Master Thesis

submitted within the UNIGIS MSc programme
Interfaculty Department of Geoinformatics - Z_GIS
University of Salzburg

“Assessing Aviation Safety comparing Fire related TFR NOTAMS in the U.S. to Satellite based Fire Patterns from OroraTech Wildfire Service”

by

Dipl.-Met.(FH) Christopher Heers
106658

A thesis submitted in partial fulfilment of the requirements of
the degree of
Master of Science – MSc

Advisor:

Dr. Lorenz Wendt

Lachendorf, September 15, 2022

Acknowledgment

Thanks to Dr. Lorenz Wendt for motivating me and accompanying me through this topic. It has been an exciting journey to combine my origin from aviation with content of the UNIGIS MSc program.

Data was provided by the OroraTech GmbH. You have been curious, responsive, and supportive. Working with you has been awesome. A special thanks goes to Max Helleis for being available all the time and a Jupyter Notebooks crash course.

Sincere thanks are given to Felix Schmidt for being the one colleague that has been able of keeping track of my code.

Thanks to the UNIGIS team for preparing me for this.

To Nicole Klotz: Thank you for recommending UNIGIS.

To my family: Thank you for continuous inspiration and support and for bearing my absence.

Christopher Heers

Lachendorf, September 15, 2022

Science Pledge

I hereby declare that the thesis is entirely the result of my work. I have cited all sources I have used in my thesis and indicated their origin. This thesis was not previously presented to another examination board and has not been published.

Christopher Heers

Lachendorf, September 15, 2022

Contents

Figures

Tables

1. List of Abbreviations

ARTCC	Air Route Traffic Control Center, operated by →FAA; identified by → ICAO location indicator; location to issue a →TFR via →NOTAM
EPA	Environmental Protection Agency in the United States
FAA	Federal Aviation Administration, regulates U.S. aviation, responsible for U.S. →FIRs
FIR	Flight Information Region, a division of airspace, identified by an →ICAO location indicator. Responsibility is delegated by →ICAO to other authorities like →FAA
ICAO	International Civil Aviation Organization, a United Nations organization, defines and provides standards for civil aviation, from →NOTAM format (not yet applied in the U.S.) to four letter location indicators, e.g. for →FIRs, and planes / transponders (ICAO 24-bit address)
NIFC	National Interagency Fire Center, supports emergency response with a focus on wildfires; consists of eight agencies, home of National Wildfire Coordinating Group →NWCG
NM	Nautical mile of 1852 meters, used in →TFR texts; not to be confused with the statute (or road) mile of 1609,34 meters, used for flight visibility and map scale bars in this research
NOTAM	Notice To Air Missions (sometimes Notice To Airmen), describes any conditions (other than weather) that can be hazardous to aviation. Is provided by aviation authorities.
NWCG	National Wildfire Coordinating Group, establishes standards for coordinated wildfire operation procedures, has 11 member agencies
TFR	Temporary Flight Restriction, a means with regulatory character to temporary close an airspace. Provided as →NOTAM text containing either a coordinate and a radius or an array of coordinates defining the closed area and explanatory text giving the reason.

2. Abstract

As a result of climate change, wildfires have become more dangerous, larger and more expensive than they have ever been before. Aerial firefighting is crucial for containing spreading wildfires. But it causes pilots to reach the limit of their skills. For risk prevention, airspace can be dedicated to fire fighting and temporary become closed for other traffic. As there can be no physical signposting mounted in the sky, knowing the when and where is necessary for preflight of other pilots. This is achieved by a Notice-to-Airmissions- or NOTAM-system that issues temporary flight restrictions (TFRs). TFRs close the airspace for manned aircraft as well as for drones.

The goal of this study was to examine the relation between satellite-based fire observation data and restricted airspace for aerial firefighting to assess safety of the involved aircraft. Early and precise airspace restriction allows aerial firefighters to focus on their mission and prevents them from collision with drones and other aircraft. The study area was defined to be the 10 westerly U.S. flight information regions (FIRs). Fire cluster polygons from that study area were considered active fires and drawn from OroraTech´s Wildfire Service. OroraTech is a NewSpace startup, founded in Munich in 2018. Their Wildfire Service provides early detection and real-time monitoring of fires, combining data from earth observation and weather satellites. Data can be consumed as GeoJSON via manual export and via an API. A complete set of TFR texts from the Federal Aviation Administration FAA via online NOTAM Archive for the study area was converted into GeoJSON format successfully, combining manual downloading with VBA, Power Queries and Python with GeoPandas. Aircraft state vectors (aircraft locations) of aerial fire fighters were available since August of 2021. Thus, the examined time frame was chosen to be August until (and including) October 2021.

All gathered data was examined then in ArcGIS Pro to design workflows to answer questions of spatiotemporal relations: How long does it take until a TFR is issued for a fire? Applying Python with GeoPandas and involving requests to OroraTech´s API, a model was created to find the first TFR issued for each fire event. Then, lists showing the timespan between fire detection and TFR issue time were generated. Out of 240 fires for which a first TFR got identified, 214 fires (89%) had been detected by the Wildfire Service prior to TFRs issue time.

Does a TFR area cover enough area to conduct aerial firefighting safely? A Python script, again with GeoPandas and requests to OroraTech´s API, delivered TFRs and fire events, where the aerial fire fighting area had not been covered properly.

These workflows only combining fire cluster and TFR polygons provided locations and times via both, fire cluster ids and TFR NOTAM numbers. These indicate *potential* cases where aerial firefighters had to work without a protective TFR dedicating the airspace to just firefighting.

Is the presence of aerial firefighters in the vicinity of a fire covered by a TFR? State vectors of 60 fire fighting aircraft within the observed time frame were analyzed. ArcGIS, Python with GeoPandas and this time, MovingPandas as well, were applied to unveil true situations where these aircraft had to operate unsheltered. 13.4% of the aircraft movements had not been flown in an airspace dedicated to only fire fighting aircraft. 42 of the 60 aircraft considered here had been uncovered by a TFR at least once within the observed three months.

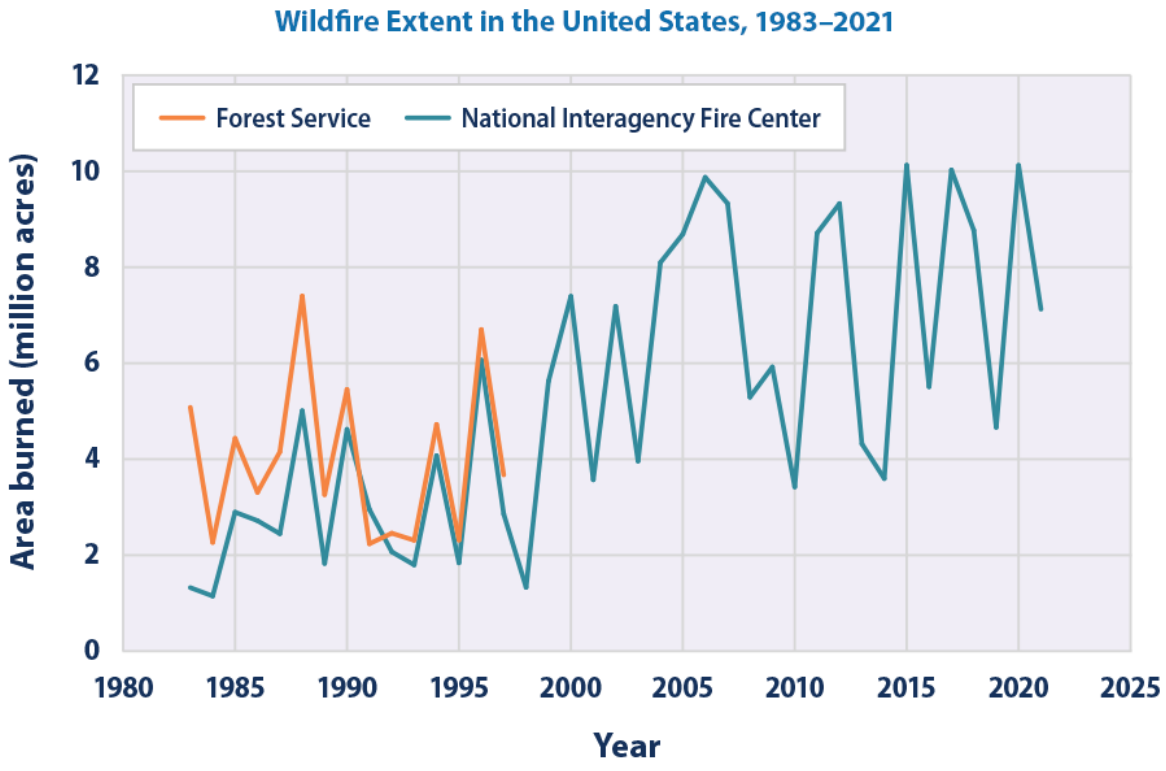
The completeness of TFR-fire-correspondence was examined to indicate that using fire clusters from the OroraTech as a kind of benchmark for the TFRs was justifiable.

All in all, there is room for improvement in airspace restriction for aerial firefighting and data from OroraTech can foster further research and TFR management and thus enhance aviation safety.

Keywords: *Active fire detection , aerial firefighting , air tanker , aviation safety , flight restriction , forest fire , hot spot detection , NOTAM , spatiotemporal analysis , TFR , wildfire*

3. Introduction

Dangerous and large wildfires, as a result of climate change, have been predicted for the United States, Tang et al. (2015, p. 19). Data collected by the United States Environmental Protection Agency (EPA) documents the increasing wildfire extent and the surge in damage wildfires are causing, US EPA (2021), see Figure 1.



Data sources:
- NIFC (National Interagency Fire Center). 2022. Total wildland fires and acres (1983–2022). Accessed June 2022. www.nifc.gov/fireInfo/fireInfo_stats_totalFires.html.
- Short, K.C. 2015. Sources and implications of bias and uncertainty in a century of U.S. wildfire activity data. Int. J. Wildland Fire 24(7):883–891.

For more information, visit U.S. EPA's "Climate Change Indicators in the United States" at www.epa.gov/climate-indicators.

Figure 1 The total wildfire extent increases in the U.S.

Dealing with wildfire suppression, Wotton et al. (2017, p. 3) recaps a classification of fires describing the limit of ground resources without aerial support (and also the limit of the latter). With aerial firefighters being crucial, their safety within the concerned airspace is to be granted. Satellite data considered as actual fires here was compared to airspace restrictions to find out if aerial firefighters had been working without being in danger to collide with other aircraft or drones. With the availability of near-real-time fire detection data from OroraTech, the idea arose to make a first step to evaluate a possible application that requires currentness of data: The timely reservation of airspace to safely perform aerial firefighting. This research can help authorities to decide whether they can use OroraTech data to effectively assign temporary flight restriction zones faster than by the current procedures.

This thesis tries to unveil certain spatiotemporal conditions. In geographic information science, this has been done for a long time now with point data, e.g. Knox & Bartlett (1964). For polygon data that is involved in all analyses here, only few methods are known to manage the time factor, Robertson et al. (2007, p. 209). To represent specifically wildfires, YUAN (1997, p. 732ff) identifies four conceptual models (Text 1). Models 2 and 3 apply for vector data like it was used here[1]. Originally meant for GIS layers, these were used to characterize the data from OroraTech (4.1.1).

> "1. Locational snapshots: layers of cells (spatial objects) are linked to points in time (temporal objects) and attributes of fuel moisture content, slope, etc. (semantic objects).
>
> 2. Fire entities: a fire event (a semantic object) is linked to a set of points or duration in time (temporal objects) and areas of fire runs (spatial objects).
>
> 3. Entity snapshots: fire events (semantic objects) are linked to a point in time (a temporal object) and areas of fire runs (spatial objects); note that entity snapshots differs from fire entities because entity snapshots temporally aggregate all fire runs from the start of a fire event to the time specified.
>
> 4. Fire mosaics: a set of landscape patches (spatial objects) is linked to points in time (temporal objects) and attributes of vegetation types, evidence of burns, etc. (semantic objects)."

Text 1 Conceptual models for fire representation in a GIS layer, from YUAN (1997).

All cited regulations and directives have been read regarding U.S. airspace. This chapter summarizes the state of research concerning active fire detection, starting with a few words on OroraTech Wildfire Service. The following section introduces the terms needed in connection with airspace reservation. Then it is time to dedicate some lines to those, whose safety is on trial here: Aerial firefighters.

Regarding Ostermann et al. (2020), to enable reproducibility, all used Python scripts are provided as Jupyter Notebooks, see Kluyver et al. (2016). Other code shown inside the thesis is included without line numbering, so it can be copied "ready to use". Absolute file paths are contained but evaded wherever possible. Geoprocessing tool settings are provided as screenshots to not miss any checkbox or other detail.

[1] Model description uses the term "fire run" which is defined as "The rapid advance of the head of a fire with a marked change in fire line intensity and rate of spread from that noted before and after the advance" - https://inciweb.nwcg.gov/terminology/

3.1. OroraTech Wildfire Service

For early forest fire detection, satellite data is the first choice concerning coverage area while there is room for improvement in response time compared to other systems. Barmpoutis et al. (2020, p. 14) provide a radar chart comparing different systems, see Figure 2.

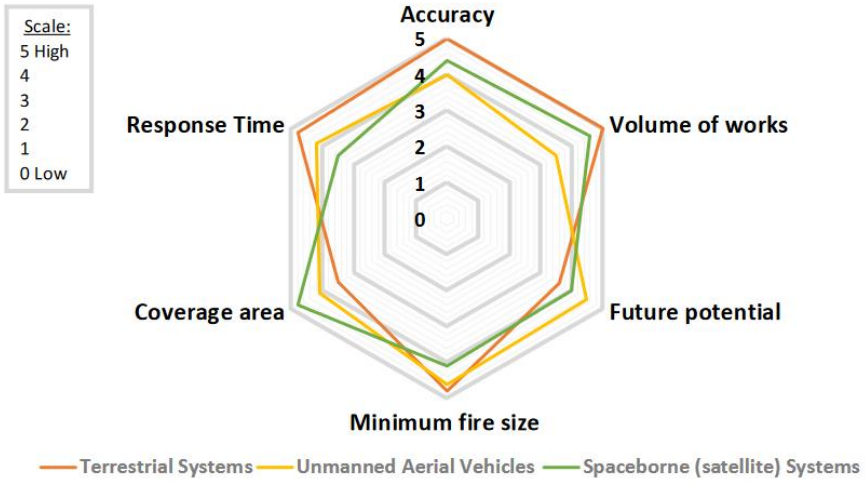


Figure 2 Radar chart comparing fire detection systems in six categories

Founded 2018 in Munich, the OroraTech GmbH provides satellite based near real time active fire detection data via its Wildfire Service. With regard to the incorporated satellites and sensors, Hantson et al. (2013) documented opportunities and limitations of MODIS (Moderate Resolution Imaging Spectroradiometer) hotspot data, comparing it to burned areas. Schroeder et al. (2014, pp. 94–96) did an assessment for VIIRS (Visible Infrared Imaging Radiometer Suite) showing more accurate fire spread information than MODIS (however, not quantifying their results but providing images) and even challenging VIIRS with a 1.25 m radius experimental fire burning at ~ 1000 K that got detected during nighttime.

One current data source for fire detection is the EU's European Forest Fire Information System (EFFIS), European Commission (n.d.-b). EFFIS relies on MODIS and VIIRS, European Commission (n.d.-a). The NASA and USDA Forest Service initiative "Fire Information for Resource Management System" (FIRMS) provides hotspots from MODIS and VIIRS as well[2]. A third source is FIRECAST[3], also relying on MODIS and VIIRS and additionally offering to filter by confidence. With MODIS and VIIRS being the oldest instruments in use for hot spot detection, the concept of OroraTech to bring these together with all available recent work is considered to be a success. OroraTech offers data from more satellites, more algorithms and a sophisticated web application interface:

As of July 2022, the Wildfire Service incorporates 21 satellites (Table 1). SLSTR from SENTINEL, evaluated by Wooster et al. (2012), is involved as well as Landsat 8 that has proven its usability, assessed by Schroeder et al. (2016, p. 218). OroraTech uses existing algorithms side by side with contextual thresholding algorithms developed in-house. Table 2 shows the algorithms and third-party products in use. The Wildfire Service will soon get strengthened by OroraTech´s own nanosatellites to

[2] FIRMS: https://firms.modaps.eosdis.nasa.gov/usfs/
[3] FIRECAST: https://firecast.conservation.org/DataMaps/LiveView

facilitate real-time response and reach a minimum viable constellation by the end of 2023[4], compensating in all categories that Barmpoutis et al. (2020) distinguished. The first of its kind satellite was successfully launched in January 2022 and will become part of a fleet of 100 satellites over the following years. The Wildfire Service rejects false positives, aggregates spatiotemporal hotspots and builds clusters identifying coherent fire pixels[5]. These fire clusters were used in this thesis, as well as OroraTech´s API to access a fire cluster´s composition of hotspots within a certain time interval, where this became necessary. Figure 3 Shows a fire cluster from the Wildfire Service and its web application interface, using only data from polar orbiting satellites. That particular fire cluster was examined later in detail exploring the datasets in section 4.2.3.2.

Table 1 List of satellites incorporated into Wildfire Service

Satellites incorporated into Wildfire Service	
Polar orbiting	**Geostationary**
AQUA	GK2A
FENGYUN-3D	GOES-16
LANDSAT-8	GOES-17
LANDSAT-9	Himawari-8
Met-Op-B	Meteosat-8
Met-Op-C	Meteosat-9
NOAA-20	Meteosat-10
SENTINEL-2A	Meteosat-11
SENTINEL-2B	
SENTINEL-3A	
SENTINEL-3B	
SUOMI-NPP	
TERRA	

[4] Press release: https://ororatech.com/wp-content/uploads/2022/01/OroraTech-Press-Release-First-Satellite-Launch.pdf

[5] https://ororatech.com/wildfire-service/

Table 2 List of Wildfire Service active fire detection algorithms and products

Wildfire Service active fire detection algorithms and products	
Algorithm or product	**Reference**
OT-S (OroraTech-Sentinel in-house)	Based on Wooster et al. (2012)
OT-V (OroraTech-VIIRS in-house)	Based on Schroeder & Giglio (2017)
OT-SWIR (OroraTech in-house)	Based on Schroeder et al. (2016)
OT-AI (OroraTech in-house)	Based on de Almeida Pereira et al. (2021)
MODIS-Collection6-Active-Fire-Product	Giglio et al. (2016) and https://www.earthdata.nasa.gov/learn/find-data/near-real-time/firms/mcd14dl-nrt
VIIRS-Active-Fire-Product	Schroeder et al. (2014) and https://www.earthdata.nasa.gov/learn/find-data/near-real-time/firms/viirs-i-band-375-m-active-fire-data
SENTINEL-FRP	Wooster et al. (2012) and https://www.eumetsat.int/S3-NRT-FRP
GOES (ABI-L2-FDCF-M6 ABI Level 2 Fire/Hot Spot Characterization product)[6]	Hall et al. (2019) and https://data.noaa.gov/dataset/dataset/noaa-goes-r-series-advanced-baseline-imager-abi-level-2-fire-hot-spot-characterization-fdc
SRSS-Himawari-8[7]	https://www.eorc.jaxa.jp/ptree/documents/README_H08_L2WLF.txt
EUMETSAT_FIRG / EUMETSAT_FIRC[8]	https://www-cdn.eumetsat.int/files/2020-04/pdf_fir_pg.pdf

[6] GOES is the U.S. geostationary weather satellite program. No hotspots generated from geostationary satellites ´data were used in this research

[7] Himawari is a Japanese geostationary satellite, and its data was not used in this research

[8] These products are derived from data of European geostationary satellites not covering the U.S.

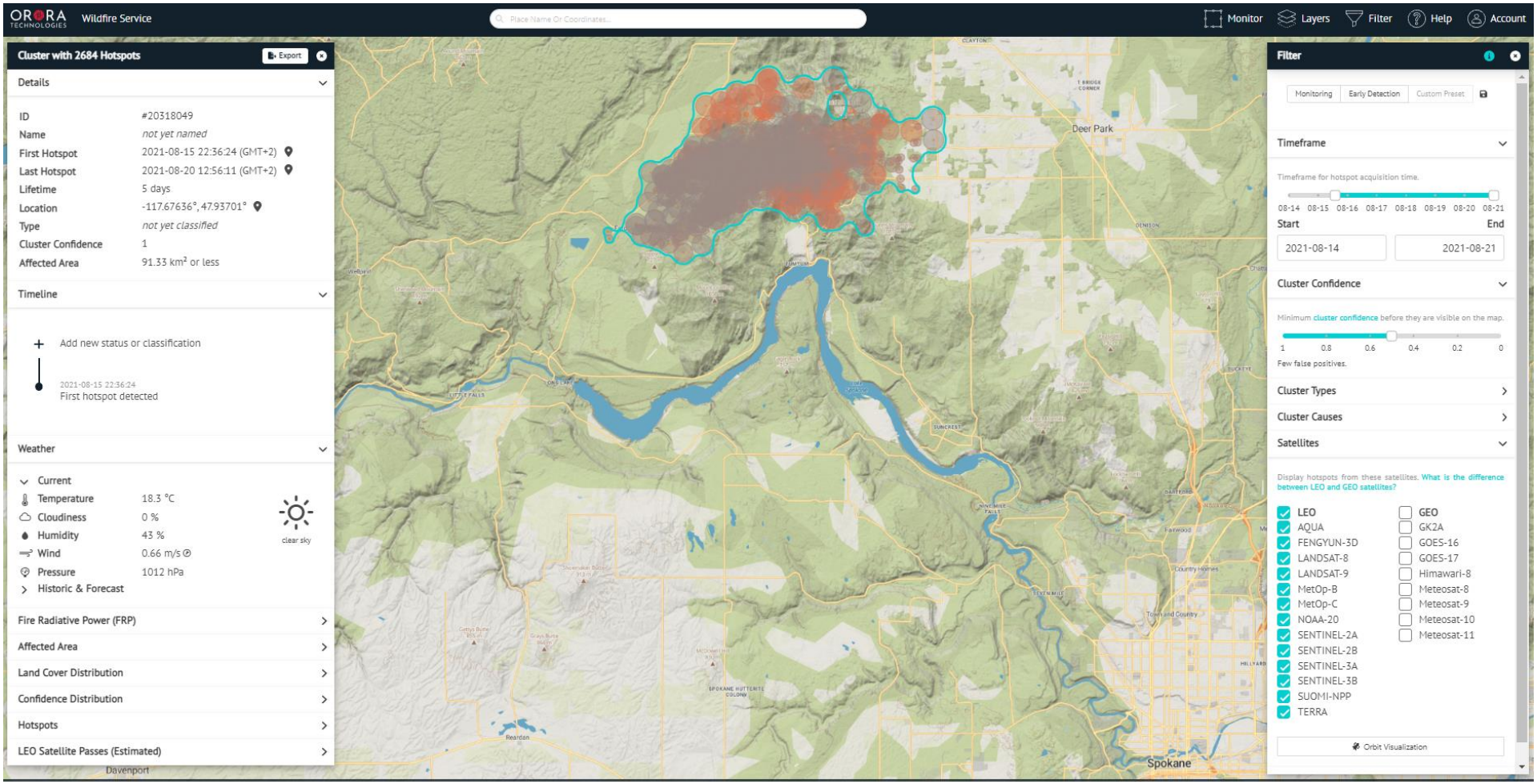


Figure 3 A fire cluster in the northwest of Spokane shown in the Wildfire Service web app

3.2. Notices to Airmissions and Temporary Flight Restrictions

A s there can be no road signs mounted somewhere in the sky, pilots need to consume aeronautical information concerning their route before they take off. Notices to Airmissions (NOTAM) provide many kinds of information. One is the Temporary Flight Restriction (TFR) that temporarily closes an airspace or restricts its usage to certain purposes. The NOTAM system is also used to restrict airspace for aerial firefighting in case of wildfires, FAA (2022a), FAA (2021). Authorities are obliged to monitor all issued wildfire TFR, USDA, Forest Service (2021). The TFRs can have circular or polygonal shape and can cover multiple fire events by intention. The goal for authorities is not more NOTAM data, but high precision: Hoeft et al. (2005, p. 92) mention the overwhelming information amount that pilots receive before takeoff, which has a negative impact on flight safety. Safety is improved by reduction of bumped up briefing packages[9]. There have been incidents, also within the data examined here, that TFRs got cancelled even before they became effective (valid), thereby populating the NOTAM system without having any meaning.

National Wildfire Coordinating Group (2018, p. 7) points out that TFRs have regulatory character, in contrast to advisory NOTAMS. This is the basis for law enforcement against any violators, be it pilots of drones/unmanned aircraft systems (UAS) or pilots of manned aircraft. With wildfires being events attracting media and other airborne spectators, a timely issued information *and* deterrent against misbehavior seems necessary: It is a fact that especially UAS traffic has disrupted several aerial firefighting activities year after year[10]. Drone pilots can easily inform themselves via app[11] – but only, if proper TFRs exist at all.

National Wildfire Coordinating Group (2018, p. 112f) advises to issue a wildfire TFR with a radius of at least 5 nautical miles (NM) while it is recommended at the same time, to tailor TFRs to the individual incident´s needs. Consequently, a TFR can have a circular or an angular shape. Currently, the process of a TFR creation is complex: It involves a form to be filled out and sent by the responsible local authority or agency[12], then the requested TFR needs to pass a list of criteria determining its need and the TFR´s extent can finally become subject to negotiations with the FAA, National Wildfire Coordinating Group (2018, p. 111ff). The National Wildfire Coordinating Group (2022a, p. 118) rates hazards in connection with TFR (e.g. if TFR is not promoted properly or an incident expands) as “catastrophic” in severity. This is reason enough to study the accuracy of wildfire related TFRs.

3.3. Aerial Firefighting

The basic wildfire-fighting strategy is described by Mutthulakshmi et al. (2020, p. 646): By aerial water bombing, it is possible to temporary contain the spread of a fire by creating holding lines. These persist as long as the water, foam or (the mostly red) fire retardant does reduce the flammability of the vegetation. So, aircraft are rarely capable of putting out a fire entirely, but their key ability is to

[9] https://www.avweb.com/aviation-news/icao-updates-effort-to-clean-up-notam-garbage/

[10] Interfering drones reports:
2016 https://www.cnbc.com/2016/07/26/feds-turn-up-the-heat-in-the-fight-against-drones-interfering-in-wildfires.html
2017 https://www.popularmechanics.com/technology/gadgets/a27273/drones-stopping-aerial-firefighting/
2018 https://www.popularmechanics.com/flight/drones/a21599465/drones-interrupt-fire-fighting-wildfires/
2019 https://weather.com/news/news/2019-11-02-drones-grounded-firefighting-aircraft-maria-fire

[11] FAA app https://www.faa.gov/uas/getting_started/b4ufly

[12] TFR request form is available here: https://www.nwcg.gov/sites/default/files/committee/docs/iasc-interagency-tfr-request-form.pdf

maintain control over a wildfire (early detected, at its best) until ground forces can reach it. The aircraft fly the so called "initial attack"[13]

Various types of aircraft are in use to perform aerial firefighting. From helicopters to a Boeing 737 all sizes and types are observed within the data used here. Unlike in commercial aviation, firefighting aircraft pilots must take care of separation (meaning, to keep a secure distance to other aircraft), so even if they are among themselves, airspace coordination is fundamental to preserve safety, National Wildfire Coordinating Group (2022a, p. 93). Amongst others, this is a reason to set up a Fire Traffic Area (FTA) with a horizontal radius of 5 NM and 2500 feet vertical extension, National Wildfire Coordinating Group (2022b). Another characteristic of aerial firefighting is the altitude, low as 60 meters, to fly at while dropping water. The bigger the aircraft, the more unusual it is for it to fly at close-to-ground-altitudes. While having reached an experimental stage recently, nighttime aerial firefighting is still not a common practice[14] due to its riskiness.

Even before UAS hit the airspace, back in 1998 surveyed aerial firefighting personnel considered airspace intrusion being a risk, USDA FOREST SERVICE, DEPARTMENT OF INTERIOR (1998, p. 7). The same study describes the workload of fire fighting pilots being at the upper limits of human capability, so there is no room for any distraction caused by uninvolved aircraft. UAS are considered an aircraft as well by the FAA and must not intrude a TFR, National Wildfire Coordinating Group (2018, p. 130). In recent years, UAS have repeatedly delayed or even hampered aerial fire fighting operations (see also above in section 3.2).

Aside from dropping water or fire retardants, an aerial supervision is performed (and mandatory under certain circumstances). National Wildfire Coordinating Group (2022a, p. 36) attaches importance to visibility which must meet FAA Visual Flight Rules (VFR). The following link and Figure 4 show typical patterns of movement during aerial firefighting: https://graphics.reuters.com/CALIFORNIA-WILDFIRE/AIRCRAFT/bdwpkzmyyvm/.

[13]As described in https://priceonomics.com/does-using-airplanes-to-put-out-forest-fires/

[14]Nighttime aerial firefighting needs technical improvements allowing for higher drop altitudes, https://www.optimistdaily.com/2020/01/aerial-firefighting-at-night-is-now-possible-with-new-high-altitude-drop-system/, and/or better night vision, https://aerialfiremag.com/2020/03/23/night-aerial-firefighting-taking-the-fight-24-7/

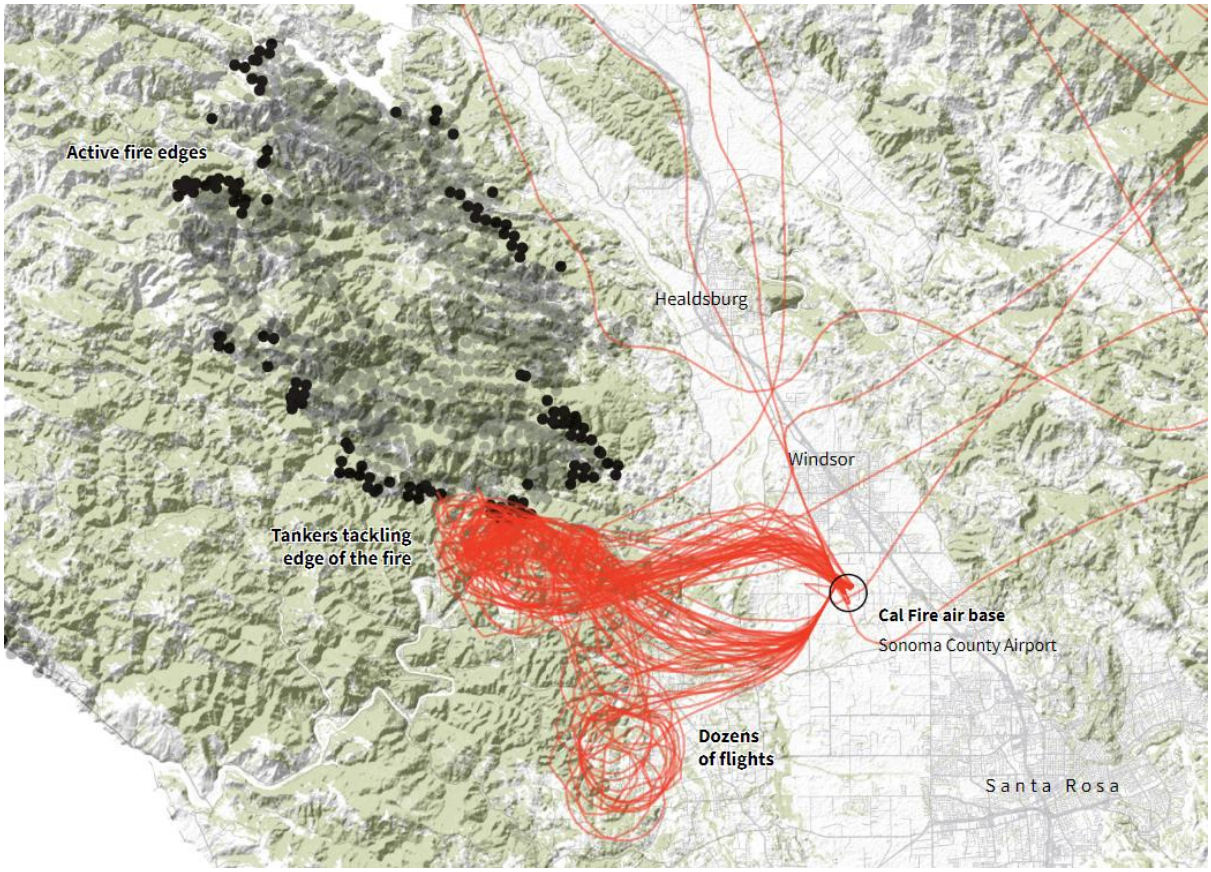


Figure 4 Movement patterns during aerial firefighting from https://graphics.reuters.com/CALIFORNIA-WILDFIRE/AIRCRAFT/bdwpkzmyyvm/

3.4. Study Area: 10 Flight Information Regions in the western U.S.

Flight Information Regions (FIR) are divisions of airspace to allow for administration and air traffic control. The study area being the western part of the United States is due to this EPA data showing the most annual burned acreage there, US EPA (2021). Data from three months, August 2021 until October 2021 was collected here. A map is provided in Figure 14 in the corresponding data section 4.1.4.

3.5. Contribution, Research Questions and Overview

This thesis can show to airspace authorities whether incorporating data from OroraTech facilitates their activities fostering aviation safety. Readers from the GIS community find out about application of their work to the aviation domain. The results can unveil situations, where a TFR was not appropriate and could not guarantee aircraft safety. This study is research for relations between satellite detected fire clusters (considered actual fires), state vectors (locations) of known aerial firefighting planes and fire related TFR:

(1) Complete TFR data of 10 FIRs from August to October 2021 got examined and converted from text into a spatial data format. (2) Rules for appropriateness of a TFR were defined. Questions and objectives were:

(3) The now available datasets got explored to gather an understanding of the relation between fires, fire fighting aircraft and TFRs.

(4) How long was the timespan between satellite detection and TFR issue time? If this took too long, the initial attack was probably launched while there was no safe airspace provided.

(5) Did a TFR area cover enough area to conduct aerial firefighting safely? Did TFR area cover fire pattern plus minimum VFR flight visibility (1 or 3 miles, according to airspace)? If a TFR was not setup properly or a fire spread out of it, the TFR might not have protected firefighting aircraft anymore.

(6) Evaluate whether or how often the presence of aerial firefighters in the vicinity of a fire was not covered by a TFR. With state vectors of 60 firefighting aircraft detected in the vicinity of active fires available, their actual safety during multiple aircraft movements was assessed.

During the work with the data, the impression arose that TFRs could be issued at erroneous coordinates or OroraTech´s Wildfire Service provides no fire clusters for these coordinates. So, the last objective was:

(7) Did each TFR have a corresponding fire cluster? A TFR without a fire cluster was likely issued in the wrong place or may indicate a blind spot of OroraTech´s Wildfire Service. The completeness of TFR-fire-correspondence was examined. Putting TFRs on trial, the suitability of the compared fire clusters should be acceptable.

The upcoming chapter 4 describes the data and its acquisition in section 4.1, followed by methods used (section 4.2). The methods section explains how TFR texts were turned into spatial data format and which steps were necessary to answer the objectives. Finally, results are presented and discussed (chapter 5), before the conclusion (chapter 6) that gives a summary and provides ideas for further research. Chapter 7 holds the references. Chapter 8 is the appendix, containing code, a GitHub link, produced tables and data logs.

4. Data and Methods

The first section of this chapter describes the data used in this thesis, going through the four input data types and sources one after another. The second section explains the workflow, the tools used and the careful considerations that must be made.

4.1. Data

Only the fire clusters from OroraTech could directly be displayed in a GIS or undergo spatial operations with Python modules. All other data had to be converted to vector data first. But that is not the only reason, why fire clusters from OroraTech are valuable. TFR NOTAMs are plain text and fire fighting aircraft comes as point data while FIRs are to be traced from ArcGIS Map Viewer to gather polygons. Aircraft state vectors (aircraft locations) of aerial fire fighters were available since sometime in July of 2021. Thus, the examined time frame was chosen to be August until (and including) October 2021 for all data incorporated by this thesis.

4.1.1. Fire Clusters from OroraTech as Polygons

Before data from OroraTech got on hand, "classic" fire perimeters were available. For the study area here, the National Interagency Fire Center (NIFC) is responsible and publishes fire perimeters[15]. They can be expected to be more precise than hotspot data from satellites, so there is a need to explain why they were not used here. The fire perimeters have an essential disadvantage: Concerning decisions on airspace restriction, fire perimeter data suffers from its idleness. Perimeters are created from GPS walks (in case of small fires) as well as from satellite images even months after the event. Robertson et al. (2007) conducted a case study of the 2003 Ball Creek Fire with dense data from a local authority, where fire perimeter data from approximately every second day was available. But this temporal resolution is still ineligible for imposing or supervising airspace restrictions. Furthermore, fire perimeters may be incomplete. Several events from 2021 are still not covered yet by NIFC fire perimeters created until May 2022 which is documented by the perimeter attribute "Polygon Create Date".

Fire clusters from OroraTech are assembled from fire detections by multiple satellites. For this study, data from polar orbiting satellites was chosen. Their consistent higher accuracy[16] compared to geostationary satellites leads to applicable fire clusters for a decent comparison to TFRs. Data was provided in GeoJSON format, either from manual export from a web map interface or via an API[17]. Speaking in conceptual models for GIS layers (see Text 1), the manual export was rather an "Entity Snapshot" representing the largest aggregation of hotspots at the end of the configured timeframe, while the API allowed for going towards "Fire Entities", being able to fetch each fire cluster within each timeframe (within the temporal resolution of the satellite data). A manually exported dataset over the whole observation time will be referred to as "three-month-dataset" while results from API requests will be mentioned as "current" or just "API data" in this study. Figure 5 shows a fire cluster polygon in ArcGIS Pro.

[15] NIFC fire perimeters: https://data-nifc.opendata.arcgis.com/datasets/nifc::wfigs-wildland-fire-perimeters-full-history/explore?location=44.044862%2C-116.209075%2C6.66

[16] VIIRS has a spatial resolution of 375 m and is capable of detecting even small fires, as mentioned in the introduction at 3.1, while thermal channels from geostationary satellites are at 2-5 kilometers.

[17] Having an account for the Wildfire Service, the API documentation can be found here: https://app.ororatech.com/api

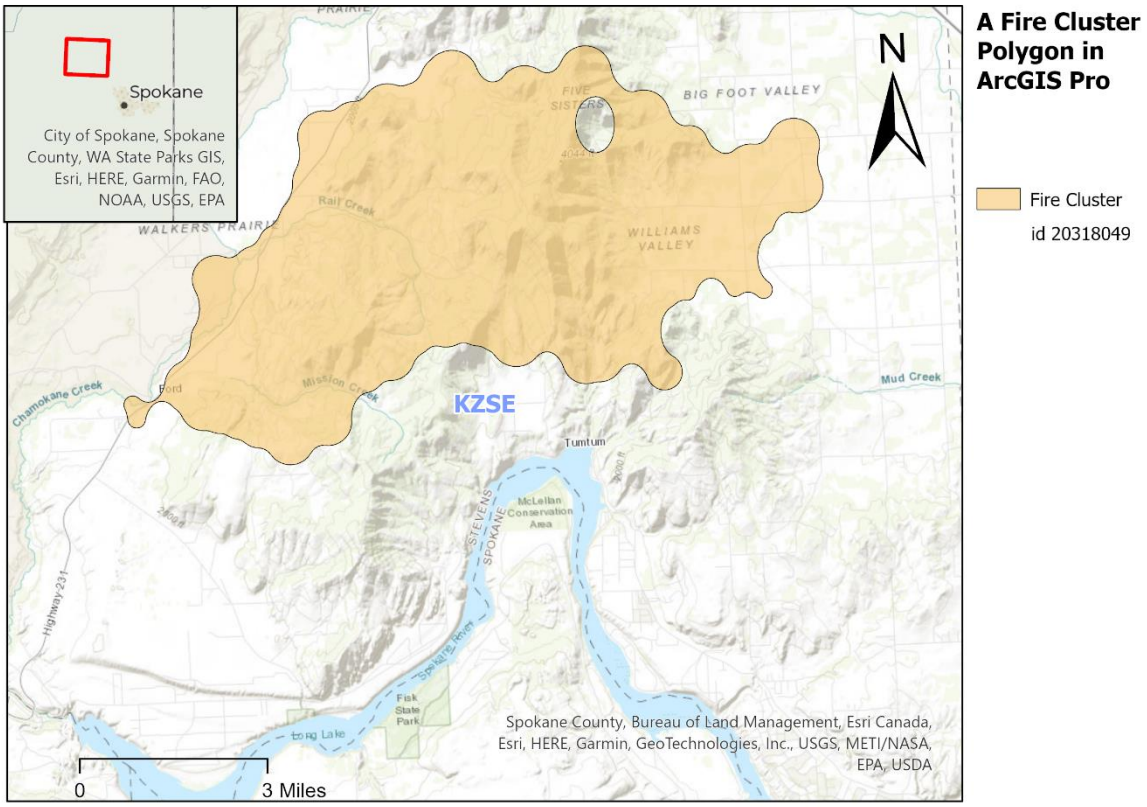


Figure 5 A fire cluster polygon in ArcGIS Pro

The available attributes include a cluster id. Another important attribute is confidence which can take a value between 0 and 1 (in tenth part increments). Basically, it describes how certain it is that an actual fire got detected. This work followed recommendation from OroraTech to use a minimum confidence value of 0.5 for analysis of historical data. Further attributes are oldest and newest acquisition as datetime objects, which means when a fire got captured by a satellite, and oldest and newest detection, which means, when the data was downlinked and processed, so one could have known about the fire from then from the satellite data. (The gap between acquisition and detection is the timespan OroraTech aims to minimize by its own nanosatellites. Currently, for Sentinel-2, this may be up to 12 hours). An important detail (for 5.1) is the technical functionality of the API: Requesting OroraTech´s API uses acquisition time of single hotspots for filtering (while the oldest acquisition time of a fire cluster always remains the same acquisition time of its very first hotspot. When hotspots within the filtered acquisition time range are found, it dynamically fetches the corresponding oldest detection time (which can lead to oldest detection times in the future, from a search window´s perspective). Such API requests were used here (4.2.4 and 4.2.5). Concerning the shape, a fire cluster consists of circular polygons derived from the detected hotspots (attribute: num_fires) which are based on the ground sampling distance (GSD) of the detecting satellites. A types attribute whose values are assigned by OroraTech´s user community, describes the source of a fire cluster. Values and meaning are listed by Table 3.

Table 3 The cluster types and the according value of the "types" attribute within OroraTech´s fire cluster data

Cluster Type	Value
Empty	0
Fire	1
Industry	2
Volcano	3
FalseDetection	4
Forest	5
Cropland	6
Grassland	7
Structure	8
Factory	9
Steel	10
Flare	11
Cement	12
Peat	13
Natural	14
NaturalOther	15
Reflection	16
Solar	17
Sun	18
Processing	19

4.1.2. NOTAM Texts describing TFRs

Complete TFR data from 10 flight information regions (FIR) during August to October 2021 got examined. The U.S. TFR NOTAMS are issued in a format called "domestic". That means, they do not have to comply entirely with ICAO rules. Automatically processing the domestic format in consequence needed flexible parsing algorithms. If only a contact information changes, a TFR is cancelled and a new one is issued, most likely in the same location.

4.1.2.1. How to read a TFR NOTAM

An original TFR NOTAM text from the assessed dataset is provided with Text 2.

!FDC 1/8922 ZSE WA..AIRSPACE 22NM NW OF SPOKANE, WA..

TEMPORARY FLIGHT RESTRICTIONS WI AN AREA DEFINED AS 7NM RADIUS OF 475345N1174830W (GEG319021.2) SFC-7500FT. TO PROVIDE A SAFE ENVIRONMENT FOR FIRE FIGHTING AVIATION OPS. PURSUANT TO 14 CFR SECTION 91.137(A)(2) TEMPORARY FLIGHT RESTRICTIONS ARE IN EFFECT. THE WASHINGTON DEPARTMENT OF NATURAL RESOURCES

TEL 509-685-6900 OR FREQ 126.8250/THE FORD CORKSCREW FIRE IS IN CHARGE OF THE OPS. SEATTLE /ZSE/ ARTCC TEL 253-351-3698 IS THE FAA CDN FACILITY. DLY 1600-0400 2108161600-2109010400EST

Text 2 An original TFR NOTAM text, here for a circular polygon defining a radius around a center coordinate

The NOTAM gets decoded according to Order 7930.2, FAA National Headquarters (2021):

!FDC states, this is a flight data center NOTAM (and thus regulatory).

1/8922 ZSE WA: Number "1" tells the issue year – 2021 here. "8922" is a sequential number. Together they represent a kind of ID (within ten years). "ZSE WA" reveals, the TFR is within airspace taken care of by the Seattle Air Route Traffic Control Center (ARTCC ZSE, equals: FIR KZSE) and the restriction is found in the state Washington (WA)

AIRSPACE 22NM NW OF SPOKANE, WA..: This a NOTAM of type "AIRSPACE" (not to be confused with its subtype described in the following paragraph that can be "TEMPORARY FLIGHT RESTRICTIONS" but "AIRSPACE" once more as well), so it provides information about an area. This area is centered approximately 22 nautical miles northwest of the city of Spokane, Washington.

TEMPORARY FLIGHT RESTRICTIONS WI AN AREA DEFINED AS 7NM RADIUS OF 475345N1174830W (GEG319021.2) SFC-7500FT. – So, this is a TFR. "WI" means within. The area affected in this case is a circle with 7 NM radius around a WGS84 DMS coordinate. In brackets follows a coordinate format relative to a navigation aid/beacon or airport, here 21.2 NM int the direction of 319 degrees from Spokane airport, location indicator (K)GEG (this may not accurately match the given coordinate). Then, the vertical bound is given as SFC (surface) up to 7500 feet above mean sea level.

TO PROVIDE A SAFE ENVIRONMENT FOR FIRE FIGHTING AVIATION OPS. PURSUANT TO 14 CFR SECTION 91.137(A)(2) TEMPORARY FLIGHT RESTRICTIONS ARE IN EFFECT. THE WASHINGTON DEPARTMENT OF NATURAL RESOURCES – Key sentence with justification and authority citation, followed by responsible authority: Washington Department of Natural Resources.

TEL 509-685-6900 OR FREQ 126.8250/THE FORD CORKSCREW FIRE IS IN CHARGE OF THE OPS. SEATTLE /ZSE/ ARTCC TEL 253-351-3698 IS THE FAA CDN FACILITY. Provides contact information via phone and radiocommunication. Ford Corkscrew Fire is the fire brigade or contractor in charge, Seattle ARTCC is the FAA coordinating (CDN) facility.

DLY 1600-0400 2108161600-2109010400EST The NOTAM (and therefor the TFR) is active daily from 1600 to 0400 UTC, in 2021 from August 16th, 1600 UTC to September 1st, 0400 UTC, where "EST" means "estimated". (With Spokane, WA being 7 hours behind UTC during summer, this means, most of the daylight time is covered which goes with the fact that most aerial firefighting is performed in broad daylight.)

4.1.2.2. How to obtain TFR NOTAMs

Current and active TFRs in a spatial format are provided by a dedicated FAA webpage[18]. Cirium´s laminardata provides even API access[19] to current data, cost free for only a couple of weeks though. Visiting these sources on daily basis does not grant a complete TFR dataset (and not a meaningful sample size either). So, it was decided to acquire historical data.

Historical temporary flight restrictions were accessed via FAA FNS NOTAM Search webpage[20]. The Archive Search (Figure 6) displays NOTAMs that were active for a selected location (FIR in this case) on a selected date (Figure 7).

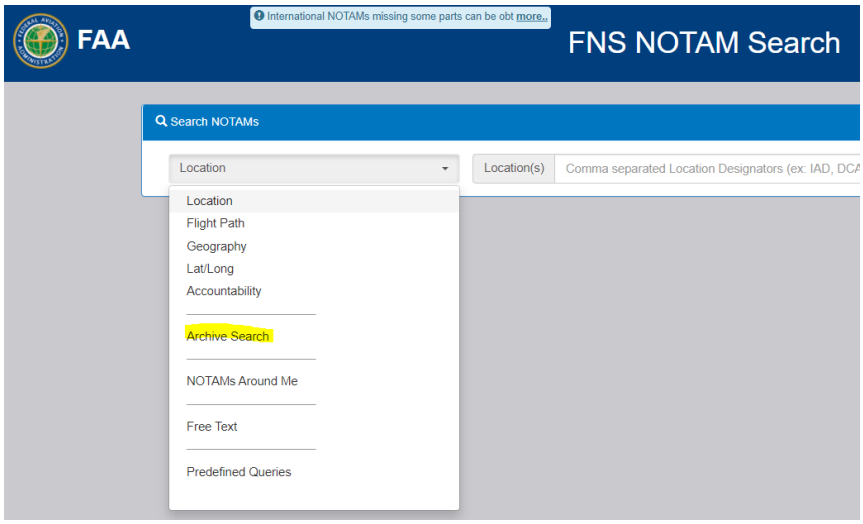


Figure 6 On FAA NOTAM Search webpage, open the dropdown menu to access Archive Search

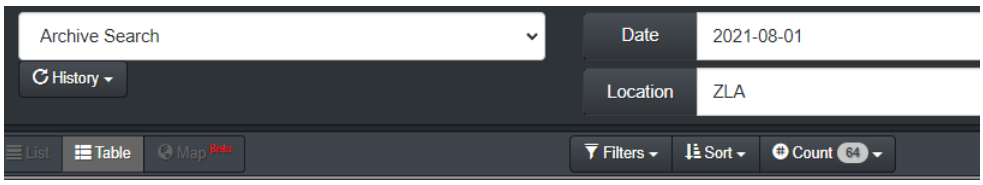


Figure 7 Archive Search allows looking for NOTAMs active on a single date for a single location only, here at the beginning of the thesis´ time frame on August 1st for Los Angeles FIR (ZLA)

With a sample size of 10 FIRs over three months, August to October 2021, this search could have up to 920 possible result sets. Those were filtered to contain TFR only (Figure 8).

[18]Current TFRs, download as shape file one at a time: https://tfr.faa.gov/tfr2/list.jsp
[19]Laminardata NOTAM ad TFR API https://developer.laminardata.aero/documentation/notamdata/v2
[20] FAA FNS NOTAM Search https://notams.aim.faa.gov/notamSearch/nsapp.html#/

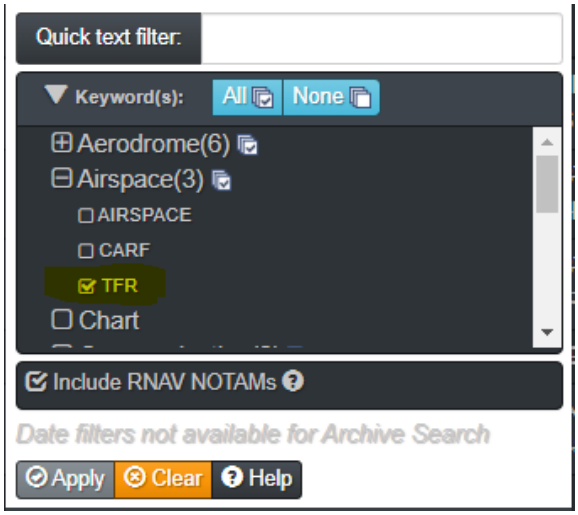


Figure 8 Archive Search has filter capabilities to choose TFR NOTAMs only

A functionality to download to as Excel .xls file (Figure 9) was used. Manually downloading 31 files of one month for one FIR took about 8 minutes. Days with no TFR being active within an FIR were documented (see appendix 8.5).

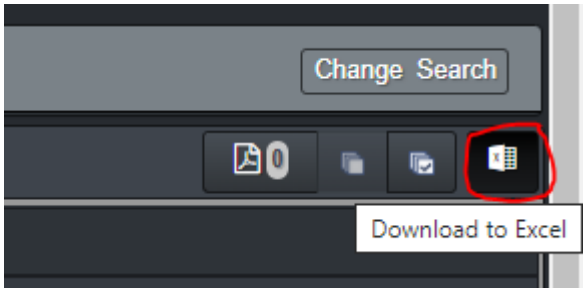


Figure 9 Filtered results can be downloaded as XLS files clicking the button in the top right corner of the Archive Search page

Concerning the filtered results, the downloaded file did contain the full NOTAM texts except for those that were only listed as “CANCELLED BY FDC…”, like number 1/9094 shown below (Figure 10).

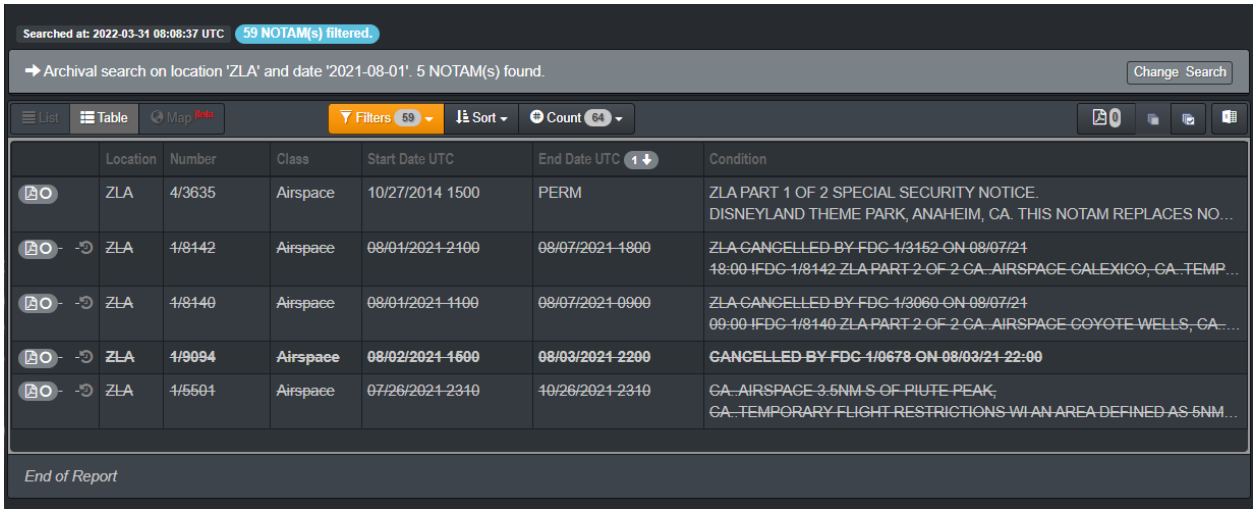


Figure 10 The filtered results are listed with those being crossed out where the end date has been reached

These cancelled NOTAMs had to be revisited one by one, manually opening the history tab (Figure 11) to include their full text into the study. Because NOTAMs do appear on multiple dates until reaching end or cancel date, this revision was be performed later from a dataset clean of duplicates (4.2.1).

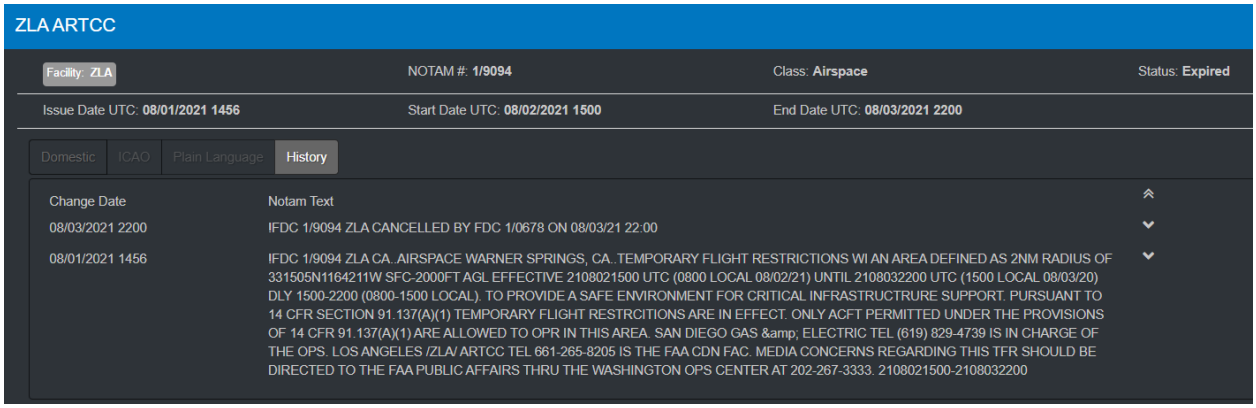


Figure 11 Having clicked a NOTAM in the list (Figure 10), the History tab allows for accessing the full text also of a cancelled NOTAM

Downloaded XLS files were named using a timestamp of their creation by default. The top rows contained some metadata within only one field per row while column headers were located below (Figure 12).

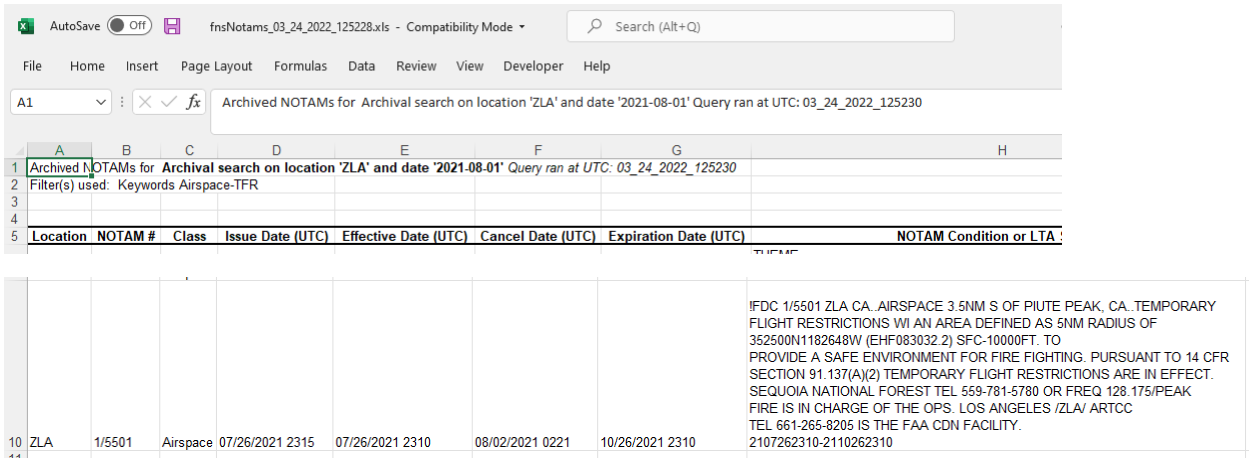


Figure 12 Screenshot (to show formatting issues) of an XLS file containing TFRs, downloaded from FAA NOTAM Search / Archive Search

4.1.3. Locations of Firefighter Planes as Point Data

OroraTech has collected positional data from fire fighting aircraft. This point data was provided as a CSV file and could be gathered in OroraTech´s cooperation with The OpenSky Network[21]. All aircraft having a transponder for Automatic Dependent Surveillance–Broadcast (ADS-B) can be tracked. Attributes of the point data (among others) were the aircraft ID, direction, velocity, GPS altitude and a timestamp (to the split second, in UTC).

ADS-B is not capable of delivering complete trajectories of active firefighters, especially for the most critical low level flight phase when the signal to SSR antenna (or other receivers) is blocked by obstacles, Lilly et al. (2021, p. 2), see (Figure 13).

[21] The OpenSky Network - Free ADS-B and Mode S data for Research (opensky-network.org)

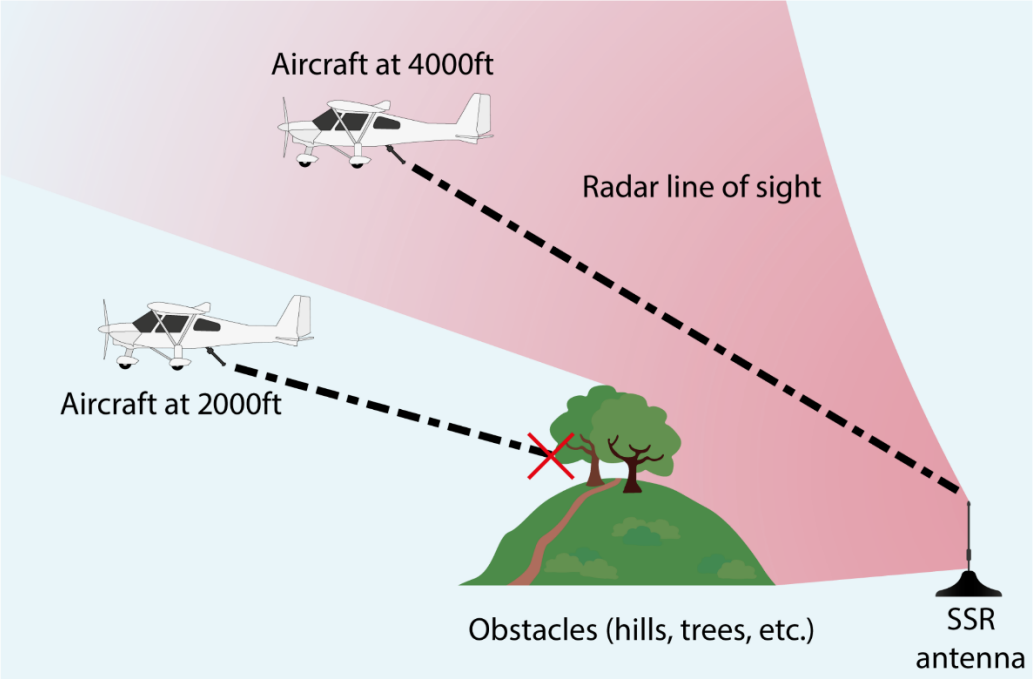


Figure 13 Obstacles blocking signals, making aircraft at low level "disappear" form ADS-B datasets[22].

4.1.4. The Study Area as FIRs derived from Map Image Layer

Collection of TFR data, as the entire area of interest, was based on flight information regions (FIR). FIR data from 2015 got viewed cost free via ArcGIS Map Viewer[23]. Current data is only available at a high cost[24].

Based on the free ICAO data, vector data got created by hand (Figure 14) to later be able to define areas of interest (AOI) to export fire clusters from OroraTech Wildfire Service. Scope was the entire perimeter of the 10 western U.S. FIRs to match the TFR AOI´s shape (Figure 15). To define AOIs/export regions, the Wildfire Service is capable of reading WKT. Export regions in WKT were produced in ArcGIS Pro from the FIR polygons (see Jupyter Notebook Gain_FIRs_as_WKT.ipynb within appendix 8.2.1 and on GitHub).

[22] Figure published under https://creativecommons.org/licenses/by/4.0/, Available from: https://www.researchgate.net/figure/An-aircraft-at-a-low-altitude-is-difficult-to-track-with-ADS-B-Mode-S-where-the-terrain_fig1_350130867

[23] https://uia.maps.arcgis.com/apps/mapviewer/index.html?webmap=724dfc8916604483a0ab06b4f3cbe57f

[24] See ICAO store https://store.icao.int/en/data/flight-information-regions-fir

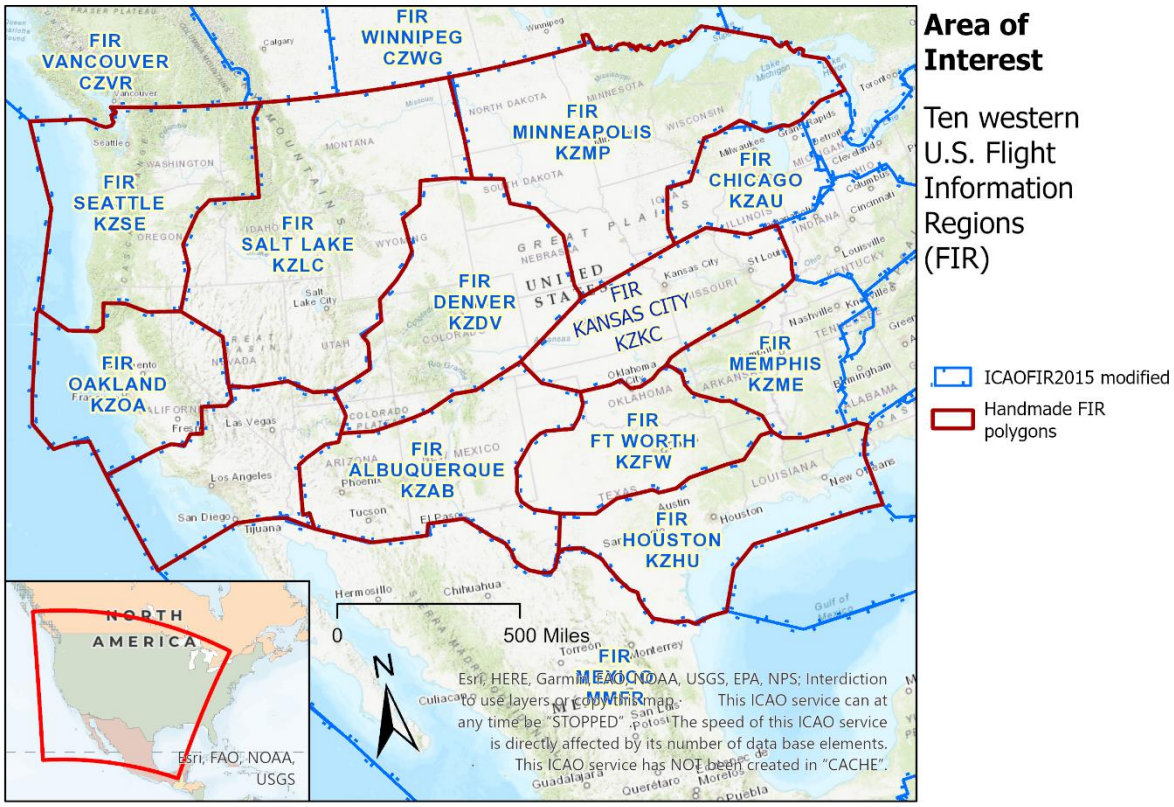


Figure 14 10 FIRs, blue: 2015 ICAO free dataset[25], dark red: Polygon Feature Class, drawn based on the ICAO dataset.

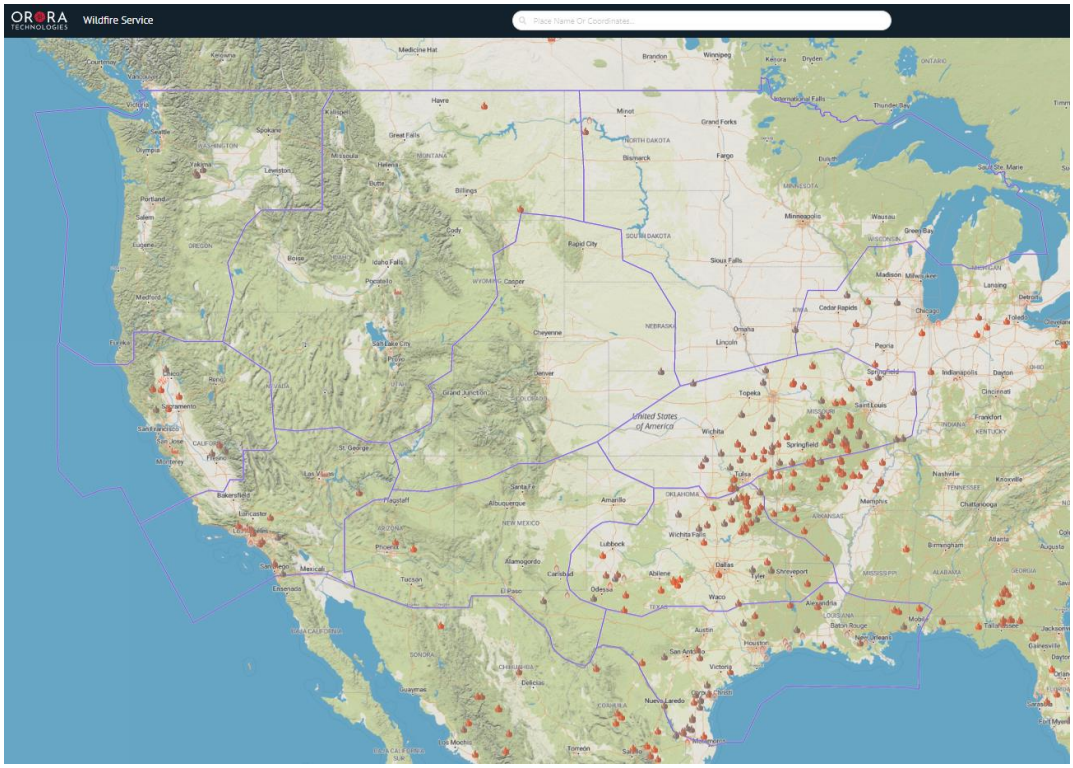


Figure 15 FIRs displayed in OroraTech´s Wildfire Service

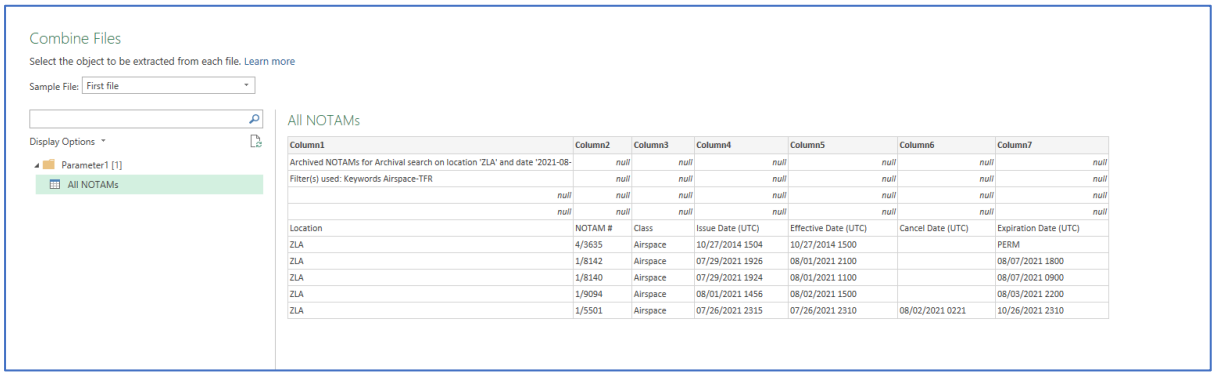
[25] 2015 ICAO free dataset available via https://gis.icao.int/arcgis/rest/services/FIRWORLD/MapServer

4.2. Methods

This section starts explaining, how TFR NOTAM texts got parsed and converted into GeoJSON format (4.2.1). The thought process behind considering a TFR "appropriate" or not is explained (4.2.2). The following exploration (4.2.3) already contains its results to be able to proceed to the upcoming three analyses building the core of the thesis (4.2.4, 4.2.6 and 4.2.5). The final analysis started from the TFR side and checked the fire cluster data looking for TFRs where no fire cluster could be detected (4.2.7).

4.2.1. From Text to GeoJSON: Turning TFRs to a spatial Data Format

TFRs were downloaded as Excel Workbooks (4.1.2.2). It was not possible to easily concatenate the downloaded files using Power Queries at this stage. For instance, with the original structure, the eighth column containing the NOTAM text was not recognized reliably (Figure 16).



Column1	Column2	Column3	Column4	Column5	Column6	Column7
Archived NOTAMs for Archival search on location 'ZLA' and date '2021-08-	null	null	null	null	null	null
Filter(s) used: Keywords Airspace-TFR	null	null	null	null	null	null
null	null	null	null	null	null	null
null	null	null	null	null	null	null
Location	NOTAM #	Class	Issue Date (UTC)	Effective Date (UTC)	Cancel Date (UTC)	Expiration Date (UTC)
ZLA	4/3635	Airspace	10/27/2014 1504	10/27/2014 1500		PERM
ZLA	1/8142	Airspace	07/29/2021 1926	08/01/2021 2100		08/07/2021 1800
ZLA	1/8140	Airspace	07/29/2021 1924	08/01/2021 1100		08/07/2021 0900
ZLA	1/9094	Airspace	08/01/2021 1456	08/02/2021 1500		08/03/2021 2200
ZLA	1/5501	Airspace	07/26/2021 2315	07/26/2021 2310	08/02/2021 0221	10/26/2021 2310

Figure 16 Power Queries does not recognize a column containing the NOTAM text

To be able to handle these files, a Visual Basic Module was coded (8.2, Code 2). It was meant to save the downloaded files with a filename derived from the metadata in an FIR-related folder and to get rid of the first four lines resulting in the column headers being the top row. The macro wrote a log to discover mistakes possibly made during the repetitive task of switching date and downloading (Figure 17). The full log is provided by section 8.6 and together with 8.5 it proves that every day within the observed timespan was visited via NOTAM Search, thus making the TFR dataset complete.

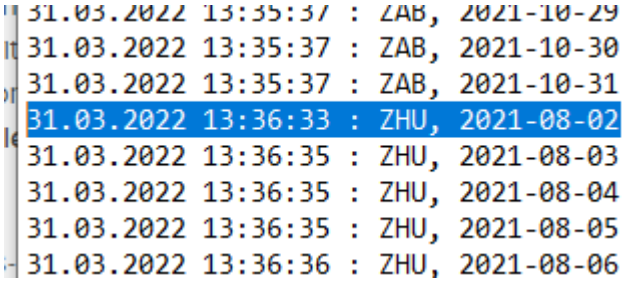


Figure 17 Screenshot from the log file created to keep an overview over files processed by the VBA script. Here, a dataset for ZHU, 2021-08-01 is either missing or no TFR was active that day (the latter was the case, see 8.5)

With the files prepared by the VBA macro, Power Queries was used to add a Source column as well as to remove duplicates from the NOTAM # column. In a new .xlsx file, the daily data was gathered from where it was stored locally (Figure 18).

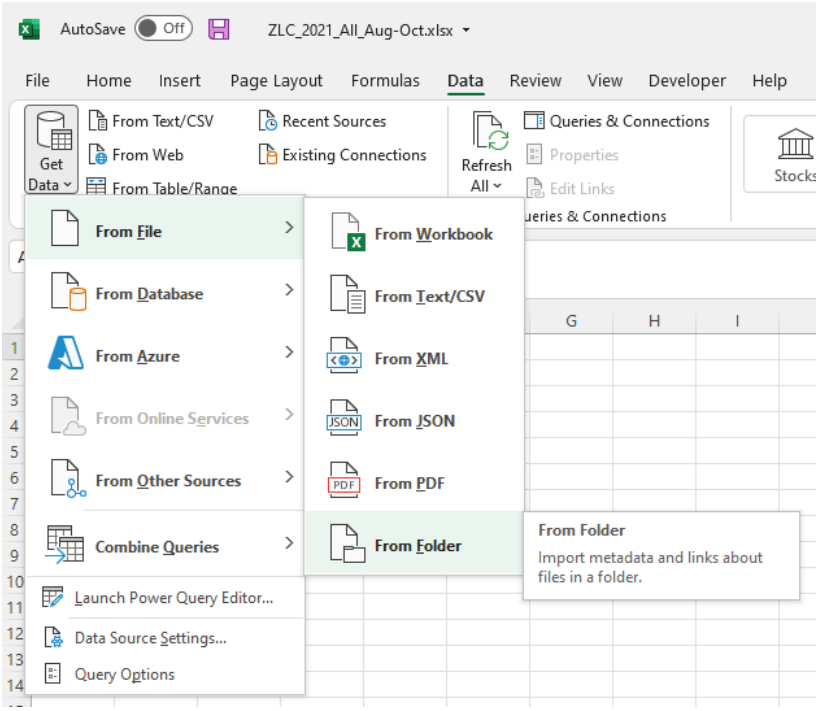


Figure 18 Power Queries is launched via Data → Get Data → From File → From Folder

Selecting Combine and Transform Data (Figure 19), and then the All NOTAMs sheet (Figure 20), the following process was based on what Power Queries fetched from the sample file within the location. When it recognized the original column names (Figure 21), in Power Queries Editor the Advanced Editor (Figure 22) was started right away, 8.2-Code 3 was pasted and the folder location adjusted.

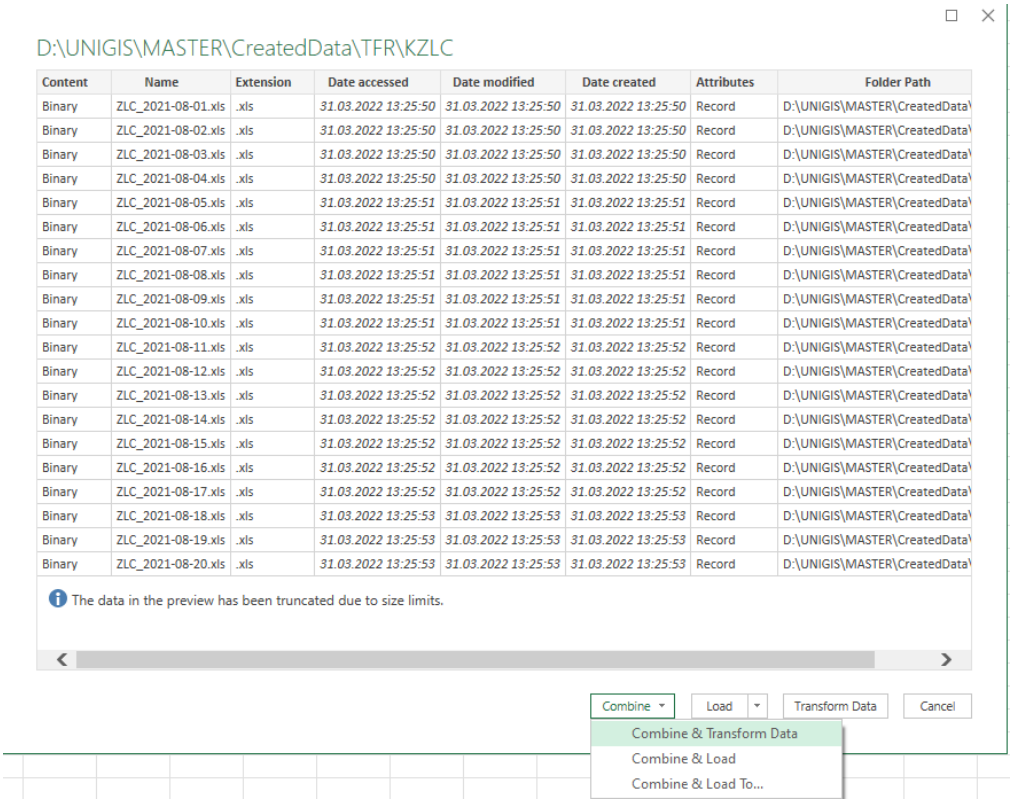


Figure 19 To concatenate all TFR data, Combine & Transform Data is selected running Power Queries.

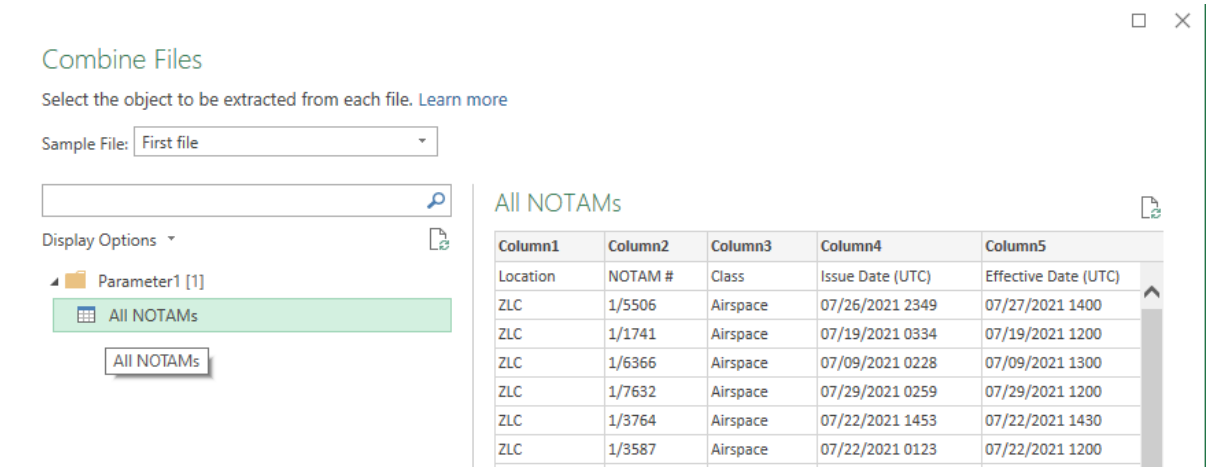


Figure 20 From the accessed sample file, the All NOTAMs sheet is selected. The preview on the right is composed according to the sample´s content. In this case of FIR KZLC, the column names are not recognized properly.

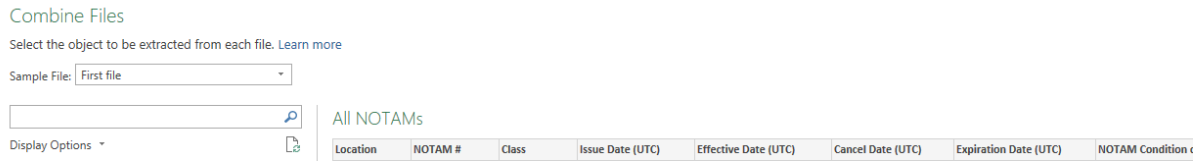


Figure 21 PowerQueries recognizing the original column names correctly

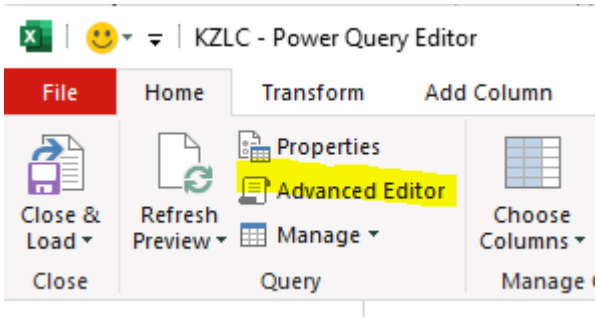


Figure 22 With Power Queries started, the Advanced Editor can be launched

With the example of KZLC (Figure 20), that started with numbered columns only, also duplicates needed to be removed first, then the first row had to be used as headers and after that the first Source column got its name and values were cleaned of file endings. If this occurred, 8.2-Code 4 had to be used and adjusted.

After the proper code was run, Close & Load (Figure 23) added a sheet and filled it with the NOTAM data (Figure 24).

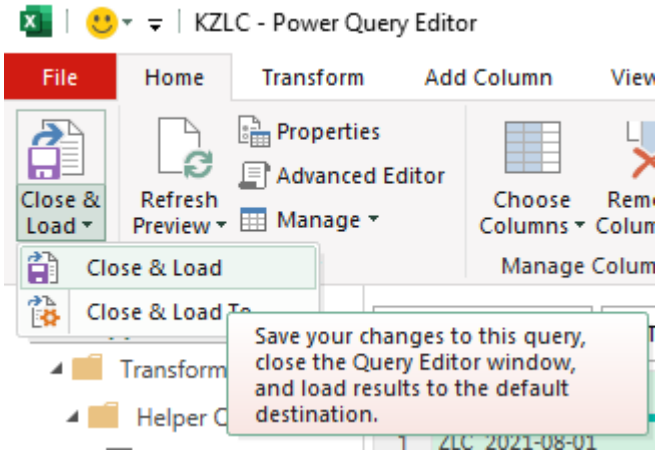


Figure 23 Click Close & Load to add a sheet filled with the concatenated data

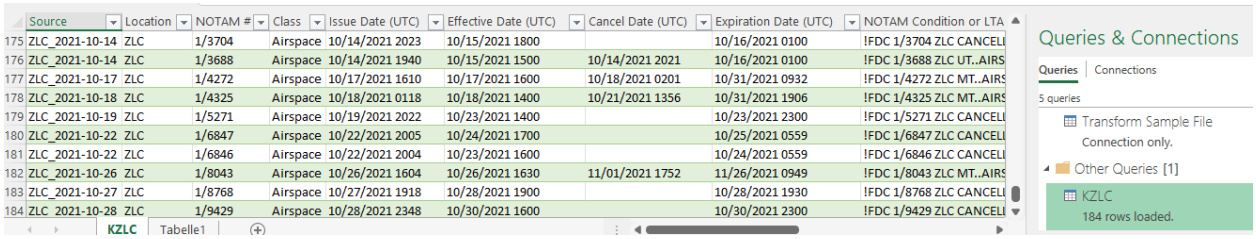


Figure 24 Resulting Excel sheet with unique TFR occurrence over the observed time frame

At this point the resulting sheet was still linked to its data sources (Figure 25).

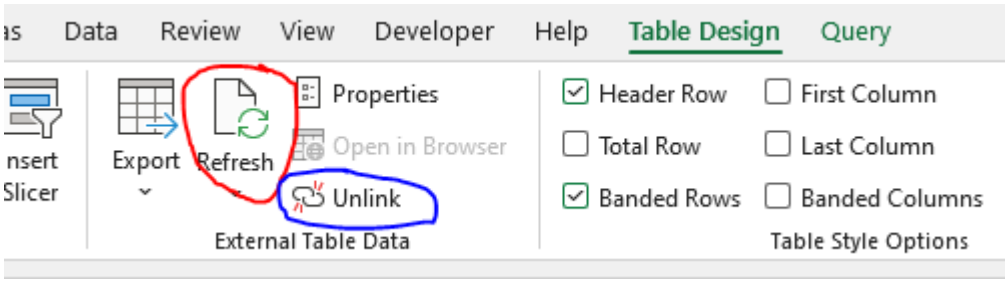


Figure 25 The sheet with all TFRs is still linked to its source and can either be refreshed or unlinked to maintain data integrity

Those NOTAM texts that got omitted within the downloaded .xls files due to cancellations were revisited manually. Table 4 shows how many wildfire related TFRs (containing string “FIRE FIGHTING”) were additionally discovered this way.

Table 4 Results of manual revision of cancelled TFRs

FIR	Amount of manually revisited TFR	Amount of manually discovered wildfire-related TFR	NOTAM numbers revisited manually, wildfire-related in red
KZAB	16	0	1/1201; 1/5325; 1/8503; 1/0049; 1/2525; 1/7151; 1/0369; 1/0700; 1/0993; 1/2141; 1/3204; 1/4079; 1/4615; 1/9027; 1/9028; 1/9026
KZDV	12	9	1/1585; 1/2974; 1/3132; 1/3161; 1/3169; 1/3190; 1/5720; 1/0214; 1/4623; 1/6295; 1/8503; 1/0442
KZFW	12	1	1/4670; 1/9383; 1/0462; 1/1638; 1/1887; 1/1888; 1/4017; 1/4020; 1/5573; 1/6252; 1/9732; 1/9677
KZHU	14	0	1/9853; 1/3990; 1/4554; 1/1569; 1/2581; 1/2320; 1/5695; 1/6776; 1/9530; 1/1767; 1/2036; 1/2647; 1/9234; 1/9235
KZKC	39	0	1/8566; 1/0925; 1/0927; 1/1478; 1/2946; 1/3819; 1/3991; 1/5055; 1/2472; 1/3723; 1/7072; 1/0604; 1/0613; 1/1408; 1/2634; 1/3089; 1/5790; 1/6995; 1/6922; 1/9130; 1/0021; 1/1479; 1/1486; 1/2957; 1/4165; 1/6517; 1/6226; 1/8027; 1/8029; 1/8032; 1/1519; 1/1520; 1/2246; 1/2254; 1/2302; 1/3689; 1/4926; 1/5655; 1/5656
KZLA	5	1	1/3840; 1/2493; 1/3109; 1/6221; 1/1924;
KZLC	61	18	1/9211; 1/7269; 1/8960; 1/1237; 1/1602; 1/1603; 1/2502; 1/2500; 1/2930; 1/2499; 1/3834; 1/4559; 1/4821; 1/4779; 1/7013; 1/8240; 1/8070; 1/8455; 1/9072; 1/0288; 1/0247; 1/0248; 1/3522; 1/5550; 1/1401; 1/5325; 1/5324; 1/5322; 1/5321; 1/5297; 1/4755; 1/5401; 1/5420; 1/5400; 1/5421; 1/5399; 1/5398; 1/5395; 1/5392; 1/5360; 1/5887; 1/2758; 1/2763; 1/2762; 1/2753; 1/3499; 1/7972; 1/1795; 1/1794; 1/1793; 1/1796; 1/2644; 1/3687; 1/3686; 1/3704; 1/5271; 1/6847; 1/6846; 1/8768; 1/9429; 1/9430;
KZMP	39	0	1/9698; 1/1457; 1/3184; 1/4121; 1/4421; 1/5282; 1/6082; 1/6097; 1/6105; 1/6085; 1/9725; 1/0984; 1/1953; 1/2503; 1/4926; 1/4757; 1/5536; 1/7505; 1/8332; 1/9225; 1/6919; 1/0049; 1/0120; 1/1570; 1/1791; 1/3361; 1/5427; 1/4999; 1/8036; 1/8034; 1/1766; 1/3181; 1/4442; 1/4950; 1/4947; 1/4888; 1/7947; 1/8403; 1/9717
KZOA	9	2	1/7851; 1/9462; 1/0429; 1/0428; 1/3128; 1/5713; 1/9523; 1/3898; 1/6160
KZSE	22	15	1/7785; 1/9106; 1/3048; 1/8961; 1/9568; 1/1480; 1/1396; 1/3196; 1/3836; 1/4930; 1/5421; 1/5729; 1/7243; 1/1622; 1/8794; 1/9376; 1/2073; 1/2581; 1/2585; 1/3886; 1/8874; 1/6506

The revisited NOTAMS were parsed and converted into GeoJSON format by a Python script, one file per FIR. The script can be found as Jupyter Notebook Fire_NOTAM_to_spatial.ipynb in the appendix (8.2-Code 5). This script contains a few provisions for sanity checking. And these paid off: Different notations of the radius for circular TFRs (which got finally parsed correctly) could get implemented writing the script.

Multi-part NOTAMs could occur with only one of their parts and were not completed in the dataset as they usually do not result from fires but are issued for other reasons like defining “national defense airspace” close to the Mexican border (Figure 26) or “space OPS area” for Houston FIR (Figure 27). Kansas City FIR (KZKC) and Houston FIR (KZHU) turned out not to have a single wildfire related TFR issued.

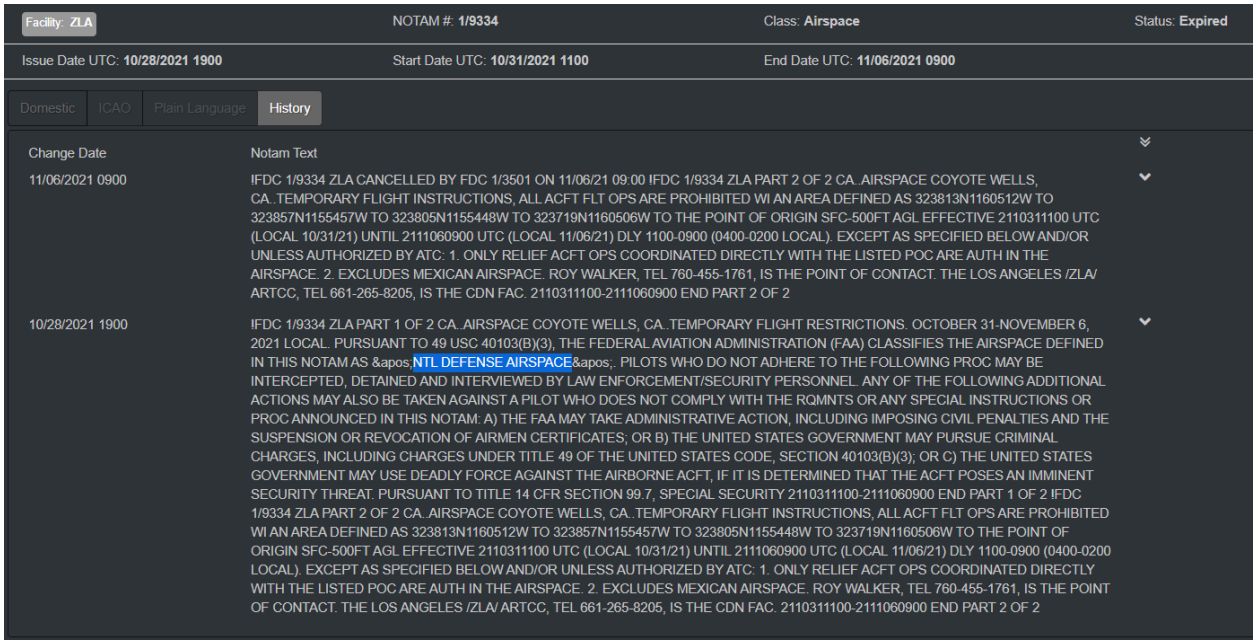


Figure 26 TFR full text as an example for a "national defense airspace" TFR not completed or contemplated by this research

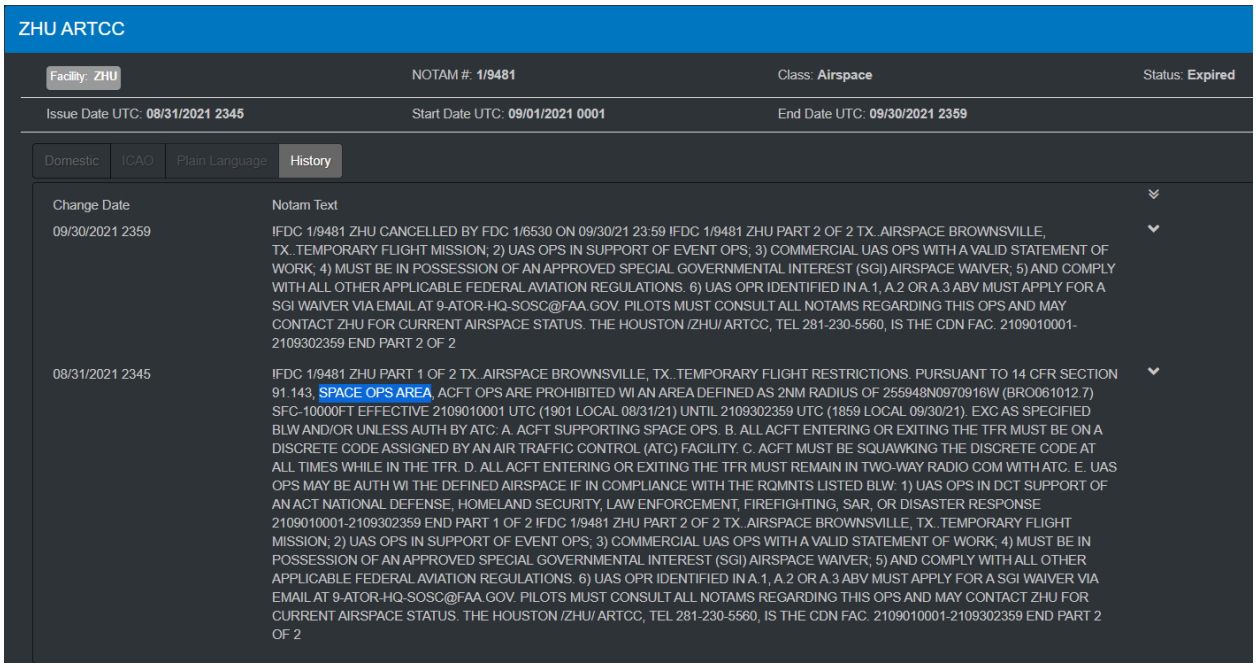


Figure 27 TFR full text as an example for a" space OPS area" TFR not completed or contemplated by this research

4.2.2. Defining Appropriateness of a TFR

The following considerations were made to define an appropriateness of a TFR for aerial firefighting, based on section 3.3. First, talking of fires as polygons, firefighting aircraft are not dropping their load on centroids but right outside along the edges. So, a TFR should keep these edges clear of other traffic. The distance for this is determined by visibility, as it is key for fire fighting aircraft operations. A Fire Traffic Area (FTA) with its vertical extension of 2500 feet usually touches two different airspace classes, "G" and "E" (see Figure 28).

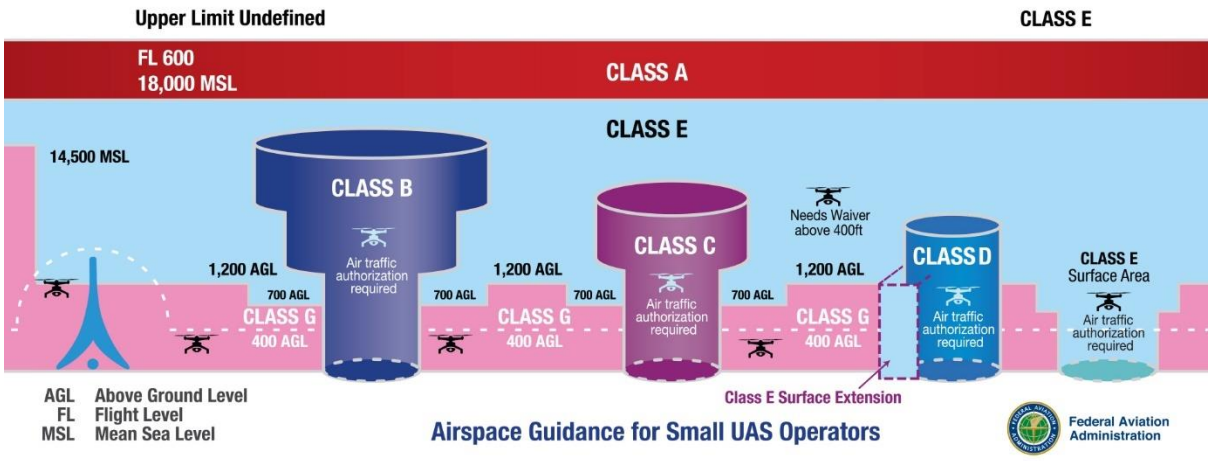


Figure 28 Airspace classes, numbers in feet; graphic from FAA https://www.faa.gov/uas/recreational_fliers/where_can_i_fly/airspace_101/media/airspace_classes_large.jpg

To fly without air traffic control, just seeing where to go and avoiding other aircraft, is called operating under Visual Flight Rules (VFR). Under VFR, moving in class E airspace requires a minimum visibility of 3 statute miles[26], while doing so in class G requires a minimum visibility of 1 statute mile (SM), FAA (2022b). Thus, it was decided that a TFR is considered appropriate, if not only the original edges but also fire clusters buffered by these minimum visibility values remain contained by a TFR. Where applicable, three runs of the related scripts were made, one with 3 statute miles buffer, one with 1 statute mile buffer and one with the original fire cluster size.

4.2.3. Exploring the Datasets

All input data was turned into a spatial data format where necessary. Then, datasets were loaded into ArcGIS Pro to get an overview and to gather statistics where necessary.

[26] To make things more complicated (for European, non-aviation, non-nautics affiliate readers) this does not equal nautical miles. 1 statute mile = 1609.344 meters.

4.2.3.1. Fire Clusters and TFRs explored

The manually exported fire cluster data from OroraTech was already delivered in GeoJSON format. Considering the types in the dataset, only 56 of the 2926 fire clusters are assigned another permanent or artificial heat source such as flare or solar, Figure 29 shows the distribution according to the types from Table 3. The input dataset was not cleaned with regard to types, the type-attribute remained included in resulting datasets to unveil surprises.

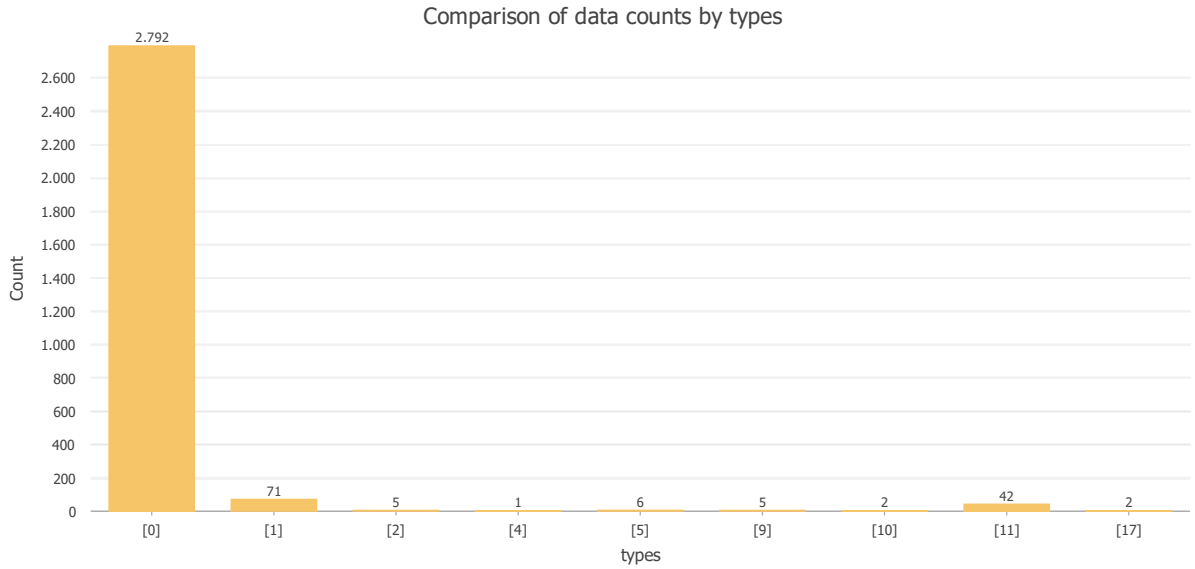


Figure 29 Distribution of fire types with regard to Table 3: [0] is unclassified, [1] is fire, [5] is forest, these should reoccur in later results. [4] is false detection and all remaining types are artificial and more or less static heat sources that are not expected to cause a TFR.

How the TFR NOTAM texts became polygons is described in 4.2.1 on page 23. The tool *JSON To Features* brought them into ArcGIS Pro. To add all eight datasets at once, a small script was used (Add_TFRs_from_geojson.ipynb). A map section shows the fire clusters on top of TFRs in Figure 30. Getting on track of this here, it turned out that TFR issuing ARTCCs (and therefor TFR “locations”) did not always stick to their geographical FIR boundaries. Thus, it appeared that the entire fire cluster dataset had to be used by the following workflows, even though only one FIR´s TFRs were later concerned at a time. As found in some FIRs, ongoing fire activity and multiple TFRs issued lead to overlapping polygons. This was taken care of in the design of the analysis workflows. The fact that some TFRs seemed to be issued without a corresponding fire cluster caused the last objective to be worked on: A TFR without corresponding fire had possibly been issued in the wrong place or may still indicate a blind spot of OroraTech´s Wildfire Service.

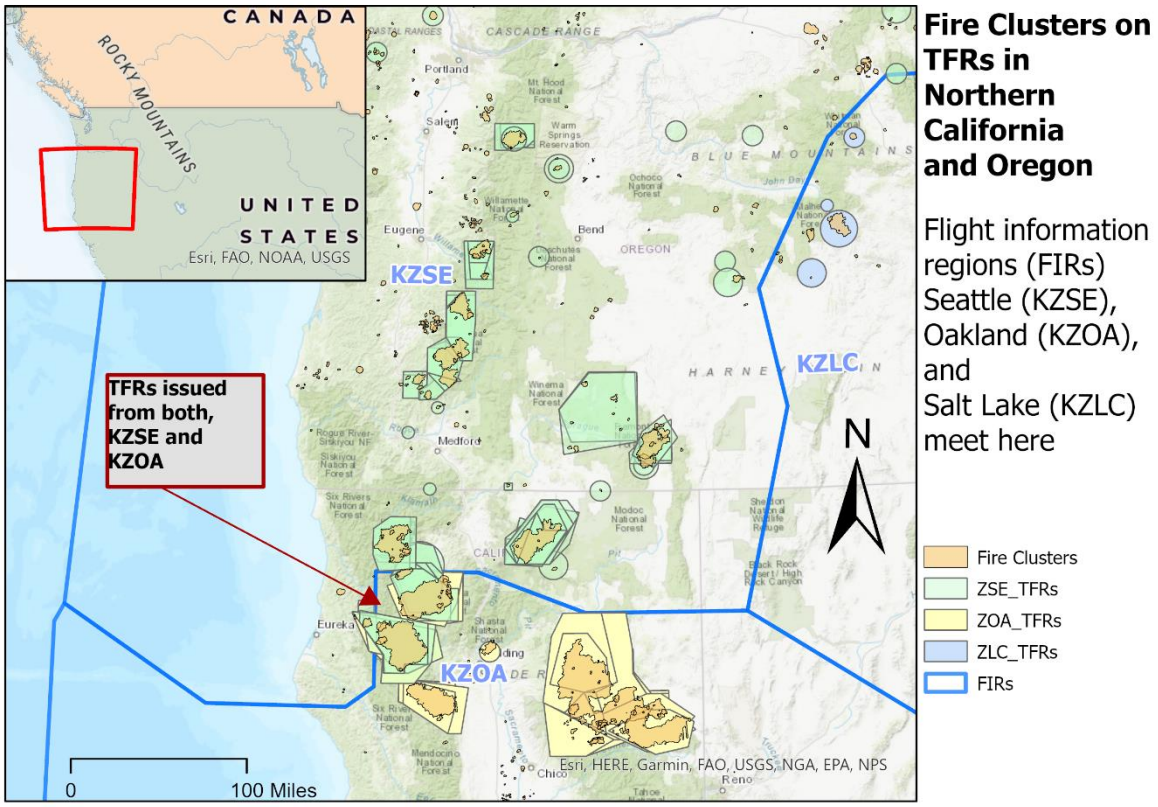


Figure 30 Fire clusters are overlaid on the TFRs, which may be issued across FIR borders. Possibly repeated, but in any case overlapping and close-by TFRs and fire clusters can be found. A few TFRs (here from KZSE) appear to have been issued without a fire cluster.

Figure 30 does also show circular as well as angular shapes of the TFRs. By design of the script (Fire_NOTAM_to_spatial.ipynb), angular shaped TFRs got a radius of 0. From a merged dataset (ArcGIS Pro *Merge* tool on all TFR layers), all 295 angular TFRs were sorted out[27]. Then, to assure that the above appropriateness-decision from 4.2.2 did not become self-fulfilling prophecy, statistics of TFR radius were provided for the remaining 247 circular areas, shown here by Table 5 and Figure 31. The smallest TFR has a radius of 1 NM, the largest has a radius of 12 NM. Less than 10% (27 circular TFRs) have a radius below 5 NM. Hence, the appropriateness-decision to use 1 and 2 statute miles for a buffer was not expected to skew results right from the start.

[27] We already get the scent that they are shaped to fit a "fire cluster´s need" or to contain more fires. But there are circular TFRs with a smaller radius than the recommended minimum of 5 NM (NWCG).

Table 5 Statistics of Radius from all circular TFRs

Statistics of Radius from all circular TFRs				
Count	**Min** [NM]	**Max** [NM]	**Mean**	**Median**
247	1	12	5.7	5

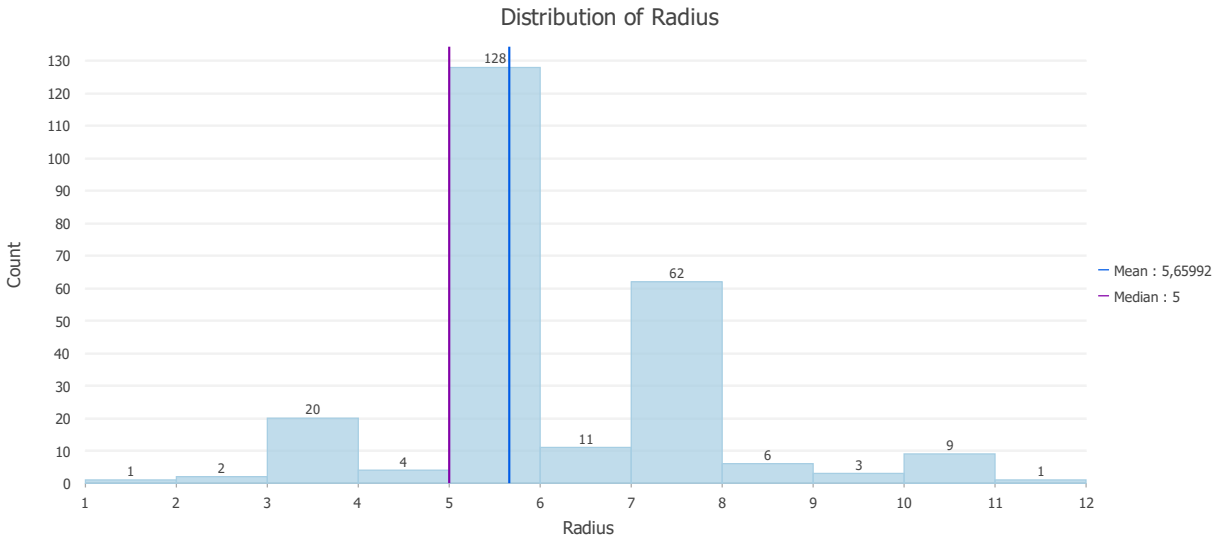


Figure 31 Derived for all circular TFRs: Histogram showing the distribution of Radius. 220 of 247 circular TFRs have a radius of 5 nautical miles and more

4.2.3.2. Aircraft state vectors explored

The original dataset contains only aircraft that are known for being used for firefighting. Received CSV file was converted into GeoJSON format by a Python script (Aircraft_States_to_GeoJSON.ipynb). State vectors are generally point data. Visualizing 384370 points in Arc GIS Pro (Figure 32 , via tool *JSON To Features*) yielded the following insights: Firefighting aircraft movement in 2021 concentrated in the western U.S. as well as the TFRs and fires, as expected from US EPA (2021) data.

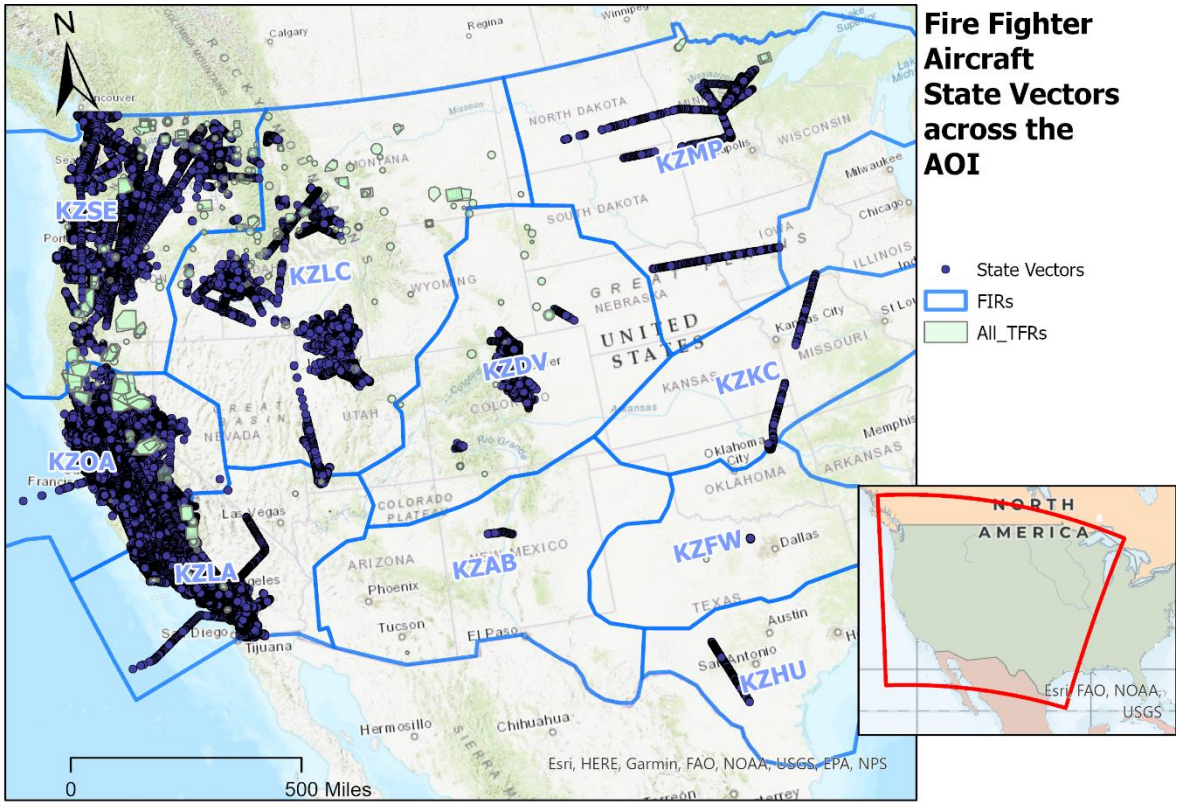


Figure 32 The entire dataset of aircraft state vectors within the study area visualized in ArcGIS Pro below TFR polygons. Firefighting aircraft concentrate in the western U.S. as well as the TFRs. Represented like this, there is not yet a structure recognizable.

To get a better overview, the dataset was narrowed down by joining fire clusters on aircraft states with a 3 miles search radius (see 4.2.2) and then keeping only those, where the state vector timestamp is within a fire´s acquisition time range. Necessary steps within ArcGIS Pro are shown in Figure 33, Figure 34 and Figure 35. This created a first guess dataset of aircraft close to fires which were actually burning at that time. Then, it could be decided whether objective (5), did a TFR area cover enough area to conduct aerial firefighting safely, was to be tackled at all.

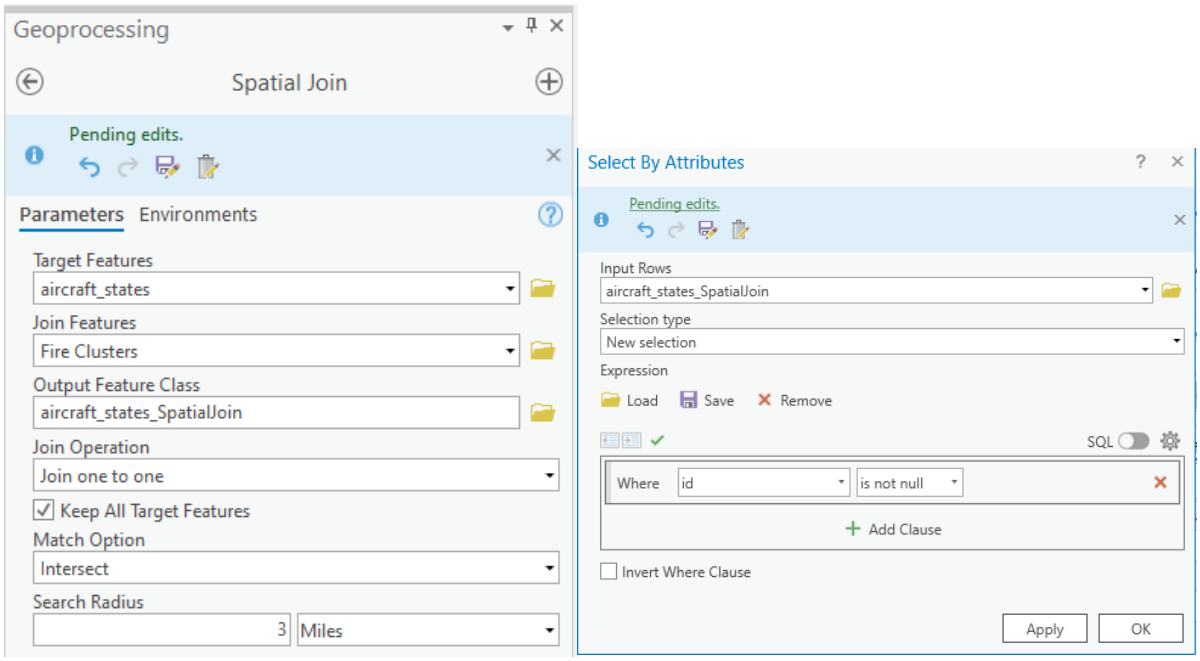


Figure 33 A Spatial Join of Fire Clusters polygons on aircraft_states points adds fire cluster attributes to all points met (left). When Keep All Target Features is checked, a new Select by Attributes must be made to keep only points where a polygon matched (right).

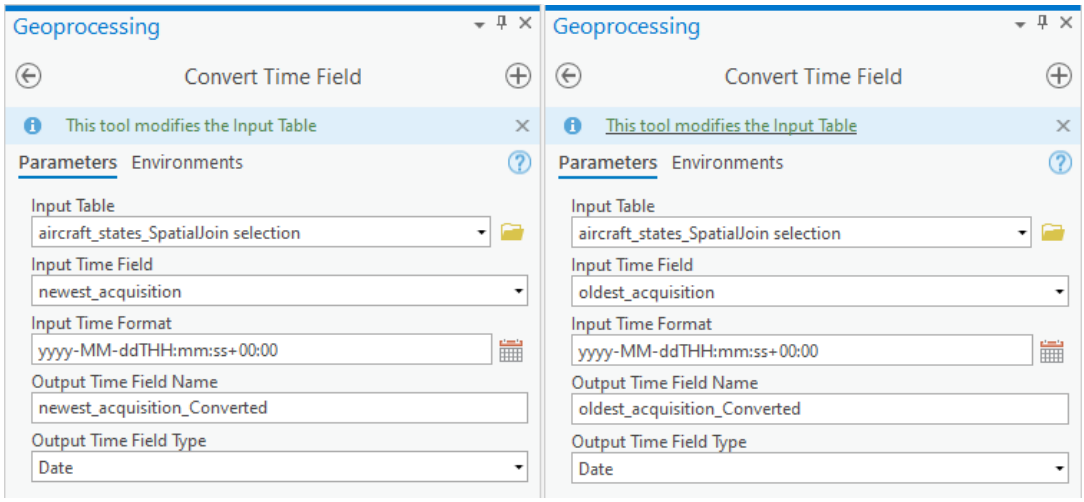


Figure 34 Convert Time Field produces a Date field that can be compared.

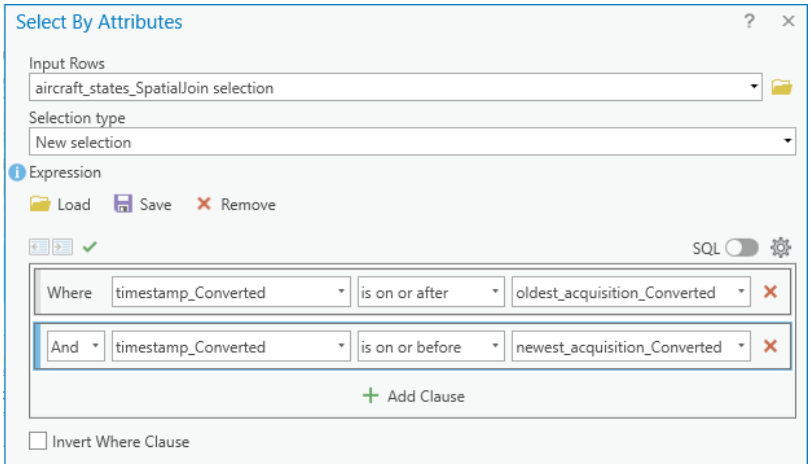


Figure 35 Selecting points by attribute in between oldest and newest acquisition time provides a point set for fire fighting aircraft being over presumably active fires

Keeping as much data as possible for as long as possible lead to static heat sources still being included. Therefore, only rows with type attribute [0] unclassified, [1] fire and [5] forest were kept. The result looked promising: Applying Summary Statistics on the aircraft id showed at least 60 different aircraft represented by more than 1 point.

To foster a better understanding, an attempt to create trajectories with MovingPandas was made, Graser (2019) and Graser & Dragaschnig (2020), see Aircraft_ovr_fires_to_trajectories.ipynb. The goal was to create a countable instance like “flights” or “aircraft movements” for following analysis. This needed an export from ArcGIS Pro as GeoJSON file in advance (Figure 36).

Figure 36 Features To JSON setting to export the aircraft point data over presumably ongoing fires

The script produced connected trajectories per plane, as well as split trajectories using a minimum gap size of 1 hour to create separate aircraft movements because it turned out that especially huge planes are sent to locations across several FIRs (Figure 37).

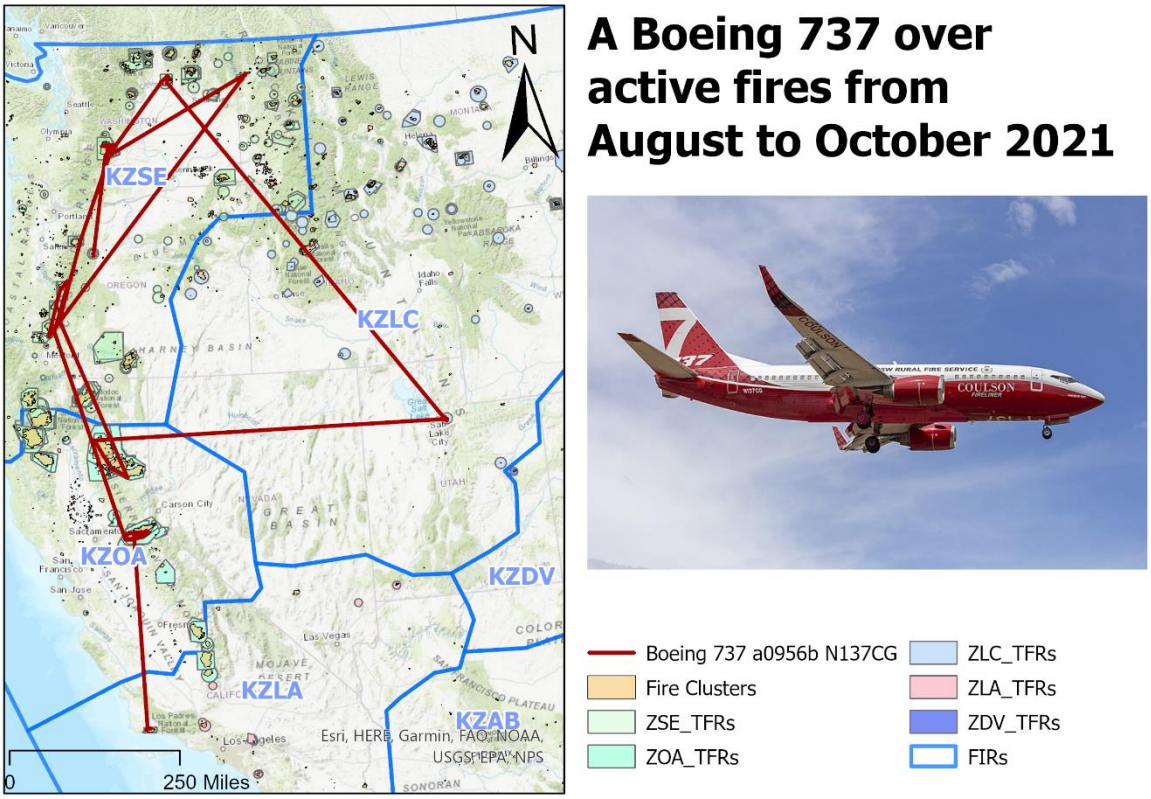


Figure 37 The dark red trajectory (built from state vectors found over active fires only and not yet split) shows a Boeing 737 being used across four different FIR and several fire clusters and TFRs. With ICAO identifier a0956b and callsign N137CG, this is actually one of the planes depicted in Aerial firefighting - Wikipedia (picture by Bidgee/ Robert Myers, published under CC Creative Commons — Attribution-ShareAlike 3.0 Unported — CC BY-SA 3.0)

Reading the split trajectories back to ArcGIS Pro completed the picture, as explained with the following example (and Figure 38): TFR 1/8595 got issued at 01:21 UTC becoming effective at 02:00 UTC due to a fire close to Chamokane Creek in the northwest of Spokane. At 01:47 UTC an approaching firefighter plane (ICAO 24 id c01aeb, callsign: TNK52) disappeared in a valley (or crossing a ridge, most probably to an ADS-B blind spot as explained in 4.1.3). At 02:35 the firefighter appeared again, considering the 3-mile vicinity of the entire cluster. As a matter of fact, the aircraft entered a 7 NM TFR shortly before it became effective. As this random sample shows, it is worth an attempt to examine safety of actual fire fighting aircraft to look for them being guarded by TFRs.

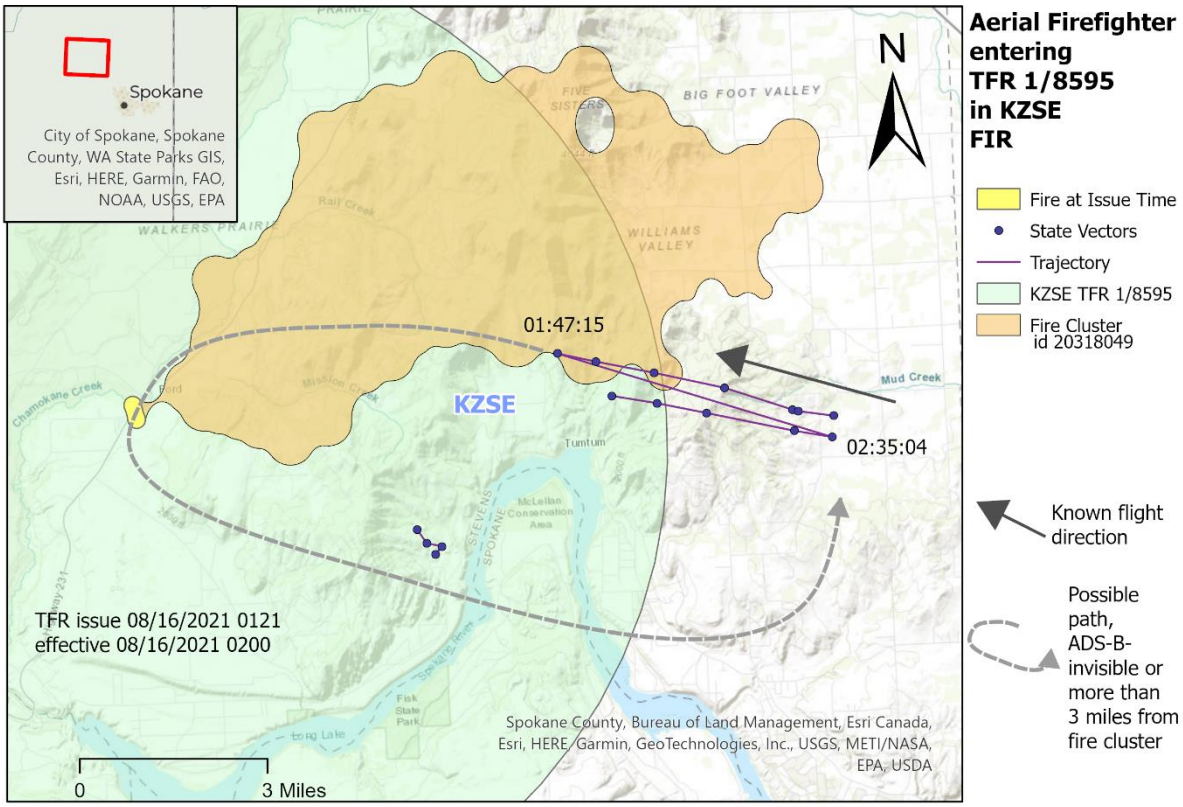


Figure 38 Firefighter plane "TNK52" is represented by state vector point data and trajectory, originally found via Spatial Join with the fire cluster as a first guess. The solid arrow indicates flight direction. Most possibly on its way to the (by then) small fire (yellow polygon, gathered via API), the plane´s ADS-B signal is blocked by a ridge since 01:47:15 UTC when it has already entered the green TFR, that is not yet effective. (The separate flight track with the four vertices does not belong to TNK52 but to a8401a. TNK52 is a Convair CV580, converted from a CV340, an aircraft designed in the 1950s. a8401a is a Beechcraft B200, a small but younger plane with a ceiling more than twice the one of the CV580, so no wonder that the B200 can appear via ADS-B while the CV does not.)

4.2.4. The Time between Fire Detection and TFR Issue

How long was the airspace potentially unsafe for an initial attack? The idea was to first create a relation between fire clusters and TFRs. As educated guess to start with, the three-month-fire-clusters (from manual export) and TFRs were spatially joined by predicate 'intersect'. From these TFR-intersecting fire clusters, their oldest detection and acquisition time *prior* to the intersected TFR issue time had to be considered. To respectively request these fire clusters in their shape and with their acquisition and detection time prior to a TFR being issued, OroraTech´s API was used. The payload (data to submit) for API requests for parts of fire clusters being active needed a bounding box and a timeframe. The bounding box was derived from the three-month-fire-clusters. Timeframe bounds were determined by a number of minutes before TFR´s issue time and the issue time itself, plus going forward a few hours as well to match also the clusters known from ground information that got acquired via satellite shortly after TFR issue time. As the exported three-month fire clusters might touch close-by TFRs that were not meant to be issued for them, only the relation between API results and TFRs was to be considered. That required another spatial join by predicate 'intersect'. Sometimes multiple TFRs had been issued for the same fire event. This analysis treated the very first TFR, so duplicates of fires with later TFRs were dropped[28]. Then, a "current" fire cluster´s oldest detection and acquisition time could

[28] However, as TFRs are not linked in any way, follow-up TFRs for the same event could hardly be identified but expectedly produced greater numbers here.

be subtracted from the TFR issue time. The analysis was performed via Python, see Jupyter Notebook Get_Detection-Issue-Time_Gap.ipynb or the appendix at 8.2.4. Figure 39 shows the necessary steps as flowchart. The calculations were made to answer the following questions:

TFR issue time - oldest_acquisition ← How much time did it take at least until a TFR was issued since there had actually been a fire?

TFR issue time - oldest_detection ← How much time did it take at least until a TFR is issued since one could have known about it from OroraTech´s data.

For both questions only a minimum timespan (“at least”) could be calculated: Generally, it is possible that a fire burns for a while until a satellite passes at all or until a satellite pass happens while cloud cover allows for discovering a hotspot.

The search period before TFR´s issue time was set to 24 hours (1440 minutes), as request result clusters needed to get narrowed down to not touch nearby TFRs and it was assumed that longer waiting times before issuing a TFR had been intended. Fires with an older oldest_acquisition time were removed from the start as an initial attack on these fires was likely to have been dropped already before the observed time frame. It was found that a search period after TFR´s issue time set to 6 hours (360 minutes) was capable to match also clusters known from ground information that had been acquired via satellite after TFR issue time.

Results from TFRs issued in July before the observed time frame may suffer from a boundary value problem and should not be taken for granted because the API request bounding box did possibly not cover older hotspots. These TFRs could accidentally have become the first TFR of a fire cluster that was detected later. Knowing this, the old TFRs needed to get purged here right from the start.

An alternate, less complicated run was tried with a script not incorporating the API request. The results here were similar for trustworthy values but contained more false matches for areas with overlapping patterns. The script is available via GitHub as well (version 2 of Jupyter Notebook Get_Detection-Issue-Time_Gap.ipynb). The fact that TFRs do become issued beyond and across FIR borders (see 4.2.3.1) can skew results of the chosen per-FIR-approach. But an API request for all data at once requires much more time (with a consistent internet connection). The less complicated version can be run on a dataset containing all TFRs at once.

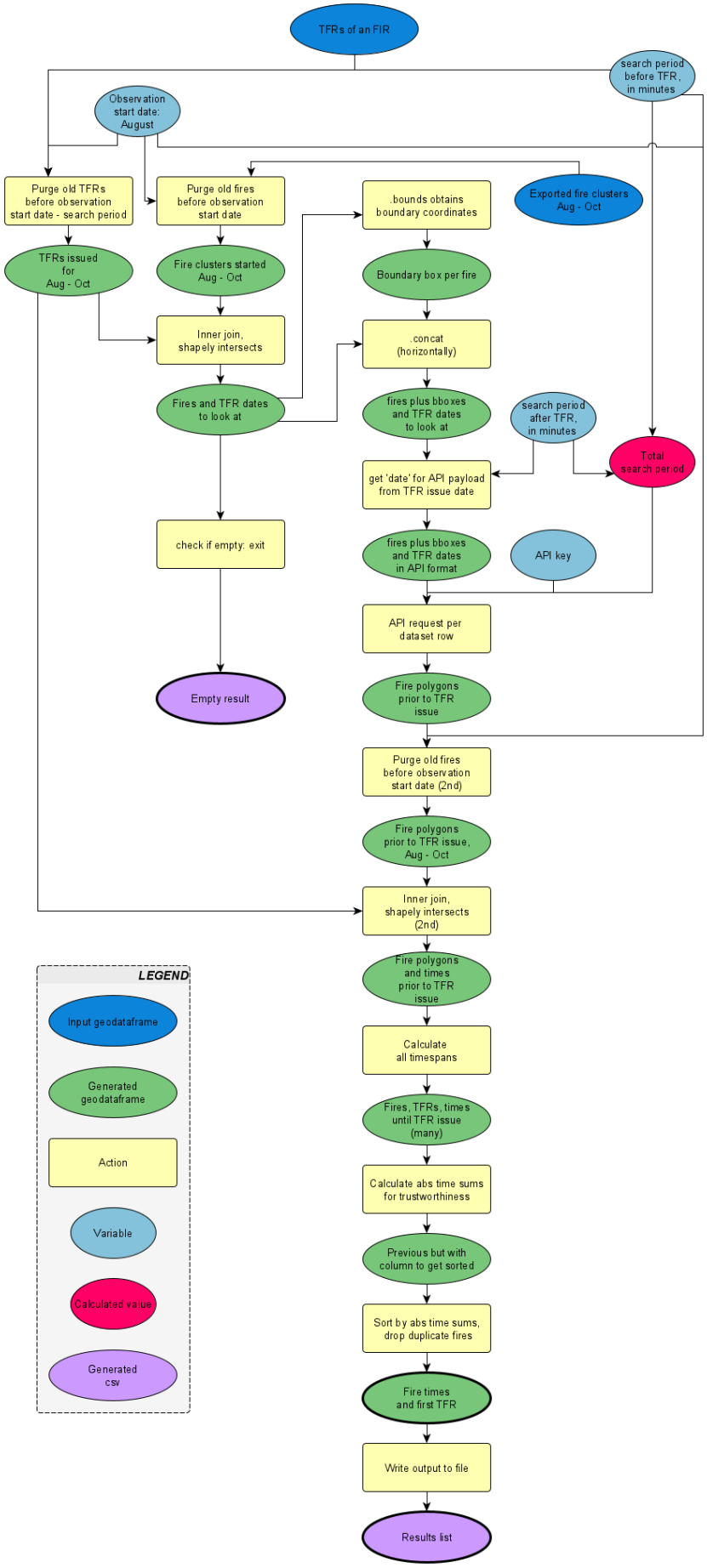


Figure 39 A graph (not a model) representing the steps within Get_Detection-Issue-Time_Gap.ipynb to calculate timespans between a fire cluster´s detection and acquisition time and TFR issue time. After a first guess, response fire clusters from API are used.

4.2.5. Coverage Quality Assessment

Did a TFR area cover enough area to conduct aerial firefighting safely? The developed method is based on the following thoughts: Crowley et al. (2019, p. 305f) showed non-linear growth of the fire perimeter. The shape of a fire can temporally be a multipolygon: Secondary fires can get ignited by spotting downwind of the main fire before fires merge again, Martin & Hillen (2016). Thus, if a safety issue was to be assumed from the comparison of TFR to any fire cluster from the tree-month-dataset, the comparison was necessary to be done again with the fire cluster composed of hotspots being active within a TFR´s valid timeframe.

The objective was to discover situations where a TFR did not cover the area which fire fighting aircraft operations needed, regarding appropriateness definition (4.2.2). Input data were the TFR polygons within an FIR and the exported three-month fire clusters. Then, OroraTech´s API was used to request those parts of fire clusters being active within TFR valid times. These were compared then. In detail:

The fire clusters from the tree-month period were used as first guess. They got buffered by visibility as explained in (4.2.2). Therefore, three runs (0 SM, 1 SM and 3 SM) were performed. Two spatial joins by predicates 'overlap' and 'contains'[29] delivered those TFR candidates that needed a closer look. An accurate fire cluster at a TFR´s valid timeframe was collected via OroraTech´s API: The payload (data to submit) for API requests for parts of fire clusters being active needed a bounding box and a timeframe. The bounding box was derived from the fire clusters, timeframe bounds were limited by TFR´s cancel or expiration time and its effective time. The API request results got buffered again. If there was an overlap between TFRs and API request results (or again a containment), then those TFRs could be considered inappropriate from an aerial firefighting perspective. A graph of what the script was planned to do is shown in Figure 40. The resulting geodataframe was then written to a GeoJSON file and event counts were logged to a text file.

The code is attached as Get_TFR-exceeding_Fires_from_API.ipynb and can also be visited on GitHub or the within the appendix at 8.2.5.

Depending on the situation, this resulting GeoJSON file may have contained duplicates. Follow -up task was to remove those and write the particular column to a csv list to show those fire cluster ids or TFR numbers, where a (buffered) fire cluster had gone outside the TFR (see Get_Events_from_Fires_from_API.ipynb).

Then, to assure that no result (no TFR assessed) was interfered by artificial heat sources or a false-positive, one last small script was applied to count for mismatching types (considering Table 3): Get_Types_from_exceeding_Fires.ipynb.

[29] GeoPandas refers to shapely predicates which can be found here: https://shapely.readthedocs.io/en/stable/manual.html#binary-predicates

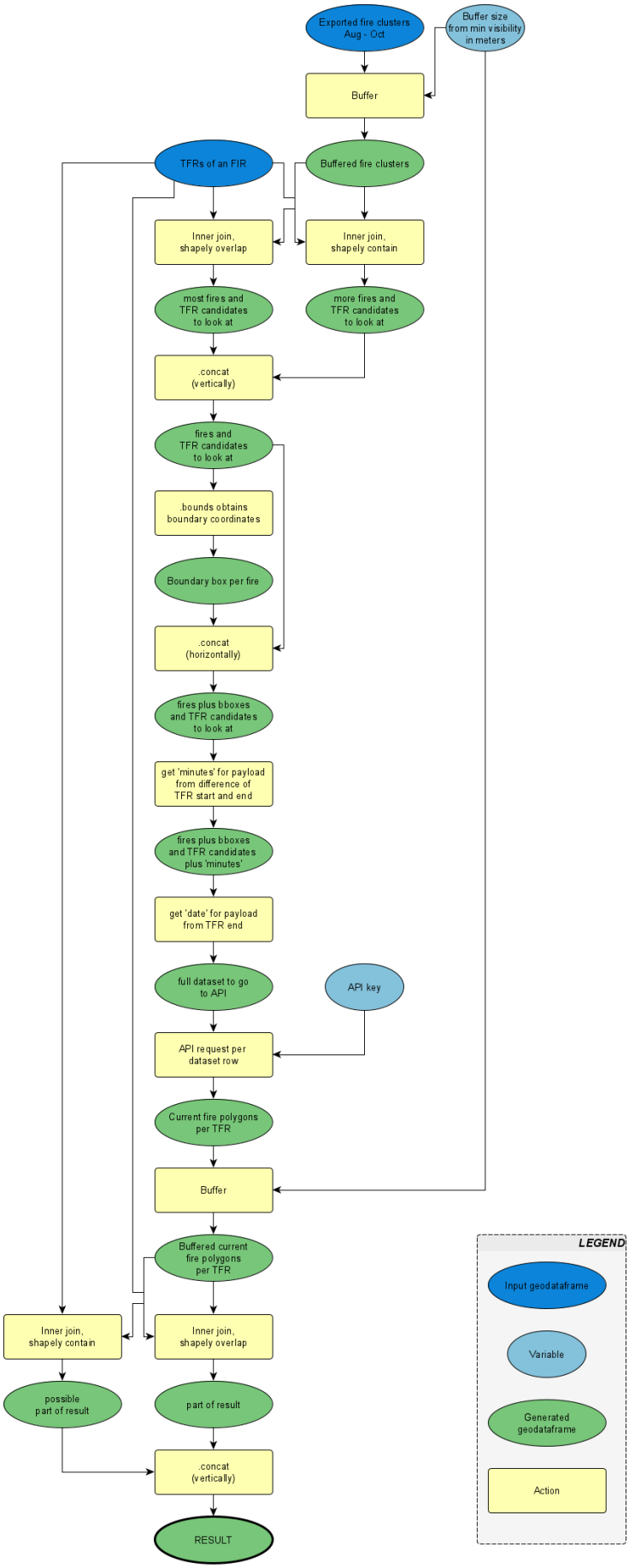


Figure 40 A graph (not a model) representing the steps within Get_TFR-exceeding_Fires_from_API.ipynb to collect TFRs that can be considered inappropriate from an aerial firefighting perspective. A follow up Get_Events_from_Fires_from_API.ipynb then deliver lists of TFRs and fire events

4.2.6. Safety of actual Fire Fighting Aircraft

How often was the presence of aerial firefighters in the vicinity of a fire not covered by a TFR? Having limited the fire fighting aircraft´s state vectors to those most likely being connected to active fires during data exploration (4.2.3.2) lead to split trajectories. So, these were already available for the following analysis. 641 aircraft movements were counted within the observed 10 FIRs within 3 months. The basic question was whether there had been TFRs intersecting the trajectories, being effective during flight time of the trajectories. Having started already with ArcGIS Pro, the *Spatial Join* was performed there (Figure 41).

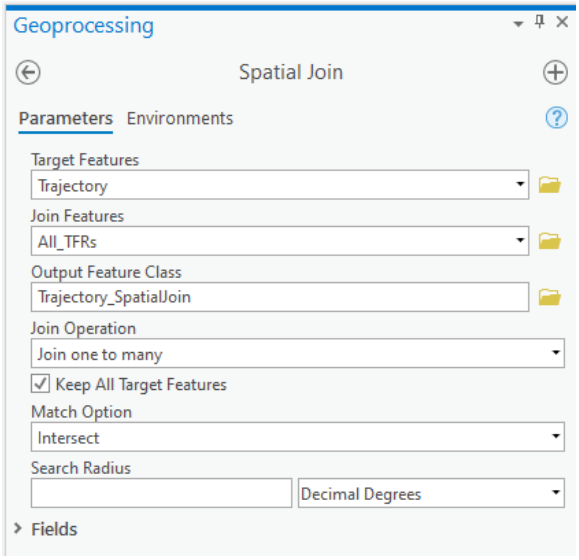


Figure 41 Spatial Join settings to connect TFRs to aircraft trajectories

Where the output for the Join_Count was equal to 0, a firefighter aircraft movement without TFR coverage was found. For the other rows, date calculations were necessary. This would have meant cumbersome steps in ArcGIS Pro with the need to convert dates and select by attributes multiple times. Such calculations were done for the previous objective (4.2.4) in Python, so the data got exported to GeoJSON (Figure 42) and the task was continued with Get_Dates_Aircraft_and_TFRs.ipynb. Here, a function tagged those flights within a TFR´s effective time and those which were not. Clean of duplicates, these were then counted.

Figure 42 Features To JSON setting to export trajectories dataset that has the TFRs joined

4.2.7. Completeness of TFR-Fire-Correspondence

Did each TFR have a corresponding fire cluster? The merged TFR dataset from exploration (4.2.3.1) was reused and processed to GeoJSON in ArcGIS Pro via tool *Features To JSON*. A second three-month-dataset of fire clusters, this time at the lowest confidence (thus, containing more polygons) was prepared. Then, starting with the formerly used three-month-dataset of fires at a confidence of 0.5 and more, a Python script was run to do the following:

- Spatially left join (intersect) all TFRs and fires with the TFRs in first place, identifying solitary TFR without a fire.
- Sort out eventually erroneous TFRs that are cancelled before becoming effective,
- Sort out TFRs issued before fire cluster timeframe (starting 08/01/2021 00:00 UTC) because their comparison is prone to suffering from the boundary value problem that simply no fire clusters are fetched from July.
- The remaining TFRs are now spatially left joined (intersect) on the second three-month-dataset of fire clusters at 0.1 confidence.
- From the created geodataframe those TFRs are picked that still have no fire cluster id associated. These are written into a results GeoJSON file that can be examined.

The Python script is provided by Compare_TFRs_to_Fireclusters.ipynb.

5. Results and Discussion

Results of the steps from the Methods section (4.2) are presented and discussed here. Results from 4.2.1From Text to GeoJSON: Turning TFRs to a spatial Data Format, 4.2.2 Defining Appropriateness of a TFR and 4.2.3 Exploring the Datasets were prerequisite for the treatment of the follow up objectives and thus already presented within their subsections. In summary, only 8 of 10 FIRs contained fire related TFRs.

(Objective 1) Complete TFR data of 10 FIRs from August to October 2021 got examined and converted from text into a spatial data format: 542 TFRs were identified and converted into spatial format. The created TFR dataset can be considered complete, screenshots of locations and date where and when no effective TFR was found are attached to the appendix in 8.5. Generated TFR polygons do occur on a map for the first time within this thesis in Figure 30.

(Objective 2) Concerning appropriateness of a TFR, it can be decided to run the script assessing coverage quality (4.2.5) with the original fire cluster size as well as with buffer sizes of 1 and 3 statute miles.

(Objective 3) During exploration it turned out that these buffer sizes were not prone to producing predefined results, but valuable results could be expected. Available aircraft state vectors remained at a usable sample size of 60 aircraft which had performed presumably 641 movements also when data was spatially and temporally limited to ongoing fires, so the safety of these fire fighting aircraft could be assessed in 4.2.6.

5.1. Results and Discussion of the Time Gap between Fire Detection and TFR Issue

(Objective 4) How long was the timespan between satellite detection and TFR issue time? The script used here (Get_Detection-Issue-Time_Gap.ipynb) generated a result list per FIR considering the first TFR issued for a fire cluster. With the parameters defining the time looked forwards and backwards from a TFR´s issue time set to 24 hours backwards and 6 hours forwards, the appended tables were created. The two columns “Timespan Detection” and “Timespan Acquisition” contain calculated values as minutes (while the geodataframe from the script has additional “d days hh:mm:ss” format) and are presented and uploaded to GitHub as Excel Workbooks for convenience. This is the time, how long it took until an airspace around a fire was secured for firefighting aircraft. “Timespan Detection” refers to the moment, since data was sent and processed and one could actually have known about the fire, while “Timespan Acquisition” refers to the moment when a satellite recorded the fire cluster for the very first time finding its first hotspot. With KZAB having had no intersecting TFRs and KZKC and KZHU having had no wildfire TFRs at all, the result set consists of seven tables (Table 8: KZDV, Table 9: KZFW, Table 10: KZLA, Table 11: KZLC, Table 12: KZMP, Table 13:KZOA, Table 14: KZSE, all appended in section 8.3).

There were incidents where a TFR was issued based on ground information while satellites had not discovered a fire yet. So, a TFR dedicated to a fire cluster might have become issued hours before the cluster appeared within OroraTech´s data. If this had happened more hours before TFR issue time than it was set via parameter of the script, the cluster was omitted from the results list because the API request going back from the TFR issue time did not retrieve it. This case is represented by a large negative value in the results tables with no better temporal matching of the same TFR. Within the API request, going forward a few hours as well could also match these cases, a screenshot of an example is provided by Figure 43. This way, reasonable (up to half a day) negative values imply, that the TFR had been issued before a satellite acquired it (Timespan Acquisition) and/or before one could have known about it from satellite data (Timespan Detection). Out of 240 fires, there were only 18 incidents

where a TFR had been issued before a satellite acquired a fire and 26 incidents where a TFR had been issued before the Wildfire Service could detect and inform about a fire. In the data, there are 3 occasions that a TFR was issued within 1 hour after a satellite had acquired the fire. So, for 219 of these fires, a TFR was probably issued hours after a satellite had acquired the fire, meaning it took hours, until the airspace was secured for aerial firefighting. It is not known whether this was due to complications in TFR creation process or due to late discovery of the fire via the currently utilized means. See Conclusions at 6.1 for some more in-depth thoughts.

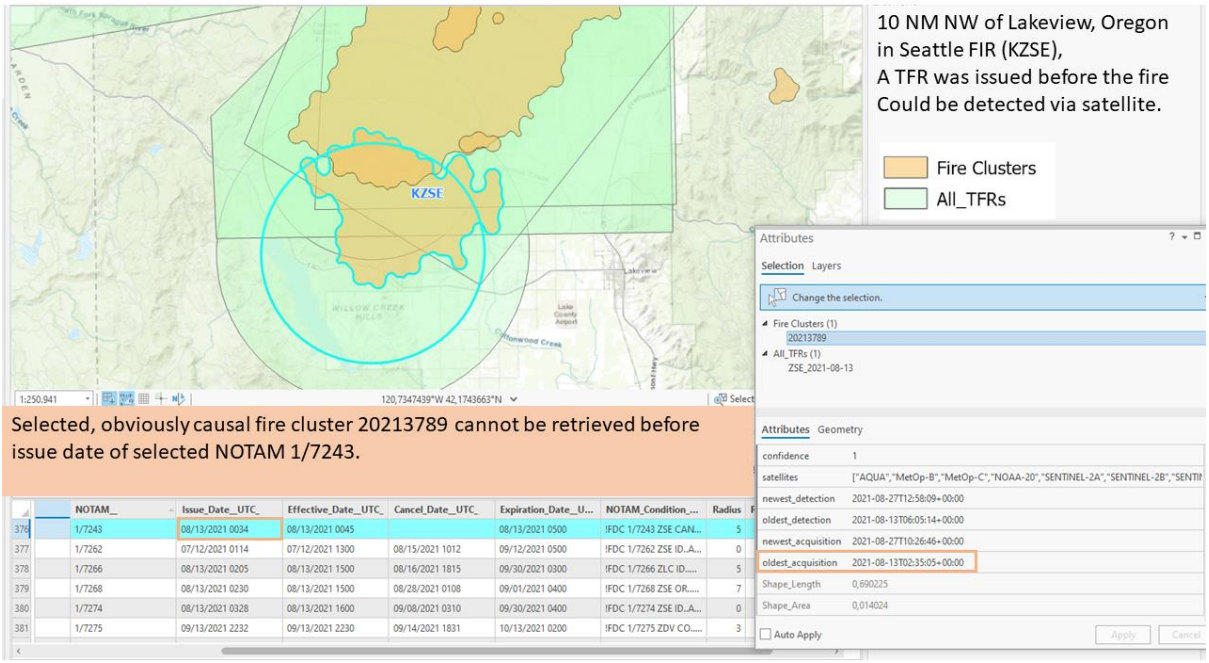


Figure 43 The fire cluster with the id 20213789 would be omitted in the results (Table 14) if the API request from the used script had only looked backwards from the issue date of the TFR. Instead, the large cluster to the north (id 21250885) from September would become tied to NOTAM 1/7243 because it intersects it. Going forward a few hours as well now leads to both ids being listed in Table 14.

Multiple occurrences of the same TFR had to stay allowed within the results (intended by regulations, see 3.2). The fire with the smallest absolute values in acquisition and detection timespan was most probably the event that triggered the TFR. To not pretend a certainty that cannot be guaranteed, duplicated TFRs were not removed from the results set.

The listed fire type shows that none of these first response TFRs has been issued for a known false alarm or permanent hotspot while the latter has become unlikely during script development because fire clusters with an old acquisition time were dropped anyway.

5.2. Results and Discussion of Coverage Quality Assessment

(Objective 5) Did a TFR area cover enough area to conduct aerial firefighting safely? If a TFR is not setup properly or a fire spreads out of it, the TFR may not protect firefighting aircraft anymore. There were cases where an unbuffered fire cluster was not totally covered by a TFR. Regarding minimum flight visibility (compare 4.2.2), the amount of inappropriate TFRs did even rise. The applied script (Get_TFR-exceeding_Fires_from_API.ipynb) logged the numbers per TFR and buffer size (as visibility-distance) as shown in Table 6. The simplest case occurred in Ft Worth FIR (ZFW): 1 fire event got 3 consecutive TFRs issued of which none was sized appropriately to contain the fire cluster during TFR effective time (Figure 44). The full result log is appended as Text 3 within 8.3.

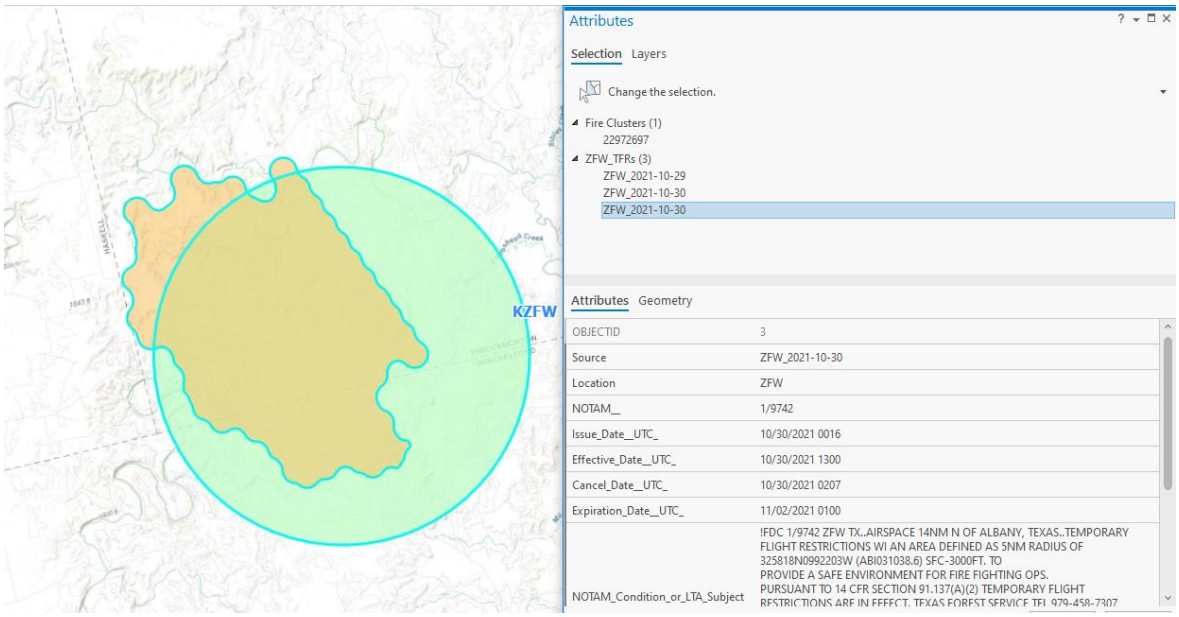


Figure 44 Screenshot of results for ZFW in ArcGIS Pro: All three TFRs from the Attributes pane were issued as the green-filled circle, which does not contain the entire orange fire cluster being active during these TFRs´ effective time

Table 6 Results of coverage quality assessment

FIR	TFRs evaluated	TFRs cancelled before being effective	Buffer Distance [SM]	TFRs where a fire leaves it	fire events leaving a TFR
ZDV	28	1	0	1	1
			1	1	1
			3	5	3
ZFW	3	1	0	3	1
			1	3	1
			3	3	1
ZLA	40	0	0	12	5
			1	23	14
			3	37	32
ZLC	114	1	0	34	24
			1	43	37
			3	61	81
ZMP	10	1	0	5	2
			1	7	2
			3	7	2
ZOA	99	2	0	71	34
			1	89	92
			3	96	162
ZSE	247	7	0	145	75
			1	180	177
			3	208	298

If multiple fire clusters and TFRs overlapped, the arising interim geodataframe got many rows, which happened more often in Seattle FIR (KZSE, screenshot in Figure 45) and Oakland FIR (KZOA). This led to large result counts and even more fire events leaving a TFR than TFRs had been included. So, a follow-up script (Get_Events_from_Fires_from_API.ipynb) was coded to provide GeoJSON output and a list containing resulting fire cluster ids and NOTAM numbers. This can enable authorities that have detailed knowledge about intentions behind a single TFR to investigate further. None of the results was influenced by artificial (permanent) heat sources or false positives, Get_Types_from_exceeding_Fires.ipynb checked for such types.

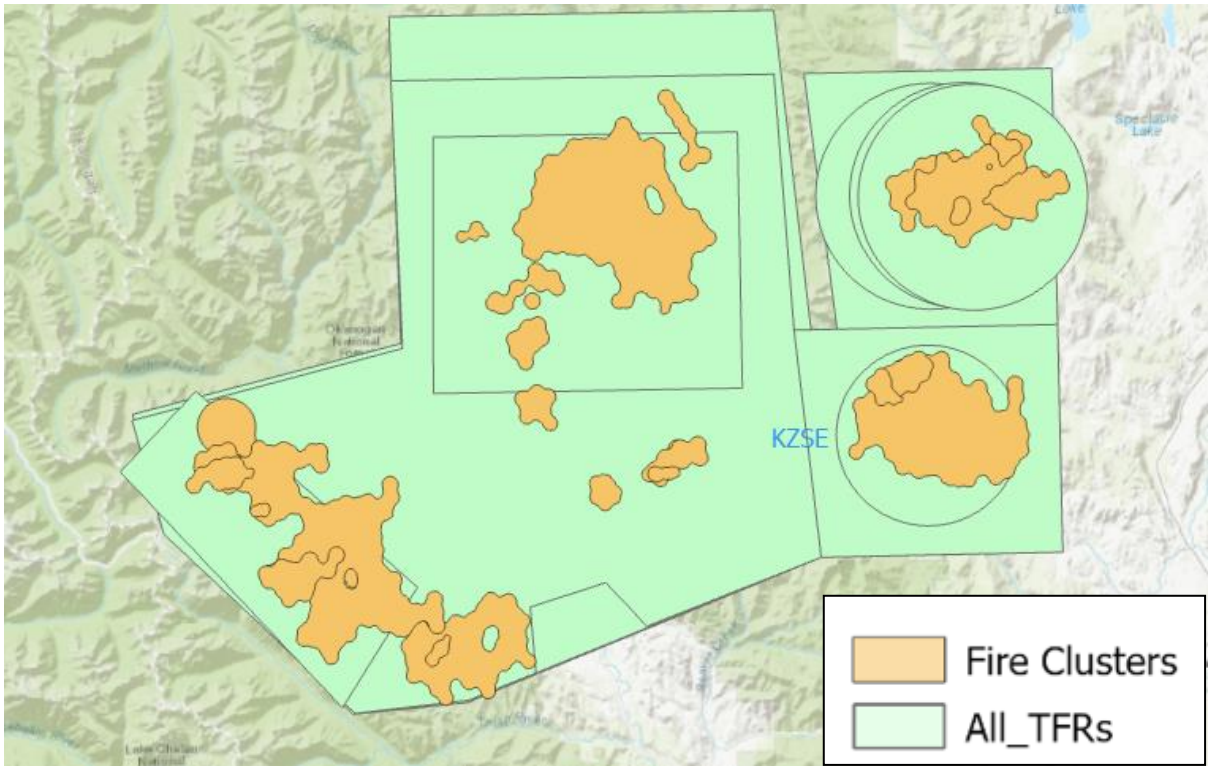


Figure 45 Multiple TFRs (green) issued over the three months in close vicinity with numerous, still unbuffered fire clusters (orange) that already overlap the TFRs can be found in Seattle FIR (KZSE). These clusters do cause a bloated first guess geodataframe.

5.3. Results and Discussion of Safety of actual Fire Fighting Aircraft

(Objective 6) How often was the presence of aerial firefighters in the vicinity of a fire is not covered by a TFR? Figure 46 shows rounded results as pie chart.

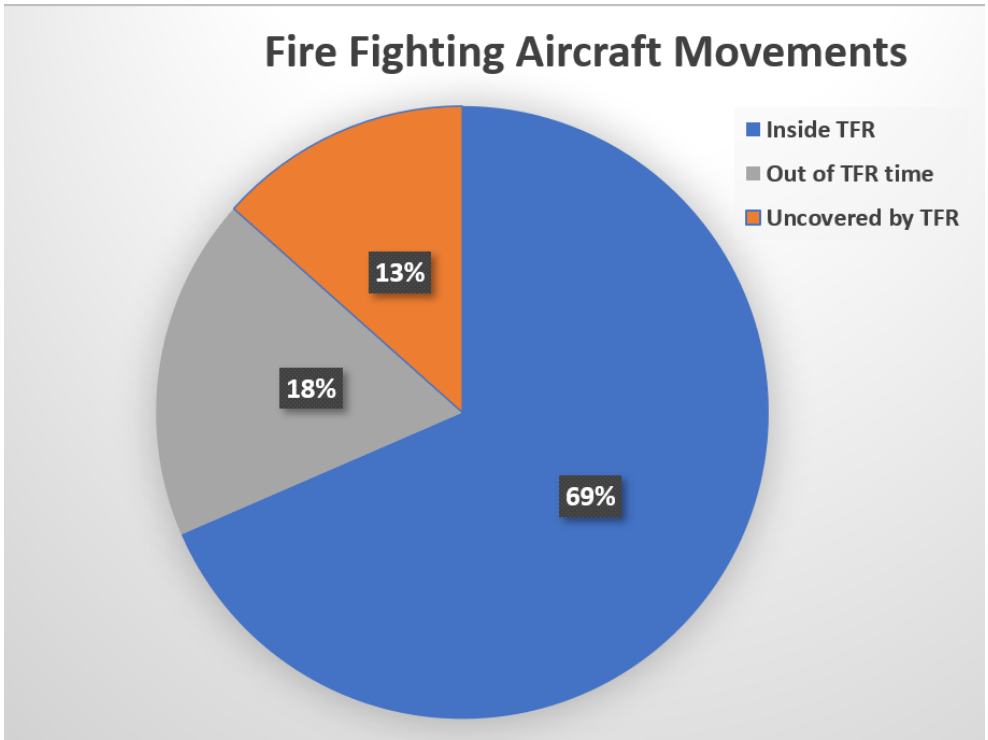


Figure 46 Movements of fire fighting aircraft inside TFRs (69%), out of effective time (18%) and totally uncovered (13%)

Splitting trajectories of 60 fire fighting aircraft led to 641 aircraft movements that were examined. The trajectories were created from aircraft locations not exceeding a distance of 3 miles from the three-month-fire-clusters. Only aircraft locations with a timestamp in between oldest and newest fire detection were considered. As of Get_Dates_Aircraft_and_TFRs.ipynb, 439 (ca. 69%) of the aircraft movements were within a spatially joined TFR´s effective time. These movements were considered as conducted in safe conditions. 116 (ca. 18%) aircraft movements were within TFRs spatially but not within any TFR´s effective time. A TFR issue had probably been too late here. 86 (ca. 13%) aircraft movements were not covered by any TFR at all. Because the trajectory IDs still contain the aircraft id, it can be said that within the 86 totally uncovered movements 42 of the 60 aircraft had to operate at least once outside of any TFR coverage with all the dangers that this implies. Pilots of drones and manned aircraft were not knowing that the firefighters were there. An excerpt from the data is depicted in Figure 47.

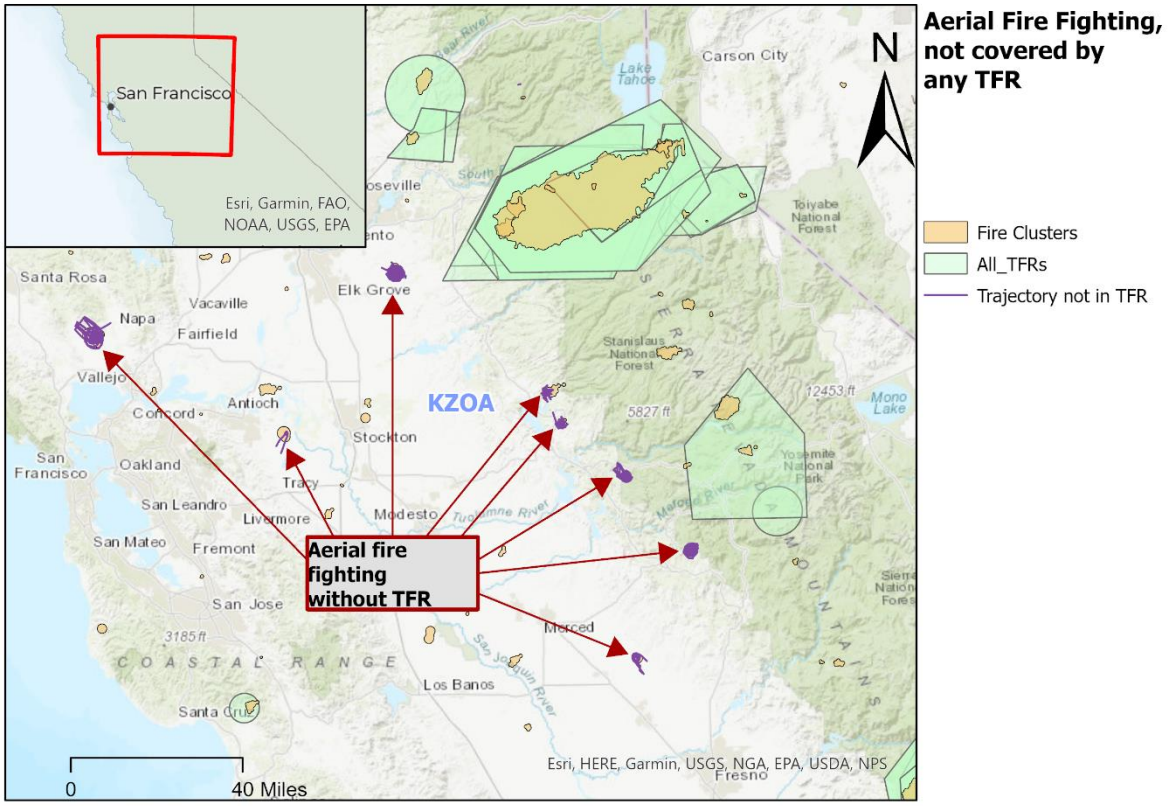


Figure 47 The map shows a part of California within Oakland FIR (KZOA). Trajectories distinct from TFRs but related to active fire clusters (within their detection time range) are shown. The movement patterns show that the aerial firefighters were not just passing by the fires by chance.

Whether all uncovered trajectories were indeed dedicated to firefighting at that moment when the state vectors were captured remains uncertain. For how many of the uncovered aircraft movements this state was intended or just a lack of the complicated request and creation process was not examined in this study.

No fire related trajectories were found within the two FIRs that did not have fire related TFRs issued (Houston KZHU and Kansas City KZKC).

5.4. Results and Discussion of Completeness of TFR-Fire-Correspondence

(Objective 7) Did each TFR have a corresponding fire cluster? The used Python script (Jupyter Notebook Compare_TFRs_to_Fireclusters.ipynb) detected 56 TFRs that could not be tied to a fire id from the tree-month-dataset at 0.5 confidence. None of these TFRs had been cancelled before they became effective, so they were not to be considered erroneous. 17 TFRs had already been issued prior to the observed timespan. They were dropped from examination because related fire clusters were not necessarily contained within examined dataset starting observation August 1st.

From the remaining 39 questionable TFRs, fire clusters at lower confidence down to 0.1 were found at least spatially related for 36 TFRs (out of originally 542 TFRs). Finally, for 3 TFRs, no associated fire cluster was found. From the created geodataframe, those TFRs were picked that still had no fire cluster id associated. These were written into a results GeoJSON file to be examined. Table 7 holds the details.

Table 7 TFRs that do not have a fire cluster associated

FIR	NOTAM	Issue_Date	Location from NOTAM Text	Center Coordinates from NOTAM Text
ZSE	1/9095	08/01/2021 1555	22NM S MEDFORD	420048N1225724W
ZAB	1/5987	10/20/2021 2254	23NM SE OF TUCSON	314232N1105136W
ZAB	1/6009	10/21/2021 0034	23NM SE OF TUCSON	314232N1105136W

6. Conclusion

The goal was to examine the relation between satellite-based fire observation data and restricted airspace for aerial firefighting to assess safety of the involved aircraft. A study area was defined to be the 10 westerly U.S. FIRs. Fire cluster polygons from that study area to be considered active fires were drawn from OroraTech´s Wildfire Service. A complete set of TFR texts from the FAA NOTAM Archive for the study area was turned into GeoJSON format successfully, combining manual downloading with VBA, Power Queries and Python. It was possible to overcome data quality issues by code and with manual effort. Aircraft state vectors (aircraft locations) of aerial fire fighters have been available since July of 2021. Thus, the examined time frame was chosen to be August until (and including) October 2021.

Available datasets were explored in a what-you-see-is-what-you-get-fashion with ArcGIS Pro to enable for designing the following workflows. It was found that, to gather enumerable entities, aircraft state vectors needed to become connected to trajectories which got then split at every 1-hour time gap to represent aircraft movements.

6.1. Concerning the Time Gap between Fire Detection and TFR Issue Time

Concerning the time gap between fire detection and TFR issue time, the results set was not narrowed down or evaluated any further. Out of 240 fires, a first TFR was issued prior to fire dates 18 times concerning acquisition and 26 times concerning detection. For the other captured cases, it took several minutes up to days until a TFR provided a safe environment for aerial fire fighting operations. Longer waiting times before a TFR got issued might depend on the single case or the complicated creation process. It was possibly the case that aviation operations were planned to take place later, due to weather conditions or lacking resources. Also, the fact that most aerial firefighting is usually conducted during daytime was not yet incorporated: A fire detected at night might not require immediate action. TFRs occurring only once within the results might deliver skewed timespan values due to the TFR originally being issued for a fire cluster that had lower confidence than the used 0.5-dataset. TFRs issued across FIR borders caused confusion here. An example is fire cluster id 21370680 that received attention from both sides: KZLA and KZOA with the first TFR issued from KZOA side. This has led to a skewed result in the KZLA data (Table 10). For negative values, the related fire might have occurred within an already existing TFR. The outcome here has become a model approximating the true fire and TFR combinations. This model does produce outliers but will improve along with OroraTech´s data density when the nanosatellites start work in near future. Up to now it seems not eligible to compute one overall result.

6.2. Concerning Coverage Quality

A number of mismatching combinations of fire clusters and TFRs was discovered and logged (Text 3 at 8.4). Where the spatial relation of fire clusters and TFRs was 1:1, results are trustworthy. This was not found to be the case within all FIRs so it was not eligible to compute one overall result as valid percentage.

It is not known when in its lifetime a TFR became inappropriate. A follow up inspection in a GIS must be made to know whether a subsequent TFR was issued. Did the buffered fire cluster intersect with another TFR that got issued later than the first one? This question is still open.

When multiple fire clusters and TFRs occurred close together (compare Figure 45 showing obvious issues during evaluation of coverage quality), result counts could get flawed, when buffered fire clusters had multiple overlaps or might even contain entire TFRs (Figure 48). An issue with the designed workflow is, that a bounding box was needed by the API to retrieve the current fire clusters.

Scanning this boundary box might lead to more response results as a request per fire cluster id would. An improvement to the API to provide fire entity snapshots just via id and timeframe could be helpful. Or this could possibly become solved with an entirely changed workflow, storing even more data to create time slices to perform the analysis. If a dedicated join attribute to connect fire clusters and TFRs (in another way than spatially) could be invented for future data, a clear assignment would be possible as well.

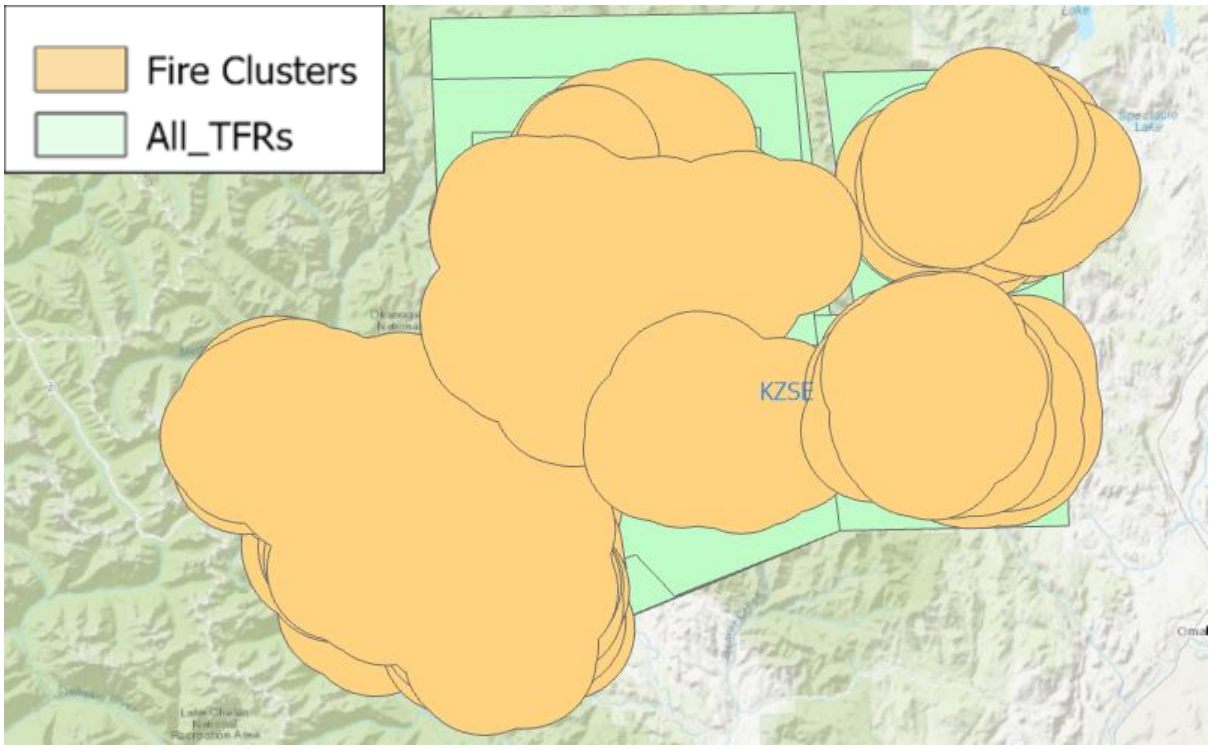


Figure 48 With the same scope as in Figure 45, resulting buffered fire clusters with detection time within formerly overlapped (or contained) TFR´s active time become numerous. For this screenshot the buffer size is exaggerated to ca. 3.5 miles to demonstrate the effect of possible multiple intersects leading to the large case values from Text 3 at 8.4 and aircraft buffered fire clusters as aircraft acting areas covering (almost) entire TFRs they do not "belong" to.

Also, factual correct coverage issues found here might not necessarily mean a danger or could have even been intended: If wind direction was considered, it would be conceivable that considering not the entire fire area but a downwind section of it is the correct place to issue a TFR.

6.3. Concerning safety of actual Fire Fighting Aircraft

There is no information about whether an absence of airspace dedication has been intended in any or even all the observed cases. However, only 68.5% of the fire fighter aircraft movements within 3 statute mile vicinity of an active fire were made under TFR protection while 31.5% were not. These distribute as follows: 18.1% were made in an only previously or afterwards covered area, 13.4% movements were flown in an area not covered at all. 42 of the 60 aircraft considered here were uncovered by a TFR at least once within the observed three months.

For higher accuracy, it would be applicable to involve all airspace classes: A flight considered uncovered so far may have been conducted in a controlled airspace (categories "B", "C", "D", compare Figure 28) close to an airport. There, it is air traffic control taking care of aircraft separation. And UAS pilots have to acquire permission to fly there.

6.4. Concerning Completeness of TFR-Fire-Correspondence

Fire cluster data from OroraTech can be considered not entirely complete but dense enough for this research with only 3 TFRs not related. With the provided information, the cause can be investigated. Example:

For the assumed fire in Seattle FIR (ZSE) in Oregon, 22 nautical miles south of Medford, meteorological data from the nearest airport (KMFR, Rogue Valley International-Medford Airport) can be retrieved via ogimet.com:

http://www.ogimet.com/display_metars2.php?lang=en&lugar=KMFR&tipo=ALL&ord=REV&nil=SI&fmt=html&ano=2021&mes=07&day=31&hora=08&anof=2021&mesf=08&dayf=01&horaf=20&minf=59&send=send

With skies being “CLR” most of the time around TFR issue time, cloud cover will not be the reason for probably missing a fire with satellite detection. Satellite orbits can be checked next then.

6.5. Prospect of future Work and Data Application

All in all, this research copes with two kinds of linkage problems: Neither are consecutive TFRs linked in any way with each other, nor are the causative events connected to one or more TFRs. The question can be raised, whether a higher percentage of aerial firefighter movements could be conducted under coverage of an appropriate TFR, if a consistent database of fire events was involved into the TFR creation process. From a research perspective, it would be a good reason to link wildfire TFRs to OroraTech´s fire cluster identifiers right from the start: This would enable for more detailed research as well as for managing wildfire TFRs. Satellite data can serve as uniform source for monitoring airspace restrictions. This does already seem reasonable with the current satellite data available: Out of the 240 fires for which a first TFR got identified, 214 fires (89%) had been detected by the Wildfire Service before a TFR was issued, at least with the applied model parameters. Yet, not all fires were covered by the Wildfire Service as fast as some TFRs got issued. But the cube satellites that will be launched in the future are expected to improve data density a lot.

Improved satellite fire monitoring can have indirect impact on airspace management for aerial firefighting. The satellite data is supposed to help simulating fires like Mutthulakshmi et al. (2020) did. This, in turn, helps improving fire fighting strategies. Additionally, TFRs could be issued based on fire spread forecasts one day.

To improve the situation of the coverage quality of the TFRs, one could check the radius size of the circular TFRs, whether the detected issues occur statistically accumulated at certain, especially small radius sizes (e.g., below the recommended 5 NM minimum).

Considering the number of aircraft, a sample size of currently 60 could be increased to gather more robust results. At least the result set narrows down a list to probably start a case study, contacting the companies and authorities the 60 aircraft belong to. The operations from state vector exploration (4.2.3.2) are worth being put into a script as well. Another GIS approach could be the following: Excluding the issue with ADS-B blind areas for low flying aircraft over structured terrain for a moment, findings from Olive et al. (2020) could be used to identify actually fire fighting aircraft even more precisely from the results gathered by this thesis.

If the overall TFR creation process has to stay as is, at least an automatic fill out of the TFR request form could be designed: https://www.nwcg.gov/sites/default/files/committee/docs/iasc-interagency-tfr-request-form.pdf

OroraTech´s Wildfire Service provides more data attributes than used here. One is the fire radiative power. Using this data, another potential research goal can be, whether and when it can be foreseen, if aerial firefighting becomes necessary for a fire pattern.

For the future, there is room for improvement in airspace restriction for aerial firefighting and data from OroraTech can foster both, further research and TFR management, and thus enhance aviation safety.

7. References

Barmpoutis, P., Papaioannou, P., Dimitropoulos, K., & Grammalidis, N. (2020). A Review on Early Forest Fire Detection Systems Using Optical Remote Sensing. *Sensors*, *20*(22), 6442. https://doi.org/10.3390/s20226442

Crowley, M. A., Cardille, J. A., White, J. C., & Wulder, M. A. (2019). Multi-sensor, multi-scale, Bayesian data synthesis for mapping within-year wildfire progression. *Remote Sensing Letters*, *10*(3), 302–311. https://doi.org/10.1080/2150704X.2018.1536300

de Almeida Pereira, G. H., Fusioka, A. M., Nassu, B. T., & Minetto, R. (2021). Active fire detection in Landsat-8 imagery: A large-scale dataset and a deep-learning study. *ISPRS Journal of Photogrammetry and Remote Sensing*, *178*, 171–186. https://doi.org/10.1016/j.isprsjprs.2021.06.002

European Commission. (n.d.-a). *Copernicus EMS - EFFIS - Active Fire Detection*. Retrieved March 24, 2021, from https://effis.jrc.ec.europa.eu/about-effis/technical-background/active-fire-detection

European Commission. (n.d.-b). *Copernicus EMS - EFFIS - Brief History*. Retrieved March 30, 2021, from https://effis.jrc.ec.europa.eu/about-effis/brief-history

FAA. (2021, December 2). *ENR 5.1 Prohibited, Restricted, and Other Areas*. Aeronautical Information Publication. https://www.faa.gov/air_traffic/publications/atpubs/aip_html/part2_enr_section_5.1.html

FAA. (2022a, March 21). *14 CFR 91.137—Temporary flight restrictions in the vicinity of disaster/hazard areas.* https://www.ecfr.gov/current/title-14/chapter-I/subchapter-F/part-91/subpart-B/subject-group-ECFRe4c59b5f5506932/section-91.137

FAA. (2022b, June 2). *14 CFR 91.155—Basic VFR weather minimums.* https://www.ecfr.gov/current/title-14/chapter-I/subchapter-F/part-91/subpart-B/subject-group-ECFR4d5279ba676bedc/section-91.155

FAA National Headquarters. (2021, December 2). *FAA Order 7930.2S*. https://www.faa.gov/air_traffic/publications/atpubs/notam_html/

Giglio, L., Schroeder, W., & Justice, C. O. (2016). The collection 6 MODIS active fire detection algorithm and fire products. *Remote Sensing of Environment*, *178*, 31–41. https://doi.org/10.1016/j.rse.2016.02.054

Graser, A. (2019). MovingPandas: Efficient Structures for Movement Data in Python. *GI_Forum 2019, Volume 7,* 54–68. https://doi.org/10.1553/giscience2019_01_s54

Graser, A., & Dragaschnig, M. (2020). *Exploring movement data in notebook environments*. 5.

Hall, J. V., Zhang, R., Schroeder, W., Huang, C., & Giglio, L. (2019). Validation of GOES-16 ABI and MSG SEVIRI active fire products. *International Journal of Applied Earth Observation and Geoinformation*, *83*, 101928. https://doi.org/10.1016/j.jag.2019.101928

Hantson, S., Padilla, M., Corti, D., & Chuvieco, E. (2013). Strengths and weaknesses of MODIS hotspots to characterize global fire occurrence. *Remote Sensing of Environment*, *131*, 152–159. https://doi.org/10.1016/j.rse.2012.12.004

Hoeft, R. M., Kochan, J. A., & Jentsch, F. (2005). A Human Factors Analysis of the Current U.S. Notices to Airmen (NOTAM) System. *The International Journal of Aviation Psychology*, *15*(1), 91–109. https://doi.org/10.1207/s15327108ijap1501_5

Kluyver, T., Ragan-Kelley, B., Perez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., & [Unknown, J. (2016, January 1). *Jupyter Notebooks – a publishing format for reproducible computational workflows*.

Knox, E. G., & Bartlett, M. S. (1964). The Detection of Space-Time Interactions. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *13*(1), 25–30. https://doi.org/10.2307/2985220

Lilly, B., Cetinkaya, D., & Durak, U. (2021). Tracking Light Aircraft with Smartphones at Low Altitudes. *Information*, *12*(3), 105. https://doi.org/10.3390/info12030105

Martin, J., & Hillen, T. (2016). The Spotting Distribution of Wildfires. *Applied Sciences*, *6*(6), 177. https://doi.org/10.3390/app6060177

Mutthulakshmi, K., Wee, M. R. E., Wong, Y. C. K., Lai, J. W., Koh, J. M., Acharya, U. R., & Cheong, K. H. (2020). Simulating forest fire spread and fire-fighting using cellular automata. *Chinese Journal of Physics*, *65*, 642–650. https://doi.org/10.1016/j.cjph.2020.04.001

National Wildfire Coordinating Group. (2018). *NWCG Standards for Airspace Coordination*. 171.

National Wildfire Coordinating Group. (2022a, January). *NWCG Standards for Aerial Supervision*. https://www.nwcg.gov/publications/505

National Wildfire Coordinating Group. (2022b). *NWCG Fire Traffic Area (FTA)*. https://www.nwcg.gov/sites/default/files/publications/pms-505d.pdf

Olive, X., Sun, J., Lafage, A., & Basora, L. (2020). Detecting Events in Aircraft Trajectories: Rule-Based and Data-Driven Approaches. *Proceedings*, *59*(1), 8. https://doi.org/10.3390/proceedings2020059008

Ostermann, F., Nüst, D., Granell, C., Hofer, B., & Konkol, M. (2020). *Reproducible Research and GIScience: An evaluation using GIScience conference papers* [Preprint]. Geographic Information Sciences. https://doi.org/10.31223/X5ZK5V

Robertson, C., Nelson, T. A., Boots, B., & Wulder, M. A. (2007). STAMP: Spatial–temporal analysis of moving polygons. *Journal of Geographical Systems*, *9*(3), 207–227.

Schroeder & Giglio. (2017). *Visible InfraredImaging Radiometer Suite (VIIRS) 750 m Active Fire Detection and Characterization Algorithm Theoretical Basis Document 1.0*. NASA. https://viirsland.gsfc.nasa.gov/PDF/VIIRS_activefire_750m_ATBD.pdf

Schroeder, W., Oliva, P., Giglio, L., & Csiszar, I. A. (2014). The New VIIRS 375m active fire detection data product: Algorithm description and initial assessment. *Remote Sensing of Environment*, *143*, 85–96. https://doi.org/10.1016/j.rse.2013.12.008

Schroeder, W., Oliva, P., Giglio, L., Quayle, B., Lorenz, E., & Morelli, F. (2016). Active fire detection using Landsat-8/OLI data. *Remote Sensing of Environment*, *185*, 210–220. https://doi.org/10.1016/j.rse.2015.08.032

Tang, Y., Zhong, S., Luo, L., Bian, X., Heilman, W. E., & Winkler, J. (2015). The Potential Impact of Regional Climate Change on Fire Weather in the United States. *Annals of the Association of American Geographers*, *105*(1), 1–21. https://doi.org/10.1080/00045608.2014.968892

US EPA, O. (2021, April). *Climate Change Indicators: Wildfires* [Reports and Assessments]. https://www.epa.gov/climate-indicators/climate-change-indicators-wildfires

USDA, Forest Service. (2021). *Interagency Aviation Information Bulletin No. IA IB 21-06*.

USDA FOREST SERVICE, DEPARTMENT OF INTERIOR. (1998). *NATIONAL STUDY OF TACTICAL AERIAL RESOURCE MANAGEMENT TO SUPPORT INITIAL ATTACK AND LARGE FIRE SUPPRESSION*. https://www.fs.usda.gov/sites/default/files/media_wysiwyg/tarms.pdf

Wooster, M. J., Xu, W., & Nightingale, T. (2012). