Leveraging Data for Green Infrastructure Performance Analysis and Prediction

By

Andrew Kurzweil

Thesis

Submitted to Department of Civil and Environmental Engineering

College of Engineering

Villanova University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE, CIVIL ENGINEERING

May 2021

VILLANOVA
UNIVERSITY
College of Engineering

Andrew Kurzweil

Leveraging Data for Green Infrastructure Performance Analysis and Prediction

May 2021

Advisors:

Robert G. Traver, Ph.D., P.E., D. WRE, F.EWRI, F.ASCE ,

Bridget Wadzuk, Ph.D., and

Gerald Zaremba, Ph.D.

Villanova University

College of Engineering

Center for Resilient Water Systems

Department of Civil and Environmental Engineering

800 East Lancaster Avenue

Villanova, Pennsylvania 19085

The data presented herein were collected as part of a research initiative by Villanova University's Center for Resilient Water Systems (VCRWS) under a grant funded by the Pennsylvania Department of Transportation (PennDOT). These data are to be considered preliminary, and may not be used elsewhere except with explicit written permission from PennDOT or VCRWS. PennDOT and VCRWS make no guarantees as to the integrity of these data, which are provided for informational purposes only and are subject to change upon further review.

The opinions presented in this publication are those of the author and do not necessarily express the opinions of the Pennsylvania Department of Transportation or AECOM. Reference in this report to any commercial product, process, or service, or the use of any trade, firm, or corporation name is for general informational purposes only and does not constitute an endorsement or certification of any kind by the author. This project is a research initiative of the Villanova Center for Resilient Water Systems.

Abstract

The need for reliable, transferrable, and broadly accepted performance metrics for green stormwater infrastructure (GSI) is widely discussed across the industry. The present research is a case study of instrumentation practices, data storage, and a proposal for two site-agnostic performance indicators because they do not account for site parameters. FAIR (Findable, Accessible, Interoperable, Reusable) principles for data storage and access ensure that historical data and future data are collected and organized in a manner that allows for continued study while maintaining flexibility and controlled vocabulary for robust data descriptions. Recession rate and infiltration rate are widely understood to be good indicators of GSI performance, but are difficult to test, rely on manual labor by onsite personnel, and test only discrete points within a site. Continuous monitoring data can be used to approximate a site's average recession and infiltration rates using just a few sensors. Analyzing trends in the average of these rates on a storm-event basis gives insight into performance trends. A relationship between recession rate and temperature was found to exist, with a correlation coefficient of 0.735, which has been found in previous studies. At the case study site, recession rates between 40 and 120 mm/hr can be expected, depending on time of year and water temperature, and average infiltration drying rates as high as 9.5 cm/hr can be expected, although further study and comparison is needed to reduce the high variance seen in these results. A network of reliable sensors at spatially distributed GSI sites will allow for near-real-time data-driven decisions about maintenance and repair resource allocation, in addition to the analysis of long-term trends in performance.

Acknowledgement

This thesis is a result of my studies at Villanova University and my research as a member of the Center for Resilient Water Systems' PennDOT Hydrology Team. I would like to thank my advisors, Dr. Robert Traver, Dr. Bridget Wadzuk, and Dr. Gerald Zaremba, for their endless support, for sharing their wealth of knowledge and experience, and for entertaining my continuous requests to try new things. Madhat Fares, Danielle Galloway, Shaelynn Heffernan, and Matina Shakya, my fellow graduate students, made field work a breeze and provided many hours of troubleshooting, laughs, bouncing ideas around, and more troubleshooting. The Pennsylvania Department of Transportation and AECOM's partnership were key in inspiring, funding, and critiquing our work, and I am forever grateful for their support. Last, but certainly not least, thank you to my family for providing a rich educational background and my friends for their encouragement and support along the way.

Table of Contents

List of Figures

List of Tables

Introduction

Modern cities are facing new and quickly emerging threats from increasingly frequent large storms, which wreak havoc on urban streams and tax urban drainage and sewer systems to dangerous levels. In addition to intensifying storm events, urban centers are facing rapid expansion of impervious surfaces due to redevelopment, aging sewer infrastructure, and potentially rising sea levels. The combination of these factors means that flash flooding, sewage overflow or backup, and the associated pollution spread are increasingly becoming dangerous issues for urban communities. The US Environmental Protection Agency's (EPA) Clean Water Act (CWA) of 1972 (USEPA, 2009) outlines a set of regulations for wastewater and stormwater discharge to bodies of water. Specific requirements for areas with combined sewer systems (CSS) to reduce overflow frequency and volume are also outlined, with plans agreed upon by the EPA and municipal constituents. The Philadelphia Water Department (PWD) took a green infrastructure approach to meeting the requirements and, in 2009, released a 25-year plan called "Green City, Clean Waters" that implements a variety of community based initiatives and redevelopment requirements for reducing runoff from directly connected impervious areas (DCIA).

A major improvement to stormwater infrastructure is the design and implementation of green spaces, known as green stormwater infrastructure (GSI), in highly urbanized areas through the creative use of rooftops, sidewalks, and underutilized curbside lanes during retrofit projects. GSI encompasses a wide variety of site types, and specific implementations must be highly customized to each location. Philadelphia is quickly becoming the new home of tree trenches, porous pavement, curb bumpouts, and a variety of bioinfiltration locations, to name just a few GSI types. These green spaces, in addition to removing stormwater and reducing pollution, provide numerous social benefits to their neighbors (Callahan, 2019; Taguchi et al., 2020), although much work remains to be done in quantifying these benefits.

The present research aims to quantify the ways in which continuous monitoring, data storage, and well defined performance criteria can improve and inform the design process. Leveraging new monitoring technology, GSI systems can be updated with ever-evolving and continuously updated information, GSI systems' performance has the potential to expand greatly, making the cities of tomorrow sustainable, clean environments.

1.1 Study Site

The Pennsylvania Department of Transportation's (PennDOT) I-95 Girard Avenue Interchange Stormwater Project 25-year project to reconstruct eight of the fifty-one total miles between the New Jersey and Delaware borders (Figure 1.1), includes the construction of a series of GSI systems along the rehabilitated highway. Stormwater management practice (SMP) A, the specific study

location, is located at the west end of the Girard Avenue Interchange (GIR) GR2 subsection of the project (Figure 1.2) alongside the northbound lanes of the raised highway between Frankford Avenue and Shackamaxon Street. All GSI systems on the project are designed to handle runoff from a one-inch storm with a maximum allowable drawdown period of 72 hours, as per Philadelphia Water Department requirements for redeveloped areas (Philadelphia Water Department, 2021).

Figure 1.1.: I-95 Girard Avenue Interchange Stormwater Project map (PennDOT, 2018).

Figure 1.2.: I-95 Girard Avenue Interchange Stormwater Project GIR-GR2 section map (PennDOT, 2018).

SMP A is a linear bioinfiltration type garden nearly 100m in length but only 10-12m wide due to tight urban geometry constraining the site. Infiltration occurs in over 80% of the garden, with just a small portion in the center lined to protect neighboring structures. To better understand the site's hydraulic performance, monitoring equipment was installed in mid-2017 to quantify volume and soil state, and aid in the development of an EPA Stormwater Management Model (SWMM) by former Villanova Center for Resilient Water Systems graduate Elizabeth Calt (2018). Further work has been undertaken to improve the monitoring network's reliability and accuracy. This site is funded by PennDOT through the AECOM University partners program with the goal of investigating the long term performance of GSI as a stormwater management tool.

1.2 Research Goals

The primary goal of this work is to identify multi-faceted strategies for improved long-term monitoring solutions that will enable more thorough analysis of GSI. The steps outlined in the following chapters are intended to provide insight to PennDOT, AECOM, and their project partners in the continued study of wide-scale GSI projects on the I-95 Girard Avenue Interchange Stormwater Project, as well as to other partners in the city of Philadelphia who seek to expand GSI systems to the watershed scale and beyond. The creation of more uniform monitoring and long-term analysis practices will help move the design of GSI towards practices that are longer lasting, easier to maintain, and less prone to construction errors or failure. The steps outlined include three separate facets:

- The use of digital sensors with more accurate sensing and data transmission makes continuous monitoring more reliable and less prone to unrecoverable errors.
- Storing data in a flexible schema, such as the Stormwater Infrastructure Data Model in use at Villanova, enables long term study and comparison of a multitude of GSI locations.
- Event statistics generated from continuous monitoring data, such as recession rate and soil drying rate, can be used to approximate more labor-intensive field and lab tests to ensure continued performance.

Each of these topics will be discussed in detail in the following three chapters, with a focus on lessons learned through implementation at SMP A.

Standardized Data Collection

2.1 Background

In urbanized environments, the high percentage of surface area covered in impermeable asphalt or buildings causes increased runoff and reduced evaporation and infiltration (Figure 2.1). Urban environments have long faced challenges when it comes to handling this increased runoff in a ecologically friendly manner that avoids disastrous flooding while also minimizing the impact to stream and river health from pollutants and erosive flow rates. As cities' impervious footprints grow, sewer networks have became overwhelmed by the volumes seen during storm events, and outfall locations were added to avoid sewage backup into homes and streets. In Philadelphia, combined sewer systems (CSS) were the preferred method of handling large volumes of rainfall and sewage for nearly the city's entire history (Akhavan and Jianpeng, 2015), but have a major negative impact on the environment via combined sewer overflows (CSO). The EPA's oversight of CSO events, outlined in the Clean Water Act of 1972 and regulated by the National Pollutant Discharge Elimination System (NPDES) permit program (USEPA, 2009), require that cities address urban runoff and impose significant fines for overflow events. Only recently have green initiatives such as Philadelphia's "Green City Clean Water" expanded, allowing the city to expedite the creation of environmentally friendly means of removing stormwater (Philadelphia Water Department, 2018; Callahan, 2019). These systems allow urban stormwater, which primarily falls on impervious surfaces and leads to increased runoff as compared to natural environments, to be captured and managed largely in place (Heffernan et al., 2016) by a combination of green infrastructure methods including rain gardens, tree trenches, green roofs, and pervious concrete. The type of system chosen for a particular site depends on existing site infrastructure (e.g., buried utilities, surrounding structures, etc.), grading, available space, and loading ratio (directly connected impervious area (DCIA) vs system footprint).

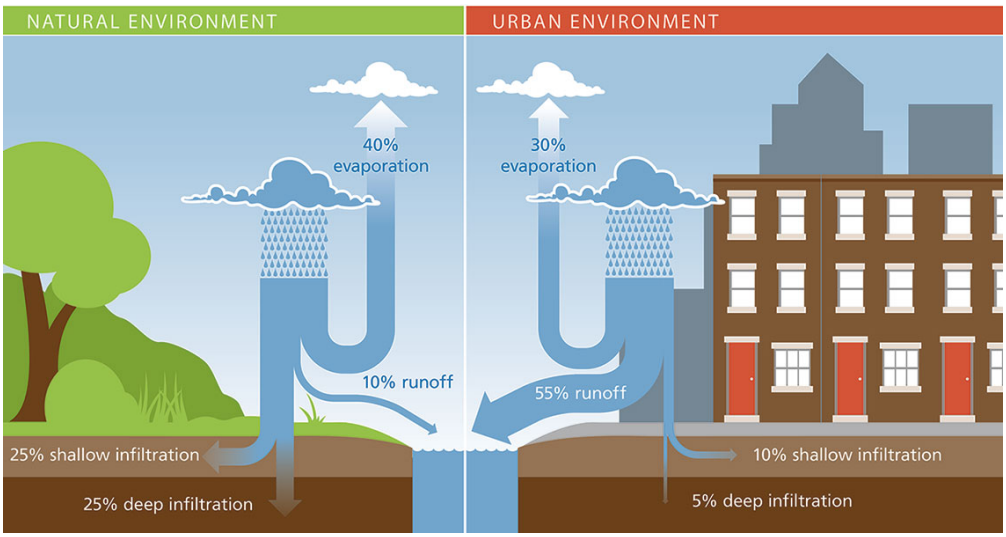

Figure 2.1.: Natural vs Urban water budget (Philadelphia Water Department, 2021).

Measuring any natural or anthropogenic system is not without its challenges: atmospheric and physical parameters are subject to rapid change from weather patterns and equipment must be able to withstand harsh outdoor conditions for extended periods of time. After all, nature obeys the laws of entropy, and energy tends to dissipate towards its lowest potential, and in doing so becomes a major force with which to be reckoned. The water cycle that forms our climate is driven by solar heat inputs to this energy system. Rainfall is generated from the accumulation of evaporated water in the atmosphere, which then condenses and falls to the ground. Due to the randomness of entropy, no two storms will ever be the same in magnitude, duration, or intensity profile, so constructed solutions must be flexible enough to handle storms ranging from slow, drawn out events to short, intense events (Dourte et al., 2015; Maier et al., 2020).

Collecting valid data is the first step to any rigorous experiment. The methods used to collect this data must be reliable and easily repeatable for other parties who wish to replicate results. That is to say identical inputs to a measurement system, given the same external conditions, should yield the same results. This is a fundamental premise of the scientific method. The most important aspects of monitoring a green stormwater infrastructure (GSI) system are a reliable data logger with minimal downtime, connected to a central data server (or "cloud") that collects data at a regular interval and redundantly stores that data for further analysis. The data logger must be able to poll all connected sensors at timely intervals such that data measurements are associated with an accurate timestamp. Measurements must be accurate, while their precision can be based on the unit of measure. The ability to continuously monitor GSI conditions enables an understanding of design specifications versus real world performance. While models are useful tools for understanding and managing expectations for GSI, they can only go so far at capturing real world knowledge and the many interactions of GSI conditions that create nearly endless combinations of unique conditions.

This chapter outlines a framework of best practices pertaining to the continuous remote sensing of GSI conditions developed for stormwater management practice (SMP) A in the GR2 section of the Pennsylvania Department of Transportation's (PennDOT) I-95 Girard Avenue Interchange Stormwater Project. This framework has been developed to be reliable, consistent, easy to implement, and expandable to other GSI systems across Philadelphia with minimal additional configuration. A primary goal of this research effort is to not only inform better data collection, but also to enable data to be collected at scale across many sites in a project or across many different projects. A consistent, uniform framework for monitoring sites would allow for more widespread analysis of GSI's performance, maintenance needs, and impact on the natural environment (Burcin et al., 2014). This aligns well with the I-95 Girard Avenue Interchange Stormwater Project's long term horizon for construction, monitoring, and analysis of individual GSI systems and wider wholistic project or regional scale analysis. A uniform approach across many individual sites ensures that statistical contrasts at the regional scale are accurate and unbiased. To that end, measurements must be precise and data recording and transmission must be unhindered by the inherent randomness or other inconsistency of environmental systems. Myriad challenges exist in monitoring GSI systems, but the focus of this discussion is accurate measurements of ponded and flowing water, which exhibits a wide

variety of properties ranging from depth and velocity to random formations of currents and internal feedback in the form of turbulence (Mays, 2010). This monitoring framework has been established based on work performed at SMP A, a well developed and successfully performing GSI installation, for use across other PennDOT sites or for wider adoption across a regional scale in Philadelphia to enable long-term, meaningful comparisons of GSI performance across many systems. The framework is intended as a basis for answering questions about ongoing maintenance requirements, identifying under-performing sites, and quantifying with high accuracy the environmental benefits of successful systems.

2.2 Site Configuration

Situated along the south side of Interstate 95 northbound, SMP A handles inflow from the elevated roadway surface's rainfall runoff that flows through a network of pipes beneath the highway. Most of the DCIA lies in the northbound highway section, per LiDAR surveys conducted in 2017. SMP A is a linear bioswale type rain garden split into three sections by two check dams placed at roughly the third points of the garden (Figure 2.2).

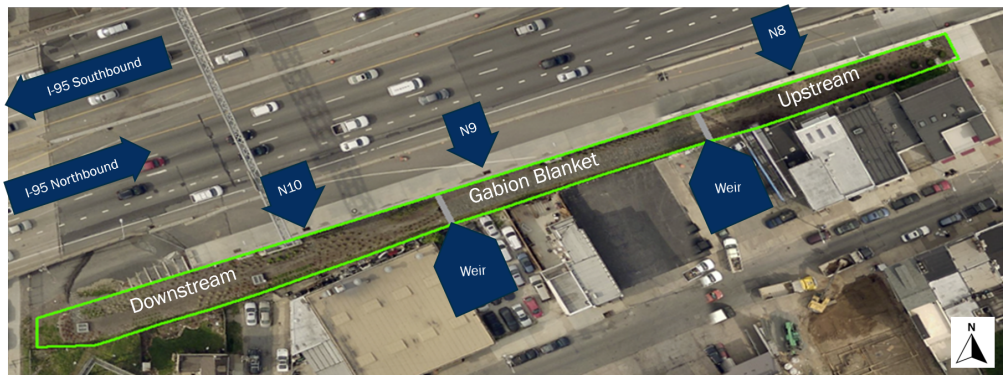

Figure 2.2.: SMP A site layout.

The upstream portion contains one inlet (N8) that is a 76.2cm diameter reinforced concrete pipe (RCP). This upstream section is planted with local, salt tolerant grasses and small coniferous trees. The surface is mulched during annual maintenance cycles, and the cross-sectional geometry is gently sloping (Figure 2.3). The upstream section terminates with a 45° steel weir plate situated in the center of a plywood check dam that reduces flow rates, encourages infiltration by retaining some water, and measures inter-garden flow once overtopped using the standard weir equation (Figure 2.4).

The center section has an impermeable fabric beneath the entire south half of the section to prevent infiltration from damaging neighboring structures' foundations. The entire section is constructed with gabion basket devices, which are intended to help prevent erosion and cut down on weed growth.

Figure 2.3.: SMP A upstream profile during a simulated runoff test, September 2020. Note the shallow side slopes.

Figure 2.4.: Check dam at the downstream end of the central gabion blanket section of SMP A.

There is a single 45.7cm RCP inlet (N9) in this section and the cross section has more pronounced slopes (Figure 2.5). The gabion blanket section is similarly terminated at the downstream end by a 45° steel weir plate flush with a check dam to retain as much water as possible.

The downstream portion of the garden contains one more 45.7cm RCP inlet (N10), as well as two concrete overflow structures connected to Philadelphia's CSS identified as B1 and B2. These structures have pressure transducer (PT) devices attached to the outside of the structure for measuring the ponded water level. Inside the structures are 22.5° steel weir plates inside covering the CSS connection (Figure 2.6), with another pair of PT devices for measuring the outflow from the system, again using the standard weir equation.

Due to SMP A's location at the border between the completed GR2 section of the project and the upcoming GR1 section to the west, there is a temporary construction ramp at the downstream end of

Figure 2.5.: Gabion blanket section showing highly sloped banks.

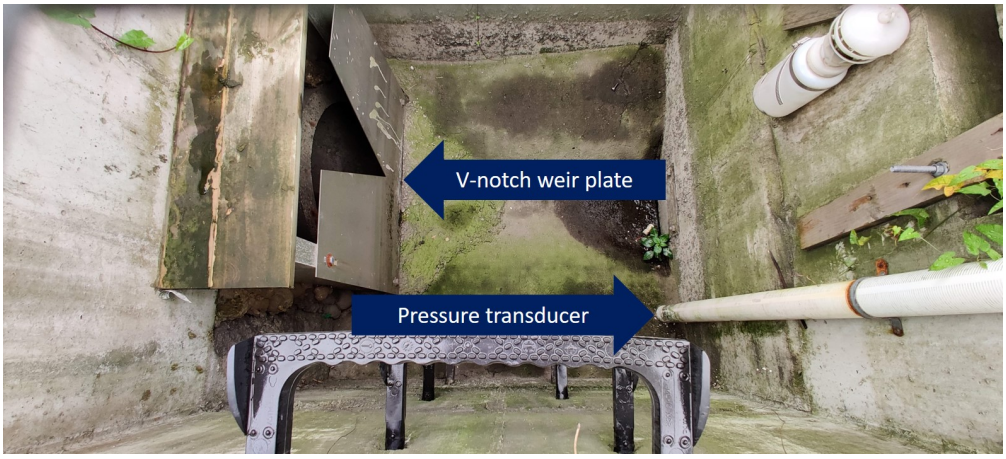

Figure 2.6.: B1 outlet structure with weir plate covering CSS connection.

the basin that leads from the elevated roadway to grade level adjacent to Frankford Ave. The ramp is not intended to convey inflow, but the grade of the highway, misalignment of inlet grates (Figure 2.7) on the highway, and lack of barrier at the top of the ramp mean that a substantial amount of runoff enters the basin after flowing down the ramp. All pressure transducers used at SMP A are a model CS451 digital pressure transducer from Campbell Scientific and are used in their SDI-12 communication mode.

2.2.1 Measurement: Sensors and Structures

To aid in repeatable, accurate, and timely measurements, sensors are deployed at a variety of locations throughout SMP A. Pressure transducers in a variety of locations monitor both ponding

Figure 2.7.: Misaligned inlet grates allow water to bypass and flow down the temporary ramp.

level (outside the B1 and B2 outlet structures) and the depth behind weir plates (inside B1 and B2, and behind two check dams separating garden sections). The flow rate over weir plates can be calculated using the standard weir equation (USBR, 2001).

Inlet flow at N8, N9, and N10 was originally measured by a single BlueSiren Dual Wave Doppler Area-Velocity (AV) sensor at each inlet. These sensors are fixed to expandable steel bands that secure the apparatus inside the end of the RCP by means of a screw jack. Located at the downstream end of the AV sensor is a PT that measures flow depth, and outputs a 0-5V analog signal proportionate to the observed depth. The voltage response must be calibrated with several known depths to establish a valid conversion equation (see section 2.3). At the upstream end of the AV sensor, the dual wave doppler measures flow velocity and outputs an 8-bit serial signal, with the number of doppler pulses reflected by the water corresponding to the velocity in millimeters per second. Both parts of the AV sensor require 12V direct current (DC) power supply, which is managed by the CR6 data logger.

Soil moisture is measured at two locations using sensors at 10cm, 35cm, and 60cm depths, with an additional, redundant sensor at 35cm for quality assurance purposes. Stevens Water Hydraprobe sensors (Figure 2.8) are used to measure soil moisture level, conductivity, resistivity, temperature, plus real and imaginary dielectric permittivity. Hydraprobes use four stainless steel prongs to measure the soil's parameters and "take into account the energy storage and energy loss across the soil area using a 50MHz radio frequency wave" (Stevens Water, 2021). The sensors are located in the middle of the downstream basin between outlet structures B1 and B2, as well as in the upstream basin approximately 2 meters from the check dam.

Figure 2.8.: Stevens Water Hydraprobe (Stevens Water, 2021).

Additional climate sensors include air temperature and relative humidity, solar radiation, wind speed and direction, and barometric pressure. These sensors are located atop a 60-foot telephone pole adjacent to the I-95 bridge over Frankford Avenue that extends roughly 40 feet above the highway surface for measurements uninterrupted by traffic or highway structures. These sensors are collectively referred to as the "Weather Station," and are a physically separate network from the main garden.

2.2.2 Data Collection and Transmission

The entire sensor network is controlled and monitored by three Campbell Scientific CR6 data logger devices. These are located on poles attached to the B2 outlet, at the base of the weather station pole, and on a small post in the ground next to inlet N8. The data loggers are inside protective boxes and have solar panels mounted adjacent to provide power and charge the 12V battery also located in each box. The CR6 runs a program written in Campbell Scientific's CRBasic programming language, which is a procedural language similar to the BASIC family of languages. The program defines variables, storage tables, polling and logging frequencies, and monitors the CR6's health and battery status.

Communications at SMP A are achieved via Campbell Scientific's Pakbus protocol, which allows multiple CR6 loggers at one site to be connected via external Wi-Fi modules (model NL240 and NL241) or RF devices (model RF407). In early 2019, the existing RF devices were swapped out for Wi-Fi devices which improved range, reliability, data link speeds, and security. Campbell Scientific LoggerNet software is used to remotely manage the data loggers, collect data every three hours, and identify issues that require on-site maintenance.

Remote connections are achieved via a Verizon 4G-LTE cellular modem which provides internet access for remote connections and download of data. The modem is paired with the CR6 at the Weather Station, as this setup has the fewest instruments attached, and therefore the lowest power requirements. Remote access is necessary for downloading data at a regular interval and monitoring the network's health.

There are four types of measurement performed by the CR6 data loggers: analog voltage difference, serial, pulse count, and digital. Each have their advantages and disadvantages.

Analog voltage difference sensors, such as the BlueSiren AV depth reading, convert a 5V or 12V DC supply into a 0-5V response, which is read by the data logging device. This method requires a calibration equation to translate the response voltage into meaningful units for the measured variable. The reading is performed by the CR6 device and is almost always separated from the point of measurement (sensor) by a substantial length of wire (minimum 10m, maximum 90m). Calibration is performed by applying a known reading to the sensor (depth of water in the case of a PT sensor), and recording the response voltage at several different readings. A linear relationship (slope and intercept) are then calculated from the points collected, and this equation is written directly into the CR6 program. Due to electrical resistance of copper wire varying with temperature (Eargle, 2002), the resistance of a copper wire increases by 15% over a 40° Celsius range when warmed by direct sunlight. This is amplified by the long length of wire running between the sensors and data logger - nearly the entire 90-meter length of SMP A in some cases. Change in resistance from the calibration condition is not compensated for by the CR6, so measurements made at temperatures different from the calibration temperature are subject to error. Historically, wiring has been installed at least partially above ground, exposing it to sunlight which can cause swings in temperature. Existing in-situ calibration procedures may help reduce this, but further study is warranted.

Serial sensors send pulses of electrical signals at a given voltage and baud rate. The CR6 data logger reads these serial values by opening a serial buffer on the port a serial sensor is connected to and can respond to a wide variety of serial protocols and refresh rates. The Blue Siren sensors used at SMP A communicate velocity via UART - TTL at a baud rate of 4800. Serial communications are less prone to error or disturbance introduced by the transmitting wires but have limits on how far a valid signal will transmit. Additionally, only one serial sensor can be active on any given data logger port at once, so some inlet AV sensors are connected to switched 12V power outputs on the CR6 to allow multiple sensors to share a single physical port. This complicates both the physical wiring of the sensors at the site, as well as the code required to handle toggling the power state in coordination with taking measurements. Additionally, it is unknown what effect this power cycling of the AV sensors has on their performance.

Pulse count communications are used primarily by rain gauges. Measurement occurs when a specific event happens, such as the tip of a tipping rain gauge. The sensor briefly connects the measurement port to ground, creating an electrical pulse that is detected by the CR6. This means pulse count sensors are among the simplest in terms of wiring, as they require only a measurement and ground lead.

Finally, digital sensors, including the CS451 PT, Hydraprobes, pyranometer, and thermometer, use the SDI-12 protocol which allows up to 62 addressed sensors on a single communications port. The SDI-12 protocol allows the CR6 to poll sensors for up-to-date data on demand, and read the results

back nearly instantaneously. The digital transmission of data eliminates the need to run separate communications cables for each sensor, and consolidates the ports used on a CR6 to just one for most applications. Sensors are able to relay several measurements to the CR6 in an array of values without concerns surrounding wire length or environmental conditions.

