Norwegian University
of Life Sciences

Masters Thesis 2024 30 ECTS
Faculty of Science and Technology

A comparative study of soil temperature models, including machine learning models

Mats Hoem Olsen
Master of Science in Data Science

Forword

I would like to thank my advisors and friends. Also the Big Bang for happening.

Glossary

D | H | K | L | M | R | S

D

DataFrame
A table of values. The name is from the python library Pandas used in this study.. 8

H

Hashmap
A list of items where their unique placmnt in the list is detemend by their unique refrence key using a function that maps the key to a placement in the list.. 8

K

Kilden
Norwegian Institute of Bioeconomy Research Kilden. 6

L

LMT
Norwegian Institute of Bioeconomy Research LandbruksMeteorologisk service. 6, 9

Long Short Term-Memory
A Recurent Neural Network with a memory cell to distribute information along the other RNN cells.. 5

LSTM
Long Short Term-Memory. 5

M

MET
The Norwegian Meteorological Institute. 6, 9

MSTL
Multiple Seasonal-Trend decomposition using LOESS. 9

Multiple Seasonal-Trend decomposition using LOESS
Based in the traditional Seasonal-Trend decomposition using LOESS it decomposes a time-series into several seasons, trend, and residual[16].. 9, III

R

Recurent Neural Network
A Neural network that passes information between cells in the same layers.. 5

S

Seasonal-Trend decomposition using LOESS

Takes a timeseries and decomposing it into a trend component, season component, and residual component using local regression for smoothing[15].. 9, III

STL

Seasonal-Trend decomposition using LOESS. 9

Contents

1 Introduction

In agriculture soil temperature is one of the important parameters to put into consideration when thinking about pest prevention, conservation, and yield prediction. The reasoning for this is that knowing the soil temperature is knowing climate change [1], water management [2], yield [3], nitrogen processes [4] in the soil, calculation of plant-growth [5], when seeds start to sprout [5], potential flooding and erosions[6], and predicting when insect eggs hatch that were laid last winter. Being able to predict the soil temperature into the future will be a huge advantage for farmers, civilians, and scientists.

If it's important, why don't institutions measure it everywhere? There are several reasons for this, but a common reason is that it's expensive to install new equipment on old weather stations. Sometimes the weather station do have the sensors in the fields reading soil temperature at given levels, but due to technical misadventures and unforeseen phenomenons there might be gaps or misreadings that need to be replaced with approximations or NULL values[1]. There are algorithms, models, and statistical tools to interpolate these missing values but they have their drawbacks. For instance approximation by global mean, which is a common method used in timeseries[7]. This method is preserved global statistics, however does not represent local changes. Further more for a good estimation of soil temperature it is useful to include exogenous[2] features.

There has been done research into heat conductivity in soil that has lead to differential equations[8], however these equations[8, 9] are computationally expensive and difficult to simulate, or calculate[4]. To add to the complexity the heat dynamics change depending on soil temperature

In this study 4 methods will be compared and evaluated for the sake of further research into interpolation of missing data in northic countries based on as few features as possible. This study has chosen 2 types of models; Analytical, and Data-Driven models. There will also be base models to compare against, one for each model type.

2 Norwegian introduction

I landbruket er jordtemperatur en av de viktige parametrene å ta i betraktning når man tenker på skadedyrforebygging, bevaring, og avlingsprediksjon. Begrunnelsen for dette er at å kjenne til jordtemperaturen er å kjenne til klimaendringer [1], vannforvaltning [2], utbytte [3], nitrogenprosesser [4], potensielle overfloder of skred[6], plantevekst [5], når frø begynner å spire [5], og forutsi når insektegg klekkes som ble lagt sist vinter. Å kunne forutsi jordtemperaturen inn i fremtiden vil være en stor fordel for bønder, og forskere.

Hvis det er viktig, hvorfor måler ikke institusjoner det overalt? Det er flere årsaker til dette, men en vanlig årsak er at det er dyrt å installere nytt utstyr på gamle værstasjoner. Noen ganger har værstasjonen sensorene i feltene som leser jordtemperatur på gitte nivåer, men på grunn av tekniske feil eller uforutsette fenomener kan det være hull eller feilavlesninger som må erstattes med tilnærminger eller NULL-verdier[3]. Det finnes algoritmer, modeller og statistiske verktøy for å interpolere disse manglende verdiene, men de har sine ulemper. For eksempel tilnærming ved global gjennomsnitt, som er en vanlig metode som brukes i tidsserier[7]. Denne metoden er bevart global statistikk, men representerer ikke lokale endringer. Ytterligere mer for en god estimering av jordtemperatur er det nyttig å inkludere eksogene[4] variabler.

Det har vært gjort forskning på varmeledningsevne i jord som har ført til differensialligninger[8], men disse ligningene[8, 9] er dyre og vanskelige å simulere eller beregne[4]. Videre på grunn av

[1]These values are different from 0 as they represent "no data" and can't be used to do calculations.
[2]Variable that can affect the model, but is not not directly described by the model.
[3]Disse verdiene er forskjellige fra 0 siden de representerer "ingen data" og ikke kan brukes til å gjøre beregninger.
[4]Variabel som kan påvirke modellen, men som ikke er direkte beskrevet av modellen.

arten av andre partielle derivater ville den numeriske ustabiliteten være for stor for praktiske midler.

I denne studien vil 4 metoder bli sammenlignet og evaluert for videre forskning på interpolering av manglende data i nordlige land basert på så få funksjoner som mulig. Denne studien har valgt 2 typer modeller; Analytiske og datadrevne modeller. Det vil også være basismodeller å sammenligne mot, en for hver modelltype.

3 Previous works

This section discusses the theory behind the models used in the

3.1 Linear regression

The regression model will be for the sake of convenience be expressed as the following expression

$$\left(\vec{F} \circ \mathbf{A}\right) \vec{\beta} = \vec{y} + \vec{\varepsilon}$$

Where $\vec{F}$ is a vector function with following domain $\vec{F} : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times p}$ where $m, n, p \in \mathbb{N}$, $\mathbf{A}$ is the data in matrix form with dimensions $\mathbb{R}^{m \times n}$, $\vec{\beta}$ is the regression terms, $\vec{y}$ is the target (TJM), and $\vec{\varepsilon}$ is the error from modeling. The $\circ$ operator is the composition of $\vec{F}$ and $\mathbf{A}$, is a short way of writing $\vec{F}(\mathbf{A})$.

The $\vec{F}$ is not important, just that your data is shaped by a function.

This basic model to express the linearity of the components to soil temperature. This will function as the base model for regression models.

3.2 Plauborg linear regression model with Fourier terms

Making a linear regression model for soil temperature sensitive to time without introducing more computational heavy operation would be to introduce features that reflect time. In the paper "Simple model for 10 cm soil temperature in different soils with short grass" the author chose to extend the features from air temperature to include also day of year and the air temperature from those days. This means the following F function that Plauborg used would be

$$\vec{F} := [air_t, air_{t-1}, air_{t-2}, air_{t-3}, \sin(\omega t), \cos(\omega t), \sin(2 * \omega t), \cos(2 * \omega t)]^T$$

Where air_t is the air temperature at time t expressed in day of year, ω is the angular frequency to make the argument of sine and cosine expressed in radians. The sine/cosine elements in the F function represent the variations through the day by fitting $\vec{\beta}$ to the yearly variation. To adapt the authors model to an hourly time unit would be to either

1. Extend the F function to include a larger ω coefficient to reflect hourly oscillations in conjunction with daily fluxiations

2. Refit the Fourier terms with a larger ω coefficient to make the oscillations more representative of daily temperature changes.

The larger coefficient could be expressed as $\pi/12$ while the smaller ω for daily values would be rescaled to $\pi/4380$.

The problem with this approsh would be Fouriers Sine-Cosine series approximation which would suggest that Plauborg's method could be subject to overfitting with addition of more terms. On the other hand it gives us a way to compute the coefficients α_i and γ_i for sine and cosine terms respectively, though it would be more numerically stable with a pseudo-inverse computation or a max log likelihood approach. Need to compute condition number of solutions.

3.3 Rankin's finite difference method of simplified heat flow in snow covered soil

A more direct method based on laws of physics develop by Karvonen involves forming a Finite Difference Method (FDM) around point of interest with simplifications to the equations described in *A model for predicting the effect of drainage on soil moisture, soil temperature and crop yield.* A team of researchers collaborating with the original author found an algorithm by making simplifications to the general differential equations forming a iterative 2-step procedure seen at the procedure 1.

Algorithm 1: Rankin algorithm

Data: D, f_d
Result: T_Z
$\alpha_t \leftarrow \frac{\partial T/\partial t}{\partial^2 T/\partial z^2}$;
for $t \in T$ **do**
  $T_*^{t+1} \leftarrow T_Z^t + \Delta t \times \frac{\alpha_t}{(2Z)^2} \times (T_{air}^t - T_Z^t)$;
  $T_Z^{t+1} \leftarrow T_*^{t+1} * e^{-f_d \times D}$;
end for

Where $\alpha_t = K_T/C_A$ is the Thermal diffusivity from Fourier's law in thermodynamics, K_T is average soil thermal conductivity, C_A is the apparent heat capacity, and f_d is the damping parameter that has to be empirically derived however for this study it will be estimated from the data through the following estimation

$$f_d \approx \frac{-\ln\left(\frac{T_Z^{t+1}}{T_Z^t + \Delta t \frac{\alpha_t}{(2Z)^2}(T_{air}^t - T_Z^t)}\right)}{2D}$$

The approximation used in the algorithmn 1 assumes that K_T is not dependend on depth . To make the approximation of α_t more accurate the inclusion of rain (θ) to introduce variation can be approximated with

$$\alpha_t \approx \frac{b_1 + b_2\theta + b_3\sqrt{\theta}}{a_1 + a_2\theta}$$

proposed by Kodešová *et al.*[11][5]. To make the computation easier of this Padé-Puiseux[6] approximation hybrid we will realize that α_t is expressed by

$$\frac{b_1 + b_2\theta + b_3\sqrt{\theta}}{a_1 + a_2\theta} \approx \alpha_t \approx \frac{(T_z^{t+1} - T_{air}) * (2z)^2}{(T_{air} - T_z^t) * \Delta t}$$

Thereby only needing a linear regression of two F-functions; $F_1 = [1, \theta, \sqrt{\theta}]^T$ and $F_2 = [1, \theta]^T$ rather than a three step approximation. This algorithm (algorithm 1) will approximate the following integral

$$T = \int_{t_0}^{t_{max}} \frac{K_T}{C_A} \frac{\partial^2 T}{\partial z^2} dt$$

via a Finite Difference Method, although other methods are possible with higher accuracy[7]. Must verify for this case! This study will use the FDM used by the author for the purpose

[5] This representation was not proposed by the author however the linear approximations was proposed to approximate K_T and C_A respectfully. Since $\theta \propto m_w$ we can substitute water content with rain in mm since the area is constant and during all messurement the soil type will be the same, however this would need to be resestimated if a station contains a different soil type as the constant has a wide range of values[11].

[6] Padé Approximation is a of the form $\frac{\sum_{i=0}^{\infty} c_i x^i}{\sum_{j=0}^{\infty} c_j x^j}$ and a Puiseux series is a $\sum_{j=N}^{\infty} c_j x^{j/N}$

[7] For example fourth degree Runge-Kutta method[12] which converges quicker than forward-Euler method or FDM.

of making the results in this study comparable with the study presented in the paper "A simple model for predicting soil temperature in snow-covered and seasonally frozen soil."

For inital values this study are utelizing 2 methods under different assumtions:

$$T_z^0 \approx \frac{k\exp(D)}{1+\exp(D)\times(k-1)} \times T_{air}$$

Where k is $K_T * \Delta t/(C_A * (2Z)^2)$, and D is $-f_d * Snow_{Depth}$. This assumes constant air temperature above a constant layer of snow, though unrealistic since air temperature has a tendensy to change during the day due to solar radiation and other climate factors that can cool down or heat up the air. Another problem is the fact that the snow level ramins the same which is also untrue.

3.4 Long Short Term Memory model

The most common problem in Neural networks is the vanishing gradient problem where updating the first few layers of a large network becomes exponentially more difficult since the adjustments gets smaller and smaller for each layer towards the start rather than the reverse. Long Short Term-Memory changes this by caring information from the previous cells forward thereby allowing updating earlier cells with bigger impact than the standard approach[13]. LSTM is part of a family of Recurent Neural Network's that passes information to other cells in the same layer.

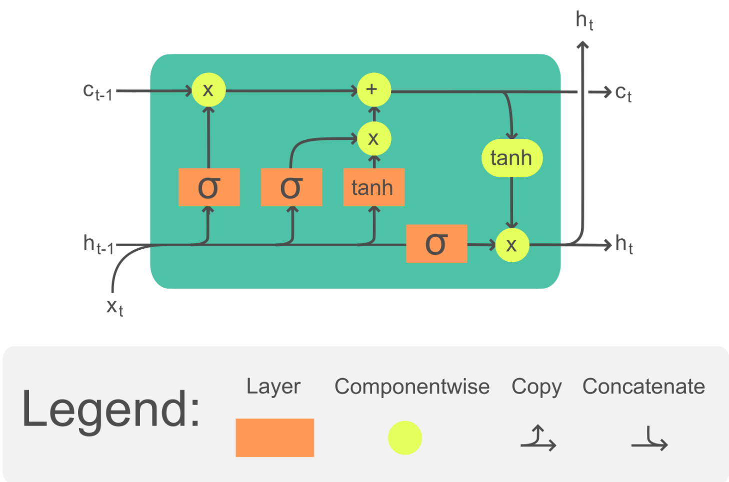

Figure 1: LSTM cell Artist: Chevalier [14]

3.5 Attention aware LSTM model

4 Method

4.1 Source of data

For this comparative study the following data sources will be used

1. Norwegian Institute of Bioeconomy Research LandbruksMeteorologisk service (LMT)
2. Xgeo
3. Norwegian Institute of Bioeconomy Research Kilden (Kilden)
4. The Norwegian Meteorological Institute (MET)

4.2 Dataset

The dataset is chosen from four regions in Norway; Innlandet, Vestfold, Trøndelag, and Østfold. From each region are four stations picked:

Innlandet
1. Kise
2. Ilseng
3. Apelsvoll
4. Gausdal

Trøndelag
1. Kvithamar
2. Rissa
3. Frosta
4. Mære

Østfold
1. Rygge
2. Rakkestad
3. Tomb
4. Øsaker

Vestfold
1. Lier
2. Ramnes
3. Tjølling
4. Sande

All stations are sampled from the date[8] 03-01 to 10-31 from 2016 to 2020. The features rain (RR), mean soil temperature at 10cm (TJM10), mean soil temperature at 20cm (TJM20), and air temperature at 2m (TM) are sampled from the LMT database. The snow parameter is sampled from MET via Xgeo for imputed values in areas where there are no messured values. The soil type, and soil texture is sampled from Kilden from Norwegian Institute of Bioeconomy Research.

4.2.1 Selection process

The selection process for finding these station can be compiled into these steps

1. Recommendation from Norwegian Institute of Bioeconomy Research
2. Compute the missing values in the data
3. Missing values analyse
4. Searching LMT database for alternative station candidates if current data is insufficient
5. If some station was replaced the repeat step 2

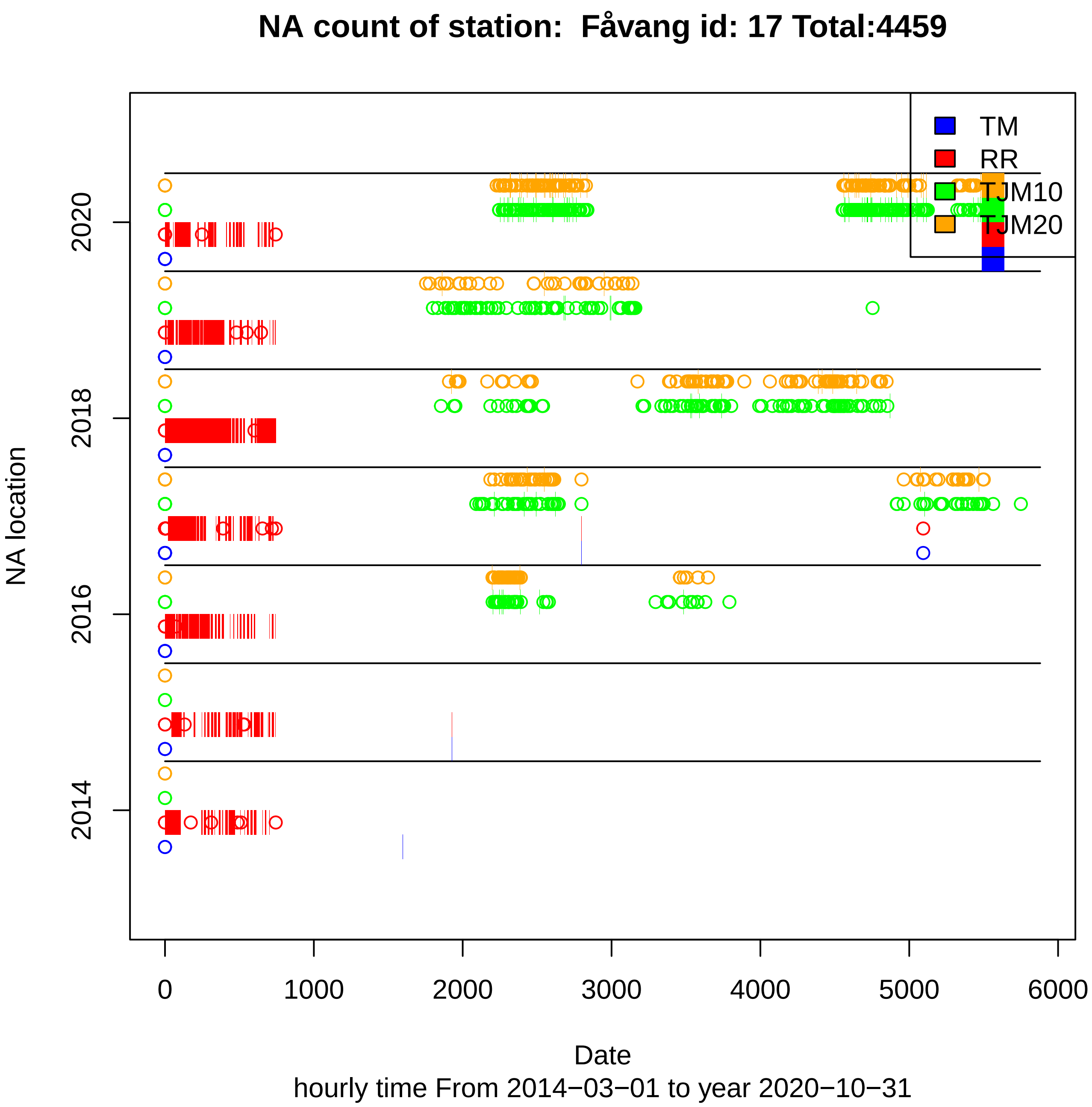

Figure 2: Visual representation of missing values at station 17 from 2014 to 2020

FROST	SQL approximate Code
Stations with rain	**SELECT** StationName **FROM** FROST ↪ **WHERE LIMIT** 4
Station ID	**SELECT** StationID, LMTID **FROM** FROST, ↪ LMT **WHERE**
LMT	Code
Meteorological data	**SELECT** ID,**date**,TM,RR,TJM10,TJM20 **FROM** ↪ LMT **WHERE date IN BETWEEN year** ↪ −03−01 **year**−10−31 **AND** ID = LMTID

Table 1: SQL version of the query requests sent to the different institutions.

