Evaporation and water storage in semi-arid region of Kitui, Kenya

Master thesis Hydrology

Name:	Timothy Tiggeloven
Student number:	2506768
Supervisor:	Ralph Lasage
Date:	19-07-2017

Abstract

In the near future climate change is expected to increase the severity of droughts in semi-arid regions. Moreover rainy seasons are expected to become shorter and more intense. Subsequently, water stress will increase, as inhabitants are dependent on rain-fed agriculture. Recent examples of increased water stress are the droughts in the Horn of Africa. Water harvesting is showing a potential adaptation measure to cope with the expected increase in water stress. In Kitui, Kenya water harvesting systems like sand dams, cisterns and open ponds are used to harvest water. Little is known about the influence of evaporation on these water harvesting systems. In this research sixteen methods of evaporation are compared with the residual of the energy balance and six methods are analyzed for their influence of modeled water harvesting systems (i.e. open pond and sand dam). The Priesley-Taylor method shows the best results (r^2 of 0.99, Nash-Sutcliffe efficiency of 0.78). Other methods showing agreeable results are Granger-Gray, Morton's CRAE, Brutsaert-Strickler, Penman-Monteith and Bowen Ratio Energy Balance. Six analyzed evaporation methods for modeled water storage show a deviation ranging between of -24 to 10 % and -27 to 8 % of evaporative fracture for open pond and sand dam respectively. The deviation of evaporative fracture equals up to 3.5 and 2.5 % of total water harvested, in open pond and sand dam respectively, for the analyzed period. The influence of the evaporation and selected evaporation methods, hence seem less important than other fluxes when looking into water harvesting systems.

Table of Contents

1 Introduction

Threats to water availability, demand and pollution are a worldwide problem, making water security and its indicators a much-discussed topic in scientific research. About 80% of the global population is suffering from these water security-related problems (Vörösmarty et al., 2010). Due to climate change, water security is projected to be stressed even more wherein the freshwater resources of semi-arid and arid regions are particularly exposed (Jiménez, 2014). Of the global population 35.5 % lives in all dry lands and about 14.4 % of the world's inhabitants in semi-arid regions (EMG, 2011). As consequence of dry seasons, the inhabitants in semi-arid regions already have to cope with periods of water scarcity. According to the IPCC (2012), global change is expected to increase the severity of droughts, intensify precipitation during a period of rainfall and wet seasons are expected to become shorter in semi-arid regions. Recent examples of extensive drought periods in semi-arid regions are found in the Horn of Africa (Nicholson, 2013), wherein millions of people suffer the consequences of food shortages due to crop failure.[1] Taking climate change into account, the frequency of water shortage will increase in the near future in semi-arid regions, making the need to grasp for solution to this worldwide problem even more.

To supply water to habitants in semi-arid regions during dry seasons, precipitation during wet seasons need to be stored with shorter and more intensive periods of rainfall. According to Lasage et al. (2008), sand dams are a successful adaption to cope with the expected increase in severity of droughts. This is in agreement with other studies (Quilis et al., 2009; Olufayo et al., 2009). A sand dam is a subsurface water storage system in which water flowing from an ephemeral river can be stored in the sandy sediments before the dam. The dam itself is a simple barrier in the drainage channel in which the sand dam regulates the water levels in the river sands as well as the surrounding area (Munywoki et al., 2004). Subsurface storage strongly reduces evaporation and contamination (Hut, 2008). SASOL, a local NGO in Kenya, constructed more than 700 sand dams serving more than 150.000 people in the Kitui County, showing the potential of this water harvesting structure and increasing the water availability in this semi-arid region (Munywoki et al., 2004). There are also other ways to store water, like an open pond and a water tank. An open pond is a simple form of surface storage in which the water is fully exposed to contamination and evaporation. While an open pond is described as an open system, a water tank is a closed system, which is fed by the runoff of rainwater from a catchment or rooftop. The performance of these water harvesting systems (WHS) is among others assessed by Lasage & Verburg (2015), Ngigi (2003) and politically addressed by Ertsen & Hut (2009).

Fluxes of rainfall, runoff and evaporation influences the water storage and related costs and benefits of the sand dams and other WHS. Little is known of the water losses due to evaporation on the performance of a sand dam and other WHS during

[1] http://www.fao.org/news/story/en/item/468941/icode/ (retrieved: 28-06-2017)

and after periods of rainfall. Recent studies argue that evaporation will not occur below 0.9 m beneath surface in sandy semi-arid areas with losses around 3 or 4 % of the total inflow (Love et al., 2011). Furthermore, Aerts et al. (2007) argues decreasing rates of evaporation with depth in a sandy soil. There are several methods of approaching evaporation, ranging from temperature-based techniques to highly complex methods including several meteorological factors. Selecting different methods of approaching evaporation will result in different outcomes of the modeled water storage. According to Allen et al. (1998), the more complex FAO Penman-Monteith is the recommended method of calculating evaporation. Best model prediction of Penman-Monteith and adaptions of the equation where found by Tanny et al. (2008) when comparing with open pan evaporation and are thus in agreement with Allen et al. (2008). While Szilagyi & Jozsa (2008) argue that aridity advection and complementary evaporation methods are better equipped when dealing with drought conditions. Next to the complex methods there are also methods that include a few meteorological parameters or even only temperature. While the more complex methods are often recommended, due to practical constraints the more simple methods are also often used. McMahoon et al. (2013) and Guo et al. (2016) made guides to calculate evaporation using various methods, wherein the former included a statistical overview of the various methods related to lysimeters, eddy covariance or other evaporation methods. Furthermore, Rosenberry et al. (2007) compared 15 evaporation methods. In this research various evaporation methods will be assessed and tested for their statistical differences. Subsequently, a sensitivity analysis of the various evaporation methods on water harvesting will be implemented. The main question of this research is:

What is the sensitivity of water harvesting systems for evaporation, considering multiple methods approaching evaporation rates?

In order to answer the main question, the impact of evaporation methods, ranging from simple to complex, on the performance of two water harvesting systems is assessed, using field data gathered in Kitui, Kenya and a water storage model. Chapter 2 gives a general overview of the study area Kitui, chapter 3 describes the methods used and chapter 4 elaborates on the results. The discussion is described in chapter 5 and lastly, chapter 6 concludes.

2 Study area

2.1 General overview of the Kitui District

The Kitui County is a semi-arid region located in Eastern Kenya. Kitui counts approximately 500 thousand inhabitants with an annual population growth rate of 2.6 % and a total land area of approximately 20,000 square kilometers. The altitude varies between 400 m and 1800 m above sea level and due to these local differences in height annual rainfall ranges from 500 mm to 1000 mm. Kitui is depicted in Figure 1, where elevation maps are shown of Kenya and Kitui county. South Eastern Kenya University (SEKU) and Nyumbani village are also shown, as these places are included in the research area.

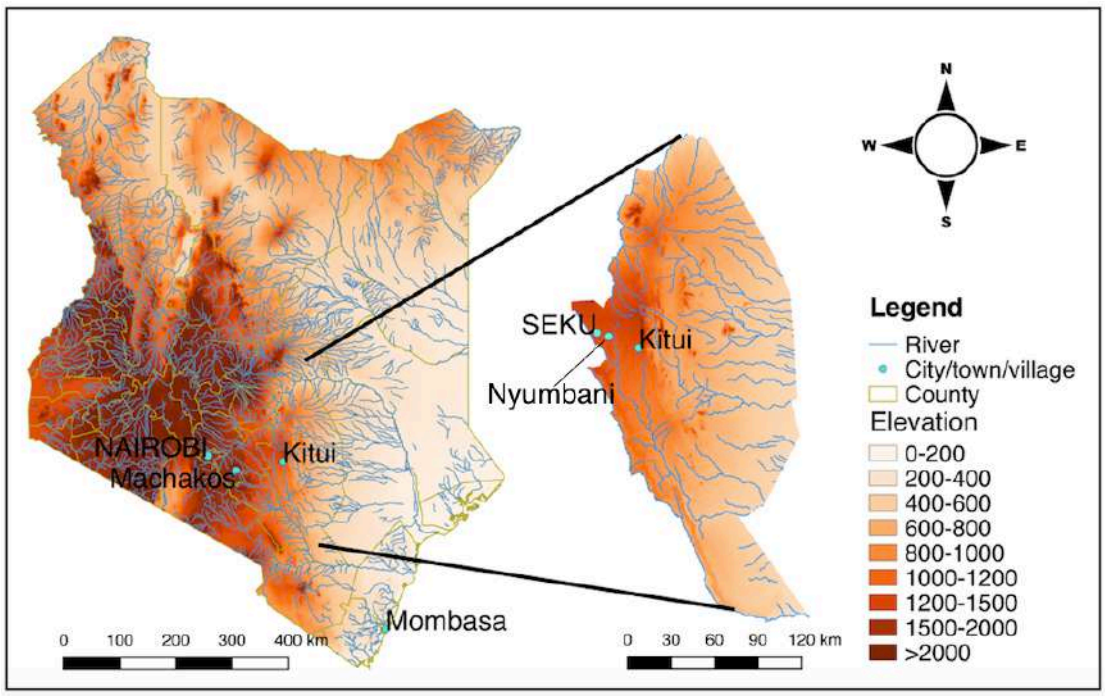

Figure 1: Elevation map of Kenya (left) and Kitui (right), where Nyumbani village and South Eastern Kenya University (SEKU) are indicated.

Rainfall mostly falls in two wet seasons spanning a couple of months per year. The first wet season is called 'the long rains' and falls between late March and early June. The second wet season is the so-called 'short rains' and falls between late Octobers throughout November. The long rains are highly erratic and reliable while the short rains are more consistent. Based on historical rainfall data from the KNMI climate explorer, a weather station in Kitui measured 990 mm rainfall per year on average. The first wet season contributes 468 mm on average and the second wet season 517 mm of rainfall. As shown in Figure 2, most of the rain falls in April and November with average amounts of 228 and 300 mm respectively. Figure 3 depicts the sum of rainfall per wet season with standardized anomalies based on historical rainfall data obtained from KNMI climate explorer. The figure clearly shows the large deviations in rainfall per year with the lowest point a standardized anomaly of -3 and 217 mm of rainfall in 1983. The highest point is reached in 1968 with standardized anomaly of 3 and close to 1800 mm of rainfall. The amount of rainfall is of great

importance for the inhabitants as their water supply and agriculture is mostly dependent on the rainfall. The mean temperature per year is 21 ºC and the potential evaporation is approximately 1500 to 1600 mm per year (Lasage et al., 2008).

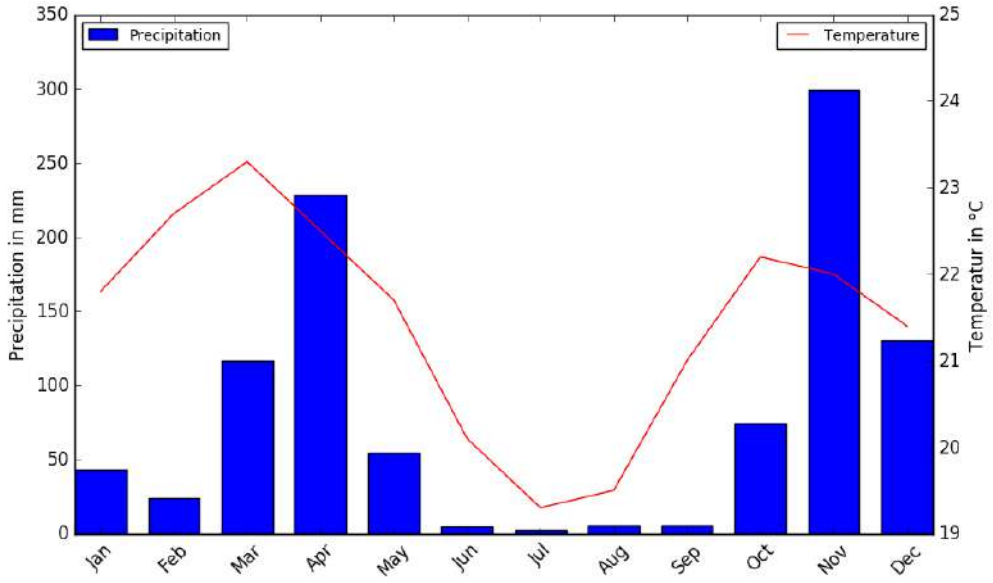

Figure 2: Mean monthly rainfall and temperature in Kitui, Kenya for the period 1904-1990 (Source: KNMI Climate Explorer and climate-data.org).

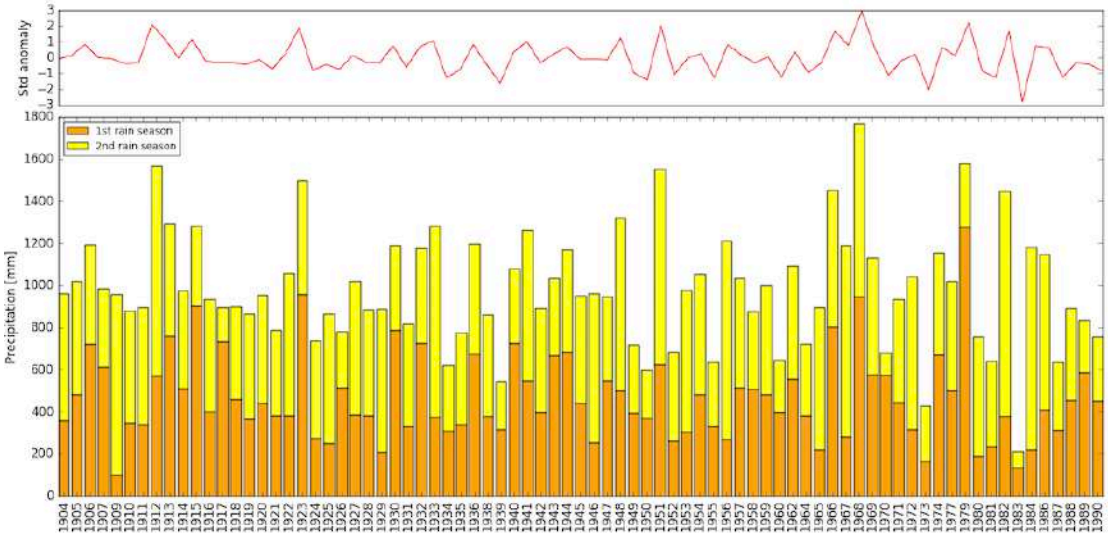

Figure 3: Historical rainfall data for the first and second wet season for the period 1904-1990 with standardized anomalies on top (Source: KNMI climate explorer).

2.2 Vegetation in Kitui

The Kitui County mostly consists of savannas and drought deciduous woodlands. Variety of vegetation and species distribution in the semi-arid region depends on topographical features as valley, slope or hillside (Tanaka et al., 2000). The most dominant tree species are the Lannea triphylla and Commiphora Africana. The Acacia tree species has become valuable and scarce due to usefulness of firewood and timber (Hayashi, 1996). Natural grasses and shrubs of lantana camara and other species are abundant in the area (Munywoki et al., 2004). Figure 4 depicts the

vegetation in the study area at the end of November 2016. After some rainfall, natural grasses are starting to grow on the otherwise dry soil.

Figure 4: Vegetation in the study area at SEKU ground.

2.3 Water availability, local facilities and economy of Kitui

According to the KNBS (2013), 60 % of the district of Kitui was beneath the country's poverty line, making it one of the poorest regions of Kenya. Although the main source of income of 80 % of the population is rain fed agriculture, only 2 % of the area has high agricultural potential and 32 % of medium potential (KNBS, 2015). Irrigated agriculture only takes place on small plots on the riverbank and in 2004/2005 approximately 50 % of the population in Kitui received food aid. Other sources of income are charcoal burning, brick making and basket breading (Lasage et al., 2008). Water availability is scarce in the region and only 6 % of the population of Kitui has access to potable water, leaving inhabitants to walk up to 20 kilometers for clean water. Furthermore, only 45 % of the population has access to water for domestic use while fewer have access to water that is fit to drink (Lasage et al., 2008). An alternative natural water source for many inhabitants in rural Kenya is water storage with rooftop harvesting or scooping holes in sediments of rivers in order to reach the groundwater. Inhabitants that are relying on rain fed water storage have to cope with declining and depleting water levels at the end of the dry seasons. In Figure 5 a scoop-hole is depicted at the end of a dry season leaving inhabitant to dig a couple of meters in order to reach the depleting source of water.

Figure 5: Scoping hole at the end of the dry season in October 2016.

2.4 Hydrology and sand dams in Kitui

Kitui counts two perennial rivers. The Tana River is located at the north boundary of the region and the Athi River at the southwestern boundary. Both rivers discharge to the Indian Ocean and most of the Kitui district consists of the catchment area of the Tana River. During rainy seasons, precipitation-fed ephemeral rivers start flowing in the region. These temporarily flowing rivers are for most of the inhabitants in the region more important. Erratic rainfall patterns in combination with poor drainage of the abundant clayey soils in Kitui results in a scarcity of surface water and groundwater resources (Pauw et al., 2008). The Sahelian Solution Foundation (SASOL), a local NGO founded in 1992, together with local communities in Kitui has set up a rural water conservation program and constructed over 700 sand dams attempting to tackle the water scarcity of the dry periods in the region. A sand dam is a simple concrete dam placed in a drainage channel in which water is forced to infiltrate the river sediment and can be used as storage for later uses. The sandy sediment of the river helps purify the infiltrating water and eliminates most of the evaporation. Reaching the water is normally done with a pump, but inhabitants also make use of scoop holes.

Figure 6: Sand dam at Nyumbani village in Kitui county.

3 Methods

Meteorological input is needed in order to calculate water storage and different methods of evaporation. In Kitui, Kenya an automatic weather station (AWS) has been set up and installed in order to measure different meteorological factors. The raw input of the AWS has been processed to daily climate data and other meteorological factors, not included in the measurements, are calculated with the processed data. Daily climatological data has been obtained for the period 7th of October 2016 until 21st of December 2016, 7th of April 2017 until 19th of April 2017, and 18th of May 2017 until 22nd of May 2017, with a gap of 25th and 26th of October 2016. The small and large gaps are due to measurement failure. In total 90 days of daily climate data is obtained. Equation 1 depicts the direct forcings of the energy balance. Taking the evaporation term as residual has closed the energy balance. This has been used as reference point for statistical analysis for the different evaporation methods, which are assessed in this research. The results of calculating evaporation using these different methods have been used as input for a water storage model to obtain a sensitivity of water storage to the different methods of evaporation. Calculations have been made based on two WHS and the results are statistically analyzed in order to validate if the evaporation methods differ significantly. Figure 7 gives a schematic overview of the methods section and the sections in which they are elaborated. The Python scripts used for the models and processing of the data can be found in the appendix.

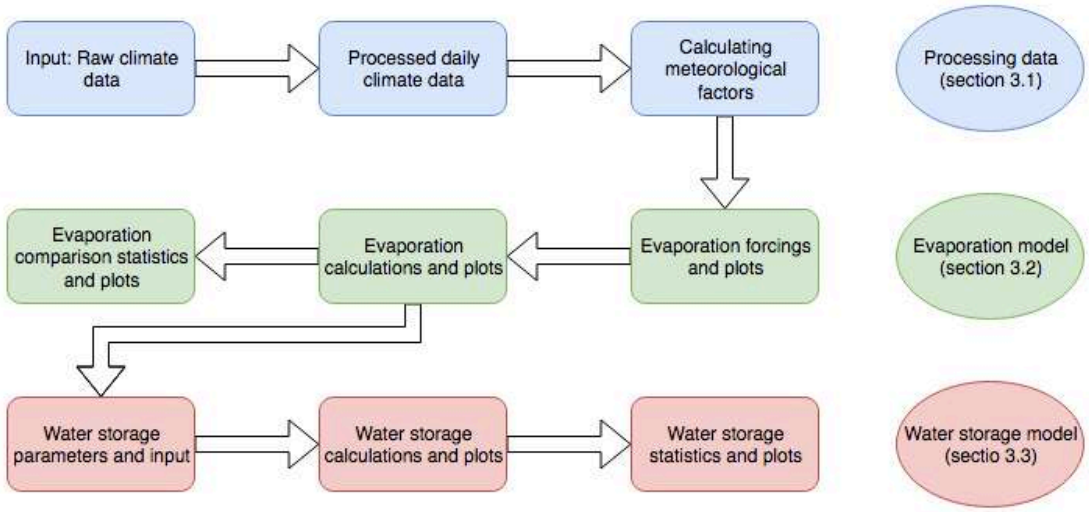

Figure 7: Schematic overview of the methods used. (Meteorological factors: e.g. latent heat of vaporization, actual and saturation vapor pressure, and aerodynamic resistance).

3.1 Processing data

The studies of calculating evaporation using various methods are highly discussed topics in scientific research (McMahon et al., 2013; Guo et al., 2016; Rosenberry et al., 2007). In these studies experimental designs and theoretical equations are set up to calculate reference crop, potential and actual evaporation. Most of the evaporation methods are based on the energy balance (McMahoon et al., 2013; Tanny et al., 2007). The energy balance is depicted in Equation 1 and is the general form of the energy budget in terms of a flux:

$$R_n - \lambda E - H - G = 0 \quad (1)$$

where R_n is the specific flux of net incoming radiation, λE the latent heat flux, H the specific flux of sensible heat into the atmosphere, and G the specific flux of heat conducted into the earth (Brutsaert, 2013; Allen et al., 1998). The latter can be assumed to be zero when dealing with daily time steps (Allen et al., 1998). The AWS measures meteorological factors needed to calculate evaporation and other terms of the energy balance summarized in Table 1. Some meteorological factors are not directly measured with the AWS, but can be calculated using measured data (e.g. vapor pressure and aerodynamic resistance). Closing the balance and dividing the latent heat flux by the latent heat of vaporization allows one to calculate the evaporation as residual in mm day-1. Figure 8 gives an overview of influencing factors of Equation 1. According to Guo et al. (2016), the term water advected energy only applies for open-water bodies and can be neglected for this research. Radiation, ground heat exchange and sensible heat flux are depicted within the energy balance box and both influence evaporation. Resistances, vapor gradient and wind moreover influence Mass transfer of water vapor.

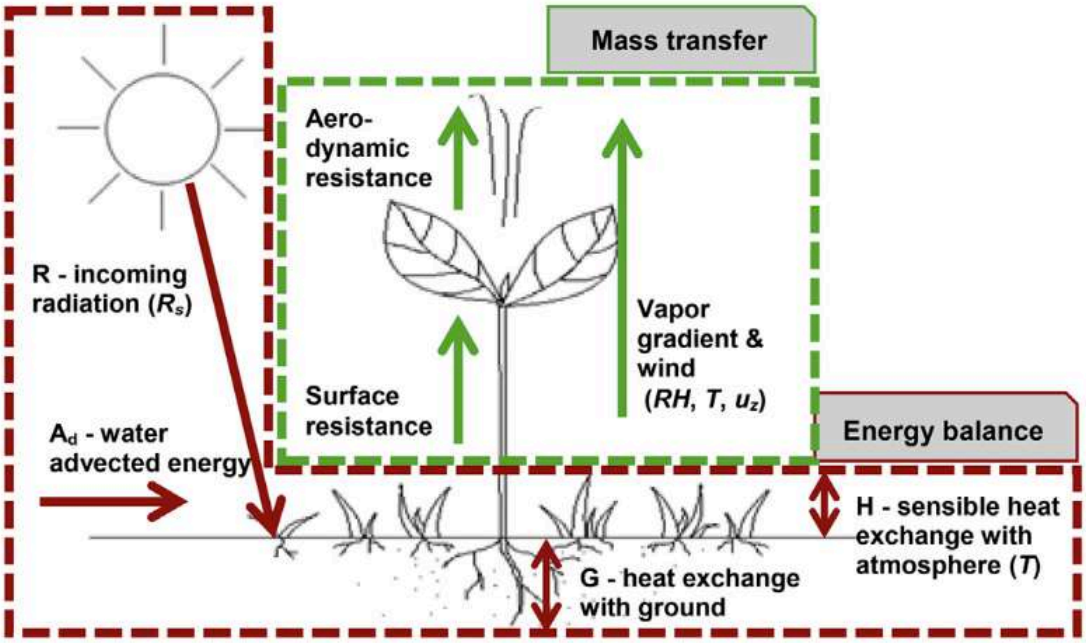

Figure 8: Overview of important forcings of evaporation (Adopted from: Guo et al., 2016).

3.1.1 Measuring meteorological factors

An AWS has been set up to log a continuous series of measurements of meteorological factors in the study area. At the core of the station a CR1000 data logger is installed and connected to the various devices that measures the factors. The AWS is attached to a solar panel in order to provide a power supply and keep the data logger operational. Every five seconds a measurement is done by the attached instrument and every 15 minutes data is logged in the system. In most cases the measurement logged is the average of the previous 15 minutes. For rainfall the measurements are summed. Table 1 summarizes the instruments used and their specifications.

Table 1: Summary of the measuring instruments

Component	Instrument	Units	Accuracy	Resolution	Height
Rainfall	Campbell Scientific ARG100 Rain Collector	mm	±2 % up to 50 mm hr^{-1}	0.1 mm	1 m
Air temperature	Vaisala HMP45C Temperature and Relative Humidity Probe	ºC	±0.2 °C	0.1 °C	1.25 m and 2 m
Relative humidity	Vaisala HMP45C Temperature and Relative Humidity Probe	%	±2 %	0.10 %	1.25 m and 2 m
Solar radiation	Skye SKS1110 Pyranometer	W m^{-2}	< ±0.2 %	0.1 W m^{-2}	1.5 m
Net radiation	Kipp & Zonnen NR-LITE 2 Net radiometer	W m^{-2}	< ±3 % (at 1000 W m^{-2})	0.1 W m^{-2}	1.5 m
Wind speed	DS-2 Sonic Anenometer	m s^{-1}	±3 %	0.01 m s^{-1}	2.25 m
Soil temperature	Campbell Scientific 107 Temperature Probe	ºC	±0.2 °C	0.1 °C	0.05 m depth
Soil heat flux	Campbell Scientific Hukseflux HFP01 Heat Flux Plate	W m^{-2}	Within -15 % to +5 %	0.1 W m^{-2}	0.05 m depth
Soil water content	Campbell Scientific CS616 Water Content Reflectometers	%	±2.5 %	0.10%	0.05 and 0.10 m depth

The AWS has been set up to meet the WMO specifications (Jarraud, 2008). A rain gauge tipping bucket measures rainfall in mm at a height of 1 m. As depicted in Figure 9 there are two temperature and relative humidity instruments at the height of 1.25 m and 1.75 m, which agrees with the WMO standard between 1.25-2.00 m. At 1.5 m height an arm is perpendicularly attached to the main pole and carries two radiation instruments. Due to potential influences of the main pole, the arm is 1 m long. At the end of the arm the net radiometer is attached and in the middle the solar radiometer. There are two plots in which measurements are done, i.e. the soil plot and sand dam sediment plot. The latter plot contains a soil temperature probe, soil water content reflectometer and a soil heat flux all placed at a depth of 0.05 m. The soil plot contains the same but with a second soil water content reflectometer at the depth of 0.1 m. More information about the devices used is found in the appendix. The location, field equipment and description of the automatic weather station are depicted in Figure 9.

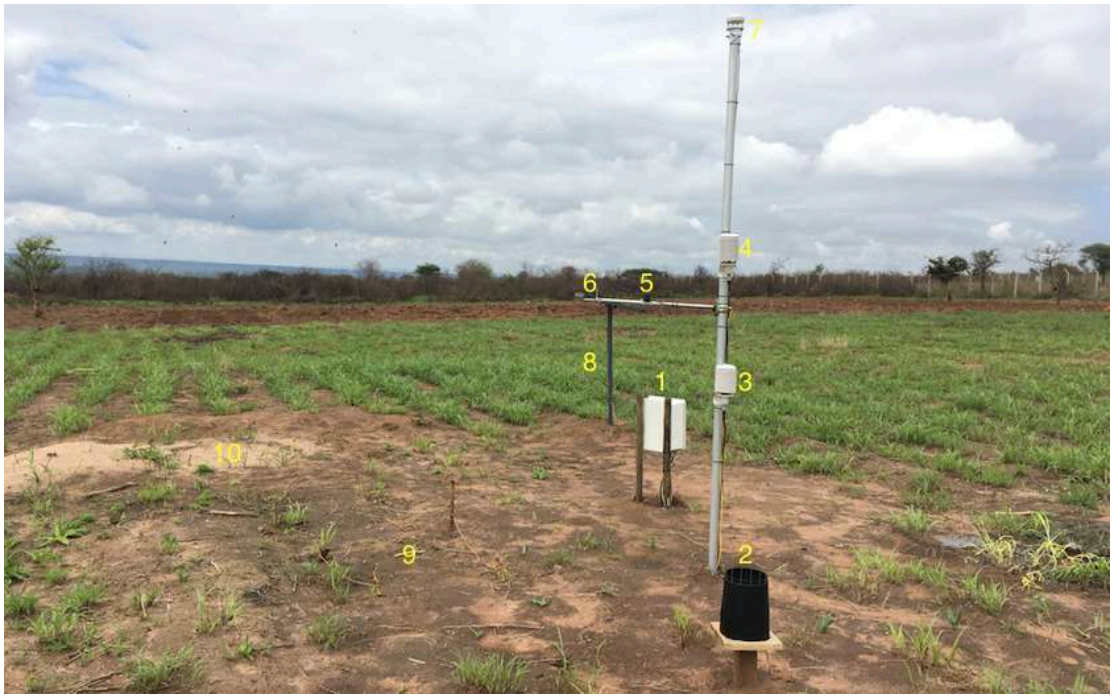

Figure 9: Field equipment and surroundings of the automatic weather station at SEKU ground. The following devices and plots are numbered; data logger (1), rain gauge (2), temperature and humidity device (3 & 4), solar radiation meter (5), net radiation meter (6), wind speed and direction on top of the pole (7), solar panel on top of dark gray pole behind arm of net radiation meter (8), soil plot (9), and sand dam sediment plot (10).

3.1.2 Processing climate data

The 15-minute intervals between the measurements are processed to daily data, where the measurement day starts at 08:00. For the factors wind speed, solar radiation, net radiation, soil water content and soil heat flux the daily averages are calculated. Temperature and humidity are processed to daily data by summing the daily maximum and minimum divided by 2. Precipitation measurements are summed to daily data. Because most of the evaporation techniques require radiation in MJ m^{-2}, the solar, net radiation and soil heat flux are converted from W m^{-2} to MJ m^{-2}. When measurements fail, the days that possess gaps are discarded and, furthermore, measurements with NaN are replaced by daily mean, maximum or minimum. The photoperiod hours per day and extraterrestrial radiation has been calculated by the following set of equations:

$$d_r = 1 + 0.033 \cos(2\,J\,\pi/365) \quad (2)$$

$$\delta = 0.409 \sin(2\,J\,\pi/365 - 1.39\,) \quad (3)$$

$$\varphi = latitude\ \pi/180 \quad (4)$$

$$\omega = \cos^{-1}(\tan\phi \tan\delta) \quad (5)$$

$$L = 24\,\omega/\pi \quad (6)$$

, where J is the Julian day and number of the day of the year, d_r the inverse relative distance Earth-Sun, δ is the sun declination, φ the latitude in radians and ω is the hour angle at sunrise or sunset (Allen et al, 1998; Iqbal, 2012; Guo et al., 2016). The hours per day, L, is needed as input in order to calculate an evaporation models specified in section 3.2. The set of Equations 2 to 6 is hereafter referred to as the

sunrise equation. With the sunrise equation, the extraterrestrial radiation can be calculated by filling in the following equation:

$$R_e = \frac{24(60)}{\pi} G_{sc} d_r [\omega \ \sin(\varphi) \ \sin(\delta) + \cos(\varphi) \cos(\delta) \sin(\omega)] \quad (7)$$

, where G_{sc} is the solar constant and R_e the extraterrestrial radiation in MJ day^{-1} (Allen et al, 1998). Due to measurement failure of net radiation during the period of measuring in 2017, the net radiation is estimated by filling in the set of equations 8 to 11.

$$R_{so} = (0.75 + 2(10^{-5})z)R_e \quad (8)$$

$$R_{ns} = (1-\propto)R_s \quad (9)$$

$$R_{nl} = \sigma \left[\frac{{T_{max,K}}^4 + {T_{min,K}}^4}{2}\right] (0.34 - 0.14\sqrt{e_a}) \left(1.35 \frac{R_s}{R_{so}} - 0.35\right) \quad (10)$$

$$R_n = R_{ns} - R_{nl} \quad (11)$$

, where R_{so} is the clear-sky solar radiation, R_{ns} is the net solar or net shortwave radiation, $\propto$ is the albedo of the study area, R_s is the measured solar radiation, R_{nl} is the net longwave radiation, e_a is the actual vapor pressure described in the next section and R_n the net radiation (Allen et al., 1998). $T_{max,K}$ and $T_{min,K}$ are the maximum and minimum temperature of the day in Kelvin degrees and the average albedo is computed, for the period where net radiation measures are available, by taking it as the residual and filling in the set of equation. An average value of 0.39 is estimated.

3.1.3 Calculating meteorological factors

Next to the measured meteorological factors, other factors are used for calculating evaporation using different methods. Resistances and vapor pressures are examples of such influencing factors on evaporation. This section describes the often-used factors as input for evaporation not measured, but calculated with constants and measured fluxes. An overview of all the measured and calculated factors required as input for the evaporation model is shown in the Figure 10.

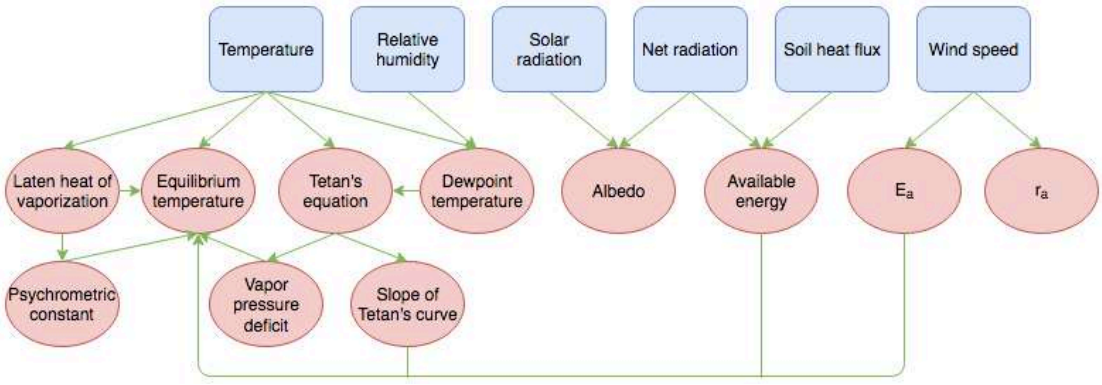

Figure 10: Overview of the measured meteorological factors (blue) and calculated meteorological factors (red). The green arrows depict the flow of input.

Wind speed is an important factor in meteorology and influences the evaporation rate by altering the aerodynamic resistance. At the surface wind speed is slowest and increases with height. Due to these differences in wind speed at different heights, a standard height of 2 m is required in order to calculate evaporation (Allen et al., 1998). To adjust wind speed data from the measured 3.25 m height to 2 m height, the following equation is used:

$$U_2 = U_z \frac{4.87}{\ln(67.8z - 5.42)} \tag{12}$$

, where z and U_z is the wind speed measured height and measurements respectively (Allen et al., 1998). With wind speed data at a height of 2 m, the aerodynamic resistance factor can be calculated with equation 13.

$$r_a = \frac{ln\left[\frac{z_m - d}{z_{om}}\right] ln\left[\frac{z_h - d}{z_{oh}}\right]}{k^2 U_z} \tag{13}$$

, where z_m and z_h are the height of wind speed and relative humidity measurements respectively, d is the zero plane displacement height, z_{om} and z_{oh} are the roughness length governing momentum transfer and transfer of heat and vapor respectively, and k the von Karman's constant. The aerodynamic resistance r_a is the resistance to transfer of heat and water vapor from the surface into the air above the surface and is expressed in s m^{-1} (Allen et al., 1998). Next to the aerodynamic resistance there is also surface resistance and is approximated by the following equation:

$$r_s = \frac{r_l}{LAI} \tag{14}$$

, where r_l is the bulk surface resistance of the leaf and LAI the leaf area index. The bulk surface resistance is for the grass reference surface is approximated to 70 s m^{-1} (Allen et al., 1998). The temperature dew point is calculated with the following equation:

$$T_d = T_K \left[1 - \frac{T_K \ln\left(\frac{RH}{100}\right)}{e_v / R_w}\right]^{-1} - 273.15 \tag{15}$$

, where $\boldsymbol{T_K}$ is the temperature in Kelvin degrees, RH is the relative humidity, e_v is the enthalpy of vaporization, and R_w is the gas constant for water vapor (Lawrence, 2005). The latent heat of vaporization is calculated using equation 16:

$$\lambda = 2.501 - (2.361 \times 10^{-3}) T_a \tag{16}$$

, where T_a is temperature in degrees Kelvin and λ the latent heat of vaporization in MJ kg^{-2} (Allen et al., 1998). With the latent heat of vaporization, the psychrometric constant can be computed with the following calculation:

$$\gamma = \frac{c_p P}{\varepsilon \lambda} \tag{17}$$

, where c_p is the specific heat at constant pressure, P is the atmospheric pressure, ε is the ratio molecular weight of water vapor and dry air and γ the psychrometric constant in kPaºC^{-1}. As shown in Figure 10, the Teten's equation is an important factor and allows calculations of saturation, actual and equilibrium vapor pressure. The saturation vapor pressure at a specific temperature can be calculated with the Teten's equation:

$$e°(T) = 0.6108 e^{\left[\frac{17.27 T_a}{T_a + 237.3}\right]} \tag{18}$$

$$e_s = \frac{e°(T_{min}) + e°(T_{max})}{2} \tag{19}$$

To calculate the mean saturation vapor pressure e_s, the average between $e°(T_{min})$ and $e°(T_{max})$ is calculated (Monteith & Unsworth, 2007). Actual vapor pressure is derived from relative humidity and calculated with one of the following formulas:

$$e_a = \frac{e°(T_{min})\frac{RH_{max}}{100} + e°(T_{max})\frac{RH_{min}}{100}}{2} = e°(T_d) \tag{20}$$

Both saturation and actual vapor pressure are calculated in kPa (Allen et al., 1998; Guo et al., 2016). In this study actual vapor pressure is calculated with the second formula. Once actual and saturation vapor pressure are computed, the vapor pressure deficit is be expressed as:

$$VPD = e_s - e_a \tag{21}$$

The slope of the saturation vapor pressure curve in kPaºC^{-1} is calculated with the following equation (Allen et al., 1998):

$$\Delta = \frac{4098\, e_s}{(T_a + 237.3)^2} \tag{22}$$

The Penman model (Penman, 1948) and a couple of derivations of this model makes use of a wind function as aerodynamic component in the evaporation calculation and is later revised as follows:

$$E_a = (1.313 + 1.381 U_2) VPD \tag{23}$$

, where the wind function is denoted between brackets (Penman, 1956). The wet environment surface temperature or equilibrium temperature, as described by

Szilagyi & Jozsa (2008), is needed in order to calculate evaporation models of Szilagyi-Jozsa and Morton CRAE (Morton, 1983). It can be estimated iteratively by the following equation since all other terms are known:

$$\frac{R_n}{\lambda E_{PEN}} = 1 + \frac{\gamma(T_e - T_a)}{e_s^* - e_a} \tag{24}$$

, where T_e is the equilibrium temperature and e_s^* is the saturation vapor pressure at T_e (Szilagyi & Jozsa, 2008; Guo et al., 2016). The temperature equilibrium is assumed to be lower than the air temperature. Sensible heat flux in Wm^{-2} is calculated with the following equation:

$$H = c_p P \frac{(T_s - T_a)}{r_a} \tag{25}$$

, where T_s and T_a are surface and air temperature respectively (Monteith & Unsowrth, 2007; Liu et al., 2007).

3.2 Evaporation model

The energy balance of Equation 1 is used as a starting point for deriving evaporation models. More complex equations include the fluxes of the energy balance directly or indirectly and more simple equations are derived empirically or calibrated with evaporation pans or lysimeters. In this model the evaporation methods are based on these energy fluxes or forcings. Net radiation and soil heat flux are measured directly with the AWS. The latent heat flux can be calculated with the evaporation methods or can be calculated by filling in the energy balance. To convert energy expressed in MJ day^{-1} to water depth in mm day^{-1}, the latent heat of vaporization is used as conversion factor (Allen et al., 1998). The calculated evaporation flux by closing the energy balance will be used as a reference for the evaporation methods. This method is different than the more commonly used Bowen Ratio Energy Balance in semi-arid areas and is also used by Yamanaka et al. (2007) and Duan & Bastiaanssen (2017). The advantage of this method is that it is easier to compare the observed and calculated energy fluxes of the evaporation methods. As stated by Tamanaka et al. (2007), the disadvantage of this method is that the errors/biases in calculating the forcings of the energy balance will cumulate in the latent heat flux. Furthermore, Tanny et al. (2007) also includes the closure of the energy balance in their research. This method will further be referred as the energy balance closure (EBC) method. There are several methods used in literature in which to calculate evaporation and for this research sixteen methods have been selected (McMahoon et al., 2013; Guo et al., 2016). These methods will be categorized as temperature-based, radiation-based, more complex techniques with additional factors as wind and resistances-based, and lastly, complementary relationship derivations of actual evaporation. While the former two are dealing mostly with empirically or calibrated techniques for specific locations, the latter are more directly derived from the energy balance. All evaporation methods are calculated in mm day^{-1} and the terms E and ET

are expressed as evaporation and (reference crop) evapotranspiration respectively. An overview of the evaporation models and requirements of meteorological measurements are shown in the Figure 11.

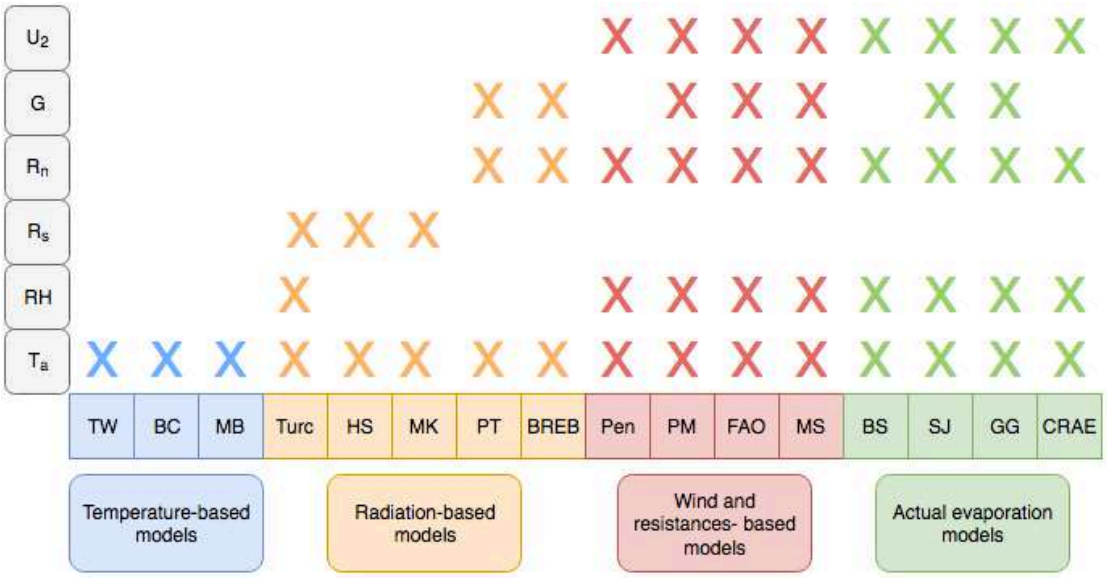

Figure 11: Overview of the four categories of evaporation models and requirements of meteorological measurements for computation. The abbreviations for the evaporation models are mentioned for each model in section 3.2.1-4.

3.2.1 Temperature-based

One of the earlier works in estimating evaporation, and is still used, is the Thornthwaite method. This method introduced temperature as a parameter for evaporation (Monteith, 1994). The Thornthwaite formula is expressed as follows:

$$E_{TW} = 16\frac{L}{360}\left(10\frac{T_a}{I}\right)^{\propto_{TW}} \tag{26}$$

, where L divided by 360 is a correctional term to transform monthly to daily evaporation, T_a is temperature, I is a thermal index, and $\propto_{TW}$ is a function of I (Thornthwaite, 1948; Pereira & Pruitt, 2004). Shortly after the publication by Thornthwaite, the Blaney-Criddle temperature based evaporation method surfaced in the scientific community in 1950. The Blaney-Criddle method was used by the FAO (Allen & Pruitt, 1986; Doorenbos et al., 1992) and is calculated with equation 27:

$$E_{BC} = c\left[p_y(0.46T_a + 8)\right] \tag{27}$$

, where c is a adjustment factor based on sunshine hours per day and minimum relative humidity, and p_y is the ratio of actual daytime hours per day and total annual daytime hours (Doorenbos et al., 1992; Guo et al., 2016). Another temperature-based evaporation model is the McGuiness-Bordne and makes use of extraterrestrial radiation calculated based on the sunrise equation, resulting in the following equation:

$$E_{MB} = \frac{1}{68\lambda}\left(R_e(T_a + 5)\right) \quad (28)$$

, where R_e is the extraterrestrial radiation (Oudin et al., 2005). Although this equation makes directly use of a radiation term, the Thornthwaite and Blaney-Criddle indirectly makes use of the sunrise equation.

3.2.2 Radiation-based

Radiation based temperature are not dependent on one factor like the temperature based methods, but also make use of net or solar radiation and other factors like relative humidity and latent heat of vaporization. The radiation-based methods are similar to the empirically temperature-based methods and/or derived from physical-based methods. Turc (1961) showed that evaporation rates can be estimated with a simple climatic formula and is expressed with the following equation:

$$E_{Turc} = 0.0133(23.88R_s + 50)\left(\frac{T_a}{T_a + 15}\right) + \left(1 + \frac{50 - RH}{70}\right) \quad (29)$$

The Turc equation estimates evaporation as a function of solar radiation, relative humidity and temperature (Guo et al., 2016). Another radiation-based evaporation method is the Hargreaves-Samani (Hargreaves & Samani, 1985) and is calculated as follows:

$$E_{HS} = 0.0135R_s(T_a + 17.8) \quad (30)$$

This method is a function of solar radiation, air temperature and latent heat of vaporization. Makkink (1957) simplified the Penman equation, described in the next section, and is expressed with equation 30:

$$E_{MK} = c_1\left(\frac{\Delta}{\Delta + \gamma}\frac{R_s}{\lambda}\right) - c_2 \quad (31)$$

, where c_1 and c_2 are constants of 0.61 (dimensionless) and 0.12 (mm day^{-1}) respectively. The Makkink equation is calibrated to cool climate conditions where the surface is covered with reference crop (De Bruin, 1981; Alexandris et al., 2008). A more physical radiation-based evaporation method is approached by Priestley & Taylor (1972) and is described with the following equation:

$$E_{PT} = \propto_{PT}\left(\frac{\Delta}{\Delta + \gamma}\frac{R_n - G}{\lambda}\right) \quad (32)$$

, where $\propto_{PT}$ is the Priestley-Taylor coefficient and equals 1.26 for advection-free saturated surfaces (Priestley & Taylor, 1972). According to Jensen et al. (1990), the Priestley-Taylor coefficient can be set between 1.70 and 1.75 for semi-arid regions. The Priesley-Taylor coefficient is optimized by filling in the EBC results as evaporation and averaging over the period. Inman-Bamber & McGlinchey (2003) did

a similar experiment in which they calculated evaporation rates based on the Bowen ratio energy balance (BREB) with a automatic weather station. Furthermore, Szilagyi & Jozsa (2007) and Rosenberry et al. (2007) show different ways to calculate BREB. In this research the BREB method uses, the temperature and actual vapor pressure gradient between the lower and upper arm as input, resulting in the following two equations:

$$\beta = \gamma \frac{T_{a,L} - T_{a,U}}{e_{a,L} - e_{a,U}} \tag{33}$$

$$E_{BREB} = \frac{R_n - G}{1 + \beta} \tag{34}$$

, where β is the Bowen ratio and the subscripts L and U are lower arm and upper arm respectively (Dexler et al., 2004; Nagler et al., 2005). Other ways to calculate the Bowen ratio are with saturation vapor pressure gradient, equilibrium temperature and vapor pressure gradient and vapor pressure deficit (Rosenberry et al., 2007; Inman-Bamber & McGlinchey, 2003; Szilagyi & Jozsa, 2007).

3.2.3 Wind and resistances-based

One of the landmarks towards a physical-based model of evaporation was the publication of Penman (1948), which was the first to combine an energy equation with an aerodynamic approach for estimating evaporation. The use of net radiation as energy in the equation eliminates the use of temperature, which results in the following equation, known as the Penman or Penman combination equation (McMahoon et al., 2013):

$$E_{Pen} = \frac{\Delta}{\Delta + \gamma} \frac{R_n}{\lambda} + \frac{\gamma}{\Delta + \gamma} E_a \tag{35}$$

, where E_a is the aerodynamic component based on a wind speed function and the vapor pressure deficit. Another landmark was reached when Monteith et al. (1965) combined the Penman equation by linking the equation's aerodynamic component of saturated surfaces through turbulent transport with the constraint of the energy balance (Dolman et al., 2014). The Penman-Monteith equation makes use of aerodynamic and surface resistances and is expressed as followed (Allen et al., 1998):

$$ET_{PM} = \frac{1}{\lambda} \frac{\Delta(R_n - G) + p_a c_p \dfrac{(e_s - e_a)}{r_a}}{\Delta + \gamma \left(1 + \dfrac{r_s}{r_a}\right)} \tag{36}$$

, where p_a is the mean air density at constant pressure. The Penman-Monteith equation is a physical-based equation, which incorporates all components of the energy balance. Because of the many factors involved and calibration of the resistances terms the Penman-Monteith equation has become complex. The FAO decided to make guidelines for reference crop ET and has adopted a simplified

version of the Penman-Monteith equation by filling in reference crop estimates and eliminating the resistances terms, resulting in the following equation (Allen et al., 1998):

$$ET_{FAO} = \frac{0.408\Delta(R_n - G) + \gamma \frac{900}{T + 273} U_2(e_s - e_a)}{\Delta + \gamma(1 + 0.34U_2)} \quad (37)$$

, where ET_{FAO} is the reference crop, with a height of 0.12 meter, evapotranspiration for a vegetated surface of grasses. Allen et al. (1998) describes the derivation of the FAO Penman-Monteith equation for the grass reference crop. Another adoption of the Penman-Monteith equation is the Matt-Shuttleworth equation by Shuttleworth (2006). This equation involves the general applications of Penman-Monteith equation, but now for all well watered crops instead of a reference crop. The Matt-Shuttleworth equation is calibrated to well-watered evaporative surfaces in windy semi-arid region and described with the following equation (Shuttleworth & Wallace, 2009):

$$ET_{MS} = \frac{\Delta(R_n - G) + \left(\frac{VPD_{50}}{VPD_2}\right) \frac{p_a c_p U_2 VPD_2}{R_c^{50}}}{\Delta + \gamma\left(1 + \frac{r_s U_2}{R_c^{50}}\right)} \quad (38)$$

, where VPD_{50} and VPD_2 are the vapor pressure deficit at 50 and 2 meters respectively, and R_c^{50} is an aerodynamic coefficient. The height of 50 meters is chosen arbitrarily and the calculations of the terms at 50 meters height and derivation of the equation are provided by Shuttleworth (2006).

3.2.4 Actual evaporation

Bouchet (1963) set up a method in which to approach actual evaporation and hypothesized that actual and potential evaporation depends on each other in a complementary way. This complementary relationship exists via feedbacks between the land and the atmosphere (McMahoon et al., 2013). Bouchet (1963) proposed this complementary relationship with the following equation:

$$E_{Act} = 2E_{Wet} - E_{Pot} \quad (39)$$

, where actual evaporation equals two wet environment evaporation minus potential evaporation. According to Huntington et al. (2011), potential evaporation will increase and actual evaporation will decrease when moisture availability decreases. Actual evaporation will be zero when no moisture is available. However, when the landscape becomes fully saturated the actual, wet and potential evaporation will all be equal to each other (McMahoon et al., 2013). This complementary relationship is important when measuring evaporation in semi-arid areas, and thus included in the evaporation model. A conceptual representation of the complimentary relationship is depicted in Figure 12.

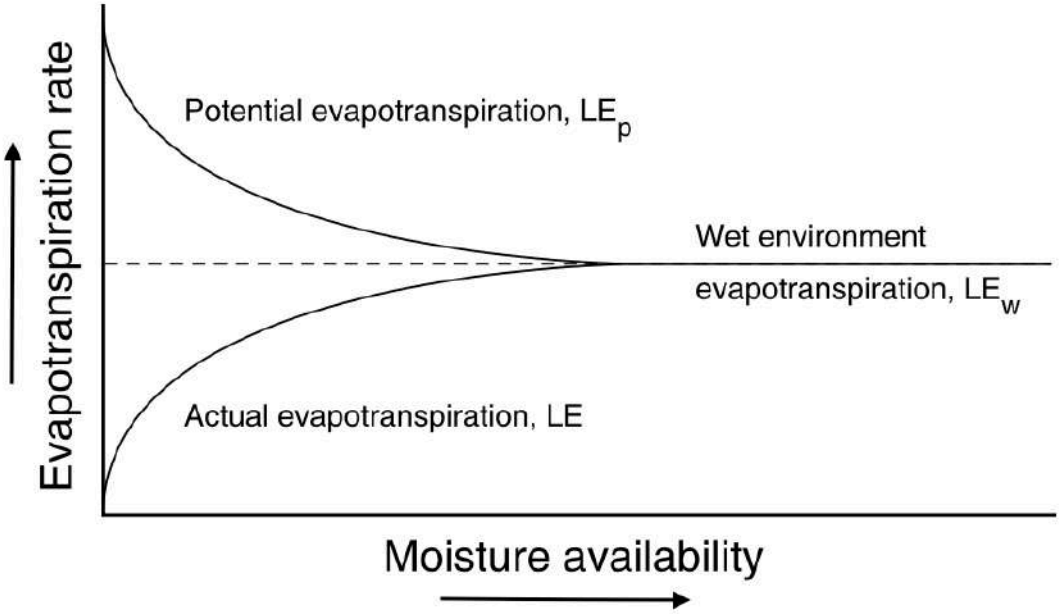

Figure 12: Conceptual representation of the complementary relationship in terms of latent heat flux (Adopted from: Huntington et al., 2011).

Brutsaert & Strickler (1979) proposed the advection-aridity model based on the symmetric complementary relationship approach. Their model results in the following formula:

$$E_{BS} = (2 \propto_{PT} - 1)\frac{\Delta}{\Delta + \gamma}\frac{R_n}{\lambda} - \frac{\gamma}{\Delta + \gamma}E_a \tag{40}$$

Similar to the Brutsaert-Strickler formula, Szilagyi & Jozsa (2008) proposed a new modified advection-aridity model, resulting in the following equation:

$$E_{SJ} = 2E_{PT}(T_e) - E_{Pen} \tag{41}$$

, where the term $E_{PT}(T_e)$ is the Priestley-Taylor equation calculated with the equilibrium temperature. Next to the complementary relationship, Granger & Gray (1989) proposed another method to estimate actual evaporation. The Granger-Gray method is based on the Penman equation and establishes a new dimensionless parameter called the relative drying power (Granger & Gray, 1989). This parameter accounts for the departure of saturated conditions and thus approach actual evaporation, expressed as followed:

$$E_{GG} = \frac{\Delta G_g}{\Delta G_g + \gamma}\frac{R_n - G}{\lambda} + \frac{\gamma G_g}{\Delta G_g + \gamma}E_a \tag{42}$$

, where the term G_g is the ratio actual to potential evaporation and a function of the relative drying power derived and discussed by Granger & Gray (1989). Another approach of estimating actual evaporation is the model of Morton (1983). According to McMahoon et al. (2013), Morton was at the forefront of evaporation analysis since

1965 and culminates in the mid-80s with three models for actual evaporation for land, shallow and deep lakes. Morton (1983) proposed his CRAE model for actual evaporation on land and is based on the complementary relationship described earlier. Furthermore, Nash (1989) discussed Morton's CRAE model to be a valuable extension to the Penman equation in that it allows evaporation estimations under limited water supply. The model is tested for 143 basins and is expressed with the following equation (Morton, 1983):

$$ET_{CRAE} = \frac{1}{\lambda}\left(R_n - [\gamma P f_v + 4\epsilon_s \sigma (T_e + 273)^3](T_e - T_a)\right) \quad (43)$$

, where λ is in W day^{-1}, f_v is the vapor transfer coefficient, ϵ_s is the surface emissivity, and σ is the Stafan-Boltzmann coefficient. The Morton CRAE model is the most complex of evaporation methods used in this research.

3.2.5 Statistical analysis

In this research sixteen evaporation methods are used. To compare these different evaporation methods statistical analysis is applied. All the methods have been correlated with one another and the correlation coefficients have been produced in a single heatmap. The Pearson's product correlation coefficient is used and is calculated as followed:

$$p_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \quad (44)$$

, where $cov(X,Y)$ is the covariance between variable X and Y, and σ the standard deviation (Benesty et al., 2009). P-values of the correlation are calculated with a t-distribution. r^2 values between each method have been computed and are also produced in a single heatmap. The r^2 values are calculated with taking the square of the correlation coefficient (Benesty et al., 2009) and have been used by Szilagyi & Jozsa (2007) among others to compare different evaporation methods. Individual regression plots between specific methods are produced with trend line and deviation from the reference EBC method. Significance of each evaporation method against the reference EBC method is calculated with p-values. The EBC method and the different evaporation models were analyzed using the following linear regression equation:

$$Y = mX + b \quad (45)$$

, where Y is the EBC method, m and b constants, and X the evaporation model. Similar regression plots have been produced as shown in the paper by Tanny et al., (2007). Furthermore, for all evaporation methods with the EBC method the index of agreement (IA) has been calculated. The index of agreement is a relative difference measure and is expressed as (Willmott, 1982):

$$IA = 1 - \left[\frac{\sum_{i=1}^{n}(X_i - Y_i)^2}{\sum_{1=1}^{n}(|X_i - \bar{Y}| + |Y_i - \bar{Y}|)^2}\right] \quad (46)$$

, where $\bar{Y}$ is the mean of variable Y and perfect agreement between X and Y is reached when IA equals 1 (Todd et al., 2000). The root mean square error (RMSE) is calculated with the following equation (Willmott, 1982):

$$RMSE = \sqrt{\left[n^{-1}\sum_{i=1}^{n}(X_i - Y_i)^2\right]} \quad (47)$$

According to Willmott (1982), the RMSE is one of the best overall measures of model performance as they summarize the difference between the two variables and has been used by McMahoon et al. (2013) when comparing different evaporation methods. Furthermore, the Nash and Sutcliffe (1970) efficiency coefficient is calculated with the following formula:

$$NSE = 1 - \left[\frac{\sum_{i=1}^{n}(X_i - Y_i)^2}{\sum_{1=1}^{n}(X_i - \bar{X})^2}\right] \quad (48)$$

, where NSE is the Nash-Sutcliffe efficiency coefficient ranging from -∞ to 1, with 1 meaning a perfect match. According to Schaefli & Gupta (2007), the Nash and Sutcliffe efficiency coefficient is a powerful tool in hydrological modeling and therefore included in the evaporation model. Performances of a comparison analysis of evaporation methods, including the IA, RMSE, NSE has, among others, been used by Legates & McCabe (1999), Todd et al. (2000) and Tanny et al. (2007). Furthermore, a sensitivity analysis is done for the different methods of calculating the Bowen ratio and optimizing the Priesley-Taylor coefficient for the Priesley-Taylor, Szilagyi-Jozsa and Brutsaert-Strickler methods.

3.3 Water storage model

To test the impact of choice of evaporation method on available water in a WHS, a simple water storage model is used (Tiggeloven, 2015). Six evaporation methods have been selected to calculate evaporation with this model, which cover the categories of methods, and performance of the statistical analysis has been produced to compare the sensitivity of the WHS to the selected evaporation methods. To measure the effectiveness of different evaporation time series as input a couple water harvesting systems are used in the water storage model. The model runs on daily time steps for the period 7th of October until 21st of December 2016s in which, the water storage increases with rainfall and declines with evaporation and daily usage of inhabitants of the region and community. Rainfall is measured at the AWS and because of poor results for the second rainy season; the WHS are set to full capacity at the beginning of the model run. After rainfall, runoff takes place after a threshold of 10 mm day^{-1} and has been integrated in the model with catchment area and runoff coefficient of 0.58. The threshold value is based on Li et al. (2004), which

measures a threshold between 7-9 mm day^{-1} for dry soils and Cammeraat (2004). The ROC value is based on two bare fallow plots (van de Giesen et al., 2000) To compare the performance of the WHS and the different evaporation methods as input, the results of the total evaporative fracture has been divided by the capacity of the WHS. The RMSE compares the deviations of the water storage during the period of modeling. The usage of water has been set to 0.012 m^3 per day per inhabitant (Lasage et al., 2015). Figure 13 depicts a schematic overview of the evaporation method and the flow on input, parameters and calculation. First global parameters and specific WHS parameters are assigned, then the storage calculations are done. Output of the model contains of the calculated evaporative fracture, storage level and total water harvested. The dimensions and usage of the modeled sand dam and open pond as WHS have been specified in the next two sections.

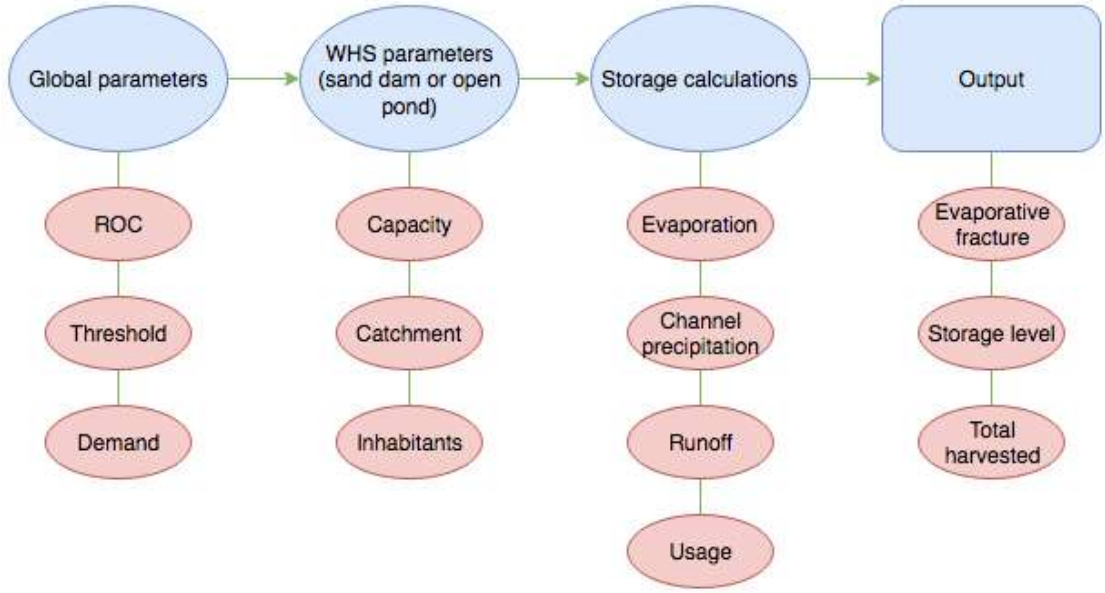

Figure 13: Overview of the evaporation model.

3.3.1 Sand dam

The modeled sand dam has estimated dimensions of the channel width, length and average depth of the sand dam located near Nyumbani Village, Kitui. The dimensions have been set to 40 m length, 9.25 m width and an average depth of 3 m. Subsequently, the capacity of the sand dam reaches 1098 m^3 and the catchment has been set to 5.2 km^2 and meets the specification of the ephemeral river at Nyumbani Village. For evaporation in a sand dam only the top 90 cm of the sediment water storage is affected (Love et al., 2011). Furthermore, assumptions has been made that a runoff coefficient determines the percentage of water that reaches the dam and all the water can potentially be stored and infiltrated directly in the sediment of the sand dam. The community at Nyumbani Village exists of 1100 inhabitants.

3.3.2 Open pond

For the modeled open pond, smaller dimensions have been selected than the sand dam, as open pond systems supply water to a couple of households (Lasage & Verburg, 2015). The dimensions have been set to 5 m by 5 m with a depth of 2 m. The catchment has been set to 300 m^2, making the open pond ideal for

approximately ten households. Average household size is assumed to consist of 5.8 persons (Lasage et al., 2015).

4 Results

4.1 Diurnal forcings and rainfall

During the second rainy season of 2016, 175 mm of precipitation has been measured at the AWS in Kitui. This amount is a poor result for the region and is much below the average of 517 mm for the second rainy season according to the historical data with a standardized anomaly of -1.6. Only three years in the historical data have a lower anomaly than the one measured and where 1917, 1970 and 1983. The average measured temperature is 22 ºC with small deviations of 2 ºC. An overview of the temperature and rainfall during the period of measurements is given in Figure 14. After a dry October, precipitation was starting to fall in November with the peak between 14th and 21st of November.

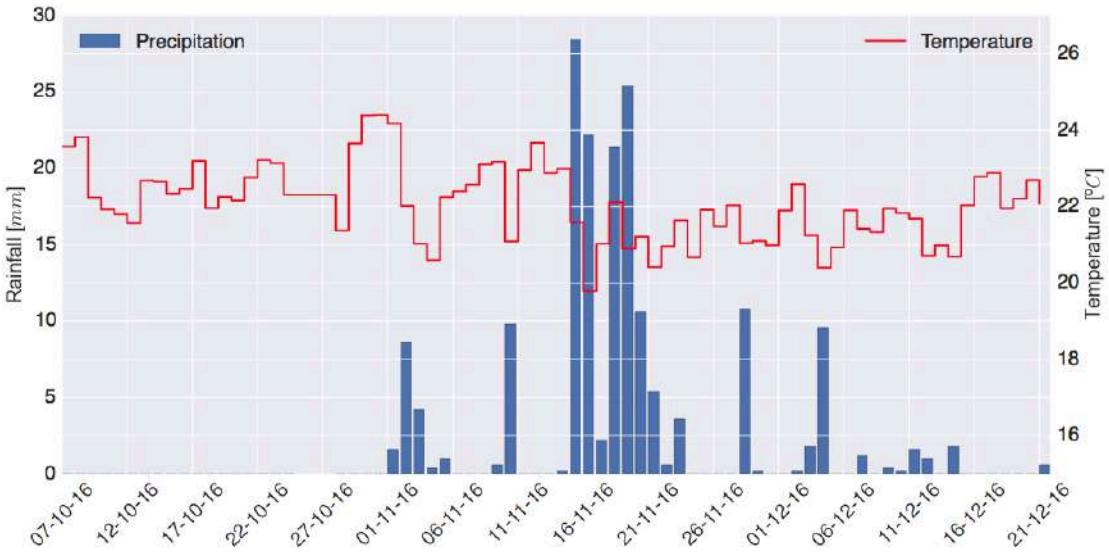

Figure 14: Rainfall and temperature measurements during the period 7th of October till 21st of December in Kitui, Kenya. The red line is temperature and bars represents rainfall.

The primary forcings of the energy balance are depicted in a diurnal plot of 16th and 17th of October in Figure 15. The sensible heat flux is relatively stable during the day and night compared to the net radiation and soil heat flux. The net radiation reaches its peak during the middle of the day, while the soil heat flux reaches its peaker at the end of the afternoon. The latent heat flux has been derived by the Penman-Monteith equation, described in Equation 36. Cloud interactions causes the net radiation flux to have spikes during the day.

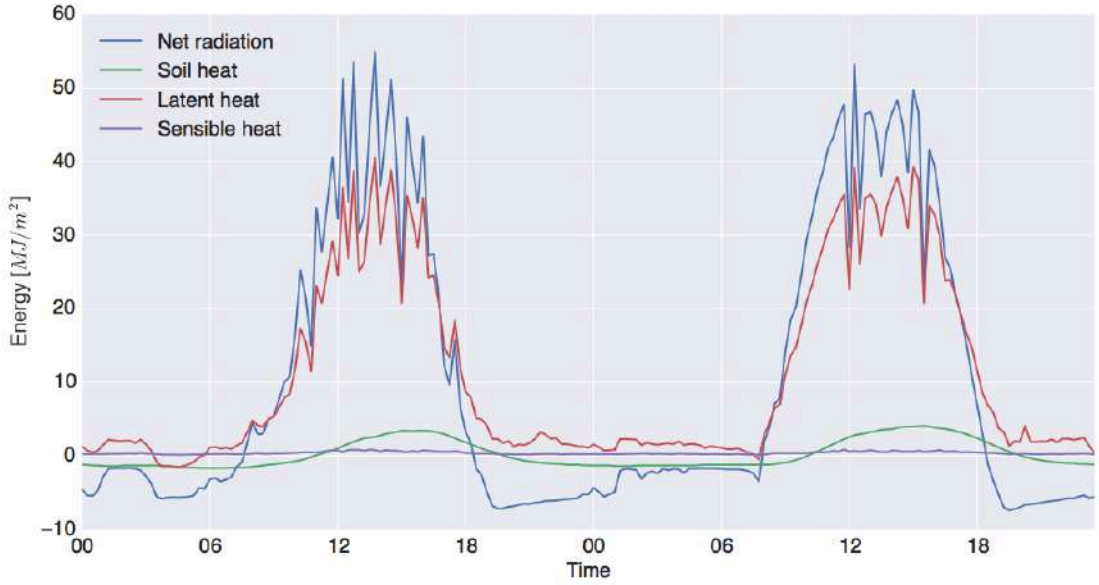

Figure 15: Diurnal energy fluxes during 16th and 17th of October 2016.

The equilibrium temperature has been calculated on a daily time step with the assumption that it should be lower than the air temperature. The results of the calculation are depicted in Figure 16, in which the equilibrium temperature is compared with the air temperature. The r^2 value shows little agreement between changes in air temperature to changes in equilibrium temperature. Next to the regression plot of the equilibrium temperature, the measured versus calculated net radiation plot is shown. The calculation of net radiation overestimates at low radiation values and underestimates at higher radiation values. An r^2 value of 0.7 is calculated. Both plots have a p-value lower than 0.01 computed with the Pearson correlation.

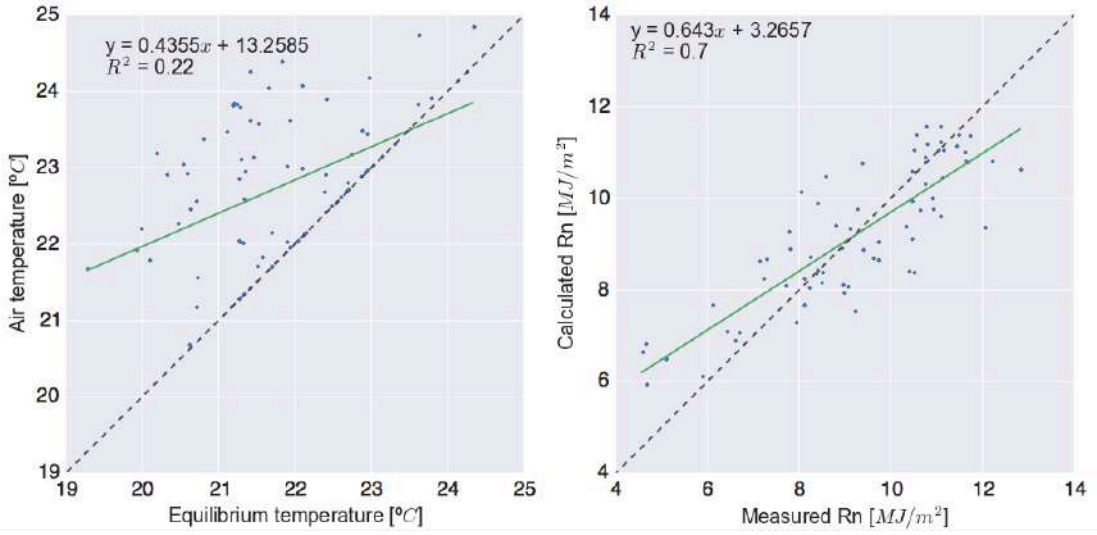

Figure 16: Regression plot of the equilibrium temperature versus the air temperature (left) and measured net radiation versus calculated radiation (right). The green line represents the regression line and the dashed the 1:1 ratio.

An overview of the average and standard deviation for all measured and calculated meteorological factors is given in Table 2. The soil heat flux is on average close to zero.

Table 2: Summary statistics of the measured and calculated meteorological factors.

	T1	T2	RH1	RH2	Rs	Rn	WS	G	Re	Te
Mean	*22.79*	*22.13*	*68.46*	*70.26*	*68.46*	*9.21*	*2.43*	*-0.06*	*36.60*	*21.88*
Std dev	*0.95*	*3.63*	*7.80*	*10.27*	*7.80*	*1.84*	*0.62*	*0.48*	*0.70*	*1.01*
	Td	**lv**	**psy**	**es**	**ea**	**VPD**	**slope**	**ra**	**Ea**	**H**
Mean	*16.49*	*2.45*	*0.07*	*2.91*	*1.88*	*1.02*	*0.17*	*178,96*	*4.92*	*0.22*
Std dev	*1.45*	*<0.01*	*<0.01*	*0.19*	*0.17*	*0.29*	*<0.01*	*52.03*	*2.02*	*0.09*

*T1: Temperature 1.25m [ºC]; T2: Temperature 1.75m [ºC]; RH1: Relative humidity 1.25m [%]; RH2: Relative humidity 1.75m [%]; Rs: Solar radiation [MJ day^{-1}]; Rn: Net radiation [MJ day^{-1}]; WS: Wind speed [m s^{-1}]; G: Soil heat flux [MJ day^{-1}]; Re: Extraterrestrial radiation [MJ day^{-1}]; Te: Equilibrium temperature [ºC]; Td: dew point temperature [ºC]; lv: Latent heat of vaporization [MJ day^{-1}]; psy: Psychrometric constant [kPaºC^{-1}]; es: saturation vapor pressure [kPa]; ea: actual vapor pressure [kPa]; VPD: Vapor pressure deficit [kPa]; slope: Slope of saturation vapor pressure curve [kPaºC^{-1}]; ra: aerodynamic resistance [day m^{-1}]; Ea: Wind-function [-]; H: sensible heat flux [MJ day^{-1}].

4.2 Evaporation model results

Sixteen methods of evaporation have been calculated with the processed data. An overview of the daily evaporation per category for the period of measuring has been provided in Figure 17 in which the EBC method is depicted in every subplot. The first subplot groups the temperature-based methods. As they are only based on temperature and the temperature fluctuations are within 2 ºC per day, the temperature-based methods evaporation fluctuations are also expected to be similar. The Thornthwaite method is the most sensitive to temperature fluctuations and has the highest standard deviation of the three temperature-based methods as shown in Table 3. The Blaney-Criddle method and McGuinnes-Bordne have the highest evaporation results on average.

The second subplot in Figure 17 depicts the radiation-based methods. These methods have higher fluctuations and are sensitive to radiation input. The Turc, Haergraves-Samani and Makkink methods share similar behavior, however the Makkink method is continuously lower than the former two methods. The Priesley-Taylor method has generally the same behavior as the other radiation-based methods, and closely matches the EBC method. The BREB method clearly shows different behavior and is on average lower than the EBC method, in contrast to the other radiation-based methods, which have on general higher values than the EBC method.

The wind and resistances-based evaporation method are depicted in the third subplot of Figure 17, and generally agree more with the EBC method than the radiation- and temperature-based methods. The results of the Penman and FAO PM methods as well as Penman-Monteith and Matt-Shuttleworth show similar behavior and results. As all the methods in this category are derived from the Penman method, the outcome and behavior are similar and close to the EBC method with the exception of the first third of the measuring period.

The last subplot contains the actual evaporation and complementary relationship methods. Large deviations between the methods are found in which the Brutsaert-

Strickler and Szilagyi-Jozsa deviate below the EBC method in the first half of the measuring period and the Morton CRAE method above. Overall, the Granger-Grey and Morton CRAE show much agreement with the EBC method.

Table 3: Summary statistics for the sixteen assessed evaporation methods.

	TW	BC	MB	Turc	HS	MK	PT	BREB	
Mean	*2.82*	*4.99*	*6.12*	*5.16*	*4.69*	*3.96*	*3.41*	*2.79*	
Std dev	*0.34*	*0.12*	*0.28*	*1.01*	*1.09*	*0.90*	*0.60*	*0.54*	
	Pen	**PM**	**FAO**	**MS**	**BS**	**SJ**	**GG**	**Mort**	**EBC**
Mean	*4.09*	*3.44*	*3.97*	*3.53*	*2.69*	*2.63*	*3.25*	*3.99*	*3.70*
Std dev	*0.93*	*0.66*	*0.86*	*0.79*	*0.79*	*0.85*	*0.56*	*0.79*	*0.65*

* TW: Thornthwaite (1948); BC: Blaney-Criddle; MB: McGuinnes-Bordne; HS: Hargreaves-Samani; MK: Makkink; PT: Priesley-Taylor; BREB: Bowen Ratio Energy Balance; Pen: Penman (1948); PM: Penman-Monteith; FAO: FAO Penman-Monteith derivation; MS: Matt-Shuttleworth; BS: Brutsaert-Strickler; SJ: Szilagyi-Jozsa; GG: Granger-Grey; Mort: Morton CRAE; EBC: Energy Balance Closure.

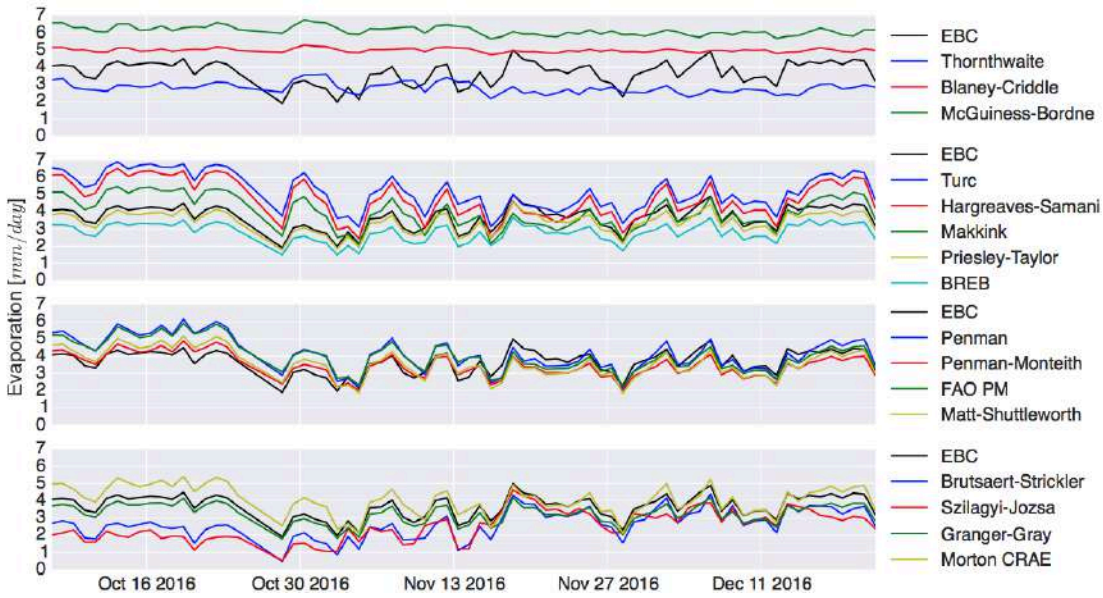

Figure 17: Overview of the daily measured evaporation of the 16 assessed methods grouped per category in every subplot.

To compare the assessed evaporation methods a correlation and r^2 heatmap has been produced and is depicted in Figure 18. The evaporation methods are showing agreeable results for the correlation and r^2 when comparing within their category. The temperature-based methods have a correlation value above 0.8 and r^2 value above 0.6 with one another. Furthermore, the Makkink, Turc and Hargreaves-Samani show high values of the correlation and r^2 results and the above discussed similarities between the wind and resistances-based method also show high values for correlation and r^2. Despite the poor results of the Brutsaert-Strickler and Szilagyi-Jozsa and other methods, agreeable results are found when compared with Priesley-Taylor, BREB and one another. In comparison with the EBC method a couple of evaporation methods have agreeable results, namely the Priesley-Taylor, BREB, Brutsaert-Strickler, Granger-Grey and in lesser extent Morton CRAE and Szilagyi-Jozsa.