2.3 Challenges

Water flowing through pipes or across surfaces is affected by a variety of factors: slope, roughness, temperature, and geometry (Mays, 2010). The slightest imperfections in the surface or inconsistencies in the roughness can introduce turbulence, a non-uniform flow regime not well suited to measurement by sensors best suited for steady, uniform conditions. The act of falling from highway level to ground level, where measurement takes place, means that water has a high amount of kinetic energy and is thus more likely to splash around, disturbing calmer water and imparting some of its energy. Turbulence is best handled by some form of flow straightener, calming device, or impoundment behind a flow-control device such as a weir plate that uses temporarily stored water to remove energy from incoming water. At lower flow rates, this can also be accomplished through the use of high permeability foam, which has the added benefit of filtering out larger debris particles washed in from the highway surface.

Attempts to characterize the "average" storm year have been successful (Albright and Schramm, 2018) and provide a means of estimating rainfall over a given period of interest. This rainfall can be used as the input for a system model, such as was created by Elizabeth Calt, MSCE ‘18 (Calt, 2018). The Environmental Protection Agency’s (EPA) Stormwater Management Model (SWMM) created by Calt estimates runoff, storage, and overflow volumes and was used to estimate the expected values for the sensors measuring flow through the three RCP inlets during both historical rainfall records, and two design storms (2 year, 24 hour and 10 year, 24 hour).

Historical data from inlet N8, N9, and N10 show that the depth portion of the AV sensors is frequently negative, which is indicative of improper calibration or sensor malfunction. Despite sensors regularly being calibrated carefully, and returning valid measurements immediately after calibration, measurement drift continues to be an issue. Little of the data in the period of study is valid, with many negative values occurring during storm events, so the best estimates of inflow rates are from the aforementioned SWMM model developed by Elizabeth Calt. In addition to negative inflow rates being invalid, the model suggests that flow rates lower than what the Blue Siren AV sensors are capable of detecting may occur frequently, depending on the size and shape of the runoff hydrograph.

Finally, the calibration procedure for existing Blue Siren AV sensors is both tricky to perform and subjective based on human error. After removing the sensor from the steel band holding it inside the pipe, the sensor is submerged in a five-gallon bucket with water measured by hand to the nearest

millimeter. Simultaneously, the data logger program must be updated to read out the raw voltage received, and these values are recorded in an Excel spreadsheet. After taking measurements at several depths, typically ranging from 25-150mm, the hand measured depths and corresponding voltages are transformed into a simple linear relationship (slope and intercept), which is then updated and enabled in the data logger's program. While little to no drift over time is noticeable in historical calibration records, there is some oscillation that occurs do to the imprecision of human measurement during the calibration process.

2.4 Solutions

2.4.1 Temporary Ramp

In June 2019, approximately 60 sandbags were added to an existing asphalt berm on the ramp. These sandbags were placed to guide water through an 0.8-foot (20.32cm) H-flume with a PT sensor to capture the hydraulics of the ramp (Figure 2.9, OpenChannelFlow, 2021). The flume is sourced from OpenChannelFlow, a company in Boise, ID that focuses on agricultural flow measurement devices.

Figure 2.9.: Temporary ramp at the downstream end of SMP A.

The addition of the H-flume near the B1 outlet structure was driven by a desire for better understanding of hydraulic inputs to the basin that were bypassing storm drains on the roadway surface and instead flowing down the temporary ramp. The H-flume is made of reinforced fiberglass and is sized based on preliminary output from Elizabeth Calt's SWMM model using both design storms and historical rainfall data collected at the site through early 2019. It has a 1-meter (3-foot) approach section, and two cylindrical measurement wells on either side of the pour point which are 150mm deep (Figure 2.10). Using the known offset of these wells, the data recorded by a CS451 PT can be used along with the calibration curve supplied by OpenChannelFlow to establish the flow rates observed:

$$F = -0.00707921 + 0.04898248\ d^{0.5} + 21.70307374\ d^{1.5} + 365.9132927\ d^{2.5} \tag{2.1}$$

where F is flow in liters per second and d is depth of flow in meters, which gives this flume a range of 0.0085 - 12.94 L/s (OpenChannelFlow, 2021). Flumes have long been an agriculture industry standard for measuring flows into and out of fields and ponds, and they are well equipped to handle a large range of flow rates. While the sandbag wall has been tampered with on several occasions, it largely remains intact and able to direct water through the flume (Figure 2.11). Only minor leakage was observed during a fall 2020 simulated runoff test (SRT) when a high flow rate was pumped into the site.

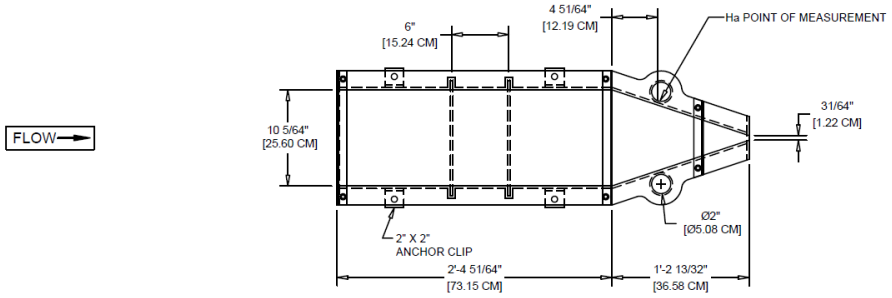

Figure 2.10.: 0.8-foot OpenChannelFlow flume plan view (OpenChannelFlow, 2021).

2.4.2 Inlet Flow Measurement

To estimate the AV sensors' range, a suite of tests were performed using a lab simulation of field conditions. Sensors were attached to an identical steel band and placed inside a 20cm PVC pipe. The roughness of the pipe is different than that of the RCP installed in SMP A. However, given the turbulence introduced by the flow falling from highway level to ground level a short distance upstream of the measurement point, roughness does not control the flow profile (Mays, 2010).

Figure 2.11.: Flume installation at the bottom of construction ramp, 2019.

The lower bound for Blue Siren AV sensors was determined in Villanova's water resources lab by calibrating a sensor using the same procedures used in the field, and then recording the sensor output and flume flow rate simultaneously. Analysis showed (Figure 2.12) that the AV sensors are not valid below 4.5L/s in the 20cm test pipe. This deviation was found to occur separately in depth and velocity readings around an average of approximately 25mm depth and approximately 30mm/s velocity, which equates to 0.1 L/s in a 45.7cm pipe (N9, N10) or 0.14 L/s in a 76.2cm pipe (N8), although it was not possible to test flow rates this low in the lab. A second Blue Siren system consisting of a serial ultrasonic depth sensor mounted to the top of the pipe combined with a "microvelocity" sensor that has a lower profile than the current Dual Wave doppler AV sensor was tested in an identical fashion and found to have lower velocity accuracy, but consistently better depth accuracy. Figure 2.12 compares the two Area-Velocity measurement methods, with the insert at right showing both systems measuring significantly lower than the reference flow rate.

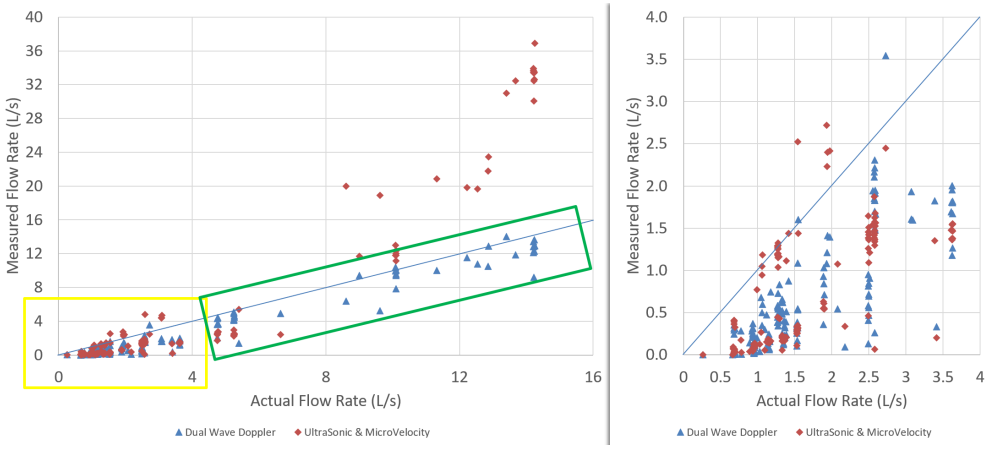

Figure 2.12.: Area-Velocity measurement comparison results.

In order to capture lower flow rates, a restricting structure such as a weir plate or flume is necessary, so that the flow is forced through a narrower opening where it can be measured more precisely. The solution chosen for implementation is a weir plate sized to fit inside a 10.2cm PVC pipe (Figure 2.13), which sits at the outlet of a catch basin small enough to fit under the pour point of the RCP inlets (Figure 2.14). This solution minimizes the disturbance to existing infrastructure and involves the least standing water between storms, which is a concern for biological safety during hot summer months.

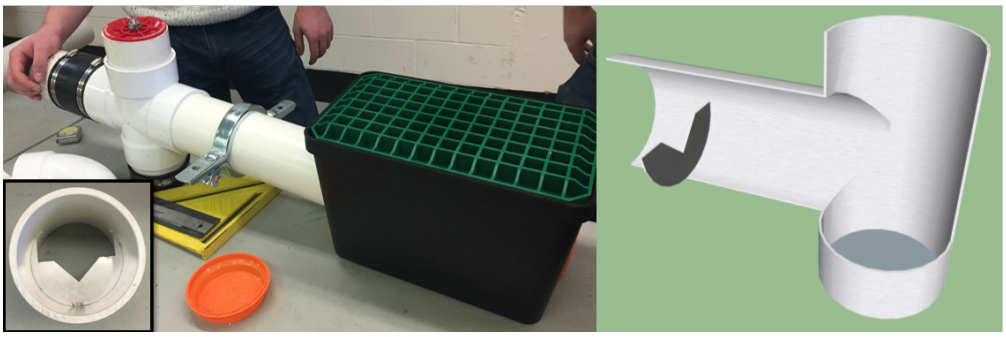

Figure 2.13.: (Left) Low flow measurement setup prototype being constructed in the lab. (Right) Cutaway section of the device outlet.

Figure 2.14.: Low flow measurement installation.

Since installation during summer 2020, the low flow measurement systems at N8, N9, and N10 have captured at least a portion of the flow of each event. However, several issues remain: the capture efficiency of the inlet (green plastic grate) is lower than anticipated, as a significant percent of flow splashes off and is not captured in the system (Figure 2.15). Despite this, the inlets show a clear response to rainfall depths as seen in Figure 2.16. This data, collected at N9 during a 47mm storm

event show a total volume of 3.2m^3, which equates to just 3.7mm of rainfall over the approximate drainage area of 882m^2 at N9. While this could be explained by changes to the drainage area since the LiDAR survey was performed in 2017, bypass due to inlet grate design, or poor capture efficiency at the pour point into the garden, it is evident that the system needs further tuning.

Figure 2.15.: Splashing at N9 low flow installation during September 2020 SRT.

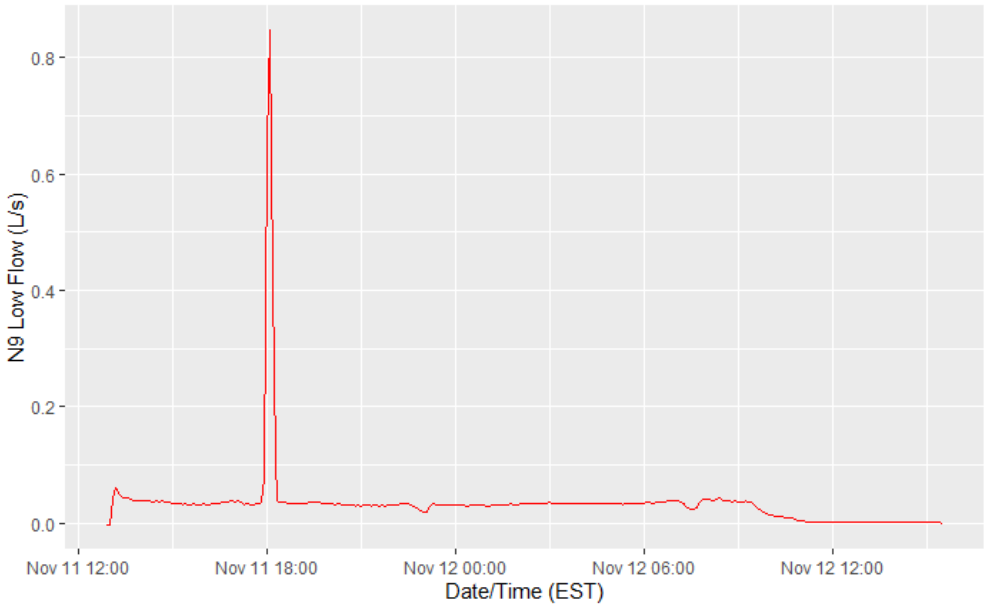

Figure 2.16.: N9 low flow data for 47mm storm event.

2.4.3 Future Work

Investigation into the combination of Blue Siren AV sensor for velocity readings and Massa Ultrasonic depth sensor is ongoing. Preliminary lab results (Figure 2.17) indicate that this system will be sufficiently accurate in combination with the aforementioned low flow system due to their overlap in flow range, shown in orange.

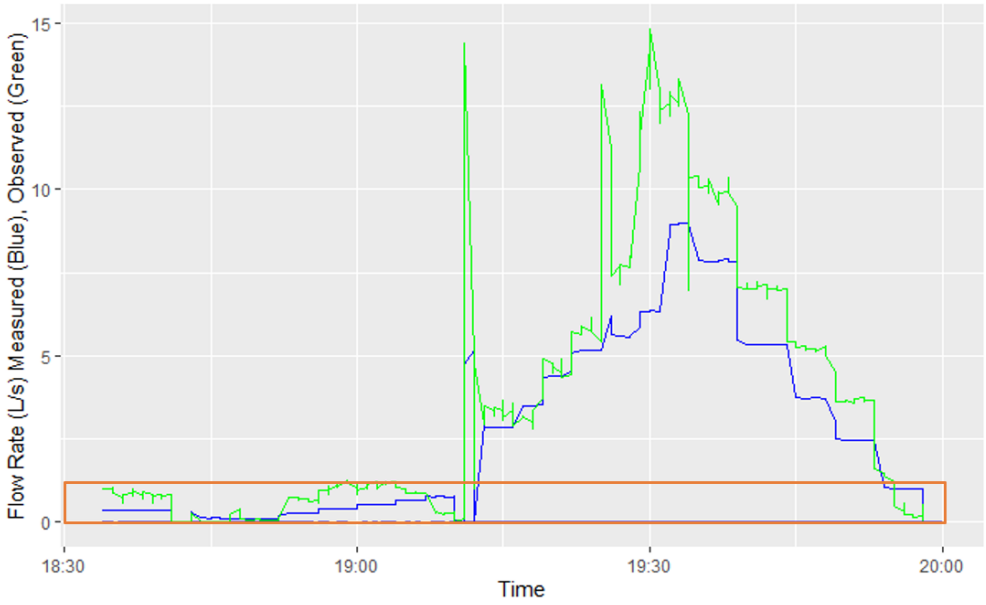

Figure 2.17.: Ultrasonic depth plus Blue Siren velocity preliminary flow comparison.

This system will require further tuning to ensure that the flow rate measurements produced are sufficiently accurate. The preliminary results show some evidence of hysteresis, when increasing and decreasing flow rates respond differently, but more controlled lab testing is necessary to identify areas for improving this combination of sensors and the CRBasic program that combines the data they produce into a flow rate. Further testing at more granular flow rates (smaller jumps in flow rate) will help demonstrate whether this system is ready for site implementation.

Finally, due to the variability of analog signal measurement over the long distances in SMP A, conversion to digital measurement will further increase accuracy and reliability. All lab tests discussed in this chapter have used an analog to digital converter (ADC) between the sensor and data logger as a proof of concept. The device of choice is a Vegetronix SDI-12 Analog Sensor Translator (Figure 2.18), which supports up to four 0-5V sensors attached and is fully compliant with the SDI-12 protocol. This inexpensive device will greatly simplify the wiring schematics necessary for using two different sensors (Ultrasonic and AV) for measuring flow at the inlets, as well as shorten the length of wire over which an analog signal must travel.

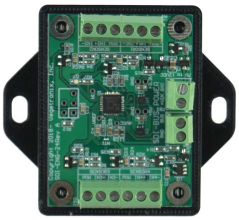

Figure 2.18.: Vegetronix SDI-12 Analog Sensor Translator.

2.5 Conclusions

Measuring flow in natural, uncontrolled conditions is difficult and requires creativity. Understanding the intricacies of sensor configurations, communications, randomness of hydraulic characteristics, and opportunities for error can help mitigate invalid data. Digital communications significantly increase a sensor's ability to report valid data, ensuring that measurements taken in one location are accurately represented and stored by a data logger several hundred feet away. The increased confidence in data collected will enable future analysis of longer-term trends, so investments and efforts made now in upgrading systems can have lasting benefits for GSI studies across all sites. Furthermore, significantly less time will be consumed processing the data in quality control procedures leading to more resources available for actual analysis.

Robust Data Storage

3.1 Background

The ability to gain insights from any data set is rooted in good organization of that data. Large data collections quickly become overwhelming, complex, and site-specific without a comprehensive system for tagging and differentiating data. None of these qualities are good for developing or deploying easily transferable metrics for evaluating and comparing performance at many GSI sites. Instead, consistent metadata and uniform data format are two powerful tools for ensuring that data remains accessible and globally applicable in the sense that two data points collected at different sites can be easily understood to measure the same parameter. The ability to streamline the combination of data from many different sites into a single database with consistent data descriptions has the potential to unlock new insights into performance, long-term and average maintenance needs, and lifespan predictions. The accessibility of consistently described data reduces the time spent deciphering the meaning, source, and integrity of a dataset.

The need for consistent data definition is not new but has greatly expanded with the decreasing barriers to entry that enable far wider application of monitoring equipment and data collection. Civil infrastructure has historically taken an approach to building that favors careful planning, site analysis, and construction validation. The ability to conduct long term monitoring at a wide-spread scale has been lacking due to the difficulty of collecting and analyzing the data necessary at said scale (DelGrosso et al., 2019). The ongoing data revolution has the potential to change that, with modern database techniques enabling consistency and accessibility (Maidment, 2008; Abdallah and Rosenberg, 2019). Several examples of centralized water data repositories exist already including the International Best Management Practices (BMP) database, which provides many observational data points across a range of GSI types and locations (Low Impact Development Center, 2004), the EPA's Storage and Retrieval System (STORET), and USGS' Nation Weather Information System (NWIS). However, the data in the International BMP Database is largely focused on discrete samples, and the structure of all these systems is not sufficiently standardized to allow for efficient organization of continuously recorded time series data or collaboration with data across different systems. While discrete data is an important tool in the study of GSI, continuous data is also highly useful and presents a much larger challenge due to the vastly larger scale of the data collected (Wadzuk et al., 2021b). A collection of discrete data points may number in the several hundred or perhaps several thousand, while continuous data easily scales to several million values per site over the course of a multi-year study. A system of standardized "controlled vocabulary" could help alleviate some of these issues (Maidment, 2008), but would require a great effort to sort through many years of historical data, not all of which may be easily identified and traced back to its proper origin. Therefore, a modern system with the flexibility to accommodate the scope and size of data collected is necessary to enable multi-year, multi-GSI system studies that can fully take advantage of the expansion in data collection (Wadzuk et al., 2021a; Meng et al., 2017).

The Consortium of Universities for the Advancement of Hydrologic Studies, Inc (CUAHSI) have created the Observations Data Model (ODM), which aligns better with the needs of continuous monitoring data and has features for publishing data across different research groups (Horsburgh et al., 2008; Maidment, 2008). However, there are limitations on data categorization because variable and site codes are used to uniquely identify a data point, so sites with multiple identical variables (e.g., two inflow measurements at different locations within a site) are not possible without creating redundant variables. Additionally, there are data privacy concerns for research sponsors interested in organizing their data without making it publicly available, as required by the included WaterML integrations. The schema for the ODM provides a robust means for storing time series data, and is flexible enough to be expanded with the additional metadata collected at various VCRWS project sites, including at SMP A on the PennDOT project. The ODM is the basis for the expanded platform implemented to handle GSI data at VCRWS.

3.2 VCRWS Stormwater Infrastructure Data Model (SIDM)

Recently implemented at VCRWS, the Stormwater Infrastructure Data Model (SIDM), is a modern database approach to disparate data organization using relational tables. By abstracting the metadata layer of information away, spatial and temporal linkages between data can be stored without repetition of the metadata itself, making storing and accessing data easier and faster. The SIDM is based on the schema of the ODM and shares many of the same features, but goes a step further by introducing the ability to associate data with specific sample locations within a given site as well as the implementation of protected sharing features. The web interface was also refreshed to have a modern look and easy navigation. User permissions and data sharing abilities are facilitated by pages accessible to site administrators. New metadata definitions for sites, sampling locations, variables, and methods must be created via their respective pages before data related to the new definitions can be imported. Data import is available to coordinators for each site, allowing the addition of a single data point for discretely sampled data, or uploading a formatted data file for continuous data.

Throughout the development process, the database was designed to align with principles of FAIR (Findable, Accessible, Interoperable, Reusable) data (Wilkinson et al., 2016). The SIDM's updated data format is intended to meet these criteria to enable more robust and sustainable study of data stored within. The SIDM makes finding data easy through a simple GUI search page with filters for each type of metadata. The simple, yet effective interface for building queries to search the database allows authorized users the ability to download all data values and associated metadata to a CSV file with the click of a button, making all data easily accessible. Metadata is stored in data logs, which remain accessible even when their associated data points have been removed. This has the potential for further analysis with expanded capabilities such as an application programming interface (API) for easy pre-processing and cleaning. The data model is reusable, as any given combination of space

(site, sample location), time, and variable codes uniquely identifies a data point and can be used as a reference for further analysis. This traceable metadata associated with every data value allows future users to understand what conditions under which the data was collected. In addition to making data easier to query, this means that auditing work to determine when and how measurement procedures or methods change becomes possible because each new iteration of a data collection method can be tagged with unique metadata.

3.2.1 Metadata Standardization and Data Formatting

The central feature of the SIDM is the relational nature of the metadata abstraction (Horsburgh et al., 2008). Keeping data and metadata separate allows for more efficient storage because there is no duplication of information and for the flexibility to expand the scope of data stored without updating the storage structure. Each database table uses a primary key column to uniquely identify records. These primary keys, when used in other related tables to identify pieces of related information, are called foreign keys. Foreign keys establish the relationship between a data record and its corresponding site, sample location, variable, and other metadata of interest. Each of these pieces of metadata are stored in their own tables with consistent information describing each record within (Figure 3.1, enlarged in Appendix A). By linking the related records with the appropriate keys using a unique identifier, duplicate information is not created. The structure of these relationships allows infinite combinations of spatial and temporal data, while remaining flexible enough to accommodate new or changed data streams without the need for actual structural updates to the data format.

Data is collected in a wide format, where each unique variable is stored in its own column and a new row of data is appended to the table for each observational timestamp. From a storage space perspective, this means that there are $N * M$ rows in the table, where N is the number of observations and M is the number of unique variables. This is sufficient for storing a dataset related to one and only one site where variables directly correspond to sampling locations. However, each newly collected variable requires the addition of another column to the table or the creation of a new table, which results in the storage of many missing variables or disjointed data records, respectively. This approach does not work well when used to combine data from multiple sites, as the column names are the sole form of metadata available to describe variable origins.

An alternative approach, and the one taken by the SIDM, is to abstract the information about a data point's origin (site, sampling location, variable, detection limit, collection method, sample type, and offset information) to separate tables, since this information pertains to many data records. Storing the identifying metadata only once and referring to it via foreign key constraints in the data records table creates consistency in both metadata format and data record format, which enables faster querying of the data records through the use of covering indices. Columns in a database containing keys are indexed for faster performance, which requires additional storage space but significantly reduces computational time (Lightstone et al., 2010). A covering index is a composite index made

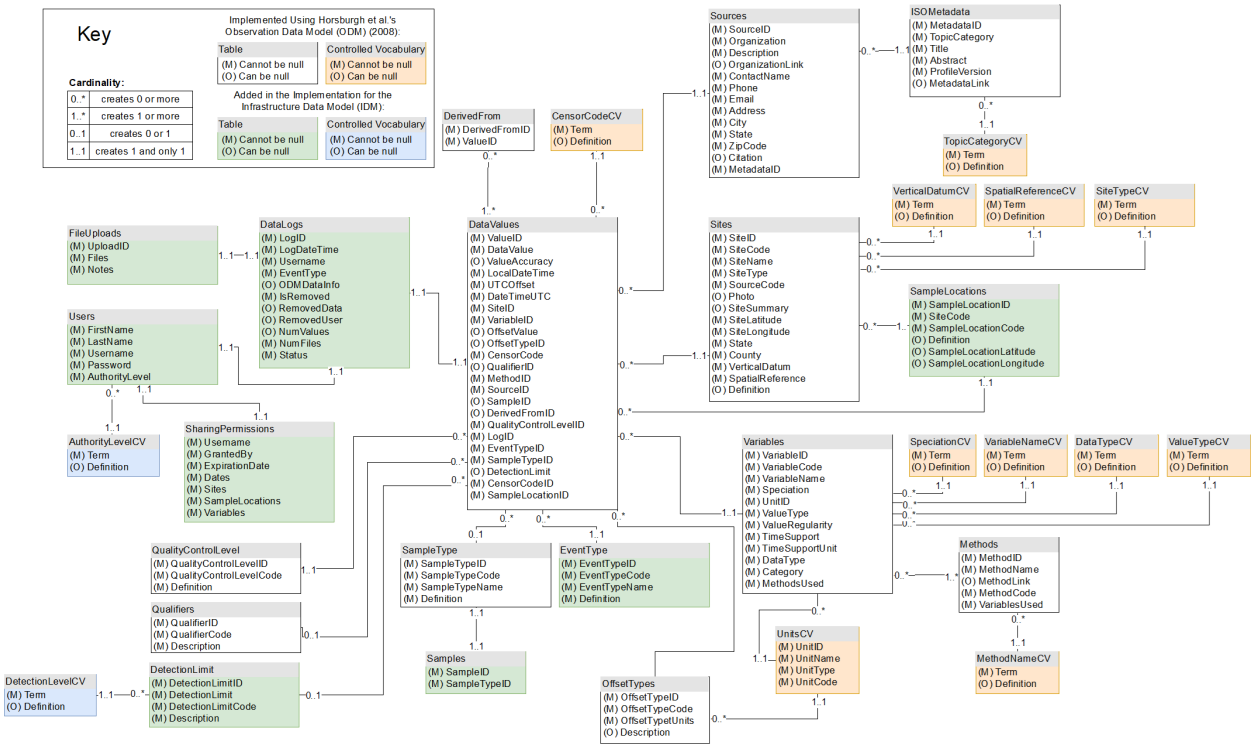

Figure 3.1.: SIDM database schema (Smith et al., submitted).

up of a set of all possible columns that could appear in a query, meaning the database only has to consult the index values in order to return results. This format is referred to as long format, as the number of columns in the data records table is fixed according to the desired metadata for each point and a nearly infinite number of observation rows can be appended without any change to the database schema. This format only requires that the additional metadata for new variables, sites, or sampling locations be added to the appropriate tables before new data records can be imported.

