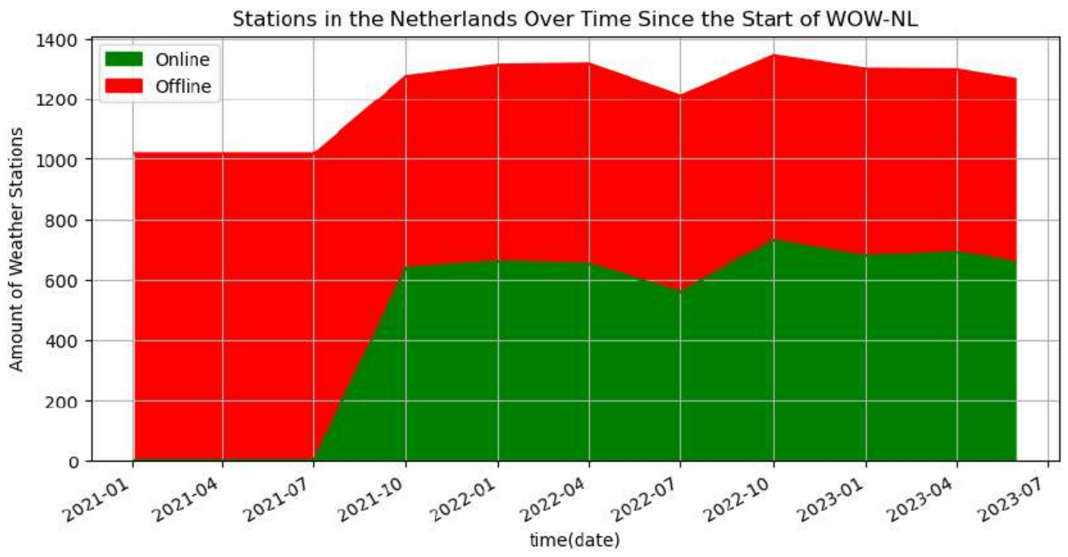


Figure 6.13: Station Online/Offline Count over Time Since the Start of the WOW-NL Network

The second part of availability is looking at the whole WOW-NL Network. How many stations are online over time? The WOWbot is able to extract all the stations in the Netherlands. We manually changed the portal to different dates and extracted the list of station IDs that were online and offline at each time.

After looking at the wow-nl network, we saw how the first data stations were added to the network was on 04 January 2021. This is only 2 years ago. With this information, we chose to look at how the network for each quarter. Quarters defined as: 1 January, 1 April, 1 July and 1 October. Q1 of 2021 is the first data where data was available, this is 04 January. In figure 6.13 we can see the results of the analysis. As we can see, in the beginning of the network, no stations where online and where reporting information. In the first 3 quarters of 2021 no data was reported on the wow-nl network. In q4 2021 the first batch of stations began reporting. This looks like this was the moment all stations at the same time were allowed to report information. In Q3 2022 there was a small dip of stations being removed from the network, after that in Q4 a spike occur reporting the highest stations population size of 1342 Stations in the network. Taking 2021-10 as a starting point, the average ration of online/all stations equals

Metadata	Ratio
2021-10-01	0.499215
2022-01-01	0.497330
2022-04-01	0.504943
2022-07-01	0.538017
2022-10-01	0.455291
2023-01-01	0.476888
2023-04-01	0.466049
2023-05-29	0.481423
Average	**0.49**

Table 6.1: Ratio of Online Weather Stations in the WOW-NL over time

The availability of the WOW-NL network has been examined over time. The analysis reveals a significant amount of unknown information regarding availability. While we have knowledge of the reporting time for 57.6% of the network, the frequency of the time series remains undefined. This lack of information limits our understanding of the temporal resolution of the data. It is important to note that the WOW-NL network is relatively new, having been established in 2021, with data reporting commencing in the fourth quarter of that year. As a young network, it is expected that there are still ongoing developments and improvements being made to enhance its functionality and data accessibility. One notable finding is that, on average, approximately 49% of the stations in the network are online and available at any given time. This means that about half of the stations may be intermittently or temporarily unavailable over time. Interestingly, this does not align with one of the metadata entries called "activity" which says that 99.3% of the stations are active. It is worth emphasizing that this unavailability does not pertain to the same stations consistently, but rather varies among different stations based on their reporting frequency.

These findings underscore the importance of continuous monitoring and evaluation of network availability to ensure the reliability and usefulness of the collected data. As the WOW-NL network evolves and matures, efforts should be directed towards improving data availability and establishing standardized reporting frequencies for a more comprehensive and consistent dataset. This dataset can be for example be generated by the network based on the connectivity of the devices to their server, making the result the ground truth and not an ideal. As a second measure, the time series output can also be checked. Sometimes the station can report only zeros. In this case, the network will again be able to detect this.

6.4. Credibility

Credibility is a hard value to assess. Most of the time, credibility is based on historical evidence of structured action which lead to good results. If someone has many times in the past showed competence, it is reasonable to assume the same competence can be expected this time. There is one downside to credibility, it is very easily lost when not consistent. These are the two factors which define credibility in our definition as outlined in the method chapter introducing the framework.

1. Historical Evidence
2. Consistency

The same way the analogy of a person in the previous indention holds, the credibility of a PWS can be treated the same way. Then again, Personal Weather Stations are managed by people, and we are assessing their credibility to gain information about the quality and value of a particular weather station. The first thing we are able to get from the metadata we already assemble is the motivation for the weather station, which should be the reason for the maintainer to manage the station well. Secondly, for some stations, we have a website that is linked to the project or organization of the person managing the station. From this, we are able to see what kind of organizations and people have stations. As we can see in figure 6.14, Waterlab and the TU Delft is a prominent signal in the website database of stations in the Netherlands. The biggest signal however is the 'mijneigenweer' website.

Mijneigenweer.nl is a site that offers you the possibility to have a complete plug play weather website. Your own weather data online on a complete weather website without hassle. The complete setup including upload of your weather data is arranged by us. This means there are a lot of amateurs that use this system for their weather station registration.

There are a lot of small websites, as can be seen in figure 6.14. This is because of the uniqueness of the websites for a personal weather station that is incidentally places by a citizen. For example, 'weerstationziewent'. The format of *"weersation + locationname"* can be seen multiple times as format for the websites.

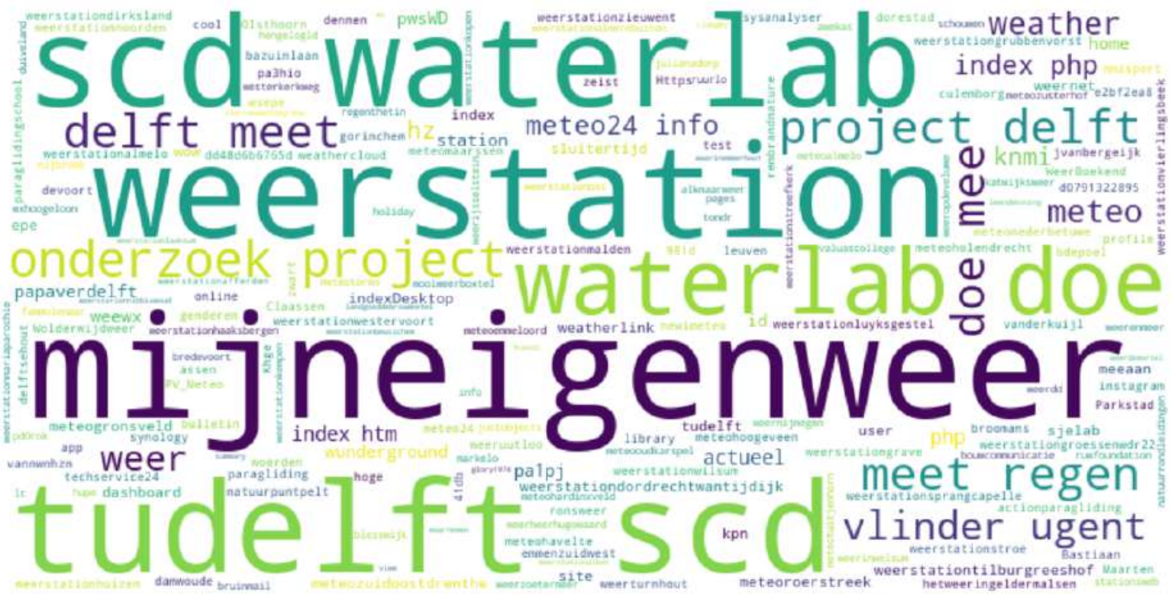


Figure 6.14: Websites Data of PWS Owners in the Netherlands

Next to the websites, there is also an option to add the name of your organization to the station metadata. From this database, we are able to also gain a database of organizations that are contributing to the PWS Network. From all the stations, 23.6% added an organization name to the station. In figure 6.15 the organizations based on frequency are plotted. We can immediately see that the signals are much clearer on the organizations that are related to the stations in the network. For example, 'VWK'

(Dutch Society for Meteorology and Climatology), 'Ghent University' and also again off course 'Delft' are clearly represented. Next to that, HZ University of Applied Sciences can also be seen. Taking a look at figure 6.15 will give an overview of the kind of organizations that use the WOW-NL network, especially when looking at the smaller frequency organizations.

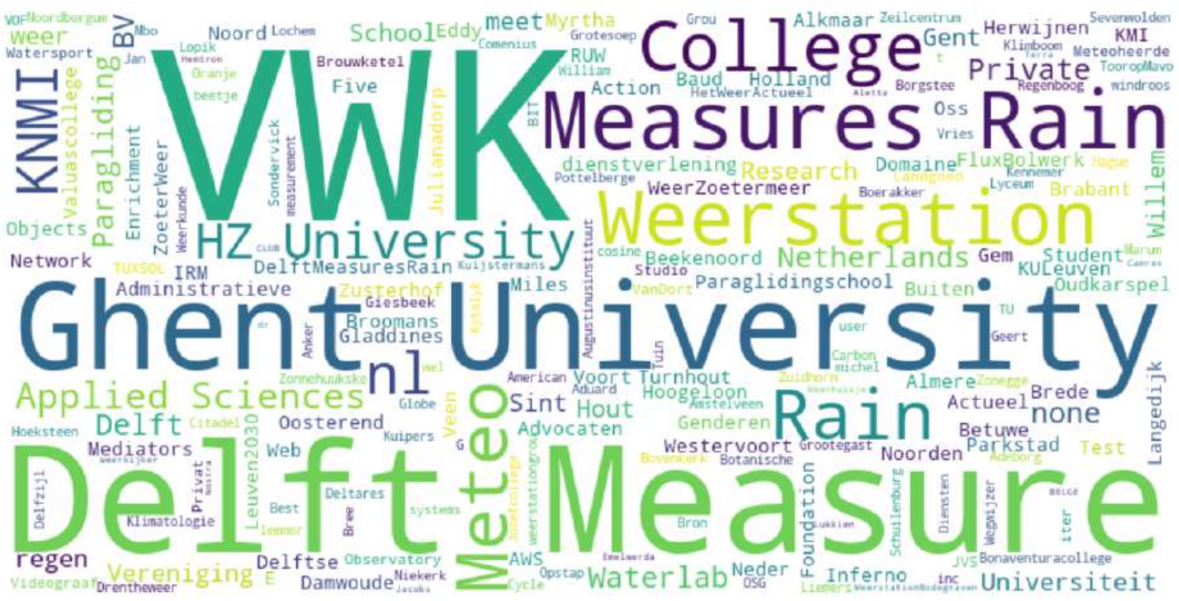


Figure 6.15: Organization Data of PWS Owners in the Netherlands

With this database, we get to know what organizations are managing the stations. To get to a system to assess the credibility, we need to correlate the value of the stations to the organization or maintainer of the stations. This way next time we see a station that is managed by an owner that has a good track record, this will add to the value of that station. This poses a problem, as we have seen in the previous analyses there is no good understanding of value yet.

This is where the framework for assessing the value of a personal weather station comes in. This framework gives you 4 other categories to look at that gives us the assessment of value we want. This means that for the future a more systematic way of managing the other 4 categories of value (Accuracy, Availability, Uniqueness, Device Setup) will be the base of the Credibility category. The only step to take from there will be to correlate the value of the station to the users. Giving us a way to determine the credibility of users based on a track record of consistency.

Something we are able to do at this stage of the PWS network is being resourceful. When we look at the rating system, it is using 3 metadata entries to base a 5-star rating on. When metadata is not delivered, this is harshly punished in the rating. In some sense the rating system set in place by the KNMI is not a value rating or a quality rating but rather a credibility rating. When we look at it in this way, we can even expand the analysis to all metadata entries. How much has been delivered by the user about the station, the more it is, the higher the credibility. With this, we can create a proxy for credibility, even though the official way would be to use the other value categories to determine the value and correlate it to the users. This still gives a good initial look at how credible the stations are. Note how the limitation is that each station is seen to be owned by a different person. When we do the analyses, we get the bar chart in figure 6.16.

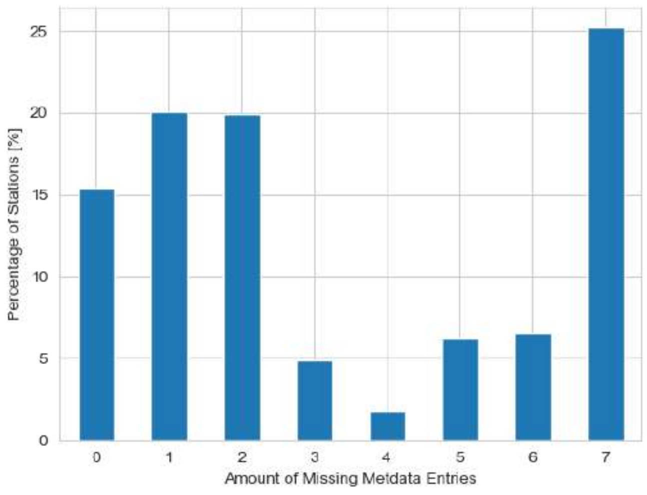
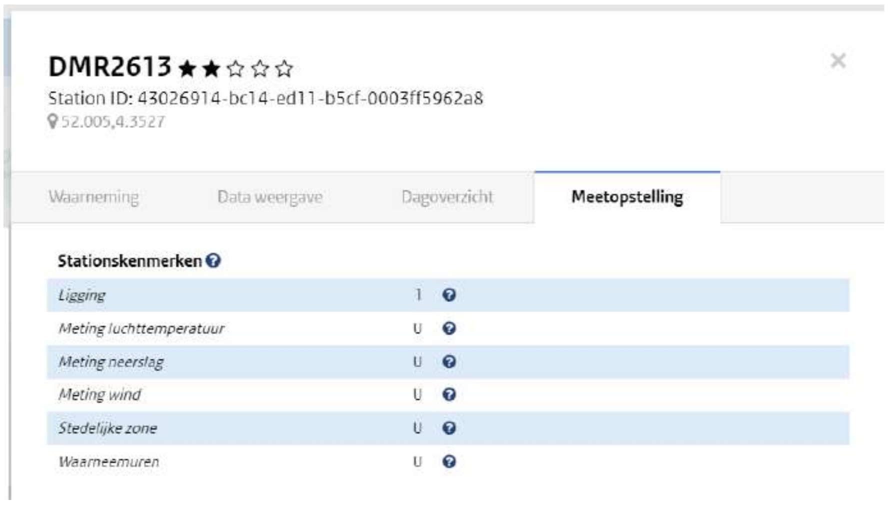


Figure 6.16: Percentage of PWS stations in the Netherlands for a certain amount of missing metadata entries as a bar chart

Interestingly, we do notice how the system created some peculiar results. For example, one of the stations managed by the TU delft in the water lab project has been found in the system. It is being managed for a project, which does give the idea that it is a high credible maintainer. But when we look at its metadata and rating we see how for example station DMR2613 has 5/6 metadata entries missing. This is an example of a weather station assumed to be credible because a university is maintaining it, but in real life based on the metadata is not as managed well. To make a conclusion in the future the framework can be implemented to be able to assess the other 4 categories like discussed in the previous sections and following correlate these results with the usernames in the network. Therefore, the 'Credibility' categories can only be rightly assessed once a good way to determine the value of station is possible.


DMR2613 ★★☆☆☆

Station ID: 43026914-bc14-ed11-b5cf-0003ff5962a8

52.005,4.3527

Stationskenmerken

Ligging	1
Meting luchttemperatuur	U
Meting neerslag	U
Meting wind	U
Stedelijke zone	U
Waarneemuren	U

Figure 6.17: TU Delft PWS example of missing Metadata entries

Credibility is an important category within the assessment of weather stations, as it relates to the perceived value of a station and how it correlates with user trust. This category not only defines the value of a station but also considers the user's perspective. It proves particularly useful when users have multiple stations and one needs to make conclusions about their value without relying solely on other categories or information. It is worth noting that organizations such as TU Delft and Gent University, which frequently utilize the network, often assert ownership of specific stations. However, despite being managed by reputable institutions, it is essential to objectively evaluate these stations to ensure their credibility. This evaluation becomes particularly relevant when considering instances of missing metadata, as exemplified by the case of TU Delft in the results chapter.

Objective assessment of stations becomes crucial in verifying their value and maintaining the overall credibility of the network. It serves as a means to double-check and validate the information provided by these organizations. By conducting thorough assessments and addressing any missing or incomplete metadata, the network can enhance its reliability and foster greater confidence among users and stakeholders. However, the condition for assessing credibility is a good base to assessing objective "Value" of a PWS and then link that to the personas behind the stations.

7 Conclusion

Significant progress has been made in developing the basis of a framework to assess the value of a Personal Weather Station. The research process involved overcoming numerous obstacles, which have provided valuable insights and lessons for the future development of a more refined PWS network assessment system. A framework comprising five values—Device Setup, Uniqueness, Availability, Credibility, and Accuracy—has been developed.

To answer, our first research question: “How do current PWS networks quantify the value of a PWS station? And how good or reliable is this rating system?” we executed the first ever analysis of a national PWS network enabled by a web bot for the WOW-NL network. A total of 1031 weather stations in the Netherlands were successfully extracted using the WOWbot. A database with all the metadata about the personal weather stations in the Netherlands has been generated, eventually 981 stations were accessible using the bot even though 1031 were extracted from the map. The database can be accessed and downloaded for further use at `https://github.com/AdilAyi/WOWbot/blob/main/Final_Database_Metadata_WOWNL.csv`.

The identification of challenges and issues within the existing metadata structure has shed light on areas that require attention and resolution. The resolutions underscore the importance of addressing ambiguities, inconsistencies, and subjective assessments in the metadata, particularly in relation to the rating system based on three variables in the current system. Answering the second part of the question, we conclude the rating system to be unreliable because of its dependency on 3 main variable, with the main one to be location, which is unknown 35.7% of the time. The other 2 variables are sensor classifications that are both unknown as ~ 40% of the time. When an entry is unknown, the rating penalize this by removing stars, making the rating not really a quality rating but more of a commitment rating. By refining these aspects, the network can move closer to a more objective and accurate assessment of the stations’ value. Recommendation for how to actualize these improvements are discussed in depth for each value in the discussion subsections in each section, and an explicit feature summary can be read in table 7.1.

Where	Improvements
Metadata	• Device Model Registration • Device add-on registration, like a radiation screen. • Accurate GPS coordinates as basis for all other location based metadata • Location Particularities: [Roof, Wall, Garden] to clarify ambiguity from the GPS coordinates • Citizen Registration with a Profile, so more information about the person can be gathered about motivation • Device Maintenance Registration: What was the last time the device has been recalibrated?
PWS Network	• Track Device Connectivity to Network • Systemize Accuracy Metrics from the Data Output in the Platform itself • Gamify the Experience with transparency on the Platform • Have credibility metrics in the system, leaderboards of users can be one of the features that not only tracks this but also again increases commitment by the citizens • Having multiple rating systems, For example by value category as defined in the framework we propose.

Table 7.1: Future Improvements for Weather Citizen Science Networks

With the results of the analysis, we got an overview of all information known about a PWS in networks, from which we were able to detect clusters that describe one value of a personal weather station. From this process we made a framework but also saw clearly how there remain some information gaps. With the bots, we created proxies to approximate these gaps. Eventually, we were able to answer the second research question: “How valuable are the personal weather stations in the Netherlands based on our framework?”. For each Value (except “Accuracy”) a map or a selection can be made based on metadata or proxies that estimate the value. For example, spatially unique station are calculated based on a cutoff value from a Nearest Neighbor Analysis. The “availability” of stations over time since the Q4 2021 has been calculated making selection possible, stations that were live longer therefore have a higher availability. Devices with or without radiation screen and tipping bucket can be selected. Lastly, station credibility has been approximated by measuring the amount of missing metadata entries in the WOW network. Therefore, being able to pick stations based on 8 levels of credibility. Full detailed assessment of each Value Category can be read in chapter 6.

