Resultat fra arbejdsgruppe vedr. varmestress i 2021	Ansvarlig	THA
	Oprettet	07-12-2021
Projekt: 5517, Datadrevet management i mælkeproduktionen	Side	1 af 6

STØTTET AF
mælkeafgiftsfonden

Varmestress

Varmestress er en tilstand som opstår når koen er i et miljø med høj temperatur og/eller høj luftfugtighed. Landmanden kan via forskellige tiltag omkring ventilation og overbrusning undgå problemer. Formålet med indsatsen, var at give mælkeproducenterne et værktøj til at forudsige varmestress og dermed mulighed for at sætte ind med afværgeforanstaltninger. Det kunne være en prognose for risikoen for varmestress, der bygger på vejrudsigten samt den historik der tidligere har været omkring varmestress på hans ejendom.

Arbejdsgruppe

JNI, FOGH, THA, HNM, PRA, PSP & KFOG

Faglig baggrund

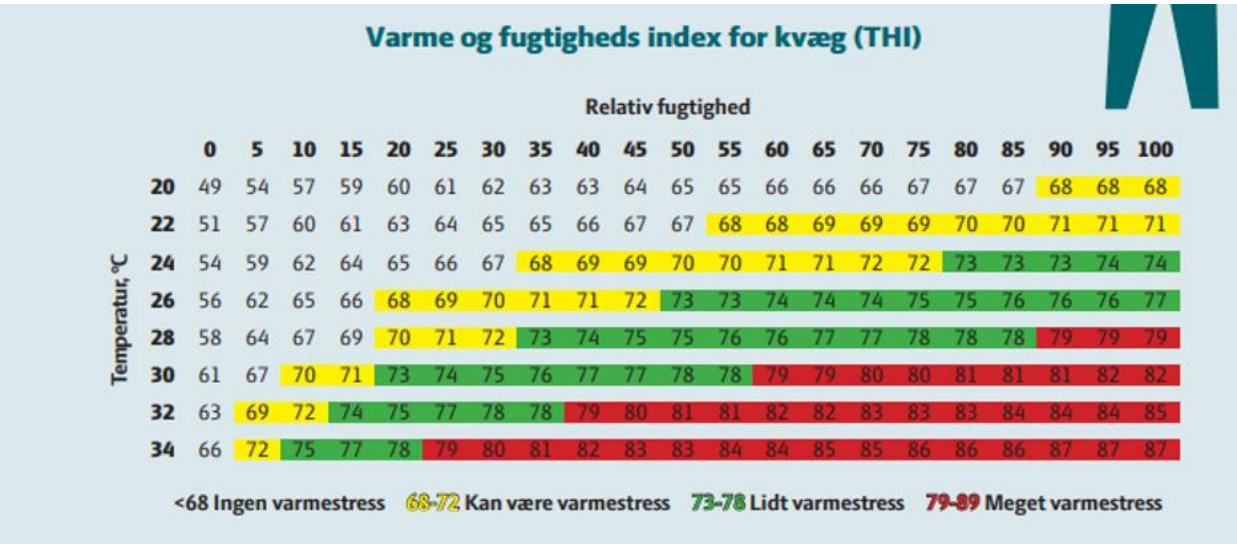
Varmestress påvirker køerne negativt på mange parametre: Ædelysten falder, ydelsen falder 15 – 40 %, immunforsvaret svækkes, og risikoen for yverbetændelse og klovlidelser stiger. Ved længerevarende varmestress forringes også reproduktionen i form af bl.a. svagere brunst og lavere drægtighedsprocent.

Varmestress opstår ved høj temperatur og luftfugtighed. En anerkendt model til sammenvejning af temperatur og luftfugtighed THI-indekset. I arbejdet med THI er anvendt denne definition:

THI = T − (0.55 − 0.55 ×RH) × (T − 58),
hvor T er staldtemperatur i °F, og RH er relativ fugtighed udtrykt som decimal.

Definitionen er fra hentet fra Cook et al., 2007 (J.DairySci. 90:1674-16825). hvis kilde er "*National Oceanic and Atmospheric Administration (NOAA). 1976. Livestock hot weather stress. US Dept. Commerce, Natl. Weather Serv. Central Reg. Operations Manual Lett. C31-76. NOAA, Kansas City, MO*."

Respons på THI er bedriftsafhængigter og kan opstå fra et THI på 68 og opefter. Figur 1 viser sammenhængen mellem temperatur, relativ fugtighed for risiko fra varmestress. Det ses at det ikke kun er ved høje temperaturer, men også kan opstå ved temperaturer på 20^{o}C, hvis fugtigheden er høj. Figur 1



Figur 1. Illustration af THI og mulige påvirkning af mælkeproduktionen.

Der er tidligere set en sammenhæng mellem THI og mælkeindvejning samt THI og gns. tankcelletal på landsniveau. Data som indikerer at der også er en sammenhæng lokalt (se Figur 2**Fejl! Henvisnings-kilde ikke fundet.**).

Figur 2 Grafisk visning af THI, mælkeindvejning og celletal for 2018 og 2019 (alt vist på landsniveau). Det ses at der i sommeren 2018 og 2019 er et fald i mælkeindvejningen og efterfølgende en stigning i celletal, når THI stiger til omkring 70.

Identificerede opgaver

Der er i arbejdsgruppen identificeret og løst en række opgaver. Det konkrete arbejde er beskrevet i bilag A og bilag B. Arbejdet har omfattet:

- Træk af UTM position på alle mælkeleverende ejendomme
- Træk af temperatur og relativ fugtighed fra ejendommenes position
- Beregning af THI for ejendommene
- Grafisk præsentation af THI data

- Træk af data vedr. mælkeproduktion fra de mælkeleverende ejendomme samt beregning af 2 dages rullende gennemsnit på mælkeydelse
- Korrelationsanalyse for at identificerer ejendomme høj sammenhæng mellem THI og påvirkning af mælkeydelse og celletal efter hhv. 2, 4 og optil 20 dage efter THI værdien

Resultater fra 2021 – kort opsummering

Det gennemført arbejde viser at det er:

- Muligt at beregne THI for positionen for den enkelte stald og vejrudsigten fra DMI. Data fra DMI kommer for 2 timeres perioder. Dette er i modellen summeret over døgnet.
- Det er muligt at finde besætninger der har en tydelig korrelation mellem THI større end 72 og celletal 10 – 14 dage efter besætningen var udsat for det høje THI.

Diskussion og videre arbejde

Analysen viser, at der kan påvises en sammenhæng mellem varmestress og celletal og at denne er meget afhængig af bedriften. Det kan man tolke, som at management betyder meget for det respons der ses efter at dyrene, er blevet udsat for perioder med højt THI. Nogle besætningsejere har rutiner og staldindretning som kan afhjælpe problemstillingen. Derfor kan de være svært at vedligeholde AI modeller, som kan give bedriftsspecifikke alarmer. Hvis fx en besætningsejer baserer sit management på modellen, vil gentræning af modellen på nye data, vise at besætningen ikke er påvirket af varmestress. Derved vil bedriften ikke få alarmer ved opdatering af modellen.

Anbefaling vedr. prototype og implementering

På basis af arbejdet anbefales det at afprøve en prototype med beregning af THI på baggrund af ejendoms lokation og vejrudsigten fra DMI. THI kan så omsættes til en alarm der udtrykker risikoen for varmestress fx vist i 3 niveauer (lav, mellem og høj). Yderligere arbejde med korrelationsanalysen kan give grundlag for anbefalinger og handlingsanvisning for den enkelte bedrift. Fx ”data fra 2018, 2019 og 2020 viser at din bedrift har en stor sammenhæng mellem THI og celletal op til 14 dage efter perioden med højt THI.” Teksten kan suppleres med konkrete anbefalinger.

Bilag A

Notat udarbejdet i forbindelse med træk af mælkedata til brug for analyserne

Bilag B

Noter og programmer fra dataklargøring og datamodellering. Noterne er en anonymiseret version af den Jupyter Notebook som er skrevet undervejs i processen.

Bilag A Beskrivelse af mælkedata, tabel r5ajni.PROJ_DDM_MLKtilTHI

Data med celletal og mælkeydelse siden foråret 2018 og frem til juni 2021 for alle mælkeleverende ejendomme ligger i datatabel på Kvægdatabasen: **r5ajni.PROJ_DDM_MLKtilTHI**
I alt 2.058.716 datalinjer med følgende kolonner:

Kolonne	Type	Null	Beskrivelse	Eksempel
CHRNR	integer	not null	Ejendomsnr. = CHRnr.	32345
MEJNR	integer	not null	Mejerinr., hvortil der er leveret mælk	1
LEVNR	integer	not null	Leverandørnummer hos mejeriet	32345
DATO	datetime	not null	Dato for indvejning	01-07-2018
CELLETAL	integer	null	Celletal	225.000
KGMLK	integer	null	Indvejet mælk i kg	13.565
FEDT	FLOAT	null	Fedt procent i mælken	4,42
PROTEIN	FLOAT	null	Protein procent i mælken	3,65
KGEKM	FLOAT	null	Kg EKM = EnergiKorrigeret Mælk	14.512,649

KG EKM

Kg EKM er beregnet efter standardformel ud fra indvejet mælkemængde sammen med fedt og protein:

3. *Bilag 2, tabel 3, 2. afsnit,* affattes således:
»Korrektionsformlerne for malkekøer type 1 er baseret på kg energikorrigeret mælk (EKM), medens de tilsvarende type 2 formler er baseret på produceret kg mælk. Har man ikke tilgængelige tal fra ydelseskontrollen, kan kg produceret mælk beregnes som kg leveret mælk x 1,055. Til beregning af kg EKM bruges følgende formel: Kg EKM = kg produceret mælk × (383×fedt% + 242×protein% + 783,2)/3140, hvor fedt% og protein% er bestemt ved kemiske analyser.«

Periodeafgrænsning

Jeg har valgt at hente data fra hele perioden: forår 2018 til juni 2021. Man kan så selv indsnævre til sommermånederne 2018 - 2020.

Forskel på mejerier, datafrekvens osv.

Ofte er der indvejning af mælk hver anden dag. Men for måske en tredjedel af leverandørerne også *hver* dag. For de fleste mejerier er der måling af fedt, protein og celletal i forbindelse med hver afhentning. Men det er ikke tilfældet for alle mejerier. Og selv om det er standarden for et mejeri, kan der være enkelte leveringer, hvor der er faldet et prøvesvar væk.
Såfremt der ikke er fedt- eller protein-måling, tillader man normalt at bruge målinger fra seneste kendte prøve. I denne datasammenstilling har jeg dog *ikke* forsøgt at flytte prøveresultater til næste indvejning, hvis der var denne mangel. Men det kan man så gøre selv.
De pæneste data - og de fleste - vil man få, hvis man tager data fra mejnr = 1, Arla. Det vil dreje sig om op mod 90% af alle indvejningsdata. Men også for mejerinumre, som er op til og med 2515 (de såkaldte MAIS-mejerier), vil der være nærmest regelmæssige data. Blandt disse mejerier kan der dog være nogen, hvor vi kun har ugentlige data om indvejning og/eller ugentlige data på fedt, protein og celletal.
Bemærk, at der kan være uregelmæssigheder i frekvensen af data. Også for Arla-leverandører. Selv om der leveres mælk hver anden dag, kan dagen godt skifte, hvis mejeriet omlægger sine afhentningsruter. Så vil der pludseligt kun gå én dag mellem to afhentninger. (Lovmæssigt skal der afhentes senest efter 2 døgn, og der bør derfor ikke kunne gå tre dage.)
En anden slags uregelmæssighed kan være, at mejeriet kun afhenter en del af mælketankens indhold den ene dag, og så afhenter hele mælketankens indhold den næste dag. Det er oftest for store mælkeprocenter, at det kan ske, og mejeriet bruger dette til at fylde deres tankbiler op, inden de kører tilbage til mejeriet.

En tredje uregelmæssighed kan være, at der afhentes lige omkring midnat. Men én nat kan det være lige før midnat og den næsten nat, kan det være lige efter. På den måde vil det se ud som om der er dage, hvor der ikke afhentes og andre dage, hvor der afhentes dobbelt.
En fjerde mulig uregelmæssighed kan være AMS-besætninger (besætninger med malkerobotter, dvs. hvor der bliver malker hen over hele dagen). Derfor kan indvejningsmængden svinge fra gang til gang, fordi tankbilen måske afhenter mælken på lidt forskellige tidspunkter af døgnet.
Og ja, så er der helt sikkert også andre muligheder for afvigelser. Både i virkeligheden og i dataflowet.

Datateknik

Data er udtrukket med R - primært fra de to tabeller i Kvægdatabasen: H6601.MDAT og H8110.bedom. Udtræk fra disse tager ca. 15 minutter for 3-4 år.
Datasammenstilling tager kun få minutter. Dataindlæsning retur til Kvægdatabasen tager ca. 1½ time en hverdags formiddag.

Bilag B – Noter og programmer fra dataklargøring og datamodelering arbejdet med varmestress.

Noterne er en anonymiseret version af den Jupyter Notebook som er anvendt ved arbejdet.

DAT-1902 Dairy production data for all farms

In this notebook we fetch THI and dairy production data for all farms and perform a exploratory data analysis to understand the data and its relations.

Reproduce:

Conda env: from `py38_DAT-1902_AP2.yml` - created with the command `conda create -n py38_DAT-1902_AP2 python==3.8 adlfs pandas brotli geopandas dask python-dotenv holoviews geoviews hvplot ipykernel fastparquet xlrd openpyxl cx_oracle`

Install Oracle client on a Ubuntu machine to access KvægDB:

1. Download the RPM file at https://www.oracle.com/database/technologies/instant-client/linux-x86-64-downloads.html download `oracle-instantclient-basic-21.4.0.0.0-1.x86_64.rpm` (note version 21 is supported, as tested in this notebook, using cx_Oracle connection - not sqlalchemy connection which dit not work!)
2. Install apt packages needed by Ubuntu to install the RPM file: `sudo apt install alien libaio1`
3. Install the RMP file: `sudo alien -i oracle-instantclient-basic-21.4.0.0.0-1.x86_64.rpm`

In [1]:
```
# Standard Py packages
import os
import getpass
from pathlib import Path
import site

# 3-party Py Packages.
import adlfs
from dotenv import load_dotenv
from cartopy import crs
import cx_Oracle
import geoviews as gv
import holoviews as hv
import hvplot.pandas
import numpy as np
import pandas as pd

# Local Py packages.
site.addsitedir('data_munging')
from cattle import load_gdf_chr_locations

print(f'Oracle client version: {".".join([str(x) for x in cx_Oracle.clientversion()])}')

# Global parameters
# Load environment variable
load_dotenv()
THI_data_path = Path('curated/datascience/5517_MAF_Datadrevet_management_i_mælkeproduktionen/AP2/varmestress/CHR_THI_years_2018-2019-2020_
```

```
Oracle client version: 21.4.0.0.0
```

Fetch farm, THI and dairy production data

In [2]:
```
# Select single farm for show case.
df_farms = load_gdf_chr_locations(Path('input_data/MlkBesGis.xlsx'))
```

In [3]:
```
# Fetch the whole parquet file to the local disk.
local_file = f'input_data/{THI_data_path.name}'
df_wheather_farms = pd.read_parquet(local_file).rename_axis(['CHRNR', 'time'])
df_wheather_farms.head().style.hide_index()
```

```
/usr/local/continuum/miniconda3/envs/py38_DAT-1902_AP2/lib/python3.8/site-packages/fastparquet/core.py:278: UserWarning: Non-categorical multi-index is likely brittle
  warnings.warn("Non-categorical multi-index is likely brittle")
```

Out[3]:

DMI_air_temperature_Celsius	DMI_air_relative_humidity	DMI_air_temperature_Fahrenheit	THI_Fahrenheit
7.367000	91.755997	45.260601	45.838230
7.505000	92.536003	45.508999	46.021778
8.208000	89.722000	46.774399	47.408970
8.456000	87.030998	47.220802	47.989677
8.229000	86.723999	46.812199	47.629108

%%time # Fetch dairy production data from KvægDB Oracle Database. db_credentials = { 'dsn': os.getenv('KVDB_DSN') } db_credentials['user'] = os.getenv('KVDB_USER') db_credentials['password'] = os.getenv('KVDB_PASS') dairy_production_query = ''' SELECT * FROM r5ajni.PROJ_DDM_MLKtilTHI ''' # Connect to KvægDB. connection = cx_Oracle.connect(**db_credentials) print('Returning query result as pandas DataFrame...') df_tank_dairy = pd.read_sql_query(sql=dairy_production_query, con=connection)

In [4]:
```
df_tank_dairy = pd.read_parquet('input_data/PROJ_DDM_MLKtilTHI.parquet.brotli')  # Local copy of r5ajni.PROJ_DDM_MLKtilTHI
df_tank_dairy = df_tank_dairy.set_index(['CHRNR', 'DATO']).sort_index()
df_tank_dairy.head().style.hide_index().hide_columns(subset=['MEJNR', 'LEVNR'])
```

Out[4]:

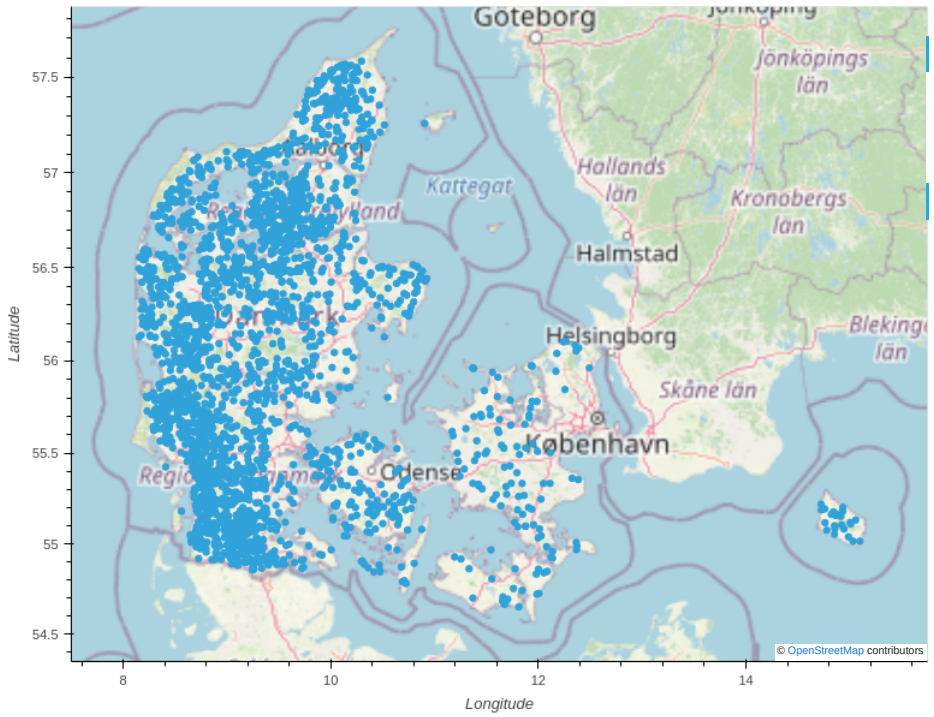
CELLETAL	KGMLK	FEDT	PROTEIN	KGEKM
nan	6038.000000	nan	nan	nan
nan	6111.000000	nan	nan	nan
nan	6092.000000	nan	nan	nan
nan	6073.000000	nan	nan	nan
nan	6022.000000	nan	nan	nan

Exploratory data analysis - before 4. meeting

In [5]:
```
# Plot all danish dairy farms on a map.
(gv.tile_sources.Wikipedia *
```

```
df_farms.reset_index().to_crs(crs.GOOGLE_MERCATOR.proj4_init).hvplot()).opts(
   width=800, height=600)
```

Out[5]:



In [6]:

```
%%time
# Compute daily wheather observations using daily max.
df_wheather_farms_daily = df_wheather_farms.reset_index(
    level=0).rename(columns={'level_0': 'CHRNR'}).groupby(
        'CHRNR').resample('1D').max().drop(columns='CHRNR').sort_index()
df_wheather_farms_daily

# Set same index on dataframes.
df_wheather_farms_daily = df_wheather_farms_daily.rename_axis(df_tank_dairy.index.names)
```

```
CPU times: user 15.6 s, sys: 1.71 s, total: 17.3 s
Wall time: 17.6 s
```

In [7]:

```
%%time
# Add daily max THI to dairy production data
df_dairy_THI = df_tank_dairy.merge(
    df_wheather_farms_daily, left_index=True, right_index=True)
df_dairy_THI.head().style.hide_index().hide_columns(subset=['MEJNR', 'LEVNR'])
```

```
CPU times: user 39.6 s, sys: 533 ms, total: 40.1 s
Wall time: 40.7 s
```

Out[7]:

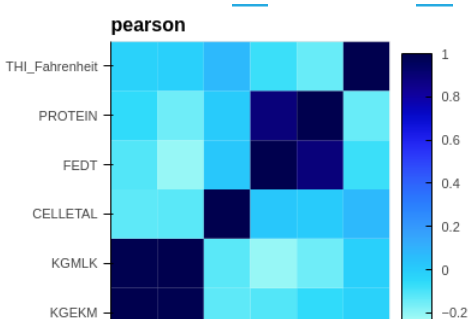
CELLETAL	KGMLK	FEDT	PROTEIN	KGEKM	DMI_air_temperature_Celsius	DMI_air_relative_humidity	DMI_air_temperature_Fahrenheit	THI_Fahrenheit
321000.000000	6121.000000	3.880000	3.450000	6051.095840	10.305000	99.694000	50.549000	51.699036
516000.000000	6097.000000	4.030000	3.470000	6148.319650	12.924000	85.247002	55.263199	56.037224
381000.000000	6125.000000	3.870000	3.520000	6080.623010	17.158001	88.541000	62.884399	61.330254
226000.000000	6123.000000	3.790000	3.430000	5976.418500	20.638000	83.912003	69.148399	65.806976
320000.000000	6429.000000	3.790000	3.480000	6299.867190	21.042999	82.918999	69.877396	66.317535

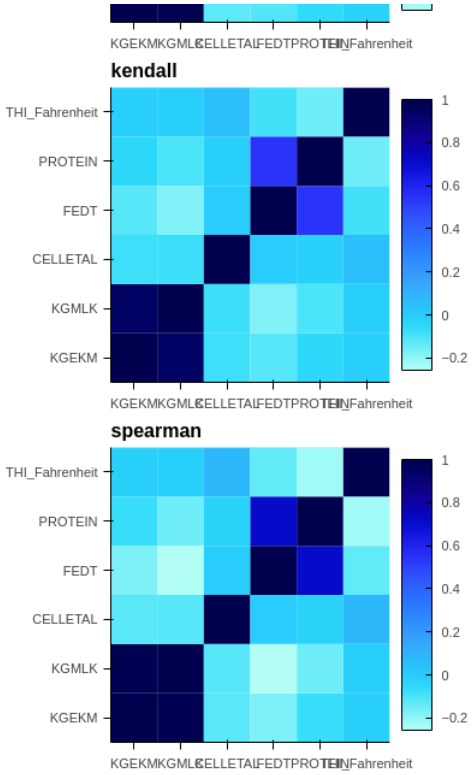
In [8]:

```
# Single CHRNR analysis
CHRNR = int(Path('chrnr.txt').read_text())  # CHR to focus analyse on
dairy_cols = ['KGEKM', 'KGMLK', 'CELLETAL', 'FEDT', 'PROTEIN', 'THI_Fahrenheit']

# Corrations.
hv.Layout([
    df_dairy_THI[dairy_cols].corr(method).hvplot.heatmap(title=method, width=400)
    for method in ['pearson', 'kendall', 'spearman']]).cols(1)  # no cols
```

```
/usr/local/continuum/miniconda3/envs/py38_DAT-1902_AP2/lib/python3.8/site-packages/scipy/stats/stats.py:4812: RuntimeWarning: overflow enc
ountered in long_scalars
  (2 * xtie * ytie) / m + x0 * y0 / (9 * m * (size - 2)))
```

Out[8]:

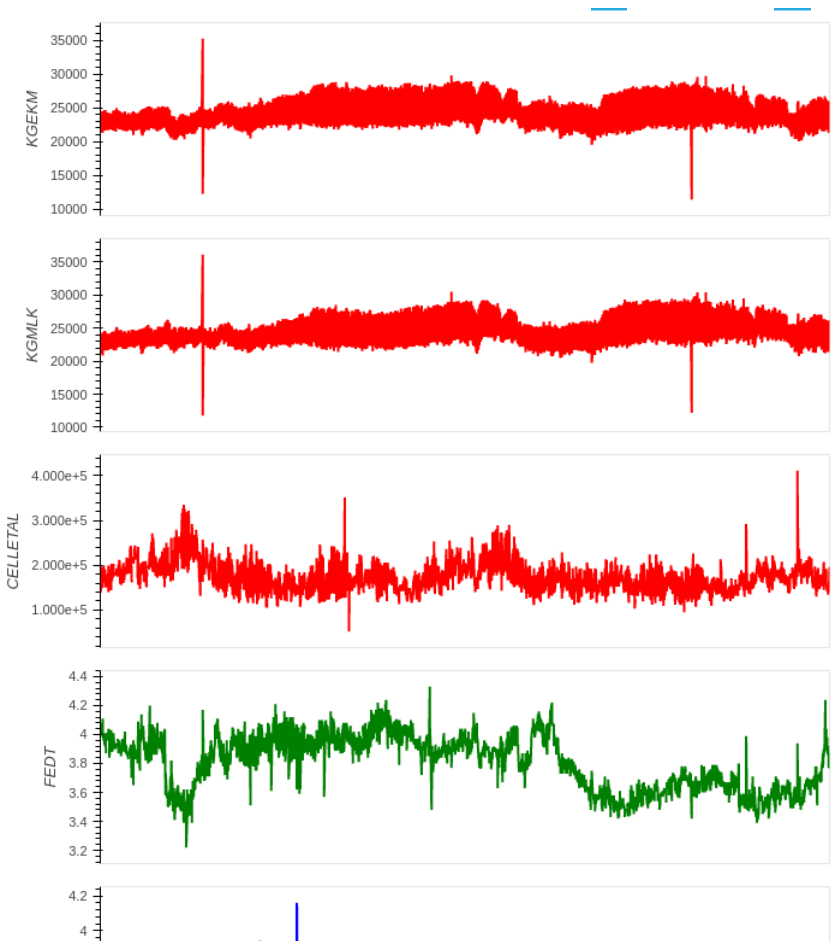


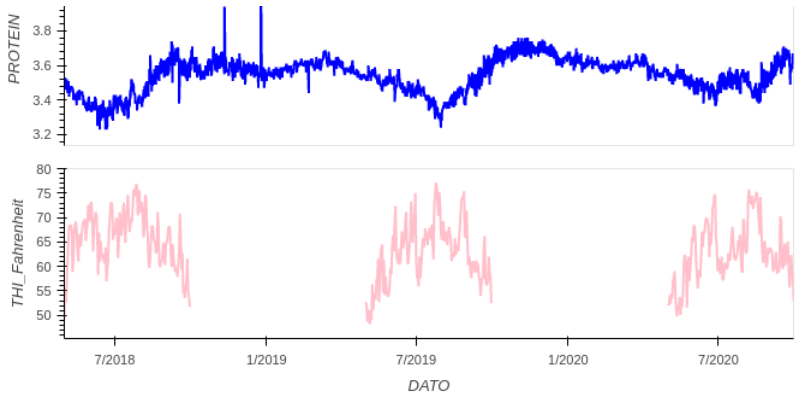


In [9]:

```
# Show time line with THI and dairy production.
col_color = {
    'KGEKM': 'red',
    'KGMLK': 'red',
    'CELLETAL': 'red',
    'FEDT': 'green',
    'PROTEIN': 'blue',
    'THI_Fahrenheit': 'pink'}
plots = []
for col in dairy_cols:
    xaxis = None if col != dairy_cols[-1] else True
    height = 180 if col != dairy_cols[-1] else 200
    plots.append(df_dairy_THI.loc[CHRNR].hvplot(
        y=col, xaxis=xaxis, color=col_color[col], height=height))
hv.Layout(plots).cols(1)
```

Out[9]:





Exploratory data analysis - before 5. meeting

Compute THI-hours and THI-days

Similar to heating degree days we compute the THI hours, where the THI value have been about a threshold. This measurement should show when during the heat-season the THI accumulates thus has bigger effect on the cattle and also enable an daily aggregated measurement which contains more info from the hourly samples than just a simple daily max, which we up-to-now have used.

Conclusion

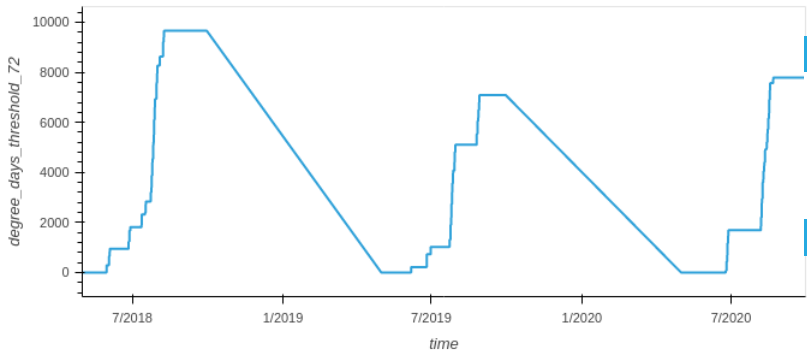
The computed THI hours (i.e. cumulative sum of hourly THI values over `threshold` per year) and THI days (i.e. daily sum of "THI hours") show how the THI accumulates in small intervals, thus leaving plateaus over the summer where the THI is not over the `mild=72` threshold.

However, the interesting events of the THI days, is when the values is increasing over multiple days in a row - thus the cattle is exposed to heat stress multiple days in a row.

In [10]:
```
# Define levels of heatstrees using THI, source: https://www.pericoli.com/EN/news/120/Temperature-Humidity-Index-what-you-need-to-know-abo
heat_stress_THI_edge_levels = {
    'none': 0,
    'mild': 72,
    'moderate': 80,
    'severe': 90
}
```

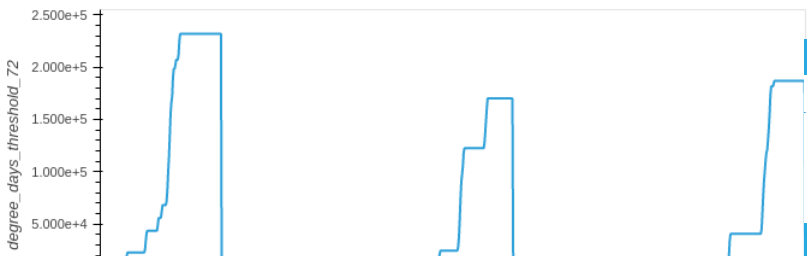
In [11]:
```
def _yearly_THI_hourly(df_temp, threshold):
    df_year_THI_hours = df_temp.where(df_temp - threshold > 0, 0).cumsum().rename(
        columns={df_temp.columns[0]: f'degree_days_threshold_{threshold}'})
    return df_year_THI_hours

def THI_hourly(df_temp, threshold):
    """Cumulative sum of hourly THI values over `threshold` per year."""
    assert len(df_temp.columns) == 1, (
        'Can not determine temputure column, as multiple columns are present!')
    df_THI_hour = df_temp.groupby(df_temp.index.year).apply(
        _yearly_THI_hourly, threshold)
    return df_THI_hour

THI_hourly(df_wheather_farms.loc[CHRNR][['THI_Fahrenheit']], threshold=heat_stress_THI_edge_levels['mild']).hvplot()
```

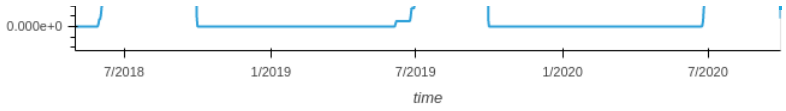
Out[11]:



In [12]:
```
def THI_daily(df_temp, threshold):
    df_THI_hour = THI_hourly(df_temp, threshold)
    df_THI_day = df_THI_hour.resample('1D').sum()
    return df_THI_day

THI_daily(df_wheather_farms.loc[CHRNR][['THI_Fahrenheit']], threshold=heat_stress_THI_edge_levels['mild']).hvplot()
```

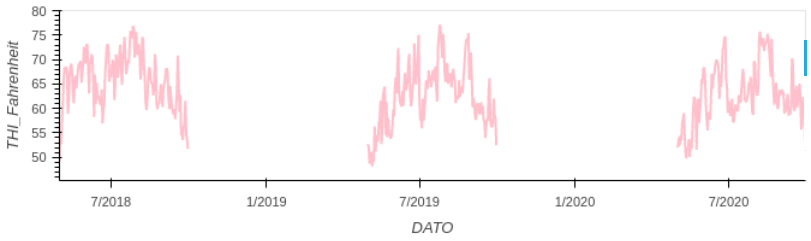
Out[12]:





In [13]:
```
plots[-1]
```

Out[13]:



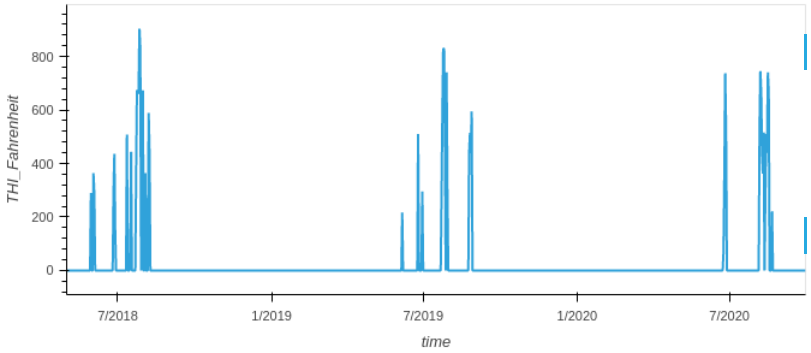
Simple THI sum per day

Conclusion

Simply taking the daily sum if the hourly THI values also show useful information, which is that compared to the daily max values, the days with more hours with this THI actually can be ween in the daily sum.

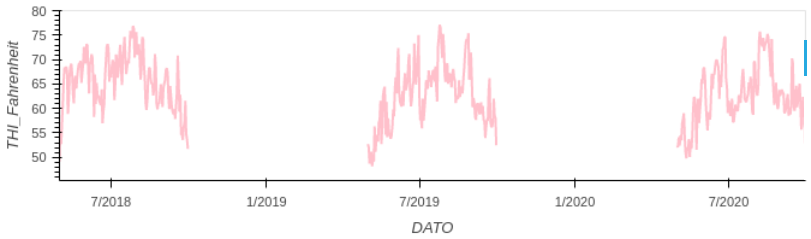
In [14]:
```
def THI_daily_sum(df_temp, threshold=0):
    df_THI_day = df_temp.where(df_temp > threshold).resample('1D').sum()
    return df_THI_day

THI_daily_sum(df_wheather_farms.loc[CHRNR][['THI_Fahrenheit']], threshold=heat_stress_THI_edge_levels['mild']).hvplot()
```

Out[14]:



In [15]:
```
plots[-1]
```

Out[15]:



Compute 2-days running mean of dairy production

Compute a daily dairy production. Some dairy produces gets their tank emptied multiple times per day, thus such observations for the same date must be resampled to only 1 per date. This can be done by multiple ways.

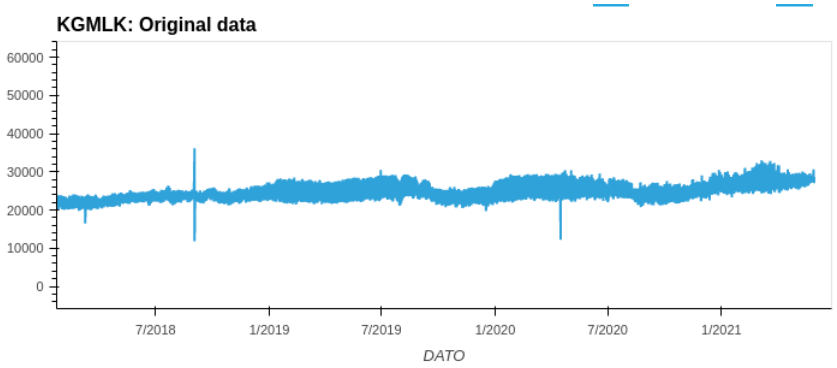
Conclusion

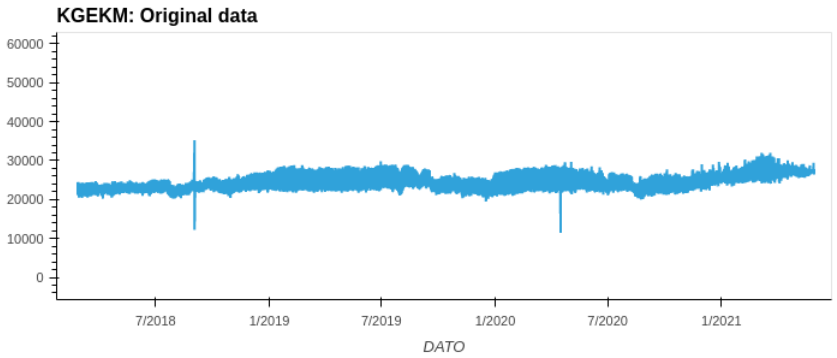
Based on the observations of KGMLK and KGEKM we propose the most correct resampling is to 1 day interval using the sum of the multiple observations of the same date (see the plot "KGMLK: 1 day resample using sum"). However, when comparing KGMLK and KGEKM we see that KGEKM has some dates with NaN where the KGMLK exists. This need to be explained.

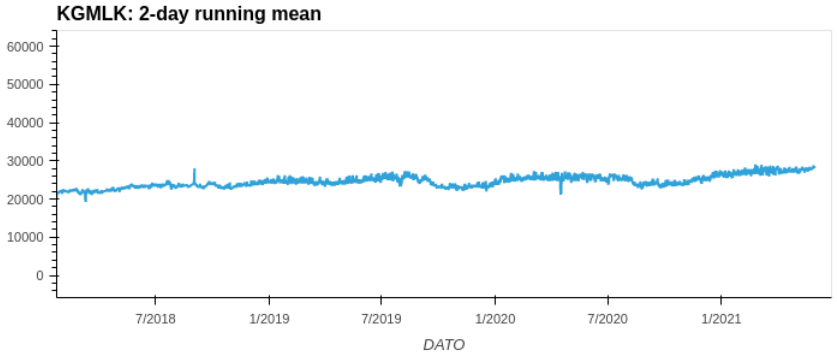
Our statistics of observation frequency for all CHR numbers, show that most fields has +1200 observation over the whole time interval 2018 to 2021. However, CHRs differs in the number of observations per day. Some few (3) has up to 3 observations for the same date. However, most CHRs has in max 1 observation per date, and the average number of most CHRs is between 0.5 and 1. Meaning most CHRs get the tank emptied each or each other day.

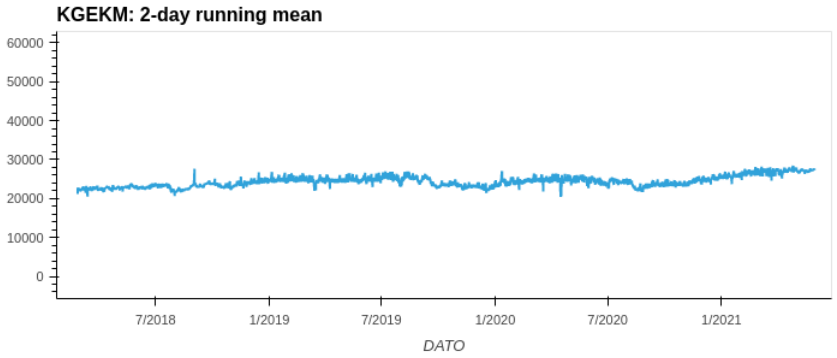
In [16]:
```
(
    df_tank_dairy.loc[CHRNR].KGMLK.hvplot(title='KGMLK: Original data') +
    df_tank_dairy.loc[CHRNR].KGEKM.hvplot(title='KGEKM: Original data') +
    df_tank_dairy.loc[CHRNR].KGMLK.rolling('2D').mean().hvplot(title='KGMLK: 2-day running mean') +
    df_tank_dairy.loc[CHRNR].KGEKM.rolling('2D').mean().hvplot(title='KGEKM: 2-day running mean') +
    df_tank_dairy.loc[CHRNR].KGMLK.resample('1D').sum().hvplot(title='KGMLK: 1 day resample using sum') +
    df_tank_dairy.loc[CHRNR].KGEKM.resample('1D').sum().hvplot(title='KGEKM: 1 day resample using sum') +
    df_tank_dairy.loc[CHRNR].KGMLK.resample('1D').sum().rolling('2D').mean().hvplot(title='KGMLK: 1 day resample using sum + 2-day running
    df_tank_dairy.loc[CHRNR].KGEKM.resample('1D').sum().rolling('2D').mean().hvplot(title='KGEKM: 1 day resample using sum + 2-day running
).cols(1)  # cols(2)
```

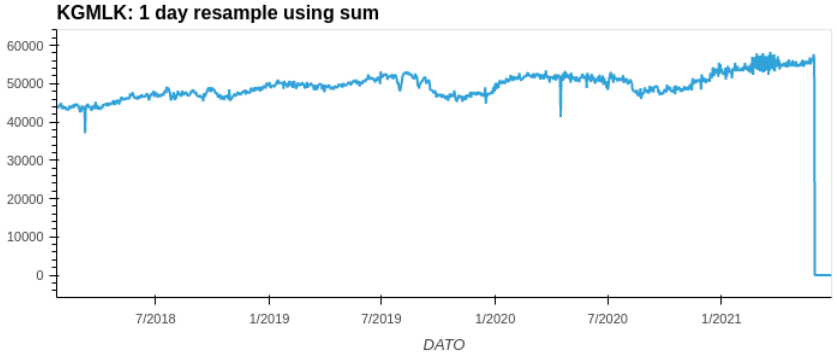
Out[16]:

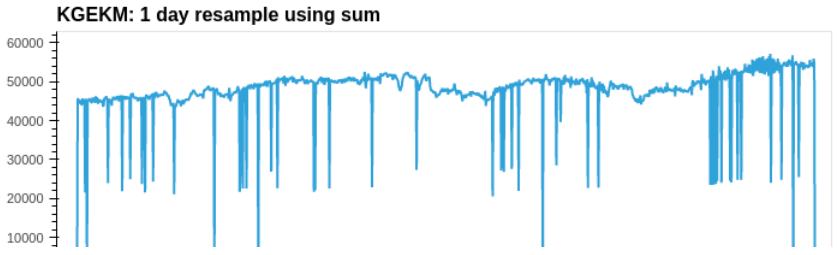

KGMLK: Original data
60000
50000
40000
30000
20000
10000
0
7/2018
1/2019
7/2019
1/2020
7/2020
1/2021
DATO

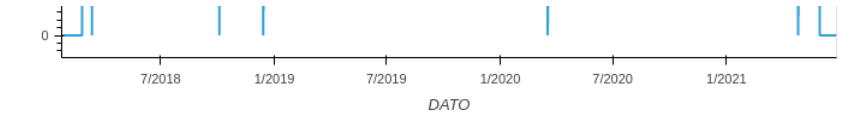

KGEKM: Original data
60000
50000
40000
30000
20000
10000
0
7/2018
1/2019
7/2019
1/2020
7/2020
1/2021
DATO

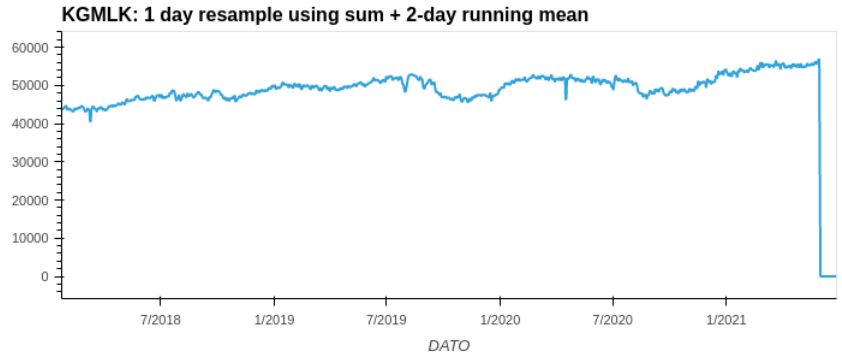

KGMLK: 2-day running mean
60000
50000
40000
30000
20000
10000
0
7/2018
1/2019
7/2019
1/2020
7/2020
1/2021
DATO

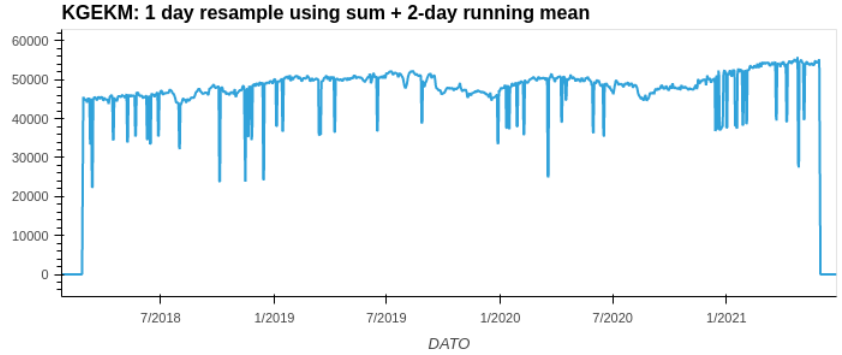

KGEKM: 2-day running mean
60000
50000
40000
30000
20000
10000
0
7/2018
1/2019
7/2019
1/2020
7/2020
1/2021
DATO


KGMLK: 1 day resample using sum
60000
50000
40000
30000
20000
10000
0
7/2018
1/2019
7/2019
1/2020
7/2020
1/2021
DATO


KGEKM: 1 day resample using sum
60000
50000
40000
30000
20000
10000







In [17]:

```
# Some observation has KGMLK but NaN KGEKM, this is also seen in the curve plots above.
df_tank_dairy.loc[CHRNR].loc['2018-03-12'].style.hide_columns(['MEJNR', 'LEVNR'])
```

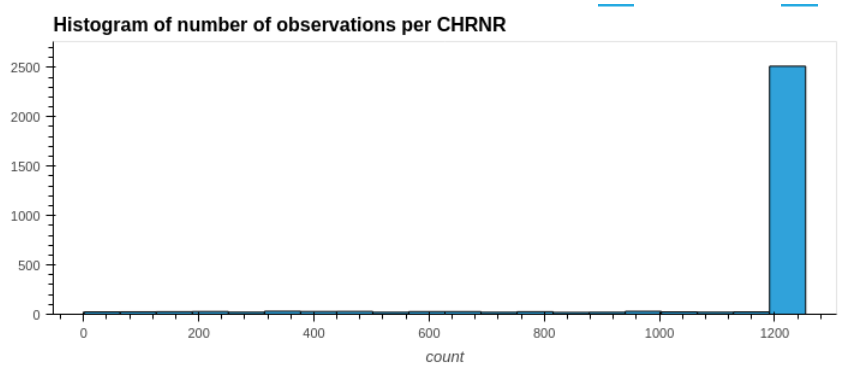
Out[17]:

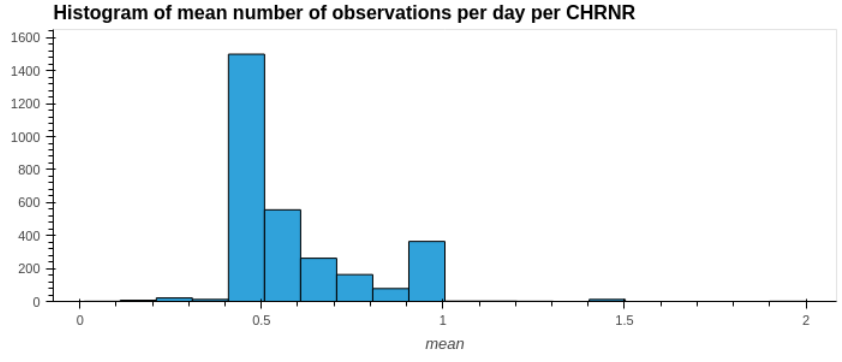
DATO	CELLETAL	KGMLK	FEDT	PROTEIN	KGEKM
2018-03-12 00:00:00	nan	23671.000000	nan	nan	nan
2018-03-12 00:00:00	nan	20711.000000	nan	nan	nan

In [18]:

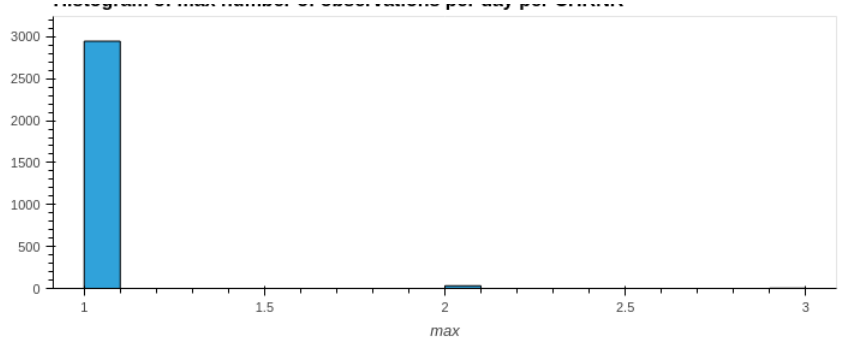
```
# Statistics of observation frequency for all CHR number.
df_stats = df_tank_dairy.groupby('CHRNR').apply(
    lambda x: x.groupby('DATO').size().reindex(
        pd.date_range(x.index.get_level_values('DATO').min(), x.index.get_level_values('DATO').max()),
        fill_value=0).describe())
(
    df_stats['count'].hvplot.hist(title='Histogram of number of observations per CHRNR') +
    df_stats['mean'].hvplot.hist(title='Histogram of mean number of observations per day per CHRNR') +
    df_stats['max'].hvplot.hist(title='Histogram of max number of observations per day per CHRNR')
).cols(1)
```

Out[18]:





Histogram of max number of observations per day per CHRNR



Correlation analyses of all CHRNRs

Correlation analyses of all CHRNRs between THI and KGEKM/Celletal after 2, 4, and up to 20 days delay of KGEKM/Celletal (this also include log10(Celletal). This new correlation analyses should be mad on new resampled observations to 1 day interval and only one observation per date - where the old data could contain multiple observations per date. Thus, daily THI is the daily sum of hourly THI values over the `mild=72` threshold and daily dairy production attributes is aggregated individually using mean or sum.

Conclusion

Plotting the new resampled observations besides the old shows clearly the difference, where the new daily dairy data has much less fluctuations (i.e. less wide curve) as only one observation per date exists for `CHRNR=31945`.

The correlation analyses shows a increasing correlation between daily max THI (shifted `x` days) and Celletal (Pearson from 0.051 to 0.089) up until the 12 days shift, where after it decreases again. But there in no improved correlation between THI and the other dairy attributes.

In [19]:
```
# Compute the logarithm base 10 of CELLETAL.
df_tank_dairy = df_tank_dairy.assign(CELLETAL_log10=np.log10(df_tank_dairy['CELLETAL']))
df_tank_dairy.head().style.hide_index().hide_columns(subset=['MEJNR', 'LEVNR'])
```

```
/usr/local/continuum/miniconda3/envs/py38_DAT-1902_AP2/lib/python3.8/site-packages/pandas/core/arraylike.py:364: RuntimeWarning: divide by zero encountered in log10
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[19]:

CELLETAL	KGMLK	FEDT	PROTEIN	KGEKM	CELLETAL_log10
nan	6038.000000	nan	nan	nan	nan
nan	6111.000000	nan	nan	nan	nan
nan	6092.000000	nan	nan	nan	nan
nan	6073.000000	nan	nan	nan	nan
nan	6022.000000	nan	nan	nan	nan

In [20]:
```
%%time
# First we resamples dairy observation to 1day interval.
df_tank_dairy_daily = df_tank_dairy.groupby('CHRNR').apply(
    lambda x: x.resample(rule='1D', level='DATO').agg({
        'MEJNR': 'mean',
        'LEVNR': 'mean',
        'CELLETAL': 'mean',
        'CELLETAL_log10': 'mean',
        'FEDT': 'mean',
        'PROTEIN': 'mean',
        'KGMLK': 'sum',
        'KGEKM': 'sum'}).rolling('2D').mean())
df_tank_dairy_daily.head().style.hide_index().hide_columns(subset=['MEJNR', 'LEVNR'])
```

```
CPU times: user 19.7 s, sys: 181 ms, total: 19.9 s
Wall time: 20.1 s
```

Out[20]:

CELLETAL	CELLETAL_log10	FEDT	PROTEIN	KGMLK	KGEKM
nan	nan	nan	nan	6038.000000	0.000000
nan	nan	nan	nan	3019.000000	0.000000
nan	nan	nan	nan	3055.500000	0.000000
nan	nan	nan	nan	3055.500000	0.000000
nan	nan	nan	nan	3046.000000	0.000000

In [21]:
```
%%time
def THI_daily_sum(df_temp, threshold=0):
    df_THI_day = df_temp.groupby('CHRNR').apply(
        lambda x: x.where(df_temp > threshold).resample(
            rule='1D', level='time').sum()).add_suffix(
                f'_sum_threshold{threshold}')
    return df_THI_day

df_wheather_farms_daily_sum = THI_daily_sum(
        df_wheather_farms[['THI_Fahrenheit']], threshold=heat_stress_THI_edge_levels['mild'])
df_wheather_farms_daily_sum.head().style.hide_index()
```

```
CPU times: user 8min 49s, sys: 1.69 s, total: 8min 51s
Wall time: 8min 54s
```

Out[21]:

THI_Fahrenheit_sum_threshold72
0.000000
0.000000
0.000000
0.000000

THI_Fahrenheit_sum_threshold72
0.000000

In [22]:
```
%%time
# Add daily sum THI to dairy production data
df_dairy_THI_new = df_tank_dairy_daily.merge(
    df_wheather_farms_daily_sum.rename_axis(df_tank_dairy_daily.index.names),
    left_index=True, right_index=True)
df_dairy_THI_new.head().style.hide_index().hide_columns(subset=['MEJNR', 'LEVNR'])
```

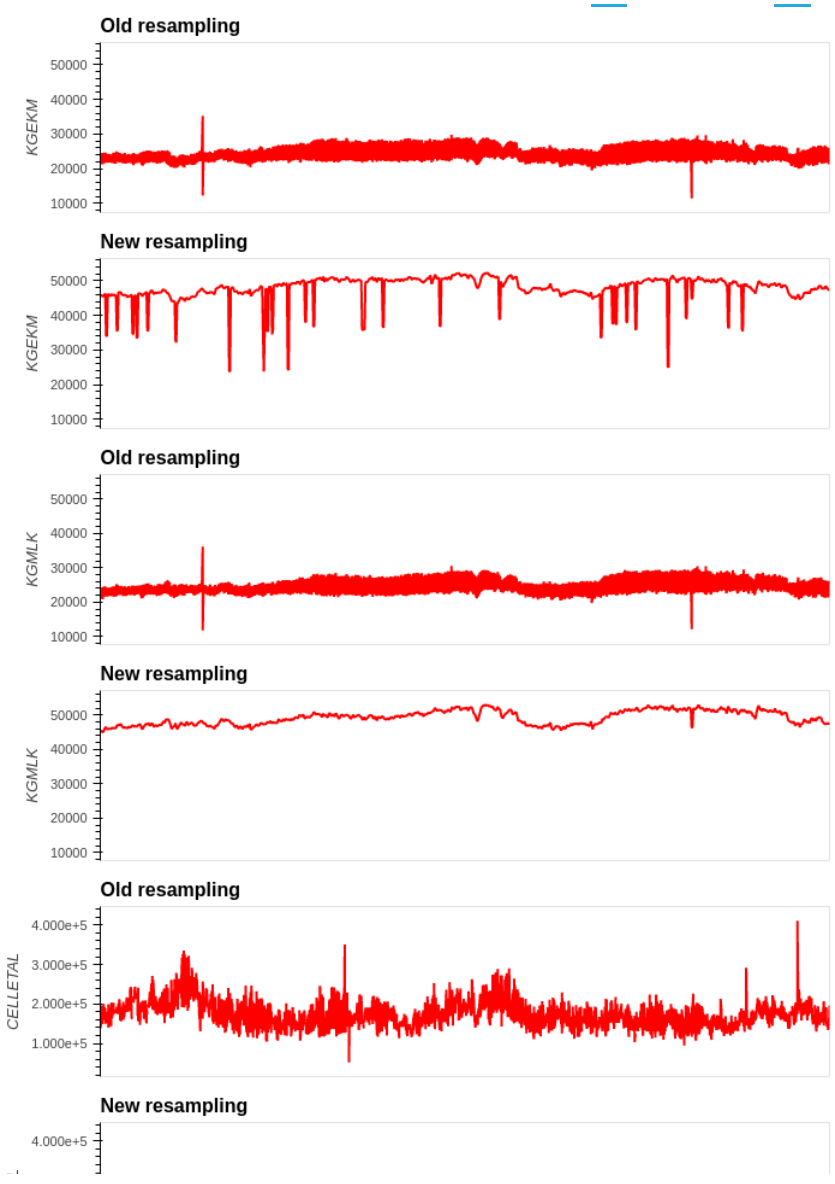
```
CPU times: user 47.5 s, sys: 754 ms, total: 48.3 s
Wall time: 49 s
```

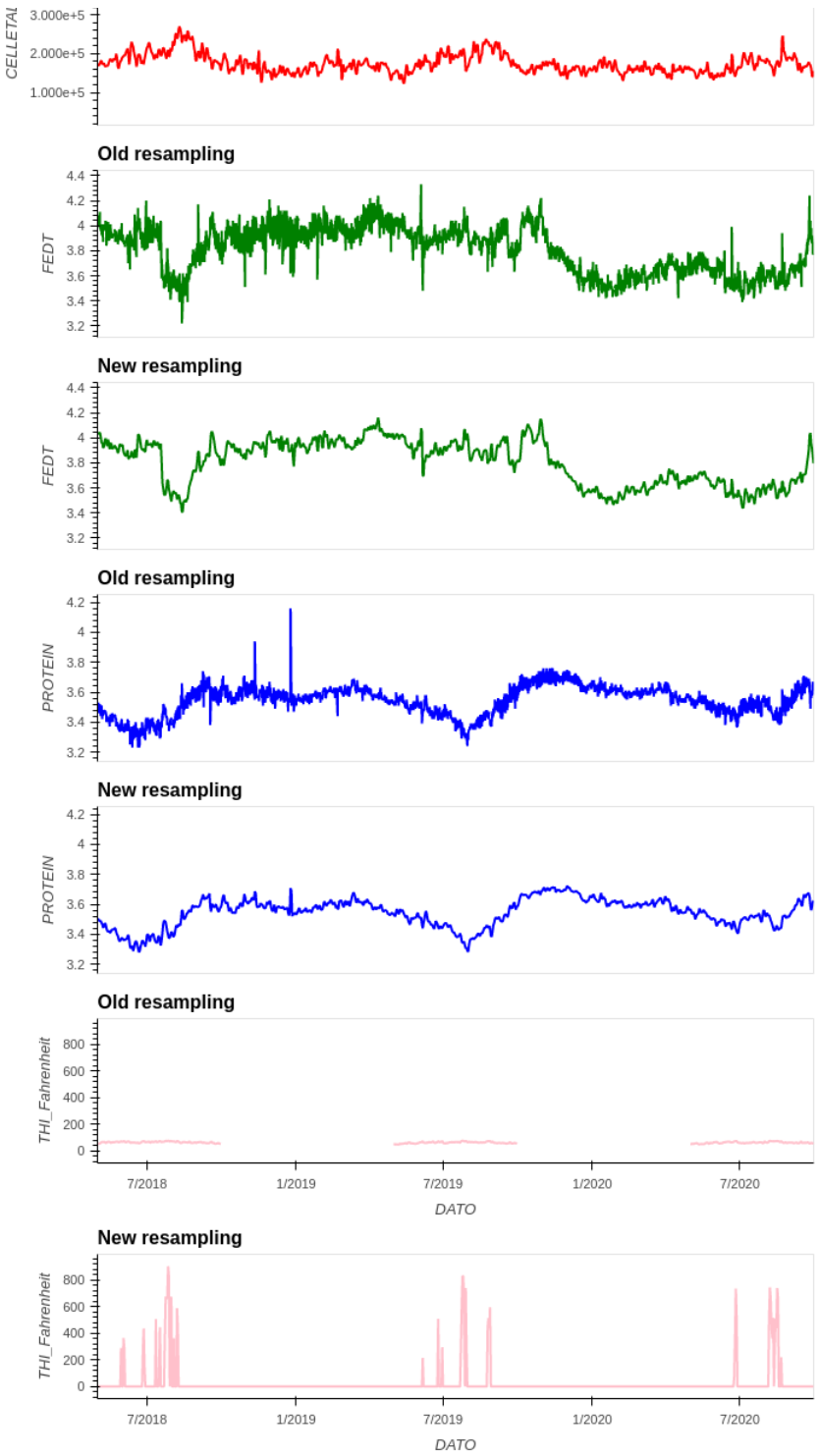
Out[22]:

CELLETAL	CELLETAL_log10	FEDT	PROTEIN	KGMLK	KGEKM	THI_Fahrenheit_sum_threshold72
321000.000000	5.506505	3.880000	3.450000	3060.500000	3025.547920	0.000000
321000.000000	5.506505	3.880000	3.450000	3060.500000	3025.547920	0.000000
516000.000000	5.712650	4.030000	3.470000	3048.500000	3074.159825	0.000000
516000.000000	5.712650	4.030000	3.470000	3048.500000	3074.159825	0.000000
381000.000000	5.580925	3.870000	3.520000	3062.500000	3040.311505	0.000000

In [23]:
```
# Show time line with THI and dairy production (NB: same plot as for the "previous" `df_dairy_THI` dataset!)
col_color = {
    'KGEKM': 'red',
    'KGMLK': 'red',
    'CELLETAL': 'red',
    'FEDT': 'green',
    'PROTEIN': 'blue',
    'THI_Fahrenheit': 'pink'}
plots = []
for col in dairy_cols:
    xaxis = None if col != dairy_cols[-1] else True
    height = 180 if col != dairy_cols[-1] else 200
    plots.append(df_dairy_THI.loc[CHRNR].hvplot(
        y=col, xaxis=xaxis, color=col_color[col], height=height, title='Old resampling'))
    plots.append(df_dairy_THI_new.rename(
        columns={'THI_Fahrenheit_sum_threshold72': 'THI_Fahrenheit'}).loc[CHRNR].hvplot(
            y=col, xaxis=xaxis, color=col_color[col], height=height, title='New resampling'))
hv.Layout(plots).cols(1)  # cols(2)
```

Out[23]:





In [24]:

```
df_dairy_THI_new = df_dairy_THI_new.assign(
        **{f'THI_Fahrenheit_sum_threshold72_{days}days': df_dairy_THI_new['THI_Fahrenheit_sum_threshold72'].shift(days)
           for days in range(2, 22, 2)})
df_dairy_THI_new.head().style.hide_index().hide_columns(subset=['MEJNR', 'LEVNR'])
```

Out[24]:

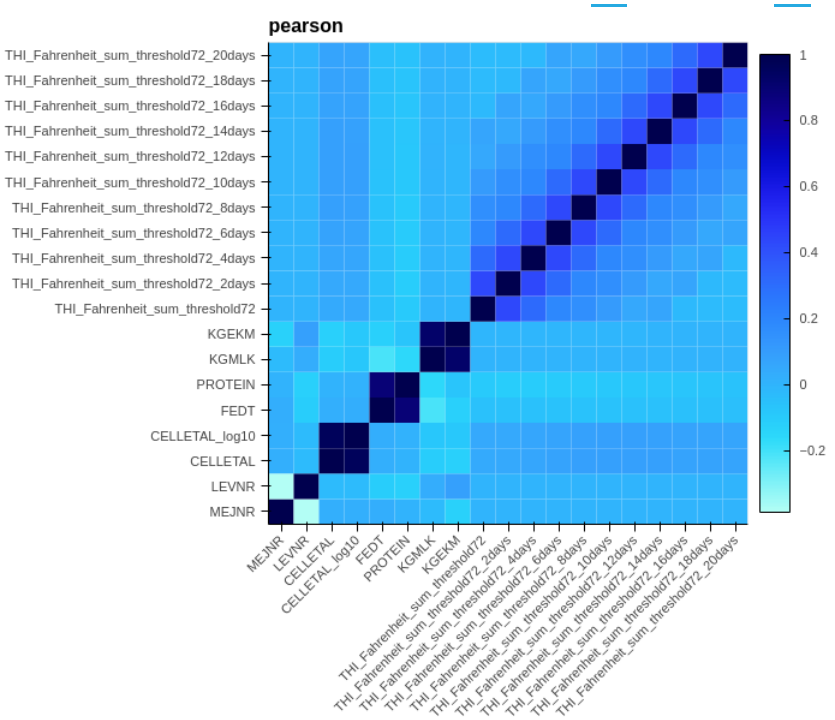
CELLETAL	CELLETAL_log10	FEDT	PROTEIN	KGMLK	KGEKM	THI_Fahrenheit_sum_threshold72	THI_Fahrenheit_sum_threshold72_2days	THI_Fahrenheit_sum_t
321000.000000	5.506505	3.880000	3.450000	3060.500000	3025.547920	0.000000	nan	
321000.000000	5.506505	3.880000	3.450000	3060.500000	3025.547920	0.000000	nan	
516000.000000	5.712650	4.030000	3.470000	3048.500000	3074.159825	0.000000	0.000000	
516000.000000	5.712650	4.030000	3.470000	3048.500000	3074.159825	0.000000	0.000000	
381000.000000	5.580925	3.870000	3.520000	3062.500000	3040.311505	0.000000	0.000000	

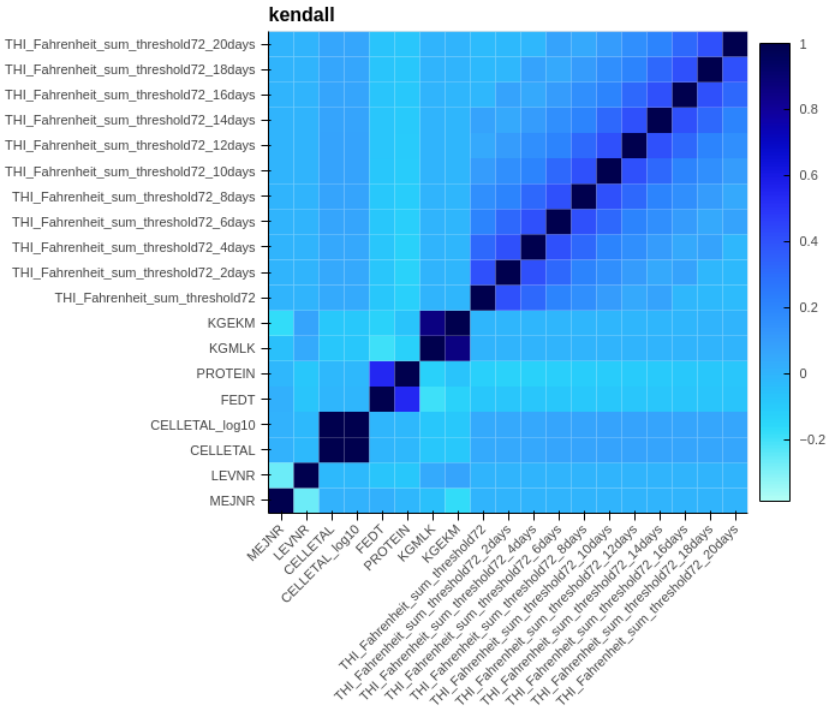
In [25]:

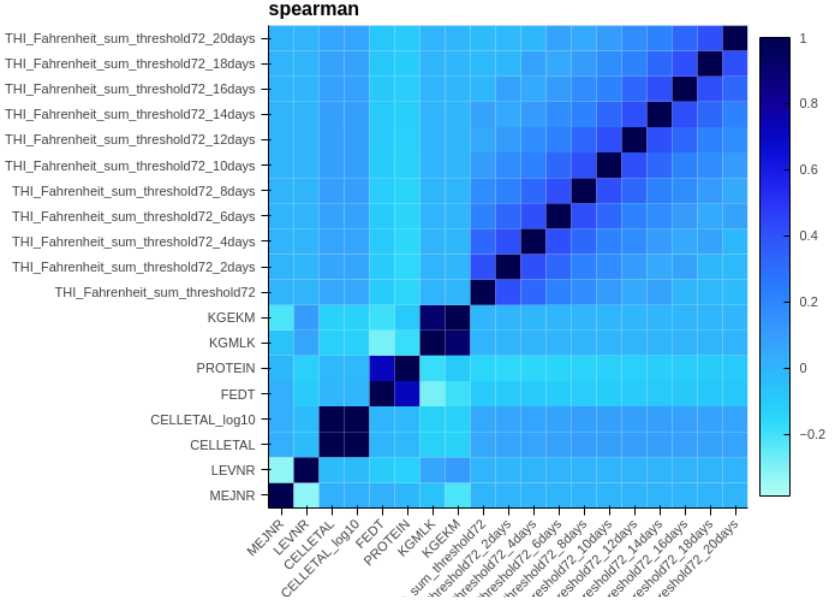
```
%%time
# Corrations.
hv.Layout([
    df_dairy_THI_new.corr(
            method).hvplot.heatmap(title=method, width=700, height=600).opts(xrotation=45)
    for method in ['pearson', 'kendall', 'spearman']]).cols(1)
```

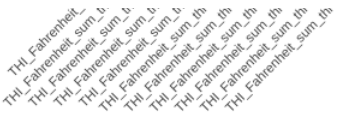
```
/usr/local/continuum/miniconda3/envs/py38_DAT-1902_AP2/lib/python3.8/site-packages/scipy/stats/stats.py:4812: RuntimeWarning: overflow enc
ountered in long_scalars
  (2 * xtie * ytie) / m + x0 * y0 / (9 * m * (size - 2)))
CPU times: user 1min 41s, sys: 2.92 s, total: 1min 44s
Wall time: 1min 44s
```

Out[25]:


pearson
THI_Fahrenheit_sum_threshold72_20days
THI_Fahrenheit_sum_threshold72_18days
THI_Fahrenheit_sum_threshold72_16days
THI_Fahrenheit_sum_threshold72_14days
THI_Fahrenheit_sum_threshold72_12days
THI_Fahrenheit_sum_threshold72_10days
THI_Fahrenheit_sum_threshold72_8days
THI_Fahrenheit_sum_threshold72_6days
THI_Fahrenheit_sum_threshold72_4days
THI_Fahrenheit_sum_threshold72_2days
THI_Fahrenheit_sum_threshold72
KGEKM
KGMLK
PROTEIN
FEDT
CELLETAL_log10
CELLETAL
LEVNR
MEJNR
1
0.8
0.6
0.4
0.2
0
−0.2


kendall
THI_Fahrenheit_sum_threshold72_20days
THI_Fahrenheit_sum_threshold72_18days
THI_Fahrenheit_sum_threshold72_16days
THI_Fahrenheit_sum_threshold72_14days
THI_Fahrenheit_sum_threshold72_12days
THI_Fahrenheit_sum_threshold72_10days
THI_Fahrenheit_sum_threshold72_8days
THI_Fahrenheit_sum_threshold72_6days
THI_Fahrenheit_sum_threshold72_4days
THI_Fahrenheit_sum_threshold72_2days
THI_Fahrenheit_sum_threshold72
KGEKM
KGMLK
PROTEIN
FEDT
CELLETAL_log10
CELLETAL
LEVNR
MEJNR
1
0.8
0.6
0.4
0.2
0
−0.2


spearman
THI_Fahrenheit_sum_threshold72_20days
THI_Fahrenheit_sum_threshold72_18days
THI_Fahrenheit_sum_threshold72_16days
THI_Fahrenheit_sum_threshold72_14days
THI_Fahrenheit_sum_threshold72_12days
THI_Fahrenheit_sum_threshold72_10days
THI_Fahrenheit_sum_threshold72_8days
THI_Fahrenheit_sum_threshold72_6days
THI_Fahrenheit_sum_threshold72_4days
THI_Fahrenheit_sum_threshold72_2days
THI_Fahrenheit_sum_threshold72
KGEKM
KGMLK
PROTEIN
FEDT
CELLETAL_log10
CELLETAL
LEVNR
MEJNR
1
0.8
0.6
0.4
0.2
0
−0.2



Correlation analysis of each CHRNR to find CHRNR prone heat stress

The goal with these correlation analyses of each CHRNR is to find CHRNR which is prone to heat stress, because when we know which is this we can help them by warning of high THI, or help educate them to handle the heat stress better, e.g by adding cooling machinery in the stable.

To simplify the correlation analyses we:

- negate the milk amount attributes (`KGMLK` and `KGEKM`), such that all correlation coefficient with THI will be possible. This enables our selection of CHRNR prone to heat stress, to be based on the maximum coefficients of milk production and cell-amount (`CELLETAL` and `CELLETAL_log10`).
- remove all CHRNR not delivering to Arla (`MEJNR=1`), because non-Arla CHRNR may get the tank-milk collected ones every week, instead ones every day. Thus we have much better data with Arla CHRNR.

Based on correlation coefficients we select top 3 CHRNRs with:

1. maximum sum correlation coefficients of all negative milk amount attributes (i.e. `KGMLK` and `KGEKM`) and all THI day-shifts (2 to 20 days shifted)
2. maximum sum correlation coefficients of all cell-amount attributes (i.e. only `CELLETAL`) and all THI day-shifts (2 to 20 days shifted)
3. maximum correlation coefficient between cell-amount (i.e. only `CELLETAL`) and THI shifted 12 days - as our correlation analysis over all CHRNRs show that 12 days has the highest coefficient.

Conclusion

The CHRNRs we found all have correlation coefficients about 0.3 to 0.5 at one or more if the THI-shifts. The time-series data plot of the top 3 CHRNRs also visually shows the correlation between THI and decreasing milk amount and THI (daily sum with threshold of 72) and increasing cell-amount. Thus, we conclude the data of the found CHRNRs supports our assumption, of having a strong possibility of having had heat stress during THI over 72.

Future work

- We can turn the THI threshold such that more farms with heat stress problems (i.e. correlation between THI and cell-amount/milk production) can be detected and helped.

In [26]:
```
# Negate the milk production attributes
df_dairy_THI_new = df_dairy_THI_new.assign(
    KGMLK_neg=-df_dairy_THI_new['KGMLK'],
    KGEKM_neg=-df_dairy_THI_new['KGEKM'])
```

In [27]:
```
THI_columns = df_dairy_THI_new.columns[df_dairy_THI_new.columns.str.contains('THI')]
THI_columns
```

Out[27]:
```
Index(['THI_Fahrenheit_sum_threshold72',
       'THI_Fahrenheit_sum_threshold72_2days',
       'THI_Fahrenheit_sum_threshold72_4days',
       'THI_Fahrenheit_sum_threshold72_6days',
       'THI_Fahrenheit_sum_threshold72_8days',
       'THI_Fahrenheit_sum_threshold72_10days',
       'THI_Fahrenheit_sum_threshold72_12days',
       'THI_Fahrenheit_sum_threshold72_14days',
       'THI_Fahrenheit_sum_threshold72_16days',
       'THI_Fahrenheit_sum_threshold72_18days',
       'THI_Fahrenheit_sum_threshold72_20days'],
      dtype='object')
```

In [28]:
```
%%time
# Remove non-Arla CHRNRs, then compute pearson correlation coefficients.
df_dairy_THI_CHR_corr = df_dairy_THI_new[df_dairy_THI_new['MEJNR'] == 1].drop(
    columns=['MEJNR', 'LEVNR', 'FEDT', 'PROTEIN', 'KGMLK', 'KGEKM']).groupby(
        'CHRNR').apply(lambda x: x.corr().loc[['CELLETAL', 'CELLETAL_log10', 'KGMLK_neg', 'KGEKM_neg'], THI_columns])
df_dairy_THI_CHR_corr.head().style.hide_index()
```

```
CPU times: user 4.53 s, sys: 136 ms, total: 4.66 s
Wall time: 5.15 s
```

Out[28]:

THI_Fahrenheit_sum_threshold72	THI_Fahrenheit_sum_threshold72_2days	THI_Fahrenheit_sum_threshold72_4days	THI_Fahrenheit_sum_threshold72_6days	THI_Fahrenheit_sum_th
0.135470	0.123296	0.144322	0.201679	
0.130748	0.122953	0.142575	0.189274	
0.029802	0.010922	0.013372	0.014064	
0.040453	0.047472	0.023820	0.048950	
-0.039654	-0.016006	0.002621	0.022307	

In [29]:
```
# Find top 3 CHRNRs with max sum of corr.coefficients between negative milk-production and all THI-shifts.
ser_max_sum_milk_coef = df_dairy_THI_CHR_corr.loc[pd.IndexSlice[: , ['KGMLK_neg', 'KGEKM_neg']], :].sum(axis=1).groupby('CHRNR').sum().sor
ser_max_sum_milk_coef.head().to_frame().style.hide_index()
```

Out[29]:

0
5.857980
5.639146
5.476943
5.398231
5.344443

In [30]:
```
CHR_max_sum_milk_coef_top3 = ser_max_sum_milk_coef.head(3).index.to_list()
```

In [31]:
```
# Find top 3 CHRNRs with max sum of corr.coefficients between cell-amount and THI-shifts.
ser_max_sum_cell_coef = df_dairy_THI_CHR_corr.loc[pd.IndexSlice[: , ['CELLETAL']], :].sum(axis=1).groupby('CHRNR').sum().sort_values(ascen
ser_max_sum_cell_coef.head().to_frame().style.hide_index()
```

Out[31]:

0
4.131540
3.480650
3.431103
3.347583
3.300973

In [32]:
```
CHR_max_sum_cell_coef_top3 = ser_max_sum_cell_coef.head(3).index.to_list()
```

In [33]:
```
# Find top 3 CHRNRs with max of corr.coefficient between cell-amount and THI shifted 12 days.
ser_max_THI12_cell_coef = df_dairy_THI_CHR_corr.loc[pd.IndexSlice[: , ['CELLETAL']], 'THI_Fahrenheit_sum_threshold72_12days'].groupby('CHR
ser_max_THI12_cell_coef.head().to_frame().style.hide_index()
```

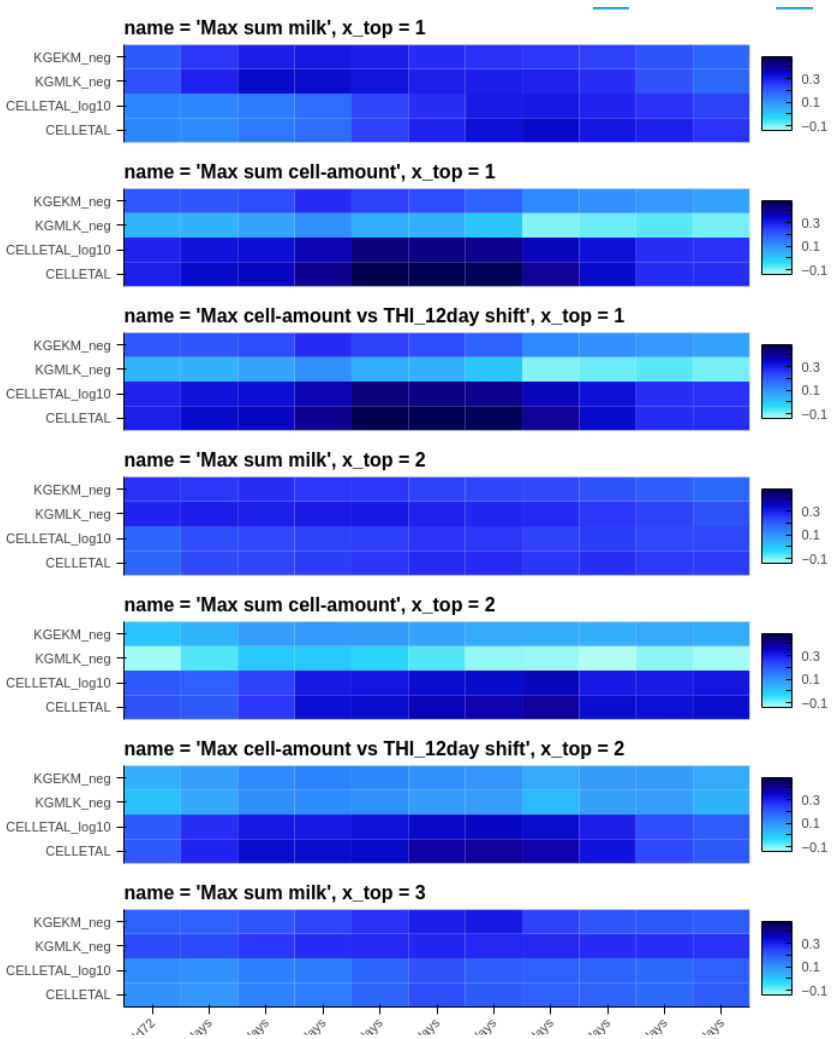
Out[33]:

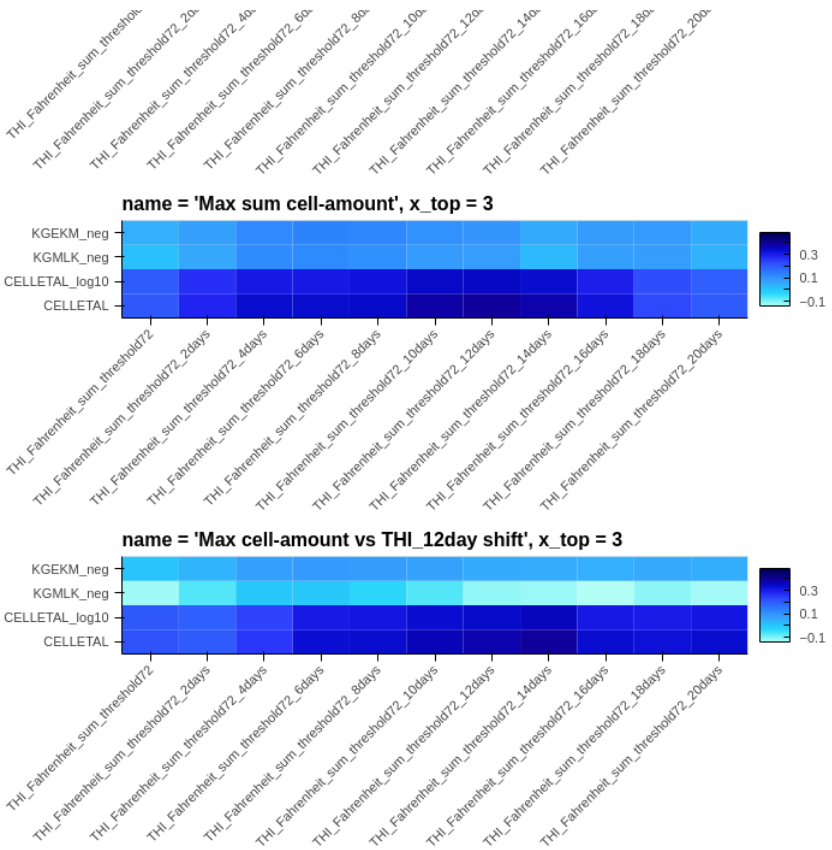
THI_Fahrenheit_sum_threshold72_12days
0.473549
0.391932
0.375035
0.373138
0.357515

In [34]:
```
CHR_max_THI12_cell_coef_top3 = ser_max_THI12_cell_coef.head(3).index.to_list()
```

In [35]:
```
# Visualize correlation coefficients over all THI-shifts for each top 3 CHRNR conditions.
plots = []
for i_top in range(0, 3):
    for name, CHR_top3 in [
            ['Max sum milk', CHR_max_sum_milk_coef_top3],
            ['Max sum cell-amount', CHR_max_sum_cell_coef_top3],
            ['Max cell-amount vs THI_12day shift', CHR_max_THI12_cell_coef_top3]]:
        CHR = CHR_top3[i_top]
        xaxis = None if i_top != 2 else True
        height = 120 if i_top != 2 else 280
        x_top = i_top + 1
        plots.append(df_dairy_THI_CHR_corr.loc[CHR].hvplot.heatmap(title=f'{name = }, {x_top = }', xaxis=xaxis, height=height).opts(xrotat
hv.Layout(plots).opts(shared_axes=True).cols(1)  # cols(3)
```

Out[35]:

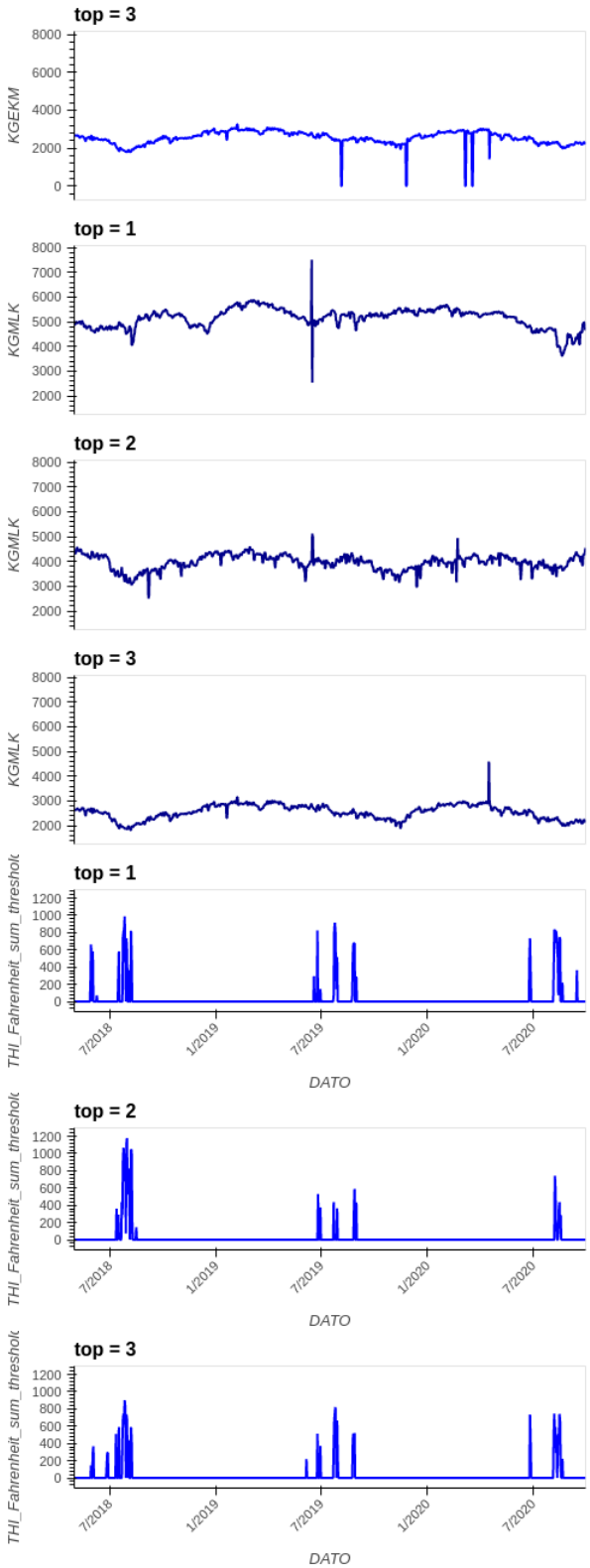




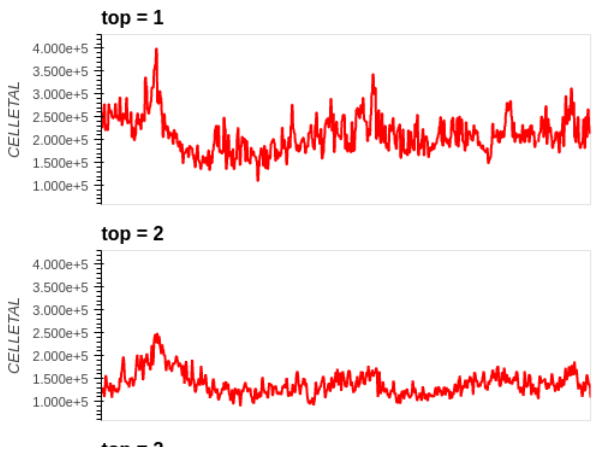
In [36]:

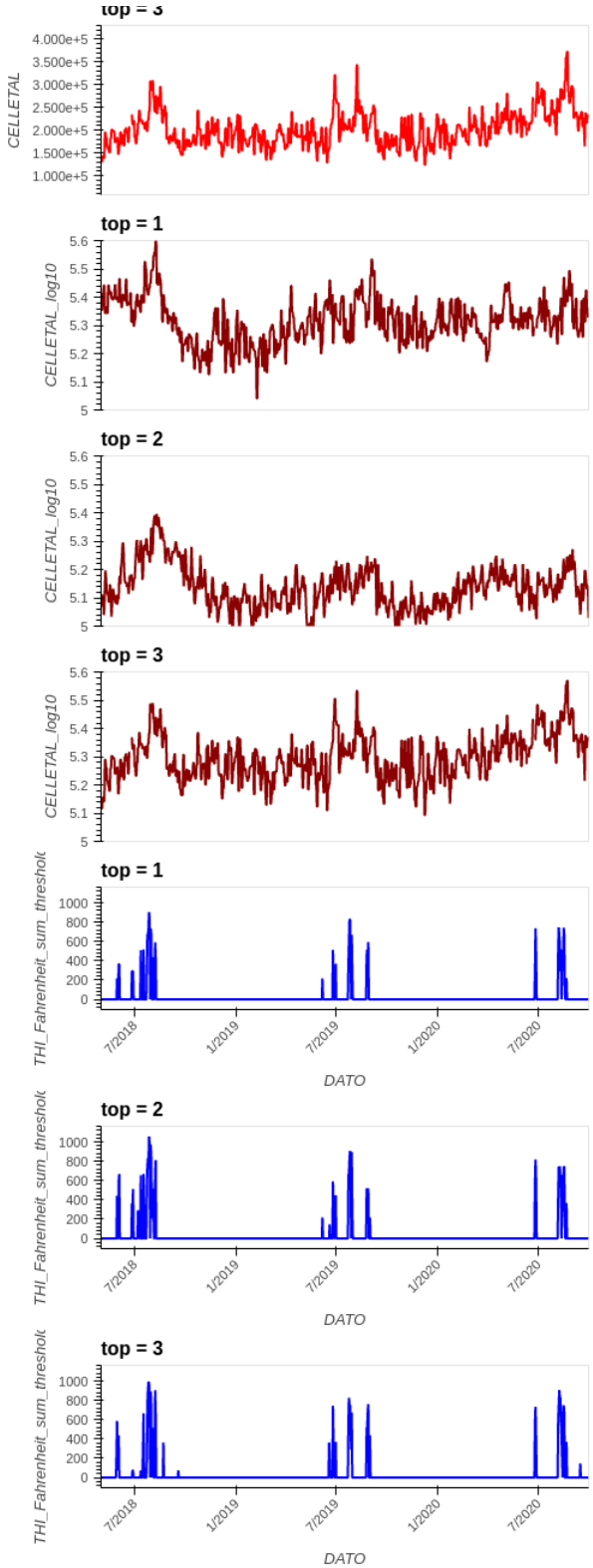
```
# Visualize relevant attributes over time for each top 3 CHRNRs conditions.
def plot_top3(top_3_list, select_att='cell'):
    if select_att == 'cell':
        col_color = {
            'CELLETAL': 'red',
            'CELLETAL_log10': 'darkred',
            'THI_Fahrenheit_sum_threshold72': 'blue'}
    elif select_att == 'milk':
        col_color = {
            'KGEKM': 'blue',
            'KGMLK': 'darkblue',
            'THI_Fahrenheit_sum_threshold72': 'blue'}
    else:
        raise NotImplementedError()

    plots = []
    for col in col_color.keys():
        xaxis = None if col != list(col_color.keys())[-1] else True
        lim = (100_000, 400_000) if col == 'CELLETAL' else None
        lim = (np.log10(100_000), np.log10(400_000)) if col == 'CELLETAL_log10' else None
        height = 180 if col != list(col_color.keys())[-1] else 200
        for i, CHR in enumerate(top_3_list):
            x = i + 1
            plots.append(df_dairy_THI_new.loc[CHR].hvplot(
                y=col, xaxis=xaxis, color=col_color[col], width=500, height=height, title=f'top = {x}', ylim=lim).opts(
                    xrotation=45))
    plot = hv.Layout(plots).cols(1)  # cols(len(top_3_list))
    return plot

for name, CHR_top3, select_att in [
            ['Max sum milk (look for drop in milk amount)', CHR_max_sum_milk_coef_top3, 'milk'],
            ['Max sum cell-amount (look for raise in cell amount)', CHR_max_sum_cell_coef_top3, 'cell'],
            ['Max cell-amount vs THI_12day shift (look for raise in cell amount)', CHR_max_THI12_cell_coef_top3, 'cell']]:
    print(f'{name = }')
    display(plot_top3(CHR_top3, select_att))
```

```
name = 'Max sum milk (look for drop in milk amount)'
```

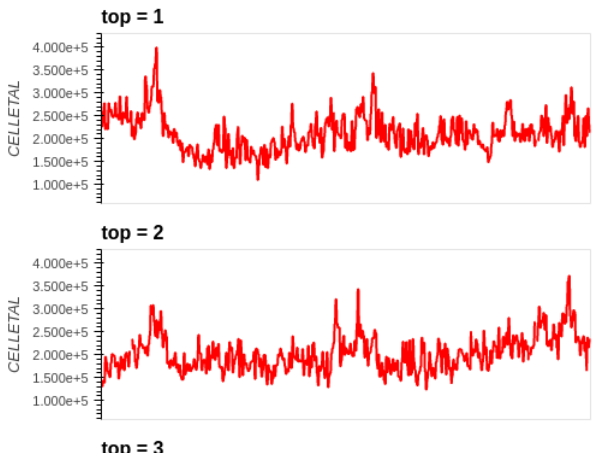


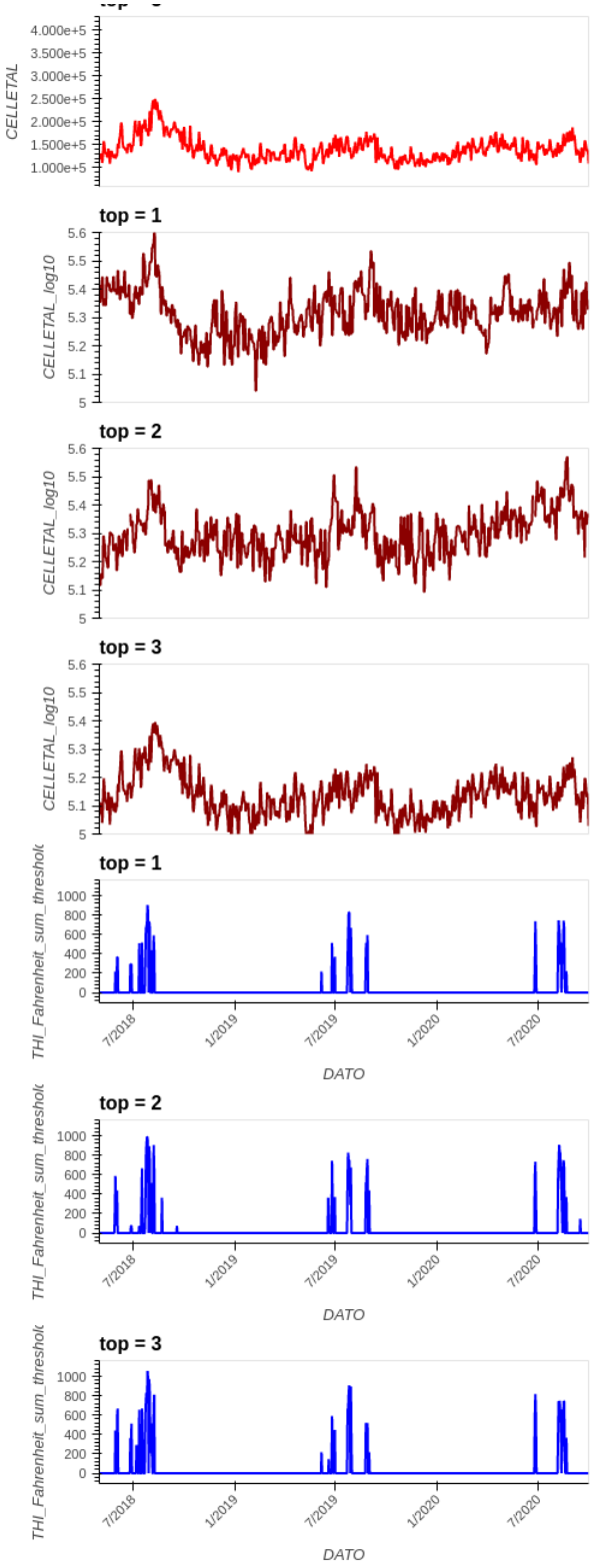

```
name = 'Max sum cell-amount (look for raise in cell amount)'
```






```
name = 'Max cell-amount vs THI_12day shift (look for raise in cell amount)'
```




CELLETAL
4.000e+5
3.500e+5
3.000e+5
2.500e+5
2.000e+5
1.500e+5
1.000e+5
top = 1
CELLETAL_log10
5.6
5.5
5.4
5.3
5.2
5.1
5
top = 2
CELLETAL_log10
top = 3
CELLETAL_log10
top = 1
THI_Fahrenheit_sum_threshold
1000
800
600
400
200
0
7/2018
1/2019
7/2019
1/2020
7/2020
DATO
top = 2
THI_Fahrenheit_sum_threshold
DATO
top = 3
THI_Fahrenheit_sum_threshold
DATO