Transforming continuous monitoring data from wide to long format requires a process called melting, which can be accomplished via packages in nearly any modern general purpose programing language. During melting, original column names are mapped via a secondary input file containing key-value pairs of corresponding original variable names and the expanded metadata for site code, sample location, and variable name. This expanded metadata is appended to each data record and then inserted into its own row in the output file, along with the other global inputs available (detection limit, sample type, offset type, event type) since these values are expected to be constant for chunks of continuous monitoring data collected in a single file. For simplicity of use for those not familiar with programming, a graphical user interface version of this process was created using the R Shiny (RStudio Team, 2020) platform (Figure 3.2). The web application expects an input file containing raw data with one 'timestamp' column followed by any number of variable columns and a secondary mapping file input with the aforementioned key-value metadata pairs, and specifies sensible global defaults for the other necessary metadata. Additional date filtering is available if partial file upload is desired. This is useful in the case where data is appended monthly to site's data file, as that data

file can still be processed by the program, with the appropriate filtering in place. Input data is easily processed at the click of a button, and the output is a file ready for upload directly through the SIDM interface.

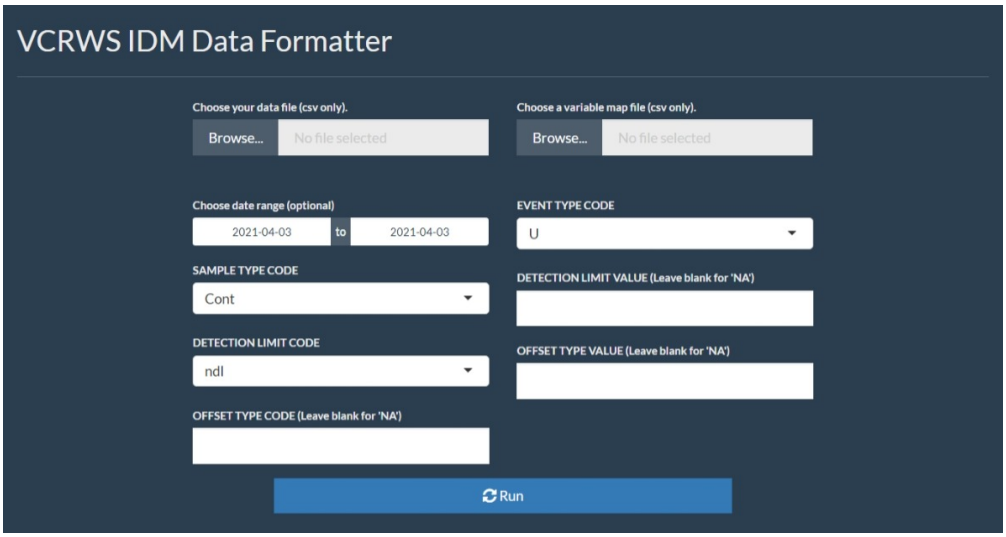

Figure 3.2.: R Shiny web app for data formatting (Smith et al., submitted).

3.2.2 Query Performance

The SIDM web application consists of several pages for uploading data, managing user authentication and authorization, and creating new controlled language for expanding the available metadata. User accessible pages include a map of all sites showing relative locations with links to a brief site description (Figure 3.3).

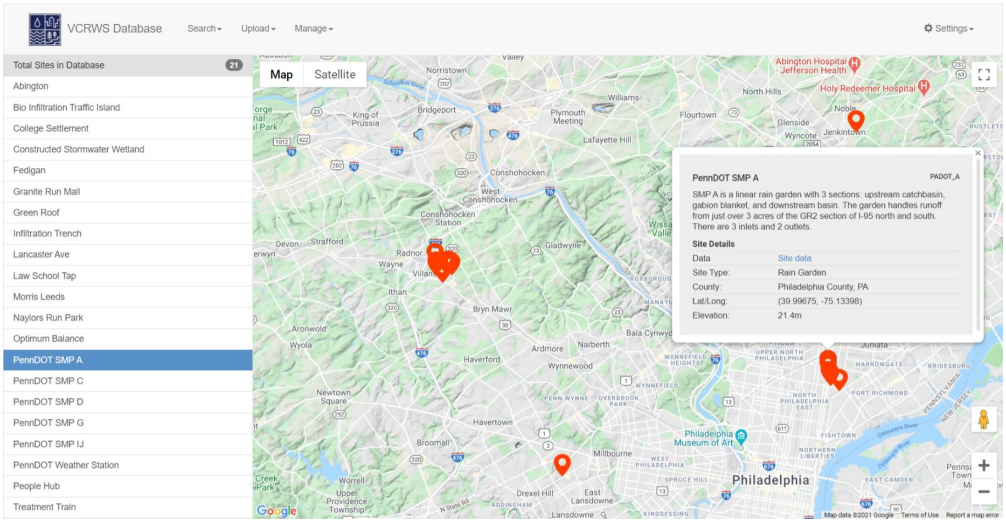

Figure 3.3.: SIDM homepage showing geographical details for SMP A.

The query builder page has filter selectors for each metadata parameter that allow a user to select one or more data identifiers, as well as a date range filter. Once run, the query returns a paginated table of all data requested and gives the option to plot the points in single or multi-axis plots or to download the raw values for analysis in another software package. The query page is also available to outside users who have access through data sharing links so that project partners can easily access the data.

Query performance is fairly quick given the size of data stored in the SIDM (100M+ data values recorded across all sites as of writing). Current performance is reflected in 3.1, and shows a sampling of common queries. The indexing operations required for this improved performance mean that importing data takes longer than it otherwise would without the need to create indices. This tradeoff is appropriate because many more query operations than import operations are expected, so the overall computational time is decreased, and compute time is more expensive than the increase in storage space consumed by indices.

Table 3.1.: SIDM query performance benchmarks (Smith et al., submitted).

Benchmark	Specifications	Performance (seconds to completion)		
		Query	Export to File	Graph + Statistics
Small Dataset	5 Sample Locations, 6 Variables, 900 Data Points, 57.1KB	0.062	8.27	1.11
Large Dataset	2 Sample Locations, 2 Variables, 955,762 Data Points, 65.5MB	3.62	30.51	7.73

3.3 Future Work

3.3.1 Streaming Data Ingest

Currently, the process of extracting, transforming, and loading (ETL) data from the data loggers into the SIDM requires manual input and transfer of files. Currently, each data logger appends data to a separate file on a remote server, which is accessed monthly to check the data, transform it, and load it into the SIDM. The process is time consuming, subject to human error, and means that delays of data reaching the SIDM of a month or more are typical. A better approach would be automatic streaming and loading of the data logger output directly to the SIDM on a close to real-time basis. An update to the Campbell Scientific CR6 data logger's operating system, which is commonly used in VCRWS monitoring programs, enables the use of the message queueing telemetry transport (MQTT) framework. MQTT is an internet-of-things (IoT) system that provides a lightweight method for

sending and receiving data measurements between smart devices and servers (MQTT, 2021). An update to the SIDM could include an API to which the CR6 data loggers can send MQTT messages with up-to-date measurements, greatly enhancing the ability of the platform to automatically ingest new data. A set of rules for each site and each variable could be used to further enhance this process by evaluating each incoming data point against a set of pre-specified criteria to ensure the integrity of the data or send alerts to users if measurements are out of bounds. This ingestion system will have the added benefit of requiring more strict linking of physical sensors with their data stores and the rules used to define valid data immediately upon deployment. Similar data loggers that already have remote connection capabilities could easily have data streaming capabilities enabled via a firmware update.

3.3.2 Programmatic Data Retrieval

The data sharing system of the ODM was removed from the SIDM's feature set to limit the publicity of preliminary data and protect research sponsors who wish to keep their data private. While there are new methods for sharing specific query results, these still revolve around downloading CSV files containing the data. There are no methods for programmatically sharing data or querying the database. Analysis relies on correctly identifying and downloading data prior to beginning analysis, making individual queries difficult to communicate to other researchers.

Modern API design should be used to extend the existing query builder to allow any programming language access to the results of a query through javascript object notation (JSON), a common method for sending large data payloads using the HTTP protocol. Every common programming language (R, python, MATLAB) used in statistical and modeling efforts supports making HTTP requests and handling JSON response data. An analysis script could query this API endpoint with the requirements for a dataset of interest and store the server's response before continuing with the necessary calculations. This would reduce the amount of time required for analysis as a user can directly load data from the server into their analysis program of choice without the need to shuffle data files around, and gathering new data becomes as simple as resending the query request to the server.

3.4 Conclusions

Making data storage FAIR means that data is easily accessible, and analyses performed within a site become easier to transfer to similar sites. The ability to scale analysis across multiple sites will greatly expand the scope of research that is able to be conducted on larger GSI systems. This is accomplished through controlled vocabulary for describing data and consistent, yet flexible, formatting of the database schema. Allowing the addition of new variables at any site or sampling location means that

the database can continue to be expanded with new sites and variables without losing the integrity of the historical data already captured. Furthermore, the increasing accessibility of continuous monitoring through inexpensive IoT devices will mean that future sites are able to be monitored and added to the database easily, greatly expanding the value of consistent formatting and data descriptions. More specifically, it will be possible to understand how individual systems' performance impacts a larger system. Finally, VCRWS' SIDM has made significant progress in accomplishing these goals, as VCRWS data stretching back to 2003 across many sites is now contained in a single format. The SIDM will unlock the potential for expanded collaboration between researchers, better understandings of long-term monitoring results, and more robust analyses of multi-site GSI systems.

Data-Driven Performance Analysis

4.1 Background

Green stormwater infrastructure (GSI) systems' performance has historically been difficult to measure due to poor or nonexistent monitoring and inconsistent performance metrics by which to evaluate systems (Meng et al., 2017). There are many reasons for this lack of consistency, but much of it stems from the wide range of possible precipitation volumes and intensities producing an equally wide variety of responses by these GSI systems. Most systems are designed and constructed using specifications drawn up over the last 25 or more years. The National Pollutant Discharge Elimination System (NPDES) is a common national set of requirements arising from the Clean Water Act (CWA) of 1972 and outlines maximum discharge rates and pollutant loads from certain site types, broken down by activity (industrial, construction, municipal, etc.) (USEPA, 2009). However, NPDES requirements are typically not verified through a post-construction monitoring plan nor are NPDES criteria tailored to be more suitable for the local environment.

There are few well-defined and agreed upon performance metrics that can be used to quantify a GSI's response to storm events over its lifespan (Davis et al., 2012). Infiltration and evapotranspiration capacity of newly constructed systems is occasionally tested for construction acceptance, but these tests are only performed once at discrete randomly selected points within the system and do not convey a wholistic representation of the GSI's capacity (Brown and Hunt, 2012; DelGrosso et al., 2019). Longer-term, whole system monitoring and performance analysis using key performance indicators (KPI) that align with and augment existing lab and field tests have the potential to unlock new insights into and understanding of maintenance needs, construction methodology, and design choices by connecting design specifications to real-world performance (Geberemariam, 2017).

The lack of post-construction monitoring and analysis poses a major roadblock to improving recommendations for the design process that could lead to higher GSI longevity, lower the risk of GSI failure or under-performance, and expedite the creation of uniform standards for GSI comparisons between geographically distinct sites or projects. Even when these monitoring requirements are in place, diverse site conditions, geographies, and climates necessitate a standardized framework for quantifying performance and comparing between potentially vastly different sites, such as the International BMP database, SIDM, and others discussed in Chapter 3. This chapter outlines proposed key performance indicators (KPI) tailored to an infiltration-type rain garden GSI by looking at historical data for SMP A, located in PennDOT's GR2 section of the I-95 Girard Avenue Interchange Stormwater Project (Figure 2.2). These robust monitoring and analysis techniques will lead to consistent results that can be applied across many sites while ensuring that outside factors do not influence performance measurement results. The following KPI measures are intended as derivatives of existing, more complicated lab or field tests that are widely accepted, such as infiltration testing, soil texture tests, soil porosity and bulk density tests, to name a few, but are based solely on data

collected automatically by a the SMP A sensor network. This usage of existing data to corroborate more difficult testing procedures' results will allow easier comprehension of the KPIs and wider transferability. Further, using sensor data will enable cost-effective, sustained monitoring to evaluate performance over time, that more complicated, labor-intensive, and discrete field and laboratory testing programs cannot capture. A standard approach to analysis will open the door to suggested improvements for designs and further exploration of GSI's importance to more sustainable urban environments.

Recession rate, or the change in depth of water ponded over time, is hypothesized to be a potential key performance indicator (KPI) for GSI because it provides an easy to measure proxy for soil health. Soil health, in the context of GSI infiltration, is defined as a lack of compaction, clogging, or other infiltration inhibiting issues (Sokolovskaya et al., 2021). These properties of soil heavily influence hydraulic conductivity and the shape of a soil-water characteristic curve (SWCC) (Figure 4.1). Hydraulic conductivity of soil is the property that defines the ability of soil to pull water from the surface and through the soil column, or the ability to reduce pressure differentials within different sections of soil. The SWCC defines how a given soil responds to varying saturation levels by describing the suction force applied to water in contact with the soil across a range of saturation. Saturation of soil occurs when all the void space is filled with water and the movement of water through the soil column reverts to a gravity driven system that is largely influenced by soil characteristics. Saturated soil generally has the lowest hydraulic conductivity, referred to as K_{sat}, among the range of all volumetric water content (VWC) values possible for that given soil (Eyo et al., 2020). Different soils can have significantly different hydraulic conductivity ranges, and engineered soils with favorable properties (higher saturated hydraulic conductivity) are generally specified for GSI design where infiltration is a desirable treatment method. Infiltration capacity is a function of soil saturation level, and a completely saturated soil will have the lowest infiltration rate, much like it has the lowest hydraulic conductivity. Once fully saturated, infiltration continues at a fixed rate, which is a proxy for saturated hydraulic conductivity. The saturated infiltration rate is hypothesized to be a KPI for system health as it indicates capacity to store water below the soil's surface. This infiltration rate continues even after ponding has ended, as the zone of fully saturated soil extends above the water table level where full saturation is essentially permanent. Significant, long-term water table mounding has not been found to occur as a direct result of infiltrating GSI practices (Tu and Traver, 2019; Machusick et al., 2011), meaning that infiltrating water generally recedes until it reaches the normal water table level. Therefore, infiltration should be expected to continue downward at a constant rate resulting in the steady recession of the saturated zone along a 'drying front' similar to the 'wetting front' seen at the beginning of saturation. The following sections discuss the data collected at SMP A used to support these hypotheses and show that the recession rate and soil drying rate can be used as proxies for soil health and extended to overall GSI performance.

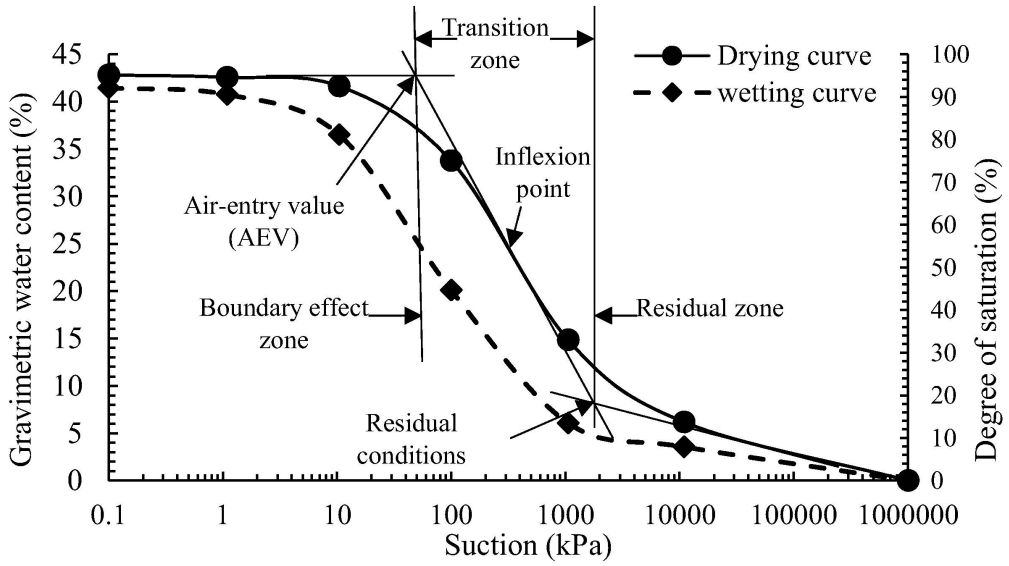

Figure 4.1.: Example SWCC (Eyo et al., 2020).

4.2 Data and Modeling

Data collected was 5 minute averages of 1 minute records (values were measured every minute and stored every 5 minutes as the average of the previous 5 values) over the course of the 3 year study (May 2017 - November 2020). This data resolution captures enough detail without producing an overwhelming number of data records. To better synthesize the data collected, summary statistics for predefined hydrological events are generated based on either a total amount of rainfall followed by a minimum dry time (typically 6 hours), or based on characteristics of other variables of interest, namely ponding and soil moisture values.

4.2.1 Rainfall Events

Using clearly defined rules to identify events is the first step to generating meaningful summary statistics that are applicable across time and space, and useful for comparing GSI with dissimilar properties. In a GSI context, events are storms that result in a measurable quantity of precipitation. Small events typically do not have a pronounced impact on GSI, but the effectiveness of GSI systems is typically greater for smaller or less intense storm events (Liu et al., 2020). However, these smaller events make up the bulk of storm events observed at SMP A and by others (Albright and Schramm, 2018), with 95% of observed total rainfall less than 42mm (Figure 4.2). For this analysis, a ‘standard,’ or ‘normal’ storm event begins when rainfall is first recorded and continues until 6 hours after the last observed rainfall, as studies have shown storms separated by 6 or more hours to be meteorologically separate events (Wadzuk et al., 2017). This means the minimum event

duration is 6 hours, and any periods without rainfall of less than 6 hours are continuations of the same storm event. This definition is used as the standard both for generating storm summary data in monthly reports to PennDOT and for all analyses henceforth that do not directly involve soil moisture observations. During the period of interest, there were 414 observed standard events with a mean duration of 12.2 hours, a median of 9.5 hours, and a standard deviation of 7.5 hours. The data are highly right-skewed, with a maximum of 52.2 hours, and appear to follow a roughly chi-square distribution (Figure 4.3).

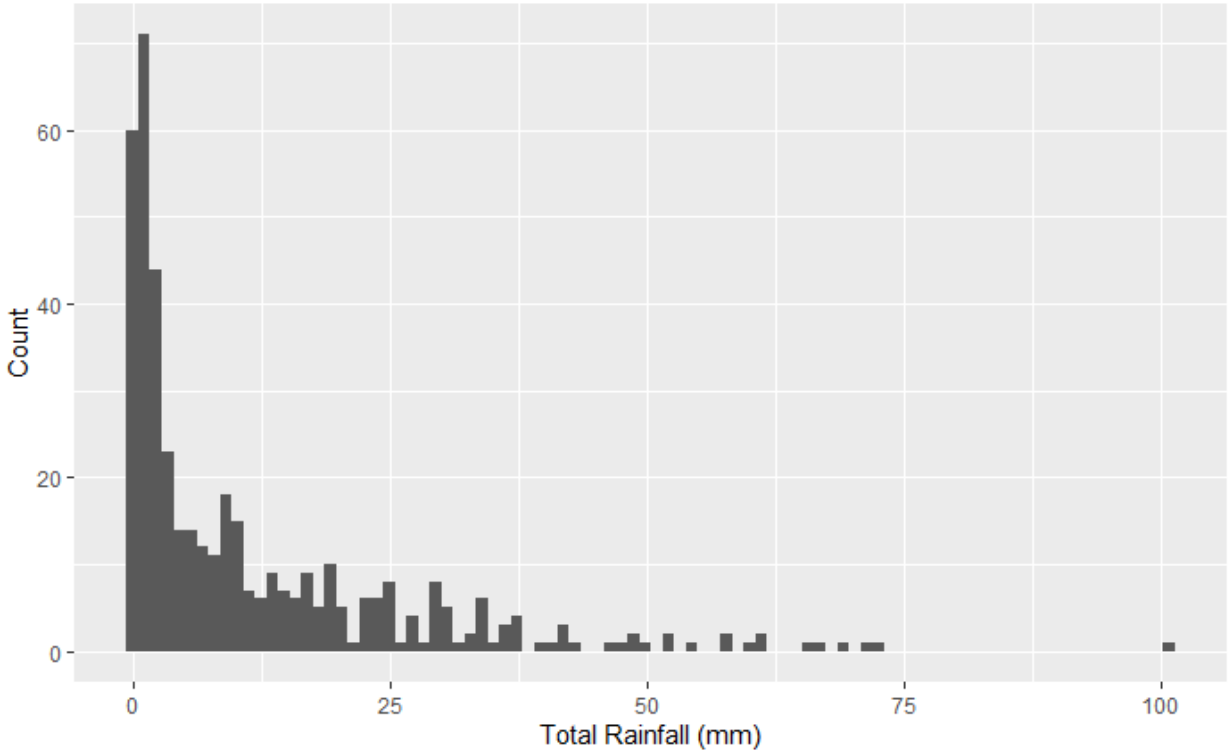

Figure 4.2.: Distribution of event total rainfall depths.

Events are distributed throughout a Gregorian 365-day year approximately uniformly, as seen in Figure 4.4, although there are some notable event clusters in both late spring (around ordinal day 150), and during hurricane season (after ordinal day 200). There are notably fewer events during the winter months, but this is likely due to the instrumentation's lack of sensitivity to snowfall. Snowfall events are also less taxing on GSI systems in general, although snowmelt events may have similar characteristics to a rainfall event but have not been studied.

4.2.2 Soil Moisture Events

Events involving soil moisture require a modified approach to event definition due to the prolonged nature of soil moisture's response to storm events. Soil moisture, and by extension ponded water

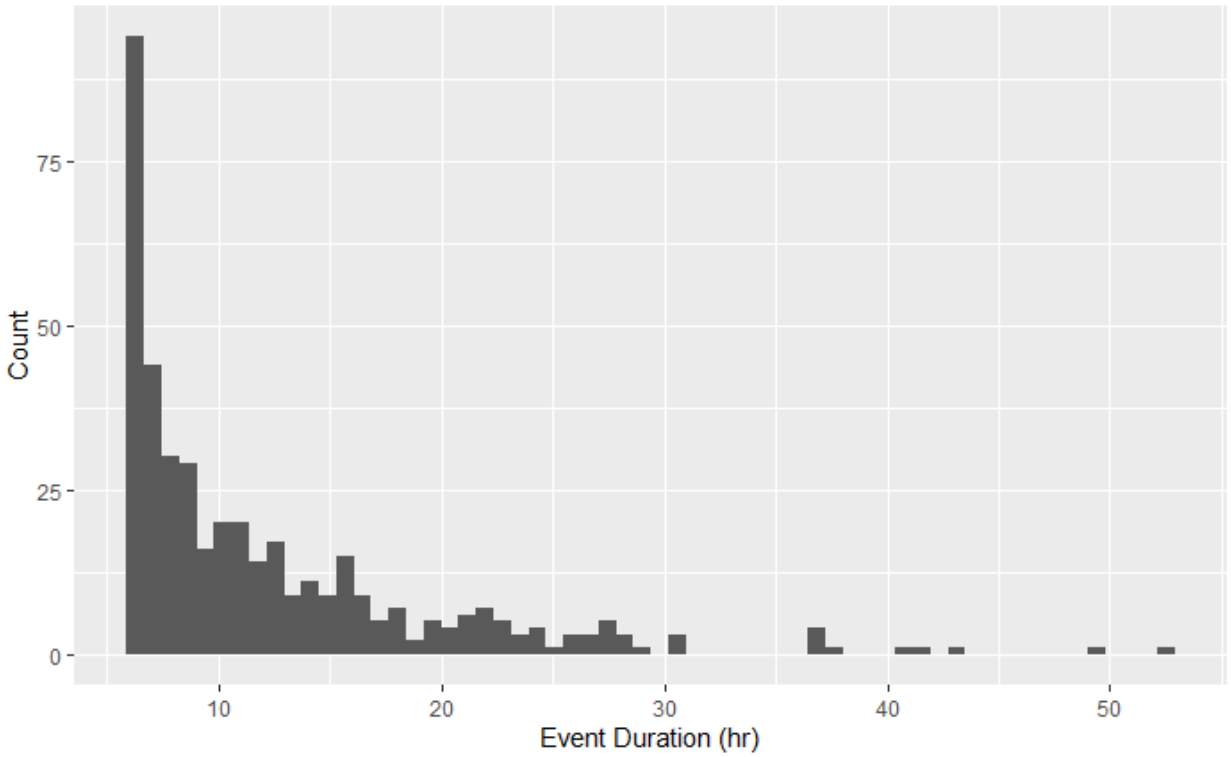

Figure 4.3.: Distribution of event lengths.

atop saturated soils, take longer than 6 hours to recover after storms of interest. Studies have shown that the recovery period for soils is 72-96 hours (Davis et al., 2009; Wadzuk et al., 2017). To define events for these analyses, both rainfall and soil moisture levels are considered. As before, an event begins with the first observed rainfall. However, the end of the event is defined by a user-defined minimum event time (6 hour default) as well as a threshold ponding level or soil moisture level (either may be used, depending on the type of analysis being performed). A simple moving average of the number of steps comprising the minimum event duration (in other words, the data interval - 5 minutes - divided into the minimum event time of interest) provides a window into the past or the future at a given time step, and can be used to define an event as follows:

$$\begin{aligned} Event = \{ & A = [moving\_average(rainfall, Q/P) > 0], \\ & B = [moving\_average(rainfall, Q/P) == 0 \\ & \cap\, moving\_average(K, Q/P) < R] \} \end{aligned} \tag{4.1}$$

where A is an indicator for the beginning of an event, B is an indicator for the end of an event, Q is the minimum event duration, P is the observation interval, R is a threshold value, and K is the variable of interest. Timestamps falling between successive A and B values are assigned a unique, random string of eight characters generated from the first timestamp in the series. The distribution of these specialized events is similar to that of the standard events above, although there are fewer (214) of them due to the increased length of events capturing a greater share of rainfall per event.

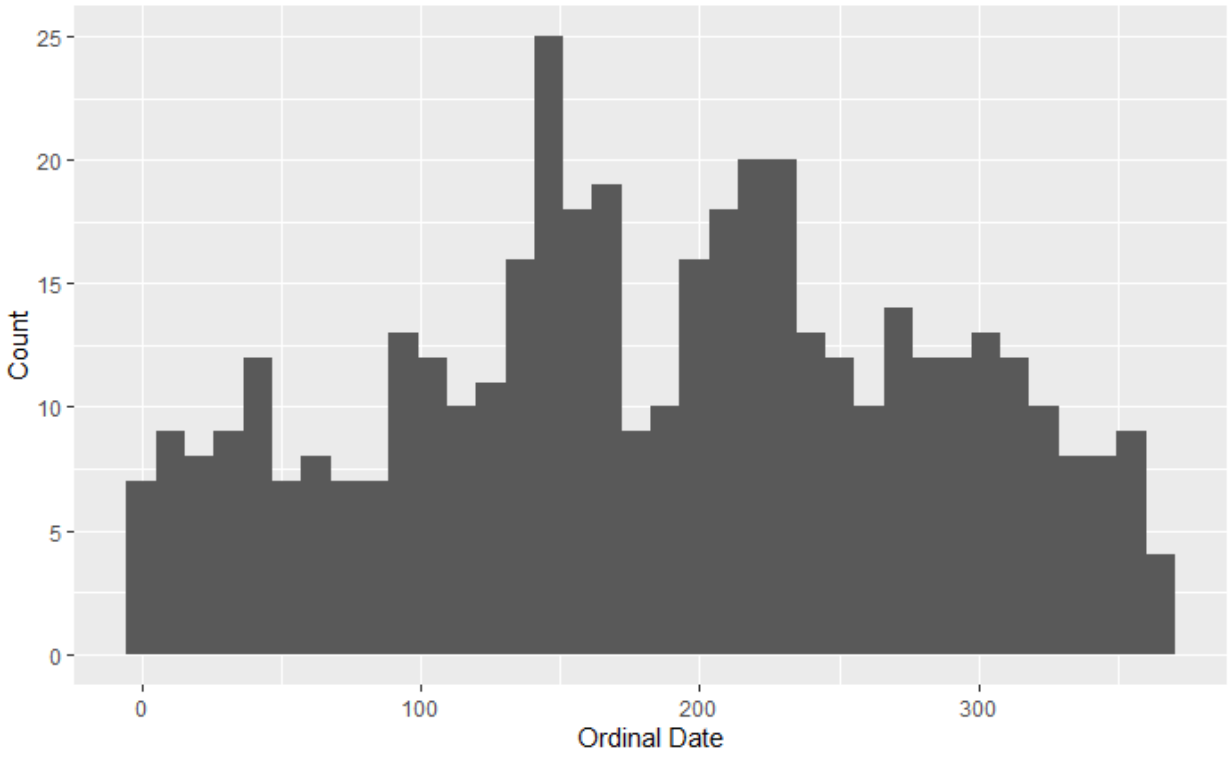

Figure 4.4.: Distribution of event ordinal dates.

4.3 Key Performance Indicator Definitions

4.3.1 Recession Rate

Calculating a water level differential across consecutive data records can be used to determine periods of increasing or decreasing volume in the ponded storage. Recession of the pond is indicated by a negative change in level, although no rainfall or inflow can happen simultaneously to ensure that change in level is solely due to outflow from the system (i.e., infiltration or overflow). Additionally, if the recession rate due to only infiltration is desired, the water level must be below the overflow point of any outlet structures. The magnitude of recession is directly correlated to infiltration rate, which is governed by the type and health of the GSI soils (Gregory et al., 2006; Horton et al., 1994). Larger recession rates indicate a faster drawdown of the water level, while smaller values could be due to prolonged saturation conditions or under-performance due to compaction or clogging of soil pores. Comparing average recession rates calculated over the duration of a storm to other simultaneously recorded atmospheric (temperature, relative humidity, barometric pressure) and GSI state (current pond level) data shows the relationships that have the most significant impact on GSI performance. Changes in the recession rate over the period of a storm or over long periods of time (e.g., seasonal, annual, etc.) indicate changes in the soil health (Jenkins et al., 2010; Zukowski et al., 2016).

The relationships between recession and atmospheric variables or GSI state are complicated by several factors, namely the timing and size of an event, the pre-event geomorphology and hydrologic state of the GSI, and their collective interaction. The timing and size of a storm event, which can be best described by a combination of time of year, hyetograph, and the length of time since the previous storm event, are important because these play the largest role in determining the GSI pre-storm state's response to specific storm conditions. A GSI system will respond differently at different times of the year to two events with identical hyetographs, while the same GSI with identical starting conditions will respond differently to two different hyetographs (Braga et al., 2007; Joshi et al., 2019). Varying any single variable used to describe an event changes the GSI system's initial response to that event.

The variability of the pre-event state of the GSI necessitates an adaptable baseline to use in evaluating performance. Performance cannot simply be stated as the change from pre-event conditions because this would make comparisons between dissimilar pre-event states unnecessarily complex. For example, a storm taking place during early spring in the Philadelphia, PA region could have a wide range of pre-storm soil moisture, air and water temperatures, or plant growth conditions, to name a few, all of which will impact how the system responds at the beginning of a storm. Additionally, the fact that suction head is highest when the soil is dry (Eyo et al., 2020, Figure 4.1), means that infiltration loss is greatest at the beginning of an event, no matter the specific initial conditions, because that is when the soil is driest. This inverse relationship indicates that the most consistent method of comparison will be when the system reaches equilibrium and is treating water at a constant rate. For recession rates, this means looking at the trailing end of a storm, when inflow has ceased, ponding level has reached its peak, and the soil has reached saturation. Examining the end of a storm removes the influence of a specific storm event's timing characteristics and focuses the performance evaluation on how the system reduces ponded volume and recovers from saturation. Full infiltration of ponded water is required within 72 hours in Philadelphia. Observations show that the majority of storms have complete infiltration of the ponded water within 6 hours post-rainfall.

4.3.2 Infiltration Drying Rate

The recession of water within the soil column also reflects the soil health of the site. Using soil moisture probes at 10, 35, and 60 cm allows the calculation of an average rate of the descent of saturation conditions, or 'drying front', as the soil capacity recovers. This subsurface drying rate should remain consistent between events of different sizes (Wadzuk et al., 2017), provided the soils reach saturation prior to the end of measurable ponding. Soil moisture values during a storm event typically produce a curve like that in Figure 4.5. The soil moisture curve tends to respond with a sharp upward jump, 5-20 minutes in length, at the beginning of the storm event's inflow when the wetting front passes each sensor and soil approaches full saturation. This increase in saturation coincides with a sharp drop in suction head, as the soil's ability to pull water in quickly is greatly

reduced once saturated. The soil moisture curve has been characterized with a series of seven points in work completed by Matina Shakya, a Ph.D. Candidate working on soil moisture measurements for VCRWS/PennDOT. The beginning and end of this increase in soil moisture at the inflow onset are identified as points A and B, respectively. Point A is typically below 0.40, or 40% volumetric water content (VWC), while points B and C are typically between 0.45 and 0.50, or 45-50% VWC. Soil moisture levels remain at saturation conditions for the duration of ponding, due to continuous infiltration. The reduction in volume from the system due to evaporation, evapotranspiration, or any other means is assumed to be negligible compared to infiltration during a storm and in the hours immediately after. When ponding has ended, infiltration continues as the saturated zone recedes down into the soil profile (Figure 4.6). The trailing limb of the saturation curve can be identified by two additional points, C and D (Figure 4.5), identified as the drop off point and inflection point of the soil moisture curve, respectively. In Figure 4.6, the difference in time between steps 2 and 3 at each sensor depth (10, 35, 60cm) is the subject of interest.

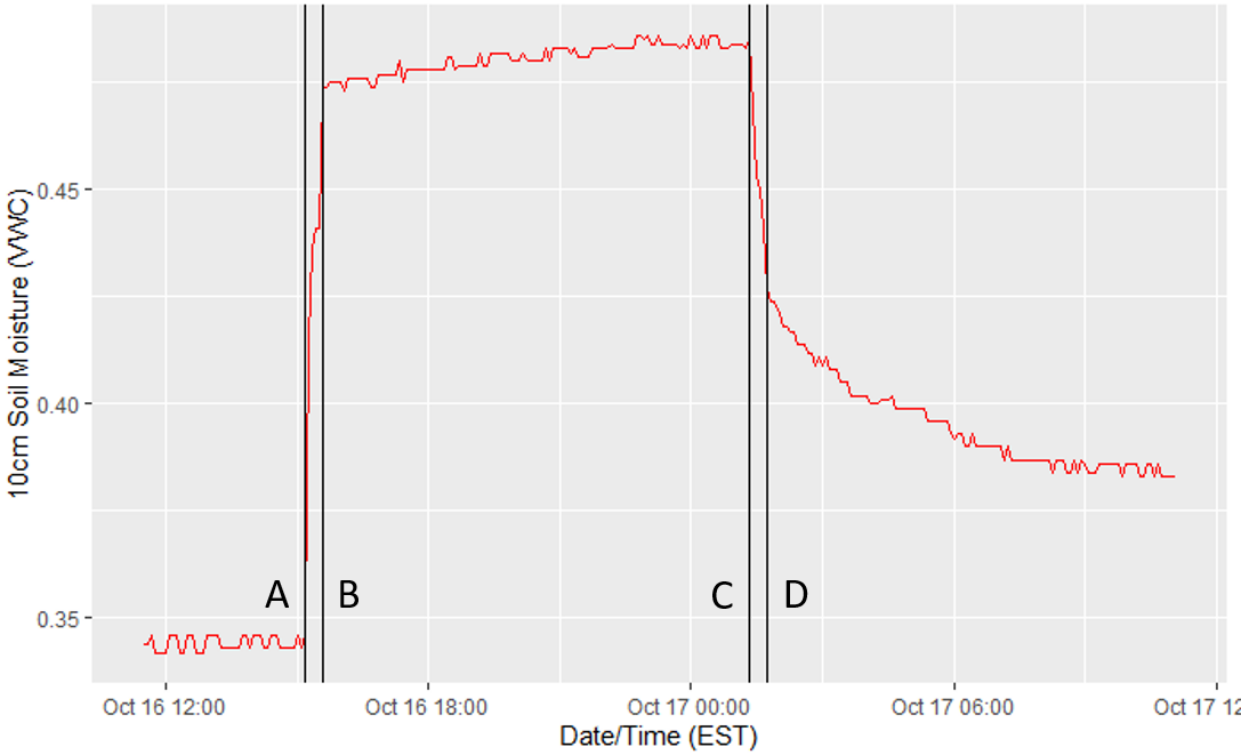

Figure 4.5.: Typical soil moisture curve. Vertical black lines denote points A, B, C, and D from left to right, respectively.

The calculation of this metric is slightly more complicated than raw recession rate of the ponded water presented earlier. The inflection points A, B, C, and D are calculated as the four primary peaks in the absolute value of the first differential of the soil moisture data:

$$InflectionPoint\{A, B, C, D\} = localmax(|\theta_i - \theta_{i-1}|)_j \tag{4.2}$$

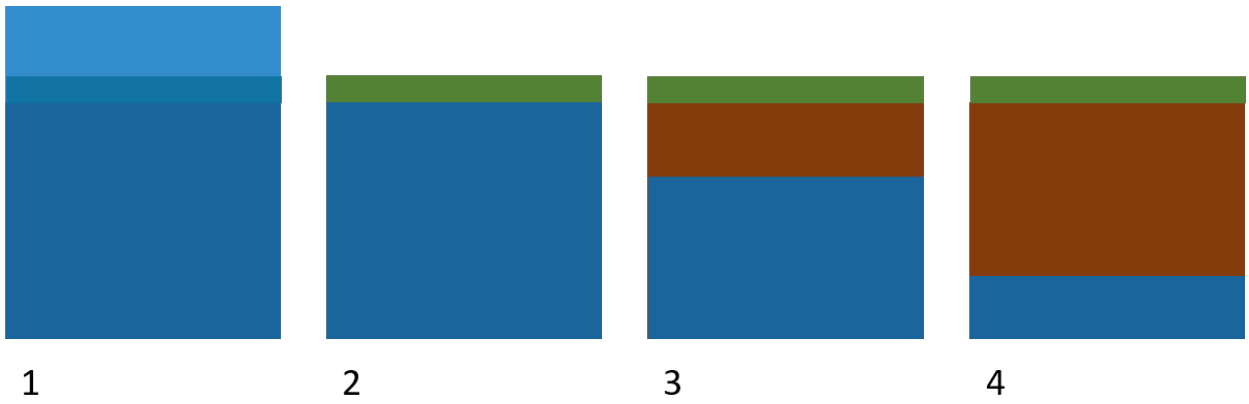

Figure 4.6.: Illustration of infiltration drying rate showing 1. ponded water, 2. end of ponding, 3. top-most soil returns to field capacity, 4. saturation zone reaches equilibrium with water table.

where i = 1,2...n for n soil moisture observations (θ) in a given storm and where j = 1,2,3,4 corresponding to A,B,C,D. These points are easily calculated using the 'findpeaks' function from the R package 'pracma' (`pracma::findpeaks(...)`). The function takes a vector of data, the absolute value of the differential soil moisture data applicable to one storm event in this case and returns the ordered indices of the requested number of local maximums (four in this case). Depending on the specific values of soil moisture seen during an event, the four indices may be returned in any order, largely based on the magnitude of difference between any two given points, so the resulting points are sorted and assigned to A through D respectively. However, due to the highly variable pre-event state of the GSI, storm events that begin with saturated VWC values (>0.44 VWC) often misidentify some or all of these points, so these events have been disregarded in the context of this analysis.

4.4 Results and Discussion

4.4.1 Recession Rate

Temperature Dependence

The weather station at SMP A records air temperature, and each pressure transducer (PT) records the temperature of surrounding water. The two types of temperature are highly correlated (Pearson correlation = 0.981). This analysis only considers water temperature since the water is in direct contact with the soil and displays some variance of viscosity at different temperatures. Testing for variations in recession rate across the range of observed water temperatures shows a linear relationship (Figure 4.7), as evidenced by the high correlation coefficient of 0.735 and R^2 of 0.54.

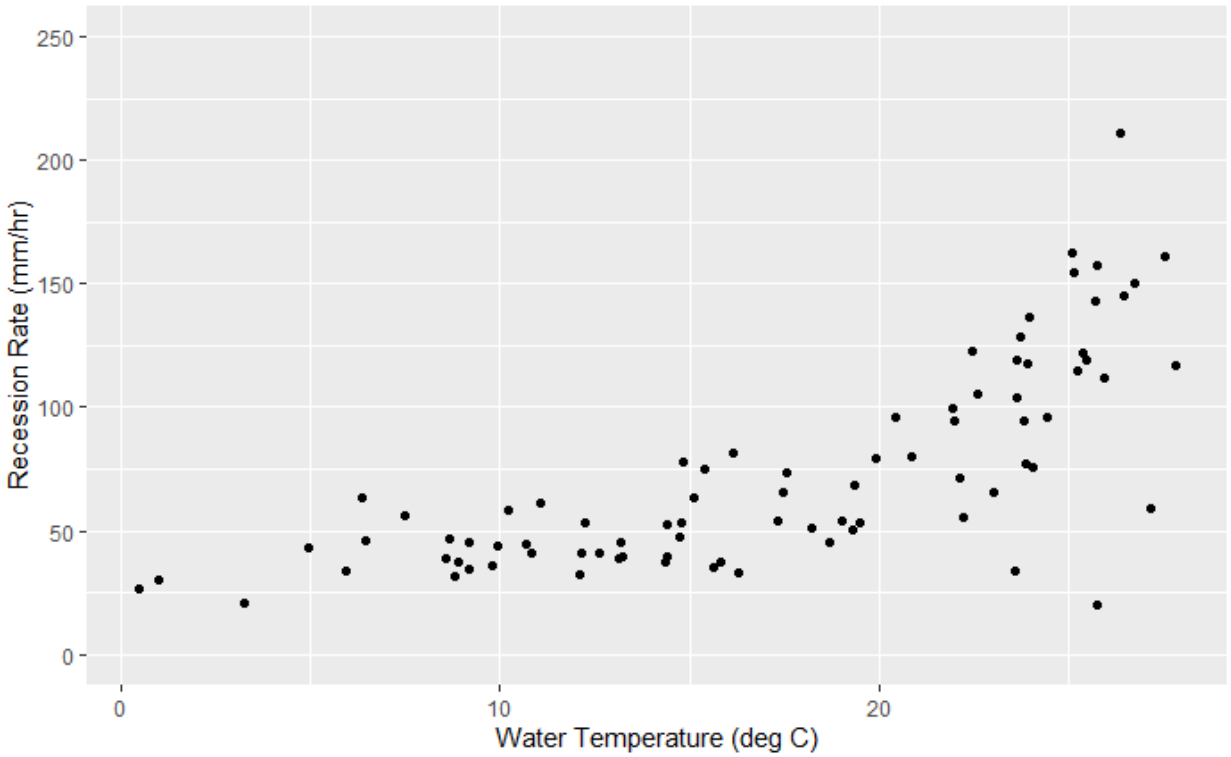

Figure 4.7.: Average water temperature vs recession rate for SMP A storm event.

The relationship between recession rate and ponded water temperature becomes more apparent when temperature is separated into similar temperature range bins (Figure 4.8). Five bins were chosen to ensure the number of events per bin was at least 10 for most bins. The trend of higher observed recession rates at higher temperatures is clear. The lowest bin ((-0.05,5.83]) has only seven events but follows an identical trend to the upper four bins in magnitude of mean and quartiles (indicated by upper and lower vertical lines). This trend aligns with several physical and hydrologic models, namely that warmer water is less viscous and flows easier, especially through the soil (Emerson and Traver, 2008). This finding means that GSI performance can be expected to increase during warmer months, both due to increased infiltration capacity and increased water uptake and evapotranspiration by the plant mass (Bartens et al., 2008). Furthermore, it has been shown that soils that have experienced multiple freeze-thaw cycles have higher values of saturated hydraulic conductivity due to the formation of internal ice crystal structures within frozen soils, which can expand pore space and increase the "development of macroscopic cracks and microscopic voids" (Asare et al., 1999). Therefore, soils that experience freezing conditions followed by warm conditions, as is typical in a climate with a distinct harsh winter season, can expect to see a natural "loosening" of the soil via the creation of these additional void spaces. This increase in void spaces has the potential to cause preferential flow paths through the soil, which will lead to increased infiltration rates up to the frost line.

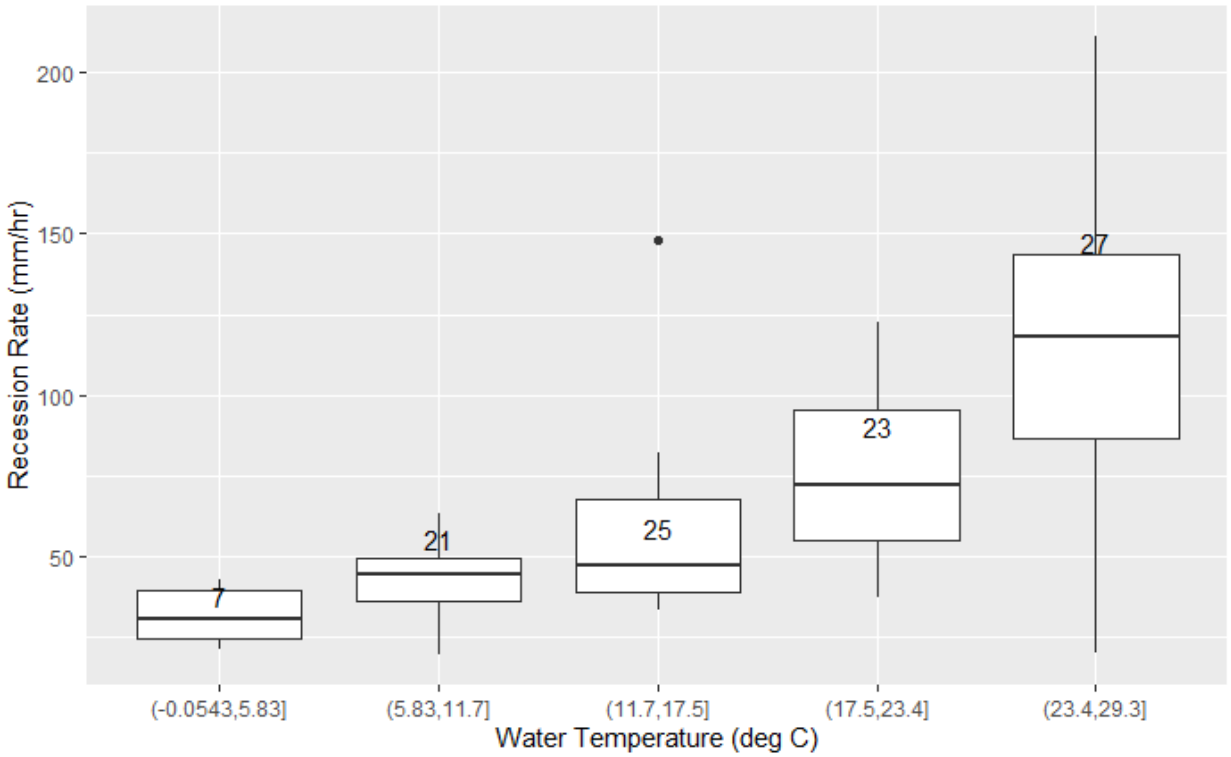

Figure 4.8.: Average water temperature vs recession rate, separated into bins, with the number of storm events per bin indicated.

Insignificant Relationships

Both relative humidity (RH) and barometric pressure, recorded in a similar fashion to air temperature at SMP A, showed no significant relationship with recession rate, with correlation coefficients of 0.148 and 0.25 and R^2 values of 0.022 and 0.064 respectively. Comparing the average recession rate to RH bins and pressure bins shows no correlation between different values for these variables as compared to the average recession rate (Figure 4.9 and 4.10).

From these observations, GSI performance is not dependent on barometric pressure or relative humidity, which largely only impact the soil-air interface. The lack of relationship between the atmosphere and sub-surface infiltration mechanics is expected, since the soil contains enough mass to prevent significant influence from the relatively fast-changing atmosphere.

The level of ponded water collected in the rain garden similarly has no significant effect on the rate of recession, with a correlation of -0.2 and R^2 of 0.043. This runs contrary to expectations that greater head pressure at the soil-ponding interface would lead to higher recession rates. The rate of recession is consistent across nearly the entire ponding depth range (Figure 4.11). The lack of correlation, while surprising, could be due to the fact that sub-surface saturation leads to high enough pore water pressure such that it cancels out the head pressure seen at typical ponding depths of up to a maximum of less than 1 meter. Additionally, the shallow profile of the garden bed means

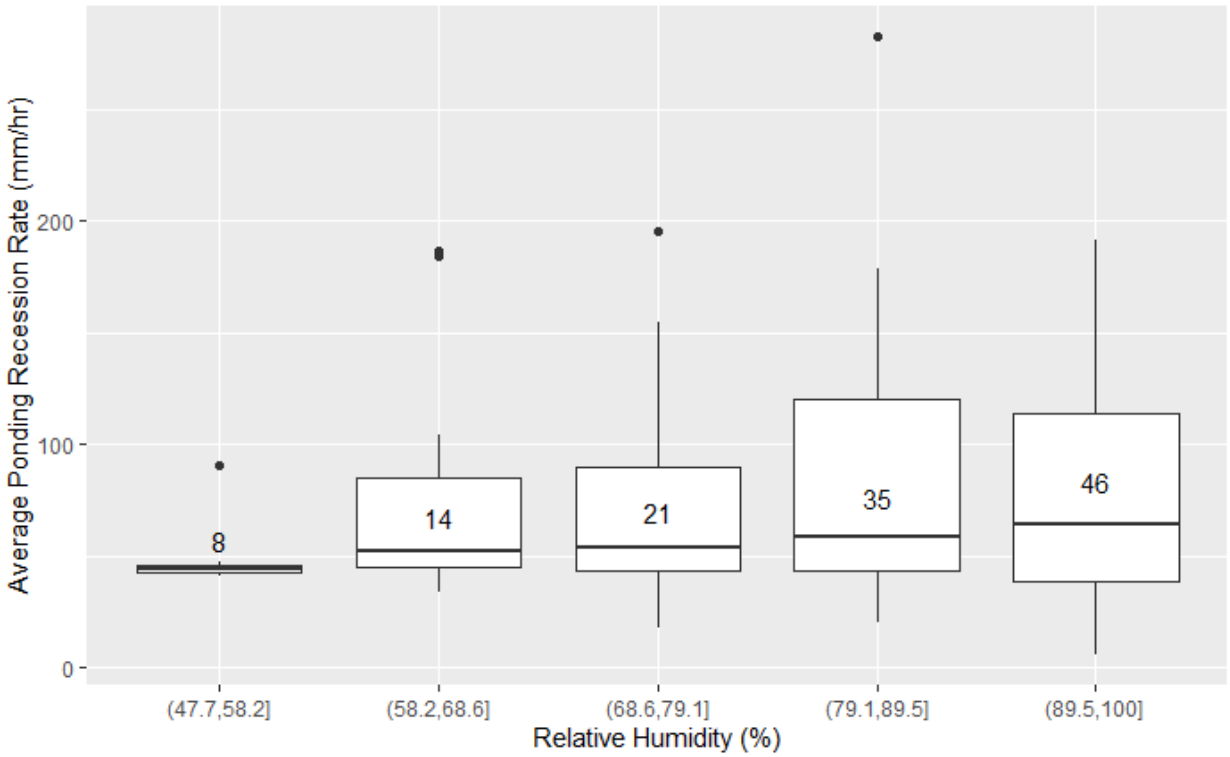

Figure 4.9.: Average Relative Humidity vs Recession Rate.

that a depth measurement taken in the middle of the garden captures the deepest point, and less significant depths are experienced by the majority of the surface area (Sokolovskaya et al., 2021). While the slope estimate and F-statistic in a linear model for recession rate and pond depth are both significant at the 0.05 level, the low R^2 value means the model is only predicting the mean recession rate.

Regression

Using multiple linear regression, a numeric relationship between the recession rate and pond temperature, max pond depth, ponding duration, total rainfall, and average rainfall intensity can be determined. The expected value of this relationship can determine if a GSI system is functioning nominally. Values that are outside the expected relationship boundaries are a cause for concern, and a trend of values outside the norm would indicate that maintenance or further evaluation of the system is required. While the numbers presented here are specific to PennDOT's SMP A, this approach, and indeed the entire scope of data collection, storage, and analysis would be of tremendous value to any system or group of systems with similar properties.

Regressing the following parameters onto ponding recession rate (mm/hr) results in the listed effects (Table 4.1), which results in Equation 4.3. The R-squared value is 0.9023, with an F-statistic

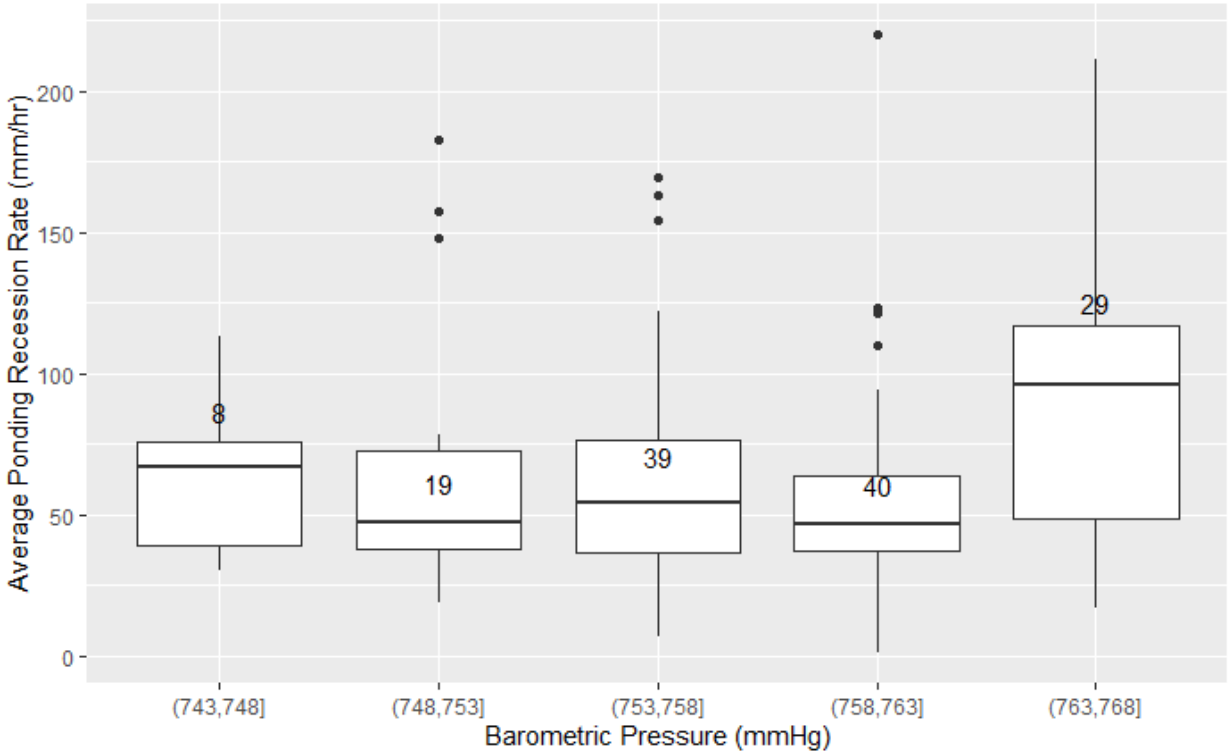

Figure 4.10.: Average Barometric Pressure vs Recession Rate.

(significance test) of 312.1 on 5 and 169 degrees of freedom (p-value = $< 2.2e-16$), indicating the partial slopes ($\beta_1 \ldots \beta_5$) are not equal to 0. There are, however, normality concerns, as demonstrated by the normal quantile-quantile plot (Figure 4.12), which shows a departure from linearity beyond ± 1.5 quantiles, suggesting that there have been a number of extreme observations. This is expected, as the data are not normally distributed, so an additional nonparametric test is required to confirm the trend seen. The relationship lacks an intercept because the mean response is expected to be 0 mm/hr, corresponding to no ponding recession when all other variables are also 0.

Table 4.1.: Ponding recession rate regression results.

	Coefficient	Estimate	Std. Error	t value	Pr($>\|t\|$)
β_1	B1 pond mean temp (°C)	-0.238	0.085	-2.801	0.005697
β_2	B1 pond max depth (mm)	-0.117	0.006	-20.622	<2e-16
β_3	B1 ponding duration (hr)	0.981	0.225	4.361	2.24e-05
β_4	Total Rainfall (mm)	0.182	0.064	2.846	0.004980
β_5	Average Intensity (mm/hr)	-0.377	0.110	-3.416	0.000797

$$y = -\beta_1 * 0.2384 - \beta_2 * 0.11721 + \beta_3 * 0.98104 + \beta_4 * 0.1821 - \beta_5 * 0.3771 \tag{4.3}$$

The regression model in Equation 4.3 was chosen for its high R-squared value, which reflects the percent of variation in the response (recession rate) explained by the total variation in the model

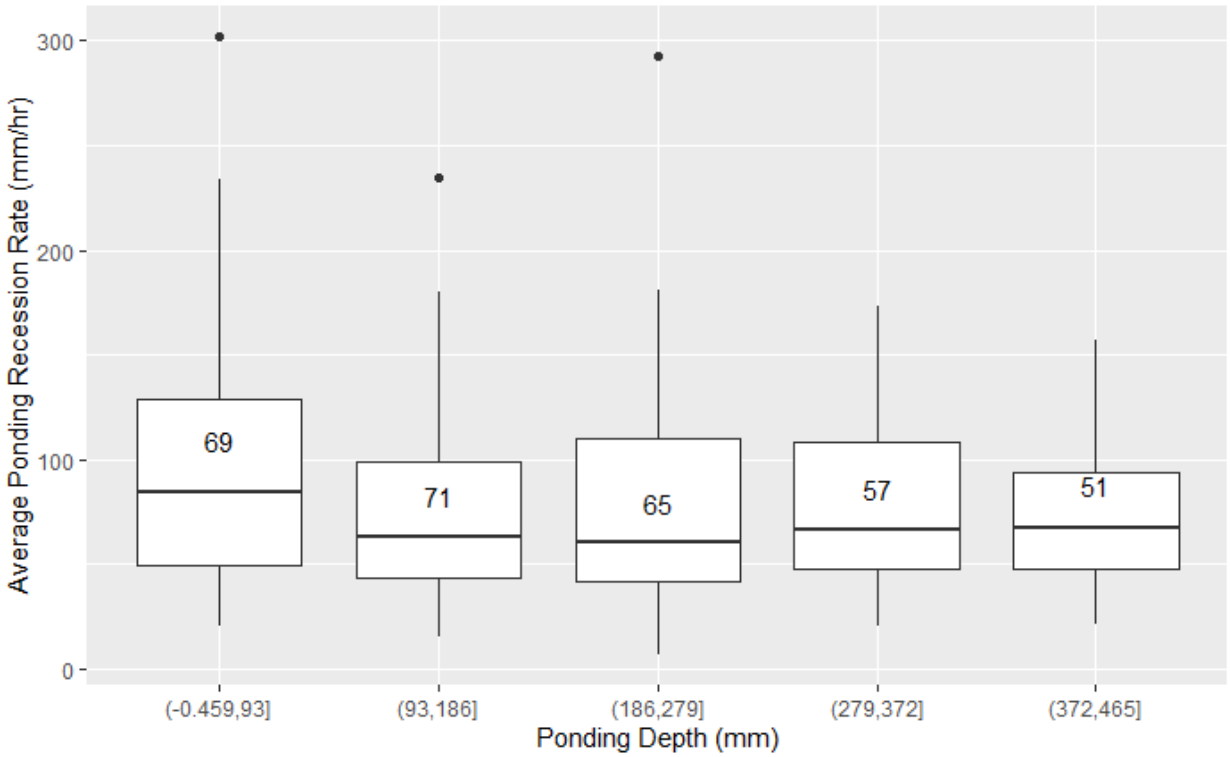

Figure 4.11.: Average Ponding Depth vs Recession Rate.

parameters, as well as its overall significance (high F-statistic with low p-value). Parameters kept in the model (listed in Table 4.1) have individual p-values < 0.05. Estimates of recession rate predict the mean response for each unit change of the model parameters multiplied by their respective coefficients. Other models considered included some combination of the final parameters, water temperature, event duration, and ordinal day. While the equation predicts the mean response given a set of inputs and will not always be precise, a confidence interval of 95% can indicate values that are outside the expected range. Consecutive events with abnormally high or low recession values will indicate a need for inspection or intervention.

The model is not without its shortcomings, however, and does not account for departures from normality of the training data. The Kendall seasonal trend statistic, τ, tests for seasonal trends in monotonic, nonparametric data. The τ statistic is a ratio of the probabilities of the observed order of the data to that of a different ordering. That is, assuming two variables (X and Y, for example) are independent, the expected value of τ is 0, as the likelihood of the observed ordering is equal to that of any other ordering for two independent series Abdi, 2007. Calculating τ for comparison between B1 recession rate and the variables found in the regression equation 4.3 results in the values found in Table 4.2. All the relationships are significant enough to reject the claim that the pairwise comparisons are independent, suggesting the trends observed, and by extension their inclusion in a normal multiple regression, are significant, despite the lack of normality.

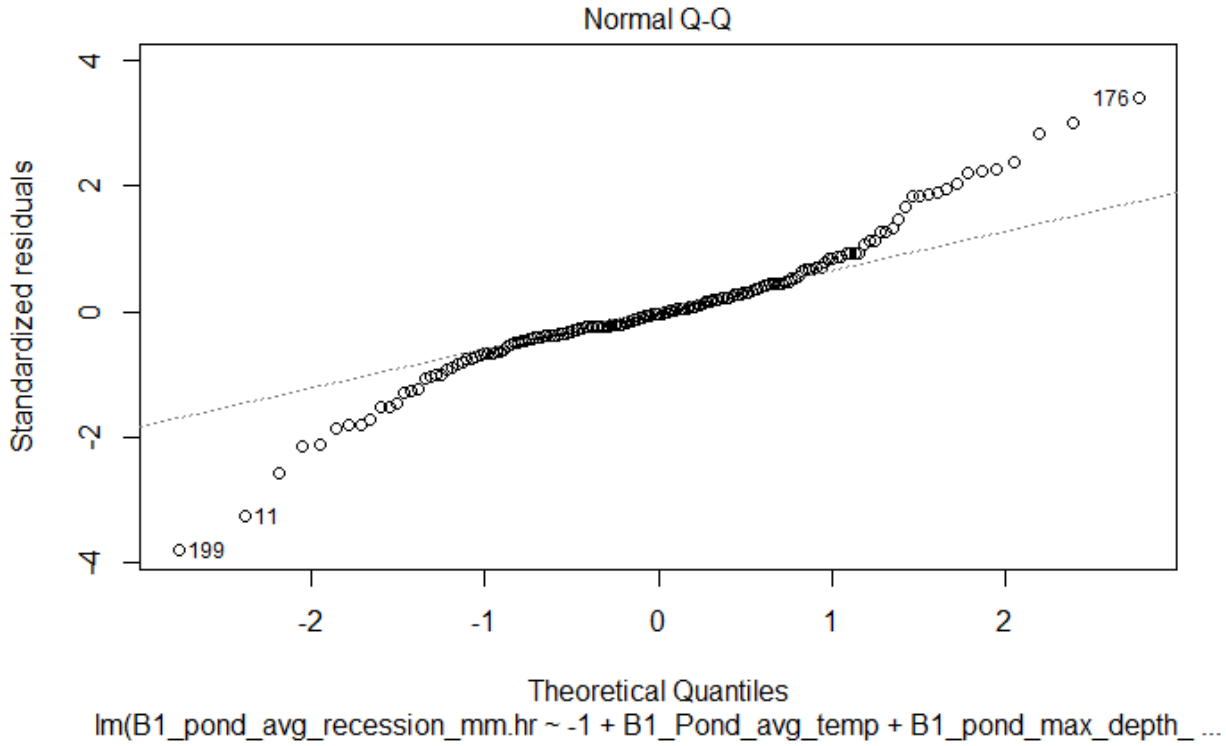

Figure 4.12.: Normal Q-Q plot for ponding recession regression.

Table 4.2.: Kendall seasonal trend results.

Variable	Kendall τ	p-value
B1 pond mean temp (°C)	0.122	0.001711
B1 pond max depth (mm)	0.915	<2.2e-16
B1 ponding duration (hr)	0.829	<2.2e-16
Total Rainfall (mm)	0.548	<2.2e-16
Average Intensity (mm/hr)	0.410	<2.2e-16

4.4.2 Infiltration Drying Rate

There are 76 valid events during the period of interest, essentially a subset of the standard events, quantified with different ending criteria to focus on soil moisture recovery. For each event, the rate of drying for the three zones between the surface and the three soil moisture sensors at progressively deeper placements is calculated as distance/time. Figures 4.13, 4.14, and 4.15 demonstrate that this rate can be expected to fall within 0-5 cm per hour. The 95% percentile falls at 9.5 hours between 10 and 35cm (approx 2.63cm/hr), and 7.4 hours between 35 and 60cm (approx. 3.37cm/hr), meaning 95% of events can be expected to exhibit drying rates faster than this. The distribution does not display correlation with water temperature (Pearson correlation 0.047, p-value 0.34), and has remained consistent over time, as displayed in Figure 4.16.

The infiltration drying rate compares favorably to Saturo constant-head infiltration testing performed at the site across three growing seasons (repeat tests performed in downstream and upstream basins at discrete locations). These tests showed a mean K_{sat} value of 1.3-6.2 cm/hour, which compares favorably with the mean infiltration drying rate of 1.8-9.5 cm/hour. Both metrics have large variances, and it would be helpful to do more direct comparison testing to investigate if the variability is consistent between them.

4.5 Conclusions

The methods described here attempt to evaluate methods for comparing storm events across time and space. Using recession and infiltration KPIs to evaluate a system of GSI performance across an entire region will enable insights into long term design successes, and early detection of system errors. The data have shown that performance is most highly impacted by temperature, which oscillates with an annual seasonal period. GSI health can generally be quantified in terms of recession and drying rates for ponding level and soil moisture, respectively. Higher rates for each quantify indicate faster transfer of water into the subterranean water table, and faster reduction in soil moisture that will lead to quicker preparation for the next storm.

In general, average recession rates between 40 and 120 mm/hr can be expected, depending on time of year and water temperature. Average drying rates as high as 9.5 cm/hr can be expected, although this value would be expected to vary from GSI to GSI based on soil design criteria. These statistics are not dependent on GSI parameters such as surface area, loading ratio, or number of inlets or outlets, and are therefore independent of site configuration. The statistics can be calculated for any GSI system, or part of a system, to determine if infiltration rates and recovery rates fall within the expected range. Thanks to the few parameters needed, namely ponding depth and soil moisture level, few sensors are necessary, keeping costs for this kind of monitoring low, making it accessible to be deployed at a larger number of sites, or at multiple locations within a single site. While this work intentionally does not address spatial variability within a single site for infiltration or drying rates, the data collected are assumed to represent the average conditions throughout SMP A, and the same assumption would be valid for the statistics calculated at other sites in a similar manner.

The recession and infiltration rates are affected by conditions both above and below the soil surface and provide an acceptable proxy for soil health. The recession rate is shown to be affected by temperature and could be reduced by clogging of the soil surface or compaction of lower soil layers, leading to reduced rates. Similarly, the infiltration drying rate is shown to be consistent across a wide variety of GSI conditions and could be adversely affected by changing soil properties that make the GSI function less efficiently. The established relationships and expected values for each are useful as indicators for long term performance, and departures from expectations are cause for further investigation.

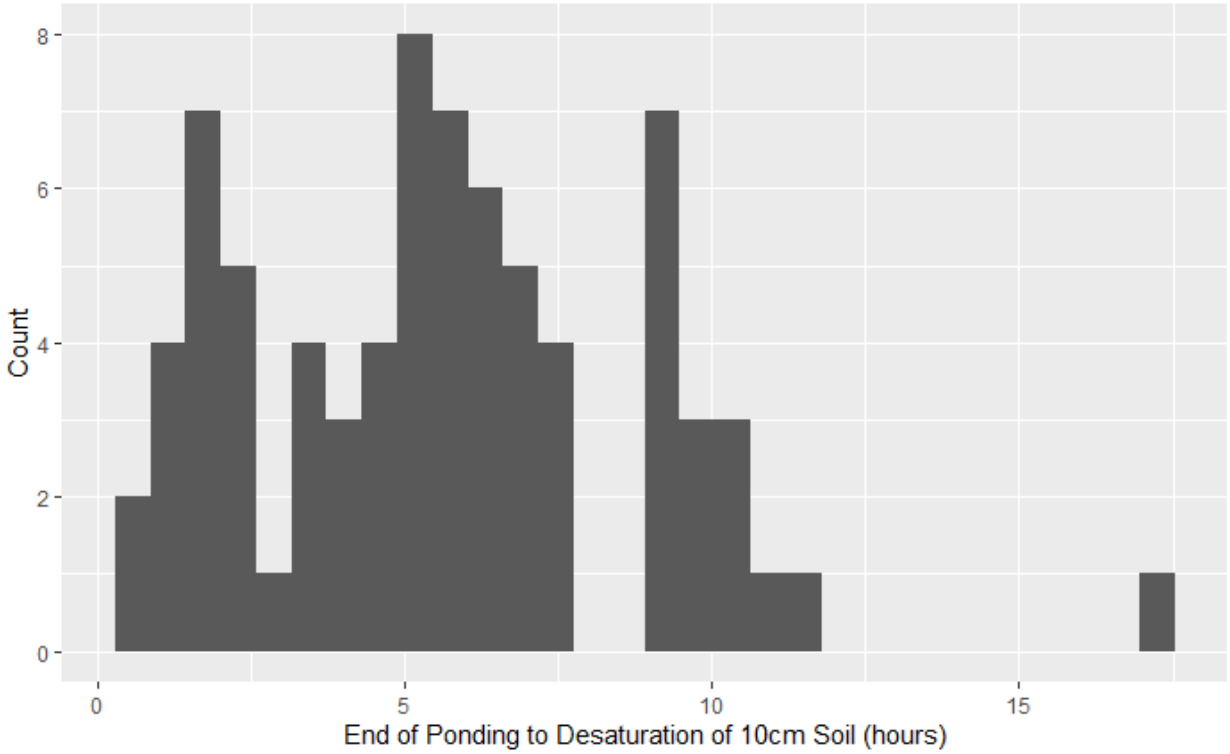

Figure 4.13.: Time to Desaturation from end of ponding to 10cm.

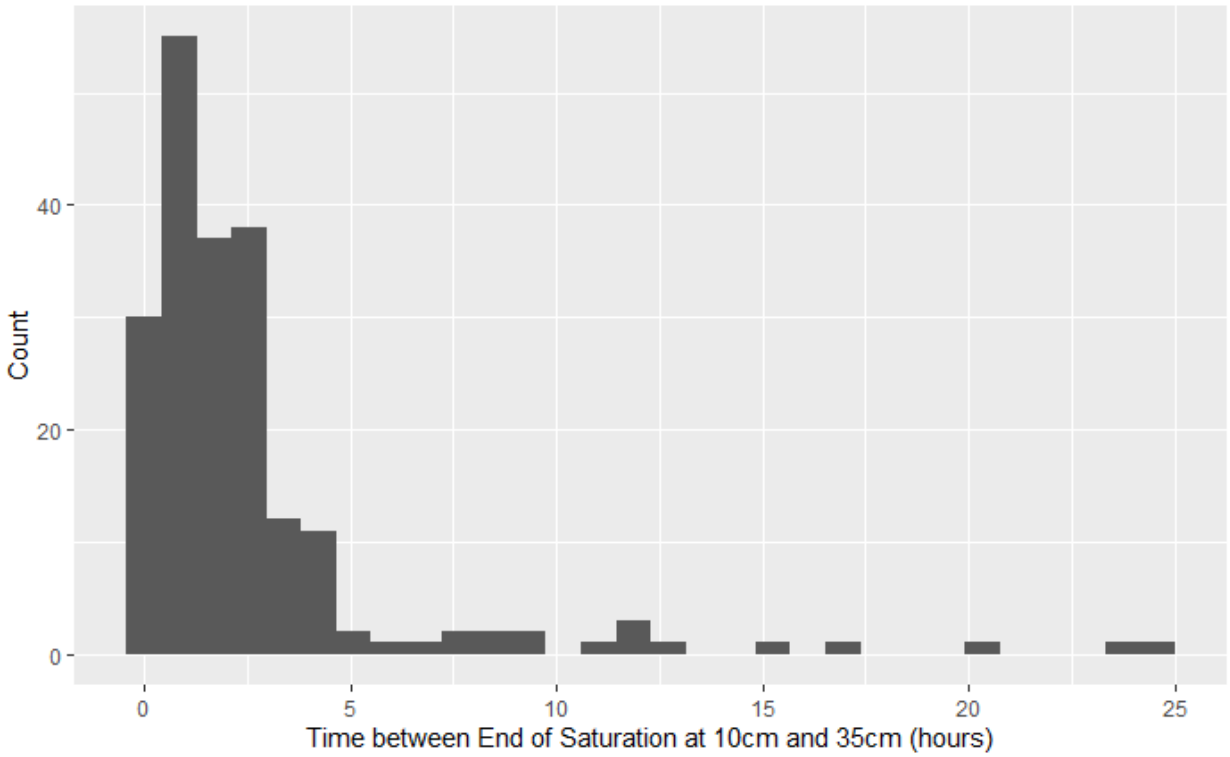

Figure 4.14.: Time to Desaturation from 10 to 35cm.

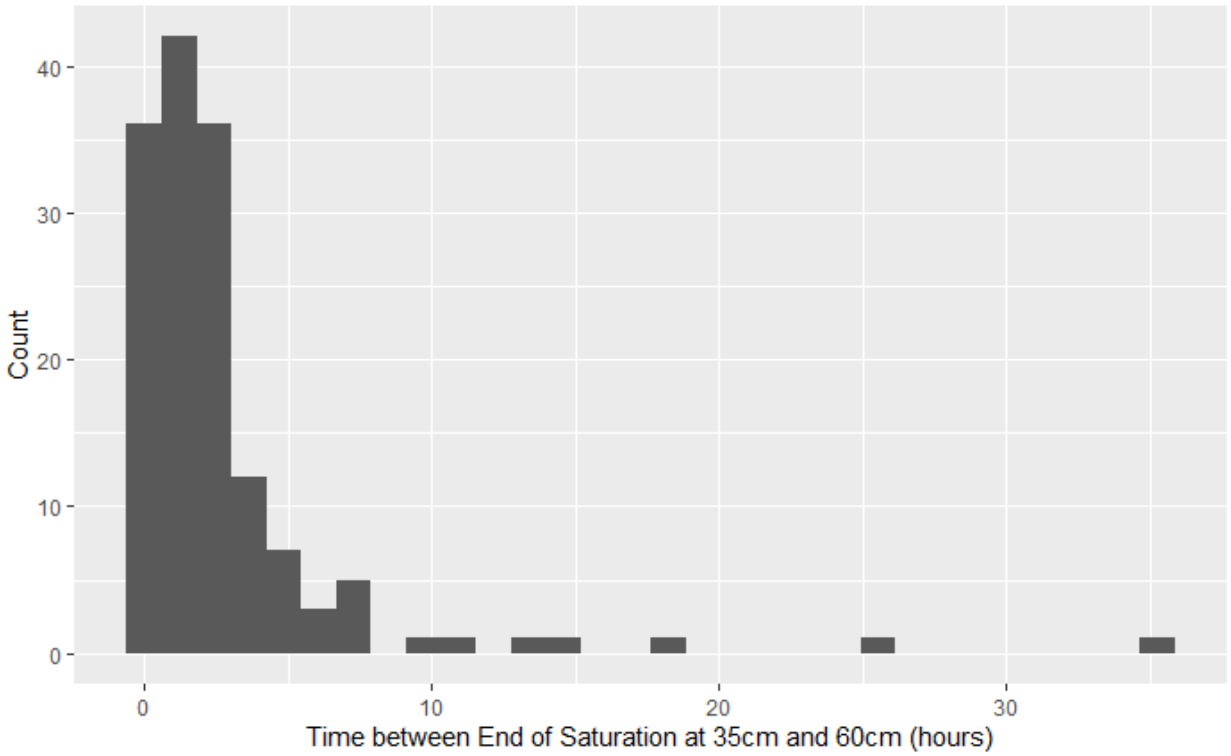

Figure 4.15.: Time to Desaturation from 35 to 60cm.

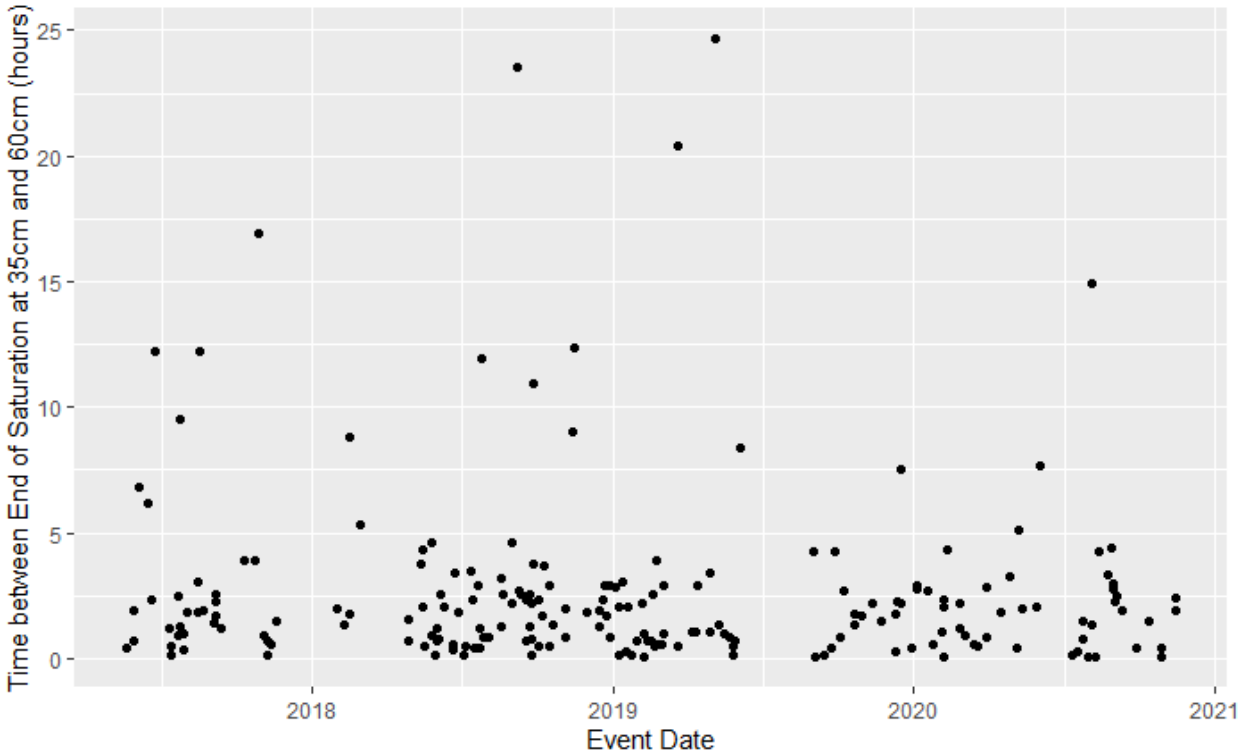

Figure 4.16.: Event Date vs Time to Desaturation from 10 to 35cm.

Conclusion

This research work shows the benefits of a multi-faceted approach to long-term monitoring and analysis of GSI systems. The integration of accurate data collection, robust data storage, and flexible yet relatable analysis methods across many sites has the potential to enable new insights into design and construction practices and reduce maintenance costs. The work outlined herein lays the groundwork for carrying out long-term monitoring by addressing common instrumentation issues and potential solutions, describing a flexible framework for data storage, and preliminary work on data-driven performance indicators.

Ensuring that sensor equipment is well suited to conduct appropriate measurements and report valid data allows for an accurate, uninterrupted collection of data. Understanding the intricacies of sensor configurations, communications, randomness of hydraulic characteristics, and opportunities for error helps mitigate invalid data. Sensor, wiring schematic, and data logger choices should reflect these nuanced factors. Digital communications significantly increase a sensor's ability to report valid data, ensuring that measurements taken in one location are correctly captured and stored by a physically separated data logger. The increased confidence in data collected will enable future analysis of long-term trends, so investments and efforts made now in upgrading systems can have lasting benefits for GSI studies across all sites. Furthermore, the improved data quality will free up researcher resources that can be focused on deriving insights and analyzing the data.

The FAIR principles are an easy check for data accessibility and make transferring analyses between sites simpler. A flexible data format enables analyses that are easily scalable, since consistent metadata and controlled vocabulary are used across all sites. This flexibility enables the expansion of the scope of research that is able to be conducted on larger GSI systems. This is accomplished through controlled vocabulary for describing data and consistent, yet flexible, formatting of the database schema. The long format of the main data table allows nearly infinite additional metadata combinations, meaning that future projects and sites will not require any updates to the schema. This flexibility to add future sites is a key feature of the SIDM, as it enables long term studies to encompass both historical and future data easily in a single repository. Future work to directly integrate monitoring equipment with the data ingestion process could mean that data becomes available in the SIDM almost instantaneously through the use of IoT protocols and methodologies, greatly expanding the value of consistent formatting and data descriptions. The SIDM will unlock the potential for expanded collaboration between researchers and among project partners such as PennDOT and AECOM, better understandings of long-term monitoring results, and more robust analyses of multi-site GSI systems.

Lastly, the analyses outlined within are well suited to leveraging the data, as the results are consistent with labor intensive field and lab tests commonly accepted for infiltration capacity. Recession and infiltration KPIs can be thought of as a proxy to these more complicated tests, such as the single and double ring infiltrometer or Saturo constant head tests, such that the long-term monitoring data is

able to act as a substitute for routine GSI health checks. The results of these health checks, which can be performed remotely by virtue of the connected monitoring network, can be used to determine the need for maintenance at a particular site or the need for more thorough inspection in cases of negatively trending performance. The data have shown that performance is most highly impacted by temperature, which oscillates with an annual seasonal period. This relationship is expected due to the change in water viscosity across the range of common temperatures in Philadelphia. In general, GSI health is related to the recession and drying rates of the ponding level and soil moisture, respectively. Both relationships are positive, such that higher rates indicate faster transmission of water through the engineered media layers and faster reduction in soil moisture leading to rapid recovery. Results show that average recession rates between 40 and 120 mm/hr can be expected at the case study site, depending on time of year and water temperature. Drying rate averages as high as 9.5 cm/hr over the course of a storm can be expected, although this value would be expected to vary between GSI systems based on soil design and pre-storm conditions.

These performance indicators are independent from GSI characteristics such as surface area, loading ratio, or number of inlets or outlets, and are therefore easily transferrable to other sites regardless of configuration. The KPIs can be calculated for any GSI system, or part of a system, to determine if infiltration rates and recovery rates fall within the expected range, aiding in the design process by enabling feedback regarding target performance. Few sensors are necessary for performing these analyses, namely a pressure transducer for ponding depth and a pair of soil moisture level sensors at a known separation distance, which helps keep costs for this kind of monitoring low. The use of a consistent data format for a collection of GSI systems will allow these KPIs to provided up-to-date and at-a-glance information about the health of an entire system, making monitoring and managing maintenance requirements much more simple. Therefore, implementing a basic monitoring network at many sites is economical, as robust data storage and analysis practices allow for data-driven decisions about allocation of maintenance and repair capital based on near real-time data, reduce wasted time and resources, and enable insight into long-term system-wide trends.

Bibliography

Abdallah, Adel M. and David E. Rosenberg (2019). “A data model to manage data for water resources systems modeling”. In: *Environmental Modelling and Software* 115, pp. 113–127 (cit. on p. 20).

Abdi, Hervé (2007). “The Kendall Rank Correlation Coefficient”. In: *Encyclopedia of Measurement and Statistics* (cit. on p. 41).

Akhavan, Bloorchian Azadeh and Zhou Jianpeng (2015). *Alternative Treatment of Flow Monitoring Data to Evaluate the Impact of Green Infrastructure on Stormwater Volume Reduction in Combined Sewers* (cit. on p. 4).

Albright, Cara Melissa and Harrison Schramm (2018). “Improvements and applications in climate data analysis for determining reference rainfall years”. In: *Journal of Applied Meteorology and Climatology* 57.2, pp. 413–420 (cit. on pp. 12, 30).

Asare, S. N., R. P. Rudra, W. T. Dickinson, and G. J. Wall (1999). “Effect of freeze-thaw cycle on the parameters of the green and ampt infiltration equation”. In: *Journal of Agricultural and Engineering Research* 73.3, pp. 265–274 (cit. on p. 37).

Bartens, Julia, Susan D. Day, J. Roger Harris, Joseph E. Dove, and Theresa M. Wynn (2008). “Can Urban Tree Roots Improve Infiltration through Compacted Subsoils for Stormwater Management?” In: *Journal of Environmental Quality* 37.6, pp. 2048–2057 (cit. on p. 37).

Braga, Andrea, Michael Horst, and Robert G. Traver (2007). “Temperature Effects on the Infiltration Rate through an Infiltration Basin BMP”. In: *Journal of Irrigation and Drainage Engineering* 133.6, pp. 593–601 (cit. on p. 34).

Brown, Robert A and William F Hunt (2012). “Improving bioretention/biofiltration performance with restorative maintenance”. In: *Water Science and Technology* 65.2, pp. 361–367 (cit. on p. 28).

Burcin, Becerik-Gerber, Siddiqui Mohsin K., Brilakis Ioannis, et al. (2014). “Civil Engineering Grand Challenges: Opportunities for Data Sensing, Information Analysis, and Knowledge Discovery”. In: *Journal of Computing in Civil Engineering* 28.4, p. 4014013 (cit. on p. 5).

Callahan, Bernadette M (2019). “Green Stormwater Infrastructure for CSO Control”. In: *World Environmental and Water Resources Congress 2019*, pp. 184–189 (cit. on pp. 1, 4).

Calt, Elizabeth (2018). “Comparing the Hydrologic Performance of a Linear Cascading Bioswale to Traditional Bioinfiltration in a Highly Urbanized Setting”. Master’s Thesis. Villanova University (cit. on pp. 2, 12).

Davis, Allen P., William F. Hunt, Robert G. Traver, and Michael Clar (2009). “Bioretention Technology: Overview of Current Practice and Future Needs”. In: *Journal of Environmental Engineering* 135.3, pp. 109–117 (cit. on p. 32).

Davis, Allen P., Robert G. Traver, William F. Hunt, et al. (2012). “Hydrologic Performance of Bioretention Storm-Water Control Measures”. In: *Journal of Hydrologic Engineering* 17.5, pp. 604–614 (cit. on p. 28).

DelGrosso, Zack L., Clayton C. Hodges, and Randel L. Dymond (2019). “Identifying Key Factors for Implementation and Maintenance of Green Stormwater Infrastructure”. In: *Journal of Sustainable Water in the Built Environment* 5.3, p. 05019002 (cit. on pp. 20, 28).

Dourte, Daniel R., Clyde W. Fraisse, and Wendy Lin Bartels (2015). “Exploring changes in rainfall intensity and seasonal variability in the Southeastern U.S.: Stakeholder engagement, observations, and adaptation”. In: *Climate Risk Management* 7, pp. 11–19 (cit. on p. 5).

Eargle, John M (2002). “Resistance Change with Temperature for Copper”. In: *Electroacoustical Reference Data*. Boston, MA: Springer US, pp. 106–107 (cit. on p. 11).

Emerson, Clay H. and Robert G. Traver (2008). “Multiyear and Seasonal Variation of Infiltration from Storm-Water Best Management Practices”. In: *Journal of Irrigation and Drainage Engineering* 134.5, pp. 598–605 (cit. on p. 37).

Eyo, E. U., S. Ng’ambi, and S. J. Abbey (2020). *An overview of soil–water characteristic curves of stabilised soils and their influential factors* (cit. on pp. 29, 30, 34).

Geberemariam, Thewodros K (2017). *Post construction green infrastructure performance monitoring parameters and their functional components* (cit. on p. 28).

Gregory, J H, Michael Dukes, Pierce Jones, and Grady Miller (2006). “Effect of Urban Soil Compaction on Infiltration Rate”. In: *Journal of Soil and Water Conservation* 61, pp. 117–124 (cit. on p. 33).

Heffernan, Taylor, Stephen White, Tyler Krechmer, et al. (2016). “Green Stormwater Infrastructure Monitoring of Philadelphia’s Green City, Clean Waters Program”. In: *World Environmental and Water Resources Congress 2016*, pp. 115–124 (cit. on p. 4).

Horsburgh, Jeffery S., David G. Tarboton, David R. Maidment, and Ilya Zaslavsky (2008). “A relational model for environmental and water resources data”. In: *Water Resources Research* 44.5, p. 5406 (cit. on pp. 21, 22).

Horton, R., M. D. Ankeny, and R. R. Allmaras (1994). “Effects of Compaction on Soil Hydraulic Properties”. In: *Developments in Agricultural Engineering*. Vol. 11. C. Elsevier, pp. 141–165 (cit. on p. 33).

Jenkins, Jennifer K. Gilbert, Bridget M. Wadzuk, and Andrea L. Welker (2010). “Fines Accumulation and Distribution in a Storm-Water Rain Garden Nine Years Postconstruction”. In: *Journal of Irrigation and Drainage Engineering* 136.12, pp. 862–869 (cit. on p. 33).

Joshi, Deep C., Sascha C. Iden, Andre Peters, Bhabani S. Das, and Wolfgang Durner (2019). “Temperature Dependence of Soil Hydraulic Properties: Transient Measurements and Modeling”. In: *Soil Science Society of America Journal* 83.6, pp. 1628–1636 (cit. on p. 34).

Lightstone, S S, T J Teorey, and T Nadeau (2010). *Physical Database Design: The Database Professional’s Guide to Exploiting Indexes, Views, Storage, and More*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science (cit. on p. 22).

Liu, Wen, Qi Feng, Ravinesh C. Deo, Lei Yao, and Wei Wei (2020). “Experimental Study on the Rainfall-Runoff Responses of Typical Urban Surfaces and Two Green Infrastructures Using Scale-Based Models”. In: *Environmental Management* 66.4, pp. 683–693 (cit. on p. 30).

Low Impact Development Center (2004). *National Low Impact Development Clearinghouse* (cit. on p. 20).

Machusick, Matthew, Andrea Welker, and Robert Traver (2011). “Groundwater Mounding at a Storm-Water Infiltration BMP”. In: *Journal of Irrigation and Drainage Engineering* 137.3, pp. 154–160 (cit. on p. 29).

Maidment, David R. (2008). “Bringing Water Data Together”. In: *Journal of Water Resources Planning and Management* 134.2, pp. 95–96 (cit. on pp. 20, 21).

Maier, Roman, Gerald Krebs, Markus Pichler, Dirk Muschalla, and Günter Gruber (2020). “Spatial rainfall variability in urban environments-high-density precipitation measurements on a city-scale”. In: *Water (Switzerland)* 12.4 (cit. on p. 5).

Mays, L W (2010). *Water Resources Engineering*. Wiley (cit. on pp. 6, 12, 14).

Meng, Ting, David Hsu, and Bridget Wadzuk (2017). “Green and Smart: Perspectives of City and Water Agency Officials in Pennsylvania toward Adopting New Infrastructure Technologies for Stormwater Management”. In: *Journal of Sustainable Water in the Built Environment* 3.2, p. 05017001 (cit. on pp. 20, 28).

MQTT (2021). *MQTT: The Standard for IoT Messaging* (cit. on p. 26).

OpenChannelFlow (2021). *Flow Characteristics of Blood* (cit. on pp. 13, 14).

Philadelphia Water Department (2018). *5 Down, 20 to Go: Celebrating 5 Years of Cleaner Water and Greener Neighborhoods* (cit. on p. 4).

– (2021). *City of Philadelphia: Watershed Protection* (cit. on pp. 2, 4).

RStudio Team (2020). *RStudio: Integrated Development Environment for R*. RStudio, PBC. Boston, MA (cit. on p. 23).

Sokolovskaya, Natalya, Ali Ebrahimian, and Bridget Wadzuk (2021). “Modeling Infiltration in Green Stormwater Infrastructure: Effect of Geometric Shape”. In: *Journal of Sustainable Water in the Built Environment* 7.2, p. 04020020 (cit. on pp. 29, 39).

Stevens Water (2021). *HydraProbe Data Sheet* (cit. on pp. 9, 10).

Taguchi, Vinicius J., Peter T. Weiss, John S. Gulliver, et al. (2020). *It is not easy being green: Recognizing unintended consequences of green stormwater infrastructure* (cit. on p. 1).

Tu, Min-Cheng and Robert Traver (2019). “Water Table Fluctuation from Green Infrastructure Sidewalk Planters in Philadelphia”. In: *Journal of Irrigation and Drainage Engineering* 145.2, p. 05018008 (cit. on p. 29).

USBR (2001). *Water Measurement Manual - Chapter 7 - Weirs*. Tech. rep. (cit. on p. 9).

USEPA (2009). *Urban stormwater management in the United States*, pp. 1–598 (cit. on pp. 1, 4, 28).

Wadzuk, Bridget, Bridget Gile, Virginia Smith, Ali Ebrahimian, and Robert Traver (2021a). “Call for a Dynamic Approach to GSI Maintenance”. In: *Journal of Sustainable Water in the Built Environment* 7.2, p. 02521001 (cit. on p. 20).

Wadzuk, Bridget, Bridget Gile, Virginia Smith, et al. (2021b). “Moving Towards Dynamic and Data-driven GSI Maintenance”. Villanova, PA (cit. on p. 20).

Wadzuk, Bridget M., Conor Lewellyn, Ryan Lee, and Robert G. Traver (2017). “Green Infrastructure Recovery: Analysis of the Influence of Back-to-Back Rainfall Events”. In: *Journal of Sustainable Water in the Built Environment* 3.1, p. 04017001 (cit. on pp. 30, 32, 34).

Wilkinson, Mark D., Michel Dumontier, IJsbrand Jan Aalbersberg, et al. (2016). “The FAIR Guiding Principles for scientific data management and stewardship”. In: *Scientific Data* 3 (cit. on p. 21).

Zukowski, Zachary, Clay H. Emerson, Andrea L. Welker, and Brendon Achey (2016). “Evaluation of Field Hydraulic Conductivity Data: Comparing Spot Infiltrometer Test Data to Continuous Recession Data”. In: American Society of Civil Engineers (ASCE), pp. 517–526 (cit. on p. 33).

Appendix A

Figure A.1.: SIDM database schema (Smith et al., submitted).

Appendix B

The following R files are also available at `www.github.com/andrewkurzweil/MSCE-Thesis/R`.

Data ingest, cleaning, basic event statistics:

```
## PennDOT Event Creation Processor
## Written by Andrew Kurzweil, MSCE/MSAS 2021
## FOR SMP A DATA ONLY!!

# Import necessary packages
# Packages must be installed using "install.packages('package.name')" first (R Studio typically prompts for missing packages, though)

library("readxl")
library("GISTools")
library("pracma")
library("ids")
library("Amelia")
library("dplyr")
library("ggplot2")
library("reshape")
library("lubridate")
library("RcppRoll")
library("rmarkdown")
library("imputeTS")
# set base directory where your data input files are stored and output directories should be rooted
#base.dir = "X:/1-PennDOT/H&H/Analysis/A_Analysis/EVENTS/"
base.dir = "K:/PennDOT DATA TEMP/DATA/"

setwd(base.dir)

#####################################
############# FUNCTIONS #############
#####################################


#' this function identifies inflection points in soil moisture data
#' point_number corresponds to ABCD as follows:
#' a = 1
#' b = 2
#' c = 3
#' d = 4
#'
#' Points' significance are outlined in work done by Matina Shakya (PhD 2021)
#'
identify_soil_moisture_inflection <-
  function(timestamps, data, point_number) {
```

```
    if (length(data) * 0.5 <= sum(is.na(data)) ) {
      return(NA)
    } else {
      # remove NA values from data
      data = na_replace(data, fill = 0)
      # first differential of data
      diff = data - lag(data)

      peak_points = sort(findpeaks(abs(diff), npeaks = 4, minpeakdistance =
    2)[,2])

      return(as.POSIXct(timestamps[peak_points[point_number] - 1], origin =
    '1970-01-01', tz = 'UTC'))
    }
} ## end soil moisture point identification

calculate_AV_Flow_CMS <- function(depth, velocity, pipe.diameter) {
  # depth: recorded depth value (mm)
  # velocity: recorded velocity value (mm/s)
  # pipe.diameter: size of pipe (mm)
  # ALL UNITS MUST MATCH!
  # See https://en.wikipedia.org/wiki/Circular_segment for formulas used
  radius <- pipe.diameter / 2
  depth[depth < 5 |
          is.na(depth)] = 0            # remove depths below threshold and NULL
    depth values
  velocity[velocity < 5 |
             is.na(velocity)] = 0   # remove velocities below threshold and NULL
    velocity values
  depth[depth > radius] = radius                  # area calculation maxes out
    at 50% full pipe !!! NEED TO SOLVE THIS !!!
  theta <-
    2 * acos(1 - (depth / radius))             # central angle of filled portion
    of pipe
  area <-
    (radius * radius / 2) * (theta - sin(theta))  # area of circular segment
    filled by water
  volume <-
    area * velocity                       # Area*Velocity, result in cubic mm
    per second
  volume <-
    volume / (1000 * 1000 * 1000)             # convert cubic mm to cubic m
  as.numeric(volume)                          # return volume vector
}

calculate_V_notch_flow_CMS <- function(depth, well, angle) {
  # depth: raw data for outlet depth
  # well: depth below invert of V-notch that should be ignored
  # angle: angle of V-notch weir
```

```
  depth[is.na(depth)] = 0     # remove NA values
  depth = depth - well          # subtract well depth
  depth[depth < 0] = 0          # remove negative values
  depth = depth / 304.8         # convert mm to feet
  flow = 0.585 * 4.2799 * tan(deg2rad(angle) / 2) * depth ** (5 / 2) ##
    generalized weir equation for English units
  flow = flow / 35.315          # convert CFS to CMS
  flow
}

assign.event.ids <- function(rainfall, time_window_hrs)
{
  # rainfall: a vector of rainfall summations for each timestep
  # has.rainfall: vector of logicals indicating whether rain occured at a given
    timestamp
  has.rainfall = rainfall != 0

  # calculate a 6-hour moving average for purposes of identifying period of no
    rainfall easily (when moving average = 0, no rainfall in last 6 hours)
  # (there are 72 5-minute intervals in 6 hours)
  rainfall.moving.avg = movavg(rainfall, time_window_hrs * 12, "s")

  # set up event.start, event.end, and ID.list vectors
  event.start = logical(length = length(rainfall))
  event.end = logical(length = length(rainfall))
  ID.list = vector(length = length(rainfall))

  # iterate over rainfall vector to populate event.start and event.end logical
    vectors
  for (n in seq_along(rainfall[-length(rainfall)])) {
    # if current timestep has rainfall and there has not been rain in the last
    `time_window_hrs` hours (moving average is 0), this is the start of an event
    if (has.rainfall[n] && rainfall.moving.avg[n - 1] == 0)
    {
      event.start[n] = TRUE
    }
    else
    {
      event.start[n] = FALSE
    }

    # if moving average reaches 0 in the next timestep, current timestep is the
    end of an event
    if (rainfall.moving.avg[n] != 0 &&
        rainfall.moving.avg[n + 1] == 0)
    {
      event.end[n] = TRUE
    }
    else
```

```
    {
      event.end[n] = FALSE
    }
  }

  # initial random event ID
  event.ID = random_id(bytes = 4)
  fill = FALSE

  for (n in seq_along(rainfall)) {
    #generate new ID when new event starts and start filling it in
    if (event.start[n] == TRUE) {
      set.seed(as.numeric(dat$TIMESTAMP[n])) # set seed as timestamp of
    beginning of event to get consistent STORMID results
      event.ID = random_id(bytes = 4, use_openssl = FALSE)  # random string 8
    characters long based on set.seed value. use_openssl = FALSE forces use of
    internal R randomizer
      fill = TRUE
    }

    # determine if event ID gets filled
    if (fill) {
      ID.list[n] = event.ID
    }
    else {
      ID.list[n] = NA
    }

    # stop filling event ID when event stops
    if (event.end[n] == TRUE) {
      fill = FALSE
    }

  }

  as.factor(ID.list)

}

assign.ponding.event.ids <- function(timestamps,
                                     rainfall,
                                     ponding,
                                     pre_rain_window_mins = 30,
                                     post_ponding_window_mins = 120,
                                     min_event_duration_mins = 360) {

  # min_event_duration_mins cannot be less than post_ponding_window_mins
  if(min_event_duration_mins < post_ponding_window_mins)
    post_ponding_window_mins = min_event_duration_mins
```

```
171
172   # rainfall: a vector of rainfall summations for each timestep
173   # has.rainfall: vector of logicals indicating whether rain occured at a given
        timestamp
174   has.rainfall = rainfall != 0
175
176   rainfall[is.na(rainfall)] = 0
177   # calculate a 6-hour moving average for purposes of identifying period of no
        rainfall easily (when moving average = 0, no rainfall in last 6 hours)
178   # (there are 72 5-minute intervals in 6 hours)
179   if(pre_rain_window_mins > 5)
180   {
181     rainfall.moving.avg = movavg(rainfall, pre_rain_window_mins / 5, 's')
182   } else {
183     rainfall.moving.avg = rainfall
184   }
185
186   ponding[is.na(ponding)] = 0
187   if (post_ponding_window_mins > 5)
188   {
189     ponding.moving.avg = movavg(ponding, post_ponding_window_mins / 5, "s")
190   }
191   else
192   {
193     ponding.moving.avg = ponding
194   }
195   # set up event.start, event.end, and ID.list vectors
196   event.start = logical(length = length(rainfall))
197   event.end = logical(length = length(rainfall))
198   ID.list = vector(length = length(rainfall))
199
200   # event length indicator
201   i = 0
202   in_event = FALSE
203
204   # iterate over rainfall vector to populate event.start and event.end logical
        vectors
205   for (n in ((post_ponding_window_mins / 5) + 1):(length(rainfall) - 1 -
        pre_rain_window_mins /
206                                                   5)) {
207     # determine if an event has started at (n)
208     if (rainfall.moving.avg[n + pre_rain_window_mins / 5] == 0 &&
209         rainfall.moving.avg[n + 1 + pre_rain_window_mins / 5] != 0 &&
210         !in_event)
211     {
212       i = 0
213       event.start[n] = TRUE
214       in_event = TRUE
215     }
```

```
    else
    {
      i = i + 5
      event.start[n] = FALSE
    }

    # determine if an event has ended at (n)
    if (in_event && # event currently happening

        (
          (ponding.moving.avg[n] == 0 && # ponding currently 0
          i >= min_event_duration_mins && # min event duration reached
          ponding.moving.avg[n - (post_ponding_window_mins / 5)] == 0) # ponding X steps ago was < 5
        ||
         (ponding.moving.avg[n - (post_ponding_window_mins / 5)] != 0 && # ponding post_ponding_window_mins in past > 5
          ponding.moving.avg[n + 1 - (post_ponding_window_mins / 5)] == 0 && # ponding post_ponding_window_mins in past + 5 <= 5
          ponding.moving.avg[n] == 0) # current ponding level below 5
        )
      )
    {
      event.end[n] = TRUE
      in_event = FALSE
    }
    else
    {
      event.end[n] = FALSE
    }
  }

  # initial random event ID
  event.ID = random_id(bytes = 4)
  fill = FALSE

  for (n in seq_along(rainfall)) {
    #generate new ID when new event starts and start filling it in
    if (event.start[n] == TRUE) {
      set.seed(as.numeric(timestamps[n])) # set seed as timestamp of beginning of event to get consistent STORMID results
      event.ID = random_id(bytes = 4, use_openssl = FALSE)  # random string 8 characters long based on set.seed value. use_openssl = FALSE forces use of internal R randomizer
      fill = TRUE
    }

    # determine if event ID gets filled
    ID.list[n] = ifelse(fill,event.ID,NA)
```

```
259
260     # stop filling event ID when event stops
261     if (event.end[n] == TRUE) {
262       fill = FALSE
263     }
264
265   }
266
267   as.factor(ID.list)
268
269 }
270
271 assign.soil_moisture.event.ids <- function(timestamps,
272                                     rainfall,
273                                     soil_moisture,
274                                     pre_rain_window_mins = 30,
275                                     post_soil_moisture_window_mins = 120,
276                                     min_event_duration_mins = 360) {
277
278   # min_event_duration_mins cannot be less than post_ponding_window_mins
279   if(min_event_duration_mins < post_soil_moisture_window_mins)
      post_soil_moisture_window_mins = min_event_duration_mins
280
281   # rainfall: a vector of rainfall summations for each timestep
282   # has.rainfall: vector of logicals indicating whether rain occured at a given
      timestamp
283   has.rainfall = rainfall != 0
284
285   rainfall[is.na(rainfall)] = 0
286   # calculate a 6-hour moving average for purposes of identifying period of no
      rainfall easily (when moving average = 0, no rainfall in last 6 hours)
287   # (there are 72 5-minute intervals in 6 hours)
288   if(pre_rain_window_mins > 5)
289   {
290     rainfall.moving.avg = movavg(rainfall, pre_rain_window_mins / 5, 's')
291   } else {
292     rainfall.moving.avg = rainfall
293   }
294
295   #soil_moisture = na_ma(soil_moisture)
296   if (post_soil_moisture_window_mins > 5)
297   {
298     soil_moisture.ma = na_interpolation(movavg(soil_moisture,
      post_soil_moisture_window_mins / 5, "s"))
299   }
300   else
301   {
302     soil_moisture.ma = na_locf(soil_moisture)
303   }
```

```
# set up event.start, event.end, and ID.list vectors
event.start = logical(length = length(rainfall))
event.end = logical(length = length(rainfall))
ID.list = vector(length = length(rainfall))

# event length indicator
i = 0
in_event = FALSE
threshold = 0

# iterate over rainfall vector to populate event.start and event.end logical vectors
for (n in ((post_soil_moisture_window_mins / 5) + 1):(length(rainfall) - 1 - pre_rain_window_mins /
                                                      5))
{
  if ( !(is.na(soil_moisture[n]) || is.na(soil_moisture[n - 1]) ) ) {
    # determine if an event has started at (n)
    if (rainfall.moving.avg[n + pre_rain_window_mins / 5] == 0 &&
        rainfall.moving.avg[n + 1 + pre_rain_window_mins / 5] != 0 &&
        !in_event)
    {
      threshold = soil_moisture.ma[n]
      i = 0
      event.start[n] = TRUE
      in_event = TRUE
    }
    else
    {
      i = i + 5
      event.start[n] = FALSE
    }

    # if there is a jump of 0.06 or greater, end the event at the previous timestep and start a new one at this timestep
    if (in_event &&
        (soil_moisture[n] - soil_moisture[n - 1] > 0.06))
    {
      in_event = FALSE
      event.start[n - 2] = TRUE
      event.end[n - 3] = TRUE
    }

    # determine if an event has ended at (n)
    if (in_event && # event currently happening

        (
          (
            soil_moisture.ma[n] <= threshold && # ponding currently 0
```

```
              i >= min_event_duration_mins &&
              # min event duration reached
              soil_moisture.ma[n - (post_soil_moisture_window_mins / 5)] <=
    threshold
            ) # ponding X steps ago was < 5
            ||
            (
              soil_moisture.ma[n - (post_soil_moisture_window_mins / 5)] >
    threshold &&
              # ponding post_ponding_window_mins in past > 5
              soil_moisture.ma[n + 1 - (post_soil_moisture_window_mins / 5)] <=
    threshold &&
              # ponding post_ponding_window_mins in past + 5 <= 5
              soil_moisture.ma[n] <= threshold
            ) # current ponding level below 5
          ))
      {
        event.end[n] = TRUE
        in_event = FALSE
      }
      else
      {
        event.end[n] = FALSE
      }
    }

  }

  # initial random event ID
  event.ID = random_id(bytes = 4)
  fill = FALSE

  for (n in seq_along(rainfall)) {
    #generate new ID when new event starts and start filling it in
    if (event.start[n] == TRUE) {
      set.seed(as.numeric(timestamps[n])) # set seed as timestamp of beginning
    of event to get consistent STORMID results
      event.ID = random_id(bytes = 4, use_openssl = FALSE)  # random string 8
    characters long based on set.seed value. use_openssl = FALSE forces use of
    internal R randomizer
      fill = TRUE
    }

    # determine if event ID gets filled
    ID.list[n] = ifelse(fill,event.ID,NA)

    # stop filling event ID when event stops
    if (event.end[n] == TRUE) {
      fill = FALSE
```

```
    }

  }

  as.factor(ID.list)

}

is_saturated <- function(soil_moisture) {
  return(ifelse(soil_moisture %between% c(0.43,0.50), TRUE, FALSE))
}

# end functions

#####################################
############# CONSTANTS #############
#####################################

# Edit these based on your project needs, but keep variable names unchanged

# read in data and treat listed values as NA/NULL/missing. Deal with them later, as necessary. MUST DECLARE COLUMNS TYPES. R is bad at guessing...
SMPAdata <-
  read_excel(
    #"X:/Grad Student Folders/Andrew Kurzweil/database/PennDOT Database Organization/SMPA20172019.xlsx",
    "SMPA20172019.xlsx",
    # File Location
    # values to ignore
    na = c(
      ".",
      "NA",
      "",
      "?",
      "-9999",
      "-999900",
      "NaN",
      "7999",
      "-8888",
      "#VALUE!",
      "-8898",
      "NAN"
    ),
    # column types
    col_types = c(
      "date",
      "numeric",
      "numeric",
      "numeric",
```

```
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
        "numeric",
```

```
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric"
    )
  )
WSdata <-
  read_excel(
    #"X:/Grad Student Folders/Andrew Kurzweil/database/PennDOT Database
    Organization/WS20162019.xlsx",
    "WS20162019.xlsx",
    # File Location
    # values to ignore
    na = c(
      ".",
      "NA",
      "",
      "?",
      "-9999",
      "-999900",
      "NaN",
      "7999",
      "-8888",
      "#VALUE!",
      "-8898",
      "NAN"
    ),
    # column types
    col_types = c(
      "date",
```

```
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric",
      "numeric"
    )
  )
# Confirm that data types are what we want (all numeric except TIMESTAMP)
str(SMPAdata)
str(WSdata)

# merge dataframes by matching "TIMESTAMP" column. Creates a single, unified dataset to work from, with all data in the same place
dat <- merge(WSdata, SMPAdata, by = "TIMESTAMP")
# confirm structure of unified dataframe
str(dat)

# assume no rainfall when rain gage has NA values. Rainfall event identifier function doesn't like NA values
dat$CS700Rain_in_TOT[is.na(dat$CS700Rain_in_TOT)] = 0

# end constants

#####################################
############# ANALYSIS ##############
#####################################

# Dependent functions are stored at the top of this script
# so that they get run first. Read all comments carefully before
# attempting to change anything.

# "Missing Map" from package "Amelia" takes a while to run, so I usually leave
# it commented out. It's helpful to see the percentage of NA values and their distribution thoughout your data
# missmap(dat, main="SMP A Missing Map", y.labels = dat$TIMESTAMP, y.at = seq_along(length(dat$TIMESTAMP)), margins = c(7,7))

# create unique ID's for each storm event based on time-series rainfall.
# See "assign.event.ids" function defined above for specifics.
dat$stormID = assign.event.ids(rainfall = dat$CS700Rain_in_TOT, time_window_hrs = 6)
```

```
dat$stormID_2hr = assign.event.ids(rainfall = dat$CS700Rain_in_TOT,
    time_window_hrs = 2)
dat$stormID_48hr = assign.event.ids(rainfall = dat$CS700Rain_in_TOT,
    time_window_hrs = 48)

dat$stormID_ponding = assign.ponding.event.ids(
  timestamps = dat$TIMESTAMP,
  rainfall = dat$CS700Rain_in_TOT,
  ponding = dat$GR2a_B1PondDepth,
  pre_rain_window_mins = 120,
  post_ponding_window_mins = 120
)

dat$storm_index = ave(dat$CS700Rain_in_TOT, dat$stormID, FUN = seq_along)
dat$cumulative_rainfall = ave(dat$CS700Rain_in_TOT, dat$stormID, FUN = cumsum)

# set parameters used to calculate flows
N8.diameter = 30 * 25.4    # millimeters
N9.diameter = 18 * 25.4    # millimeters
N10.diameter = 18 * 25.4   # millimeters

B2.outlet.well.depth = 230 # millimeters
B1.outlet.well.depth = 230 # millimeters

B2.outlet.weir.angle = 30 # degrees
B1.outlet.weir.angle = 30 # degrees

Flume.well.depth = 42 # millimeters

# calculate flow rates for AV sensors. See "calculate_AV_Flow_CMS" function
    defined above for specifics
dat$N8_In_Flow_CMS = calculate_AV_Flow_CMS(dat$GR2a_N8InletDepth,
    dat$GR2a_N8InletSpeed, N8.diameter)
dat$N9_In_Flow_CMS = calculate_AV_Flow_CMS(dat$GR2a_N9InletDepth,
    dat$GR2a_N9InletSpeed, N9.diameter)
dat$N10_In_Flow_CMS = calculate_AV_Flow_CMS(dat$GR2a_N10InletDepth,
    dat$GR2a_N10InletSpeed, N10.diameter)

dat$N8_sumvel = ave(dat$GR2a_N8InletSpeed, dat$stormID, FUN = cumsum)
dat$N9_sumvel = ave(dat$GR2a_N9InletSpeed, dat$stormID, FUN = cumsum)
dat$N10_sumvel = ave(dat$GR2a_N10InletSpeed, dat$stormID, FUN = cumsum)


dat$N8_Hwy_Flow_CMS = calculate_AV_Flow_CMS(dat$GR2a_N8HwyDepth,
    dat$GR2a_N8HwySpeed, N8.diameter)
dat$N9_Hwy_Flow_CMS = calculate_AV_Flow_CMS(dat$GR2a_N9HwyDepth,
    dat$GR2a_N9HwySpeed, N9.diameter)

dat$B1_Outlet_Flow_CMS = calculate_V_notch_flow_CMS(dat$GR2a_B1OutletDepth,
```

```
                                                    B1.outlet.well.depth,
                                                    B1.outlet.weir.angle)
dat$B2_Outlet_Flow_CMS = calculate_V_notch_flow_CMS(dat$GR2a_B2OutletDepth,
                                                    B2.outlet.well.depth,
                                                    B2.outlet.weir.angle)

dat$GR2a_Flume_Depth_Avg = (dat$GR2a_Flume_Depth_Avg - Flume.well.depth)/1000 #
    subtract well depth and convert to meters
dat$GR2a_Flume_Depth_Avg = ifelse(dat$GR2a_Flume_Depth_Avg < 0, NA,
    dat$GR2a_Flume_Depth_Avg)

# 0.8-foot HS Flume, equation from Open Channel Flow:
    https://www.openchannelflow.com/assets/uploads/documents/08-foot_HS_flume_discharge_table.p
dat$Flume_Flow_LPS = -0.00707921 + 0.04898248 * (dat$GR2a_Flume_Depth_Avg **
    0.5) + 21.70307374 * (dat$GR2a_Flume_Depth_Avg ** 1.5) + 365.9132927 *
    (dat$GR2a_Flume_Depth_Avg ** 2.5)

dat$N8_In_Volume_m3 = dat$N8_In_Flow_CMS * 300
dat$N9_In_Volume_m3 = dat$N9_In_Flow_CMS * 300
dat$N10_In_Volume_m3 = dat$N10_In_Flow_CMS * 300
dat$N8_Hwy_Volume_m3 = dat$N8_Hwy_Flow_CMS * 300
dat$N9_Hwy_Volume_m3 = dat$N9_Hwy_Flow_CMS * 300
dat$B1_Outlet_Volume_m3 = dat$B1_Outlet_Flow_CMS * 300
dat$B2_Outlet_Volume_m3 = dat$B2_Outlet_Flow_CMS * 300
dat$Flume_Volume_m3 = dat$Flume_Flow_LPS / 1000 * 300


dat$B1PondDelta = c(diff(dat$GR2a_B1PondDepth),0)
dat$B2PondDelta = c(diff(dat$GR2a_B2PondDepth),0)

dat$month <- factor(format(dat$TIMESTAMP, "%m"))
dat$year <- factor(format(dat$TIMESTAMP, "%Y"))

# add moving sums of different lengths to facilitate calculating peak intensities
dat$rainfall_sum_15min = roll_sum(dat$CS700Rain_in_TOT,
                                  n = 3,
                                  na.rm = TRUE,
                                  fill = 0)
dat$rainfall_sum_30min = roll_sum(dat$CS700Rain_in_TOT,
                                  n = 6,
                                  na.rm = TRUE,
                                  fill = 0)

# save updated DAT dataframe for use in other files, and update master Rdata
    file for long term analysis
# this technically rewrites the entire file, rather than appending new stuff, so
    be careful about deleting old data from your input files
# change the name in the save() function below if you want to create a new
    master Rdata file for some reason
```

```
setwd(base.dir)
save(dat, file = "SMPA_ALL_DATA.Rdata")


###################################
#' Stats Section
###################################


# create stats table with summary data about storms from given month
stats <- dat %>%
  filter(!is.na(stormID)) %>% # remove NA stormID values (occurs when there's no event recorded)
  arrange(TIMESTAMP) %>%      # order the dataframe by timestamp so things get calculated in chronological order
  group_by(stormID) %>%       # group by stormID (ie perform all other operations on dat as grouped by stormID. this is how it becomes a pivot table.)
  # summarize creates new columns with the names you specify according to the formulas you provide.
  summarize(
    # start of event is the smallest Timestamp, end of event is the largest
    Storm_Event_Start_Time = min(TIMESTAMP),
    Storm_Event_End_Time = max(TIMESTAMP),
    # need just storm date (no time) for reports
    Event_Date = date(min(TIMESTAMP)),
    # Total rainfall is summed over event
    Total_Rainfall_mm = sum(CS700Rain_in_TOT),
    # Peak intensity is the highest single rainfall value (multipled by 12 for hourly rate, assuming 5 minute data)
    Peak_Intensity_mm.hr = max(CS700Rain_in_TOT * 12),
    Rainfall_Duration_min = length(CS700Rain_in_TOT > 0) * 5,
    Peak_15min_Intensity_mm.hr = max(rainfall_sum_15min) * 4,
    Peak_30min_Intensity_mm.hr = max(rainfall_sum_30min) * 2,

    Mean_Air_Temp_degC = mean(AirTC_Avg),

    # max flowrates are just the instantaneous max values from previous calcs
    N8_Max_Flowrate_CMS = max(N8_In_Flow_CMS),
    N9_Max_Flowrate_CMS = max(N9_In_Flow_CMS),
    N10_Max_Flowrate_CMS = max(N10_In_Flow_CMS),
    N8_Hwy_Max_Flowrate_CMS = max(N8_Hwy_Flow_CMS),
    N9_Hwy_Max_Flowrate_CMS = max(N9_Hwy_Flow_CMS),

    # Volumes are RATE * TIME summed over the entire storm. Assumes 5 minute (300 second) data
    N8_Volume_m3 = sum(N8_In_Flow_CMS * 300),
    N9_Volume_m3 = sum(N9_In_Flow_CMS * 300),
    N10_Volume_m3 = sum(N10_In_Flow_CMS * 300),
```

```
    N8_Hwy_Volume_m3 = sum(N8_Hwy_Flow_CMS * 300),
    N9_Hwy_Volume_m3 = sum(N9_Hwy_Flow_CMS * 300),

    Flume_Volume_m3 = sum(Flume_Flow_LPS / 1000 * 300),
    Flume_Max_Flowrate_CMS = max(Flume_Flow_LPS / 1000),

    B1_Outlet_Volume_m3 = sum(B1_Outlet_Flow_CMS * 300),
    B2_Outlet_Volume_m3 = sum(B2_Outlet_Flow_CMS * 300),

    N8_peak_velocity_mm.s = max(GR2a_N8InletSpeed),
    N9_peak_velocity_mm.s = max(GR2a_N9InletSpeed),
    N10_peak_velocity_mm.s = max(GR2a_N10InletSpeed),
    N8_hwy_peak_velocity_mm.s = max(GR2a_N8HwySpeed),
    N9_hwy_peak_velocity_mm.s = max(GR2a_N9HwySpeed),

    # max (cumulative velocity) = sum (individual velocity data) ... not sure
    why I did it this way
    N8_total_vel = max(N8_sumvel),
    N9_total_vel = max(N9_sumvel),
    N10_total_vel = max(N10_sumvel),

    B1_pond_max_depth_mm = max(GR2a_B1PondDepth),
    B2_pond_max_depth_mm = max(GR2a_B2PondDepth),

    # these are not great estimates...poor definition of 'receding limb' &
    doesn't account for refilling
    B1_pond_avg_recession_mm.hr = 12 * mean(GR2a_B1PondDepth[which(TIMESTAMP >
    TIMESTAMP[which.max(GR2a_B1PondDepth)])] -
    lead(GR2a_B1PondDepth[which(TIMESTAMP >
    TIMESTAMP[which.max(GR2a_B1PondDepth)])]), na.rm = TRUE),
    B2_pond_avg_recession_mm.hr = 12 * mean(GR2a_B2PondDepth[which(TIMESTAMP >
    TIMESTAMP[which.max(GR2a_B2PondDepth)])] -
    lead(GR2a_B2PondDepth[which(TIMESTAMP >
    TIMESTAMP[which.max(GR2a_B2PondDepth)])]), na.rm = TRUE),

    # these account for possible refilling
    B1_0_100_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0 &
    GR2a_B1PondDepth > 0 & GR2a_B1PondDepth < 100], na.rm = TRUE),
    B1_100_200_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0
    & GR2a_B1PondDepth > 100 & GR2a_B1PondDepth < 200], na.rm = TRUE),
    B1_200_300_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0
    & GR2a_B1PondDepth > 200 & GR2a_B1PondDepth < 300], na.rm = TRUE),
    B1_300_400_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0
    & GR2a_B1PondDepth > 300 & GR2a_B1PondDepth < 400], na.rm = TRUE),
    B1_gt_400_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0 &
    GR2a_B1PondDepth > 400], na.rm = TRUE),

    B1_0_100_pond_avg_depth_mm = mean(GR2a_B1PondDepth[GR2a_B1PondDepth > 0 &
    GR2a_B1PondDepth < 100], na.rm = TRUE),
```

```
    B1_100_200_pond_avg_depth_mm = mean(GR2a_B1PondDepth[GR2a_B1PondDepth > 100
    & GR2a_B1PondDepth < 200], na.rm = TRUE),
    B1_200_300_pond_avg_depth_mm = mean(GR2a_B1PondDepth[GR2a_B1PondDepth > 200
    & GR2a_B1PondDepth < 300], na.rm = TRUE),
    B1_300_400_pond_avg_depth_mm = mean(GR2a_B1PondDepth[GR2a_B1PondDepth > 300
    & GR2a_B1PondDepth < 400], na.rm = TRUE),
    B1_gt_400_pond_avg_depth_mm = mean(GR2a_B1PondDepth[GR2a_B1PondDepth > 400],
    na.rm = TRUE),

    B2_0_100_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0 &
    GR2a_B1PondDepth > 0 & GR2a_B1PondDepth < 100], na.rm = TRUE),
    B2_100_200_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0
    & GR2a_B1PondDepth > 100 & GR2a_B1PondDepth < 200], na.rm = TRUE),
    B2_200_300_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0
    & GR2a_B1PondDepth > 200 & GR2a_B1PondDepth < 300], na.rm = TRUE),
    B2_300_400_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0
    & GR2a_B1PondDepth > 300 & GR2a_B1PondDepth < 400], na.rm = TRUE),
    B2_gt_400_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0 &
    GR2a_B1PondDepth > 400], na.rm = TRUE),

    B2_0_100_pond_avg_depth_mm = mean(GR2a_B2PondDepth[GR2a_B2PondDepth > 0 &
    GR2a_B2PondDepth < 100], na.rm = TRUE),
    B2_100_200_pond_avg_depth_mm = mean(GR2a_B2PondDepth[GR2a_B2PondDepth > 100
    & GR2a_B2PondDepth < 200], na.rm = TRUE),
    B2_200_300_pond_avg_depth_mm = mean(GR2a_B2PondDepth[GR2a_B2PondDepth > 200
    & GR2a_B2PondDepth < 300], na.rm = TRUE),
    B2_300_400_pond_avg_depth_mm = mean(GR2a_B2PondDepth[GR2a_B2PondDepth > 300
    & GR2a_B2PondDepth < 400], na.rm = TRUE),
    B2_gt_400_pond_avg_depth_mm = mean(GR2a_B2PondDepth[GR2a_B2PondDepth > 400],
    na.rm = TRUE),

    B1_Pond_avg_temp = mean(GR2a_B1PondTemp),
    B2_Pond_avg_temp = mean(GR2a_B2PondTemp),

    B1_ponding_duration_hr = sum(GR2a_B1PondDepth > 0) / 12,
    B2_ponding_duration_hr = sum(GR2a_B2PondDepth > 0) / 12,

    #Ponding_Delay_hrs = as.double(difftime(min(TIMESTAMP[which(GR2a_B1PondDepth
    > 0 | GR2a_B2PondDepth > 0)]), min(TIMESTAMP), units = 'hours')),

    #B2_ponding_delay_hrs = as.double(difftime(min(TIMESTAMP[GR2a_B2PondDepth >
    0]), min(TIMESTAMP[GR2a_B1PondDepth > 0]), units = 'hours'))
  ) %>%
  # arrange events by start time
  arrange(Storm_Event_Start_Time) %>%
  # mutate creates new columns based on the columns just created by summarize.
  mutate(
    Event_Duration_hr = as.double(difftime(Storm_Event_End_Time,
    Storm_Event_Start_Time, units = "hours")),
```

```
    Average_Intensity_mm.hr = Total_Rainfall_mm / (Event_Duration_hr - 6),

    Inlet_Total_Volume_m3 = N8_Volume_m3 + N9_Volume_m3 + N10_Volume_m3,

    HWY_Total_Volume_m3 = N8_Hwy_Volume_m3 + N9_Hwy_Volume_m3,

    Outlet_Total_Volume_m3 = B1_Outlet_Volume_m3 + B2_Outlet_Volume_m3,

    Time_Since_Previous_Event_hr =
    as.double(difftime(Storm_Event_Start_Time,lag(Storm_Event_End_Time, 1),
    units = "hours")),

    #B2_preponding_volume_m3 = Ponding_Delay_hrs * 50 * 175 / 1000, # assuming
    50 mm/hr abstraction, 175 m^2 surface area
    # TODO: need to find unsaturated K

    #B2_ponding_infiltration_volume_m3 = B2_ponding_duration_hr *
    B2_pond_avg_recession_mm.hr * 175 / 1000,

    #Total_Volume_captured_m3 = B2_preponding_volume_m3 +
    B2_ponding_infiltration_volume_m3,

    #Drainage_Area_m2 = Total_Volume_captured_m3 / Total_Rainfall_mm,

  ) #%>%
# order table by Total_Rainfall_mm
#arrange(Total_Rainfall_mm)

# save all stats data
save(stats, file = "SMPA_EVENT_STATS_ALL.Rdata")

write.csv(stats, file = 'SMP_A_Event_Stats_ALL.csv')
```

Ponding recession plots and regression modeling:

````
---
title: "04 Recession Model"
author: "Andrew Kurzweil"
date: "1/12/2021"
output: html_document
---

```{r}
library(ggplot2)
library(plotly)
library(dplyr)
library(tidyr)

# load data files to KNIT environment
#load("K:/PennDOT DATA TEMP/DATA/SMPA_ALL_DATA.Rdata")
load("K:/PennDOT DATA TEMP/DATA/SMPA_EVENT_STATS_ALL.Rdata")

give.n <- function(x){
  return(c(y = median(x)*1.05, label = length(x)))
  # experiment with the multiplier to find the perfect position
}
```


```{r}
dat %>%
    arrange(TIMESTAMP) %>%
    filter(!is.na(stormID), B1PondDelta < 0) %>%
    filter(CS700Rain_in_TOT < 0.5) %>%
    filter(TIMESTAMP > ymd('2018-09-15')) %>%
    mutate(B1PondBin = cut(GR2a_B1PondDepth,breaks =
    c(0,100,200,300,400,500,1000)), .keep = 'all') %>%
    group_by(stormID, B1PondBin) %>%
    summarize(
      MeanRecession = 12 * mean(B1PondDelta, na.rm = TRUE),
      .groups = 'keep'
    ) %>%
  arrange(stormID) %>%
  ggplot(aes(x = B1PondBin, y = -MeanRecession)) +
  geom_boxplot() +
  stat_summary(fun.data = give.n, geom = "text", fun = median,
                  position = position_dodge(width = 0.75))
```


```{r event length histogram}
stats %>%
  filter(Total_Rainfall_mm > 12.5) %>%
````
````

```
  ggplot() +
  geom_histogram(aes(x = Event_Duration_hr), bins = 60) +
  #stat_function(fun = pchisq, args = list(df = 200))
  labs(x = "Event Duration (hr)", y = "Count")
```

```{r event length histogram}
stats %>%
  #filter(Total_Rainfall_mm > 12.5) %>%
  ggplot() +
  geom_histogram(aes(x = Total_Rainfall_mm), bins = 90) +
  #stat_function(fun = pchisq, args = list(df = 200))
  labs(x = "Total Rainfall (mm)", y = "Count")
```

```{r}
model = stats %>%
  filter(Total_Rainfall_mm > 6) %>%
  dplyr::select(
    B1_Pond_avg_temp,
    B1_pond_max_depth_mm,
    B1_pond_avg_recession_mm.hr,
    Event_Duration_hr,
    B1_ponding_duration_hr,
    Total_Rainfall_mm,
    Average_Intensity_mm.hr,
    Event_Date
  ) %>%
  #lm(data = ., B1_pond_avg_recession_mm.hr ~ B1_Pond_avg_temp - 1)
  lm(data = ., -B1_pond_avg_recession_mm.hr ~ - 1 + B1_Pond_avg_temp +
    B1_pond_max_depth_mm + B1_ponding_duration_hr + Total_Rainfall_mm +
    Average_Intensity_mm.hr)

summary(model)
plot(model)

#plot(log(stats$B1_Pond_avg_temp), -stats$B1_0_100_pond_avg_recession_mm.hr)
```


```{r}
cor.test(x = stats$B1_pond_avg_recession_mm.hr, y = stats$B1_Pond_avg_temp,
    method = 'kendall')
cor.test(x = stats$B1_pond_avg_recession_mm.hr, y = stats$B1_pond_max_depth_mm,
    method = 'kendall')
cor.test(x = stats$B1_pond_avg_recession_mm.hr, y = stats$Event_Duration_hr,
    method = 'kendall')
cor.test(x = stats$B1_pond_avg_recession_mm.hr, y =
    stats$B1_ponding_duration_hr, method = 'kendall')
```

```r
cor.test(x = stats$B1_pond_avg_recession_mm.hr, y = stats$Total_Rainfall_mm,
    method = 'kendall')
cor.test(x = stats$B1_pond_avg_recession_mm.hr, y =
    stats$Average_Intensity_mm.hr, method = 'kendall')
```

```{r}
EnvStats::kendallSeasonalTrendTest(stats$B1_pond_avg_recession_mm.hr, season =
    quarter(stats$Event_Date), year = year(stats$Event_Date))
```

```{r}
stats %>%
  ggplot() +
  geom_histogram(aes(x = yday(Event_Date)), bins = 36) +
  labs(x = "Ordinal Date", y = "Count")
# 150th day = May 30th
# 200th day = July 19th
```

Infiltration timing plots:

```
---
title: "05 Recession Timing"
author: "Andrew Kurzweil"
date: "2/4/2021"
output: html_document
---

```{r echo=FALSE}
library(ggplot2)
library(plotly)
library(dplyr)
library(tidyr)
library(bcp)
library(imputeTS)
library(data.table)

# load data files to KNIT environment
load("K:/PennDOT DATA TEMP/DATA/THESIS_DATA_112016_112020.Rdata")
load("K:/PennDOT DATA TEMP/DATA/SMPA_EVENT_STATS_ALL.Rdata")

give.n <- function(x) {
  return(c(y = median(x) * 1.05, label = length(x)))
  # experiment with the multiplier to find the perfect position
}

```

```{r warning=FALSE}
dat$stormID_48hr = assign.event.ids(dat$CS700Rain_in_TOT, 48)
dat$stormID_24hr = assign.event.ids(dat$CS700Rain_in_TOT, 24)
dat$stormID_12hr = assign.event.ids(dat$CS700Rain_in_TOT, 12)

dat$stormID_ponding = assign.ponding.event.ids(
  timestamps = dat$TIMESTAMP,
  rainfall = dat$CS700Rain_in_TOT,
  ponding = dat$GR2a_B1PondDepth,
  pre_rain_window_mins = 0,
  post_ponding_window_mins = 360,
  min_event_duration_mins = 360
)

dat$stormID_smW10 = assign.soil_moisture.event.ids(
  timestamps = dat$TIMESTAMP,
  rainfall = dat$CS700Rain_in_TOT,
  soil_moisture = dat$GR2a_WSoilMoist10,
  pre_rain_window_mins = 0,
  post_soil_moisture_window_mins = 360,
  min_event_duration_mins = 360
```
```

```
)
```
```



```{r STATS_GENERATOR, warning=FALSE}
stats_ponding <- dat %>%
  filter(!is.na(stormID_ponding)) %>% # remove NA stormID values (occurs when there's no event recorded)
  arrange(TIMESTAMP) %>%      # order the dataframe by timestamp so things get calculated in chronological order
  group_by(stormID_ponding) %>%       # group by stormID (ie perform all other operations on dat as grouped by stormID. this is how it becomes a pivot table.)
  # summarize creates new columns with the names you specify according to the formulas you provide.
  summarize(
    # start of event is the smallest Timestamp, end of event is the largest
    Storm_Event_Start_Time = min(TIMESTAMP),
    Storm_Event_End_Time = max(TIMESTAMP),
    # need just storm date (no time) for reports
    Event_Date = date(min(TIMESTAMP)),
    # Total rainfall is summed over event
    Total_Rainfall_mm = sum(CS700Rain_in_TOT),
    # Peak intensity is the highest single rainfall value (multipled by 12 for hourly rate, assuming 5 minute data)
    Peak_Intensity_mm.hr = max(CS700Rain_in_TOT * 12),
    Rainfall_Duration_min = sum(CS700Rain_in_TOT > 0) * 5,
    Peak_15min_Intensity_mm.hr = max(rainfall_sum_15min) * 4,
    Peak_30min_Intensity_mm.hr = max(rainfall_sum_30min) * 2,

    # max flowrates are just the instantaneous max values from previous calcs
    N8_Max_Flowrate_CMS = max(N8_In_Flow_CMS),
    N9_Max_Flowrate_CMS = max(N9_In_Flow_CMS),
    N10_Max_Flowrate_CMS = max(N10_In_Flow_CMS),
    N8_Hwy_Max_Flowrate_CMS = max(N8_Hwy_Flow_CMS),
    N9_Hwy_Max_Flowrate_CMS = max(N9_Hwy_Flow_CMS),

    # Volumes are RATE * TIME summed over the entire storm. Assumes 5 minute (300 second) data
    N8_Volume_m3 = sum(N8_In_Flow_CMS * 300),
    N9_Volume_m3 = sum(N9_In_Flow_CMS * 300),
    N10_Volume_m3 = sum(N10_In_Flow_CMS * 300),
    N8_Hwy_Volume_m3 = sum(N8_Hwy_Flow_CMS * 300),
    N9_Hwy_Volume_m3 = sum(N9_Hwy_Flow_CMS * 300),

    Flume_Volume_m3 = sum(Flume_Flow_LPS / 1000 * 300),
    Flume_Max_Flowrate_CMS = max(Flume_Flow_LPS / 1000),

```
```

```
    B1_Outlet_Volume_m3 = sum(B1_Outlet_Flow_CMS * 300),
    B2_Outlet_Volume_m3 = sum(B2_Outlet_Flow_CMS * 300),

    N8_peak_velocity_mm.s = max(GR2a_N8InletSpeed),
    N9_peak_velocity_mm.s = max(GR2a_N9InletSpeed),
    N10_peak_velocity_mm.s = max(GR2a_N10InletSpeed),
    N8_hwy_peak_velocity_mm.s = max(GR2a_N8HwySpeed),
    N9_hwy_peak_velocity_mm.s = max(GR2a_N9HwySpeed),

    # max (cumulative velocity) = sum (individual velocity data) ... not sure
    why I did it this way
    N8_total_vel = max(N8_sumvel),
    N9_total_vel = max(N9_sumvel),
    N10_total_vel = max(N10_sumvel),

    B1_pond_max_depth_mm = max(GR2a_B1PondDepth),
    B2_pond_max_depth_mm = max(GR2a_B2PondDepth),

    # these are not great estimates...poor definition of 'receding limb' &
    doesn't account for refilling
    B1_pond_avg_recession_mm.hr = 12 * mean(GR2a_B1PondDepth[which(TIMESTAMP >
    TIMESTAMP[which.max(GR2a_B1PondDepth)])] -
    lead(GR2a_B1PondDepth[which(TIMESTAMP >
    TIMESTAMP[which.max(GR2a_B1PondDepth)])]), na.rm = TRUE),
    B2_pond_avg_recession_mm.hr = 12 * mean(GR2a_B2PondDepth[which(TIMESTAMP >
    TIMESTAMP[which.max(GR2a_B2PondDepth)])] -
    lead(GR2a_B2PondDepth[which(TIMESTAMP >
    TIMESTAMP[which.max(GR2a_B2PondDepth)])]), na.rm = TRUE),

    # these account for possible refilling
    B1_0_100_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0 &
                                                              GR2a_B1PondDepth
    > 0 & GR2a_B1PondDepth < 100], na.rm = TRUE),
    B1_100_200_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0 &

    GR2a_B1PondDepth > 100 & GR2a_B1PondDepth < 200], na.rm = TRUE),
    B1_200_300_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0 &

    GR2a_B1PondDepth > 200 & GR2a_B1PondDepth < 300], na.rm = TRUE),
    B1_300_400_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0 &

    GR2a_B1PondDepth > 300 & GR2a_B1PondDepth < 400], na.rm = TRUE),
    B1_gt_400_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0 &
                                                              GR2a_B1PondDepth
    > 400], na.rm = TRUE),

    B1_0_100_pond_avg_depth_mm = mean(GR2a_B1PondDepth[GR2a_B1PondDepth > 0 &
                                                       GR2a_B1PondDepth < 100],
    na.rm = TRUE),
```

```
    B1_100_200_pond_avg_depth_mm = mean(GR2a_B1PondDepth[GR2a_B1PondDepth > 100 &
                                                         GR2a_B1PondDepth <
200], na.rm = TRUE),
    B1_200_300_pond_avg_depth_mm = mean(GR2a_B1PondDepth[GR2a_B1PondDepth > 200 &
                                                         GR2a_B1PondDepth <
300], na.rm = TRUE),
    B1_300_400_pond_avg_depth_mm = mean(GR2a_B1PondDepth[GR2a_B1PondDepth > 300 &
                                                         GR2a_B1PondDepth <
400], na.rm = TRUE),
    B1_gt_400_pond_avg_depth_mm = mean(GR2a_B1PondDepth[GR2a_B1PondDepth > 400],
na.rm = TRUE),

    B2_0_100_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0 &
                                                              GR2a_B1PondDepth
> 0 & GR2a_B1PondDepth < 100], na.rm = TRUE),
    B2_100_200_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0 &

GR2a_B1PondDepth > 100 & GR2a_B1PondDepth < 200], na.rm = TRUE),
    B2_200_300_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0 &

GR2a_B1PondDepth > 200 & GR2a_B1PondDepth < 300], na.rm = TRUE),
    B2_300_400_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0 &

GR2a_B1PondDepth > 300 & GR2a_B1PondDepth < 400], na.rm = TRUE),
    B2_gt_400_pond_avg_recession_mm.hr = 12 * mean(B1PondDelta[B1PondDelta < 0 &
                                                              GR2a_B1PondDepth
> 400], na.rm = TRUE),

    B2_0_100_pond_avg_depth_mm = mean(GR2a_B2PondDepth[GR2a_B2PondDepth > 0 &
                                                       GR2a_B2PondDepth < 100],
na.rm = TRUE),
    B2_100_200_pond_avg_depth_mm = mean(GR2a_B2PondDepth[GR2a_B2PondDepth > 100 &
                                                         GR2a_B2PondDepth <
200], na.rm = TRUE),
    B2_200_300_pond_avg_depth_mm = mean(GR2a_B2PondDepth[GR2a_B2PondDepth > 200 &
                                                         GR2a_B2PondDepth <
300], na.rm = TRUE),
    B2_300_400_pond_avg_depth_mm = mean(GR2a_B2PondDepth[GR2a_B2PondDepth > 300 &
                                                         GR2a_B2PondDepth <
400], na.rm = TRUE),
    B2_gt_400_pond_avg_depth_mm = mean(GR2a_B2PondDepth[GR2a_B2PondDepth > 400],
na.rm = TRUE),

    B1_Pond_avg_temp = mean(GR2a_B1PondTemp),
    B2_Pond_avg_temp = mean(GR2a_B2PondTemp),

    B1_ponding_duration_hr = sum(GR2a_B1PondDepth > 0) / 12,
    B2_ponding_duration_hr = sum(GR2a_B2PondDepth > 0) / 12,

```

```
    B1_max_ponding_timestamp = TIMESTAMP[which.max(GR2a_B1PondDepth)],
    B2_max_ponding_timestamp = TIMESTAMP[which.max(GR2a_B2PondDepth)],

    B1_end_ponding_timestamp = max(TIMESTAMP[which(GR2a_B1PondDepth > 10) + 1]),
    B2_end_ponding_timestamp = max(TIMESTAMP[which(GR2a_B2PondDepth > 10) + 1]),

    W10_starts_saturated = is_saturated(GR2a_WSoilMoist10[1]),

    W10_A_timestamp = as.POSIXct(
      identify_soil_moisture_inflection(
          timestamps = TIMESTAMP,
          data = GR2a_WSoilMoist10,
          point_number = 1
        ),
      origin = '1970-01-01 UTC',
      tz = 'UTC'
    ),
    W10_B_timestamp = as.POSIXct(
      identify_soil_moisture_inflection(
          timestamps = TIMESTAMP,
          data = GR2a_WSoilMoist10,
          point_number = 2
        ),
      origin = '1970-01-01 UTC',
      tz = 'UTC'
    ),
    W10_C_timestamp = as.POSIXct(
      identify_soil_moisture_inflection(
          timestamps = TIMESTAMP,
          data = GR2a_WSoilMoist10,
          point_number = 3
        ),
      origin = '1970-01-01 UTC',
      tz = 'UTC'
    ),
    W10_D_timestamp = as.POSIXct(
      identify_soil_moisture_inflection(
          timestamps = TIMESTAMP,
          data = GR2a_WSoilMoist10,
          point_number = 4
        ),
      origin = '1970-01-01 UTC',
      tz = 'UTC'
    ),

    W35_A_timestamp = as.POSIXct(
      identify_soil_moisture_inflection(
          timestamps = TIMESTAMP,
          data = GR2a_WSoilMoist35,
```

```
          point_number = 1
        ),
      origin = '1970-01-01 UTC',
      tz = 'UTC'
    ),
    W35_B_timestamp = as.POSIXct(
      identify_soil_moisture_inflection(
          timestamps = TIMESTAMP,
          data = GR2a_WSoilMoist35,
          point_number = 2
        ),
      origin = '1970-01-01 UTC',
      tz = 'UTC'
    ),
    W35_C_timestamp = as.POSIXct(
      identify_soil_moisture_inflection(
          timestamps = TIMESTAMP,
          data = GR2a_WSoilMoist35,
          point_number = 3
        ),
      origin = '1970-01-01 UTC',
      tz = 'UTC'
    ),
    W35_D_timestamp = as.POSIXct(
      identify_soil_moisture_inflection(
          timestamps = TIMESTAMP,
          data = GR2a_WSoilMoist35,
          point_number = 4
        ),
      origin = '1970-01-01 UTC',
      tz = 'UTC'
    ),

    W60_A_timestamp = as.POSIXct(
      identify_soil_moisture_inflection(
          timestamps = TIMESTAMP,
          data = GR2a_WSoilMoist60,
          point_number = 1
        ),
      origin = '1970-01-01 UTC',
      tz = 'UTC'
    ),
    W60_B_timestamp = as.POSIXct(
      identify_soil_moisture_inflection(
          timestamps = TIMESTAMP,
          data = GR2a_WSoilMoist60,
          point_number = 2
        ),
      origin = '1970-01-01 UTC',
```

```
      tz = 'UTC'
    ),
    W60_C_timestamp = as.POSIXct(
      identify_soil_moisture_inflection(
          timestamps = TIMESTAMP,
          data = GR2a_WSoilMoist60,
          point_number = 3
        ),
      origin = '1970-01-01 UTC',
      tz = 'UTC'
    ),
    W60_D_timestamp = as.POSIXct(
      identify_soil_moisture_inflection(
          timestamps = TIMESTAMP,
          data = GR2a_WSoilMoist60,
          point_number = 4
        ),
      origin = '1970-01-01 UTC',
      tz = 'UTC'
    ),

    .groups = "keep",

    #Ponding_Delay_hrs = as.double(difftime(min(TIMESTAMP[which(GR2a_B1PondDepth
    > 0 | GR2a_B2PondDepth > 0)]), min(TIMESTAMP), units = 'hours')),

    #B2_ponding_delay_hrs = as.double(difftime(min(TIMESTAMP[GR2a_B2PondDepth >
    0]), min(TIMESTAMP[GR2a_B1PondDepth > 0]), units = 'hours'))
  ) %>%
  # arrange events by start time
  arrange(Storm_Event_Start_Time) %>%
  # mutate creates new columns based on the columns just created by summarize.
  mutate(
    Event_Duration_hr = as.double(
      difftime(Storm_Event_End_Time, Storm_Event_Start_Time, units = "hours")
    ),
    Average_Intensity_mm.hr = Total_Rainfall_mm / (Event_Duration_hr - 6),

    Inlet_Total_Volume_m3 = N8_Volume_m3 + N9_Volume_m3 + N10_Volume_m3,

    HWY_Total_Volume_m3 = N8_Hwy_Volume_m3 + N9_Hwy_Volume_m3,

    Outlet_Total_Volume_m3 = B1_Outlet_Volume_m3 + B2_Outlet_Volume_m3,

    Time_Since_Previous_Event_hr = as.double(difftime(
      Storm_Event_Start_Time, lag(Storm_Event_End_Time, 1), units = "hours"
    )),

    B1_end_pond_to_W10_end_sat_hrs = as.double(
```

```
      difftime(W10_C_timestamp, B1_end_ponding_timestamp, units = "hours")
    ),

    W10_to_W35_hrs = as.double(difftime(W35_C_timestamp, W10_C_timestamp, units
    = 'hours')),

    W35_to_W60_hrs = as.double(difftime(W60_C_timestamp, W35_C_timestamp, units
    = 'hours')),

    #B2_preponding_volume_m3 = Ponding_Delay_hrs * 50 * 175 / 1000, # assuming
    50 mm/hr abstraction, 175 m^2 surface area
    # TODO: need to find unsaturated K

    #B2_ponding_infiltration_volume_m3 = B2_ponding_duration_hr *
    B2_pond_avg_recession_mm.hr * 175 / 1000,

    #Total_Volume_captured_m3 = B2_preponding_volume_m3 +
    B2_ponding_infiltration_volume_m3,

    #Drainage_Area_m2 = Total_Volume_captured_m3 / Total_Rainfall_mm,

  )
```

```{r}
dat %>%
  #filter(stormID == 'a8623845') %>% # 07-10-2020 ~4in storm
  #filter(stormID == '37d73f34') %>% # 10-11-2020 ~1.125 in storm
  #filter(stormID_ponding == '37cb6526') %>% # d90e8e97, 295a1a92, 37cb6526
  filter(stormID_smW10 == '6437be90') %>% #
  dplyr::select(TIMESTAMP, GR2a_WSoilMoist10, CS700Rain_in_TOT,
    GR2a_B1PondDepth) %>%
  arrange(TIMESTAMP) %>%
  mutate(
    diff = na_replace(GR2a_WSoilMoist10 - lag(GR2a_WSoilMoist10)),
    diff2 = na_replace(diff - lag(diff)),
    #diff2 = na_replace(((movavg(GR2a_WSoilMoist10, n = 4, type = 's') -
    lag(movavg(GR2a_WSoilMoist10, n = 4, type = 's'))) -
    lag(movavg(GR2a_WSoilMoist10, n = 4, type = 's') -
    lag(movavg(GR2a_WSoilMoist10, n = 4, type = 's'))))**0.25),
  ) %>%
# findpeaks(diff, npeaks = 3) )
  ggplot() +
  geom_line(aes(x = TIMESTAMP, y = GR2a_WSoilMoist10), color = 'red') +
  #geom_line(aes(x = TIMESTAMP, y = GR2a_B1PondDepth*0.05)) +
  #geom_line(aes(x = TIMESTAMP, y = CS700Rain_in_TOT*50), color = 'red') +
  #geom_line(aes(x = TIMESTAMP, y = diff), color = "blue") +
```

```
  geom_vline(aes(xintercept = TIMESTAMP[findpeaks(abs(diff), npeaks = 4,
    minpeakdistance = 2)[1,2]-1])) +
  geom_vline(aes(xintercept = TIMESTAMP[findpeaks(abs(diff), npeaks = 4,
    minpeakdistance = 2)[2,2]])) +
  geom_vline(aes(xintercept = TIMESTAMP[findpeaks(abs(diff), npeaks = 4,
    minpeakdistance = 2)[3,2]-1])) +
  geom_vline(aes(xintercept = TIMESTAMP[findpeaks(abs(diff), npeaks = 4,
    minpeakdistance = 2)[4,2]])) +
  labs(x = "Date/Time (EST)", y = "10cm Soil Moisture (VWC)")


```

```{r duration vs ponding gap}
ggplotly(
  stats_ponding %>%
  #stats_sm %>%
  filter(!is.na(B1_end_ponding_timestamp)) %>%
  filter(!is.na(W10_C_timestamp)) %>%
  filter(B1_end_ponding_timestamp < W10_C_timestamp) %>%
  filter(!is.na(B1_Pond_avg_temp) && B1_end_pond_to_W10_end_sat_hrs < 15)  %>%
  ggplot() +
  geom_point(aes(x = Event_Duration_hr, y = B1_end_pond_to_W10_end_sat_hrs)) +
  labs(x = 'Total Event Duration (hours)', y = 'End of Ponding to Desaturation
    of 10cm Soil (hours)') )
```

```{r ponding gap valid data}
stats_ponding %>%
  filter(!is.na(B1_end_ponding_timestamp)) %>%
  filter(!is.na(W10_C_timestamp)) %>%
  filter(B1_end_ponding_timestamp < W10_C_timestamp) %>%
  filter(!is.na(B1_Pond_avg_temp) && B1_end_pond_to_W10_end_sat_hrs < 30)
```


```{r}
#stats_sm %>%
stats_ponding %>%
  filter(!(is.na(B1_end_ponding_timestamp) || B1_end_pond_to_W10_end_sat_hrs ==
    Inf )) %>%
  filter(!is.na(W10_C_timestamp)) %>%
  filter(B1_end_ponding_timestamp < W10_C_timestamp) %>%
  filter(!is.na(B1_Pond_avg_temp))  %>%
  filter(B1_end_pond_to_W10_end_sat_hrs < 30) %>%
  ggplot() +
  geom_point(aes(x = Event_Duration_hr, y = B1_end_pond_to_W10_end_sat_hrs)) +
  labs(x = 'Total Event Duration (hours)', y = 'End of Ponding to Desaturation
    of 10cm Soil (hours)')
```

```
```{r rainfall vs ponding gap}
#stats_sm %>%
stats_ponding %>%
  filter(!(is.na(B1_end_ponding_timestamp) || B1_end_pond_to_W10_end_sat_hrs ==
    Inf )) %>%
  filter(!is.na(W10_C_timestamp)) %>%
  filter(B1_end_ponding_timestamp < W10_C_timestamp) %>%
  filter(!is.na(B1_Pond_avg_temp))  %>%
  filter(B1_end_pond_to_W10_end_sat_hrs < 30) %>%
  ggplot() +
  geom_point(aes(x = Total_Rainfall_mm, y = B1_end_pond_to_W10_end_sat_hrs)) +
  labs(x = 'Total Rainfall Depth (mm)', y = 'End of Ponding to Desaturation of
    10cm Soil (hours)')
```


```{r pond temp vs ponding gap}
#stats_sm %>%
stats_ponding %>%
  filter(!(is.na(B1_end_ponding_timestamp) || B1_end_pond_to_W10_end_sat_hrs ==
    Inf )) %>%
  filter(!is.na(W10_C_timestamp)) %>%
  filter(B1_end_ponding_timestamp < W10_C_timestamp) %>%
  filter(!is.na(B1_Pond_avg_temp))  %>%
  filter(B1_end_pond_to_W10_end_sat_hrs < 30) %>%
  ggplot() +
  geom_point(aes(x = B1_Pond_avg_temp, y = B1_end_pond_to_W10_end_sat_hrs)) +
  labs(x = 'Average Ponded Water Temperature (deg C)', y = 'End of Ponding to
    Desaturation of 10cm Soil (hours)')
```

```{r max pond depth vs ponding gap}
#stats_sm %>%
stats_ponding %>%
  filter(!(is.na(B1_end_ponding_timestamp) || B1_end_pond_to_W10_end_sat_hrs ==
    Inf )) %>%
  filter(!is.na(W10_C_timestamp)) %>%
  filter(B1_end_ponding_timestamp < W10_C_timestamp) %>%
  filter(!is.na(B1_Pond_avg_temp))  %>%
  filter(B1_end_pond_to_W10_end_sat_hrs < 30) %>%
  ggplot() +
  geom_point(aes(x = B1_pond_max_depth_mm, y = B1_end_pond_to_W10_end_sat_hrs)) +
  labs(x = 'Max Pond Depth (mm)', y = 'End of Ponding to Desaturation of 10cm
    Soil (hours)')
```


```{r ponding gap histogram}
```
```

```
#stats_sm %>%
stats_ponding %>%
  filter(!(is.na(B1_end_ponding_timestamp) || B1_end_pond_to_W10_end_sat_hrs ==
    Inf )) %>%
  filter(!is.na(W10_C_timestamp)) %>%
  filter(B1_end_ponding_timestamp < W10_C_timestamp) %>%
  filter(!is.na(B1_Pond_avg_temp))  %>%
  filter(B1_end_pond_to_W10_end_sat_hrs < 30) %>%
  #group_by('none') %>%
  #summarise(x = mean(B1_end_pond_to_W10_end_sat_hrs))
  ggplot() +
  geom_histogram(aes(x = B1_end_pond_to_W10_end_sat_hrs)) +
  labs(y = 'Count', x = 'End of Ponding to Desaturation of 10cm Soil (hours)')
```

````
```{r date vs ponding gap}
#stats_sm %>%
stats_ponding %>%
  filter(!(is.na(B1_end_ponding_timestamp) || B1_end_pond_to_W10_end_sat_hrs ==
    Inf )) %>%
  filter(!is.na(W10_C_timestamp)) %>%
  filter(B1_end_ponding_timestamp < W10_C_timestamp) %>%
  filter(!is.na(B1_Pond_avg_temp))  %>%
  filter(B1_end_pond_to_W10_end_sat_hrs < 30) %>%
  ggplot() +
  geom_point(aes(x = Event_Date, y = B1_end_pond_to_W10_end_sat_hrs, color =
    B1_Pond_avg_temp)) +
  labs(x = 'Event Date', y = 'End of Ponding to Desaturation of 10cm Soil
    (hours)')
```
````

````
```{r regression 1}
model = stats_ponding %>%
  filter(!(is.na(B1_end_ponding_timestamp) || B1_end_pond_to_W10_end_sat_hrs ==
    Inf )) %>%
  filter(!is.na(W10_C_timestamp)) %>%
  filter(B1_end_ponding_timestamp < W10_C_timestamp) %>%
  filter(!is.na(B1_Pond_avg_temp))  %>%
  filter(B1_end_pond_to_W10_end_sat_hrs < 30) %>%
  dplyr::select(B1_end_pond_to_W10_end_sat_hrs, B1_Pond_avg_temp,
    B1_pond_max_depth_mm, B1_pond_avg_recession_mm.hr, Event_Duration_hr,
    B1_ponding_duration_hr, Total_Rainfall_mm, Average_Intensity_mm.hr,
    stormID_ponding) %>%
lm(data = ., B1_end_pond_to_W10_end_sat_hrs ~ - 1 + . - stormID_ponding -
    B1_pond_max_depth_mm - B1_pond_avg_recession_mm.hr - Average_Intensity_mm.hr
    - Total_Rainfall_mm)
```
````

```
  #lm(data = ., B1_end_pond_to_W10_end_sat_hrs ~ B1_Pond_avg_temp)
summary(model)
anova(model)
```

```
# ----------------------------------------
# W10 to W35
# ----------------------------------------
```

```{r ponding gap valid data}
stats_ponding %>%
  filter(!is.na(W10_to_W35_hrs)) %>%
  filter(W10_to_W35_hrs > 0) # %>%
  filter(!is.na(B1_Pond_avg_temp) && B1_end_pond_to_W10_end_sat_hrs < 30)
```

```{r pond temp vs ponding gap}
#stats_sm %>%
stats_ponding %>%
  filter(!is.na(W10_to_W35_hrs)) %>%
  filter(W10_to_W35_hrs > 0) %>%
  ggplot() +
  geom_histogram(aes(x = W10_to_W35_hrs)) +
  labs(y = 'Count', x = 'Time between End of Saturation at 10cm and 35cm
    (hours)')
```

```{r pond temp vs ponding gap}
#stats_sm %>%
stats_ponding %>%
  filter(!is.na(B1_end_pond_to_W10_end_sat_hrs)) %>%
  filter(B1_end_pond_to_W10_end_sat_hrs > 0) %>%
  dplyr::select(B1_end_pond_to_W10_end_sat_hrs) %>%
  Hmisc::describe()
```

```{r pond temp vs ponding gap}
#stats_sm %>%
stats_ponding %>%
  filter(!is.na(W10_to_W35_hrs)) %>%
  filter(W10_to_W35_hrs > 0) %>%
```

```
  dplyr::select(W10_to_W35_hrs) %>%
  Hmisc::describe()
```

```{r pond temp vs ponding gap}
#stats_sm %>%
stats_ponding %>%
  filter(!is.na(W35_to_W60_hrs)) %>%
  filter(W35_to_W60_hrs > 0) %>%
  ggplot() +
  geom_histogram(aes(x = W35_to_W60_hrs)) +
  labs(y = 'Count', x = 'Time between End of Saturation at 35cm and 60cm (hours)')
```

```{r pond temp vs ponding gap}
#stats_sm %>%
stats_ponding %>%
  filter(!is.na(W35_to_W60_hrs)) %>%
  filter(W35_to_W60_hrs > 0) %>%
  dplyr::select(W35_to_W60_hrs) %>%
  Hmisc::describe()
```

```{r pond temp vs ponding gap}
#stats_sm %>%
stats_ponding %>%
  filter(!is.na(W10_to_W35_hrs)) %>%
  filter(W10_to_W35_hrs > 0) %>%
  ggplot() +
  geom_point(aes(x = B1_Pond_avg_temp, y = 25/W10_to_W35_hrs)) +
  labs(x = 'Average Ponded Water Temperature (deg C)', y = 'Infiltration Rate 10cm to 35cm (cm/hour)')
```

```{r}
cor.test(stats_ponding$W10_to_W35_hrs, stats_ponding$B1_Pond_avg_temp, method = 'kendall')
cor.test(stats_ponding$W10_to_W35_hrs, stats_ponding$B1_pond_max_depth_mm, method = 'kendall')
cor.test(stats_ponding$W10_to_W35_hrs, stats_ponding$Event_Duration_hr, method = 'kendall')
```

```
cor.test(stats_ponding$W10_to_W35_hrs, stats_ponding$B1_ponding_duration_hr,
    method = 'kendall')
cor.test(stats_ponding$W10_to_W35_hrs, stats_ponding$Total_Rainfall_mm, method =
    'kendall')
cor.test(stats_ponding$W10_to_W35_hrs, stats_ponding$Average_Intensity_mm.hr,
    method = 'kendall')

```


```{r}
stats_ponding %>%
  filter(!is.na(W10_to_W35_hrs)) %>%
  filter(W10_to_W35_hrs > 0) %>%
  ggplot() +
  geom_point(aes(x = Peak_Intensity_mm.hr, y = 25/W10_to_W35_hrs)) +
  labs(x = 'Peak Rainfall Intensity (mm/hr)', y = 'Infiltration Rate 10cm to
    35cm (cm/hour)')
```


Low Flow Plots


```{r}
dat %>%
  filter(stormID == '3182ac0a') %>% # 10-16-2020, 29.7mm
  #filter(stormID == '37d73f34') %>% # 10-11-2020 ~1.125 in storm
  #filter(stormID_ponding == '37cb6526') %>% # d90e8e97, 295a1a92, 37cb6526
  #filter(stormID_smW10 == '6437be90') %>% #
  dplyr::select(TIMESTAMP, N8_LF_Depth_Avg, CS700Rain_in_TOT) %>%
  arrange(TIMESTAMP) %>%
  ggplot() +
  geom_line(aes(x = TIMESTAMP, y = N8_LF_Depth_Avg), color = 'red') +
  geom_line(aes(x = TIMESTAMP, y = CS700Rain_in_TOT), color = 'blue') +
  labs(x = "Date/Time (EST)", y = "N9 Low Flow Depth (mm - Red) // Rainfall (mm
    - Blue)")
```


```{r}
dat %>%
  filter(stormID == '3182ac0a') %>% # 10-16-2020, 29.7mm
  dplyr::select(TIMESTAMP, N9_LF_Depth_Avg, CS700Rain_in_TOT) %>%
  dplyr::mutate(
    depth = N9_LF_Depth_Avg - 85,
```

```
    flow = -0.0000007 * depth**3 + 0.0002 * depth**2 + 0.0023 * depth
  ) %>%
  summarize(totalinflow = sum(flow*300),totalrainfall = sum(CS700Rain_in_TOT))
  arrange(TIMESTAMP) %>%
  ggplot() +
  geom_line(aes(x = TIMESTAMP, y = flow), color = 'red') +
  labs(x = "Date/Time (EST)", y = "N9 Low Flow (L/s)")
```