While the current framework has laid the groundwork, it is important to acknowledge that further advancements are necessary in the PWS network and the framework itself. For example, in the framework, the definitions of the values are not independent of each other but should be seen as different aspects of the stations. As an example, ’Accuracy’ is defined as the accuracy of the station weather data output. Even though it sounds like it is independent of the other values because it is only looking at the data output, the device setup or the location where the station is (Uniqueness) can cause the accuracy to change drastically based on the literature outlined in “section 6.2”. It is important to know that a lot of complications for the assessment of the value of a station can be solved by further development of the device registration in these citizen science networks. The potential is there and with adequate development these value categories can be assessed more and more accurately in the future.

The utilization of automated data extraction through the bot has demonstrated its potential in aiding the extraction of valuable information from citizen science networks like WOW-NL. This technology has significantly enhanced the efficiency and effectiveness of extracting data from numerous stations, providing a valuable tool for future research and development. However, “the WOWbot” is only applicable on the WOW-NL network and is therefore limited to this network. If anything, the results of this thesis highlight why data transparency for citizen networks is essential. We now had to innovate the wheel to be able to generate a dataset for all Personal weather stations in the Netherlands, delaying the findings of inadequate station registrations and metadata definitions for creating a framework to determine the value of a station. To advance the weather citizen science field, the ’citizen’ should be empowered to also benefit from the network he is contributing in. This transparency will spur higher quality of citizens that participate and commitment around the network because of the transparency. Also, the network maintainer will then have a higher responsibility to improve the quality of their service because of the social pressure, therefore improving the citizen science field.

Looking ahead, it is crucial to continue building upon the foundations established in this study. Further research and improvements should focus on resolving the identified issues, refining the assessment criteria, and enhancing the quality and completeness of the metadata. By doing so, the PWS network can unlock its untapped potential, offering increased spatial resolution for weather and climate observations. This progress will contribute to the broader goals of utilizing PWS data for research and development, ultimately advancing our understanding of weather and climate patterns.

Future Research:

1. *Do the Nearest Neighbor Analysis in section 6.1 for more neighbors(k). Right now, a clustering of 2 stations in an isolated area is still considered as a station that has a station close by. While perhaps an average of the nearest neighbor and second neighbor, these stations would still be considered as unique.*
2. *In the Land Type Analysis from section 6.2, The Stations could be bordering other land types that could tell us more information about its uniqueness. For example, a weather stations next to a railroad would be interesting to look at. Looking at the land cover type that is closest to the station next to the landcovertype it is on would be an interesting analysis to do.*
3. *Device Setup still is very vague on what is a good setup and what is bad to base an assessment of the category on. This is because there is not enough information on the effect of certain add-ons or setup configurations on the device "value" or directly on the Accuracy. Research on the Device itself and what would be considered a good device setup for a citizen can help with creating objective metadata questions when registering on a citizen network.*
4. *Work needs to be done on how to combine the value categories from the framework. Right now, the framework is open to subjectivity from the user of the framework. If we are able to tell for which use case which value category is important, the use of the framework would be improved because it would not be up to the user to decide (which is prone to potential misjudgment). Right now, the main use case is for selecting "valuable" Weather stations from a pre-selected population that could be bound to geographical or temporal filters.*

References

[1] URL: https://www.wunderground.com/pws/about.

[2] Peter Ackroyd. *Chatterton*. Grove Press, 1996.

[3] Simon Bell. “Quantifying uncertainty in citizen weather data”. 2015. URL: https://publications.aston.ac.uk/id/eprint/26693/1/Bell_Simon_J._2015.pdf.

[4] Tiago Bianchi. *Human and bot web traffic share 2021*. Nov. 2022. URL: https://www.statista.com/statistics/1264226/human-and-bot-web-traffic-share/.

[5] Derek W. Brown. *Measuring Rain and Drizzle vortex.plymouth.edu*. https://vortex.plymouth.edu/dept/tutorials/precip/precip3a.html. [Accessed 12-Jun-2023]. 2003.

[6] Guido Caldarelli et al. “The role of bot squads in the political propaganda on Twitter”. In: *Communications Physics* 3.1 (May 2020). DOI: 10.1038/s42005-020-0340-4. URL: https://doi.org/10.1038/s42005-020-0340-4.

[7] Google Search Central. *Introduction to robots.txt*. 2023. URL: https://developers.google.com/search/docs/crawling-indexing/robots/intro.

[8] CFDynamics. *What is a web robot? - CFDynamics — cfdynamics.com*. https://cfdynamics.com/manage/knowledgebase/article/515/what-is-a-web-robot-/. [Accessed 19-Jun-2023].

[9] Shaumik Daityari. *Selenium with Python Tutorial: How to run Automated Tests | BrowserStack — browserstack.com*. [Accessed 11-Jun-2023]. 2023. URL: https://www.browserstack.com/guide/python-selenium-to-run-web-automation-test#:~:text=Selenium%20is%20an%20open%20source,%2C%20Ruby%2C%20JavaScript%2C%20etc..

[10] Davis. *Davis Europe | Shop - davis-europe.nl*. https://www.davis-europe.nl/product/davis-7714-passive-radiation-shield/. [Accessed 12-Jun-2023].

[11] Wolfram Donat. “The Web Bot”. In: *Learn Raspberry Pi Programming with Python*. Apress, 2018, pp. 119–145. DOI: 10.1007/978-1-4842-3769-4_5. URL: https://doi.org/10.1007/978-1-4842-3769-4_5.

[12] Ken Dunham and Jim Melnick. *Malicious bots: an inside look into the cyber-criminal underground of the internet*. Auerbach Publications, 2008.

[13] Ujaval Gandhi. *Nearest Neighbor Analysis x2014; QGIS Tutorials and Tips — qgistutorials.com*. [Accessed 11-Jun-2023]. 2023. URL: https://www.qgistutorials.com/en/docs/nearest_neighbor_analysis.html.

[14] Mohammad Gharesifard, Uta Wehn, and Pieter van der Zaag. “Towards benchmarking citizen observatories: Features and functioning of online amateur weather networks”. In: *Journal of Environmental Management* 193 (2017), pp. 381–393. ISSN: 0301-4797. DOI: https://doi.org/10.1016/j.jenvman.2017.02.003. URL: https://www.sciencedirect.com/science/article/pii/S030147971730107X.

[15] *Guide to Instruments and Methods of Observation (WMO-No. 8)*. World Meteorological Organization (WMO), 2021. URL: https://library.wmo.int/index.php?lvl=notice_display&id=12407.

[16] Gerard Hazeu et al. “Landelijk Grondgebruiksbestand Nederland 2021 (LGN2021): achtergronden, methodiek en validatie”. In: (2023).

[17] KNMI. 2023. URL: https://wow.knmi.nl/over-wow-nl.

[18] KNMI. 2023. URL: https://wow.knmi.nl/meedoen/classificatie.

[19] KNMI. *Invulinstructie: aanmelden station op wow.metoffice.gov.uk*. 2021. URL: https://wow.knmi.nl/data/wow_invulinstructie.pdf.

[20] Edward Mawley. “Repobt on temperatures in two different patterns of stevenson screens”. In: *Quarterly Journal of the Royal Meteorological Society* 10.49 (July 2007), pp. 1–7. DOI: `10.1002/qj.4970104902`. URL: `https://doi.org/10.1002/qj.4970104902`.

[21] Malte Müller et al. “AROME-MetCoOp: A Nordic convective-scale operational weather prediction model”. In: *Weather and Forecasting* 32.2 (2017), pp. 609–627.

[22] Thomas N Nipen et al. “Adopting citizen observations in operational weather prediction”. In: *Bulletin of the American Meteorological Society* 101.1 (2020), E43–E57.

[23] Rahul Shah. *How to Build Word Cloud in Python? — analyticsvidhya.com*. [Accessed 11-Jun-2023]. 2023. URL: `https://www.analyticsvidhya.com/blog/2021/05/how-to-build-word-cloud-in-python/`.

[24] Computer Science Wiki. *Web crawler functions - Computer Science Wiki — computersciencewiki.org*. `https://computersciencewiki.org/index.php/Web_crawler_functions`. [Accessed 19-Jun-2023].

[25] Amber Zamora. “Making Room for Big Data: Web Scraping and an Affirmative Right to Access Publicly Available Information Online”. In: *Journal of Business, Entrepreneurship and the Law* 12 (2019), p. 203. URL: `https://heinonline.org/HOL/Page?collection=journals&handle=hein.journals/jbelw12&id=203&men_tab=srchresults`.

A
The WOWbot code

Source Code for the WOWbot, also available at `https://github.com/AdilAyi/WOWbot`

```python

from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import Select
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager

from termcolor import colored
from bs4 import BeautifulSoup
import random
import numpy as np
import pandas as pd
import pickle
import time

import geopandas as gpd
from shapely.geometry import shape,mapping, Point, Polygon, MultiPolygon

def tableDataText(table):
    rows = []
    trs = table.find_all('tr')
    for tr in trs[1:]: # for every table row
        rows.append(tr.find_all('td')[0].get_text(strip=True)[-1]) # data row
    return rows

def tableDataText2(table):
    rows = []
    trs = table.find_all('tr')
    for tr in trs[1:]: # for every table row
        rows.append(tr.find_all('td')[1].get_text(strip=True)) # data row
    return rows

def get_metadata(all_id, random_sec=0, all_id_old=None):
    "station_id now be a batch"
    "Note that all_id and all_id_old must be a list of ids"
    "random_sec makes it possible to let the bot to simulate human behaviour by having random
         delay in between metadata extraction"
    "A random_sec of 10 makes it that a random amount of seconds in between 0-10 will be
        taken in between extraction"

    #variable_names as defined in the wow_NL
    variable_names = ['Station ID',
    'Positie',
    'Hoogte (boven zeeniveau)',
    'Tijdzone',
    'Actief station',
    'Data downloaden toegestaan?',
    'Website',
    'Motivatie station',
    'Officieel station',
    'Organisatie',
    'Ligging',
    'Meting luchttemperatuur',
    'Meting neerslag',
    'Meting wind',
    'Stedelijke zone',
    'Waarneemuren',
    'lat',
    'lon',
    'star_rating',
    'description',
    'extra_info']

    #initialize dataset
    df = pd.DataFrame(columns=variable_names)
    ids_error = []

    #NL shape
    NL_shp = gpd.read_file("shapefile/nl_10km.shp")
    NL_shp = NL_shp.to_crs(4326) #change rd to wg84 crs:4326

    j=0
    for station_id in all_id:
        if station_id not in all_id_old:
            try:
                #setup driver
                #PATH = "C:\Program Files (x86)\chromedriver.exe"
                #driver = webdriver.Chrome(PATH)
                op = webdriver.ChromeOptions()
                op.add_argument('headless') #so it only happens in the console and the
                    website does not get opened

                driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()),
                    options=op)
                driver.maximize_window()

                #go to site
                driver.get("https://wow.knmi.nl/#"+station_id)

                #get coordinates
                time.sleep(1)
                l = driver.find_element(By.XPATH, '//*[@id="myModalLabel"]/span[3]')
                lat, lon = float(l.text.split(',')[0]), float(l.text.split(',')[1])

                point = Point(lon, lat) # longitude, latitude

                # Alternative: if point.within(shape)
                if any(NL_shp.contains(point)):#check if in NL polygon
                    #go to metadata page
                    l2 = driver.find_element(By.XPATH, '//*[@id="modal"]/div[2]/div/div[2]/ul
                        /li[4]')
                    l2.click()

                    #get soup
                    second_soup = BeautifulSoup(driver.page_source, 'html.parser')

                    #TABLE OF GENERAL DATA
                    subset = second_soup.find('div', attrs={'class':'site-details'})
                    table2 = subset.find('table', attrs={'class':'table table-striped'})

                    #TABLE OF METADATA
                    table = second_soup.find('table', attrs={'class':'table table-striped
                        location-attributes'})

                    #STAR RATING
                    star_rating = len(second_soup.find_all('i', attrs={'class':'fa rating fa-
                        star'}))

                    #TEXT
                    beschrijving = second_soup.find('p', attrs={'data-property': 'site.description'}).get_text()
                    extra_info = second_soup.find('p', attrs={'data-property': 'site.additional_information'}).get_text()

                    #extracted data
                    list_table = tableDataText2(table2) + tableDataText(table) + [lat, lon, star_rating, beschrijving, extra_info]
                    #print(colored(list_table, 'red'))
                    #print(colored(variable_names, 'red'))
                    #print(len(list_table) == variable_names)

                    #add to dataset and report progress in prompt
                    print(colored(f"Added station: {station_id}", 'green'))
                    print(colored(f"PROGRESS:  {j+1}/{len(all_id)}",'yellow'))

                    #save dataset to csv
                    df.loc[station_id] = list_table

                    #driver quit to reset the state
                    driver.quit()

                    #every 50 stations save
                    if j%50 == 0:
                        df.to_csv(f"NL_WOW_METADATA.csv")
                    j+=1

                    #random behaviour after extraction if needed
                    if random_sec > 0:
                        time.sleep(random.randint(1, random_sec))

            except:
                with open("ids_error.txt", "w") as f: #save ids with error to a file that assembles all error codes
                    f.write(station_id +"\n")


    #save last df with date and time
    t = time.localtime()
    current_time = time.strftime("%Y%m%d_%Hh%Mm", t)

    df.to_csv(f"data_extracted/NL_WOW_METADATA_{current_time}.csv")
    print(colored("DATA SAVED IN NL_WOW_METADATA.csv", "green"))

    return df


#all_id = ['010c7bab-c21f-e911-9462-0003ff5972a6', '81b8a6b0-4110-ed11-b5cf-0003ff5962a8', '44938373-9478-ed11-97b0-0003ff59730c', '12e7004b-1926-ed11-b5cf-0003ff59a71f', '6da1db89-2b5e-ea11-b698-0003ff599880', '9ffd043e-4a57-e711-9400-0003ff596cbf', '924726001']
all_id = np.loadtxt('station_ids.txt', dtype='str').tolist()
get_metadata(all_id) #execute
```

Requirements.txt

```
    alabaster @ file:///home/ktietz/src/ci/alabaster_1611921544520/work
anaconda-client==1.11.0
anaconda-navigator==2.3.1
anaconda-project @ file:///C:/Windows/TEMP/abs_91fu4tfkih/croots/recipe/anaconda-project_1660339890874/work
anyio @ file:///C:/ci/anyio_1644481921011/work/dist
appdirs==1.4.4
argon2-cffi @ file:///opt/conda/conda-bld/argon2-cffi_1645000214183/work
argon2-cffi-bindings @ file:///C:/ci/argon2-cffi-bindings_1644551690056/work
arrow @ file:///opt/conda/conda-bld/arrow_1649166651673/work
astroid @ file:///C:/Windows/TEMP/abs_b0dtxgpicv/croots/recipe/astroid_1659023126745/work
astropy @ file:///C:/ci/astropy_1657719656942/work
async-generator==1.10
atomicwrites==1.4.0
attrs @ file:///opt/conda/conda-bld/attrs_1642510447205/work
Automat @ file:///tmp/build/80754af9/automat_1600298431173/work
autopep8 @ file:///opt/conda/conda-bld/autopep8_1650463822033/work
Babel @ file:///tmp/build/80754af9/babel_1620871417480/work
backcall @ file:///home/ktietz/src/ci/backcall_1611930011877/work
backports.functools-lru-cache @ file:///tmp/build/80754af9/backports.functools_lru_cache_1618170165463/work
backports.tempfile @ file:///home/linux1/recipes/ci/backports.tempfile_1610991236607/work
backports.weakref==1.0.post1
bcrypt @ file:///C:/Windows/Temp/abs_36kl66t_aw/croots/recipe/bcrypt_1659554334050/work
beautifulsoup4 @ file:///C:/ci/beautifulsoup4_1650293025093/work
binaryornot @ file:///tmp/build/80754af9/binaryornot_1617751525010/work
bitarray @ file:///C:/ci/bitarray_1657729621682/work
bkcharts==0.2
black @ file:///C:/ci/black_1660239974023/work
bleach @ file:///opt/conda/conda-bld/bleach_1641577558959/work
bokeh @ file:///C:/Windows/TEMP/abs_4a259bc2-ed05-4a1f-808e-ac712cc0900cddqp8sp7/croots/recipe/bokeh_1658136660686/work
boto3 @ file:///C:/Windows/TEMP/abs_4009c406-44ba-4406-8996-204d9b11202flt4kglbk/croots/recipe/boto3_1657820114895/work
botocore @ file:///C:/ci/botocore_1657735875454/work
Bottleneck @ file:///C:/Windows/Temp/abs_3198ca53-903d-42fd-87b4-03e6d03a8381yfwsuve8/croots/recipe/bottleneck_1657175565403/work
brotlipy==0.7.0
certifi @ file:///C:/b/abs_4f5wo627a3/croots/recipe/certifi_1663615677642/work/certifi
cffi @ file:///C:/Windows/Temp/abs_6808y9x40v/croots/recipe/cffi_1659598653989/work
chardet @ file:///C:/ci/chardet_1607706937985/work
charset-normalizer @ file:///tmp/build/80754af9/charset-normalizer_1630003229654/work
click @ file:///C:/ci/click_1646038595831/work
click-plugins==1.1.1
cligj==0.7.2
cloudpickle @ file:///tmp/build/80754af9/cloudpickle_1632508026186/work
clyent==1.2.2
colorama @ file:///C:/Windows/TEMP/abs_9439aeb1-0254-449a-96f7-33ab5eb17fc8apleb4yn/croots/recipe/colorama_1657009099097/work
colorcet @ file:///C:/ci/colorcet_1651851676912/work
comtypes==1.1.10
conda==23.1.0
conda-build==3.22.0
conda-content-trust @ file:///C:/Windows/TEMP/abs_4589313d-fc62-4ccc-81c0-b801b4449e833j1ajrwu/croots/recipe/conda-content-trust_1658126379362/work
conda-pack @ file:///tmp/build/80754af9/conda-pack_1611163042455/work
conda-package-handling @ file:///C:/b/abs_fcga8w0uem/croot/conda-package-handling_1672865024290/work
conda-repo-cli==1.0.27
conda-token @ file:///Users/paulyim/miniconda3/envs/c3i/conda-bld/conda-token_1662660369760/work
conda-verify==3.4.2
conda_package_streaming @ file:///C:/b/abs_0e5n5hdal3/croot/conda-package-streaming_1670508162902/work
constantly==15.1.0
cookiecutter @ file:///opt/conda/conda-bld/cookiecutter_1649151442564/work
cryptography @ file:///C:/ci/cryptography_1652083563162/work
cssselect==1.1.0
cycler @ file:///tmp/build/80754af9/cycler_1637851556182/work
```

```
Cython @ file:///C:/b/abs_0438epndxm/croots/recipe/cython_1663692771227/work
cytoolz==0.11.0
daal4py==2021.6.0
dask @ file:///C:/ci/dask-core_1658515307198/work
datashader @ file:///C:/Windows/TEMP/abs_62cyd2dpuf/croots/recipe/datashader_1659349034750/work
datashape==0.5.4
debugpy @ file:///C:/ci/debugpy_1637091961445/work
decorator @ file:///opt/conda/conda-bld/decorator_1643638310831/work
defusedxml @ file:///tmp/build/80754af9/defusedxml_1615228127516/work
diff-match-patch @ file:///Users/ktietz/demo/mc3/conda-bld/diff-match-patch_1630511840874/work
dill @ file:///tmp/build/80754af9/dill_1623919422540/work
distributed @ file:///C:/ci/distributed_1658505715201/work
docutils @ file:///C:/Windows/TEMP/abs_24e5e278-4d1c-47eb-97b9-f761d871f482dy2vg450/croots/recipe/docutils_1657175444608/work
entrypoints @ file:///C:/ci/entrypoints_1649926621128/work
et-xmlfile==1.1.0
exceptiongroup==1.1.0
fastjsonschema @ file:///C:/Users/BUILDE~1/AppData/Local/Temp/abs_ebruxzvd08/croots/recipe/python-fastjsonschema_1661376484940/work
filelock @ file:///opt/conda/conda-bld/filelock_1647002191454/work
Fiona==1.9.0
flake8 @ file:///opt/conda/conda-bld/flake8_1648129545443/work
Flask @ file:///home/ktietz/src/ci/flask_1611932660458/work
fonttools==4.25.0
fsspec @ file:///C:/Windows/TEMP/abs_a2mhnomvfy/croots/recipe/fsspec_1659972224540/work
future @ file:///C:/ci/future_1607568713721/work
gensim @ file:///C:/ci/gensim_1646825438310/work
geopandas==0.12.2
glob2 @ file:///home/linux1/recipes/ci/glob2_1610991677669/work
greenlet @ file:///C:/ci/greenlet_1628888275363/work
h11==0.14.0
h5py @ file:///C:/ci/h5py_1659089875384/work
HeapDict @ file:///Users/ktietz/demo/mc3/conda-bld/heapdict_1630598515714/work
holoviews @ file:///C:/Windows/TEMP/abs_fb9d2988-9681-46b6-b9ab-190cfecd15d7uinyq_4z/croots/recipe/holoviews_1658171511842/work
hvplot @ file:///C:/Windows/TEMP/abs_02zoq6lck3/croots/recipe/hvplot_1659026502064/work
hyperlink @ file:///tmp/build/80754af9/hyperlink_1610130746837/work
idna @ file:///tmp/build/80754af9/idna_1637925883363/work
imagecodecs @ file:///C:/b/abs_948ub5byiu/croots/recipe/imagecodecs_1664562381493/work
imageio @ file:///C:/Windows/TEMP/abs_24c1b783-7540-4ca9-a1b1-0e8aa8e6ae64hb79ssux/croots/recipe/imageio_1658785038775/work
imagesize @ file:///C:/Windows/TEMP/abs_3cecd249-3fc4-4bfc-b80b-bb227b0d701en12vqzot/croots/recipe/imagesize_1657179501304/work
importlib-metadata @ file:///C:/ci/importlib-metadata_1648562621412/work
incremental @ file:///tmp/build/80754af9/incremental_1636629750599/work
inflection==0.5.1
iniconfig @ file:///home/linux1/recipes/ci/iniconfig_1610983019677/work
intake @ file:///opt/conda/conda-bld/intake_1647436631684/work
intervaltree @ file:///Users/ktietz/demo/mc3/conda-bld/intervaltree_1630511889664/work
ipykernel @ file:///C:/b/abs_21ykzkm7y_/croots/recipe/ipykernel_1662361803478/work
ipython @ file:///C:/Windows/TEMP/abs_45b5zb1l7q/croots/recipe/ipython_1659529855872/work
ipython-genutils @ file:///tmp/build/80754af9/ipython_genutils_1606773439826/work
ipywidgets @ file:///tmp/build/80754af9/ipywidgets_1634143127070/work
isort @ file:///tmp/build/80754af9/isort_1628603791788/work
itemadapter @ file:///tmp/build/80754af9/itemadapter_1626442940632/work
itemloaders @ file:///opt/conda/conda-bld/itemloaders_1646805235997/work
itsdangerous @ file:///tmp/build/80754af9/itsdangerous_1621432558163/work
jdcal @ file:///Users/ktietz/demo/mc3/conda-bld/jdcal_1630584345063/work
jedi @ file:///C:/ci/jedi_1644315428289/work
jellyfish @ file:///C:/ci/jellyfish_1647962783748/work
Jinja2 @ file:///tmp/build/80754af9/jinja2_1612213139570/work
jinja2-time @ file:///opt/conda/conda-bld/jinja2-time_1649251842261/work
jmespath @ file:///Users/ktietz/demo/mc3/conda-bld/jmespath_1630583964805/work
joblib @ file:///tmp/build/80754af9/joblib_1635411271373/work
json5 @ file:///tmp/build/80754af9/json5_1624432770122/work
jsonschema @ file:///C:/b/abs_59eyhnbyej/croots/recipe/jsonschema_1663375476535/work
jupyter @ file:///C:/Windows/TEMP/abs_56xfdi__li/croots/recipe/jupyter_1659349053177/work
jupyter-console @ file:///opt/conda/conda-bld/jupyter_console_1647002188872/work
jupyter-server @ file:///C:/Windows/TEMP/abs_d3c42c59-765d-4f9b-9fa3-ad5b1369485611i_yual/
```

```
    croots/recipe/jupyter_server_1658754493238/work
jupyter_client @ file:///C:/ci/jupyter_client_1661836943389/work
jupyter_core @ file:///C:/b/abs_a9330r1z_i/croots/recipe/jupyter_core_1664917313457/work
jupyterlab @ file:///C:/ci/jupyterlab_1658891142428/work
jupyterlab-pygments @ file:///tmp/build/80754af9/jupyterlab_pygments_1601490720602/work
jupyterlab-server @ file:///opt/conda/conda-bld/jupyterlab_server_1644500396812/work
jupyterlab-widgets @ file:///tmp/build/80754af9/jupyterlab_widgets_1609884341231/work
keyring @ file:///C:/ci/keyring_1638531673471/work
kiwisolver @ file:///C:/ci/kiwisolver_1653292407425/work
lazy-object-proxy @ file:///C:/ci/lazy-object-proxy_1616529288960/work
libarchive-c @ file:///tmp/build/80754af9/python-libarchive-c_1617780486945/work
llvmlite==0.38.0
locket @ file:///C:/ci/locket_1652904031364/work
lxml @ file:///C:/ci/lxml_1657527445690/work
lz4 @ file:///C:/ci/lz4_1619516674350/work
Markdown @ file:///C:/ci/markdown_1614364082838/work
MarkupSafe @ file:///C:/ci/markupsafe_1621528502553/work
matplotlib @ file:///C:/ci/matplotlib-suite_1660169687702/work
matplotlib-inline @ file:///C:/ci/matplotlib-inline_1661915841596/work
mccabe==0.6.1
menuinst @ file:///C:/Users/BUILDE~1/AppData/Local/Temp/abs_455sf5o0ct/croots/recipe/
    menuinst_1661805970842/work
mistune @ file:///C:/ci/mistune_1607359457024/work
mkl-fft==1.3.1
mkl-random @ file:///C:/ci/mkl_random_1626186184308/work
mkl-service==2.4.0
mock @ file:///tmp/build/80754af9/mock_1607622725907/work
mpmath==1.2.1
msgpack @ file:///C:/ci/msgpack-python_1652329316214/work
multipledispatch @ file:///C:/ci/multipledispatch_1607574329826/work
munch==2.5.0
munkres==1.1.4
mypy-extensions==0.4.3
navigator-updater==0.3.0
nbclassic @ file:///opt/conda/conda-bld/nbclassic_1644943264176/work
nbclient @ file:///C:/ci/nbclient_1650290387259/work
nbconvert @ file:///C:/ci/nbconvert_1649741016669/work
nbformat @ file:///C:/b/abs_1dw90o2uqb/croots/recipe/nbformat_1663744957967/work
nest-asyncio @ file:///C:/ci/nest-asyncio_1649829929390/work
networkx @ file:///C:/ci/networkx_1657716998256/work
nltk @ file:///opt/conda/conda-bld/nltk_1645628263994/work
nose @ file:///opt/conda/conda-bld/nose_1642704612149/work
notebook @ file:///C:/Windows/TEMP/abs_79abr1_60s/croots/recipe/notebook_1659083661851/work
numba @ file:///C:/ci/numba_1650394399948/work
numexpr @ file:///C:/Windows/Temp/abs_e2036a32-9fe9-47f3-a04c-dbb1c232ba4b334exiur/croots/
    recipe/numexpr_1656940304835/work
numpy @ file:///C:/ci/numpy_and_numpy_base_1653574844560/work
numpydoc @ file:///C:/Windows/TEMP/abs_30799058-86dd-4401-b621-d172137a4d87_ra3twm7/croots/
    recipe/numpydoc_1657529873713/work
olefile @ file:///Users/ktietz/demo/mc3/conda-bld/olefile_1629805411829/work
opencv-python==4.7.0.72
openpyxl==3.0.10
outcome==1.2.0
packaging @ file:///tmp/build/80754af9/packaging_1637314298585/work
pandas @ file:///C:/b/abs_cdcgk91igc/croots/recipe/pandas_1663772960432/work
pandocfilters @ file:///opt/conda/conda-bld/pandocfilters_1643405455980/work
panel @ file:///C:/ci/panel_1657899702145/work
param @ file:///tmp/build/80754af9/param_1636647414893/work
paramiko @ file:///opt/conda/conda-bld/paramiko_1640109032755/work
parsel @ file:///C:/ci/parsel_1646740216444/work
parso @ file:///opt/conda/conda-bld/parso_1641458642106/work
partd @ file:///opt/conda/conda-bld/partd_1647245470509/work
pathlib @ file:///Users/ktietz/demo/mc3/conda-bld/pathlib_1629713961906/work
pathspec @ file:///C:/Windows/TEMP/abs_581d0u45mh/croots/recipe/pathspec_1659627132171/work
patsy==0.5.2
pep8==1.7.1
pexpect @ file:///tmp/build/80754af9/pexpect_1605563209008/work
pickleshare @ file:///tmp/build/80754af9/pickleshare_1606932040724/work
Pillow==9.2.0
pkginfo @ file:///tmp/build/80754af9/pkginfo_1643162084911/work
platformdirs @ file:///C:/b/abs_73cc5cz_1u/croots/recipe/platformdirs_1662711386458/work
```

```
plotly @ file:///C:/ci/plotly_1658142442431/work
pluggy @ file:///C:/ci/pluggy_1648024580010/work
poyo @ file:///tmp/build/80754af9/poyo_1617751526755/work
prometheus-client @ file:///C:/Windows/TEMP/abs_ab9nx8qb08/croots/recipe/prometheus_client_1659455104602/work
prompt-toolkit @ file:///tmp/build/80754af9/prompt-toolkit_1633440160888/work
Protego @ file:///tmp/build/80754af9/protego_1598657180827/work
psutil @ file:///C:/Windows/Temp/abs_b2c2fd7f-9fd5-4756-95ea-8aed74d0039flsd9qufz/croots/recipe/psutil_1656431277748/work
ptyprocess @ file:///tmp/build/80754af9/ptyprocess_1609355006118/work/dist/ptyprocess-0.7.0-py2.py3-none-any.whl
py @ file:///opt/conda/conda-bld/py_1644396412707/work
pyasn1 @ file:///Users/ktietz/demo/mc3/conda-bld/pyasn1_1629708007385/work
pyasn1-modules==0.2.8
pycodestyle @ file:///tmp/build/80754af9/pycodestyle_1636635402688/work
pycosat==0.6.3
pycparser @ file:///tmp/build/80754af9/pycparser_1636541352034/work
pyct @ file:///C:/ci/pyct_1658488033428/work
pycurl==7.45.1
PyDispatcher==2.0.5
pydocstyle @ file:///tmp/build/80754af9/pydocstyle_1621600989141/work
pyerfa @ file:///C:/ci/pyerfa_1621560974055/work
pyflakes @ file:///tmp/build/80754af9/pyflakes_1636644436481/work
Pygments @ file:///opt/conda/conda-bld/pygments_1644249106324/work
PyHamcrest @ file:///tmp/build/80754af9/pyhamcrest_1615748656804/work
PyJWT @ file:///C:/ci/pyjwt_1657511236979/work
pylint @ file:///C:/Windows/TEMP/abs_518eqlbmoo/croots/recipe/pylint_1659110354241/work
pyls-spyder==0.4.0
PyNaCl @ file:///C:/Windows/Temp/abs_d5c3ajcm87/croots/recipe/pynacl_1659620667490/work
pyodbc @ file:///C:/Windows/Temp/abs_61e3jz3u05/croots/recipe/pyodbc_1659513801402/work
pyOpenSSL @ file:///opt/conda/conda-bld/pyopenssl_1643788558760/work
pyparsing @ file:///C:/Users/BUILDE~1/AppData/Local/Temp/abs_7f_7lba6rl/croots/recipe/pyparsing_1661452540662/work
pyproj==3.4.1
pyrsistent @ file:///C:/ci/pyrsistent_1636093225342/work
PySocks @ file:///C:/ci/pysocks_1605307512533/work
pytest==7.1.2
python-dateutil @ file:///tmp/build/80754af9/python-dateutil_1626374649649/work
python-dotenv==1.0.0
python-lsp-black @ file:///tmp/build/80754af9/python-lsp-black_1634232156041/work
python-lsp-jsonrpc==1.0.0
python-lsp-server @ file:///tmp/build/80754af9/python-lsp-server_1648176833691/work
python-slugify @ file:///tmp/build/80754af9/python-slugify_1620405669636/work
python-snappy @ file:///C:/ci/python-snappy_1610133405910/work
pytz @ file:///C:/Windows/TEMP/abs_90eacd4e-8eff-491e-b26e-f707eba2cbe1ujvbhqz1/croots/recipe/pytz_1654762631027/work
pyviz-comms @ file:///tmp/build/80754af9/pyviz_comms_1623747165329/work
PyWavelets @ file:///C:/ci/pywavelets_1648728084106/work
pywin32==302
pywin32-ctypes @ file:///C:/ci/pywin32-ctypes_1607553594546/work
pywinpty @ file:///C:/ci_310/pywinpty_1644230983541/work/target/wheels/pywinpty-2.0.2-cp39-none-win_amd64.whl
PyYAML==6.0
pyzmq @ file:///C:/ci/pyzmq_1657615952984/work
QDarkStyle @ file:///tmp/build/80754af9/qdarkstyle_1617386714626/work
qstylizer @ file:///tmp/build/80754af9/qstylizer_1617713584600/work/dist/qstylizer-0.1.10-py2.py3-none-any.whl
QtAwesome @ file:///tmp/build/80754af9/qtawesome_1637160816833/work
qtconsole @ file:///opt/conda/conda-bld/qtconsole_1643819126524/work
QtPy @ file:///C:/ci/qtpy_1662015096047/work
queuelib==1.5.0
regex @ file:///C:/ci/regex_1658258307256/work
requests @ file:///C:/ci/requests_1657735342357/work
requests-file @ file:///Users/ktietz/demo/mc3/conda-bld/requests-file_1629455781986/work
rope @ file:///opt/conda/conda-bld/rope_1643788605236/work
Rtree @ file:///C:/ci/rtree_1618421015405/work
ruamel-yaml-conda @ file:///C:/ci/ruamel_yaml_1616016898638/work
ruamel.yaml @ file:///C:/b/abs_30ee5qbthd/croot/ruamel.yaml_1666304562000/work
ruamel.yaml.clib @ file:///C:/b/abs_aarblxbilo/croot/ruamel.yaml.clib_1666302270884/work
s3transfer @ file:///C:/ci/s3transfer_1654512518418/work
scikit-image @ file:///C:/ci/scikit-image_1648214340990/work
```

```
scikit-learn @ file:///C:/ci/scikit-learn_1642617276183/work
scikit-learn-intelex==2021.20221004.171935
scipy==1.9.1
Scrapy @ file:///C:/Windows/TEMP/abs_f50e21j997/croots/recipe/scrapy_1659598707153/work
seaborn @ file:///tmp/build/80754af9/seaborn_1629307859561/work
selenium==4.8.2
Send2Trash @ file:///tmp/build/80754af9/send2trash_1632406701022/work
service-identity @ file:///Users/ktietz/demo/mc3/conda-bld/service_identity_1629460757137/work
shapely==2.0.1
sip==4.19.13
six @ file:///tmp/build/80754af9/six_1644875935023/work
smart-open @ file:///C:/ci/smart_open_1651235069716/work
sniffio @ file:///C:/ci/sniffio_1614030527509/work
snowballstemmer @ file:///tmp/build/80754af9/snowballstemmer_1637937080595/work
sortedcollections @ file:///tmp/build/80754af9/sortedcollections_1611172717284/work
sortedcontainers @ file:///tmp/build/80754af9/sortedcontainers_1623949099177/work
soupsieve @ file:///tmp/build/80754af9/soupsieve_1636706018808/work
Sphinx @ file:///C:/ci/sphinx_1657617205740/work
sphinxcontrib-applehelp @ file:///home/ktietz/src/ci/sphinxcontrib-applehelp_1611920841464/work
sphinxcontrib-devhelp @ file:///home/ktietz/src/ci/sphinxcontrib-devhelp_1611920923094/work
sphinxcontrib-htmlhelp @ file:///tmp/build/80754af9/sphinxcontrib-htmlhelp_1623945626792/work
sphinxcontrib-jsmath @ file:///home/ktietz/src/ci/sphinxcontrib-jsmath_1611920942228/work
sphinxcontrib-qthelp @ file:///home/ktietz/src/ci/sphinxcontrib-qthelp_1611921055322/work
sphinxcontrib-serializinghtml @ file:///tmp/build/80754af9/sphinxcontrib-serializinghtml_1624451540180/work
spyder @ file:///C:/Windows/TEMP/abs_66k5aq_mkw/croots/recipe/spyder_1659599805372/work
spyder-kernels @ file:///C:/ci/spyder-kernels_1647011408404/work
SQLAlchemy @ file:///C:/Windows/Temp/abs_f8661157-660b-49bb-a790-69ab9f3b8f7c8a8s2psb/croots/recipe/sqlalchemy_1657867864564/work
statsmodels==0.13.2
stop-words==2018.7.23
sympy @ file:///C:/ci/sympy_1647853873858/work
tables==3.6.1
tabulate @ file:///C:/ci/tabulate_1657619055201/work
TBB==0.2
tblib @ file:///Users/ktietz/demo/mc3/conda-bld/tblib_1629402031467/work
tenacity @ file:///C:/Windows/TEMP/abs_980d07a6-8e21-4174-9c17-7296219678ads7dhdov_/croots/recipe/tenacity_1657899108023/work
termcolor==2.2.0
terminado @ file:///C:/ci/terminado_1644322780199/work
testpath @ file:///C:/Windows/TEMP/abs_23c7fa33-cbb9-46dc-b7c5-590c38e2de3d4bmbngal/croots/recipe/testpath_1655908553202/work
text-unidecode @ file:///Users/ktietz/demo/mc3/conda-bld/text-unidecode_1629401354553/work
textdistance @ file:///tmp/build/80754af9/textdistance_1612461398012/work
threadpoolctl @ file:///Users/ktietz/demo/mc3/conda-bld/threadpoolctl_1629802263681/work
three-merge @ file:///tmp/build/80754af9/three-merge_1607553261110/work
tifffile @ file:///tmp/build/80754af9/tifffile_1627275862826/work
tinycss @ file:///tmp/build/80754af9/tinycss_1617713798712/work
tldextract @ file:///opt/conda/conda-bld/tldextract_1646638314385/work
toml @ file:///tmp/build/80754af9/toml_1616166611790/work
tomli @ file:///C:/Windows/TEMP/abs_ac109f85-a7b3-4b4d-bcfd-52622eceddf0hy332ojo/croots/recipe/tomli_1657175513137/work
tomlkit @ file:///C:/Windows/TEMP/abs_3296qo9v6b/croots/recipe/tomlkit_1658946894808/work
toolz @ file:///tmp/build/80754af9/toolz_1636545406491/work
tornado @ file:///C:/ci/tornado_1606924294691/work
tqdm @ file:///C:/b/abs_0axbz66qik/croots/recipe/tqdm_1664392691071/work
traitlets @ file:///tmp/build/80754af9/traitlets_1636710298902/work
trio==0.22.0
trio-websocket==0.9.2
Twisted @ file:///C:/Windows/Temp/abs_ccblv2rzfa/croots/recipe/twisted_1659592764512/work
twisted-iocpsupport @ file:///C:/ci/twisted-iocpsupport_1646798932792/work
typing_extensions @ file:///C:/Windows/TEMP/abs_dd2d0moa85/croots/recipe/typing_extensions_1659638831135/work
ujson @ file:///C:/ci/ujson_1657525944442/work
Unidecode @ file:///tmp/build/80754af9/unidecode_1614712377438/work
urllib3 @ file:///C:/Windows/TEMP/abs_65ynz4fdmi/croots/recipe/urllib3_1659110473919/work
w3lib @ file:///Users/ktietz/demo/mc3/conda-bld/w3lib_1629359764703/work
watchdog @ file:///C:/ci/watchdog_1638367441841/work
wcwidth @ file:///Users/ktietz/demo/mc3/conda-bld/wcwidth_1629357192024/work
```

```
webdriver-manager==3.8.5
webencodings==0.5.1
websocket-client @ file:///C:/ci/websocket-client_1614804375980/work
Werkzeug @ file:///opt/conda/conda-bld/werkzeug_1645628268370/work
widgetsnbextension @ file:///C:/ci/widgetsnbextension_1644991377168/work
win-inet-pton @ file:///C:/ci/win_inet_pton_1605306162074/work
win-unicode-console==0.5
wincertstore==0.2
wordcloud==1.8.2.2
wrapt @ file:///C:/Windows/Temp/abs_7c3dd407-1390-477a-b542-fd15df6a24085_diwiza/croots/
    recipe/wrapt_1657814452175/work
wsproto==1.2.0
xarray @ file:///opt/conda/conda-bld/xarray_1639166117697/work
xlrd @ file:///tmp/build/80754af9/xlrd_1608072521494/work
XlsxWriter @ file:///opt/conda/conda-bld/xlsxwriter_1649073856329/work
xlwings @ file:///C:/b/abs_41rmpiyy58/croots/recipe/xlwings_1664932818417/work
yapf @ file:///tmp/build/80754af9/yapf_1615749224965/work
zict==2.1.0
zipp @ file:///C:/ci/zipp_1652273994994/work
zope.interface @ file:///C:/ci/zope.interface_1625036252485/work
zstandard==0.18.0
```

B Analysis Source Code

Analysis of the PWS Network in the Netherlands

```python
# %%
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import plotly.express as px

# %%
station = np.loadtxt('official_station.txt', delimiter="    ", dtype='str')

# %%
df = pd.read_csv('NL_WOW_METADATA.csv')#('backup_extraction.csv')

# %%
replace_dict = {'Zeer open':'Very open', 'Open':'Open', 'Standaard':'Standard', 'Beperkt':'Restricted', 'Beschut':'Sheltered', 'Zeer Beschut':'Very Sheltered', 'Rooftop':'Rooftop', "Traffic":"Traffic", 'Unknown':"Unknown"}#{'5':'Very open', '4':'Open', '3':'Standard', '2':'Restricted', '1':'Sheltered', '0':'Very Sheltered' , 'R':'Rooftop', "T":"Traffic", 'U':"Unknown"}
replace_dict_before = {"1":'U'}

df = df.replace({"Meting luchttemperatuur": replace_dict_before, "Meting neerslag": replace_dict_before, "Meting wind":replace_dict_before, "Stedelijke zone": replace_dict_before, "Waarneemuren":replace_dict_before, "Ligging":replace_dict_before})
df = df.replace({"Ligging": replace_dict})

# %%
def pie_plot(df, column, name=''):
    sizes = df[column].value_counts().sort_index() / df[column].value_counts().sum() * 100

    colours = {'1': 'C0',
           '2': 'C1',
           '3': 'C2',
           '4': 'C3',
           '5': 'C4',
           '6': 'C5',
           '7': 'C6',
           'U': 'C7',
           'A': 'C8',
           'B': 'C9',
           'C': 'C10',
           'D': 'C11',
           '0': 'C12',
           'Unknown': 'C7'}

    labels = sorted(df[column].unique())

    #print(sorted(labels), [colours[key] for key in labels])

    plt.pie(sizes,
            autopct='%1.1f%%', startangle=140, pctdistance=1.1, labeldistance=1.2,
            colors=[colours[key] for key in labels])
    plt.axis('equal')
    plt.legend(sizes.index, loc="best", title='Category')

    if name != '':
        plt.title(name)
    else:
        plt.title(column)

    plt.tight_layout()
    plt.show()
    return None

# %%
def pie_plot2(df, column, name=''):
    sizes = df[column].value_counts().sort_index() / df[column].value_counts().sum() * 100

    plt.pie(sizes,
            autopct='%1.1f%%', startangle=140, pctdistance=1.1, labeldistance=1.2)
    plt.axis('equal')
    plt.legend(sizes.index, loc="best", title='Category')

    if name != '':
        plt.title(name)
    else:
        plt.title(column)
    plt.tight_layout()
    plt.show()
    return None

# %%
metadata_columns = df.columns[11:-5]

# %%
lowerdf = df[df["star_rating"] <=  2]
clean_lowerdf = lowerdf[metadata_columns].replace("Unknown", 'U').melt()

# %%
pie_plot2(df, 'Ligging', 'Location')

# %%
pie_plot(df, 'Waarneemuren', 'Observation Time')

# %%
pie_plot(df, 'Stedelijke zone', 'Urban area')
pie_plot(df, 'Meting luchttemperatuur', 'Air temperature')
pie_plot(df, 'Meting neerslag', 'Precipitation')
pie_plot(df, 'Meting wind', 'Wind')
pie_plot2(df, 'Actief station', 'Active station')
pie_plot2(df, 'Data downloaden toegestaan?', 'Downloadable')
pie_plot2(df, 'Officieel station', 'Official station')
pie_plot2(df, 'star_rating', 'Star Rating')
pie_plot2(df, 'Motivatie station', 'Motivation station')

# %% [markdown]
# # Wordcloud

# %%
from wordcloud import WordCloud
from wordcloud import ImageColorGenerator
from wordcloud import STOPWORDS

from stop_words import get_stop_words
stop_words = get_stop_words('dutch')

# %%
text = " ".join(i for i in df.description.dropna())
stopwords = list(STOPWORDS) + stop_words #dutch and english
wordcloud = WordCloud(stopwords=stopwords, background_color="white",width=800, height=400).
    generate(text)
```

```
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()

# %%
#without obvious things
text = text.replace('weather', '')
text = text.replace('station', '')
text = text.replace('weatherstation', '')
text = text.replace('weerstation', '')
text = text.replace('Netherlands', '')
text = text.replace('Weatherstation', '')
text = text.replace('Weer', '')
text = text.replace('Weather', '')
text = text.replace('weer', '')
text = text.replace('sensor', '')
text = text.replace('data', '')
text = text.replace('meter', '')


wordcloud2 = WordCloud(stopwords=stopwords, background_color="white",width=800, height=400).
    generate(text)
plt.figure( figsize=(30,20))
plt.imshow(wordcloud2, interpolation='bilinear')
plt.axis("off")
plt.show()

# %%
import seaborn as sns

# %%
df_prof = pd.read_csv("DIST_PROF.csv")
df_amat = pd.read_csv("DIST_AMATEUR.csv")

# %%
sns.distplot(df_prof['HubDist'])
plt.grid()
plt.title("")
plt.xlim([0, df_prof['HubDist'].max()])
plt.xlabel("Distance to Nearest Proffesional Station in Meters (m)")
plt.suptitle("Histrogram and KDE of Distance to Nearest Proffesional Station for PWS in The
    Netherlands")

# %%
sns.distplot(df_amat['MIN'])
plt.grid()
plt.xlim([0,df_amat['MIN'].max()])
plt.xlabel("Distance to Nearest Amateur Station in Meters (m)")
plt.suptitle("Histrogram and KDE of Distance to Nearest Neighbor Station for PWS in The
    Netherlands")

# %% [markdown]
# # Land Cover Analysis

# %%
lgn = pd.read_csv('LGN_PWS.csv')

# %%
replace_dict2 = {1:'1 - agrarisch gras', 2:'2 - maïs', 3:'3 - aardappelen', 4:'4 - bieten',
    5:'5 - granen', 9:'9 - boomgaarden', 11:'11 - loofbos', 16:'16 - zoet water', 18:'18 -
    bebouwing in primair bebouwd gebied', 19:'19 - bebouweing in secundair bebouwd gebied',
    20:'20 - bos in primair bebouwd gebied', 22:'22 - bos in secundair bebouwd gebied', 23:'
    23 - gras in primair bebouwd gebied', 26:'26 - bebouwing in buitengebied', 27:'27 -
    overig grondgebruik in buitengebied', 28:'28 - gras in secundair bebouwd gebied', 31:'31
    - open zand in kustgebied', 45:'45 - natuurlijk beheerde agrarische graslanden', 61:'61 -
     boomkwkerijen', 62:'62 - fruitkwekerijen', 251:'251 - hoofdinfrastructuur en
    spoorbaanlichamen', 252:'252 - halfverharde wegen, infrastructuur langzaam verkeer en
    overige infrastructuur'}

colours2 = {1: '#73df1f',
        2: '#e89919',
        3: '#b26600',
        4: '#e51f7f',
        5: '#ffff00',
        9: '#3cef45',
        11: '#33c800',
        16: '#2473ff',
        18: '#ff0000',
        19: '#730000',
        20: '#93d600',
        22: '#93aa00',
        23: '#93ff00',
        26: '#761818',
        27: '#ff645a',
        28: '#a8ef44',
        31: '#e6fb00',
        45: '#b6b639',
        61: '#ffb3a8',
        62: '#e3ff70',
        251: '#871b00',
        252: '#b02300'}

colours = {}
for i in colours2:
    colours[replace_dict2[i]] = colours2[i]

# %%
lgn = lgn.replace({'LGN_1':replace_dict2})

# %%
sizes = lgn['LGN_1'].value_counts().sort_index() / lgn['LGN_1'].value_counts().sum() * 100
labels = sorted(lgn['LGN_1'].unique())

# %%
import re

# %%
new_labels = []
for i in range(len(labels)):
    new_labels.append(labels[i] + rf" $\bf{({round(sizes.values[i],2)})}$" +'%')

sorted_labels = sorted(new_labels, key=lambda s: int(re.search(r'\d+', s).group()))

# %% [markdown]
# # Land Cover Analysis

# %%
test = pd.DataFrame(sizes)

# %%
labels_sort = sorted(list(replace_dict2.values()), key=lambda s: int(re.search(r'\d+', s).
    group()))

# %%
sizes_sorted = test.loc[labels_sort]

# %%
sizes_sorted["LGN_1"]

# %%
# plt.pie(test,
#         colors=[colours[key] for key in [i.split('$')[0][:-1] for i in labels]])#, labels =
     [i.split(' ')[0] for i in labels])
# plt.legend(sorted_labels, loc="center left", bbox_to_anchor=(1.1, 0.5), title='Category',
    fontsize='10')
# plt.title('Landcover Types of the WOW-NL Weather Stations')
# plt.tight_layout()
# plt.show()

# %%
colors = {}
```

```python
for i in range(len(colours)):
    colors[sorted_labels[i]] = colours[list(colours.keys())[i]]

# %%
sns.set_style("whitegrid")

# %%
plt.bar([i.split(' ')[0] for i in sorted_labels], sizes_sorted["LGN_1"], color=[colours[key]
    for key in [i.split('$')[0][:] for i in labels]])

plt.title("Landcover of PWS stations in the Netherlands")
plt.xlabel("LGN - Landcover Categories")
plt.ylabel("Percentage of PWS Stations (%)")
plt.xticks(rotation=90)

#plt.legend(sorted_labels, loc="center left", bbox_to_anchor=(1.1, 0.5), title='Category',
    fontsize='10')
labels = list(colors.keys())
handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for label in labels]
plt.legend(handles, labels, loc="center left", bbox_to_anchor=(1.1, 0.5), title='Category',
    fontsize='10')


plt.show()

# %% [markdown]
# # Device Setup

# %%
def pie_plot(df, column):
    sizes = df[column].value_counts().sort_index() / df[column].value_counts().sum() * 100

    colours = {'1': 'C0',
           '2': 'C1',
           '3': 'C2',
           '4': 'C3',
           '5': 'C4',
           '6': 'C5',
           '7': 'C6',
           'U': 'C7',
           'A': 'C8',
           'B': 'C9',
           'C': 'C10',
           'D': 'C11',
           '0': 'C12'}

    labels = sorted(df[column].unique())

    #print(sorted(labels), [colours[key] for key in labels])

    plt.pie(sizes,
            autopct='%1.1f%%', startangle=140, pctdistance=1.1, labeldistance=1.2,
            colors=[colours[key] for key in labels])
    plt.axis('equal')
    plt.legend(sizes.index, loc="best", title='Category')
    plt.title(column)
    plt.tight_layout()
    plt.show()
    return None

# %%
df['radiationscreen'] = (df['Meting luchttemperatuur'] == 'A') | (df['Meting luchttemperatuur
    '] == 'B') | (df['Meting luchttemperatuur'] == 'C')
df['radiationscreen'][df['Meting luchttemperatuur'] == '0'] = 'None'


# %%
plt.pie(df['radiationscreen'].value_counts(), labels=['No', 'Yes', 'No Data'], autopct='%.2f
    %%')
plt.title('PWS in the Netherlands with a Radiation Screen')
plt.show()
```

```

# %%
df['precipitation_cal'] = (df['Meting neerslag'] == 'A') | (df['Meting neerslag'] == 'B') | (
    df['Meting neerslag'] == 'C')
df['precipitation_cal'][df['Meting neerslag'] == '0'] = 'None'

plt.pie(df['precipitation_cal'].value_counts(), labels=['No', 'Yes', 'No Data'], autopct='%.2
    f%%')
plt.title("Standard rain gauge or calibrated rain gauge with tipping bucket")
plt.show()

# %%
df.columns

# %%
df['wind_cal'] = (df['Meting wind'] == 'A') | (df['Meting wind'] == 'B')
df['wind_cal'][df['Meting wind'] == '0'] = 'None'

plt.pie(df['wind_cal'].value_counts(), labels=['No', 'Yes', 'No Data'],autopct='%.2f%%')
plt.title("Wind Sensor Above the Ground")
plt.show()

# %% [markdown]
# ## Credibility

# %%
text = " ".join(i for i in df.Website.dropna())
text = text.replace('.nl', '')
text = text.replace('.net', '')
text = text.replace('https://', '')
text = text.replace('www.', '')
text = text.replace('html', '')
text = text.replace('https', '')
text = text.replace('.com', '')
text = text.replace('.org', '')
text = text.replace('.eu', '')
text = text.replace('index', '')
text = text.replace('htm', '')


stopwords = list(STOPWORDS) + stop_words #dutch and english
wordcloud = WordCloud(stopwords=stopwords, background_color="white",width=800, height=400).
    generate(text)
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
#plt.title('Website of PWS Owners in the Netherlands')
plt.show()

# %%
text = " ".join(i for i in df.Organisatie.dropna())

stopwords = list(STOPWORDS) + stop_words #dutch and english
wordcloud = WordCloud(stopwords=stopwords, background_color="white",width=800, height=400).
    generate(text)
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
#plt.title('Website of PWS Owners in the Netherlands')
plt.show()

# %%
metadata_columns = [
 'Ligging',
 'Meting luchttemperatuur',
 'Meting neerslag',
 'Meting wind',
 'Stedelijke zone',
 'Waarneemuren',
 'description',
 'extra_info',
]

# %%
check_empty = pd.concat([df[metadata_columns].iloc[:,:6] == "U", df[metadata_columns].iloc
    [:,-2:].isna()],axis=1)
amount_empty = check_empty.sum(axis=1)

# %%
plt.pie(sorted(amount_empty.value_counts()), labels=sorted(amount_empty.value_counts().index)
    ,autopct='%.2f%%')
plt.title("Amount of missing Meta Data Entries")
plt.show()
```

C
Data Plots

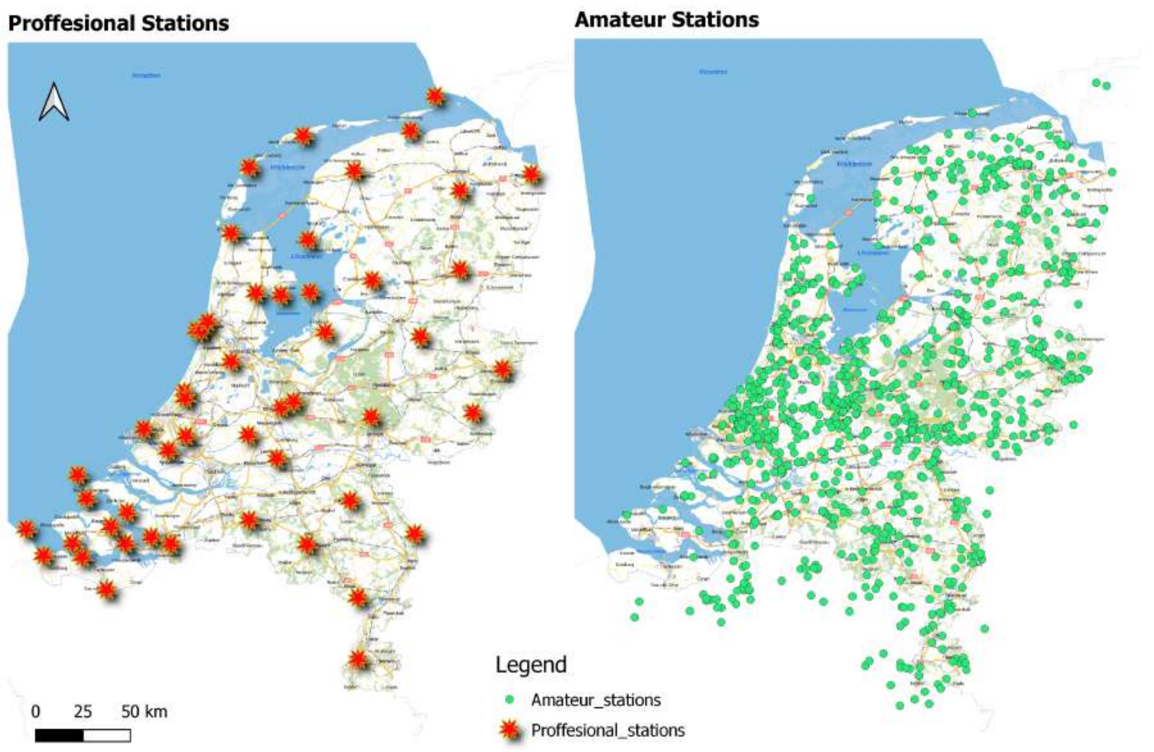


Figure C.1: Weather Stations in the Netherlands

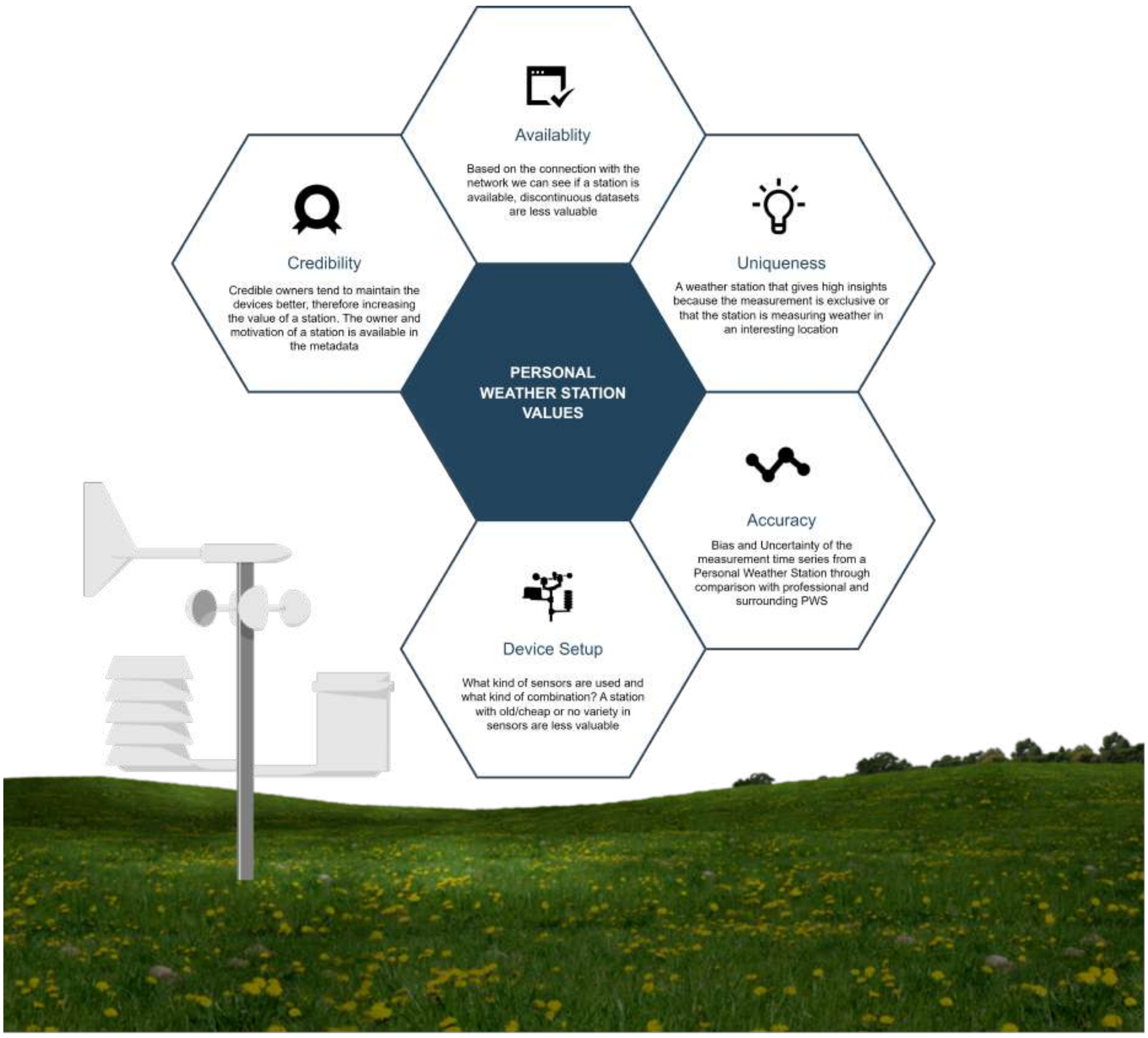

Availablity
Based on the connection with the network we can see if a station is available, discontinuous datasets are less valuable
Credibility
Credible owners tend to maintain the devices better, therefore increasing the value of a station. The owner and motivation of a station is available in the metadata
Uniqueness
A weather station that gives high insights because the measurement is exclusive or that the station is measuring weather in an interesting location
PERSONAL WEATHER STATION VALUES
Accuracy
Bias and Uncertainty of the measurement time series from a Personal Weather Station through comparison with professional and surrounding PWS
Device Setup
What kind of sensors are used and what kind of combination? A station with old/cheap or no variety in sensors are less valuable

Figure C.2: Value Framework for a Personal Weather Station