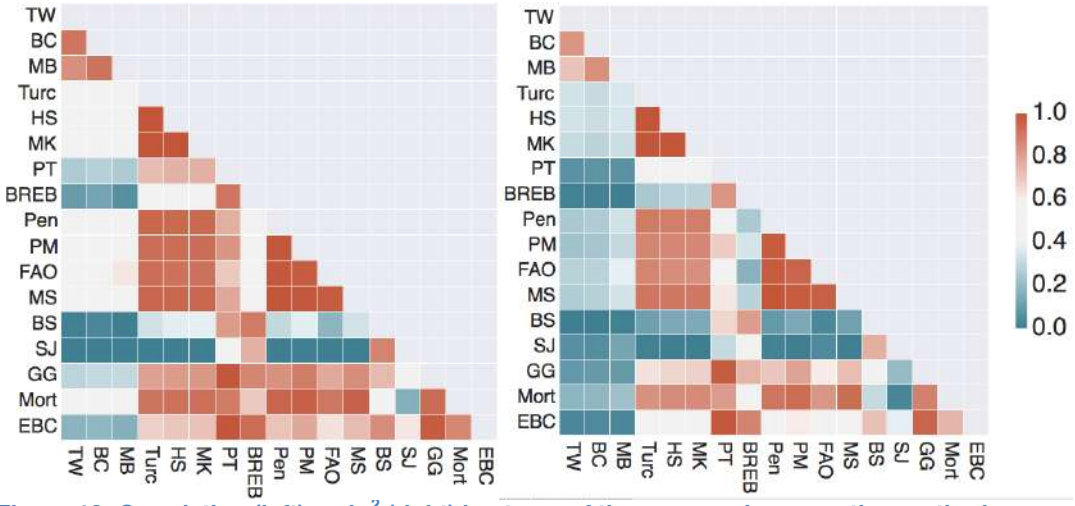

Figure 18: Correlation (left) and r^2 (right) heatmap of the assessed evaporation methods.

Regression plots of the energy balance closure with assessed evaporation methods are produced in order to analyze the evaporation methods with the incoming and outgoing fluxes. As the temperature-based evaporation methods only fluctuate sparsely, they are depicted in Figure 19 with a straight line and do only slightly respond to different input of available energy. The r^2 values of the temperature-based models are 0.04 or below. The radiation-based methods show better results with high r^2 values of 0.99 for Priesley-Taylor and 0.88 for the Bowen Ratio Energy Balance method. Notably, the mostly recommended Penman-Monteith results in an r^2 value of 0.62. Furthermore, the Brutsaert-Strickler, Granger-Grey and Morton CRAE produce agreeable r^2 results of about 0.7 and higher.

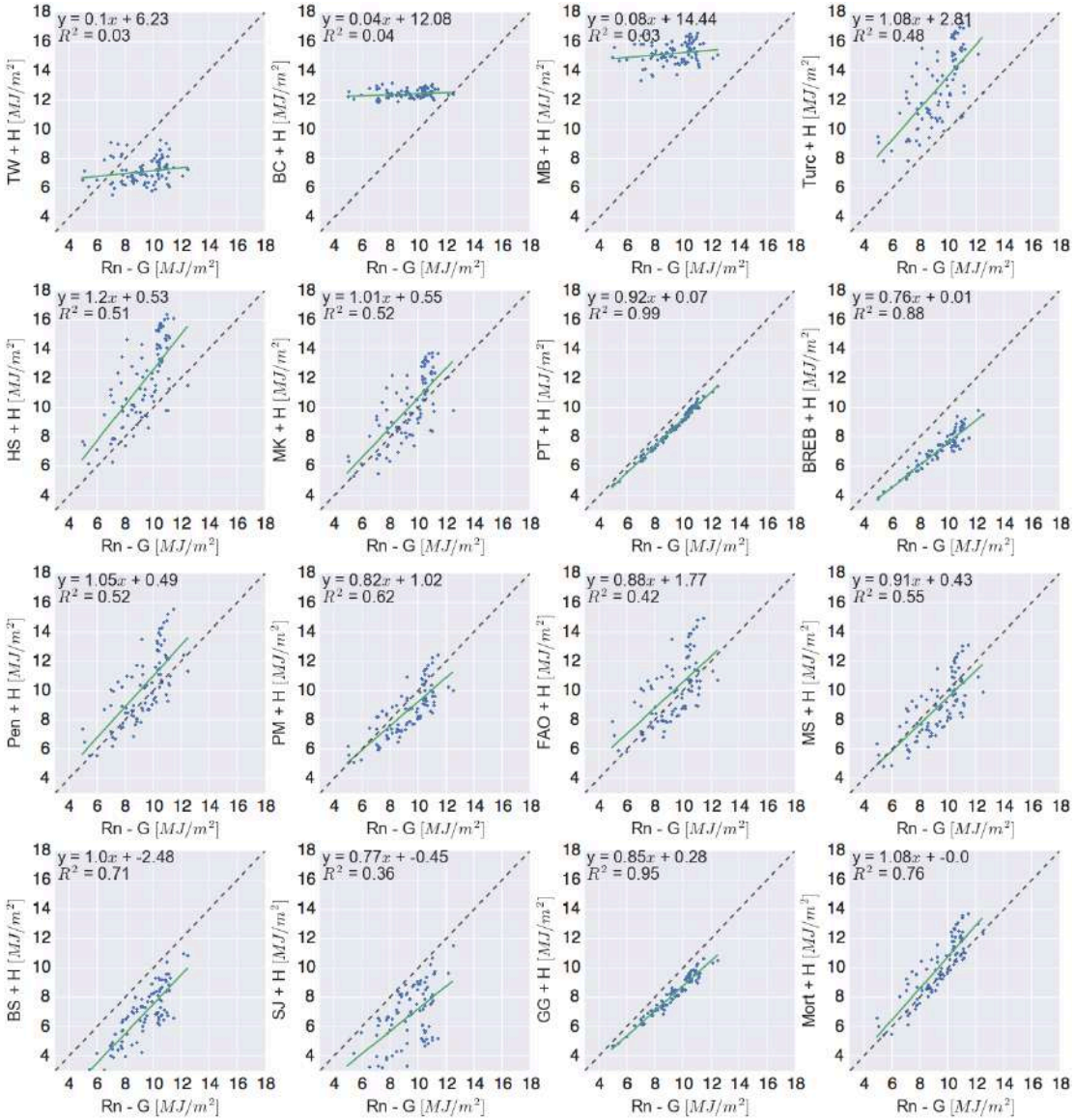

Figure 19: Regression plots of the energy balance and assessed evaporation methods in which Rn-G (available energy) is depicted versus the latent plus sensible heat fluxes. The dashed line shows perfect agreement between available energy and the heat fluxes.

Further statistical analysis is shown in Table 4 where the sixteen methods of evaporation are compared with the EBC method with Root Mean Square Error, Index of Agreement, Nash-Sutcliffe efficiency and corresponding p-values with Pearson correlation analysis. Only the temperature-based evaporation methods are significantly different from the EBC method and five methods have a Nash-Sutcliffe efficiency coefficient higher than zero, implicating agreement between the measurements. The largest deviations of all four statistical models from the EBC method are found with the McGuiness-Bordne and the best results of all four models are computed with the Priesley-Taylor and Granger-Gray method. Other evaporation methods with agreeable results are the Penman-Monteith, Morton CRAE and Matt-Shuttleworth with positive values for the Nash-Sutcliffe efficiency coefficient, Root Mean Square Error value of below 3 and Index of agreement values higher than 0.8.

Table 4: Overview of the results of the statistical analyses of the 16 evaporation methods versus the EBC method.

	TW	BC	MB	Turc	HS	MK	PT	BREB
RMSE	6.05	7.89	13.75	8.95	6.89	3.74	1.66	4.98
IA	0.32	0.12	0.16	0.57	0.65	0.79	0.94	0.64
NSE	-1.90	-3.92	-13.93	-5.33	-2.75	-0.11	0.78	-0.96
P-value	0.10	0.07	0.12	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
	Pen	**PM**	**FAO**	**MS**	**BS**	**SJ**	**GG**	**Mort**
RMSE	4.16	2.77	3.96	3.09	5.99	6.92	2.62	2.68
IA	0.76	0.85	0.75	0.83	0.64	0.59	0.86	0.88
NSE	-0.37	0.39	-0.24	0.24	-1.83	-2.79	0.46	0.43
P-value	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01

* RMSE: Root Mean Square Error; IA: Index of Agreement; NSE: Nash-Sutcliffe Efficiency; P-value: Calculated Probability.

As stated earlier different values of the Priesley-Taylor coefficient can be selected based on the location. Optimization of the coefficient results in a value of 1.36. In Table 5 new statistical analysis is shown for the evaporation methods, which incorporate the Priesley-Taylor coefficient. The Priesley-Taylor, Brutsaert-Strickler and Szilagyi-Jozsa all are showing better result for the statistical analysis, wherein the former results in near perfect score for the Index of Agreement and Nash-Sutcliffe efficiency. Further statistical analysis has been implemented to compute the difference between multiple approaches of calculating the Bowen ratio. As explained in the methods section, the Bowen ratio has been calculated in scientific research with at least four different approaches. Table 5 shows the statistical output of these approaches. The results for the Bowen ratio calculated with the saturation vapor pressure shows similar results with calculation of the actual vapor pressure. Both the Bowen ratio calculated with equilibrium temperature and vapor pressure deficit show more agreement with the EBC method, making the later the method with the best results after the Priesley-Taylor.

Table 5: Results of the statistical analysis for the evaporation methods with optimized Priesley-Taylor coefficient and different approaches of calculating the Bowen ratio.

	PT new	BS new	SJ new	BREB es	BREB Te	BREB VPD
RMSE	0.37	3.33	4.54	5.12	2.06	1.04
IA	1.00	0.85	0.75	0.63	0.93	0.98
NSE	0.99	0.12	-0.63	-1.07	0.67	0.91
P-value	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01

4.3 Water storage model results

Water storage has been calculated for the period 7th of October until 21st of December with evaporation of the EBC method as input, which will serve as the baseline to which the other evaporation methods are compared. Figure 18 shows the modeled water storage for the open pond and the sand dam. Usage for the open pond WHS is calculated at 0.71 m^3 day^{-1} and for the sand dam at 13.49 m^3 day^{-1} and average evaporation of 0.09 and 1.14 m^3 day^{-1} respectively. The evaporative fracture

is 0.13 and 0.08 per m^3 capacity for the open pond and sand dam respectively for the period of research.

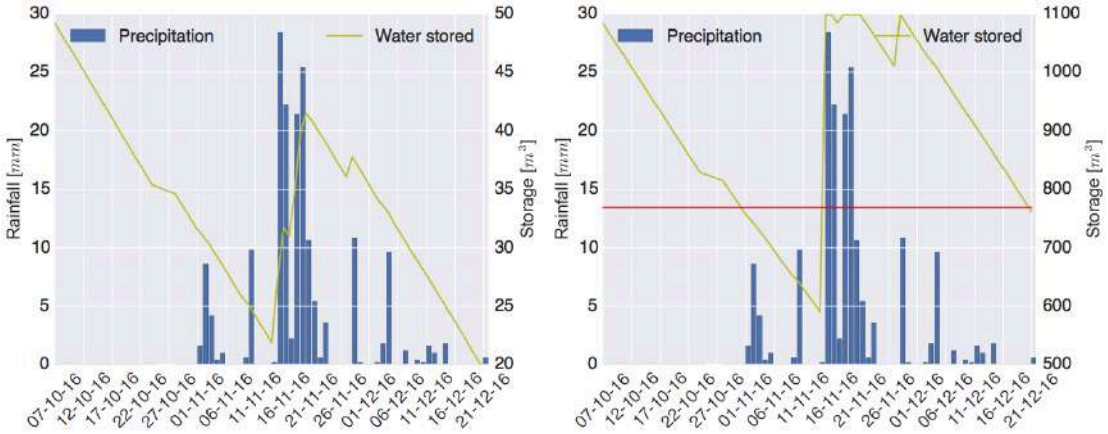

Figure 20: Modeled water storage and rainfall plots of an open pond (left) and sand dam (right). The red line displays the boundary capacity after which evaporation would not occur in the sand dam, yellow line is storage and bars represents rainfall.

Six evaporation methods have been selected on the basis of the statistical analysis and categories, and contains one temperature-based, two radiation-based, one wind and resistances-based and two actual evaporation methods. Table 6 shows the deviation of evaporative fracture in percentages and Root Mean Square Error of the selected evaporation methods to the EBC method. A range of -24 to almost 10 % has been found for the open pond WHS for the period of 73 days and a range of -27 to almost 8 % for the sand dam. Therefore, on average per day the selection of methods deviates up to 0.03 and 0.02 m^3 of evaporation per cubic meter of WHS for open pond and sand dam respectively. Subsequently, a deviation of usable water of equivalent for the period of research of up to two and twenty-five persons for open pond and sand dam respectively, are shown when selecting an evaporation method. That is within 3.5 and 2.5 % of total harvesting fluxes. Highest Root Mean Square Error has been calculated for the Thornthwaite method and lowest for the Penman-Monteith and Priesley-Taylor method.

Table 6: Evaporative fracture of capacity of WHS and deviations for the six selected evaporation methods compared to the EBC method, where OP denotes open pond, SA sand dam, EF evaporative fracture and Dev the deviation from the EBC method. All values are in percentages of capacity of the WHS.

	TW	MK	PT	PM	GG	Mort
Dev EF OP	-23.95	8.74	-7.79	-5.72	-11.90	9.87
RMSE OP	0.84	0.52	0.28	0.17	0.42	0.50
Dev EF SA	-27.28	6.82	-7.98	-6.82	-12.31	7.92
RMSE SA	5.38	4.25	1.50	2.16	2.19	3.24

* All evaporation methods in table have P-value < 0.01 in comparison with the EBC method.

5 Discussion

5.1 Evaporation methods implications

Sixteen evaporation methods are categorized into temperature-based, radiation-based, wind and resistances-based, and actual evaporation. Evaporation of the modeled water storage with these methods have been calculated for the period of 7th of October until 21st of December 2016 and compared with one another with statistical analysis. Due to measurement failure, daily climate data of an incomplete sequence of thirteen days is retrieved for the first rainy season of 2017. Rainfall results for the second rainy season was below average with standardized anomaly of -1.6. The region of Kitui differs in elevation and rainfall in different measuring areas can show different results. Temperature was relatively stable for the period and the average measured temperature agrees with the average of historical data for the measured months. Diurnal fluxes show similar results compared to Tanny et al. (2007), where the sensible heat flux fluctuates just above 0 MJ day^{-1} and, if smoothed, agreeable radiation curve. However, the latent heat flux is not measured with an Eddy Covariance device and instead calculated with the energy-based Penman-Monteith method. Subsequently, the results of the latent heat flux strongly follow the input of net radiation and declines in the early evening, whereas Tanny et al. (2007) show a higher latent heat flux in the evening. The equilibrium temperature is calculated on the assumption that it is lower than the air temperature. Huntington et al. (2011), show similar results in which equilibrium temperature deviations become larger when temperatures are between 15 to 25 ºC.

In general, the evaporation methods within their category are behaviorally similar. Temperature-based methods fluctuate slightly as their only input, temperature, is relatively stable in the period of measurement. The wind and resistances methods are very similar as they are derived from the same equation. The Priesley-Taylor shows the most agreement with the EBC method. While Brutsaert-Strickler and Szilagyi-Jozsa are derived from the Priesley-Taylor method, they show much less agreement. Due to the similar behavior, evaporation methods show high correlation and r^2 values when comparing with other evaporation methods of the same category. Summarizing results of numerous studies for calculating evaporation, McMahoon et al. (2013) shows that the ratio of mean of Priesley-Taylor to mean of Penman-Monteith has similar results in specific cases to this study. The ratio is for a semi-arid region in India 1.09 and 1.02 for an arid region in India, while this study shows a ratio of 0.99 and 1.07 for adjusted Priesley-Taylor alpha constant. Moreover, the Blaney-Criddle high ratio to Penman-Monteith of 1.44 in semi-arid India agrees with the high value of 1.35 found in this research.

When comparing the various evaporation methods with the EBC method, it has both an advantage and a disadvantage. Energy fluxes can easier be compared, however calculation of the EBC method can lead to biases, as it is not calculated by input of a single instrument, like an eddy covariance instrument. Every sensor, like temperature or wind speed measurements, has their own accuracy and resolution

summarized in Table 1. Additionally, calculation of the temperature gradient for the sensible heat flux and Bowen ratio can be computed by selecting different measurement height, potentially leading to different outcomes. Drexler et al. (2004) and Nie et al. (1992), argue that methods using sensible heat or BREB measurements reduce accuracy when the gradient between sensors approaches zero. When the gradient of the Bowen ratio is lower than 0.4, Nie et al. (1992) argue that small differences in the gradient can cause large percentage differences in latent heat flux and when Bowen ratio reaches -1, unrealistic values can be found. However, in this research no unrealistic values were computed when the Bowen ratio reaches a value lower than 0.4. Furthermore, DeMeo et al. (2003) show agreeable results when comparing energy balance methods with eddy covariance and calculated an r^2 value of 0.99. Nagler et al. (2005), argue that unlike the eddy covariance method, the validity of the latent and sensible heat calculations of the energy balance methods cannot be checked for closure. Further research is needed to validate the EBC method.

In comparison with the EBC method a couple of evaporation methods have agreeable results of correlation and r^2 values higher than 0.7 and 0.6. These methods are the Priesley-Taylor, BREB, Brutsaert-Strickler, Granger-Grey and in lesser extent Morton CRAE and Szilagyi-Jozsa. The Priesley-Taylor, Granger-Gray and the Bowen Ratio Energy Balance show regression fit with all observations close to the regression line. The temperature-based methods are performing poorly with the regression plot and observations point to no feedback when available energy changes. Consequently, the regression line is flat. Similar results for the Priesley-Taylor method where shown by the research of Rosenberry et al. (2007) in which evaporation methods are compared with the BREB vapor pressure deficit variant. Rosenberry et al. (2007), show an r^2 value of 0.97 and is close to the value of 0.99 for the r^2 calculated in this research. The temperature-based methods Thornthwaite and Blaney-Criddle show notably more agreeable results with r^2 of 0.73 in contrast to 0.03 and 0.04 retrieved in this study for Thornthwaite and Blaney-Criddle respectively.

Furthermore, the statistical methods show agreeable results for the Granger-Gray, Penman-Monteith, Matt-Shuttleworth and Morton CRAE. Notably, the Priesley-Taylor with the highest r^2 performs the best according to the statistical methods. The Makkink method has, next to the close regression fit to 1:1, an agreeable Index of Agreement value of 0.77 and Winter et al. (1996), also show agreeable results when comparing with EBC method. The temperature-based methods have the lowest results for the statistical methods. For this research the original value of 1.26 of the Priesley-Taylor alpha coefficient has been used. Optimization of this coefficient results in a value of 1.36 and is far below the value between 1.70 and 1.75 for semi-arid areas proposed by Jensen et al. (1990). With the new alpha coefficient the RMSE, NSE, IA and r^2 show that the Brutsaert-Strickler, Szilagyi-Jozsa and Priesley-Taylor methods show better results. The selected approach of calculating the Bowen ratio is of importance. Statistical analysis results improve when using vapor pressure

deficit or a calculation based on equilibrium temperature instead of actual or saturation vapor pressure gradient.

5.2 Water storage implications

With the results of the statistical analysis six evaporation methods have been selected to analyze their sensitivity to water storage. Of these six methods one temperature-based, two radiation-based, one wind and resistances-based and two actual evaporation methods have been selected. Only one wind and resistances-based method is chosen, because they show similar behavior and are all derived from one equation. The results of the water storage model show that water stored of about 13.4 % of open pond capacity and about 7.6 % of sand dam capacity is evaporated in the period of 70 days. Deviations of the evaporative fracture of the six methods to the evaporative fracture of the EBC method range from -24 to 10 % for the open pond and -27 to 8 % for the sand dam. The deviation in fluxes amount up to 3.5 and 2.5 % of total water harvested in the period of 70 days for open pond and sand dam respectively. While these values are for a 70-day period, they can grow larger if longer time periods are assigned. However, the sensitivity of the selected evaporation method on modeled water storage with longer time periods will be smaller for sand dams than open pond in which evaporation will not be assumed zero below the depth of 0.9 meter. Craig (2006) strengthens this issue by analyzing the influence of storage size, depth and water temperature to open pond evaporation and concludes that larges system will have less evaporative losses, with a decline in intensity of up to 5 %. While this research uses calculated evaporation from meteorological parameters measured in the study area, it can potentially differ when measuring the factors above open water or a riverbed. More specifically, water has a different heat capacity and albedo, and can therefore alter the calculations. Condie & Webster (1997), argue that gradients of temperature, humidity and wind speed above a water body can significantly influence evaporation. Depending on the WHS and the desired range of accuracy a sensitivity analysis of evaporative fracture resulted by evaporation methods of modeled water storage can be implemented. Selecting an evaporation method can lead up to 27 % of deviation of evaporative fracture. However, these deviations are a small fracture compared to the total water harvested flux.

6 Conclusion

For this research sixteen method of evaporation are analyzed in order to assess the influence of the methods on modeled water storage, whether the methods are complex or solely based on one meteorological factor. When analyzing the behavior of the four categories of evaporation methods, the temperature-based methods are relatively stable due to minor fluctuations in daily temperature. Subsequently, they show a standard deviation between 0.21 and 0.31 mm day^{-1}. Evaporation methods, which also have radiation and other meteorological factors as input, show more sensitivity to daily conditions and have standard deviations between 0.7 and 1.2 mm day^{-1}.

Agreeable results of correlation coefficients and r^2 values have been calculated by comparing evaporation methods within their category. The r^2 values of the heat fluxes compared with the available energy show agreeable results for the following methods: Priesley-Taylor (0.99), Bowen Ratio Energy Balance (0.88), Brutsaert-Strickler (0.71), Granger-Gray (0.95) and Morton CRAE (0.76). Evaporation methods with regression line close to the energy balance closure fit of 1:1 are Makkink, Penman-Monteith, Matt-Shuttleworth, Granger-Gray and Morton CRAE. Temperature-based evaporation methods show horizontal regression lines, as the methods are not sensitive to the available energy. The results of the statistical analysis show agreeable output for the Penman-Monteith, Priestley-Taylor, Granger-Gray and Morton CRAE. All evaporation methods, except the temperature-based methods, have significant p-values. Optimizing the Priesley-Taylor coefficient results in a value of 1.36. Sensitivity analysis for the multiple BREB approaches shows the most agreeable results for BREB calculated with vapor pressure deficit.

For the modeled water storage of the sand dam deviation of the evaporative fracture of -27 and 8 % are computed and for the open pond -24 and 10 %. The largest deviations are calculated with the Thornthwaite method. The results show values of within 3.5 and 2.5 % of total water harvested for the period for the open pond and sand dam respectively. Only when dealing with smaller fluxes of water storage or evaporation calculation accuracy, the choice of evaporation method is important. This study shows that the best results for the semi-arid region of the study area in Kitui are calculated with the calibrated Priestley-Taylor method and the BREB method computed with vapor pressure deficit. Other methods showing agreeable results are Priesley-Taylor, Granger-Gray and Morton CRAE. When dealing with limited data, e.g. one or two meteorological factors, the Priesley-Taylor shows the best results and otherwise Thornthwaite method if only temperature data is available.

Appendix

A1 List of equations

A2 Nomenclature

Symbol	Description	Unit
c	Constant used in Blaney-Criddle evaporation method	-
c_1	Constant used in Makkink evaporation method	-
c_2	Constant used in Makkink evaporation method	mm day^{-1}
c_p	Specific heat	MJ kg^{-1} ^{o}C^{-1}
d_r	Inverse relative distance Earth-Sun	-
E	*Evaporation*	
E_{act}	Actual evaporation	mm day^{-1}
E_{pot}	Potential evaporation	mm day^{-1}
E_{wet}	Wet evaporation	mm day^{-1}
ET	Reference crop evapotranspiration	mm day^{-1}
E_a	Winds speed function	mm day^{-1}
e_v	Enthalpy of vaporization	J kg^{-1}
e^{o}	*Tetan's equation*	
e_a	Actual vapor pressure	kPa
e_s	Saturation vapor pressure	kPa
e_s^*	Saturation vapor pressure with equilibrium temperature	kPa
f_v	Constant in Morton's method	W mbar^{-1}
G	Ground heat flux	MJ day^{-1}
G_g	Ratio actual to potential evaporation	-
G_{sc}	Solar constant	MJ min^{-1}
h	Height of crop	m
H	Sensible heat flux	MJ day^{-1}
J	Julian day	-
I	Thermal heat index	-
K	Von karman's constant	-
L	Length of the day	hour
LAI	Leaf area index	m^2 m^{-2}
P	Atmospheric pressure	kPa
p_y	Ratio actual and total annual daytime hours	-
r	*Resistance*	
r_a	Aerodynamic resistance	s m^{-1}
r_c^{50}	Resistance constant at height of 50 meters	s m^{-1}
r_l	Bulk stomatal resistance of well-illuminated leaf	s m^{-1}
r_s	(Bulk) surface or canopy resistance	s m^{-1}
R	*Radiation*	
R_e	Extraterrestrial radiation	MJ day^{-1}
R_n	Net radiation	MJ day^{-1}
R_{nl}	Net longwave radiation	MJ day^{-1}
R_{ns}	Net shortwave radiation	MJ day^{-1}
R_s	Solar radiation	MJ day^{-1}
R_{so}	Clear sky solar radiation	MJ day^{-1}
R_w	Gas constant for water vapor	J K^{-1} kg^{-1}

RH	Relative humidity	%
T	*Temperature*	
T_a	Air Temperature	ºC
T_d	Dew point temperature	ºC
T_e	Equilibrium temperature	ºC
T_K	Temperature in Kelvin	ºK
T_s	Surface temperature	ºC
U_z	Wind speed at height z	m s^{-1}
VPD	Vapor pressure deficit	kPa
z	*Elevation or height above sea level*	m
z_h	Height of humidity measurements	m
z_m	Height of wind speed measurements	m
z_{oh}	Roughness length governing heat and vapor transfer	m
z_{om}	Roughness length governing momentum transfer	m
$\propto_{TW}$	Function of thermal heat index	-
$\propto_{PT}$	Priesley-Taylor coefficient	-
β	Constant in Morton's method	ºC
γ	Psychrometric constant	kPa $ºC^{-1}$
δ	Solar declination	rad
Δ	Slope of saturation vapor pressure curve	kPa $ºC^{-1}$
ε	Ratio molecular weight of water vapor/dry air	-
ϵ_s	Land surface emissivity in Morton's method	-
λ	Latent heat of vaporization	MJ KG^{-1}
σ	Stefan-Boltzmann constant	W day^{-1} $ºK^{-4}$
φ	Latitude	rad
ω	Sunset hour angle	rad

*Symbols used for the statistical analysis are not included in this table.

A3 Measuring devices supplement

Rainfall is measured in a tipping bucket rain gauge in which a pulse is sending every 0.1mm of rainfall per square meter. The aerodynamically designed rain collector is shaped to minimalize errors when sampling wind driven rain and meets the specifications of the World Meteorological Organization. In order to measure the wind speed an anemometer is constructed on the weather station at the top with a height of 2 meters. The anemometer is programmed to log the average wind speed and maximum wind speed over the selected interval by sending pulses to the data logger. Furthermore, two temperature and humidity probes are constructed to the weather station and is designed to capture atmospheric temperature and relative humidity measurements. Both sensors are set in one device with a thin film polymer sensor for humidity and a resistive platinum sensor for temperature measurements. The devices are set at a height of 1.25m and 1.75m. Solar radiation is measured with a net radiometer. The net radiometer consists of a thermopile sensor, which measures the algebraic sum of incoming and outgoing radiation. It can measure both short and long wave radiation. The measured incoming radiation received from a 180 degrees view of the hemisphere consists of long-wave radiation from the sky, direct and diffusive solar radiation. The measured outgoing radiation received from the surface of the soil consists of reflected solar radiation and long-wave terrestrial radiation. The radiometer is sensitive to wind and a theoretical correction can be made calculating the corrected solar irradiance with the following equation:

$$E_{solar\,(corr)} = E_{solar} \cdot (1 + x \cdot v^{3/4}) \qquad (3)$$

where x is an empirical constant of approximately 0.01 and v the wind speed in m/s. Next to the net radiometer another radiation measuring device, a pyranometer, is attached to the weather station. The pyranometer measures solar radiation and is a high output thermally stable sensor. It gives a voltage output that is converted to solar radiation in watt per square meter. Lastly, a barometer is constructed on the weather station and measures the atmospheric pressure. This device converts volt to pressure.

Next to the devices attached on the pole there are also the subsurface related measurements. Water content reflectometers are placed in the soil in order to measure the soil moisture. This device consists of two stainless steel rods that are connected, in order to supply power, to a circuit board. Measurements are conducted by sending a wave signal from one rod to another. The travel time of the wave between the rods is depended on the dielectric permittivity of the soil. The dielectric permittivity itself is depended on the water content of the soil. The soil heat flux is measured with the Hukseflux plate and is placed in the soil. This device uses a thermopile to measure temperature gradients across the plate. The output voltage of the Hukseflux is proportional to the differential temperature. To measure the temperature in the soil a temperature probe is buried in the soil and is only suitable for shallow burial. This probe uses a thermistor to measure soil temperature. A

thermistor measures the electrical resistance that is depended on the temperature. To avoid thermal conduction it is placed horizontally.

A4 Main script

```
################################################################################
################################################################################
#### Main script for the evaporation model and water storage model          ####
#### Timothy Tiggeloven                                                     ####
#### AWS_main.py                                                            ####
#### Python 3.5.0                                                           ####
####                                                                        ####
#### Analyzes data from automatic weather station Kitui, Kenya              ####
################################################################################
################################################################################

import time
start_time = time.time()
print ('starty start')

# import modules
import csv
import datetime
import matplotlib.dates as dt
import numpy as np
import pandas as pd
from scipy.stats import pearsonr

# import self-made functions, plots, storage model and toolbox
import Evap_functions as EF
import Evap_toolbox as tool
import Evap_plots as Eplot
import Evap_WHS as WHS

'''
Part I: Load input
====================
'''

df = pd.read_csv('Kenya_Meteo6.csv', encoding = "ISO-8859-1")
Data = df.values.tolist()
Date = np.array(Data)[4:,0]
DateTime, DateYear, DateMonth, Year, Month = [], [], [], [], []
for i in Date:
    y = datetime.datetime.strptime(i, '%d/%m/%y %H:%M')
    DateTime.append(y)
    DateYear.append(datetime.datetime.timetuple(y).tm_yday)
    DateMonth.append(datetime.datetime.timetuple(y).tm_mday)
    Year.append(datetime.datetime.timetuple(y).tm_year)
    Month.append(datetime.datetime.timetuple(y).tm_mon)

'''
Part II: Decleration of variables
================================
'''

# declaring seperate lists for selected parameters
#*************************************************************#
#*** Make sure it starts on the first measurement of the day***#
#*************************************************************#
start = 351 - 1 # start of selected period
Rn_end = 7458 - start # end of selected period
end = 9089 - 1 # end of selected period
Data = np.array(Data)[start:end,1:].astype(float)

DateNum = dt.date2num(DateTime)     # convert dates into numeric
DayNum = DateYear[start:end]        # number of the day of the year
DayNumMonth = DateMonth[start:end]  # number of the day of the month
```

```
YearNum = Year[start:end]          # number of the year
MonthNum = Month[start:end]        # number of the month of the year
Time = DateNum[start:end]          # assigns time for period

Record = Data[:,0]         # number of measurement
Bat_V = Data[:,1]          # battery voltage [V]
Bat_T = Data[:,2]          # battery temperature [C]
Tsoil_1 = Data[:,3]        # temperature sediment at 0.05m depth [C]
Tsoil_2 = Data[:,4]        # temperature  soil at 0.05m depth [C]
Tair_1 = Data[:,5]         # temperature at height 1.25m [C]
Tair_2 = Data[:,6]         # temperature at height 2m [C]
RH_1 = Data[:,7]           # relative humidity at height 1.25m [%]
RH_2 = Data[:,8]           # relative humidity at height 2m [%]
Airpress_av = Data[:,9]    # average air pressure [hPa]
Airpress_corr = Data[:,10]  # corrected average air pressure [hPa]
Pyrano = Data[:,11]        # solar radiation at height 1.5m [W/m^2]
#Rn_obs = Data[:,12]       # obs net radiation at height 1.5m [W/m^2]
Rn_corr = Data[:Rn_end,13]  # corr net radiation at height 1.5m [W/m^2]
SM_uS1 = Data[:,14]        # transmittivity sediment at depth 0.05m [us]
SM_O1 = Data[:,15]         # moisture sediment at depth 0.05m [no unit]
SM_uS2 = Data[:,16]        # transmittivity soil at depth 0.05m [us]
SM_O2 = Data[:,17]         # moisture soil at depth 0.05m [no unit]
SM_uS3 = Data[:,18]        # transmittivity soil at depth 0.1m [us]
SM_O3 = Data[:,19]         # moisture soil at depth 0.1m [no unit]
HF_1 = Data[:,20]          # heat flux sediment at depth 0.05m [W/m^2]
HF_2 = Data[:,21]          # heat flux soil at depth 0.05m [W/m^2]
Rain = Data[:,22]          # precipitation at height 1m [mm]
W_Speed = Data[:,23]       # wind speed at height 2.25m [m/s]
W_Dir = Data[:,24]         # wind direction at height 2.25m [degrees]

'''
Part III: Transforming data
==============================
'''

# transform data to daily mean, min or max
Time_day = tool.todaily(Time, Time, 'min')
Tair_1max = tool.todaily(Tair_1, Time, 'max')
Tair_1min = tool.todaily(Tair_1, Time, 'min')
Tair_1mean = tool.todaily(Tair_1, Time, 'mean')
Tair_2max = tool.todaily(Tair_2, Time, 'max')
Tair_2min = tool.todaily(Tair_2, Time, 'min')
RH_1max = tool.todaily(RH_1, Time, 'max')
RH_1min = tool.todaily(RH_1, Time, 'min')
RH_1mean = tool.todaily(RH_1, Time, 'mean')
RH_2max = tool.todaily(RH_2, Time, 'max')
RH_2min = tool.todaily(RH_2, Time, 'min')
RH_2mean = tool.todaily(RH_2, Time, 'mean')
Rn_corr_mean = tool.todaily(Rn_corr, Time, 'mean')
HF_2mean = tool.todaily(HF_2, Time, 'mean')
W_Speed_mean = tool.todaily(W_Speed, Time, 'mean')
Airpress_corr_mean = tool.todaily(Airpress_corr, Time, 'mean')
Rs_mean = tool.todaily(Pyrano, Time, 'mean')
dayofyear = tool.todaily(DayNum, Time, 'min')
dayofmonth = tool.todaily(DayNumMonth, Time, 'min')
numofyear = tool.todaily(YearNum, Time, 'min')
monthofyear = tool.todaily(MonthNum, Time, 'min')
Tsoil_2mean = tool.todaily(Tsoil_2, Time, 'mean')
Tsoil_2max = tool.todaily(Tsoil_2, Time, 'max')
Tsoil_2min = tool.todaily(Tsoil_2, Time, 'min')
Rain_day = tool.todaily(Rain, Time, 'sum')
Tsurface = [((i + j) / 2 + (k + l) / 2) / 2 for i,j,k,l in zip(Tair_1min,\
          Tair_1max, Tsoil_2min, Tsoil_2max)]
```

```
# replace nan values with daily mean/max/min and between gaps
W_Speed = tool.nanreplace(W_Speed, Time, 'mean')

# watt/m2 to MJ/m2/day
Rn_corr_mean = [float(i) * 0.0864 for i in Rn_corr_mean]
Rn_corr = [float(i) * 0.0864 for i in Rn_corr]
Rs_mean = [float(i) * 0.0864 for i in Rs_mean]
HF_2mean = [float(i) * 0.0864 for i in HF_2mean]
HF_1 = [float(i) * 0.0864 for i in HF_1]
HF_2 = [float(i) * 0.0864 for i in HF_2]

'''
Part IV: Evaporation calculations
==================================
'''

# Priestley-Taylor constant [-]
alpha = 1.26

# calculate extraterrestrial radiation
suneq = tool.sunrise(dayofyear)
Re_day = suneq[1]

# calculate 15 min average fao penman-monteith evaporation
FAO_soil = EF.FAO_15min(W_Speed, Tair_1, RH_1, Rn_corr, HF_2, Airpress_corr)
FAO_sediment = EF.FAO_15min(W_Speed, Tair_1, RH_1, Rn_corr, HF_1, Airpress_corr)

# calculate meteorological variables, thornthwaite parameters and breb variables
Tmeanmax = [(i + j) / 2 for i,j in zip(Tair_1max, Tair_2max)]
Tmeanmin = [(i + j) / 2 for i,j in zip(Tair_1min, Tair_2min)]
RHmeanmax = [(i + j) / 2 for i,j in zip(RH_1max, RH_2max)]
RHmeanmin = [(i + j) / 2 for i,j in zip(RH_1min, RH_2min)]
RH_2mean = [(i + j) / 2 for i,j in zip(RH_2min, RH_2max)]
meteo_var = tool.meteo(Tair_1max, Tair_1min, RH_1max, RH_1min,\
                    Airpress_corr_mean, W_Speed_mean, Tair_2max, Tair_2min,\
                    Tsoil_2min, Tsoil_2max)
breb1 = meteo_var
breb2 = tool.meteo(Tair_2max, Tair_2min, RH_2max, RH_2min, Airpress_corr_mean,\
                W_Speed_mean, Tair_2max, Tair_2min, Tsoil_2min, Tsoil_2max)
breb3 = tool.meteo(Tmeanmax, Tmeanmin, RHmeanmax, RHmeanmin, Airpress_corr_mean\
                , W_Speed_mean, Tair_2max, Tair_2min, Tsoil_2min, Tsoil_2max)
brebsoil = tool.meteo(Tsoil_2max, Tsoil_2min, RHmeanmax, RHmeanmin,\
                    Airpress_corr_mean, W_Speed_mean, Tair_2max, Tair_2min,\
                    Tsoil_2min, Tsoil_2max)

# replace net radiation values 2017 with calculated values of solar radiation
Rn_calc = tool.solartonet(Rn_corr_mean, Rs_mean, Re_day, meteo_var[8],\
                       Tair_1max, Tair_1min)
Rn_corr_mean = Rn_calc[0]
Rn_bin = Rn_calc[2]

# calculate evaporation depended on temperature
TW_mon = EF.TW_monthly(Tair_1mean, dayofyear, dayofmonth, monthofyear,\
                    numofyear)
TW_day = EF.TW_daily(Tair_1mean, dayofyear, dayofmonth, monthofyear, numofyear)
BC_day = EF.BC(meteo_var)
MB_day = EF.MB(Re_day, meteo_var)

# calculate evaporation depended on temperature and radiation
HS_day = EF.HS(Rs_mean, meteo_var)
JH_day = EF.JH(Rs_mean, meteo_var)
Turc_day = EF.Turc(Rs_mean, meteo_var)
MK_day = EF.MK(Rs_mean, meteo_var)
PT_day = EF.PT(Rn_corr_mean, HF_2mean, meteo_var, alpha)
```

```
# calculate evaporation with resistances parameters
FAO_day = EF.FAO_daily(Rn_corr_mean, HF_2mean, meteo_var)
MS_day = EF.MS(Rn_corr_mean, meteo_var)
Penman_day = EF.Penman(Rn_corr_mean, meteo_var)
PM_day = EF.PM(Rn_corr_mean, HF_2mean, meteo_var)

# calculate actual evaporation, complementary relationship
BS_day = EF.BS(Rn_corr_mean, meteo_var, alpha)
GG_day = EF.GG(Rn_corr_mean, HF_2mean, meteo_var)
Mort_day = EF.Morton(Rn_corr_mean, meteo_var, Penman_day)
SJ_day = EF.SJ(Rn_corr_mean, HF_2mean, Airpress_corr_mean, W_Speed_mean,\
            meteo_var, Penman_day, alpha)

# calculate multiple BREB evaporation
BREB_bin = EF.BREB(Rn_corr_mean, HF_2mean, Penman_day, breb1, breb2, breb3,\
                brebsoil)
BREB_day = BREB_bin[0]
BREB_es = BREB_bin[1]
BREB_vpd = BREB_bin[2]
BREB_te = BREB_bin[3]

# convert reference evaporation per 15 min measurement to daily measurement
FAO_soil_chunk_mean = tool.todaily(FAO_soil, Time, 'mean')

'''
Part V: Statistics
==============================
'''

# prepare data for symmary statistics of the parameters
parameters = [Tair_1, meteo_var[0], meteo_var[17], RH_1mean, RH_2mean,\
            Rn_corr_mean, meteo_var[3], HF_2mean]
calc_par = [Re_day, meteo_var[12], meteo_var[1], meteo_var[2], meteo_var[4],\
           meteo_var[8], meteo_var[9], meteo_var[10], meteo_var[14],\
           meteo_var[15]]
headers1 = ['T1', 'T2', 'RH1', 'RH2', 'Rs', 'Rn', 'Ws', 'G']
headers2 = ['Re', 'Td', 'lv', 'psy', 'es', 'ea', 'VPD', 'slope', 'ra', 'H']

'''
# print the summary statistics
def printstats(head, par):
    for i,j in zip(head, par):
        print("%s mean: %s" % (i,np.nanmean(j)))
        print("%s std: %s" % (i,np.nanstd(j)))
printstats(headers1, parameters)
printstats(headers2, calc_par)
print("Rain sum: %s" % np.nansum(Rain_day))
'''

Ea_day = []
for i,j in zip(meteo_var[3], meteo_var[9]):
    Ea = (1.313 + 1.381 * i) * j
    Ea_day.append(Ea)

# calculate equilibrium heat flux (Rn, lv, penman, psy, Ta, ea, steps, sign)
Te = tool.equitemp(Rn_corr_mean, meteo_var[1], Penman_day, meteo_var[2],\
                meteo_var[0], meteo_var[8])
Ta = meteo_var[0]
He = []
for i,j,k in zip (Ta, Te, meteo_var[14]):
    z = 1.2 * 1.013 * pow(10, -3) * ((i - j) / k)
    He.append(z)
```

```
# EBC calculation
tmean = meteo_var[0]
RG = [i - j for(i,j) in zip(Rn_corr_mean, HF_2mean)]
EBC_day = [(i - j) / k for(i,j,k) in zip(RG, meteo_var[15], meteo_var[1])]

# optimal alpha (PT, BS, SJ)
alpha_list = []
for i,j,k,l,m,n in zip(meteo_var[10], meteo_var[2], Rn_corr_mean, HF_2mean,\
                    meteo_var[1], EBC_day):
   z = n / ((i / (i + j)) * (k - l) / m)
   alpha_list.append(z)

alpha_new = np.nanmean(alpha_list)
PT_opt = EF.PT(Rn_corr_mean, HF_2mean, meteo_var, alpha_new)
BS_opt = EF.BS(Rn_corr_mean, meteo_var, alpha_new)
SJ_opt = EF.SJ(Rn_corr_mean, HF_2mean, Airpress_corr_mean, W_Speed_mean,\
            meteo_var, Penman_day, alpha_new)

alpha_arid = 1.7
PT_arid = EF.PT(Rn_corr_mean, HF_2mean, meteo_var, alpha_arid)
BS_arid = EF.BS(Rn_corr_mean, meteo_var, alpha_arid)
SJ_arid = EF.SJ(Rn_corr_mean, HF_2mean, Airpress_corr_mean, W_Speed_mean,\
            meteo_var, Penman_day, alpha_arid)

# prepare data for EBC plots
TW_EH = [i * j + k for(i,j,k) in zip(TW_day, meteo_var[1], meteo_var[15])]
BC_EH = [i * j + k for(i,j,k) in zip(BC_day, meteo_var[1], meteo_var[15])]
MB_EH = [i * j + k for(i,j,k) in zip(MB_day, meteo_var[1], meteo_var[15])]
Turc_EH = [i * j + k for(i,j,k) in zip(Turc_day, meteo_var[1], meteo_var[15])]
HS_EH = [i * j + k for(i,j,k) in zip(HS_day, meteo_var[1], meteo_var[15])]
MK_EH = [i * j + k for(i,j,k) in zip(MK_day, meteo_var[1], meteo_var[15])]
PT_EH = [i * j + k for(i,j,k) in zip(PT_day, meteo_var[1], meteo_var[15])]
BREB_EH = [i * j + k for(i,j,k) in zip(BREB_day, meteo_var[1], meteo_var[15])]
Pen_EH = [i * j + k for(i,j,k) in zip(Penman_day, meteo_var[1], meteo_var[15])]
PM_EH = [i * j + k for(i,j,k) in zip(PM_day, meteo_var[1], meteo_var[15])]
FAO_EH = [i * j + k for(i,j,k) in zip(FAO_day, meteo_var[1], meteo_var[15])]
MS_EH = [i * j + k for(i,j,k) in zip(MS_day, meteo_var[1], meteo_var[15])]
BS_EH = [i * j + k for(i,j,k) in zip(BS_day, meteo_var[1], meteo_var[15])]
SJ_EH = [i * j + k for(i,j,k) in zip(SJ_day, meteo_var[1], meteo_var[15])]
GG_EH = [i * j + k for(i,j,k) in zip(GG_day, meteo_var[1], meteo_var[15])]
Mort_EH = [i * j + k for(i,j,k) in zip(Mort_day, meteo_var[1], meteo_var[15])]
diff = [i-j for(i,j) in zip(PM_EH, RG)]

# prepare data frame for correlation, r2 and more
headers = ['TW','BC','MB','Turc', 'HS', 'MK', 'PT', 'BREB', 'Pen', 'PM', 'FAO',\
        'MS','BS','SJ','GG','Mort','EBC']
Evapdf = pd.DataFrame(np.column_stack([TW_day,BC_day,MB_day,Turc_day,HS_day,\
                MK_day,PT_day,BREB_day,Penman_day,PM_day,FAO_day,MS_day,\
                BS_day,SJ_day,GG_day,Mort_day,EBC_day]), columns=headers)

corr = Evapdf.corr()
rsq = pow(corr, 2)
std = Evapdf.std()
mean = Evapdf.mean()
sqr = pow(Evapdf, 0.5)
meansqr = sqr.mean()

# calculate slope of every component in dataframe
EBC = Evapdf[['EBC']]
slopes = pd.DataFrame(np.linalg.pinv(EBC.T.dot(EBC)).dot(EBC.T).\
                dot(Evapdf.fillna(0)),['Slope'], Evapdf.columns)

headers_new = ['TW','BC','MB','Turc','HS','MK','PT','BREB','Pen','PM','FAO',\
            'MS','BS','SJ','GG','Mort','PTopt','BSopt','SJopt','PTarid',\
```

```
            'BSarid','SJarod','BRERB_es','BRERB_vpd','BRERB_te','EBC']
Evapdf_new = pd.DataFrame(np.column_stack([TW_day,BC_day,MB_day,Turc_day,\
                    HS_day,MK_day,PT_day,BREB_day,Penman_day,PM_day,\
                    FAO_day,MS_day,BS_day,SJ_day,GG_day,Mort_day,PT_opt,\
                    BS_opt,SJ_opt,PT_arid,BS_arid,SJ_arid,BREB_es,\
                    BREB_vpd,BREB_te,EBC_day]), columns=headers_new)

mean_new = Evapdf_new.mean()

# calculate RMSE, NSE and IA and store in dataframe
RMSEdf, IAdf, NSdf, PVdf = [], [], [], []
row, rowIA, rowNS, rowPV = [], [], [], []
square, IAsquare, NSsquare = 0, 0, 0
for h1 in headers_new:
   for h2 in headers_new:
      for i,j in zip(Evapdf_new[h1],Evapdf_new[h2]):
         square += pow(i - j, 2)
         IAsquare += pow(abs(i - mean_new[h2]) + abs(j - mean_new[h2]), 2)
         NSsquare += pow(i - mean_new[h1], 2)

      MSE = square / len(h1)
      RMSE = pow(MSE, 0.5)
      IA = 1 - (square / IAsquare)
      NS = 1 - (square / NSsquare)
      PC, PV = pearsonr(Evapdf_new[h1], Evapdf_new[h2])

      row.append(RMSE), rowIA.append(IA), rowNS.append(NS)
      rowPV.append(round(PV,2))

      square, IAsquare, NSsquare = 0, 0, 0

   RMSEdf.append(row),IAdf.append(rowIA),NSdf.append(rowNS),PVdf.append(rowPV)
   row, rowIA, rowNS, rowPV = [], [], [], []

RMSEdf = pd.DataFrame(np.column_stack(RMSEdf),columns=headers_new,\
                index=headers_new)
IAdf = pd.DataFrame(np.column_stack(IAdf),columns=headers_new,index=headers_new)
NSdf = pd.DataFrame(np.column_stack(NSdf),columns=headers_new,index=headers_new)
PVdf = pd.DataFrame(np.column_stack(PVdf),columns=headers_new,index=headers_new)

# calculate PM and EBC ratio
PM_ratio = []
EBC_ratio = []
for i in mean:
   ratio = i / mean[9]
   PM_ratio.append(ratio)
   ratio = i / mean[16]
   EBC_ratio.append(ratio)

# print ratios
#for i,j,k in zip(headers, PM_ratio, EBC_ratio):
#    print('%s PM ratio: %s and EBC_ratio %s' % (i,j,k))

# prepare diurnal data
start = 1211 # 16 oct
end = 1402
Rn_oct = [float(i) for i in Rn_corr]
G_oct = [float(i) for i in HF_2]
Time_oct = Time[start:end]
T1_oct = Tair_1[start:end]
T2_oct = Tair_2[start:end]
U2_oct = W_Speed[start:end]
Rn_oct = Rn_oct[start:end]
G_oct = G_oct[start:end]
```

```
P_oct = Airpress_corr[start:end]
RH_oct = RH_1[start:end]
diurnal_data = tool.diurnal(T1_oct,T2_oct,U2_oct,P_oct,RH_oct,Rn_oct,G_oct)
lv_oct = diurnal_data[0]
H_oct = diurnal_data[1]
PM_oct = diurnal_data[2]
EBC_oct = [(i-j-k)/l for i,j,k,l in zip(Rn_oct, G_oct, H_oct, lv_oct)]
LE_oct = [i*j for i,j in zip(PM_oct,lv_oct)]

'''
Part VI: Water harvesting
==============================
'''
# assign WH period
start = 0
end = 73

# calculate water storage with selected evaporation method
WHS_TW = WHS.WH(TW_day[start:end], Rain_day[start:end])
WHS_BC = WHS.WH(BC_day[start:end], Rain_day[start:end])
WHS_MB = WHS.WH(MB_day[start:end], Rain_day[start:end])
WHS_Turc = WHS.WH(Turc_day[start:end], Rain_day[start:end])
WHS_HS = WHS.WH(HS_day[start:end], Rain_day[start:end])
WHS_MK = WHS.WH(MK_day[start:end], Rain_day[start:end])
WHS_PT = WHS.WH(PT_day[start:end], Rain_day[start:end])
WHS_BREB = WHS.WH(BREB_day[start:end], Rain_day[start:end])
WHS_Pen = WHS.WH(Penman_day[start:end], Rain_day[start:end])
WHS_PM = WHS.WH(PM_day[start:end], Rain_day[start:end])
WHS_FAO = WHS.WH(FAO_day[start:end], Rain_day[start:end])
WHS_MS = WHS.WH(MS_day[start:end], Rain_day[start:end])
WHS_BS = WHS.WH(BS_day[start:end], Rain_day[start:end])
WHS_SJ = WHS.WH(SJ_day[start:end], Rain_day[start:end])
WHS_GG = WHS.WH(GG_day[start:end], Rain_day[start:end])
WHS_Mort = WHS.WH(Mort_day[start:end], Rain_day[start:end])
WHS_EBC = WHS.WH(EBC_day[start:end], Rain_day[start:end])
WHS_Time = Time_day[start:end]
WHS_Rain = Rain_day[start:end]

mean_op = np.nanmean(WHS_EBC[0])
mean_sa = np.nanmean(WHS_EBC[2])

# calculate statistics for water storage model (RMSE, NSE, IA, P-value)
list_op = [WHS_TW[0], WHS_MK[0], WHS_PT[0], WHS_PM[0], WHS_GG[0], WHS_Mort[0]]
list_sa = [WHS_TW[2], WHS_MK[2], WHS_PT[2], WHS_PM[2], WHS_GG[2], WHS_Mort[2]]
WHS_RMSE_op, WHS_RMSE_sa, WHS_IA_op, WHS_IA_sa = [], [], [], []
WHS_NS_op, WHS_NS_sa, WHS_PC_op, WHS_PV_op = [], [], [], []
WHS_PC_sa, WHS_PV_sa = [], []
sq_op, sq_sa, IAsq_op, IAsq_sa, NSsq_op, NSsq_sa = 0, 0, 0, 0, 0, 0
for i,j in zip(list_op, list_sa):
    mean_meas_op = np.nanmean(i)
    mean_meas_sa = np.nanmean(j)
    for op,sa,ebc_op,ebc_sa in zip(i,j,WHS_EBC[0],WHS_EBC[2]):
        sq_op += pow(op - ebc_op, 2)
        sq_sa += pow(sa - ebc_sa, 2)
        IAsq_op += pow(abs(op - mean_op)+abs(ebc_op - mean_op), 2)
        IAsq_sa += pow(abs(sa - mean_sa)+abs(ebc_sa - mean_sa), 2)
        NSsq_op += pow(op - mean_meas_op, 2)
        NSsq_sa += pow(op - mean_meas_sa, 2)

    MSE_op = sq_op / len(i)
    MSE_sa = sq_sa / len(j)
    RMSE_op = pow(MSE_op, 0.5)
    RMSE_sa = pow(MSE_sa, 0.5)
    IA_op = 1 - (sq_op / IAsq_op)
```

```
    IA_sa = 1 - (sq_sa / IAsq_sa)
    NS_op = 1 - (sq_op / NSsq_op)
    NS_sa = 1 - (sq_sa / NSsq_sa)
    PC_op, PV_op = pearsonr(WHS_EBC[0], i)
    PC_sa, PV_sa = pearsonr(WHS_EBC[2], j)

    WHS_RMSE_op.append(RMSE_op), WHS_RMSE_sa.append(RMSE_sa)
    WHS_IA_op.append(IA_op), WHS_IA_sa.append(IA_sa)
    WHS_NS_op.append(NS_op), WHS_NS_sa.append(NS_sa)
    WHS_PC_op.append(PC_op), WHS_PV_op.append(PV_op)
    WHS_PC_sa.append(PC_sa), WHS_PV_sa.append(PV_sa)

    sq_op, sq_sa, IAsq_op, IAsq_sa, NSsq_op, NSsq_sa = 0, 0, 0, 0, 0, 0

# TW PT MK PM GG Mort
TW_op = (WHS_TW[9] - WHS_EBC[9])# / WHS_EBC[9] * 100
TW_sa = (WHS_TW[10] - WHS_EBC[10])# / WHS_EBC[10] * 100
MK_op = (WHS_MK[9] - WHS_EBC[9])# / WHS_EBC[9] * 100
MK_sa = (WHS_MK[10] - WHS_EBC[10])# / WHS_EBC[10] * 100
PT_op = (WHS_PT[9] - WHS_EBC[9])# / WHS_EBC[9] * 100
PT_sa = (WHS_PT[10] - WHS_EBC[10])# / WHS_EBC[10] * 100
PM_op = (WHS_PM[9] - WHS_EBC[9]) #/ WHS_EBC[9] * 100
PM_sa = (WHS_PM[10] - WHS_EBC[10])# / WHS_EBC[10] * 100
GG_op = (WHS_GG[9] - WHS_EBC[9]) #/ WHS_EBC[9] * 100
GG_sa = (WHS_GG[10] - WHS_EBC[10])# / WHS_EBC[10] * 100
Mort_op = (WHS_Mort[9] - WHS_EBC[9])# / WHS_EBC[9] * 100
Mort_sa = (WHS_Mort[10] - WHS_EBC[10])# / WHS_EBC[10] * 100

#print("TW op: %s\nTW sa:%s\nMK op: %s\nMK sa:%s\nPT op: %s\nPT sa:%s\nPM op:\
#     %s\nPM sa:%s\n\GG op: %s\nGG sa:%s\nMort op: %s\nMort sa:%s\n"\
#     % (TW_op, TW_sa, MK_op, MK_sa, PT_op, PT_sa, PM_op, PM_sa, GG_op, GG_sa,\
#        Mort_op, Mort_sa))

end_time = time.time()
print('Script took ' + str((end_time - start_time)) + " seconds")

'''
Part VII: Plotting data
==============================
'''

''' Remove comment to run plot '''
#soilplot(Rn_corr, Pyrano, Time)
#Eplot.EBCplot(RG, TW_EH, BC_EH, MB_EH, Turc_EH, HS_EH, MK_EH, PT_EH, BREB_EH,\
#            'TW', 'BC', 'MB', 'Turc', 'HS', 'MK', 'PT', 'BREB', '1')
#Eplot.EBCplot(RG, Pen_EH, PM_EH, FAO_EH, MS_EH, BS_EH, SJ_EH, GG_EH, Mort_EH,\
#            'Pen', 'PM', 'FAO', 'MS', 'BS', 'SJ', 'GG', 'Mort', '2')
#Eplot.EBCplot(RG, TW_EH, MK_EH, PT_EH, BREB_EH, Pen_EH, PM_EH, GG_EH, Mort_EH,\
#            'TW', 'MK', 'PT', 'BREB', 'Pen', 'PM', 'GG', 'Mort', '3')
#Eplot.Corr_heatmap(corr, 'corr')
#Eplot.Corr_heatmap(rsq, 'rsq')
#Eplot.Diurnal(Time_oct, Rn_oct, G_oct, LE_oct, H_oct)
#Eplot.EBCvsALLplot(TW_day[:73], Penman_day[:73], FAO_day[:73], PT_day[:73],\
#                PM_day[:73], MK_day[:73], BS_day[:73], GG_day[:73],\
#                BC_day[:73], Turc_day[:73], MS_day[:73], JH_day[:73],\
#                MB_day[:73], HS_day[:73], BREB_day[:73], SJ_day[:73],\
#                Mort_day[:73], EBC_day[:73], Time_day[:73])
#Eplot.TREGplot(Te, Ta, 'Equilibrium temperature', 'Air temperature')
#Eplot.RnREGplot(Rn_corr_mean[:73], Rn_bin[:73], 'Measured Rn', 'Calculated Rn')
#Eplot.REGplot(Ta, Te, 'Equilibrium temperature', 'Air temperature',\
#            Rn_corr_mean[:73], Rn_bin[:73], 'Measured Rn', 'Calculated Rn')
#Eplot.rainplot(Rain_day[:73], Tair_1mean[:73], Time_day[:73])
#Eplot.WHSplot(WHS_Rain, WHS_EBC[0], WHS_EBC[2], WHS_EBC[4], WHS_Time)
```

A5 Toolbox script

```
################################################################################
################################################################################
#### Toolbox for evaporation model and water storage model                  ####
#### Timothy Tiggeloven                                                     ####
#### Evap_toolbox.py                                                        ####
#### Python 3.5.0                                                           ####
####                                                                        ####
#### Transforms data and calculates meteorological factors                  ####
################################################################################
################################################################################
'''
Functions in this script are list alphabatically
- diurnal: prepares diurnal data
- equitemp: calculate Te
- meteo: meterological variables library
- nanreplace: replace NaN value with daily mean
- solartonet: calculates net radiation and albedo
- sunrise: calculates sunrise equation
- todaily: transform data measurements to daily mean/max/min
'''

import math
import numpy as np
import pandas as pd

def diurnal(T1, T2, Uz, Pa, RH, Rn, G):
    Cp = 1.013 * pow(10, -3)   # specific heat at constant pressure [MJ/kg/C]
    ep = 0.622                 # ratio molecular weight vapor/dry air [-]
    ev = 2.501 * pow(10, 6)    # enthalpy of vaporization [J/kg]
    h = 0.01                   # crop in height [m]
    ka = 0.41                  # van Karmen constant [-]
    roua = 1.2                 # mean density of air [kg/m^3]
    rs = 70 / 86400            # resistance of the evaporative surface [s/m]
    Rw = 461.5                 # gas constant for water vapor [J/K/kg]
    WSh = 3.25                 # Wind speed measurements height [m]

    diurnal_data = []

    # calculate dew point temperature
    Td = []
    for i,j in zip(T1, RH):
        H = (np.log10(j) - 2) / 0.4343 + (17.62 * i) / (243.12 + i)
        z = 243.12 * H / (17.62 - H)
        K = i + 273.15
        z = (K / (1 - (K * np.log(j/100)) / (ev/Rw))) - 273.15
        Td.append(z)

    # calculate lv in [MJ/m2]
    lv = []
    for i in T1:
        z = 4185.5 * (751.78 - 0.5655 * (i + 273.15)) / 1000000
        z = (2.501 - (2.361 * pow(10, -3)) * i)
        lv.append(z)
    diurnal_data.append(lv)

    # calculate psychiometric constant
    psy = []
    for i,j in zip(Pa, lv):
        z = (Cp * i / 10) / (ep * j)
        psy.append(z)

    ea, es = [], []
    for i,j in zip(Td,T1):
```

```
        z = 0.6108 * np.exp((17.27 * i) / (237.3 + i))
        y = 0.6108 * np.exp((17.27 * j) / (237.3 + j))
        ea.append(z)
        es.append(y)

    # adjust wind speed data to reference height of 2 meter
    U2 = []
    for i in Uz:
        z = i * (4.87 / np.log(67.8 * WSh - 5.42))
        U2.append(z)

    # calculate vapour pressure deficit
    VPD = [i - j for i,j in zip(es,ea)]

    # calculate slope of saturation vapour
    slope = []
    for i,j in zip(es, T1):
        z = (4098 * i) / np.power(j + 237.3, 2)
        slope.append(z)

    # calculate aerodynamic resistance with d, zom, zoh and wind speed
    d = 2 / 3 * h
    zom = 0.123 * h
    zoh = 0.0123 * h
    ra = []
    for i in U2:
        z = np.log((2 - d) / zom) * np.log((1.25 - d) / zoh) / (pow(ka, 2) * i)\
            / 86400
        ra.append(z)

    # calculate Penman-Monteith
    ET0 = []
    for i,j,k,l,m,n,o in zip(psy, slope, Rn, G, VPD, ra, lv):
        z = ((j * (k - l) + roua * Cp * m / n) / (j + i * (1 + rs /n))) / o
        ET0.append(z)

    # calculate sensible heat flux [MJ/m2]
    H = []
    for i,j,k in zip (T2, T1, ra):
        z = roua * Cp * ((i - j) / k)
        H.append(z)

    return lv, H, ET0

def equitemp(Rn, lv, penman, psy, Ta, ea):
    sign = 0.01
    steps =  0.001
    lim = 3000
    Te = []
    a = 0
    for i,j,k,l,m,n in zip(Rn, lv, penman, psy, Ta, ea):
        x = m - steps
        diff = 1
        difflow = 1
        Tlow = x
        countlow = 0
        count = 0
        a += 1
        while (diff >= sign):
            es = 0.6108 * np.exp((17.27 * x) / (237.3 + x))
            z = (i / (j * k)) - (1 + (l * (x - m) / (es - n)))
            diff = abs(z)
            y = x
            x -= steps
```

```
            count += 1

            # save the lowest difference
            if diff < difflow:
                difflow = diff
                Tlow = y
                countlow = count
            # if limit is reached, assign lowest difference
            if count > lim:
                y = Tlow
                diff = 0

        Te.append(y)

    return Te

def meteo(tmax, tmin, RHmax, RHmin, Pa, Uz, tmax2, tmin2, soil1, soil2):
    Cp = 1.013 * pow(10, -3)   # specific heat at constant pressure [MJ/kg/C]
    Cpj = 1013                 # heat capacity of air [J/kg/C]
    ev = 2.501 * pow(10, 6)    # enthalpy of vaporization [J/kg]
    ep = 0.622                 # ratio molecular weight vapor/dry air [-]
    Ez = 1208                  # elevation of measurements [m]
    h = 0.01                  # crop in height [m]
    ka = 0.41                  # van Karmen constant [-]
    roua = 1.2                 # mean density of air [kg/m^3]
    Rw = 461.5                   # gas constant for water vapor [J/K/kg]
    WSh = 3.25                   # wind speed measurements height [m]

    # calculate mean T and RH based on max and min T and RH per day
    tmean = [(i + j) / 2 for i,j in zip(tmax, tmin)]
    tmean2 = [(i + j) / 2 for i,j in zip(tmax2, tmin2)]
    RHmean = [(i + j) / 2 for i,j in zip(RHmax, RHmin)]
    smean = [(i + j) / 2 for i,j in zip(soil1, soil2)]
    surfmean = [(i + j) / 2 for i,j in zip(smean, tmean)]

    # calculate dew point temperature
    Td = []
    for i,j in zip(tmean, RHmean):
        H = (np.log10(j) - 2) / 0.4343 + (17.62 * i) / (243.12 + i)
        z = 243.12 * H / (17.62 - H)
        K = i + 273.15
        z = (K / (1 - (K * np.log(j/100)) / (ev/Rw))) - 273.15
        Td.append(z)

    # Calculate latent heat of vaporization
    lv = []
    for i in tmean:
        z = 4185.5 * (751.78 - 0.5655 * (i + 273.15)) / 1000000
        z = 2.501 - (2.361 * pow(10, -3)) * i
        lv.append(z)

    # calculate psychiometric constant
    psy = []
    for i,j in zip(Pa, lv):
        z = (Cp * i / 10) / (ep * j)
        psy.append(z)

    # adjust wind speed data to reference height of 2 meter
    U2 = []
    for i in Uz:
        z = i * (4.87 / np.log(67.8 * WSh - 5.42))
        U2.append(z)

    # calculate mean, max, min and Td saturation vapour
```

```
    es, emean, emax, emin, esdew = [], [], [], [], []
    for i,j,k,l in zip(tmean, tmax, tmin, Td):
        z = 0.6108 * np.exp((17.27 * i) / (237.3 + i))
        y = 0.6108 * np.exp((17.27 * j) / (237.3 + j))
        x = 0.6108 * np.exp((17.27 * k) / (237.3 + k))
        w = (x + y) / 2
        v = 0.6108 * np.exp((17.27 * l) / (237.3 + l))
        emean.append(z), emax.append(y), emin.append(x), es.append(w)
        esdew.append(v)

    # calculate actual vapour
    ea = []
    for i,j,k,l in zip(RHmax, RHmin, emax, emin):
        z = ((l * i / 100) + (k * j / 100)) / 2
        ea.append(z)

    # calculate actual vapor with dew point temperature
    eadew = []
    for i in Td:
        z = 0.6108 * np.exp((17.27 * i) / (237.3 + i))
        eadew.append(z)
    ea = eadew

    # calculate vapour pressure deficit
    VPD = [i - j for i,j in zip(es,ea)]

    # calculate slope of saturation avpour
    slope = []
    for i,j in zip(emean, tmean):
        z = (4098 * i) / np.power(j + 237.3, 2)
        slope.append(z)

    # calculate aerodynamic resistance with d, zom, zoh and wind speed
    d = 2 / 3 * h
    zom = 0.123 * h
    zoh = 0.0123 * h
    ra = []
    for i in U2:
        z = np.log((2 - d) / zom) * np.log((1.25 - d) / zoh) / (pow(ka, 2) * i)\
            / 86400
        ra.append(z)

    # calculate surface temperature
    y, ts = np.array([-0.05, 1.25, 2.00]), []
    '''
    for i,j,k in zip(smean, tmean2, tmean):
        x = np.array([i,j,k])
        m, b = np.polyfit(np.log(x), y, 1)
        surface = np.exp(-b/m)
        ts.append(surface)
    '''

    # calculate sensible heat flux
    H = []
    for i,j,k in zip (tmean2, tmean, ra):
        z = roua * Cp * ((i - j) / k)
        H.append(z)

    return tmean, lv, psy, U2, es, emean, emax, emin, ea, VPD, slope, RHmean,\
           Td, esdew, ra, H, eadew, tmean2, ts, smean

def nanreplace(datalist, time, stat):
    '''
    Replaces nan values in list with daily mean/max/min
```

```
    '''
    # calculate and correct for offset of the beginning of the moment of the day
    time = [i - time[0]%1 for i in time]

    # store values in list per day or gaps
    a, b, data, NAN_data = 0, 1, [], []
    for i,j in zip(time, datalist):
        if a == 0:
            a += 1
            NAN_data.append(j)
        else:
            if (i%1 == 0):
                data.append(NAN_data)
                NAN_data = []
                NAN_data.append(j)
            elif b == len(time):
                NAN_data.append(j)
                data.append(NAN_data)
            else:
                NAN_data.append(j)
        b += 1

    # replace nan values with daily average or average between gaps
    a, b, new_data = 0, 0, []
    data = np.array(data)
    for i in data:
        for j in i:
            if (j != j):
                if stat == 'mean':
                    z = np.nanmean(data[a])
                elif stat == 'max':
                    z = np.nanmax(data[a])
                elif stat == 'min':
                    z = np.nanmin(data[a])
                if np.isnan(z) == True:
                    z = b
                    new_data.append(z)
                else:
                    new_data.append(z)
                b = z
            else:
                new_data.append(j)
        a += 1

    return new_data

def solartonet(Rn, Rs, Ra, ea, tmax, tmin):
    Ez = 1208                # elevation of measurements [m]
    sig = 4.903 * pow(10, -9)  # Stefan-Boltzmann constant [MJ/K4/m2/day]

    # calculate clear-sky solar radiation
    Rso = []
    for i in Ra:
        z = (0.75 + 2 * pow(10, -5) * Ez) * i
        Rso.append(z)

    # calculate longwave net radiation
    Rnl = []
    for i,j,k,l,m in zip(tmax, tmin, ea, Rs, Rso):
        z = sig * ((pow(i + 273.16, 4) + pow(j + 273.16, 4)) / 2) *\
            (0.34 - 0.14 * pow(k, 0.5)) * (1.35 * (l / m) - 0.35)
        Rnl.append(z)

    # calculate average albedo
```

```
    Rn_al, Rs_al, Rnl_al = Rn[:73], Rs[:73], Rnl[:73]
    albedo_list = []
    for i,j,k in zip(Rn_al, Rs_al, Rnl_al):
        z = 1 - ((i + k) / j)
        albedo_list.append(z)
    albedo = np.nanmean(albedo_list)

    # calculate net radiation for assigned period
    Rn_bin = []
    for i,j in zip(Rs, Rnl):
        z = (1 - albedo) * i - j
        Rn_bin.append(z)

    Rn_calc = Rn_al
    for i in Rn_bin[73:]:
        Rn_calc.append(i)

    return Rn_calc, albedo_list, Rn_bin

def sunrise(DayNum):
    Gsc = 0.0820              # solar constant [MJ/m2/min]
    lat = -1.1315313          # latitude in degrees [degrees]
    om = lat * math.pi / 180   # latitude in radians

    # calculate photoperiod
    daylight, Re = [], []
    for i in DayNum:
        dr = 1 + 0.033 * math.cos(2 * i * math.pi / 365)
        delta = 0.409 * math.sin(2 * i * math.pi / 365 - 1.39)
        ws = math.acos(-math.tan(om) * math.tan(delta))
        hours = 24 * ws / math.pi
        daylight.append(hours)
        z = (24 * 60 / math.pi) * Gsc * dr * (ws * math.sin(om) *\
            math.sin(delta) + math.cos(om) * math.cos(delta) * math.sin(ws))
        Re.append(z)

    return daylight, Re

def todaily(datalist, time, stat):
    '''
    Transforms data measurements to daily mean/max/min
    '''
    # calculate and correct for offset of the beginning of the moment of the day
    time = [i - time[0]%1 for i in time]

    # find gaps in data and count measurements to be discarded
    a, b, chunklen = 0, 0, []
    for i in time:
        if a == 0:
            a += 1
            b += 1
        else:
            if (i%1 == 0):
                chunklen.append(b)
                b = 1
            else:
                b += 1

    # calculate which sequence in data of days needs te be discarded
    a, delstart, delend = 0, [], []
    for i in chunklen:
        if i == 96:
            a += 1
        else:
```

```
        z = a * 96
        y = a * 96 + i
        delstart.append(z), delend.append(y)

# discard the data of days which possess gaps
delstart.reverse(), delend.reverse()
for i,j in zip(delstart, delend):
    Datax = np.delete(datalist,(np.r_[i:j]))

# yield n-sized chunks in list and succeed in list of measurements per day
chunks = lambda l, n: [l[x: x+n] for x in range(0, len(l), n)]
Data_chunk = chunks(Datax, 96)

# calculate mean, minimal and maximum values per day and store in list
if stat == 'mean':
    Data_day = [float(np.nanmean(i)) for i in Data_chunk]
elif stat == 'max':
    Data_day = [float(np.nanmax(i)) for i in Data_chunk]
elif stat == 'min':
    Data_day = [float(np.nanmin(i)) for i in Data_chunk]
elif stat == 'sum':
    Data_day = [float(np.nansum(i)) for i in Data_chunk]

return Data_day
```

A6 Evaporation model

```
################################################################################
#### Evaporation model                                                      ####
#### Timothy Tiggeloven                                                     ####
#### Evap_functions.py                                                      ####
#### Python 3.5.0                                                           ####
####                                                                        ####
#### Calculates various evaporation methods                                 ####
################################################################################
################################################################################

'''
The following methods of evaporation are used in this script (alphabetically):
- Blaney-Criddle
- Bowen Ratio Energy Balance
- Brutsaert-Strickler
- FAO Penman-Monteith (daily and 15 min average)
- Granger-Gray
- Hargreaves-Samani
- Jensen-Haise
- Makkink
- Matt-Shuttleworth
- McGuiness-Bordne
- Morton CRAE
- Penman
- Penman-Monteith
- Priestly-Taylor
- Szilagyi-Jozsa
- Thornthwaite (daily and monthly)
- Turc
'''

# import libraries
import Evap_toolbox as tool
import math
import numpy as np

# declare global variables
Cp = 1.013 * pow(10, -3)   # specific heat at constant pressure [MJ/kg/C]
ep = 0.622                 # ratio molecular weight vapor/dry air [-]
epmo = 0.92                # land surface emmissivity [-]
Ez = 1208                  # elevation of measurements [m]
fz = 28                  # constant in Morton [W/m2/mbar]
h = 0.01                 # crop in height [m]
ka = 0.41                  # van Karmen constant [-]
lat = -1.1315313           # latitude in degrees [degrees]
pe = 0.27                # mean daily percentage of annual daytime hours
roua = 1.2               # mean density of air [kg/m^3]
rs = 70 / 86400            # resistance of the evaporative surface [s/m^-1]
sigma = 5.67 * pow(10, -8) # stefan-boltzmann constant in Morton [W/m2/k4]
sign = 0.001               # significance of the equilibrium Te [-]
step = 0.01              # steps in which to iterate equilibrium Te [-]
WSh = 3.25                 # Wind speed measurements height [m]


def BC(meteo):
    # unpack meteorological variables
    tmean = meteo[0]

    # calculate evaporation using Blaney-Criddle
    E = []
    for i in tmean:
        z = pe * (0.46 * i + 8)
        E.append(z)
```

```
    return E

def BREB(Rn, GF, pen1, meteo1, meteo2, meteo3, soil):
    # unpack meteorological variables
    lv = meteo3[1]
    psy = meteo3[2]
    tmean1, tmean2, tmean3, tsoilmean = meteo1[0], meteo2[0], meteo3[0], soil[0]
    es1, es2, es3 = meteo1[4], meteo2[4], meteo3[4]
    emean1, emean2, emean3 = meteo1[5], meteo2[5], meteo3[5]
    emax1, emax2, emax3 = meteo1[6], meteo2[6], meteo3[6]
    emin1, emin2, emin3 = meteo1[7], meteo2[7], meteo3[7]
    ea1, ea2, ea3 = meteo1[8], meteo2[8], meteo3[8]
    emaxsoil, eminsoil, essoil, easoil = soil[6], soil[7], soil[4], soil[8]

    # calculate equilibrium temperature and vapor with instrument 1
    Te1 = tool.equitemp(Rn, lv, pen1, psy, tmean1, ea1)
    esTe1 = [0.6108 * np.exp((17.27 * i) / (273.3 + i)) for i in Te1]

    # calculate equilibrium temperature and vapor with mean instruments
    pen2 = Penman(Rn, meteo3)
    Te2 = tool.equitemp(Rn, lv, pen2, psy, tmean3, ea3)
    esTe2 = [0.6108 * np.exp((17.27 * i) / (273.3 + i)) for i in Te2]

    # Calculate evaporation using BREB
    BREB_ea, BREB_es, BREB_vpd, BREB_te = [], [], [], []
    for i,j,k,l,m,n,o,p,q,r,s,t,u,v,a,b,c,d,e in zip(psy, tmean1, tmean2, ea1,\
    ea2, Rn, GF, lv, Te1, esTe1, Te2, esTe2, ea3, tmean3, tsoilmean, easoil,\
    essoil, es1, es2):
        '''
        Select breb with 2 instrument arms, equilibrium temperature 1 or 2
        '''
        b_ea = i * (j - k) / (l - m)   # actual vapor gradient
        b_es = i * (j - k) / (d - e)   # saturation vapor gradient
        b_vpd = i * (j - k) / (d - l)   # vpd
        b_te = i * (r - j) / (s - l)   # equilibrium gradient
        #bowen = i * (j - k) / (h - l)   # upper and lower arm winter 2007
        #bowen = i * (q - k) / (r - m)   # equilibrium 1 lower arm
        #bowen = i * (s - v) / (t - u)   # equilibrium 2 mean upper and lower
        #bowen = i * (w - j) / (x - l)   # soil temperature and lower arm
        #bowen = i * (w - k) / (x - m)   # soil temperature and upper arm
        #bowen = i * (w - k) / (y - m)   # soil temperature and upper arm
        z = ((n - o) / (1 + b_ea)) / p
        y = ((n - o) / (1 + b_es)) / p
        x = ((n - o) / (1 + b_vpd)) / p
        w = ((n - o) / (1 + b_te)) / p

        BREB_ea.append(z),BREB_es.append(y),BREB_vpd.append(x),BREB_te.append(w)

    return BREB_es, BREB_ea, BREB_vpd, BREB_te

def BS(Rn, meteo, al):
    # unpack meteorological variables
    lv = meteo[1]
    psy = meteo[2]
    U2 = meteo[3]
    VPD = meteo[9]
    slope = meteo[10]

    # calculate evaporation using Brutsaert-Strickler
    E = []
    for i,j,k,l,m,n in zip(psy, slope, Rn, U2, VPD, lv):
        Ea = (1.313 + 1.381 * l) * m
        z = (2 * al - 1) * (j / (j + i)) * (k / n) - (i / (j + i)) * Ea
```

```
        E.append(z)

    return E

def FAO_daily(Rn, GF, meteo):
    # unpack meteorological variables
    tmean = meteo[0]
    lv = meteo[1]
    psy = meteo[2]
    U2 = meteo[3]
    es = meteo[4]
    emean = meteo[5]
    emax = meteo[6]
    emin = meteo[7]
    ea = meteo[8]
    VPD = meteo[9]
    slope = meteo[10]

    # calculate evapotranspiration using FAO Penman-Monteith
    ET = []
    for i,j,k,l,m,n,o in zip(slope, Rn, GF, VPD, tmean, U2, psy):
        z = (0.408 * i * (j - k) + o *\
            (900 / (m + 273)) * n * l) / (i + o * (1 + 0.34 * n))
        ET.append(z)

    return ET

def FAO_15min(Uz, Temp, RH, Rn, GF, Pa):
    # calculate latent heat of vaporization
    lv = []
    for i in Temp:
        z = 4185.5 * (751.78 - 0.5655 * (i + 273.15)) / 1000000
        lv.append(z)

    # calculate psychiometric constant
    psy = []
    for i,j in zip(Pa, lv):
        z = (Cp * i / 10) / (ep * j)
        psy.append(z)

    # adjust wind speed data to reference height of 2 meter
    U2 = []
    for i in Uz:
        z = i * (4.87 / np.log(67.8 * WSh - 5.42))
        U2.append(z)

    # calculate saturation vapour
    es = []
    for i in Temp:
        z = 0.6108 * np.exp((17.27 * i) / (273.3 + i))
        es.append(z)

    # calculate actual vapour
    ea = [i / 100 * j for i,j in zip(RH, es)]

    # calculate vapour pressure deficit
    VPD = [i - j for i,j in zip(es,ea)]

    # calculate slope of saturation avpour
    slope = []
    for i,j in zip(es, Temp):
        z = (4098 * i) / np.power(j + 273.3, 2)
        slope.append(z)
```

```
    # calculate evapotranspiration using FAO Penman-Monteith
    ET = []
    for i,j,k,l,m,n,o in zip(slope, Rn, GF, VPD, Temp, U2, psy):
        z = (0.408 * i * (j - k) + o * (900 / (m + 273)) * n * l)\
            / (i + o * (1 + 0.34 * n))
        ET.append(z)

    return ET

def GG(Rn, G, meteo):
    # unpack meteorological variables
    tmean = meteo[0]
    lv = meteo[1]
    psy = meteo[2]
    U2 = meteo[3]
    VPD = meteo[9]
    slope = meteo[10]

    # calculate evaporation using Granger-Gray
    E = []
    for i,j,k,l,m,n,o in zip(psy, slope, Rn, G, U2, lv, VPD):
        Ea = (1.313 + 1.381 * m) * o
        Dp = Ea / (Ea + (k - l))
        Gg = 1 / (0.793 + 0.2 * np.exp(4.902 * Dp)) + 0.006 * Dp
        z = (j * Gg * ((k - l) / n) + i * Gg * Ea) / (j * Gg + i)
        E.append(z)

    return E

def HS(Rs, meteo):
    # unpack meteorological variables
    tmean = meteo[0]
    lv = meteo[1]

    # calculate evaporation using Hargreaves-Samani
    E = []
    for i,j,k in zip(tmean, Rs, lv):
        z = 0.0135 * j * (i + 17.8) / k
        E.append(z)

    return E

def JH(Rs, meteo):
    # unpack meteorological variables
    tmean = meteo[0]
    lv = meteo[1]

    # calculate evaporation using Jensen-Haise
    E = []
    for i,j,k in zip(tmean, lv, Rs):
        z = 0.025 / j * (k * (i + 3))
        E.append(z)

    return E

def MK(Rs, meteo):
    # unpack meteorological variables
    tmean = meteo[0]
    lv = meteo[1]
    psy = meteo[2]
    emean = meteo[5]
    slope = meteo[10]

    # calculate evaporation using Makkink
```

```
    E = []
    for i,j,k,l in zip(psy, slope, Rs, lv):
        z = 0.65 * (j / (j + i)) * (k / l)
        #z = 0.61 * (j / (j + i)) * (k / l) - 0.12
        E.append(z)

    return E


def MS(Rn, meteo):
    # unpack meteorological variables
    tmean = meteo[0]
    lv = meteo[1]
    psy = meteo[2]
    U2 = meteo[3]
    VPD = meteo[9]
    slope = meteo[10]

    # calculate resistance at reference height of 50m
    rc50 = 1 / pow(0.41, 2) * np.log((50 - 0.67 * h) / (0.123 * h)) *\
          np.log((50 - 0.67 * h) / (0.0123 * h)) * (np.log((2 - 0.08)\
          / 0.0148) / np.log((50 - 0.08) / 0.0148)) / 86400

    # calculate vapour pressure deficit at reference height of 50m
    VPD50 = []
    for i,j,k,l,m in zip(slope, psy, U2, VPD, Rn):
        # calculate climatological resistance
        rclim = 86400 * ((roua * Cp * l) / (i * m))
        z = ((302 * (i + j) + 70 * j * k) / (208 * (i + j) + 70 * j * k)) +\
            ((1 / rclim) * (((302 * (i + j) + 70 * j * k) / (208 * (i + j) +\
            70 * j * k)) * (208 / k) - (302 / k)))
        VPD50.append(z)

    # calculate evapotranspiration using Matt-Shuttleworth
    ET = []
    for i,j,k,l,m,n,o in zip(psy, slope, Rn, U2, VPD, VPD50, lv):
        z = (1 / o) *((j * k + (roua * Cp * l * m / rc50) *\
            n) / (j + i * (1 + rs * l / rc50)))
        ET.append(z)

    return ET

def MB(Re, meteo):
    # unpack meteorological variables
    tmean = meteo[0]
    lv = meteo[1]

    # calculate evaporation using McGuiness-Bordne
    E = []
    for i,j,k in zip(tmean, lv, Re):
        z = 1 / (68 * j) * (k * (i + 5))
        E.append(z)

    return E

def Morton(Rn, meteo, penman):
    # unpack meteorological variables
    tmean = meteo[0]
    lv = meteo[1]
    psy = meteo[2]
    U2 = meteo[3]
    es = meteo[4]
    emean = meteo[5]
    emax = meteo[6]
```

```
    emin = meteo[7]
    ea = meteo[8]
    VPD = meteo[9]
    slope = meteo[10]
    esdew = meteo[13]

    # calculate equilibrium temperature and saturation vapor at equilibrium
    Te = tool.equitemp(Rn, lv, penman, psy, tmean, ea)
    esTe = []
    for i in Te:
        z = 6.108 * np.exp((17.27 * i) / (273.3 + i))
        esTe.append(z)

    # transform units
    Rn = [i / 0.0864 for i in Rn]
    lv = [i / 0.0864 for i in lv]
    slope = [i * 10 for i in slope]
    esdew = [i * 10 for i in esdew]
    es = [i * 10 for i in es]
    psy = [i * 10 for i in psy]

    # calculate evapotranspiration using Morton CRAE
    ET = []
    for i,j,k,l,m,n,o,p,q,r in zip(lv, Rn, psy, Te, tmean, slope, es, esdew, esTe, ea):
        psp = 1 / pow((288 - 0.0065 * Ez) / 288, 5.256)
        stab = 0.28 * (1 + p / o) + j * n / (k / 10 * 0.66 * (1 / psp)\
              * pow(psp, 0.5) * fz * (o - p))
        fv = pow(psp, 0.5) * fz / stab
        z = (1 / i) * (j - (k * fv + 4 * epmo * sigma * pow(l + 273, 3))\
           * (l - m))
        ET.append(z)

    return ET

def Penman(Rn, meteo):
    # unpack meteorological variables
    tmean = meteo[0]
    lv = meteo[1]
    psy = meteo[2]
    U2 = meteo[3]
    VPD = meteo[9]
    slope = meteo[10]

    # calculate evaporation using Penman
    E = []
    for i,j,k,l,m,n in zip(psy, slope, Rn, U2, VPD, lv):
        Ea = (1.313 + 1.381 * l) * m
        z = (j / (j + i)) * (k / n) + (i / (j + i)) * Ea
        E.append(z)

    return E

def PM(Rn, GF, meteo):
    # unpack meteorological variables
    lv = meteo[1]
    psy = meteo[2]
    U2 = meteo[3]
    VPD = meteo[9]
    slope = meteo[10]
    ra = meteo[14]

    # calculate evapotranspiration using Penman-Monteith
    ET = []
    for i,j,k,l,m,n,o in zip(psy, slope, Rn, GF, VPD, ra, lv):
```

```
        z = (1 / o) * ((j * (k - l) + roua * Cp * m / n) / (j + i * (1 + rs /n)))
        ET.append(z)

    return ET

def PT(Rn, GF, meteo, al):
    # unpack meteorological variables
    tmean = meteo[0]
    lv = meteo[1]
    psy = meteo[2]
    emean = meteo[5]
    slope = meteo[10]

    # calculate evaporation using Priesley-Taylor
    E = []
    for i,j,k,l,m in zip(slope, psy, Rn, GF, lv):
        #z = al * ((i / (i + j)) * (k / m) - (l / m))
        z = al * ((i / (i + j)) * (k - l) / m)
        E.append(z)

    return E

def SJ(Rn, GF, Pa, Uz, meteo, penman, al):
    # unpack meteorological variables
    tmean = meteo[0]
    lv = meteo[1]
    psy = meteo[2]
    U2 = meteo[3]
    emax = meteo[6]
    emin = meteo[7]
    ea = meteo[8]
    VPD = meteo[9]
    slope = meteo[10]
    RHmean = meteo[11]

    # calculate equilibrium temperatur and Priestly-Taylor equilibrium
    Te = tool.equitemp(Rn, lv, penman, psy, tmean, ea)
    meteo_Te =  tool.meteo(Te, Te, RHmean, RHmean, Pa, Uz, Te, Te, Te, Te)
    PTe = PT(Rn, GF, meteo_Te, al)

    # calculate evaporation using Szilagyi-Jozsa
    E = []
    for i,j in zip(PTe, penman):
        z = (2 * i - j)
        E.append(z)

    return E

def TW_monthly(T, DayNum, DayNumMonth, MonthNum, YearNum):
    # calculate photoperiod
    daylight, omega = [], lat * math.pi / 180
    for i in DayNum:
        delta = 0.409 * math.sin(2 * i * math.pi / 365 - 1.39)
        ws = math.acos(-math.tan(omega) * math.tan(delta))
        hours = 24 * ws / math.pi
        daylight.append(hours)

    # average temperature and photoperiod data per month
    Tmonth, Tchunk, Lmonth, Lchunk = [], [], [], []
    month, Mchunk, year, Ychunk = [], [], [], []
    a, b = 0, 0
    for i,j,k,l,m in zip(T, daylight, MonthNum, YearNum, DayNumMonth):
        if a == 0:
            Tchunk.append(i),Lchunk.append(j),Mchunk.append(k),Ychunk.append(l)
```

```
        elif (len(T) - 1) == a:
            Tchunk.append(i),Lchunk.append(j),Mchunk.append(k),Ychunk.append(l)
            z, y = np.nanmean(Tchunk), np.nanmean(Lchunk)
            w, x = np.nanmean(Mchunk), np.nanmean(Ychunk)
            Tmonth.append(z), Lmonth.append(y), month.append(w), year.append(x)
            Tchunk, Lchunk, Mchunk, Ychunk = [], [], [], []
        elif (b - m) > 0:
            z, y = np.nanmean(Tchunk), np.nanmean(Lchunk)
            w, x = np.nanmean(Mchunk), np.nanmean(Ychunk)
            Tmonth.append(z), Lmonth.append(y), month.append(w), year.append(x)
            Tchunk, Lchunk, Mchunk, Ychunk = [], [], [], []
            Tchunk.append(i),Lchunk.append(j),Mchunk.append(k),Ychunk.append(l)
        else:
            Tchunk.append(i),Lchunk.append(j),Mchunk.append(k),Ychunk.append(l)
        a += 1
        b = m

    # assign number of days in a month for the specific months in data
    daysmonth = []
    days_per_month = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    for i,j in zip(month, year):
        z = days_per_month[int(i) - 1]
        if i == 2 and (j % 4 == 0 and j % 100 != 0 or j % 400 == 0):
            z = 29
        daysmonth.append(z)

    # calculate heat index
    Heat = []
    for i in Tmonth:
        z = pow((0.2 * i), 1.514)
        Heat.append(z)
    ########## alter if working with data period longer than one year ##########
    HI = sum(Heat) * 12 / len(Heat)

    # calculate function depended on heat index
    alpha = (6.75 * pow(10, -7)) * pow(HI, 3) - (7.71 * pow(10, -5)) *\
            pow(HI, 2) + (1.7912 * pow(10, -2)) * HI + 0.49239

    # calculate evaporation using Thornthwaite
    E = []
    for i,j,k in zip(Tmonth, Lmonth, daysmonth):
        if i > 26:
            z = -415.85 + 32.24 * i - pow((0.43 * i), 2)
        else:
            z = 16 * (j / 12) * (k / 30) * pow((10 * i / HI), alpha)
        E.append(z)

    return E

def TW_daily(T, DayNum, DayNumMonth, MonthNum, YearNum):
    # calculate photoperiod
    daylight, omega = [], lat * math.pi / 180
    for i in DayNum:
        delta = 0.409 * math.sin(2 * i * math.pi / 365 - 1.39)
        ws = math.acos(-math.tan(omega) * math.tan(delta))
        hours = 24 * ws / math.pi
        daylight.append(hours)

    # average temperature and photoperiod data per month
    Tmonth, Tchunk = [], []
    a, b = 0, 0
    for i,j in zip(T, DayNumMonth):
        if a == 0:
            Tchunk.append(i)
```

```
        elif (len(T) - 1) == a:
            Tchunk.append(i)
            z = np.nanmean(Tchunk)
            Tmonth.append(z)
            Tchunk = []
        elif (b - j) > 0:
            z = np.nanmean(Tchunk)
            Tmonth.append(z)
            Tchunk = []
            Tchunk.append(i)
        else:
            Tchunk.append(i)
        a += 1
        b = j

    # assign number of days in a month for the specific months in data
    daysmonth = []
    days_per_month = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    for i,j in zip(MonthNum, YearNum):
        z = days_per_month[int(i) - 1]
        if i == 2 and (j % 4 == 0 and j % 100 != 0 or j % 400 == 0):
            z = 29
        daysmonth.append(z)

    # calculate heat index
    Heat = []
    for i in Tmonth:
        z = pow((0.2 * i), 1.514)
        Heat.append(z)
    HI = sum(Heat) * 12 / len(Heat)

    # calculate function depended on heat index
    alpha = (6.75 * pow(10, -7)) * pow(HI, 3) - (7.71 * pow(10, -5)) *\
            pow(HI, 2) + (1.7912 * pow(10, -2)) * HI + 0.49239

    # calculate evaporation using Thornthwaite
    E = []
    for i,j,k in zip(T, daylight, daysmonth):
        if i > 26:
            z = (j / (12 * k)) * (-415.85 + 32.24 * i - pow((0.43 * i), 2))
        else:
            z = 16 * (j / (12 * k)) * pow((10 * i / HI), alpha)
        E.append(z)

    return E

def Turc(Rs, meteo):
    # unpack meteorological variables
    tmean = meteo[0]
    RHmean = meteo[11]

    # calculate evaporation using Turc
    E = []
    for i,j,k in zip(tmean, RHmean, Rs):
        z = 0.0133 * (23.88 * k + 50) * (i / (i + 15)) + (1 + ((50 - j) / 70))
        E.append(z)

    return E
```

A7 Water storage model

```
################################################################################
################################################################################
#### Water storage model                                                    ####
#### Timothy Tiggeloven                                                     ####
#### Evap_WHS.py                                                            ####
#### Python 3.5.0                                                           ####
####                                                                        ####
#### Analyses different methods of evaporation for water storage            ####
################################################################################
################################################################################

def WH(evap, prec):

    # Declaration of variables describing the WHS
    cap_op = 50               # capacity of open pond [m3]
    cap_sa = 1098              # capacity of sand dam [m3]
    catchment_op = 300         # catchment size of open pond [m2]
    catchment_sa = 5200000000   # catchment size of sand dam [m2]
    demand = 85.86             # demand of water per week per person [L]
    depth_op = 2               # depth of open pond [m]
    depth_sa = 3               # mean depth of sand dam [m]
    HH = 5.8                  # mean persons per household in Ethiopia
    inlet_op = cap_op/depth_op  # inlet of open pond [m2]
    inlet_sa = cap_sa/depth_sa  # inlet of sand dam [m2]
    num_op = 10                # number of households that use open pond
    num_sa = 1100              # number of people that use sand dam
    ROC = 0.58                 # run-off coefficient
    sand_evap = 0.9            # depth until which evaporation occurs [m]
    storage_op = cap_op        # set initial storage of open pond [m3]
    storage_sa = cap_sa        # set initial storage of sand dam [m3]
    threshold = 0.01           # minimum rainfall before run-off occurs [m3]

    # calculate usage of WHS in m3
    usage_op = (HH * num_op) * (demand / 7) / 1000
    usage_sa = (num_sa) * (demand / 7) / 1000

    # capacity where evaporation does not occur in sand dam
    cap_sand_evap = inlet_sa * (depth_sa - sand_evap)

    total_op, total_sa = 0, 0
    E_list_op, E_list_sa, sand_lim = [], [], []
    store_list_op, store_list_sa, usage_list_op, usage_list_sa = [], [], [], []
    # calculate storage and usage of WHS
    for i,j in zip(evap, prec):
        E = i / 1000
        pr = j / 1000

        # channel precipitation
        storage_op += inlet_op * pr
        storage_sa += inlet_sa * pr

        # run-off
        if pr > threshold:
            storage_op += catchment_op * pr * ROC
            storage_sa += catchment_sa * pr * ROC

        # evaporation open pond
        E_op = E * inlet_op
        storage_op -= E_op
        total_op += E_op

        # evaporation sand dam
        E_sa = 0
```

```
        if storage_sa > cap_sand_evap:
            diff = storage_sa - cap_sand_evap
            if diff < E:
                E_sa = diff * inlet_sa
                storage_sa -= E_sa
                total_sa += E_sa
            else:
                E_sa = E * inlet_sa
                storage_sa -= E_sa
                total_sa += E_sa

        # check capacity, depletion and usage for open pond
        if storage_op > cap_op:
            storage_op = cap_op
            harvest_op = usage_op
        elif storage_op < 0:
            storage_op, harvest_op = 0, 0
        elif storage_op < usage_op:
            harvest_op = storage_op
            storage_op = 0
        else:
            storage_op -= usage_op
            harvest_op =  usage_op

        # check capacity, depletion and usage for sand dam
        if storage_sa > cap_sa:
            storage_sa = cap_sa
            harvest_sa = usage_sa
        elif storage_sa < 0:
            storage_sa, harvest_sa = 0, 0
        elif storage_sa < usage_sa:
            harvest_sa = storage_sa
            storage_sa = 0
        else:
            storage_sa -= usage_sa
            harvest_sa =  usage_sa

        store_list_op.append(storage_op), usage_list_op.append(harvest_op)
        store_list_sa.append(storage_sa), usage_list_sa.append(harvest_sa)
        E_list_op.append(E_op), E_list_sa.append(E_sa)
        sand_lim.append(cap_sand_evap)

    # calculate total percentage evaporative fracture of WHS
    op_WHS = total_op / cap_op
    sa_WHS = total_sa / cap_sa

    return store_list_op, usage_list_op, store_list_sa, usage_list_sa,\
           sand_lim, total_op, total_sa, E_list_op, E_list_sa, op_WHS, sa_WHS
```

A8 Pyplot script

```
################################################################################
################################################################################
#### Evaporation pyplots                                                    ####
#### Timothy Tiggeloven                                                     ####
#### Evap_plots.py                                                          ####
#### Python 3.5.0                                                           ####
####                                                                        ####
#### Plots data for the evaporation model and water storage model           ####
################################################################################
################################################################################

# read modules
import csv
import datetime
import math
import matplotlib.dates as dt
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import seaborn as sns

curdir = os.getcwd()

def Diurnal(t, Rn, G, lv, H):
    ax=plt.gca()
    xfmt = dt.DateFormatter('%H')
    ax.xaxis.set_major_formatter(xfmt)
    plt.plot_date(t, Rn,'-', label='Net radiation')
    plt.plot_date(t, G,'-', label='Soil heat')
    plt.plot_date(t, lv,'-', label='Latent heat')
    plt.plot_date(t, H,'-', label='Sensible heat')
    plt.xlabel('Time', fontsize=18)
    plt.ylabel('Energy [$MJ/m^2$]', fontsize=18)
    plt.legend(fontsize=18, loc='upper left')

    plt.xticks(fontsize=18)
    plt.tick_params(axis='both', labelsize=18)

    plt.savefig(curdir+"//Diurnal_plot.png", bbox_inches='tight', dpi=300)
    plt.show()

def Corr_heatmap(corr, name):
    # Generate a mask for the upper triangle
    mask = np.zeros_like(corr, dtype=np.bool)
    mask[np.triu_indices_from(mask)] = True

    # Set up the matplotlib figure
    fig, ax = plt.subplots(figsize=(11, 9))

    # Generate a custom diverging colormap
    cmap = sns.diverging_palette(220, 20, sep=50, n=1, as_cmap=True)

    # Draw the heatmap with the mask and correct aspect ratio
    sns.set(font_scale=3)
    g = sns.heatmap(corr, mask=mask, cmap=cmap, vmax=1, vmin=0, square=True,
                linewidths=.5, cbar_kws={"shrink": .5}, ax=ax)

    g.set_xticklabels(g.get_xticklabels(), rotation = 270, fontsize = 23)
    g.set_yticklabels(g.get_yticklabels(), rotation = 0, fontsize = 23)

    fig.savefig(curdir+"//Heatmap_%s.png" % name, bbox_inches='tight', dpi=300)
    plt.show()
```

```
    #xticklabels=2, yticklabels=2,

def TREGplot(T1, T2, n1, n2):
    v1, v2 = np.array(T1), np.array(T2)
    m1, b1 = np.polyfit(v1, v2, 1)
    eq1 = 'y = ' + str(round(m1,4)) + '$x$' ' + ' + str(round(b1,4))
    corr1 = np.corrcoef(v1, v2)[0,1]
    Rsq1 = '$R^2$ = ' + str(round(pow(corr1,2),2))

    fig = plt.figure()
    ax1 = fig.add_subplot(111)
    ax1.plot(T1, T2, '.')
    ax1.plot(v1, m1*v1+b1, '-')
    ax1.plot([18, 25], [18, 25], ls="--", c=".3")
    ax1.set_ylim(19,25)
    ax1.set_xlim(19,25)
    ax1.text(19.5, 24.5, eq1)
    ax1.text(19.5, 24.2, Rsq1)
    ax1.set_xlabel('%s [$ºC$]' % n1)
    ax1.set_ylabel('%s [$ºC$]' % n2)

    fig.savefig(curdir+"//Te_reg.png", bbox_inches='tight', dpi=300)
    plt.show()

def RnREGplot(T1, T2, n1, n2):
    v1, v2 = np.array(T1), np.array(T2)
    m1, b1 = np.polyfit(v1, v2, 1)
    eq1 = 'y = ' + str(round(m1,4)) + '$x$' ' + ' + str(round(b1,4))
    corr1 = np.corrcoef(v1, v2)[0,1]
    Rsq1 = '$R^2$ = ' + str(round(pow(corr1,2),2))

    fig = plt.figure()
    ax1 = fig.add_subplot(111)
    ax1.plot(T1, T2, '.')
    ax1.plot(v1, m1*v1+b1, '-')
    ax1.plot([0, 14], [0, 14], ls="--", c=".3")
    ax1.set_ylim(4,14)
    ax1.set_xlim(4,14)
    ax1.text(4.3, 13.5, eq1)
    ax1.text(4.3, 13, Rsq1)
    ax1.set_xlabel('%s [$MJ/m^2$]' % n1, )
    ax1.set_ylabel('%s [$MJ/m^2$]' % n2)

    fig.savefig(curdir+"//Rn_reg.png", bbox_inches='tight', dpi=300)
    plt.show()

def REGplot(T1, T2, n1, n2, R1, R2, n3, n4):
    v1, v2 = np.array(T1), np.array(T2)
    m1, b1 = np.polyfit(v1, v2, 1)
    eq1 = 'y = ' + str(round(m1,4)) + '$x$' ' + ' + str(round(b1,4))
    corr1 = np.corrcoef(v1, v2)[0,1]
    Rsq1 = '$R^2$ = ' + str(round(pow(corr1,2),2))

    v3, v4 = np.array(R1), np.array(R2)
    m2, b2 = np.polyfit(v3, v4, 1)
    eq2 = 'y = ' + str(round(m2,4)) + '$x$' ' + ' + str(round(b2,4))
    corr2 = np.corrcoef(v3, v4)[0,1]
    Rsq2 = '$R^2$ = ' + str(round(pow(corr2,2),2))

    fig = plt.figure()
    ax1 = fig.add_subplot(121)
    ax1.plot(T1, T2, '.')
    ax1.plot(v1, m1*v1+b1, '-')
    ax1.plot([18, 25], [18, 25], ls="--", c=".3")
```

```
    ax1.set_ylim(19,25)
    ax1.set_xlim(19,25)
    ax1.text(19.5, 24.5, eq1, fontsize=18)
    ax1.text(19.5, 24.2, Rsq1, fontsize=18)
    ax1.set_xlabel('%s [º$C$]' % n1, fontsize=18)
    ax1.set_ylabel('%s [º$C$]' % n2, fontsize=18)
    ax1.tick_params(axis='both', labelsize=18)

    ax2 = fig.add_subplot(122)
    ax2.plot(R1, R2, '.')
    ax2.plot(v3, m2*v3+b2, '-')
    ax2.plot([0, 14], [0, 14], ls="--", c=".3")
    ax2.set_ylim(4,14)
    ax2.set_xlim(4,14)
    ax2.text(4.3, 13.5, eq2, fontsize=18)
    ax2.text(4.3, 13, Rsq2, fontsize=18)
    ax2.set_xlabel('%s [$MJ/m^2$]' % n3, fontsize=18)
    ax2.set_ylabel('%s [$MJ/m^2$]' % n4, fontsize=18)
    ax2.tick_params(axis='both', labelsize=18)

    fig.savefig(curdir+"//reg.png", bbox_inches='tight', dpi=300)
    plt.show()

def EBCplot(RGe, EH1, EH2, EH3, EH4, EH5, EH6, EH7, EH8, n1, n2, n3, n4,\
            n5, n6, n7, n8, num):
    x = np.array(RGe)
    y1, y2, y3, y4 = np.array(EH1), np.array(EH2), np.array(EH3), np.array(EH4)
    y5, y6, y7, y8 = np.array(EH5), np.array(EH6), np.array(EH7), np.array(EH8)
    m1, b1 = np.polyfit(x, y1, 1)
    m2, b2 = np.polyfit(x, y2, 1)
    m3, b3 = np.polyfit(x, y3, 1)
    m4, b4 = np.polyfit(x, y4, 1)
    m5, b5 = np.polyfit(x, y5, 1)
    m6, b6 = np.polyfit(x, y6, 1)
    m7, b7 = np.polyfit(x, y7, 1)
    m8, b8 = np.polyfit(x, y8, 1)
    eq1 = 'y = ' + str(round(m1,2)) + '$x$' ' + ' + str(round(b1,2))
    eq2 = 'y = ' + str(round(m2,2)) + '$x$' ' + ' + str(round(b2,2))
    eq3 = 'y = ' + str(round(m3,2)) + '$x$' ' + ' + str(round(b3,2))
    eq4 = 'y = ' + str(round(m4,2)) + '$x$' ' + ' + str(round(b4,2))
    eq5 = 'y = ' + str(round(m5,2)) + '$x$' ' + ' + str(round(b5,2))
    eq6 = 'y = ' + str(round(m6,2)) + '$x$' ' + ' + str(round(b6,2))
    eq7 = 'y = ' + str(round(m7,2)) + '$x$' ' + ' + str(round(b7,2))
    eq8 = 'y = ' + str(round(m8,2)) + '$x$' ' + ' + str(round(b8,2))
    corr1 = np.corrcoef(x, y1)[0,1]
    corr2 = np.corrcoef(x, y2)[0,1]
    corr3 = np.corrcoef(x, y3)[0,1]
    corr4 = np.corrcoef(x, y4)[0,1]
    corr5 = np.corrcoef(x, y5)[0,1]
    corr6 = np.corrcoef(x, y6)[0,1]
    corr7 = np.corrcoef(x, y7)[0,1]
    corr8 = np.corrcoef(x, y8)[0,1]
    Rsq1 = '$R^2$ = ' + str(round(pow(corr1,2),2))
    Rsq2 = '$R^2$ = ' + str(round(pow(corr2,2),2))
    Rsq3 = '$R^2$ = ' + str(round(pow(corr3,2),2))
    Rsq4 = '$R^2$ = ' + str(round(pow(corr4,2),2))
    Rsq5 = '$R^2$ = ' + str(round(pow(corr5,2),2))
    Rsq6 = '$R^2$ = ' + str(round(pow(corr6,2),2))
    Rsq7 = '$R^2$ = ' + str(round(pow(corr7,2),2))
    Rsq8 = '$R^2$ = ' + str(round(pow(corr8,2),2))

    fig = plt.figure()
    ax1 = fig.add_subplot(241)
    ax1.plot(RGe, EH1, '.')
```

```
ax1.plot(x, m1*x+b1, '-')
ax1.plot([3, 18], [3, 18], ls="--", c=".3")
ax1.set_ylim(3,18)
ax1.set_xlim(3,18)
ax1.text(3.1, 17.1, eq1, fontsize=18)
ax1.text(3.1, 15.9, Rsq1, fontsize=18)
ax1.set_xlabel('Rn - G $[MJ/m^2]$', fontsize=18)
ax1.set_ylabel('%s + H $[MJ/m^2]$' % n1, fontsize=18)
ax1.tick_params(axis='both', labelsize=18)

ax2 = fig.add_subplot(242)
ax2.plot(RGe, EH2, '.')
ax2.plot(x, m2*x+b2, '-')
ax2.plot([3, 18], [3, 18], ls="--", c=".3")
ax2.set_ylim(3,18)
ax2.set_xlim(3,18)
ax2.text(3.1, 17.1, eq2, fontsize=18)
ax2.text(3.1, 15.9, Rsq2, fontsize=18)
ax2.set_xlabel('Rn - G $[MJ/m^2]$', fontsize=18)
ax2.set_ylabel('%s + H $[MJ/m^2]$' % n2, fontsize=18)
ax2.tick_params(axis='both', labelsize=18)

ax3 = fig.add_subplot(243)
ax3.plot(RGe, EH3, '.')
ax3.plot(x, m3*x+b3, '-')
ax3.plot([3, 18], [3, 18], ls="--", c=".3")
ax3.set_ylim(3,18)
ax3.set_xlim(3,18)
ax3.text(3.1, 17.1, eq3, fontsize=18)
ax3.text(3.1, 15.9, Rsq3, fontsize=18)
ax3.set_xlabel('Rn - G $[MJ/m^2]$', fontsize=18)
ax3.set_ylabel('%s + H $[MJ/m^2]$' % n3, fontsize=18)
ax3.tick_params(axis='both', labelsize=18)

ax4 = fig.add_subplot(244)
ax4.plot(RGe, EH4, '.')
ax4.plot(x, m4*x+b4, '-')
ax4.plot([3, 18], [3, 18], ls="--", c=".3")
ax4.set_ylim(3,18)
ax4.set_xlim(3,18)
ax4.text(3.1, 17.1, eq4, fontsize=18)
ax4.text(3.1, 15.9, Rsq4, fontsize=18)
ax4.set_xlabel('Rn - G $[MJ/m^2]$', fontsize=18)
ax4.set_ylabel('%s + H $[MJ/m^2]$' % n4, fontsize=18)
ax4.tick_params(axis='both', labelsize=18)

ax5 = fig.add_subplot(245)
ax5.plot(RGe, EH5, '.')
ax5.plot(x, m5*x+b5, '-')
ax5.plot([3, 18], [3, 18], ls="--", c=".3")
ax5.set_ylim(3,18)
ax5.set_xlim(3,18)
ax5.text(3.1, 17.1, eq5, fontsize=18)
ax5.text(3.1, 15.9, Rsq5, fontsize=18)
ax5.set_xlabel('Rn - G $[MJ/m^2]$', fontsize=18)
ax5.set_ylabel('%s + H $[MJ/m^2]$' % n5, fontsize=18)
ax5.tick_params(axis='both', labelsize=18)

ax6 = fig.add_subplot(246)
ax6.plot(RGe, EH6, '.')
ax6.plot(x, m6*x+b6, '-')
ax6.plot([3, 18], [3, 18], ls="--", c=".3")
ax6.set_ylim(3,18)
ax6.set_xlim(3,18)
```

```
    ax6.text(3.1, 17.1, eq6, fontsize=18)
    ax6.text(3.1, 15.9, Rsq6, fontsize=18)
    ax6.set_xlabel('Rn - G $[MJ/m^2]$', fontsize=18)
    ax6.set_ylabel('%s + H $[MJ/m^2]$' % n6, fontsize=18)
    ax6.tick_params(axis='both', labelsize=18)

    ax7 = fig.add_subplot(247)
    ax7.plot(RGe, EH7, '.')
    ax7.plot(x, m7*x+b7, '-')
    ax7.plot([3, 18], [3, 18], ls="--", c=".3")
    ax7.set_ylim(3,18)
    ax7.set_xlim(3,18)
    ax7.text(3.1, 17.1, eq7, fontsize=18)
    ax7.text(3.1, 15.9, Rsq7, fontsize=18)
    ax7.set_xlabel('Rn - G $[MJ/m^2]$', fontsize=18)
    ax7.set_ylabel('%s + H $[MJ/m^2]$' % n7, fontsize=18)
    ax7.tick_params(axis='both', labelsize=18)

    ax8 = fig.add_subplot(248)
    ax8.plot(RGe, EH8, '.')
    ax8.plot(x, m8*x+b8, '-')
    ax8.plot([3, 18], [3, 18], ls="--", c=".3")
    ax8.set_ylim(3,18)
    ax8.set_xlim(3,18)
    ax8.text(3.1, 17.1, eq8, fontsize=18)
    ax8.text(3.1, 15.9, Rsq8, fontsize=18)
    ax8.set_xlabel('Rn - G $[MJ/m^2]$', fontsize=18)
    ax8.set_ylabel('%s + H $[MJ/m^2]$' % n8, fontsize=18)
    ax8.tick_params(axis='both', labelsize=18)

    fig.savefig(curdir+"//EB_reg%s.png" % num, bbox_inches='tight', dpi=300)
    plt.show()

def EBCvsALLplot(TW, Penman, FAOday, PT, PM, MK, BS, GG, BC, Turc, MS, JH, MB,\
                 HS, BREB, SJ, Mort, EBC, t):
    fig = plt.figure()
    ax1 = fig.add_subplot(411)
    ax1.plot_date(t, EBC, 'black', label='EBC')
    ax1.plot_date(t, TW, 'b-', label='Thornthwaite')
    ax1.plot_date(t, BC, 'r-', label='Blaney-Criddle')
    ax1.plot_date(t, MB, 'g-', label='McGuiness-Bordne')
    ax1.set_xticklabels([], visible=False)
    ax1.set_ylim(0,7)
    ax1.legend(loc='center left', bbox_to_anchor=(1, 0.5), fontsize=18)
    ax1.tick_params(axis='both', labelsize=18)

    ax2 = fig.add_subplot(412)
    ax2.plot_date(t, EBC, 'black', label='EBC')
    ax2.plot_date(t, Turc, 'b-', label='Turc')
    ax2.plot_date(t, HS, 'r-', label='Hargreaves-Samani')
    ax2.plot_date(t, MK, 'g-', label='Makkink')
    ax2.plot_date(t, PT, 'y-', label='Priesley-Taylor')
    ax2.plot_date(t, BREB, 'c-', label='BREB')
    ax2.set_xticklabels([], visible=False)
    ax2.set_ylim(0,7)
    ax2.legend(loc='center left', bbox_to_anchor=(1, 0.5), fontsize=18)
    ax2.tick_params(axis='both', labelsize=18)

    ax3 = fig.add_subplot(413)
    ax3.plot_date(t, EBC, 'black', label='EBC')
    ax3.plot_date(t, Penman, 'b-', label='Penman')
    ax3.plot_date(t, PM, 'r-', label='Penman-Monteith')
    ax3.plot_date(t, FAOday, 'g-', label='FAO PM')
    ax3.plot_date(t, MS, 'y-', label='Matt-Shuttleworth')
```

```
    ax3.set_xticklabels([], visible=False)
    ax3.set_ylim(0,7)
    ax3.legend(loc='center left', bbox_to_anchor=(1, 0.5), fontsize=18)
    ax3.tick_params(axis='both', labelsize=18)

    ax4 = fig.add_subplot(414)
    ax4.plot_date(t, EBC, 'black', label='EBC')
    ax4.plot_date(t, BS, 'b-', label='Brutsaert-Strickler')
    ax4.plot_date(t, SJ, 'r-', label='Szilagyi-Jozsa')
    ax4.plot_date(t, GG, 'g-', label='Granger-Gray')
    ax4.plot_date(t, Mort, 'y-', label='Morton CRAE')
    ax4.set_ylim(0,7)
    ax4.legend(loc='center left', bbox_to_anchor=(1, 0.5), fontsize=18)
    ax4.tick_params(axis='both', labelsize=18)
    plt.xticks(fontsize=18)

    fig.text(0.04, 0.5, 'Evaporation [$mm/day$]', va='center',\
            rotation='vertical', fontsize=20)
    fig.savefig(curdir+"//EBC_ALL.png", bbox_inches='tight', dpi=300)
    plt.show()

def rainplot(prec, temp, t):
    width = 0.8
    x = np.asarray(t)
    fig = plt.figure()
    ax1 = fig.add_subplot(111)
    plt.gca().xaxis.set_major_formatter(dt.DateFormatter('%d-%m-%y'))
    plt.gca().xaxis.set_major_locator(dt.DayLocator())
    plt.xticks(np.arange(min(x), max(x)+1, 5.0))
    plt.xticks(rotation=45, fontsize=18)

    ax1.bar(x, prec, width, label='Precipitation')
    ax1.set_ylabel('Rainfall [$mm$]', fontsize=18)
    legend = ax1.legend(loc='upper left', shadow=True)
    handles, labels = ax1.get_legend_handles_labels()
    ax1.legend(handles, labels, loc='upper left', fontsize=18)
    plt.tick_params(axis='both', labelsize=18)

    ax2 = ax1.twinx()
    ax2.step(x, temp, 'r-', where='post', label='Temperature')
    ax2.set_ylabel('Temperature [ºC$]', fontsize=18)
    ax2.set_ylim(15, 27)
    legend = ax2.legend(loc='upper right', shadow=True)
    handles, labels = ax2.get_legend_handles_labels()
    ax2.legend(handles, labels, loc='upper right', fontsize=18)
    plt.tick_params(axis='both', labelsize=18)

    fig.set_size_inches(18, 10)
    fig.savefig(curdir+"//Rainplot.png", bbox_inches='tight', dpi=300)
    plt.show()

def WHSplot(prec, st1, st2, lim, t):
    width = 0.8
    x = np.asarray(t)
    fig = plt.figure()
    ax1 = fig.add_subplot(121)
    plt.gca().xaxis.set_major_formatter(dt.DateFormatter('%d-%m-%y'))
    plt.gca().xaxis.set_major_locator(dt.DayLocator())
    plt.xticks(np.arange(min(x), max(x)+1, 5.0))
    plt.xticks(rotation=45)

    ax1.bar(x, prec, width, label='Precipitation')
    ax1.set_ylabel('Rainfall [$mm$]', fontsize=18)
    legend = ax1.legend(loc='upper left', shadow=True)
```

```
handles, labels = ax1.get_legend_handles_labels()
ax1.legend(handles, labels, loc='upper left', fontsize=18)
ax1.tick_params(axis='both', labelsize=18)

ax2 = ax1.twinx()
ax2.plot(x, st1, 'y', label='Water stored')
ax2.set_ylabel('Storage [$m^3$]', fontsize=18)
ax2.set_ylim(20, 50)
legend = ax2.legend(loc='upper left', shadow=True)
handles, labels = ax2.get_legend_handles_labels()
ax2.legend(handles, labels, loc='upper right', fontsize=18)
ax2.tick_params(axis='both', labelsize=18)

ax3 = fig.add_subplot(122)
plt.gca().xaxis.set_major_formatter(dt.DateFormatter('%d-%m-%y'))
plt.gca().xaxis.set_major_locator(dt.DayLocator())
plt.xticks(np.arange(min(x), max(x)+1, 5.0))
plt.xticks(rotation=45, fontsize=18)

ax3.bar(x, prec, width, label='Precipitation')
ax3.set_ylabel('Rainfall [$mm$]', fontsize=18)
legend = ax3.legend(loc='upper left', shadow=True)
handles, labels = ax3.get_legend_handles_labels()
ax3.legend(handles, labels, loc='upper left', fontsize=18)
ax3.tick_params(axis='both', labelsize=18)

ax4 = ax3.twinx()
ax4.plot(x, st2, 'y-', label='Water stored')
ax4.plot(x, lim, 'r-')
ax4.set_ylabel('Storage [$m^3$]', fontsize=18)
legend = ax4.legend(loc='upper left', shadow=True)
handles, labels = ax4.get_legend_handles_labels()
ax4.legend(handles, labels, loc='upper right', fontsize=18)
ax4.tick_params(axis='both', labelsize=18)

fig.savefig(curdir+"//WHS_op.png", bbox_inches='tight', dpi=300)
plt.show()
```

References

Aerts, J., Lasage, R., Beets, W., de Moel, H., Mutiso, G., Mutiso, S., & de Vries, A. (2007). Robustness of sand storage dams under climate change. *Vadose Zone Journal*, *6*(3), 572-580.

Alexandris, S., Stricevic, R., & Petkovic, S. (2008). Comparative analysis of reference evapotranspiration from the surface of rainfed grass in central Serbia, calculated by six empirical methods against the Penman-Monteith formula. *European Water*, *21*(22), 17-28.

Allen, R. G., & Pruitt, W. O. (1986). Rational use of the FAO Blaney-Criddle formula. *Journal of Irrigation and Drainage Engineering*, *112*(2), 139-155.

Allen, R. G., Pereira, L. S., Raes, D., & Smith, M. (1998). FAO Irrigation and drainage paper No. 56. *Rome: Food and Agriculture Organization of the United Nations*, *56*, 97-156.

Benesty, J., Chen, J., Huang, Y., & Cohen, I. (2009). Pearson correlation coefficient. In *Noise reduction in speech processing* (pp. 1-4). Springer Berlin Heidelberg.

Bouchet, R. J. (1963). Evapotranspiration réelle et potentielle, signification climatique. *IAHS Publ*, *62*, 134-142.

De Bruin, H. A. R. (1981). The determination of (reference crop) evapotranspiration from routine weather data. In *Proceedings of Technical Meeting* (Vol. 38, pp. 25-37).

Brutsaert, W., & Stricker, H. (1979). An advection-aridity approach to estimate actual regional evapotranspiration. *Water resources research*, *15*(2), 443-450.

Brutsaert, W. (2013). *Evaporation into the atmosphere: theory, history and applications* (Vol. 1). Springer Science & Business Media.

Cammeraat, E. L. (2004). Scale dependent thresholds in hydrological and erosion response of a semi-arid catchment in southeast Spain. *Agriculture, Ecosystems & Environment*, *104*(2), 317-332.

Condie, S. A., & Webster, I. T. (1997). The influence of wind stress, temperature, and humidity gradients on evaporation from reservoirs. *Water Resources Research*, *33*(12), 2813-2822.

Craig, I. P. (2006). Comparison of precise water depth measurements on agricultural storages with open water evaporation estimates. *Agricultural Water Management*, *85*(1), 193-200.

DeMeo, G. A., Laczniak, R. J., Boyd, R. A., Smith, J. L., & Nylund, W. E. (2003). *Estimated ground-water discharge by evapotranspiration from Death Valley, California, 1997-2001* (No. 2003-4254).

Dolman, A. J., Miralles, D. G., & Jeu, R. A. (2014). Fifty years since Monteith's 1965 seminal paper: the emergence of global ecohydrology. *Ecohydrology*, *7*(3), 897-902.

Doorenbos, J., Pruitt, W. O., & Aboukhaled, A. (1992). Crop water requirements.

Drexler, J. Z., Snyder, R. L., Spano, D., Paw, U., & Tha, K. (2004). A review of models and micrometeorological methods used to estimate wetland evapotranspiration. *Hydrological Processes*, *18*(11), 2071-2101.

Duan, Z., & Bastiaanssen, W. G. M. (2017). Evaluation of three energy balance-based evaporation models for estimating monthly evaporation for five lakes

using derived heat storage changes from a hysteresis model. *Environmental Research Letters*, *12*(2), 024005.

EMG, U. (2011). Global drylands: a UN system-wide response. *United Nations Environment Management Group*, *131*.

Ertsen, M., & Hut, R. (2009). Two waterfalls do not hear each other. Sand-storage dams, science and sustainable development in Kenya. *Physics and Chemistry of the Earth, Parts A/B/C*, *34*(1), 14-22.

Van de Giesen, N. C., Stomph, T. J., & De Ridder, N. (2000). Scale effects of Hortonian overland flow and rainfall–runoff dynamics in a West African catena landscape. *Hydrological Processes*, *14*(1), 165-175.

Granger, R. J., & Gray, D. M. (1989). Evaporation from natural nonsaturated surfaces. *Journal of Hydrology*, *111*(1-4), 21-29.

Guo, D., Westra, S., & Maier, H. R. (2016). An R package for modelling actual, potential and reference evapotranspiration. *Environmental Modelling & Software*, *78*, 216-224.

Hargreaves, G. H., & Samani, Z. A. (1985). Reference crop evapotranspiration from temperature. *Appl. Eng. Agric*, *1*(2), 96-99.

Hayashi, I. (1996). Five years experiment on vegetation recovery of drought deciduous woodland in Kitui, Kenya. *Journal of Arid Environments*, *34*(3), 351-361.

Huntington, J. L., Szilagyi, J., Tyler, S. W., & Pohll, G. M. (2011). Evaluating the complementary relationship for estimating evapotranspiration from arid shrublands. *Water Resources Research*, *47*(5).

Hut, R., Ertsen, M., Joeman, N., Vergeer, N., Winsemius, H., & van de Giesen, N. (2008). Effects of sand storage dams on groundwater levels with examples from Kenya. *Physics and Chemistry of the Earth, Parts A/B/C*, *33*(1), 56-66.

Inman-Bamber, N. G., & McGlinchey, M. G. (2003). Crop coefficients and water-use estimates for sugarcane based on long-term Bowen ratio energy balance measurements. *Field Crops Research*, *83*(2), 125-138.

IPCC (2012) IPCC special report on managing the risks of extreme events and disasters to advance climate change adaptation. http://ipcc-wg2gov/SREX/

Iqbal, M. (2012). *An introduction to solar radiation*. Elsevier.

Jarraud, M. (2008). Guide to meteorological instruments and methods of observation (WMO-No. 8). *World Meteorological Organisation: Geneva, Switzerland*.

Jensen, M. E., Burman, R. D., & Allen, R. G. (1990). Evapotranspiration and irrigation water requirements. ASCE.

Jiménez Cisneros, B.E., T. Oki, N.W. Arnell, G. Benito, J.G. Cogley, P. Döll, T. Jiang, and S.S. Mwakalila, 2014: Freshwater resources. In: Climate Change 2014: Impacts, Adaptation, and Vulnerability. Part A: Global and Sectoral Aspects. Contribution of Working Group II to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change [Field, C.B., V.R. Barros, D.J. Dokken, K.J. Mach, M.D. Mastrandrea, T.E. Bilir, M. Chatterjee, K.L. Ebi, Y.O. Estrada, R.C. Genova, B. Girma, E.S. Kissel, A.N. Levy, S. MacCracken, P.R. Mastrandrea, and L.L. White (eds.)]. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, pp. 229-269.

Kenya National Bureau of Statistics (KNBS) and Society for International Development (SID). (2013). Exploring Kenya's Inequality: Pulling Apart or Pooling Together. Nairobi: KNBS and SID.

Kenya National Bureau of Statistics (KNBS). (2015). Statistical Abstract 2015. Nairobi: KNBS.

Lasage, R., Aerts, J. C. J. H., Mutiso, G. C., & De Vries, A. (2008). Potential for community based adaptation to droughts: Sand dams in Kitui, Kenya. *Physics and Chemistry of the Earth, Parts A/B/C*, *33*(1), 67-73.

Lasage, R., Aerts, J. C. J. H., Verburg, P. H., & Sileshi, A. S. (2015). The role of small scale sand dams in securing water supply under climate change in Ethiopia. *Mitigation and adaptation strategies for global change*, *20*(2), 317.

Lasage, R., & Verburg, P. H. (2015). Evaluation of small scale water harvesting techniques for semi-arid environments. *Journal of Arid Environments*, *118*, 48-57.

Lawrence, M. G. (2005). The relationship between relative humidity and the dewpoint temperature in moist air: A simple conversion and applications. *Bulletin of the American Meteorological Society*, *86*(2), 225-233.

Legates, D. R., & McCabe, G. J. (1999). Evaluating the use of "goodness-of-fit" measures in hydrologic and hydroclimatic model validation. *Water resources research*, *35*(1), 233-241.

Li, X. Y., Xie, Z. K., & Yan, X. K. (2004). Runoff characteristics of artificial catchment materials for rainwater harvesting in the semiarid regions of China. *Agricultural Water Management*, *65*(3), 211-224.

Liu, S., Lu, L., Mao, D., & Jia, L. (2007). Evaluating parameterizations of aerodynamic resistance to heat transfer using field measurements. *Hydrology and earth system sciences*, *11*(2), 769-783.

Love, D., van der Zaag, P., Uhlenbrook, S., & Owen, R. J. S. (2011). A water balance modelling approach to optimising the use of water resources in ephemeral sand rivers. *River research and applications*, *27*(7), 908-925.

Makkink, G. F. (1957). Testing the Penman formula by means of lysimeters. *J. Inst. Water Eng*, *11*(3), 277-288.

McMahon, T. A., Peel, M. C., Lowe, L., Srikanthan, R., & McVicar, T. R. (2013). Estimating actual, potential, reference crop and pan evaporation using standard meteorological data: a pragmatic synthesis. *Hydrology and Earth System Sciences*, *17*(4), 1331.

Monteith, J. L., Szeicz, G., & Waggoner, P. E. (1965). The measurement and control of stomatal resistance in the field. *Journal of Applied Ecology*, 345-355.

Monteith, J. L. (1994, September). Fifty years of potential evaporation. In *Proc. Conf. Trinity College* (pp. 29-45).

Monteith, J., & Unsworth, M. (2007). *Principles of environmental physics*. Academic Press.

Morton, F. I. (1983). Operational estimates of areal evapotranspiration and their significance to the science and practice of hydrology. *Journal of Hydrology*, *66*(1-4), 1-76.

Munywoki, J. M., Kitema, M. I., Munguti, J. M., & Mutiso, S. (2004). Kitui Sand Dams: Construction and Operation [Project Documentation]. *Kitui, Kenya: SASOL Foundation*.

Nagler, P. L., Scott, R. L., Westenburg, C., Cleverly, J. R., Glenn, E. P., & Huete, A. R. (2005). Evapotranspiration on western US rivers estimated using the Enhanced Vegetation Index from MODIS and data from eddy covariance and Bowen ratio flux towers. *Remote sensing of environment*, *97*(3), 337-351.

Nash, J.E. & Sutcliffe, J. V. (1970). River flow forecasting through conceptual models. Part I—a discussion of principles. *Journal of Hydrology, 10 (1970)*, 282–290.

Nash, J. E. (1989). Potential evaporation and "the complementary relationship". *Journal of Hydrology*, *111*(1-4), 1-7.

Ngigi, S. N. (2003). What is the limit of up-scaling rainwater harvesting in a river basin?. *Physics and Chemistry of the Earth, Parts A/B/C*, *28*(20), 943-956.

Nicholson, S. E. (2014). A detailed look at the recent drought situation in the Greater Horn of Africa. *Journal of Arid Environments*, *103*, 71-79.

Nie, D., Kanemasu, E. T., Fritschen, L. J., Weaver, H. L., Smith, E. A., Verma, S. B., Field, R. T., Kustas, W. & Stewart, J. B. (1992). An intercomparison of surface energy flux measurement systems used during FIFE 1987. *Journal of Geophysical Research: Atmospheres*, *97*(D17), 18715-18724.

Olufayo, O. A., Otieno, F. A., & Ochieng, G. M. (2009). Run-off storage in sand reservoirs as an alternative source of water supply for rural and semi-arid areas of South Africa. In *WCSET conference proceedings. August* (pp. 24-26).

Oudin, L., Hervieu, F., Michel, C., Perrin, C., Andréassian, V., Anctil, F., & Loumagne, C. (2005). Which potential evapotranspiration input for a lumped rainfall–runoff model?: Part 2—Towards a simple and efficient potential evapotranspiration model for rainfall–runoff modelling. *Journal of hydrology*, *303*(1), 290-306.

Pauw, W. P., Mutiso, S., Mutiso, G., Manzi, H. K., Lasage, R., & Aerts, J. C. (2008). An assessment of the social and economic effects of the Kitui sand Dams community based adaptation to climate change. *Nairobi: SASOL and Institute for Environmental Studies*.

Penman, H. L. (1948, April). Natural evaporation from open water, bare soil and grass. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* (Vol. 193, No. 1032, pp. 120-145). The Royal Society.

Penman, H.L. (1956), Evaporation: An introductory survey. *Netherlands Journal of Agricultural Science, vol. 4*, 9-29.

Pereira, A. R., & Pruitt, W. O. (2004). Adaptation of the Thornthwaite scheme for estimating daily reference evapotranspiration. *Agricultural Water Management*, *66*(3), 251-257.

Priestley, C. H. B., & Taylor, R. J. (1972). On the assessment of surface heat flux and evaporation using large-scale parameters. *Monthly weather review*, *100*(2), 81-92.

Quilis, R. O., Hoogmoed, M., Ertsen, M., Foppen, J. W., Hut, R., & de Vries, A. (2009). Measuring and modeling hydrological processes of sand-storage

dams on different spatial scales. *Physics and Chemistry of the Earth, Parts A/B/C*, *34*(4), 289-298.

Rosenberry, D. O., Winter, T. C., Buso, D. C., & Likens, G. E. (2007). Comparison of 15 evaporation methods applied to a small mountain lake in the northeastern USA. *Journal of Hydrology*, *340*(3), 149-166.

Schaefli, B. and Gupta, H. V. (2007), Do Nash values have value?. Hydrol. Process., 21: 2075–2080.

Shuttleworth, W. J. (2006). Towards one-step estimation of crop water requirements. *Transactions of the ASABE*, *49*(4), 925-935.

Shuttleworth, W. J., & Wallace, J. S. (2009). Calculating the water requirements of irrigated crops in Australia using the Matt-Shuttleworth approach. *Transactions of the ASABE*, *52*(6), 1895-1906.

Szilagyi, J. (2007). On the inherent asymmetric nature of the complementary relationship of evaporation. *Geophysical Research Letters*, *34*(2).

Szilagyi, J., & Jozsa, J. (2008). New findings about the complementary relationship-based evaporation estimation methods. *Journal of Hydrology*, *354*(1), 171-186.

Tanaka, S., Sugimura, T., & Mishima, S. (2000). Monitoring of vegetation extent around Kitui pilot forest (afforestation test site) in Kenya with rainfall by satellite data. *Advances in Space Research*, *26*(7), 1039-1042.

Tanny, J., Cohen, S., Assouline, S., Lange, F., Grava, A., Berger, D., ... & Parlange, M. B. (2008). Evaporation from a small water reservoir: Direct measurements and estimates. *Journal of Hydrology*, *351*(1), 218-229.

Thornthwaite, C. W. (1948). An approach toward a rational classification of climate. *Geographical review*, *38*(1), 55-94.

Tiggeloven, T. (2015). Modelling the effectiveness of household scale water harvesting structures in semi arid areas. Unpublished.

Todd, R. W., Evett, S. R., & Howell, T. A. (2000). The Bowen ratio-energy balance method for estimating latent heat flux of irrigated alfalfa evaluated in a semi-arid, advective environment. *Agricultural and Forest Meteorology*, *103*(4), 335-348.

Turc, L. (1961). Estimation of irrigation water requirements, potential evapotranspiration: a simple climatic formula evolved up to date. *Ann Agron*, *12*(1), 13-49.

Vörösmarty, C. J., McIntyre, P. B., Gessner, M. O., Dudgeon, D., Prusevich, A., Green, P., ... & Davies, P. M. (2010). Global threats to human water security and river biodiversity. *Nature*, *467*(7315), 555-561.

Willmott, C. J. (1982). Some comments on the evaluation of model performance. *Bulletin of the American Meteorological Society*, *63*(11), 1309-1313.

Winter, T. C., Rosenberry, D. O., & Sturrock, A. M. (1995). Evaluation of 11 equations for determining evaporation for a small lake in the north central United States. *Water Resources Research*, *31*(4), 983-993.

Yamanaka, T., Kaihotsu, I., Oyunbaatar, D., & Ganbold, T. (2007). Summertime soil hydrological cycle and surface energy balance on the Mongolian steppe. *Journal of arid environments*, *69*(1), 65-79.