Snow Code Description

Thomas Knecht

September 14, 2017

1 Introduction

A better understanding of the snow cover is not only important for modeling the magnitude and timing of snow melt runoff but also for land surface climate feedbacks, such as the albedo. A general method to monitor the spatial and temporal evolution of the snow cover is with satellite imagery as well as sparsely located surface measurements (Lundquist 2008). However, Lundquist et al. (2008) describes a method, using ground surface temperature (GST) measurements to determine the duration of snow cover. Schmid et al. (2012) uses as well ground surface temperatures to detect the melt-out date (MD) and the basal-ripening date (RD). The MD is described as the time, when the snow cover is molten and therefore, no more release of meltwater occurs which allows the ground surface to warm above 0°C (Schmid et al. 2012). The RD describes the time when a frozen ground surface is warmed to 0°C by melt-water percolation or by strong rain-on-snow events, which implies an isothermal regime of the snow cover and thus a melting of the snow cover (Schmid et al. 2012). The method, described by Schmid et al. (2012) is used to analyze ground surface temperatures from 47 temperature loggers located along the Ingraham Trail in the Yellowknife region and 172 temperature loggers located in the Lac de Gras Region (Northwest Territories) in order to get a better understanding of the snow cover duration as well as the snow melt period. The original code for the analysis is fitted to the data of Schmid et al. (2012) and therefore, it needed to be adjusted in order to apply it on our data.

The RD is a good indicator for the time, when the GST enters the zero curtain. However, it does not show the start of the warming induced by melt water percolation. Therefore, a new method is introduced to detect major warming periods within a snow cover period.

2 Method

The snow-code consists of six major functions and several minor functions. The major functions are used to import the data from the database, to calculate the snow cover period, to get the Ripening date, to find the zero curtains, to detect major warming periods and to create plots for the locations containing all the important information. Each function, except the import- and the plot-function, has a csv-file as output. In the following subsections, each function is explained in more detail.

2.1 Import data: f.load_data()

To import the data from the database into R, the f.load_data-function is used. This function creates a list containing data-frames of daily aggregated data of all the wanted locations.

2.1.1 Description

f.load_data-function:

This function uses a lapply to run the f.aggregate-function over a list of all wanted location names in order to import the data into R. Therefore, a csv-file containing the location names is imported and the wished time period is inserted. The output is a list, containing the data-frames of the locations.

f.aggregate-function:

This is the actual function, that imports and aggregates the data from the database. The aggregation is already part of the query that is used to import the data. With the query, the following parameters are imported:

location name **as** loc_name

observation height in meters **as** height

mean daily temperature **as** agg_avg

number of observations per day **as** agg_cnt

minimum daily temperature **as** agg_min

maximum daily temperature **as** agg_max

daily standard deviation **as** agg_sd

date and time **as** time

A subset of the time is made in order to only get the year, month and day. Furthermore, the time column needs to be in POSIXct and the following date-format has to be used: %yyyy%mm-%dd

2.1.2 Input-arguments

con: This is the connection to the permafrost database. The connection can be made with the database function: con ← dbpf_con()

inventory: A csv-file containing all the needed location names

year_start: The start year

year_end: The end year. As the year starts on the 01.01 at 00:00 and ends on the 01.01 at 00:00, the end year has to be at least one year higher than the start year.

2.1.3 Usage and example

Usage:

f.load_data(con, inventory, year_start, year_end)

Example:

Assuming we want the data from Yellowknife for the winter 2016/2017:

YKN_loggers ← f.load_data(con, ' /Documents/YKN_inventory.csv',2017,2018)

2.1.4 What could be added or changed

The latitude and longitude could be added to the query. This would help, that no extra function is needed to get the location information for each logger.

The question is, if instead of the calendar year, the hydrological year should be used to select the data. Using the hydrological year, only one winter would be shown by choosing one year, whereas with the calendar year always two years need to be selected in order to get a full winter.

2.2 Snow cover by Schmid et al. (2012): f.snow.cover.marcol()

The snow cover function consists of two major functions: f.snow.cover.marcol(con, obs_stat, out.path) and f.snow.cover1(obs_stat, con), whereof the f.snow.cover1-function does the actual calculation and the f.snow.cover.marcol-function is used to apply the f.snow.cover1-function on a list of data-frames. The code for the f.snow.cover1-function is based on the function by Schmid et al (2012) but with certain adjustments in order to apply it on our data. The threshold parameters stay the same as in the original function.

2.2.1 Description

f.snow.cover.marcol-function:

In this function, a lapply-function is used to run the f.snow.cover1-function over a list of data-frames. Furthermore, it rbinds the results into a single data-frame and exports it as a csv-file to a directory. The output is a csv-file containing the snow period for each location.

f.snow.cover1-function:

The function is based on a snow- and a temperature-index. The snow-index is based on the daily standard deviation (SD) of the daily averaged temperature. For days with a negative maximum daily temperature, a SD-threshold of 0.3 is chosen as an indicator of snow (further details can be found in Schmid et al (2012)). For positive maximum daily temperatures, a SD of 0.1 is applied as threshold. Furthermore, all days with a maximum daily temperature higher then 3°C are assumed to be snowless.

The temperature-index selects all days with a maximum daily temperature smaller than 0.5°C and is used to fill smaller gaps in the snow-index. It is assumed, that a maximum daily temperature of 0.5°C is at least needed to start the snowmelt.

The longest snow period is selected in order to retrieve the start and the melt out date.

To test the reliability of the snow period, the MDr-value is used. To calculate MDr a mean of the mean daily standard deviation (sd.day) is calculated for the time period January to March. For this period, Schmid et al (2012) retrieved a minimum mean standard deviation of 0.2 (MDr.sd) as indicator for snow. Thus:

$MDr = MDr.sd - sd.day$

If MDr is negative, it is assumed that no snow is present, whereas a positive MDr implies a snow cover.

In the end, an output table is created, containing the location name, latitude, longitude, mean annual ground surface temperature (MAGST), Snow.Start, Snow.End and MDr.

2.2.2 Input-arguments

The f.snow.cover.marcol-function has three input arguments: con, obs_stat, out.path

con: This is the connection to the permafrost database. It is needed to get the latitude and longitude for each location. The connection can be made with the database function: con ← dbpf_con()

obs_stat: A list of data-frames containing the observations. It needs to contain the following parameters with the according column names: location name **as** loc_name, mean daily temperature **as** agg_avg, minimum daily temperature **as** agg_min, maximum daily temperature **as** agg_max, daily standard deviation **as** agg_sd, date **as** time. The time needs to be in POSIXcT and the following format has to be used: %yyyy%mm-%dd

out.path: path to directory

2.2.3 Usage and example

Usage:

f.snow.cover.marcol(con, obs_stat, out.path)

Example:

Assuming we have a list containing all the Yellowknife observations called YKN_loggers:

f.snow.cover.marcol(con, YKN_loggers, ' /Documents/YKN_data/YKN_snowcover')

2.2.4 What could be added or changed

This function allows only to select the largest snow cover period. It would be nice to have a code, that can detect several snow cover periods. It is not very difficult to do this, but new selection criteria need to be introduced. Afterwards, the other functions could be applied on each single snow cover period.

2.3 Ripening date by Schmid et al. (2012): f.RD.marcol()

To detect the Ripening date, the f.RD.marcol-function is used. This function uses the f.RD.marcol1-function which does the actual calculation of the ripening date for a certain location. The code for the f.RD.marcol1-function is an adjusted version of the original code by Schmid et al (2012).

2.3.1 Description

f.RD.marcol-function:

In this function, a lapply-function is used to run the f.RD.marcol1-function over a list of data-frames. Furthermore, it rbinds the results into a single data-frame and exports it as a csv-file to a directory. The output is a csv-file containing the RD for each location.

f.RD.marcol1-function:

This function is based on the snow period of the location, a zero curtain index as well as a freeze index.

For the snow period index, the start and end date of the snow cover is used to mark all days with snow.

The zero curtain index consists of all days where the minimum and maximum daily temperatures are within a $\pm 0.25°C$ boundary.

For the freeze index, all days with a mean daily temperature smaller than -0.25°C are selected. Then, the date of the last day (freez.end) which fulfills this criteria is selected.

First, the snow period and the zero curtain indices are combined. This gives all the zero curtain days which guaranteed have as well snow cover. In a second step, all days before the freez.end are set to zero in order to just select the last zero curtain period of a winter.

The start date of the remaining zero curtain period is now selected as the Ripening Date.

To check the credibility of the Ripening Date a freezing index value (FI) is used. The FI is the accumulated temperature of all negative temperatures during a winter. It is assumed, that if the FI is larger than -50, the RD is reliable. Otherwise, the temperatures were too warm to reliably detect the RD.

In the end, an output table is created, containing the location name, latitude, longitude, mean annual ground surface temperature (MAGST), Snow.Start, Snow.End and RD.

2.3.2 Input-arguments

The f.RD.marcol-function has three input arguments: con, obs_stat, out.path

con: This is the connection to the permafrost database. It is needed to get the latitude and longitude for each location. The connection can be made with the database function: con ← dbpf_con()

obs_stat: A list of data-frames containing the observations. It needs to contain the following parameters with the according column names: location name **as** loc_name, mean daily temperature **as** agg_avg, minimum daily temperature **as** agg_min, maximum daily temperature **as** agg_max, daily standard deviation **as** agg_sd, date **as** time. The time needs to be in POSIXcT and the following format has to be used: %yyyy%mm-%dd

out.path: path to directory

2.3.3 Usage and example

Usage:

f.RD.marcol(con, obs_stat, out.path)

Example:

Assuming we have a list containing all the Yellowknife observations called YKN_loggers.

f.RD.marcol(con, YKN_loggers, ' /Documents/YKN_data/YKN_RDmarol')

2.3.4 What could be added or changed

2.4 Zero curtain periods: f.zero.curtain()

The zero curtain is defined as "the persistence of a nearly constant temperature, very close to the freezing point, during annual freezing (and occasionally during thawing) of the active layer" (NSIDC, 2017). To detect periods with nearly constant temperatures, the standard deviation of the mean daily temperature as well as the temperature deviation between two following days are chosen as indicators. The "close to the freezing point" criteria is only taken into account for the freezing period but not as strictly for the thawing periods.

To detect these zero curtain periods, the f.zero.curtain-function is used which uses the f.zero.curtain1-function. Here again, the f.zero.curtain-function is the main function that runs over a list of locations whereas the f.zero.curtain1-function does the actual calculation of the zero curtain periods for each location.

2.4.1 Description

f.zero.curtain-function:

In this function, a lapply-function is used to run the f.zero.curtain1-function over a list of data-frames. Furthermore, it rbinds the results into a single data-frame and exports it as a csv-file to a directory. The output is a csv-file containing all the zero curtain periods (can be more than one) for the locations.

f.zero.curtain1-function:

This function uses a slope- as well as a daily standard deviation- index to determine the zero curtain periods.

The slope index consists of the mean daily temperature difference between two following days. Therefore, the slope is calculated and assigned to the i+1 day.

A subset is made, using only the days with a maximum daily temperature smaller than 0.5°C. The 0.5°C is chosen as it takes the measurement uncertainty into account.

In order to detect periods with a nearly constant temperature, a threshold for the slope is defined, using the standard deviation of the slope (SDslope). At the moment the threshold is defined as 0.1∗SDslope. This results in the selection of only very flat parts.

To detect days with a nearly constant temperature, the standard deviation of the mean daily temperature (SDtemp) is used as an indicator. Therefore, a threshold for the SDtemp is defined which is currently: 0.01∗SDtemp.

Only days that fulfill both criteria are selected.

To take the "very close to the freezing point"-criteria of the zero curtain definition into account, a temperature threshold is defined. In order to distinguish freezing periods from warming periods the following requirements are applied:

Periods with a net decreasing slope are assumed to be freezing periods and therefore, the mean temperature of the period needs to be within 0.2°C of 0°C.

Periods with a net increasing slope are assumed to be thawing periods and therefore, the end of the period needs to reach at least -0.2°C.

As a last selection criteria, the zero curtain period length is taken into account. Currently, zero curtain periods smaller than 2 days are neglected.

In the end, an output table is created, containing for all zero curtain periods of a location the location name, latitude, longitude, mean annual ground surface temperature (MAGST), Snow.Start, Snow.End, RD.marcol, Zero.Curtain.Start, Zero.Curtain.End, the

Median Temperature as well as the standard deviation of the temperature of each zero curtain period.

2.4.2 Input-arguments

The f.zero.curtain-function has three input arguments: con, obs_stat, out.path

con: This is the connection to the permafrost database. It is needed to get the latitude and longitude for each location. The connection can be made with the database function: con ← dbpf_con()

obs_stat: A list of data-frames containing the observations. It needs to contain the following parameters with the according column names: location name **as** loc_name, mean daily temperature **as** agg_avg, minimum daily temperature **as** agg_min, maximum daily temperature **as** agg_max, daily standard deviation **as** agg_sd, date **as** time. The time needs to be in POSIXcT and the following format has to be used: %yyyy%mm-%dd

out.path: path to directory

2.4.3 Usage and example

Usage:

f.zero.curtain(con, obs_stat, out.path)

Example:

Assuming we have a list containing all the Yellowknife observations called YKN_loggers.

f.zero.curtain(con, YKN_loggers, ' /Documents/YKN_data/YKN_zerocurtains')

2.4.4 What could be added or changed

The thresholds need to be scientifically better explained. Furthermore, the maximum-temperature-threshold could be adjusted so that the freezing periods need to be even closer to 0°C. To exclude warmer temperatures, not a 0.5°C should be used but the actual snow cover duration. This would not be too difficult to change.

2.5 Warming periods: f.warming.periods()

To detect warming periods due to possible meltwater percolation, the f.warming.periods can be used. It uses the f.warming.periods1-function which does the actual calculation

of warming periods. The detection of the warming periods is based on the slope of the maximum daily temperature.

2.5.1 Description

f.warming.periods-function:

In this function, a lapply-function is used to run the f.warming.periods1-function over a list of data-frames. Furthermore, it rbinds the results into a single data-frame and exports it as a csv-file to a directory. The output is a csv-file containing all the warming period start dates (can be more than one) for the locations.

f.warming.periods1-function:

As a first step, the slope of the maximum daily temperature is calculated.

In a next step, a subset is made, using only the days with a maximum daily temperature smaller than 0.5°C. The 0.5°C is chosen as it takes the measurement uncertainty into account.

To detect the major warming periods during a winter, a threshold is defined. Therefore, for each location a histogram, the density as well as the 1∗SD, 2∗SD and 3∗SD were plotted. After examining all the plots, the 3∗SD seems to be a good threshold to detect outliers. This threshold is used to make a slope1-index, in order to select all outliers.

After setting all negative slopes to zero, subsets of all warming periods containing an outlier are made.

As some of these warming periods last for a longer time, a function is applied to only select the steepest parts of a certain warming period. Therefore, a threshold of 0.75∗SD-of-Slope of each warming period is defined and applied.

The warming periods need to reach a certain temperature in order to be recognized as major warming period. Therefore, a threshold depending on the winter coldness is defined. Thus, the 10% quantile of the mean daily winter temperatures is used as an indicator. The warming periods need to reach at least 40% of the 10% quantile to be selected.

Furthermore, a function is applied to filter out all the warming periods that start in positive temperatures.

To define the warming period, the start date is selected. Therefore, the date of the day before the actual start of the steepest part is selected. This is due to the fact, that the slope is always assigned to the i+1 date and thus, the actual warming starts already a day before. In addition to the start date, the accumulated slope of the steepest part of the warming period (Temp.Diff), the mean slope of the warming period (Mean.Slope) as well

as the deviation form the end of the warming period to 0°C (Temp.To.Zero) are calculated in order to describe the character of the warming period.

Another function selects the last warming period as the "TRUE" warming period before the snowmelt. Therefore, a new column is created and filled with either an F for all none-"TRUE" warming periods or a T for all "TRUE" warming periods.

In the end, an output table is created, containing for all warming periods of a location the location name, latitude, longitude, mean annual ground surface temperature (MAGST), Snow.Start, Snow.End, RD.marcol, warming.period.start, RD, Temp.Diff, Mean.Slope as well as the Temp.To.Zero.

2.5.2 Input-arguments

The f.warming.periods-function has three input arguments: con, obs_stat, out.path

con: This is the connection to the permafrost database. It is needed to get the latitude and longitude for each location. The connection can be made with the database function: con ← dbpf_con()

obs_stat: A list of data-frames containing the observations. It needs to contain the following parameters with the according column names: location name **as** loc_name, mean daily temperature **as** agg_avg, minimum daily temperature **as** agg_min, maximum daily temperature **as** agg_max, daily standard deviation **as** agg_sd, date **as** time. The time needs to be in POSIXcT and the following format has to be used: %yyyy%mm-%dd

out.path: path to directory

2.5.3 Usage and example

Usage:

f.warming.periods(con, obs_stat, out.path)

Example:

Assuming we have a list containing all the Yellowknife observations called YKN_loggers.

f.warming.periods(con, YKN_loggers, ' /Documents/YKN_data/YKN_warmingstart')

2.5.4 What could be added or changed

The selection of the steepest part of a warming period is a bit random. Is there a better way? The code has in general to be adjusted and optimized. The selection of the "TRUE" warming period needs as well some adjustment as by now, always the last warming period is assumed to be the "TRUE"-one. To exclude warmer temperatures, not a 0.5°C should be used but the actual snow cover duration. This would not be too difficult to change.

2.6 Plot function: f.plot()

The f.plot-function uses the f.single.plot-function to create plots for each location.

2.6.1 Description

f.plot-function:

In this function, a lapply-function is used to run the f.single.plot-function over a list of data-frames in order to create plots for each location.

f.single.plot-function:

The f.single.plot-function uses the f.snow.cover1-, f.RD.marcol1-, f.warming.periods1- and f.zero.curtain1-functions to calculate all the needed data for the plots.

To create the plot, the ggplot function is used.

As output, a plot is created containing the mean daily temperature, the maximum daily temperature, the minimum daily temperature, the snow cover period, the RD, the warming period start dates as well as all the zero curtain periods.

2.6.2 Input-arguments

The f.plot-function has three input arguments: con, obs_stat, out.path

con: This is the connection to the permafrost database. It is needed to get the latitude and longitude for each location. The connection can be made with the database function: con ← dbpf_con()

obs_stat: A list of data-frames containing the observations. It needs to contain the following parameters with the according column names: location name **as** loc_name, mean daily temperature **as** agg_avg, minimum daily temperature **as** agg_min, maximum daily

temperature **as** agg_max, daily standard deviation **as** agg_sd, date **as** time. The time needs to be in POSIXcT and the following format has to be used: %yyyy%mm-%dd

out.path: path to directory

2.6.3 Usage and example

Usage:

f.plot(con, obs_stat, out.path)

Example:

Assumimg we have a list containing all the Yellowknife observations called YKN_loggers.

f.plot(con, YKN_loggers, ' /Documents/YKN_data/YKN_plots/')

2.6.4 What could be added or changed

A option to choose what should be plotted would be nice. Not too difficult to add.

References

[1] M.-O. Schmid, S. Gubler, J. Fiddes, and S. Gruber. *Inferring snowpack ripening and melt-out from distributed measurements of near-surface ground temperatures.* The Cryosphere, 6, 1127-1139, 2012.

[2] Lundquist, Jessica D. and Lott, Fred. *Using inexpensive temperature sensors to monitor the duration and heterogeneity of snow-covered areas.* Water Resources Research, 44 (4), 2008.

[3] NSIDC. *https://nsidc.org/cryosphere/glossary/term/zero-curtain.* Access: 17:10 13/09/2017.