The plots of stations follow a simple representation where the y-axis represent the year and the x-axis represent the index of the data as all tables are taken from the same period. A circle represent a singluar na values, while a band represent a series of 2 or more missing values. The colours represents the features used in this comperative study. This representation of the missing values will indicate sesonal, and systematic removal of data and give an overall indication of how much data is missing. To get further insight into the data a report is generated in parallel to the plots describing precise date and time of all values and which other parameter values is also missing values in the same period. See appendix A.2 for the full detail of the report generation and appendix B for na-plots of the station chosen for this study.

4.2.2 Collection of data

The method used was a powershell[9] script that called the respective institutions servers using the "curl" program[10] to send an http request for the timeseries starting from 2014 to 2020 in the interval 1 of May to 31 of October. Code for data collection can be viewed in appendix A.1. The data is stores as an either a csv file or a json file for easy retrieval and manual control of values.

4.2.3 Labeling of stations between Nibio and MET

Since Nibio and MET have different names for the same stations one must compile a list that converts Nibio ID to MET ID. This was performed with these requests Where ID is the Nibio Id for the given station, Frost.ID is the MET id, ID.latitude is the latitude gathered from Nibio, ID.longitude is the longitude gathered from Nibio. These variables can be swaped out for the relevant station.

4.2.4 Storage of data

The storage of the data is done through two data structures; Hashmap and DataFrame from the package pandas. The transformation of data is done with a costume datatype called "DataFile-Handler" which is converted to a module for convenience. The keys for the hashmap is chosen by the naming of the data files and the pattern given to the class. To escalete modeling the data will also be exported to a binary file for faster retrieval.

The data structure used to store the data from the different stations is called "DataFileHandler" and stores the data in a tree-structure where indexes are dictated by the filename. It has

[8] Format month-day
[9] Version 7.3.11
[10] curl 8.4.0 (Windows) libcurl/8.4.0 Schannel WinIDN

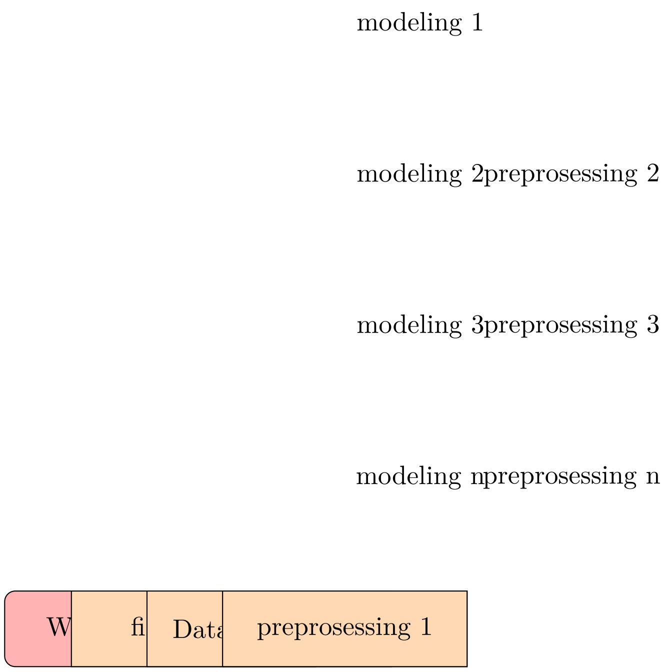

Figure 3: Compressed structure of study

several built-in functions to assist with data partitioning, and merging of data. This makes it easier to move and store all 846 720 observations from 16 station from 4 regions[11].

4.3 Data cleaning and treatment

To use the data in this study it must be cleaned and treated for training. The following methods were picked common practice in litterateur with new methods based on the decomposition of the data in the from of Seasonal-Trend decomposition using LOESS (STL)[15][12].

4.3.1 Outlier detection and removal

Though the data fetched from LMT is treated and controlled the external data from MET might not be, and this research project incorporated raw, untreated data from LMT to fill inn missing values. This paper has done empirical studies to find out which method to use in the prepossessing step of training the models. The selected methods are

1. model based
    - Autoregressive Integrated Moving Average (ARIMA)
    - LSTM
2. statistic based

[11]there are 4 stations per region.

[12]In this study we expand this for multiple seasons using Multiple Seasonal-Trend decomposition using LOESS (MSTL)[16], but the theory of this imputation method remains the same.

- backwards and forwards first observations
- rolling mean
- linear imputation

3. STL decomposition with above methods

4.3.2 Missing value imputation

The data has missing values, in particular during early Fall when there were sub-zero temperatures meaning any rain measurements done during this period would have unpredictable fluctuations since at negative temperatures water can freeze, get clogged up with residual bio-material from the surrounding area

1. linear imputation
2. backwards and forwards first available observation
3. global mean replacement
4. STL decomposition with above methods to impute components

The last method, using STL, was chosen because it would in principle be simpler to impute a less noisy signal than a noisy one.

4.4 Setup of models

The models are set up in according to the relevant paper the model is fetched from, alternatively reuse the code made by the author. When importing the data to the model there will be modifying to the original code to facilitate for the model as far as it goes. Any modifications will be in the appendix under section A. For the convenience of the reader all code is using the sklearn estimator class to make all the models discuses in this study more user friendly and compatible with sklearns other functions. The details of the models will be discussed in section 3, this section discusses the setup and implementation of the models.[13]

4.4.1 Basic Linear model

The linear model (sec 3.1) utilises in the study is created from the python model sklearn (or scikit-learn according to pythons package manager)

4.5 Use of Artificial Intelligence in this paper

In this paper there has been used Artificial Intelligence (AI), specifically Bing Chat / Copilot hosted by Microsoft Cooperation with special agreement with The Norwegian University of Life Sciences, for the following purposes:

1. Formalising sentences and rephrasing sentences.
2. Spellchecking
3. Code generation of basic consepts and structures (tree traversal, template generic class)

[13] Caution to the reader; The code used was run on a Linux subsystem on windows due to the fact that the current version of tensorflow can't run on Windows.

4. Better understanding of domain

All code have been manually check and verified in a separate environment and dedicated class for testing and verification. No confidential information or data has been past into the AI and only generic questions regarding broad topics has been prompted to the AI. Any topics discussed with the chat bot / AI were double checked with research papers and textbooks for verification, and any sources brought up by the AI was checked and verified.

5 Results

poop

6 Discussion

6.1 Future work

The models chosen in this study is not a representative sample of current knowledge of soil temperature modelling, and this study did not aim for optimizing the models beyond what the original authors have already done with the exception for base models used for comparison puposus. A more comprehensive is needed of more complex models that utelises cutting edge technologies, techniques, and theory. One of which is logic based models, for instance ASPER[17] that tries to incoerate logical descriptions of the problem and limits the model for better or equal results based on fewer samples[18]. Another approch is to incorporate randomness into the deterministic models to explain the variation in the data, forinstance fractional Brownian motion[19].

7 Conclution

Everything is okay

8 Bibliography

References

[1] Qingliang Li, Yuheng Zhu, Wei Shangguan, Xuezhi Wang, Lu Li, and Fanhua Yu, "An attention-aware LSTM model for soil moisture and soil temperature prediction," *Geoderma*, volume 409, page 115 651, Mar. 2022, ISSN: 00167061. DOI: 10.1016/j.geoderma.2021.115651. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S001670612100731X (visited on 10/05/2023).

[2] Meysam Alizamir, Ozgur Kisi, Ali Najah Ahmed, Cihan Mert, Chow Ming Fai, Sungwon Kim, Nam Won Kim, and Ahmed El-Shafie, "Advanced machine learning model for better prediction accuracy of soil temperature at different depths," *PLOS ONE*, volume 15, number 4, Lei Lin, Ed., page 25, Apr. 14, 2020, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0231055. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0231055 (visited on 09/29/2023).

[3] Ha Seon Sim, Dong Sub Kim, Min Gyu Ahn, Su Ran Ahn, and Sung Kyeom Kim, "Prediction of strawberry growth and fruit yield based on environmental and growth data in a greenhouse for soil cultivation with applied autonomous facilities," *Korean Journal of Horticultural Science and Technology*, volume 38, number 6, pages 840–849, Dec. 31, 2020, ISSN: 1226-8763, 2465-8588. DOI: 10.7235/HORT.20200076. [Online]. Available: https://www.hst-j.org/articles/doi/10.7235/HORT.20200076 (visited on 10/05/2023).

[4] Katri Rankinen, Tuomo Karvonen, and D. Butterfield, "A simple model for predicting soil temperature in snow-covered and seasonally frozen soil: Model description and testing," *Hydrology and Earth System Sciences*, volume 8, number 4, pages 706–716, Aug. 31, 2004, ISSN: 1607-7938. DOI: 10.5194/hess-8-706-2004. [Online]. Available: https://hess.copernicus.org/articles/8/706/2004/ (visited on 03/17/2023).

[5] Cong Li, Yaonan Zhang, and Xupeng Ren, "Modeling hourly soil temperature using deep BiLSTM neural network," *Algorithms*, volume 13, number 7, page 173, Jul. 17, 2020, ISSN: 1999-4893. DOI: 10.3390/a13070173. [Online]. Available: https://www.mdpi.com/1999-4893/13/7/173 (visited on 03/17/2023).

[6] Joris C. Stuurop, Sjoerd E.A.T.M. Van Der Zee, and Helen Kristine French, "The influence of soil texture and environmental conditions on frozen soil infiltration: A numerical investigation," *Cold Regions Science and Technology*, volume 194, page 103 456, Feb. 2022, ISSN: 0165232X. DOI: 10.1016/j.coldregions.2021.103456. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0165232X21002378 (visited on 01/31/2024).

[7] Mathieu Lepot, Jean-Baptiste Aubin, and François Clemens, "Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment," *Water*, volume 9, number 10, page 796, Oct. 17, 2017, ISSN: 2073-4441. DOI: 10.3390/w9100796. [Online]. Available: http://www.mdpi.com/2073-4441/9/10/796 (visited on 02/17/2024).

[8] Tuomo Karvonen, *A model for predicting the effect of drainage on soil moisture, soil temperature and crop yield*. Otaniemi, Finland: Helsinki University of Technology, Laboratory of Hydrology and Water Resources Engineering, 1988, xvi, 215, Open Library ID: OL15197205M.

[9] Jean Baptiste Joseph Fourier and Alexander Freeman, *The analytical theory of heat*. New York: Cambridge University Press, 2009, OCLC: 880311398, ISBN: 978-1-108-00178-6.

[10] Finn Plauborg, "Simple model for 10 cm soil temperature in different soils with short grass," *European Journal of Agronomy*, volume 17, number 3, pages 173–179, Oct. 2002, ISSN: 11610301. DOI: 10.1016/S1161-0301(02)00006-0. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1161030102000060 (visited on 03/17/2023).

[11] Radka Kodešová, Miroslava Vlasáková, Miroslav Fér, Daniela Teplá, Ondřej Jakšík, Pavel Neuberger, and Radomír Adamovský, "Thermal properties of representative soils of the czech republic," *Soil and Water Research*, volume 8, number 4, pages 141–150, Dec. 31, 2013, ISSN: 18015395, 18059384. DOI: 10.17221/33/2013-SWR. [Online]. Available: http://swr.agriculturejournals.cz/doi/10.17221/33/2013-SWR.html (visited on 02/29/2024).

[12] Carl Runge, "Ueber die numerische Aufl sung von Differentialgleichungen," *Mathematische Annalen*, volume 46, number 2, pages 167–178, Jun. 1895, ISSN: 0025-5831, 1432-1807. DOI: 10.1007/BF01446807. [Online]. Available: http://link.springer.com/10.1007/BF01446807 (visited on 02/29/2024).

[13] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural Computation*, volume 9, number 8, pages 1735–1780, Nov. 1, 1997, ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: https://direct.mit.edu/neco/article/9/8/1735-1780/6109 (visited on 10/18/2023).

[14] Guillaume Chevalier, *English: Schematic of the long-short term memory cell, a component of recurrent neural networks*, May 16, 2018. [Online]. Available: https://commons.wikimedia.org/wiki/File:LSTM_Cell.svg (visited on 12/10/2023).

[15] Robert B. Cleveland, William S. Cleveland, and Irma Terpenning, "STL: A seasonal-trend decomposition procedure based on loess," *Journal of Official Statistics*, volume 6, number 1, page 3, Mar. 1990, Num Pages: 3 Place: Stockholm, Sweden Publisher: Statistics Sweden (SCB), ISSN: 0282423X. [Online]. Available: https://www.proquest.com/docview/1266805989/abstract/4DFACD0236364745PQ/1 (visited on 10/11/2023).

[16] Kasun Bandara, Rob J. Hyndman, and Christoph Bergmeir, *MSTL: A seasonal-trend decomposition algorithm for time series with multiple seasonal patterns*, Jul. 28, 2021. arXiv: 2107.13462[stat]. [Online]. Available: http://arxiv.org/abs/2107.13462 (visited on 03/19/2024).

[17] Trung Hoang Le, Huiping Cao, and Tran Cao Son, *ASPER: Answer set programming enhanced neural network models for joint entity-relation extraction*, version: 1, May 24, 2023. arXiv: 2305.15374[cs]. [Online]. Available: http://arxiv.org/abs/2305.15374 (visited on 03/12/2024).

[18] Fadi Al Machot, *Bridging logic and learning: A neural-symbolic approach for enhanced reasoning in neural models (ASPER)*, Dec. 18, 2023. arXiv: 2312.11651[cs]. [Online]. Available: http://arxiv.org/abs/2312.11651 (visited on 03/12/2024).

[19] A. Di Crescenzo, B. Martinucci, and V. Mustaro, *A model based on the fractional brownian motion for the temperature fluctuation in the campi flegrei caldera*, Jul. 20, 2022. arXiv: 2110.13546[math,stat]. [Online]. Available: http://arxiv.org/abs/2110.13546 (visited on 03/12/2024).

A scrips

A.1 Powershell

A.1.1 Nibio data gathering

```
$baseUri = 'https://lmt.nibio.no/agrometbase/showweatherdata.php'
$datapath = "$($PSScriptRoot)/../../data/raw_data/nibio"

$line = Get-Content -Path "$($PSScriptRoot)/../../PRIVATE_FILES/
    ↪ frost_met_client.txt" -TotalCount 1
$FrostID = $line.Split(": ")[1]
$bases = @(
10 , 11 , 12 , 145 , 143 , 13 , 86 , 133 , 14 , 127 , 140 , 15 , 16 ,
    ↪ 17 , 18 , 19 , 20 , 110 , 21 , 121 , 87 , 22 , 23 , 24 ,
    ↪ 25 , 26 , 27 , 28 , 93 , 57 , 29 , 149 , 141 , 30 , 31 ,
    ↪ 65 , 62 , 32 , 33 , 82 , 71 , 34 , 104 , 35 , 90 , 81 ,
    ↪ 147 , 36 , 37 , 38 , 97 , 83 , 130 , 39 , 40 , 41 , 63 ,
    ↪ 42 , 131 , 134 , 142 , 43 , 108 , 44 , 64 , 129 , 45 , 46
    ↪ , 47 , 123 , 48 , 91 , 49 , 50 , 51 , 52 , 54 , 55 , 144
    ↪ , 118 , 53 , 5 , 61 , 72
)

$jobs = @()

foreach ($base in $bases) {
        foreach ($year in 2014..2022) {
                $full_path = "$($datapath)/
                    ↪ weather_data_raw_hour_stID$($base)_y$(
                    ↪ $year).csv"
                if(Test-Path $full_path -PathType Leaf){
                        continue
                }
    $jobs += Start-ThreadJob -Name "w$($base)-y$($year)" -ScriptBlock
        ↪ {
                        param($base, $baseUri, $year, $storage)
                        $form = @{
                                weatherstation=$base
                                loginterval=1
                                valuetype="value_raw"
                                date_start="$($year)-03-01"
                                date_end="$($year)-03-31"
                                format="csv"
                                separator="dot"
                        }

                        $Uri = "$($baseUri)?"

                        $Uri += "weatherstation=$($form["
                            ↪ weatherstation"])&"
```

```
                    foreach ($el in @(1,297,6,7)) { # 1, 297 -<
                        ↪ temp, nedbør
                        $Uri += "elementMeasurementTypes%5B%5
                            ↪ D=$($el)&"
                    }

                    foreach ($key in @("loginterval","valuetype
                        ↪ ","date_start","date_end","format
                        ↪ ","separator")) {
                        $Uri += "$($key)=$($form[$key])&"
                    }
                    $Uri = $Uri.Substring(0,$Uri.length-1)
                    Write-Host $Uri
                    curl $Uri --output $storage --retry 3 --retry
                        ↪ -delay 5
    } -ArgumentList $base, $baseUri, $year, $full_path
            Write-Host "Written w$($base)-y$($year)."
        }
}
if($jobs.length -eq 0) {
        Write-Host "No jobs"
} else {
        Write-Host "Downloads started..."

        Wait-Job -Job $jobs

        foreach ($job in $jobs) {
                Receive-Job -Job $job
        }
}
```

A.1.2 Frost data gathering

```
$line = Get-Content -Path "$($PSScriptRoot)/../../PRIVATE_FILES/
        ↪ frost_met_client.txt" -TotalCount 1
$FrostID = $line.Split(": ")[1]
$frosturi = "https://frost.met.no/sources/v0.jsonld?types=
        ↪ SensorSystem&geometry=nearest(POINT(%20))"
$frosturi2 = "https://frost.met.no/observations/v0.csv?"

$datapath = "$($PSScriptRoot)/../../data/info"
$datafile = "$($datapath)/StationIDInfo.csv"
$stationlist = @(
10,11,12,145,143,13,86,133,14,127,140,15,16,17,18,19,20,110,21,121,87,22,23,24,25,26
        ↪
)
```

```
$attributes = @(
        "ID","Name","Long","Lati","FrostName","ErrorDist","S0","D0","
            ↪ S1","D1","S2","D2","S3","D3","S4","D4"
)

New-Item -Path $datafile -Value "$($attributes -join ";")`n"

foreach($id in $stationlist){
        $webreq = Invoke-WebRequest -Uri "https://lmt.nibio.no/
            ↪ services/rest/weatherstation/getstation?
            ↪ weatherStationId=$($id)" | ConvertFrom-Json
        Add-Content -Path $datafile -Value (@($webreq.
            ↪ weatherStationId,$webreq.name,$webreq.latitude,
            ↪ $webreq.longitude) -join ";") -NoNewline -
            ↪ Encoding "UTF8"
        $frostlocal = curl "https://frost.met.no/sources/v0.jsonld?
            ↪ types=SensorSystem&geometry=nearest(POINT($(
            ↪ $webreq.longitude)%20$($webreq.latitude)))" -u "$
            ↪ ($FrostID):" | ConvertFrom-Json
        $frostdata = curl "https://frost.met.no/sources/v0.jsonld?
            ↪ types=SensorSystem&elements=sum(
            ↪ precipitation_amount%20PT1H)&geometry=nearest(
            ↪ POINT($($webreq.longitude)%20$($webreq.latitude))
            ↪ )&nearestmaxcount=5" -u "$($FrostID):" |
            ↪ ConvertFrom-Json

        Add-Content -Path $datafile -Value ";$($frostlocal.data.id);$
            ↪ ($frostlocal.data.distance)" -NoNewline
        foreach($i in 0..4){
                $substat = $frostdata.data[$i]
                Add-Content -Path $datafile -Value ";$(@($substat.id,
                    ↪ $substat.distance) -join ";")" -
                    ↪ NoNewline
        }
        Add-Content -Path $datafile -Value "`n" -NoNewline
        $j = 1
        do {
                        Write-Host "Attempting: id $($id) on index $(
                            ↪ $j)"
                        $weatherdata = curl "https://frost.met.no/
                            ↪ observations/v0.csv?sources=$(
                            ↪ $frostdata.data[$j].id)&
                            ↪ referencetime=2014-03-1%2F2020
                            ↪ -10-31&elements=sum(
                            ↪ precipitation_amount%20PT1H)" -u
                            ↪ "$($FrostID):"
                        $fileoutput = "$($datapath)/../raw_data/MET/
                            ↪ StationTo_$($id)_FROM_$(
```

```
                         ↪ $frostdata.data[$j].id).csv"
            try {
                  $weatherdata = $weatherdata | ConvertFrom-
                         ↪ Json
                  Write-Host "$($weatherdata."@type")"
                  if($weatherdata."@type" -eq "ErrorResponse"){
                        Write-Host "Did not find for id $($id
                               ↪ ) at index $($j)"
                  } else {
                        Write-Host "Found for id $($id) at
                               ↪ index $($j)?"
                        Add-Content -Path $fileoutput -Value
                               ↪ $weatherdata
                  }
            } catch {
                  Write-Host "Found for id $($id) at index $($j
                         ↪ )"
                  Add-Content -Path $fileoutput -Value
                         ↪ $weatherdata
            }
            $j = $j + 1
      } while($j -le 4)
}
```

A.2 R

```
##
       ↪ ------------------------------------------------------------
       ↪
library(dplyr) # for data manipulation and transformation
library(tidyverse) # for a collection of packages for data
       ↪ manipulation and visualization
library(stats) # for statistical functions and models
library(tsfeatures)
library(lubridate)
library(runner)

library(TSdist) # for calculating distance measures between time
       ↪ series
library(forecast) # for time series forecasting
library(TSA) # for time series analysis
library(tseries)
library(signal)
library(imputeTS)

library(ggplot2) # for creating beautiful and customizable
       ↪ visualizations
library(gridExtra) # for arranging multiple plots on a grid
library(RColorBrewer) # for creating color palettes for your plots
library(MLmetrics)
```

```
library(summarytools)


##
    ↪ ----------------------------------------------------------------
    ↪
# path definitions

ROOT <- "../../"

DATA_PATH <- paste0(ROOT, "data/")

DATA_INFO <- paste0(DATA_PATH, "info/")
DATA_INFO_NIBIO_FILE <- paste0(DATA_INFO ,"lmt.nibio.csv")
DATA_INFO_FROST_FILE <- paste0(DATA_INFO, "Frost_stations.csv")
DATA_FILE_SOIL_STATIONS <- paste0(DATA_INFO, "'Stasjonsliste␣
    ↪ jordtemperatur␣modellering.xlsx'")

DATA_COLLECTION <- paste0(DATA_PATH, "raw_data/")
DATA_COLLECTION_STAT <- paste0(DATA_COLLECTION, "Veret␣paa␣Aas␣2013-␣
    ↪ 2017/") # pattern -> 'Veret paa Aas 2013- 2017/Veret paa
    ↪ Aas {YYYY}.pdf'
DATA_COLLECTION_TIME <- paste0(DATA_COLLECTION, "Time␣2013-␣2023/") #
    ↪ pattern -> Time{YYYY}.xlsx
DATA_COLLECTION_NIBIO <- paste0(DATA_COLLECTION, "nibio/") # pattern
    ↪ -> weather_data_hour_stID{id}_y{year}.csv

# ID definitions

station_names <- read.csv(DATA_INFO_NIBIO_FILE,
                          header=TRUE,
                          row.names="ID",
                          colClasses=c(ID="integer",Navn="character")
                              ↪ )

nibio_id = list(
    Innlandet = c(11,17,18,26,27),
    Trøndelag = c(15,57,34,39,43),
    Østfold = c(37,41,52,118,5),
    SørVestlandet = c(14,29,32,48,22),
    Vestfold = c(30,38,42,50)
)

# function definitions

file_name.nibio <- function(station_id, year, path = NULL){
    if(is.null(path)){
        pattern = paste0(DATA_COLLECTION_NIBIO,"weather_data_hour_
            ↪ stID",station_id,"_y",year,".csv")
```

```
    } else {
        pattern = sprintf(path,station_id,year)
    }
    return(pattern)
}

data.nibio <- function(station_id,year, path = NULL){
    path <- file_name.nibio(station_id,year, path = path)
    data_nibio <- read.csv(path,
                           header=T, col.names = c("Time","TM","RR","
                           ↪ TJM10","TJM20"))
    data_nibio <- mutate(data_nibio,across(
                                    "Time",
                                  str2date))
    data_nibio <- column_to_rownames(data_nibio, var = "Time")
    data_nibio <- mutate_at(data_nibio,c("TM","RR","TJM10","TJM20"),
             ↪ as.numeric)
    return(data_nibio)
}
na.interpol.cust <- function(data, maxgap = Inf, n.p,
                                  s.window = 10, alg.option = "linear"){
    data.decomp <- stlplus::stlplus(data,n.p = n.p, s.window = s.
             ↪ window)
    data.new <- rep(0,length.out = length(data))
    for(part in c("seasonal", "trend", "remainder")){
        data.new <- data.new + na_interpolation(data.decomp$data[,
                 ↪ part],
                                              maxgap=maxgap,
                                              option = alg.option)
    }
    return(data.new)
}
str2date <- function(x) {
    return(as.POSIXlt(paste0(x,"00"),
                      format = "%Y-%m-%d %H:%M:%S%z",
                      tz="GMT"))
}

na.interplol.kal <-function(data, maxgap = Inf, n.p,
                                  s.window = 10, alg.option = "StructTS"){
    data.decomp <- stlplus::stlplus(data,n.p = n.p, s.window = s.
             ↪ window)
    data.new <- rep(0,length.out = length(data))
    for(part in c("seasonal", "trend", "remainder")){
        data.new <- data.new + na_kalman(data.decomp$data[,part],
                                              maxgap=maxgap,
                                              model = alg.option,
                                        smooth = TRUE)
    }
```

```
    return(data.new)
}

find.na.index.length <- function(x){ # antar at x er bool vektor
    i <- 1 # starting index
    na.data <- data.frame()
    while(i <= length(x)){
        sample.data <- x[i:length(x)]
        first <- match(T, sample.data, nomatch = -1)
        if(first < 0) {
            break
        }
        last <- match(F, sample.data[first:length(sample.data)],
            ↪ nomatch = length(sample.data[first:length(sample.
            ↪ data)])+1) - 2 + first

        na.data <- rbind(na.data, data.frame(Length = c(last-first +
            ↪ 1), First = c(first+i-1), Last = c(last+i-1)))
        i <- i + last
    }
    return(na.data)
}


##
    ↪ ----------------------------------------------------------------
    ↪
blocks.index <- c()
len.na <- 8
len.val <- 12

data.check <- 1:5880
i <- 0
while(i < 5880){
    i <- i + len.val - 1
    blocks.index <- append(blocks.index,seq(i,i+len.na-1))
    i <- i + len.na
}
blocks.index <- blocks.index[blocks.index <= 5880]


##
    ↪ ----------------------------------------------------------------
    ↪
#library(moments)
data_nibio_no_na <- data.nibio(14,2019)
col.name <- "TM"

faulty.data <- data_nibio_no_na
```

```r
faulty.data[blocks.index,col.name] <- NA

fixed.data <- na_interpolation(faulty.data[,col.name], option="spline
    ↪ ", method = "periodic")
abs.diff <- fixed.data - data_nibio_no_na[,col.name]
print(paste("µ",mean(abs.diff),"std:",sqrt(var(abs.diff)),"skewness:"
    ↪ ,skewness(abs.diff)))
plot((abs.diff),xlim = c(0,5880))

fixed.data <- na.interpol.cust(faulty.data[,col.name], n.p = 21,alg.
    ↪ option="spline", method = "periodic")
abs.diff <- fixed.data - data_nibio_no_na[,col.name]
print(paste("µ",mean(abs.diff),"std:",sqrt(var(abs.diff)),"skewness:"
    ↪ ,skewness(abs.diff)))
plot((abs.diff),xlim = c(0,5880))


##
    ↪ ------------------------------------------------------------------------
    ↪
# RR hadde ikke noe serlig, men hadde en rep ~= 31
# TM ~= 24?
# TJM10 ~= 24?
# TJM20 ~= 21?
perid <- c(TM = 24,TJM10 = 24, TJM20 = 24, RR = 31)

data.rle <- rle(is.na(data_nibio[,"TJM20"]))
data.max <- max(data.rle$lengths[data.rle$values])
indexes <- find.index.rle.bool(data.rle,data.max)
print(data.max)

for(col in c("TJM20")){
    input <- as.ts(na.interpol.cust(data_nibio[,col],n.p=perid[col]))
    plot(input,xlim = c(indexes[1]-100,indexes[2]+100))
    abline(v=indexes[1],col = "red")
    abline(v=indexes[2],col = "red")
    title(paste(col,"STL + naive"))
}

for(col in c("TJM20")){
    input <- as.ts(na_interpolation(data_nibio[,col]))
    plot(input,xlim = c(indexes[1]-100,indexes[2]+100))
    abline(v=indexes[1],col = "red")
    abline(v=indexes[2],col = "red")
    title(paste(col,"naive"))
}


##
```

```
    ↪ ——————————————————————————————————————————————————
    ↪

feature.name = c("TM","RR","TJM10","TJM20")
na.run.tables <- c()
full.count <- c()

notible_run <- 24*7
warning_run <- 8*2 # imputering fra begge ender

cat("Null␣count␣of␣data.",
            file = "data.txt",sep="\n")
cat(paste("notable␣runs,defined␣by␣nb␣length",notible_run,"and␣
    ↪ warning␣length",warning_run,"\n
    ↪ ##################################"),
            file = "NB_data.txt",sep="\n")

station_names <- read.csv(DATA_INFO_NIBIO_FILE,
                         header=TRUE,
                         row.names="ID",
                         colClasses=c(ID="integer",Navn="character")
                              ↪ )

na.run.station.year.feature <- list()

sub_set <- unlist(nibio_id)

all.id <- as.numeric(rownames(station_names))

for(id in all.id){
    # beginning plot
    pdf(file = paste0(ROOT,"plots/plot-",id,".pdf"))
    plot(NULL,
         sub = "hourly␣time␣From␣2014-03-01␣to␣year␣2020-10-31",
         xlab="Date", ylab="NA␣location",
         xlim = c(0,5881), ylim = c(2013,2021))

    colours <- c(TM ="blue", RR = "red",TJM10 = "green",TJM20 = "
        ↪ orange")
    lev <- seq(-1/2,1/2,length.out=5)
    names(lev) <- feature.name

    numb <- 0
    denom <- 0
    na.run.count <- matrix(rep(0,length=5880*4),nrow = 5880, ncol =
        ↪ 4)
    colnames(na.run.count) <- feature.name
    na.count <- c()
    na.count.year <- c()
```

```
na.matrix.total <- NULL
#na.run.station.year.feature[[as.character(id)]] <- c()
#data_plot <- ggplot(title = paste("NA count of staion:",station_
    ↪ names[as.character(id),],"id:",id))
na.plot <- FALSE
cat(paste("**************","station",id,"**************"),
    ↪ append=T,sep="\n",file = "NB_data.txt")
for(year in seq(2014,2020)){

    # Drawing seperating lines

    lines(c(0,5880),c(year + 1/2,year + 1/2), col = "black")

    #lev <- seq(-1/2,1/2,length.out=5)
    #names(lev) <- c("TM","RR","TJM10","TJM20")
    #lev
    #lev["TJM20"]
    #lev[match("TJM20",names(lev))+1]
    cat(paste(":::::::year",year,":::::::"),append=T,sep="\n",
        ↪ file = "NB_data.txt")
    data_nibio <- suppressWarnings(data.nibio(id,year)) # henter
        ↪ data
    data_nibio <- data_nibio[rownames(data_nibio) ,]#> paste0(
        ↪ year,"-04-01"),]
    data_nibio_raw <- suppressWarnings(data.nibio(id,
                                      year,
                                      path=paste0(DATA_COLLECTION_
                                                 ↪ NIBIO,
                                                 "weather_data_raw_
                                                        ↪ hour_
                                                        ↪ stID%i_y
                                                        ↪ %i.csv"
                                                 )
                                      ))

    data_nibio_raw[!is.na(data_nibio_raw[,"TM"]) & (data_nibio_
        ↪ raw[,"TM"] <= 0),"RR"] <- NA

    data_nibio[1:nrow(data_nibio_raw),"RR"] <- data_nibio_raw[1:
        ↪ nrow(data_nibio_raw),"RR"]

    #na.run.station.year.feature[[as.character(id)]][[as.
        ↪ character(year)]] <- c()

    # Na analesys

    cat("————Matrix representation, and pair NA's————",
        ↪ append =T,sep="\n\t",file = "NB_data.txt")
```

```
data.matrix <- as.matrix(ifelse(is.na(data_nibio),1,0))

data.matrix.sq <- t(data.matrix)%*%data.matrix
if(is.null(na.matrix.total)){
    na.matrix.total <- data.matrix.sq
} else {
    na.matrix.total <- na.matrix.total + data.matrix.sq
}

cat("\t",append=T, file = "NB_data.txt",sep = "\t")
suppressWarnings(write.table(data.matrix.sq,append =T, file =
        ↪ "NB_data.txt",sep = "\t"))

cat(paste("Total NA:",sum(diag(data.matrix.sq))), file = "NB_
        ↪ data.txt",append=T, sep="\n")

na.check <- is.na(data_nibio)
if(any(na.check)){
    if(length(na.count) == 0){
        na.count <- ifelse(na.check, 1, 0)
    } else {
        na.count <- na.count + ifelse(na.check, 1, 0)
    }
    #na.count.year[[as.character(year)]] <- sum(na.check)/(
            ↪ nrow(data_nibio)*4)
    na.plot <- TRUE

    for(cols in feature.name){ # checker run for hver kolonne
        run_table <- table(NULL)
        cat(paste("\n-----------station",id,"year",year,"
                ↪ feature",cols,"------------"),
                file = "NB_data.txt",append=T, sep="\n")
        if(sum(na.check[,cols]) > 0){
            run_na <- find.na.index.length(na.check[,cols])
            #na.run.station.year.feature[[as.character(id)
                    ↪ ]][[as.character(year)]][[as.
                    ↪ character(cols)]] <- table(run_na)
            #print(paste("year:",year,"feature:",cols))
            #print(run_na)

            points(c(0,0,0,0),lev[1:4] + year + 1/8, col =
                    ↪ colours)

            for(ind in 1:nrow(run_na)){
                c <- run_na[ind,"Length"]
                dates <- rownames(data_nibio)[c(run_na$First[
                        ↪ ind],run_na$Last[ind])]
                if(any(is.na(dates))){
                    print(dates)
```

```
        }
        cat(paste("\t-\t",dates[1],"|->",c,"run",
             ↪ ifelse(c != 1,paste("\t|->",dates
             ↪ [2]),""),"\t"),
            file = "NB_data.txt",append=T,sep="")
        # plot conditions

        if(c == 1){
            # plot dot
            points(run_na$First[ind],year + lev[cols]
                 ↪ + 1/8, col = colours[cols])
        } else {
            # plot rectangle
            rect(run_na$First[ind],year + lev[cols],
                 run_na$Last[ind],year + lev[match(
                     ↪ cols,names(lev))+1],
                 col = colours[cols], border = NA
                )
        }

        # Write condition

         if(c >= notible_run){
            cat("(NB!)",file = "NB_data.txt",append=T
                 ↪ ,sep="\n")
        } else if(c > warning_run) {
            cat("(Warning)",
                file = "NB_data.txt",append=T,sep="\n
                    ↪ ")
        } else {
            cat("",
                file = "NB_data.txt",append=T,sep="\n
                    ↪ ")
        }
        na.run.count[c,cols] <- na.run.count[c,cols]
             ↪ + 1
    }
    run_table <- t(as.matrix(table(run_na$Length)))
}

cat(paste("\n————————————Total for station",id,"
         ↪ year",year,"in feature",cols,"
         ↪ ————————————"),
          file = "NB_data.txt",append=T,sep="\n")
cat("\t",append=T,file = "NB_data.txt",sep = "\t")
suppressWarnings(write.table(run_table,file = "NB_
         ↪ data.txt",append=T,sep = "\t"))
cat(paste("\t- total :\t",sum(na.check[,cols])),
    file = "NB_data.txt",append=T,sep="\n")
```

```
            }
        } else {
            cat(paste("\t-␣year",year,"without␣NA."),
                                          file = "NB_data.txt",append=T,sep="\n
                                              ↪ ")
            if(length(full.count[[as.character(id)]]) == 0){
                full.count[[as.character(id)]] <- 1/7
            } else {
                full.count[[as.character(id)]] <- full.count[[as.
                                ↪ character(id)]] + 1/7
            }
        }
        cat(paste(":::::::END␣year",year,"END:::::::"),append=T,sep="
                    ↪ \n",file = "NB_data.txt")
    }

    legend(x = "topright",legend=feature.name, fill = colours)

    if(na.plot){
        cat(paste("==========␣END␣station",id,"END␣==========="),
                    ↪ append=T,sep="\n",file = "NB_data.txt")
        cat(paste("Staion␣nr␣",id),
            file = "data.txt",append=T,sep="\n")
        #suppressWarnings(write.table(bad_data,file = "data.txt",
                    ↪ append=T)) # add labels... somehow
        cat(paste("prosent␣of",id,":",sum(na.count)/(nrow(data_nibio)
                    ↪ *4)),
            file = "data.txt",append=T,sep="\n")
        cat(paste("prosent␣of",id,"␣for␣years:"),
            file = "data.txt",append=T,sep="\n")
        cat(paste0(unlist(na.count.year), collapse = "\n"),
            file = "data.txt",append=T,sep="\n")
        cat("\t",append=T, file = "NB_data.txt",sep = "\t")
        suppressWarnings(write.table(na.matrix.total,file = "NB_data.
                    ↪ txt",append=T,sep = "\t"))
        cat(paste("Total:",sum(diag(na.matrix.total))),file = "NB_
                    ↪ data.txt", append =T, sep = "\n")
    }
    title(main = paste0("NA␣count␣of␣station:␣", station_names[as.
                ↪ character(id),],
                         "␣id:␣",id,
                   "␣Total:",sum(diag(na.matrix.total))))
    dev.off()
}


##
        ↪ ------------------------------------------------------------
        ↪
```

```
plot(data.nibio(16,2017)[,"TM"],type="l")
plot(forecast(fit,h=24*7),xlim=c(5500,6000))


##
    ↪ ---------------------------------------------------------------------
    ↪
imput_data <- na_interpolation(as.ts(data_nibio))


##
    ↪ ---------------------------------------------------------------------
    ↪
# RR hadde ikke noe serlig, men hadde en rep ~= 31 (måned baser?)
# TM ~= 24?
# TJM10 ~= 24?
# TJM20 ~= 21?
for(col in c("TM","TJM10","TJM20")){
    acf(imput_data[,col])
    title(col)
    pacf(imput_data[,col])
    title(col)
}


##
    ↪ ---------------------------------------------------------------------
    ↪
plot(stlplus::stlplus(imput_data[,"RR"],n.p = 31, s.window = 5,s.
    ↪ degree=2))


##
    ↪ ---------------------------------------------------------------------
    ↪
data_stat_id = matrix()

for(id in nibio_id){
    csv_files <- list.files(path = DATA_COLLECTION_NIBIO,
                            pattern = regex(paste0(".*ID",id,"_y\\d{4}.
                                ↪ csv")),
                                full.names = TRUE)
    combined_data <- lapply(csv_files,
                            read.csv,
                            header=T,
                            col.names = c("Time","TM","RR","TJM10","TJM20
                                ↪ ")) %>% bind_rows()
    combined_data <- combined_data %>% column_to_rownames(., var = '
        ↪ Time')
```

```
    combined_data <- mutate_at(combined_data, c("TM", "RR", "TJM10", "
            ↪ TJM20"), as.numeric)
}


##
        ↪ ----------------------------------------------------------------
        ↪
library( datasets )
data("faithful")
# z - scores & M a h a l a n o b i s d i s t a n c e
z <- scale(imput_data) %>% as.data.frame()
mahalanobis(z , center = c(0 ,0) , cov = cov( imput_data, use = "all.
        ↪ obs" ) )
# DBSCAN & LOF
library( dbscan )
dbscan( imput_data , eps = 1)$cluster == 0
lof( imput_data , minPts = 5)
# I s o l a t i o n forest
library( isotree )
iso_mod <- isolation.forest( imput_data )
predict( iso_mod , newdata = imput_data )
# one - class SVM
library( e1071 )
svm_mod <- svm ( imput_data , type = "one-classification")
print(sum(predict( svm_mod , newdata = imput_data )))


##
        ↪ ----------------------------------------------------------------
        ↪
adf.test(imputed.data[,"TJM10"])
kpss.test(imputed.data[,"TJM10"])
pp.test(imputed.data[,"TJM10"])
```

A.3 Python

```
#!/usr/bin/env python
# coding: utf-8

# # Data visualisation
#
# We start by importing the data

# In[1]:


import sklearn
import datetime
```

```
import os
import pickle
import copy

import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy as np
import pandas as pd
import statsmodels as sm
import torch.utils.data as Data

from tensorflow.keras.layers import Input, Conv2D, BatchNormalization
    ↪ , Activation
from tensorflow.keras.layers import MaxPooling2D, Dropout,
    ↪ Conv2DTranspose
from tensorflow.keras.layers import concatenate, Concatenate
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import metrics

#sklearn → model trening
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics         import accuracy_score, mean_squared_
    ↪ error, r2_score

#sklearn → data treatment
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.models import LinearRegression

#from ILSTM_Soil_model_main import lstm_interprety_soil_moisture as
    ↪ ILSTM
from My_tools import DataFileLoader as DFL # min egen
from My_tools import StudyEstimators as SE
# path definitions

ROOT = "../../"

PLOT_PATH = ROOT + "plots/"

DATA_PATH = ROOT + "data/"

METADATA_PRELOAD_DATA = ROOT + "PRIVATE_FILES/weatherdata.bin"

DATA_INFO = DATA_PATH + "info/"
DATA_INFO_NIBIO_FILE = DATA_INFO  + "lmt.nibio.csv"
DATA_INFO_FROST_FILE = DATA_INFO + "Frost_stations.csv"
DATA_INFO_NIBIO2FROST_FILE = DATA_INFO + "StationIDInfo.csv"
```

```
DATA_FILE_SOIL_STATIONS = DATA_INFO + "'Stasjonsliste jordtemperatur 
    ↪ modellering.xlsx'"

DATA_COLLECTION = DATA_PATH + "raw_data/"
DATA_COLLECTION_STAT = DATA_COLLECTION + "Veret paa Aas 2013- 2017/"
    ↪ # pattern -> 'Veret paa Aas 2013- 2017/Veret paa Aas {
    ↪ YYYY}.pdf'
DATA_COLLECTION_TIME = DATA_COLLECTION + "Time 2013- 2023/" # pattern
    ↪ -> Time{YYYY}.xlsx
DATA_COLLECTION_NIBIO = DATA_COLLECTION + "nibio/" # pattern ->
    ↪ weather_data_hour_stID{id}_y{year}.csv
DATA_COLLECTION_MET = DATA_COLLECTION + "MET/" # pattern -> StationTo
    ↪ _{id}_FROM_{FrostID}.csv


# ID definitions

station_names = pd.read_csv(DATA_INFO_NIBIO_FILE,
                            header=0,
                            index_col = "ID")



nibio_id = {
    "Innlandet" : ["11","18","26","27"],
    "Trøndelag" : ["15","57","34","39"],
    "Østfold" : ["37","41","52","118"],
    "Vestfold" : ["30","38","42","50"] # Fjern "50" for å se om bedre
        ↪ resultat
}


# Loading data from folders

# ## Function definitions

# In[2]:


def show_plot(data,plot_kwarg):
    """
        plots timeseries, assumes dataframe with a 'Time' columns
    """
    for d in range(len(data)):
        if d not in plot_kwarg:
            plt.plot(data[d].Time, data[d].iloc[:,data[d].columns !=
                ↪ "Time"])
        else:
            plt.plot(data[d].Time, data[d].iloc[:,data[d].columns !=
```

```
            ↪ "Time"],**plot_kwarg[d])

    if "xlabel" in plot_kwarg:
        plt.xlabel = plot_kwarg["xlabel"]
    else:
        plt.xlabel = "Time"

    if "ylabel" in plot_kwarg:
        plt.ylabel = plot_kwarg["ylabel"]
    else:
        plt.ylabel = "celsius␣degrees␣°C"

def table2Latex(table, dir_path, file_name, header = "", append =
        ↪ False):
    if os.path.isfile(dir_path + file_name + ".tex") & append:
        file = open(dir_path+file_name+".tex","a", encoding="utf-8")
    else:
        file = open(dir_path+file_name+".tex","w", encoding="utf-8")

    file.write(r"\begin{tabular}{|l|" + ["c" for _ in range(len(
            ↪ header))].join("|") + "|}")
    if header != "":
        file.write(header.join("&") + r"\\\hline")
    for row in table:
        file.write(row.join("&") + r"\\\hline")
    file.write(r"\end{tabular}")


# In[3]:
if os.path.exists(METADATA_PRELOAD_DATA):
    imputed_nibio_data = pickle.load(open(METADATA_PRELOAD_DATA,"rb")
            ↪ )
    print("Fetched␣data␣from", METADATA_PRELOAD_DATA)
else:
    nibio_data_ungroup = DFL.DataFileLoader(DATA_COLLECTION_NIBIO, r"
            ↪ weather_data_hour_stID(\d{1,3})_y(\d{4}).csv")
    nibio_data_ungroup.load_data(names = ["Time","TM","RR","TJM10","
            ↪ TJM20"])
    nibio_data = nibio_data_ungroup.group_layer(nibio_id)

    nibio_data_raw_ungroup = DFL.DataFileLoader(DATA_COLLECTION_NIBIO
            ↪ ,r"weather_data_raw_hour_stID(\d{1,3})_y(\d{4}).csv")
    nibio_data_raw_ungroup.load_data(names = ["Time","TM","RR","TJM10
            ↪ ","TJM20"])
    nibio_data_raw = nibio_data_raw_ungroup.group_layer(nibio_id)

    def dataframe_merge_func(x,y):
        y.iloc[y.iloc[:,1].notna() & (y.iloc[:,1] <= 0),2] = pd.NA
        x.iloc[0:y.shape[0],2] = y.iloc[0:y.shape[0],2]
```

```
        return x

    imputed_nibio_data = nibio_data.combine(nibio_data_raw,merge_func
            ↪ = dataframe_merge_func)
    pickle.dump(imputed_nibio_data,open(METADATA_PRELOAD_DATA,"wb"))
    print("dumped␣data␣to",METADATA_PRELOAD_DATA)


def attempt_fitting(base_model, param_area, nibio_data):
    best_plauborg = {
    "Score":np.inf,
    "mse":0,
    "r2":0,
    "year":0,
    "model":None
    }

    worst_plauborg = {
    "Score":-np.inf,
    "mse":0,
    "r2":0,
    "year":0,
    "model":None,
    }

    all_plauborg = []

    search_area = param_area

    base_model = base_model()

    for regi in nibio_id.keys():
        for i in range(2014,2022):
            # First we fetch region (regi), all stations (:), then
                ↪ relevant years ("2014":str(i)). Since we only
                ↪  look at one region at the time
            # we remove the root group (shave_top_layer()), then we
                ↪ merge the years (merge_layer(level = 1),
                ↪ level 1 since level 0 would be the stations
                ↪ at this point)
            # then make a list (flatten(), the default handeling is
                ↪ to put leafs in a list)

            data = nibio_data[regi,:,"2014":str(i)].shave_top_layer()
                ↪ .merge_layer(level = 1).flatten(return_key =
                ↪ True) # shape [(key, value)] ; looks at all
                ↪ previus years including this year
            test = nibio_data[regi,:,str(i+1)].shave_top_layer().
                ↪ merge_layer(level = 1).flatten(return_key =
```

```
    ↪ True) # shape [(key, value)] ; looks at the
    ↪ next year

data = [(k,v.infer_objects(copy=False).fillna(0)) for k,v
    ↪ in data] # Removes nan in a quick manner
test = [(k,v.infer_objects(copy=False).fillna(0)) for k,v
    ↪ in test] # but will be reviced.

model = GridSearchCV(copy.deepcopy(base_model),param_grid
    ↪ =search_area,pre_dispatch=20, n_jobs = -1)
n = 0
overall_r2 = None # approximates a average r2
overall_mse = None
for d,t in zip(data,test): # fitting model with all
    ↪ stations
    model.fit(d[1].loc[:,["Time","TM"]],d[1].loc[:,["
        ↪ TJM20"]]) # regions model
    s_model = GridSearchCV(copy.deepcopy(base_model),
        ↪ param_grid=search_area, n_jobs = -1).fit(
        ↪ d[1].loc[:,["Time","TM"]],d[1].loc[:,["
        ↪ TJM20"]])# Station model
    s_pred = s_model.predict(t[1].loc[:,["Time","TM"]])

    if overall_r2 is not None:
        overall_r2 = 1-mediant(1-overall_r2, 1-r2_score(t
            ↪ [1].loc[t[1].shape[0]-s_pred.shape
            ↪ [0]:,"TJM20"].to_numpy(),s_pred))
    else:
        overall_r2 = r2_score(t[1].loc[t[1].shape[0]-s_
            ↪ pred.shape[0]:,"TJM20"].to_numpy(),s_
            ↪ pred)

    if overall_mse is not None:
        overall_mse = (t[1].loc[t[1].shape[0]-s_pred.
            ↪ shape[0]:,"TJM20"].to_numpy()-s_pred)
            ↪ **2 + (overall_mse*n)
    else:
        overall_mse = (t[1].loc[t[1].shape[0]-s_pred.
            ↪ shape[0]:,"TJM20"].to_numpy()-s_pred)
            ↪ **2
    n += len(s_pred)
    overall_mse /= n

    show_plot([
        pd.DataFrame({"Time":t[1].loc[t[1].shape[0]-s_
            ↪ pred.shape[0]:,"Time"].to_numpy().
            ↪ ravel(),"TJM20":s_pred.ravel() - t
            ↪ [1].loc[t[1].shape[0]-s_pred.shape
            ↪ [0]:,["TJM20"]].to_numpy().ravel()})
```

```
        ],{0:{"label":"spesial"}})

    show_plot([
        pd.DataFrame({"Time":t[1].loc[t[1].shape[0]-s_pred.
                ↪ shape[0]:,"Time"],"TJM20":model.predict(t
                ↪ [1].loc[:,["Time","TM"]]).ravel() - t[1].
                ↪ loc[t[1].shape[0]-s_pred.shape[0]:,["
                ↪ TJM20"]].to_numpy().ravel()})
    ],{0:{"label":"global"}})

    plt.savefig(PLOT_PATH + base_model.__name__ +"_" + regi +
            ↪ "_y"+ str(i) + ".pdf")
    plt.clf()

    mse = {k:mean_squared_error(t.loc[t.shape[0]-s_pred.shape
            ↪ [0]:,"TJM20"].to_numpy().ravel(),model.
            ↪ predict(t.loc[:,["Time","TM"]])) for k,t in
            ↪ test}
    r2 = {k:r2_score(t.loc[t.shape[0]-s_pred.shape[0]:,"TJM20
            ↪ "].to_numpy().ravel(),model.predict(t.loc[:,[
            ↪ "Time","TM"]])) for k,t in test}

    print(base_model.__name__,":",regi,"from year 2014 to 
            ↪ year",i,":\n",
          "\tMSE:", mse,
          "\n\tR2:",r2,
          "\n\tparams:",model.best_params_)
    score = max(m/r if r != 0 else np.inf for m,r in zip(mse.
            ↪ values(),r2.values()))
    model_info = {
            "Name":base_model.__name__,
            "Score":score,
            "params":model.best_params_,
            "mse":mse,
            "r2": r2,
            "r2_spes": overall_r2,
            "mse_spes": overall_mse,
            "year_max": str(i+1),
            "region": regi,
            "model": model
        }
    all_plauborg.append(model_info)
    if score < best_plauborg["Score"]:
        best_plauborg = model_info
    elif score > worst_plauborg["Score"]:
        worst_plauborg = model_info

return {"all":all_plauborg,"best":best_plauborg,"worst":worst_
        ↪ plauborg}
```

```
# In[5]:


for regi in nibio_id.keys():
    show_plot([station.loc[:,["Time","TJM20"]] for station in imputed
            ↪ _nibio_data[regi,:].shave_top_layer().merge_layer(
            ↪ level=1).flatten()],{})
    plt.legend(nibio_id[regi])
    plt.title("Område:␣{},␣feature:␣{}".format(regi,"TJM20"))
    plt.savefig(PLOT_PATH + regi + '.pdf', bbox_inches='tight') # pdf
            ↪ for vectorised grafics.
    plt.clf() # clear current figure for the next figure


# The data is splitted among two collections of data, one is a pdf
        ↪ and the other is a `.xlsx` format. We start by collecting
        ↪ the data from the hourly data collection.


# ## Linear regression function
#
# This function does a transformation of the $m\times n$ matrix (our
        ↪ dataframe) to a $m \times p$ matrix. This can be seen as
        ↪ a kernel trick where we transform the data to a more
        ↪ seperable state to improve prediction. The scema for this
        ↪ model is
# $$
#     (\vec{F}\circ \mathbf{A})\vec{\beta}=\vec{y}+\vec{\varepsilon}
# $$

# In[ ]:


def all_permute(L):
    """
␣␣␣␣␣␣␣␣makes␣a␣list␣of␣size␣2^len(L)−1␣with␣all␣combinations
␣␣␣␣"""
    from itertools import permutations
    final_list = list(L)
    for n in range(2,len(L)+1):
        final_list.extend(set(permutations(L,n)))
    return final_list

def mediant(x: float,y: float):
    """
␣␣␣␣␣␣␣␣Takes␣the␣mediant␣of␣two␣fractions
␣␣␣␣␣␣␣␣␣␣␣␣a/b␣+␣c/d␣=␣(a+c)/(b+d)
␣␣␣␣"""
    frac_x = x.as_integer_ratio()
```

```
    frac_y = y.as_integer_ratio()
    comb_xy = (frac_x[0] + frac_y[0],frac_x[1] + frac_y[1])
    return comb_xy[0]/comb_xy[1]

def combine_years(X,Y):
    """
        Combines two dataframes
    """
    if isinstance(X,list) or isinstance(Y,list):
        pass
    if X.index == Y.index:
        return [X,Y]

# Linear Regression

result_fitting = attempt_fitting(LinearRegression,{"fit_intercept":[
    ↪ True,False],"Positive":[True,False]},imputed_nibio_data)

print("Linear Regresson best:",result_fitting["best"])
print("Linear Regresson worst:",result_fitting["worst"])
print("Linear Regresson median:",sorted(result_fitting["all"],key =
    ↪ lambda x: x["Score"])[int(len(result_fitting["all"])/2)])

# ### Plauborg regression
#
# Author Plauborg used the above model to predict soil temperature,
    ↪ but used previus time to make the model more time
    ↪ dependent and fourier terms to reflect changes during the
    ↪  year.

# In[ ]:


# In[ ]:

#! Need to adjust following code
result_fitting = attempt_fitting(SE.PlauborgRegresson,{"lag_max":
    ↪ range(2,8),"fourier_sin_length":range(2,10),"fourier_cos_
    ↪ length":range(2,10)},imputed_nibio_data)

print("best:",result_fitting["best"])
print("worst:",result_fitting["worst"])
print("median:",sorted(result_fitting["all"],key = lambda x: x["Score
    ↪ "])[int(len(result_fitting["all"])/2)])

best_data = imputed_nibio_data[best_plauborg := result_fitting["best"
    ↪ ],:,best_plauborg["year_max"]].merge_layer(level = 0)
worst_data = imputed_nibio_data[worst_plauborg := result_fitting["
    ↪ worst"],:,worst_plauborg["year_max"]].merge_layer(level =
```

```
    ↪ 0)

show_plot([
    pd.DataFrame({
        "Time":best_data.Time.iloc[:5879].to_numpy().ravel(),
        "TJM20":best_plauborg["model"].predict(best_data.loc[:5878,["
            ↪ Time","TM"]]).ravel() - best_data.loc[:5878,["
            ↪ TJM20"]].to_numpy().ravel()
    }) ],
    {})
plt.title("Y_pred␣-␣Y_truth")
plt.savefig(PLOT_PATH + "Plauborg_plot_best.pdf")
show_plot([
    pd.DataFrame({
        "Time":worst_data.Time.iloc[:5879].to_numpy().ravel(),
        "TJM20":worst_plauborg["model"].predict(worst_data.loc
            ↪ [:5878,["Time","TM"]]).ravel() - worst_data.loc
            ↪ [:5878,["TJM20"]].to_numpy().ravel()
    }) ],
    {})
plt.title("Y_pred␣-␣Y_truth")
plt.savefig(PLOT_PATH + "Plauborg_plot_worst.pdf")


# In[ ]:


# imputed_nibio_data["Vestfold",:,"2019"].DictData


# ### Rankin regression
#
# This regression tries to solve the following integreal using an FDM
    ↪ .
#
# $$
# T = \int_{t_0}^{t_{max}} \frac{1}{C_{A}} \frac{\partial}{\partial z
    ↪ }\left(K_T \frac{\partial T}{\partial z}\right) dt
# $$
#
# Where T is temperature, z is depth, and t is time. In this study we
    ↪ will approximate several thing including
#
# - $K_T / C_A \approx \partial_tT/\partial^2_zT$
# - $f_S \approx -0.5\ln(T^{t+1}/T_*^{t})/D_t$

# best_rankin = {
#     "Score":np.inf,
#     "mse":0,
```

```
#       "r2":0,
#       "year":0,
#       "model":None
# }
#
# worst_rankin = {
#       "Score":-np.inf,
#       "mse":0,
#       "r2":0,
#       "year":0,
#       "model":None,
# }
#
# base_model = SE.RankinRegresson()
#
# for regi in nibio_id.keys():
#       for i in range(2014,2022):
#           # First we fetch region (regi), all stations (:), then
            ↪ relevant years ("2014":str(i)). Since we only look at one
            ↪  region at the time
#           # we remove the root group (shave_top_layer()), then we
            ↪ merge the years (merge_layer(level = 1), level 1 since
            ↪ level 0 would be the stations at this point)
#           # then make a list (flatten(), the default handeling is to
            ↪ put leafs in a list)
#
#           data = imputed_nibio_data[regi,:,"2014":str(i)].shave_top_
            ↪ layer().merge_layer(level = 1).flatten() # looks at all
            ↪ previus years including this year
#           test = imputed_nibio_data[regi,:,str(i+1)].shave_top_layer
            ↪ ().merge_layer(level = 1).flatten() # looks at the next
            ↪ year
#
#           data = [d.infer_objects(copy=False).fillna(0) for d in data
            ↪ ] # Removes nan in a quick manner
#           test = [d.infer_objects(copy=False).fillna(0) for d in test
            ↪ ] # but will be reviced.
#
#           model = copy.deepcopy(base_model)
#           overall_r2 = None
#           for d,t in zip(data,test): # fitting model with all
            ↪ stations
#               model.fit(d,d.loc[:,["TJM20"]]) # regions model
#               s_model = copy.deepcopy(base_model).fit(d,d.loc[:,["
            ↪ TJM20"]]) # Station model
#               if overall_r2 is not None:
#                   overall_r2 = 1-mediant(1-overall_r2, 1-r2_score(t["
            ↪ TJM20"].to_numpy(),s_model.predict(t)))
#               else:
```

```
#                   overall_r2 = r2_score(t["TJM20"].to_numpy(),s_model
        ↪ .predict(t))
#
#
#           print(regi,"from year 2014 to year",i,":\n",
#                 "\tMSE:",
#                 mae := [mean_squared_error(t["TJM20"].to_numpy(),
        ↪ model.predict(t)) for t in test],
#                 "\n\tR2:",
#                 r2 := [r2_score(t["TJM20"].to_numpy(),model.predict(t
        ↪ )) for t in test])
#           score = max(m/r for m,r in zip(mae,r2))
#           model_info = {
#                   "Score":score,
#                   "mse":mae,
#                   "r2": r2,
#                   "r2_spes": overall_r2,
#                   "year_max": i,
#                   "region": regi,
#                   "model": model
#               }
#           if score < best_rankin["Score"]:
#               best_rankin = model_info
#           elif score > worst_rankin["Score"]:
#               worst_rankin = model_info
#
# print(best_rankin)
# print(worst_rankin)

# ## LSTM
#
# This is a base model for testing ILSTM in the next section.

# In[ ]:

result_fitting = attempt_fitting(SE.KerasBiLSTM,{"input_shape":[24*n
        ↪ for n in range(1,7)],"lstm_units":[2*k for k in range
        ↪ (20,25)],"epochs":[4*n for n in range(30,50)]},imputed_
        ↪ nibio_data)

print("best:",result_fitting["best"])
print("worst:",result_fitting["worst"])
print("median:",sorted(result_fitting["all"],key = lambda x: x["Score
        ↪ "])[int(len(result_fitting["all"])/2)])

# In[10]:


#
```

```
        ↪ ──────────────────────────────────────────────
        ↪
all_data_daily = data_t.set_index("Time").resample("D").mean().dropna
        ↪ ().reset_index()

p_data = F_plauborg(all_data_daily)
ridge = LinearRegression().fit(p_data[50:], all_data_daily.iloc
        ↪ [50:,[-1]])
y_pred = ridge.predict(p_data[50:])
display = PredictionErrorDisplay(y_true=all_data_daily.iloc
        ↪ [50:,[-1]], y_pred=y_pred)
display.plot(kind = "actual_vs_predicted",scatter_kwargs = {
    "c": np.linspace(0,1,num = all_data_daily.iloc[50:,[-1]].shape
            ↪ [0]),
    "color": None
})
plt.show()

all_data_daily = all_data_daily.reset_index().loc[50:]

Y = pd.DataFrame(
    zip(all_data_daily["Time"].to_numpy().tolist(), y_pred.flatten())
            ↪ ,
    columns=["Time","Y_pred"])

show_plot([all_data_daily.loc[:,["Time","TJM20"]],Y, all_data_daily.
        ↪ loc[:,["Time","RR"]]],{1:{"alpha":0.5}} )
plt.legend(["Y","Y_pred"])
plt.ylim(-5,25)
plt.ylabel("°C")
plt.show()


# # ILSTM training
#
# Here we will be training a version of LSTM

# In[30]:


import copy

def ILSTM_train(raw_data, target_label,total_epoch = 50,hidden_size
        ↪ =16,lerningrate=1e-3, lead_time=1, seq_length=24, batch_
        ↪ size=16):
    data,scaler,scaler1 = ILSTM.nibio_data_transform(raw_data, target
            ↪ _label)
    data = scaler1.transform(data)
```

```
# TODO: Generate the tensor for lstm model

[data_x, data_y,data_z] = ILSTM.LSTMDataGenerator(data, lead_time
    ↪ , batch_size, seq_length)

    # concat all variables.
# TODO: Flexible valid split
data_train_x=data_x[:int((data_x.shape[0])-400*24)]
data_train_y = data_y[:int(data_x.shape[0]-400*24)]

train_data = Data.TensorDataset(data_train_x, data_train_y)
train_loader = Data.DataLoader(
     dataset=train_data,
     batch_size=batch_size,
     shuffle=False,
     num_workers=0
)

data_valid_x=data_x[int(data_x.shape[0]-400*24):int(data_x.shape
    ↪ [0]-365*24)] # -> trener 35 dager
data_valid_y=data_y[int(data_x.shape[0]-400*24):int(data_x.shape
    ↪ [0]-365*24)] # -> tester 35 dager
data_test_x=data_x[int(data_x.shape[0]-365*24):int(1.0 * data_x.
    ↪ shape[0])] # -> validerer på resterende
data_testd_z=data_z[int(data_x.shape[0]-365*24):int(1.0 * data_x.
    ↪ shape[0])] # -> stat på rest

# TODO: Flexible input shapes and optimizer
# IMVTensorLSTM,IMVFullLSTM
model = ILSTM.ILSTM_SV(data_x.shape[2],data_x.shape[1], 1, hidden
    ↪ _size).cuda()
# TODO: Trian LSTM based on the training and validation sets
model,predicor_import,temporal_import=ILSTM.train_lstm(model,
    ↪ lerningrate,total_epoch,train_loader,data_valid_x,
    ↪ data_valid_y,"./saved_models/lstm_1d.h5")

# TODO: Create predictions based on the test sets
pred, mulit_FV_aten, predicor_import,temporal_import = ILSTM.
    ↪ create_predictions(model, data_test_x,scaler)
# TODO: Computer score of R2 and RMSE

data_testd_z=data_testd_z.reshape(-1,1)
data_testd_z=data_testd_z.cpu()
data_testd_z=data_testd_z.detach().numpy()
# Unnormalize
data_testd_z=scaler.inverse_transform(data_testd_z)
ILSTM.compute_rmse_r2(data_testd_z,pred,modelname)

print(pred)
```

```
# Need to transform the data first to fit the model.

# In[26]:


def datetime2string(x):
    x["Time"] = x["Time"].apply(lambda y: y.strftime("%Y-%m-%d %X"))
    return x
station_data = imputed_nibio_data.data_transform(datetime2string).
    ↪ merge_layer(level = 1)


# In[31]:


ILSTM_train(copy.deepcopy(station_data["11"]),"TJM20",batch_size = 8,
    ↪ total_epoch = 20)
```

B Plots

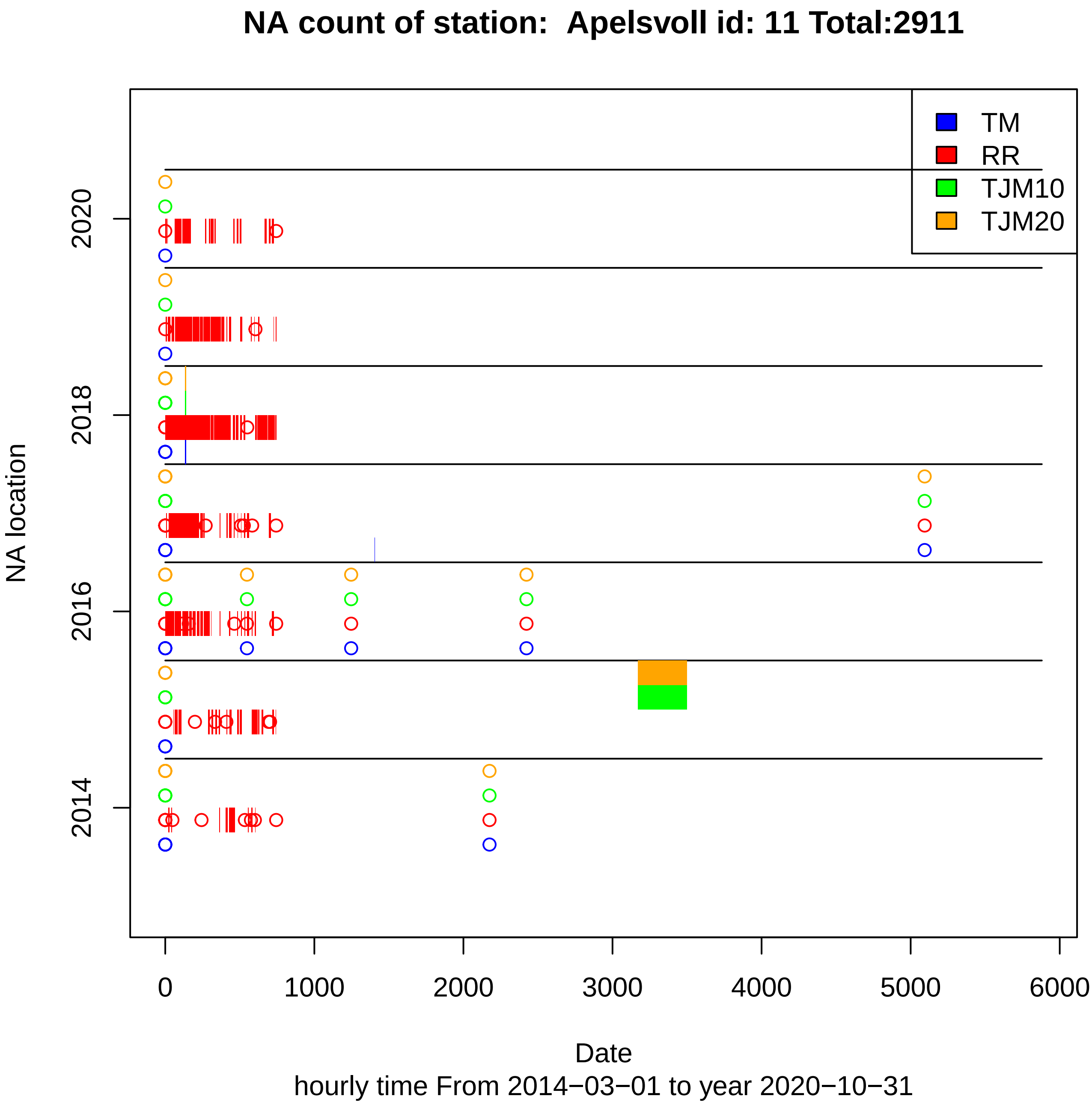

Figure 4: Station nr 11missing value plot

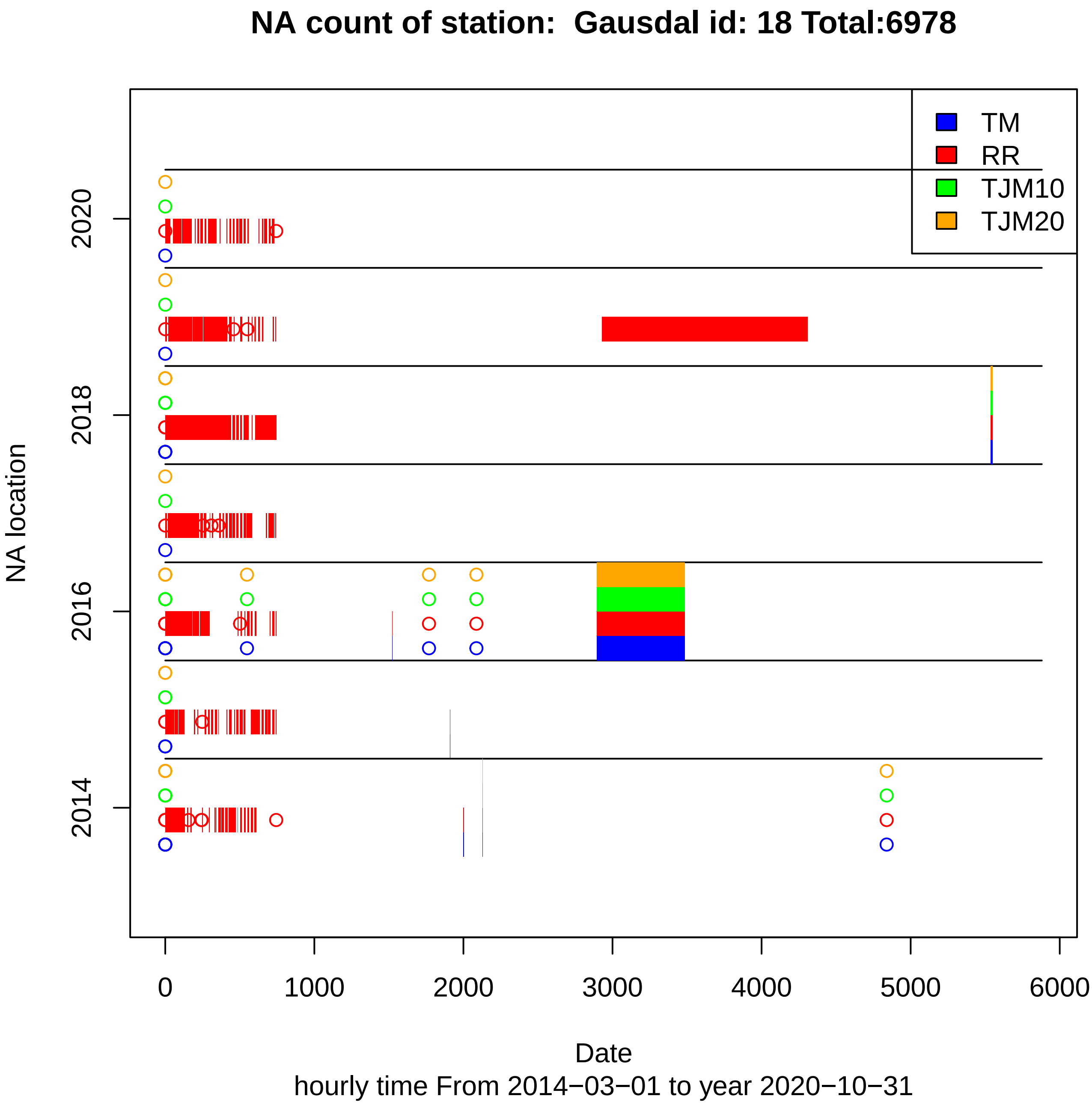

Figure 5: Station nr 18missing value plot

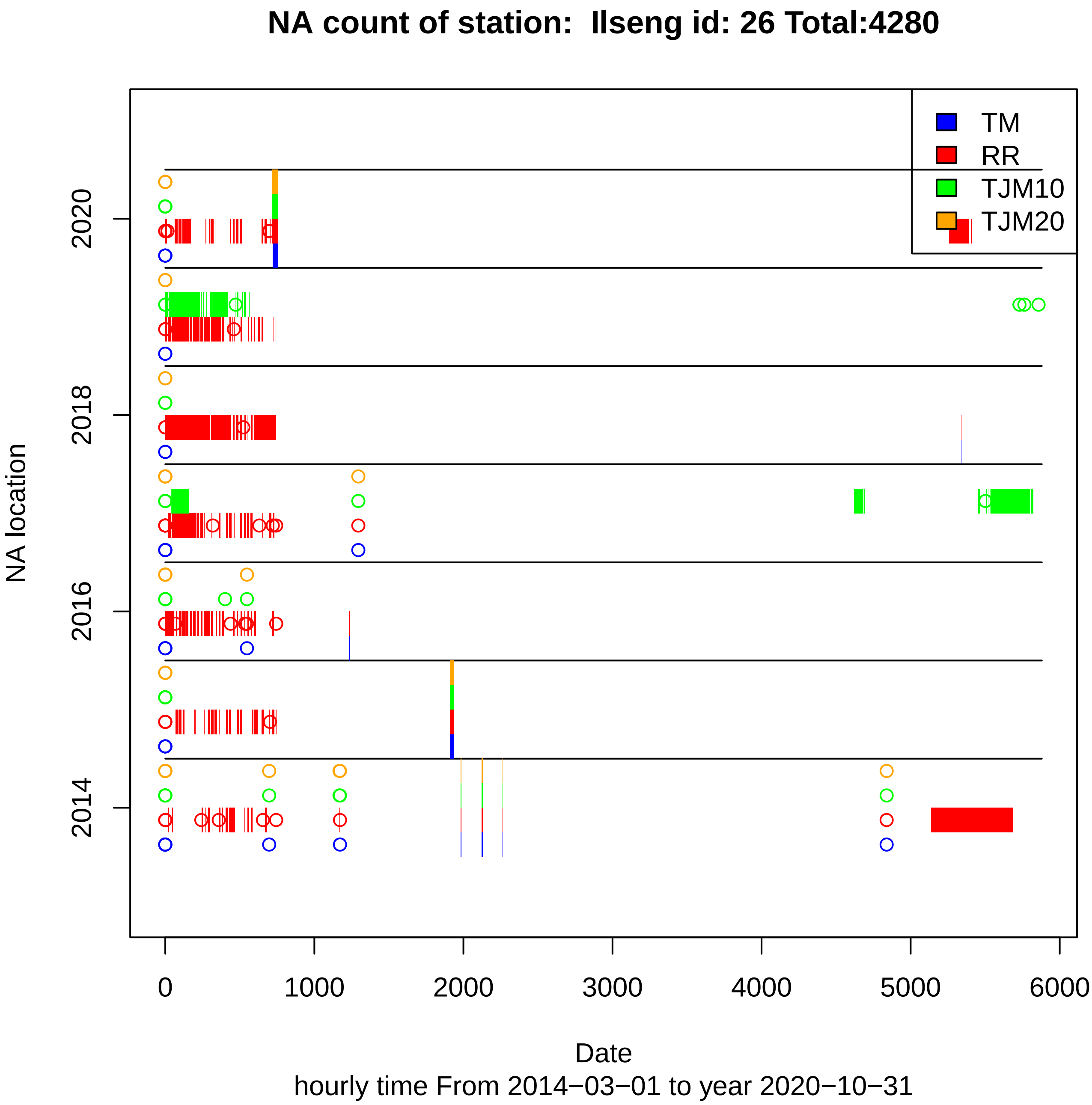

Figure 6: Station nr 26missing value plot

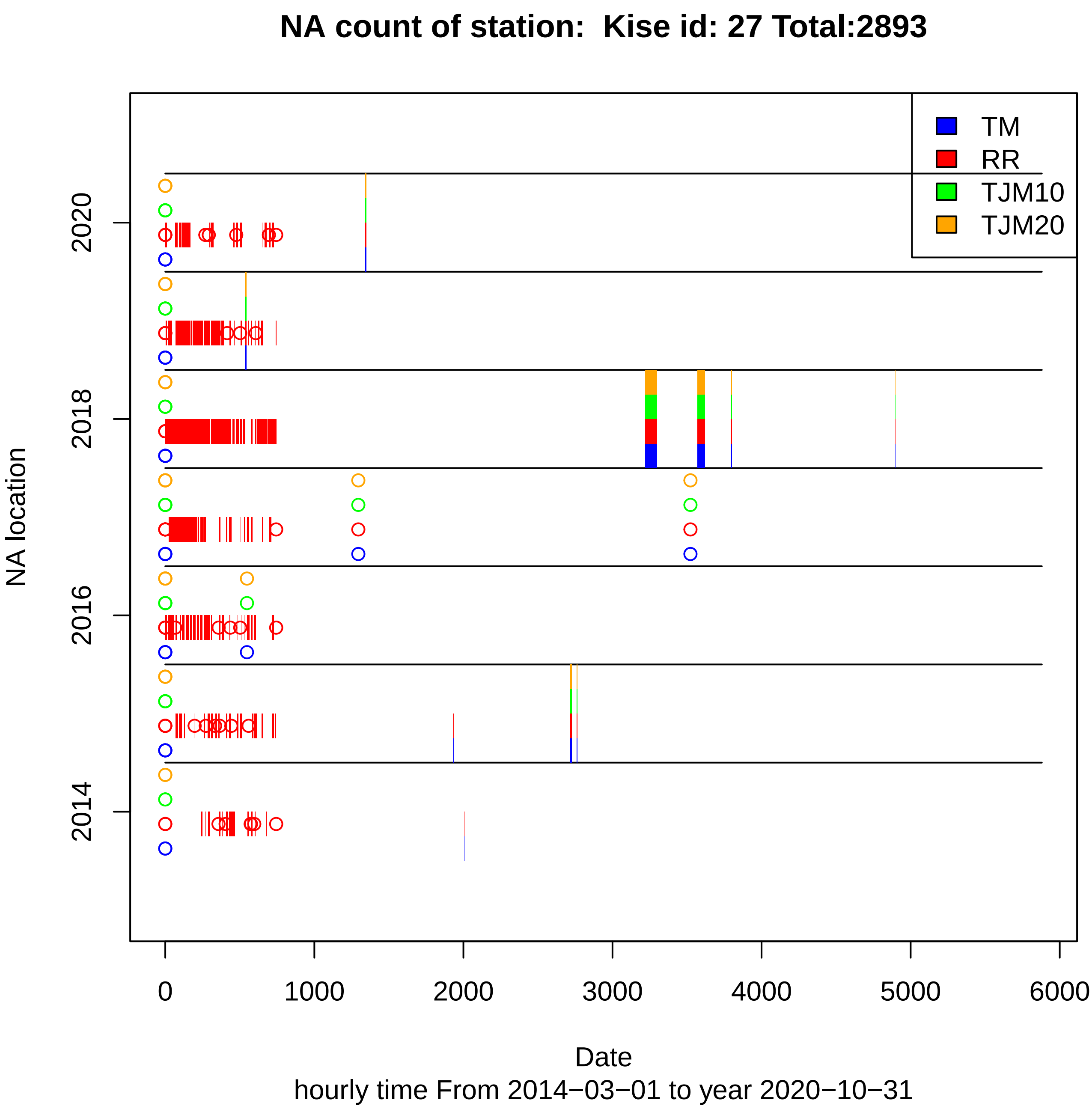

Figure 7: Station nr 27missing value plot

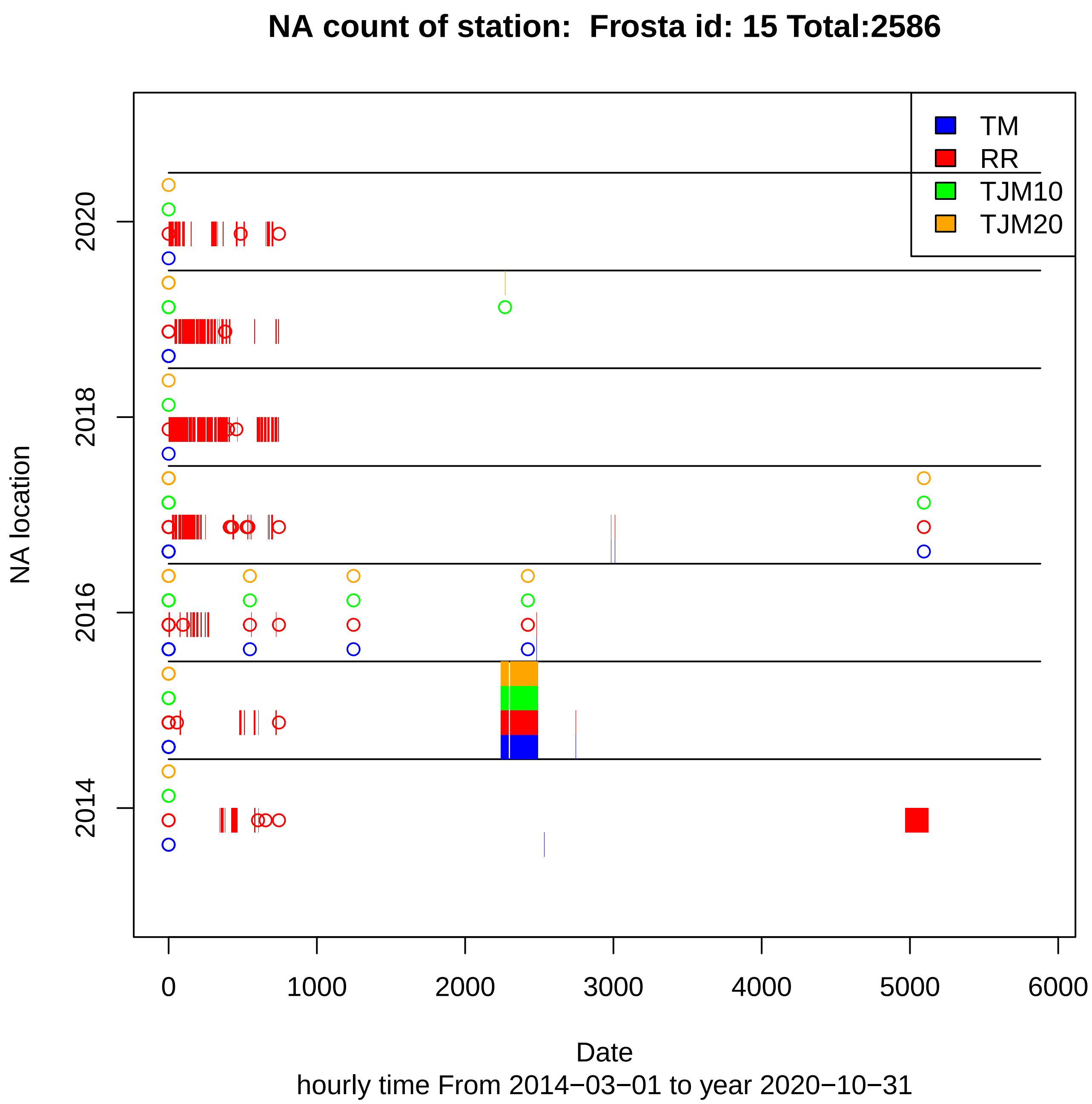

Figure 8: Station nr 15missing value plot

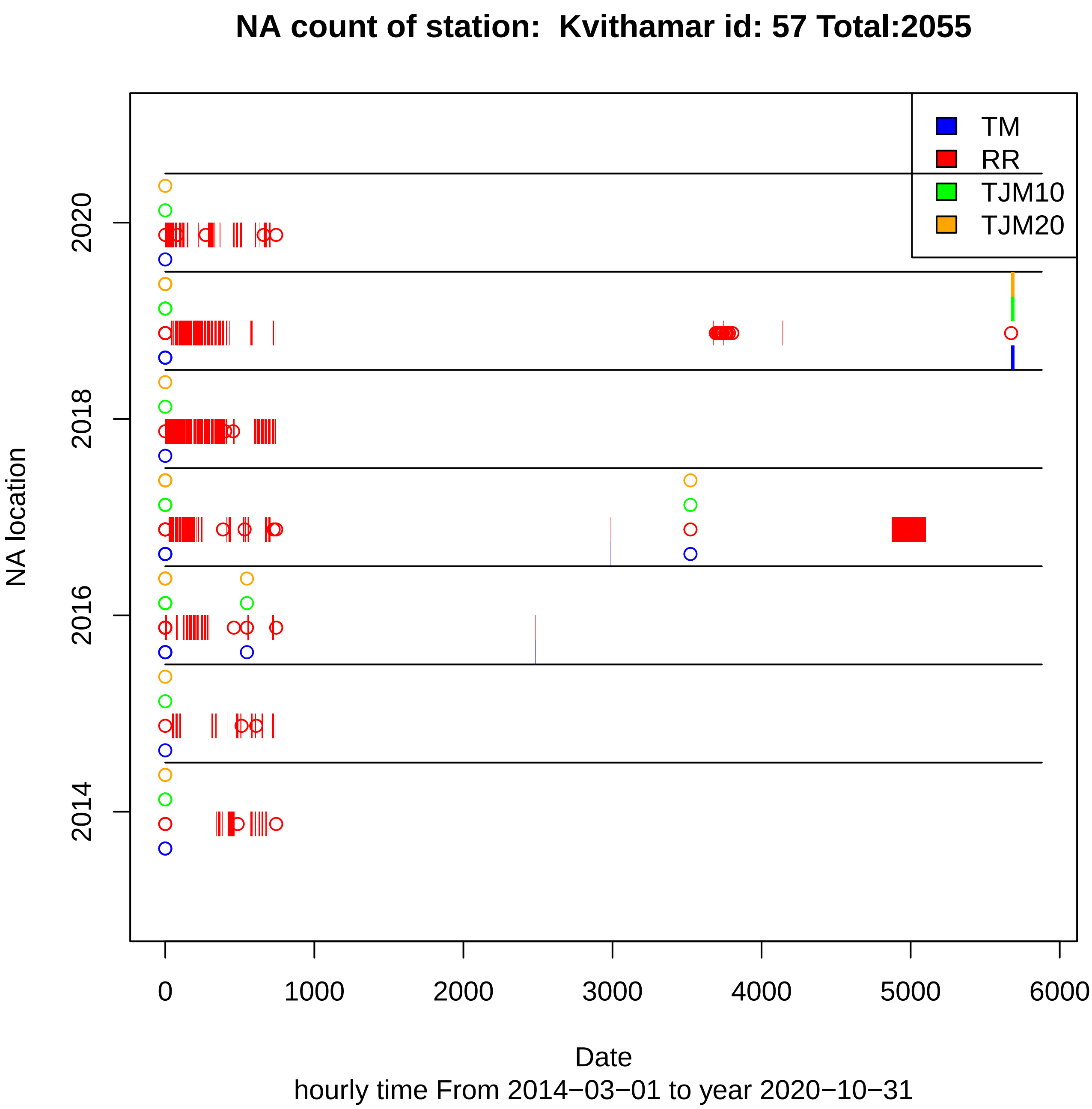

Figure 9: Station nr 57missing value plot

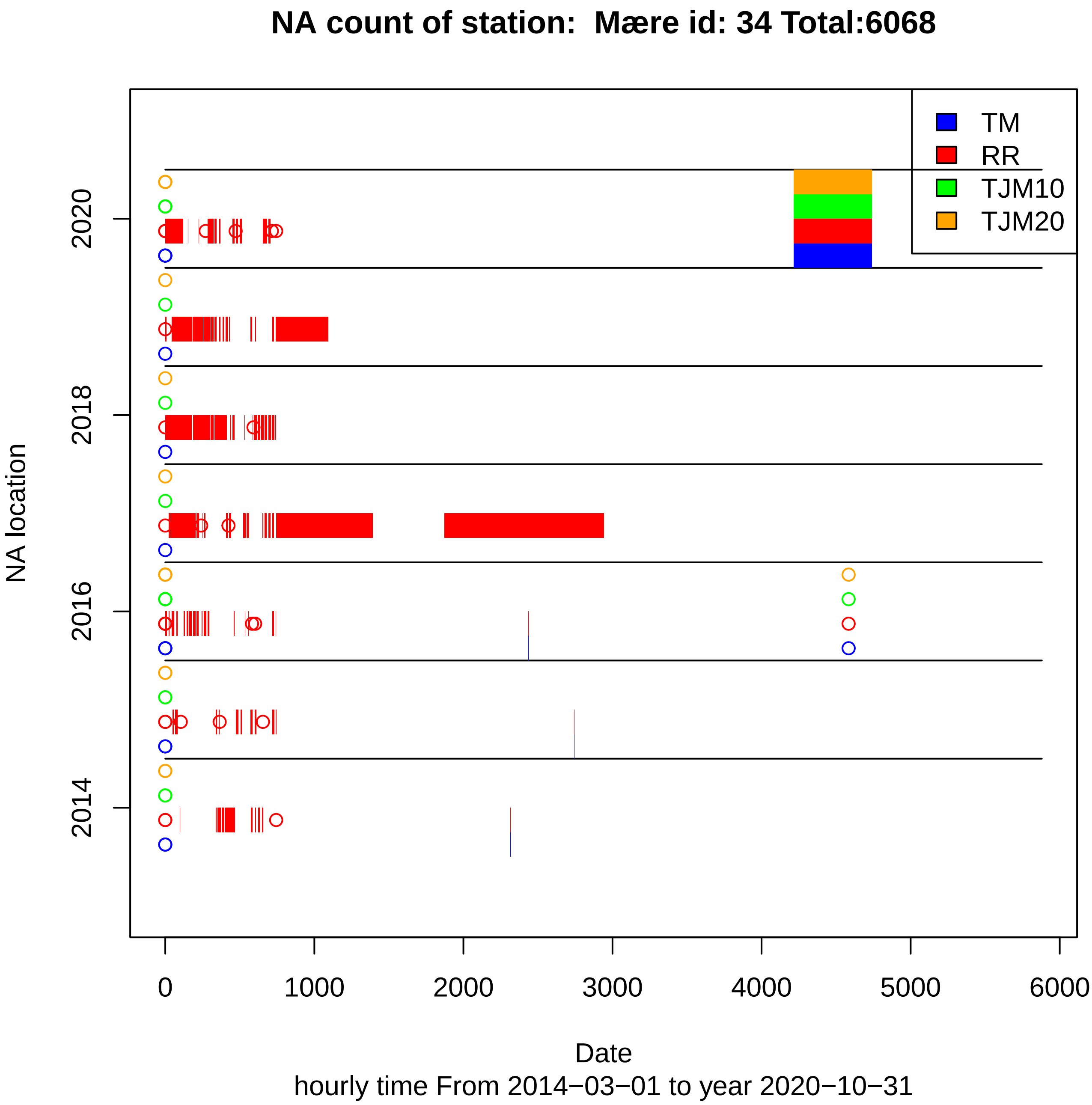

Figure 10: Station nr 34missing value plot

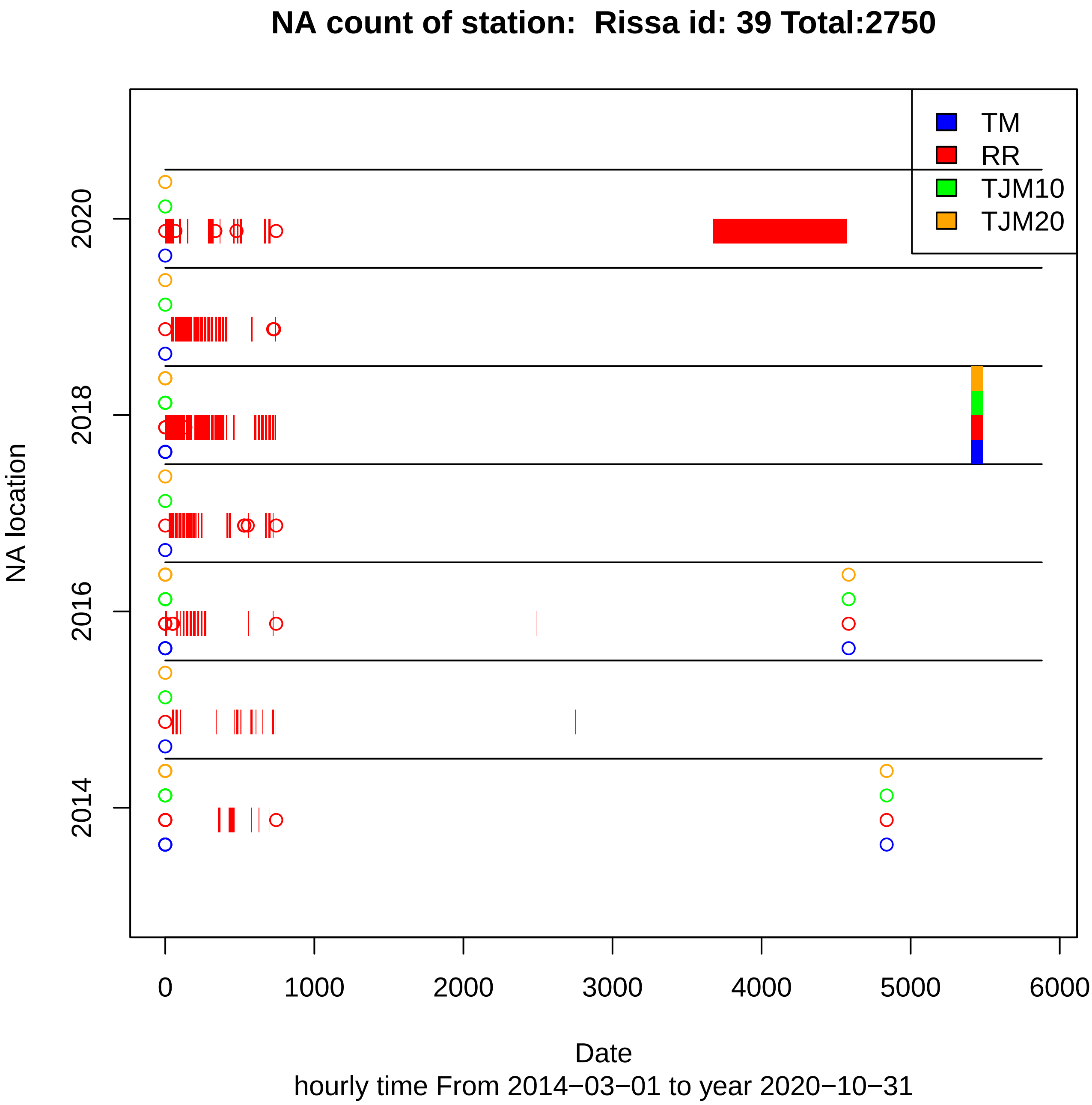

Figure 11: Station nr 39missing value plot

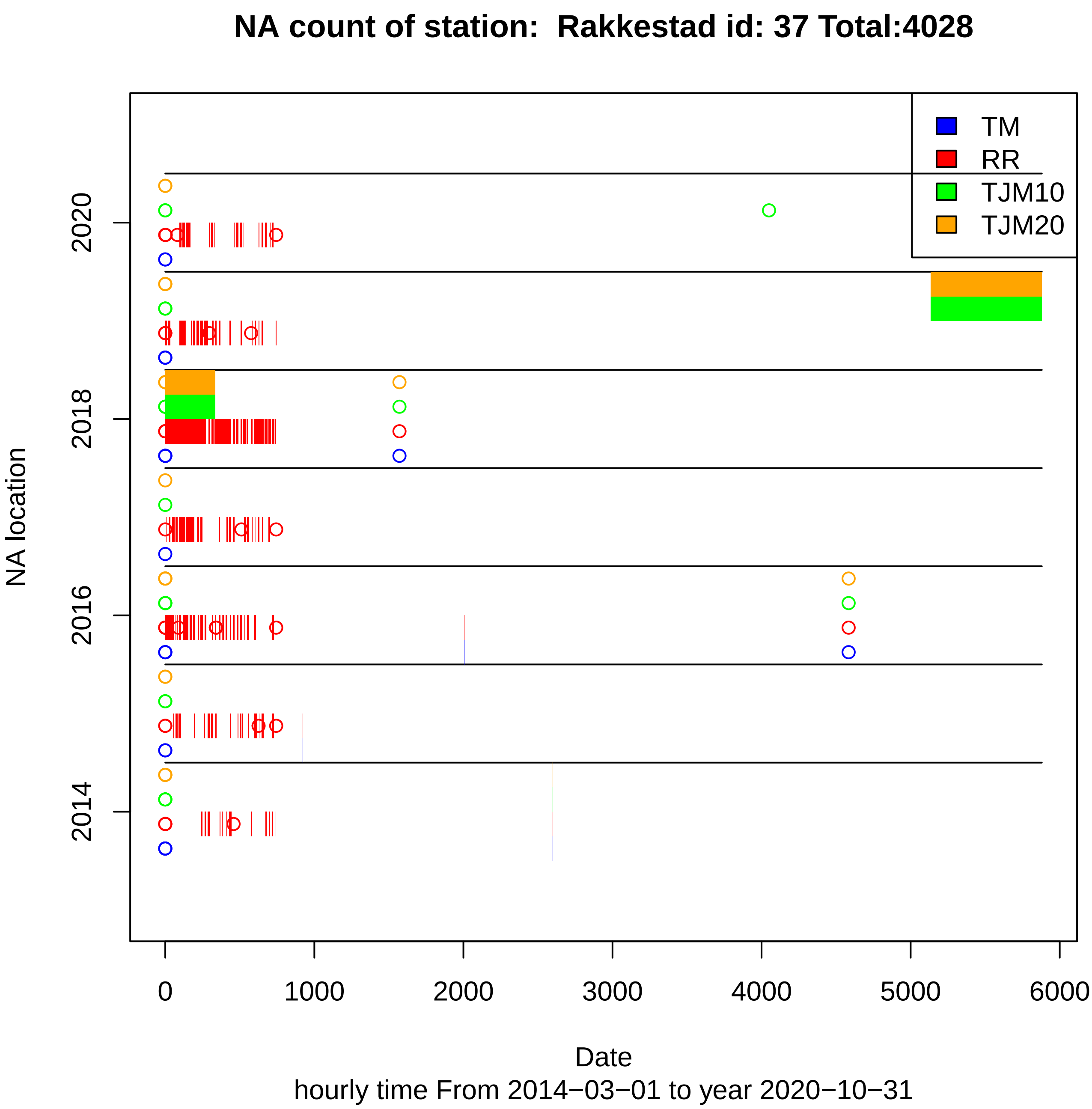

Figure 12: Station nr 37missing value plot

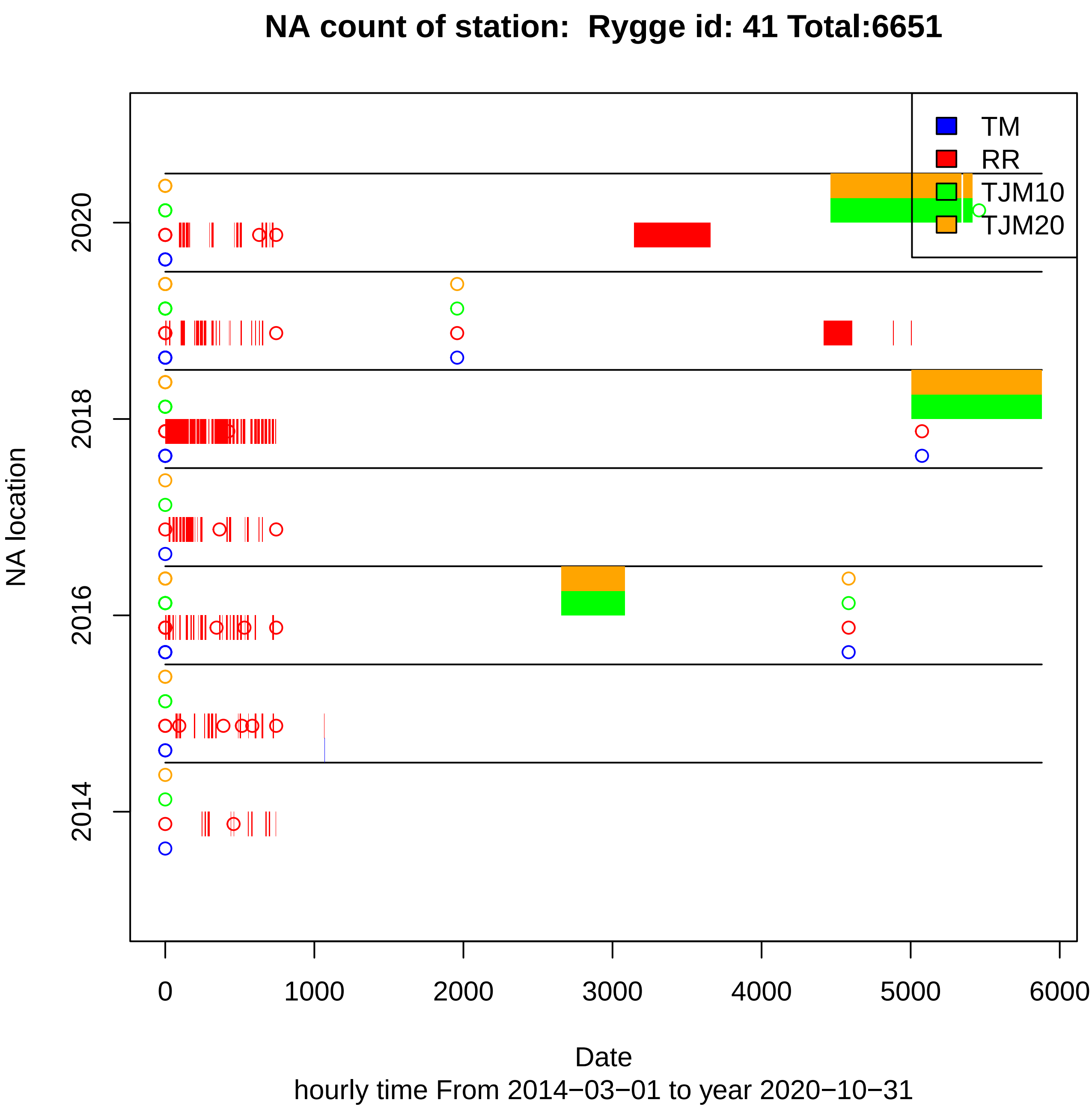

Figure 13: Station nr 41missing value plot

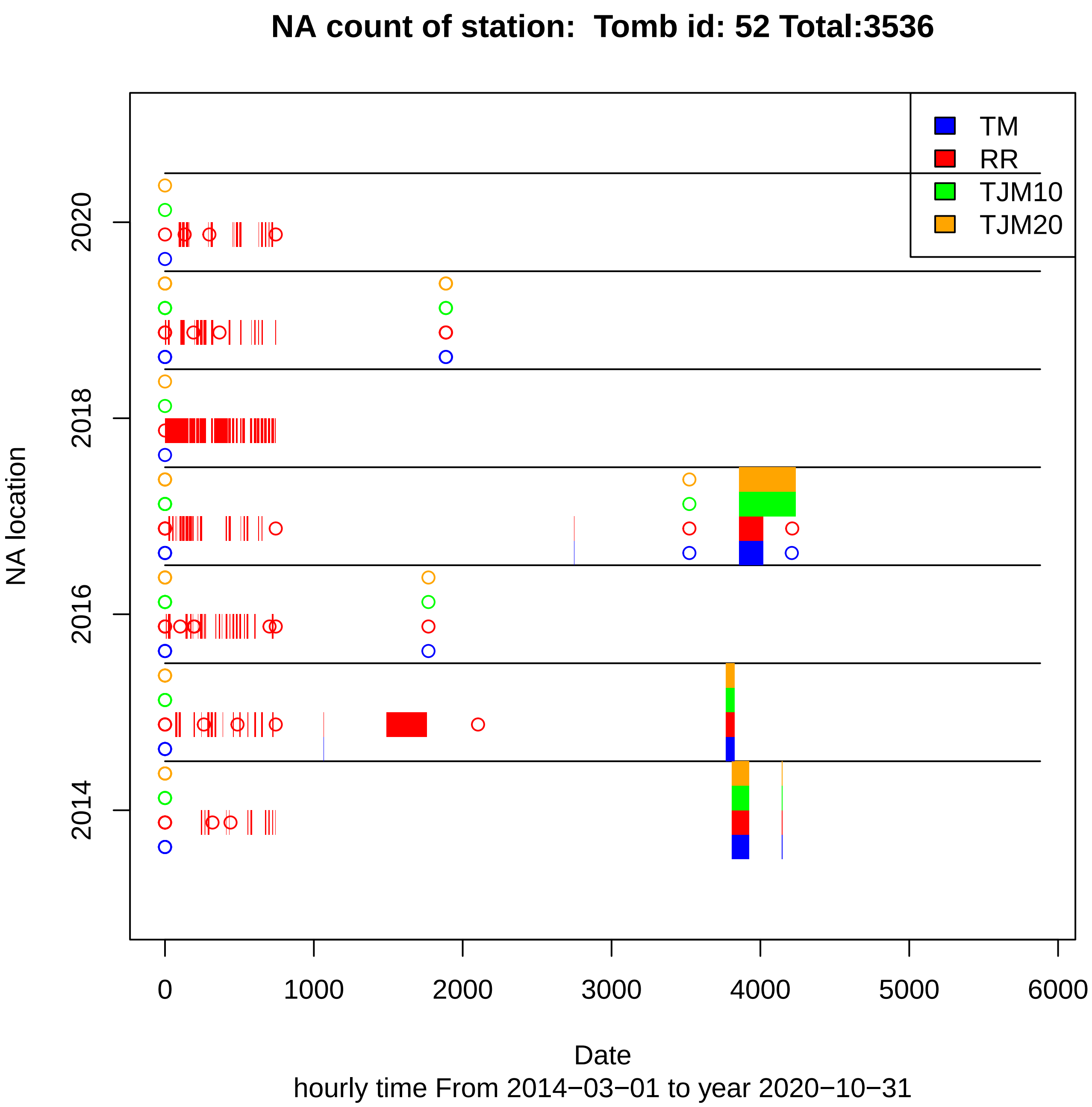

Figure 14: Station nr 52missing value plot

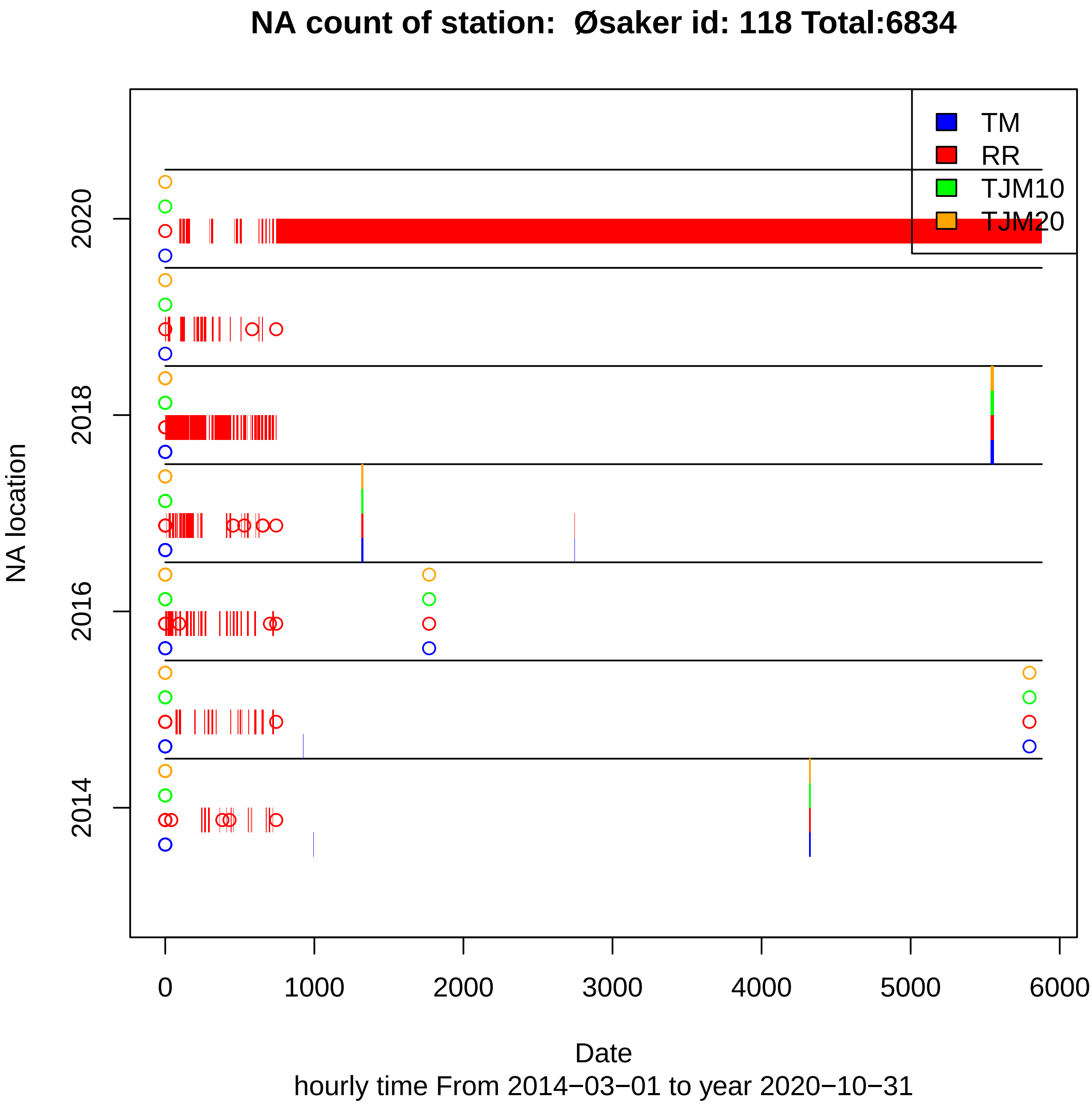

Figure 15: Station nr 118missing value plot

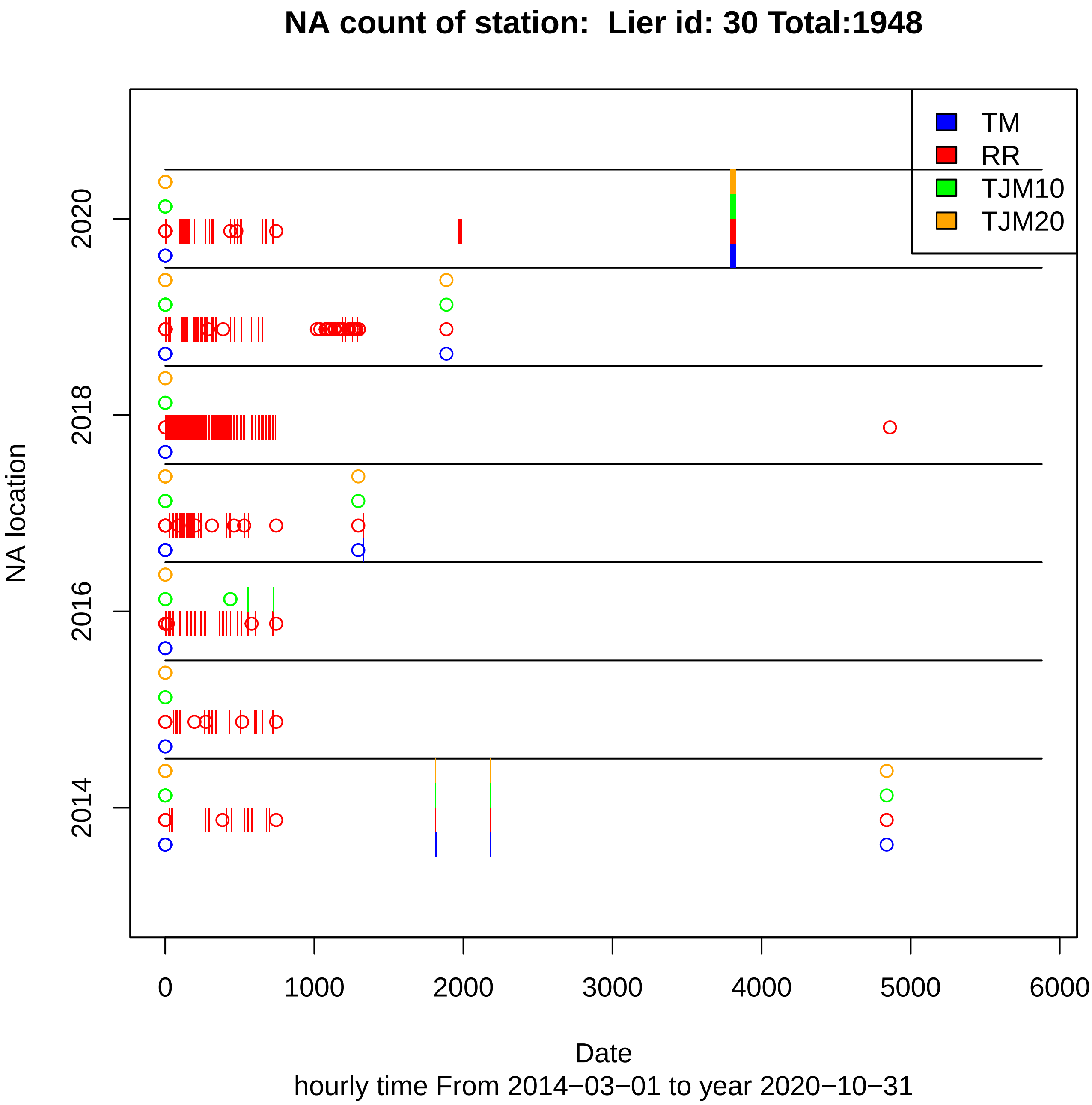

Figure 16: Station nr 30missing value plot

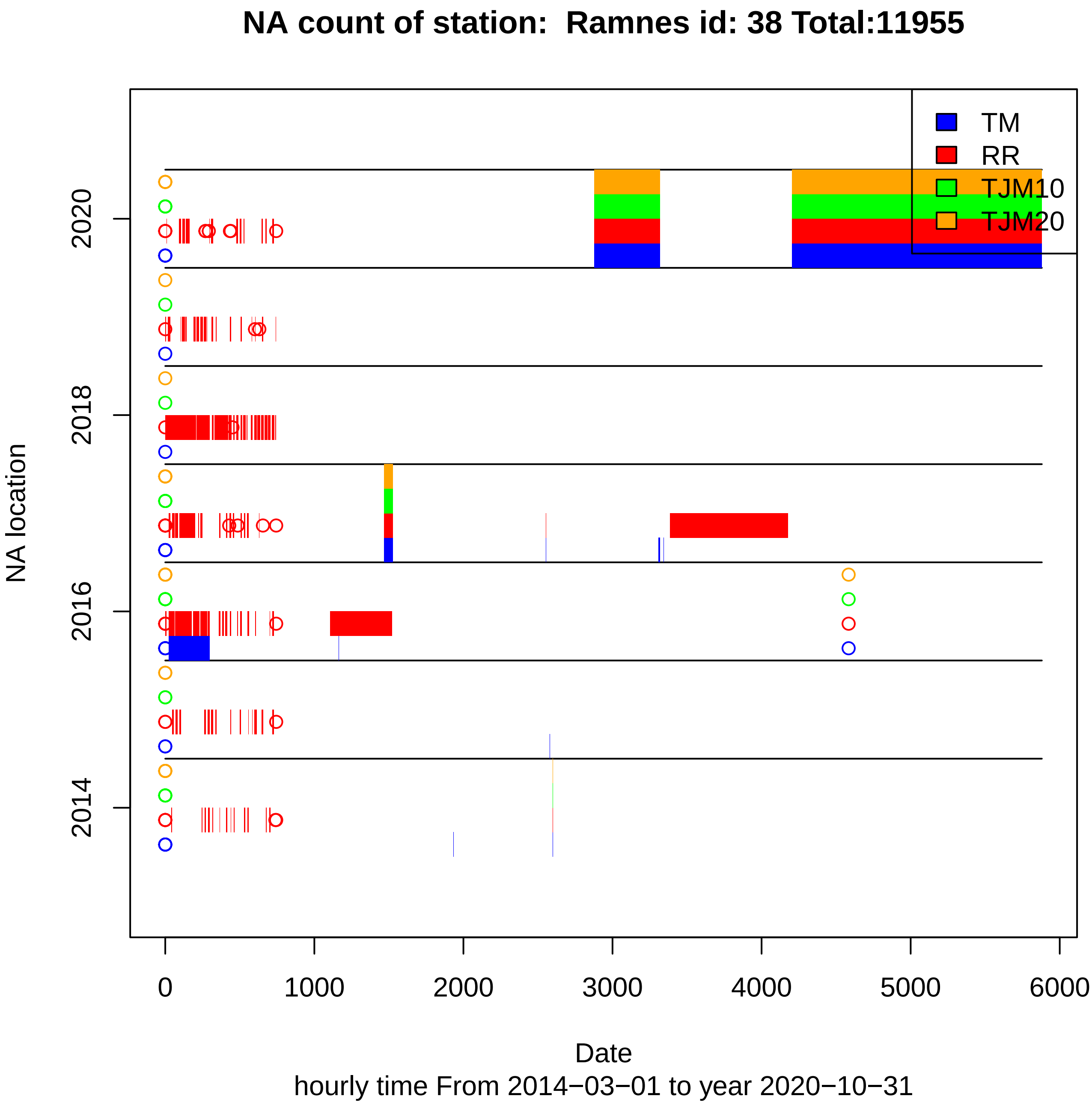

Figure 17: Station nr 38missing value plot

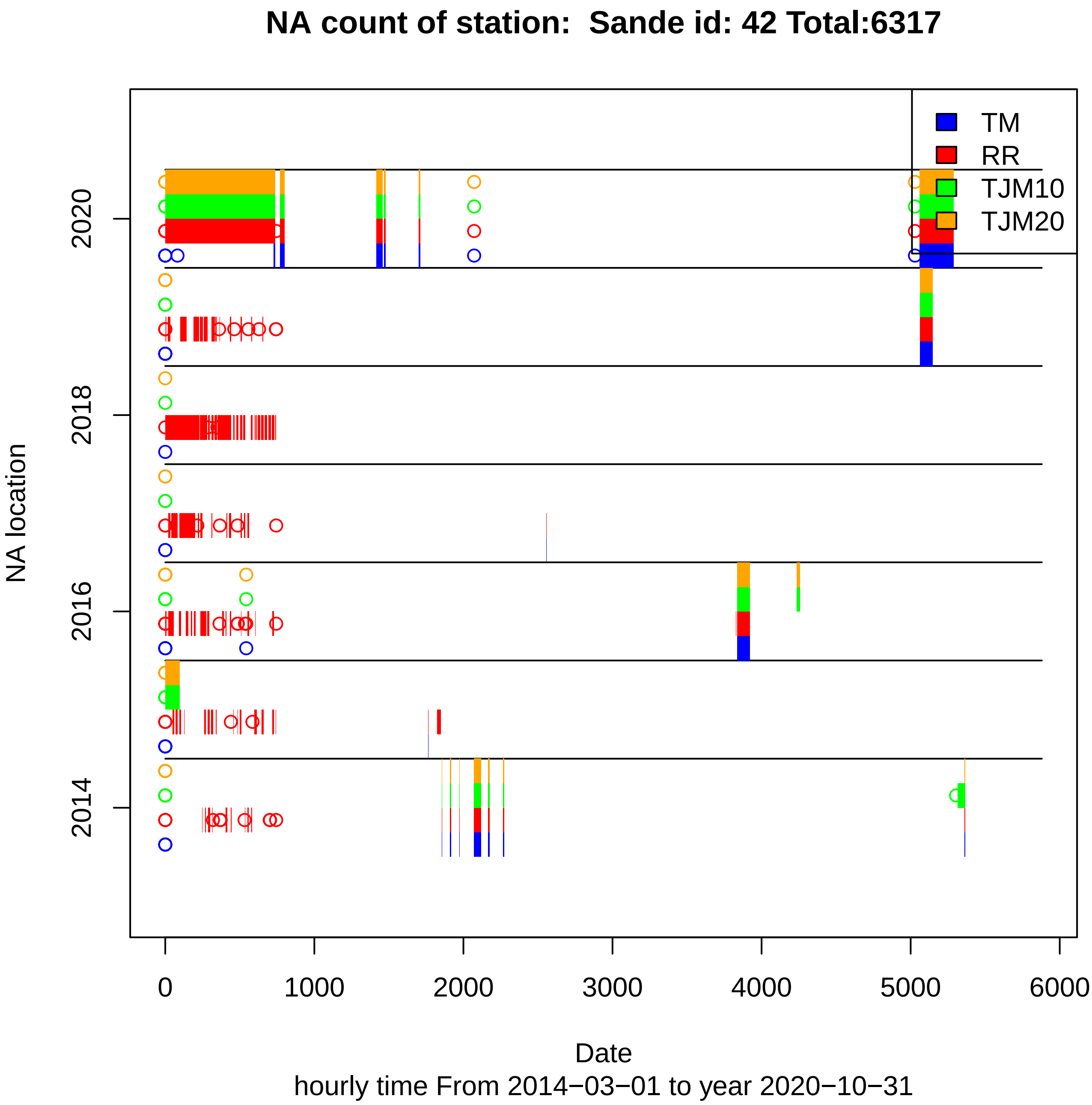

Figure 18: Station nr 42missing value plot

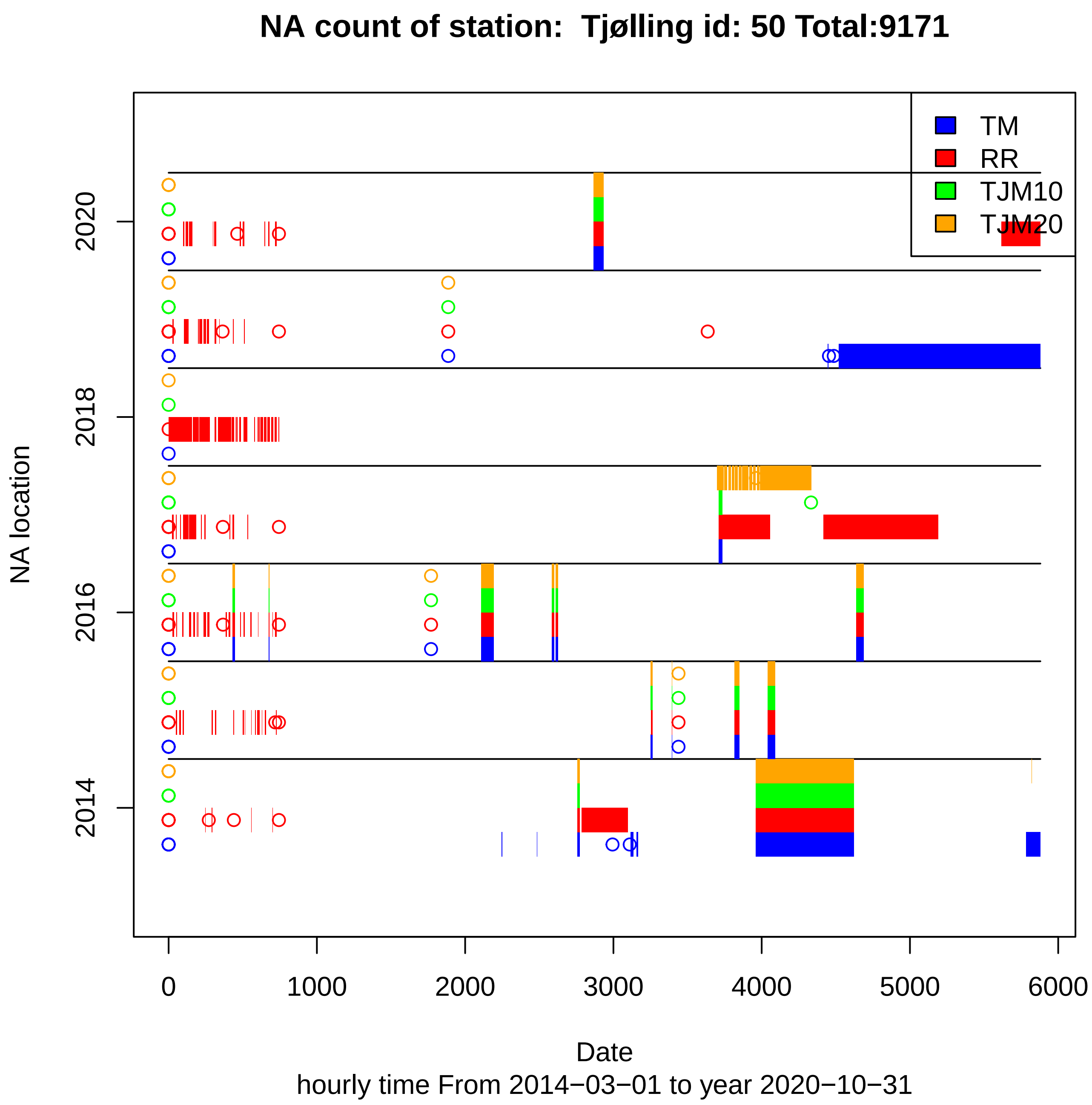

Figure 19: Station nr 50missing value plot

C Tables

Table

Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway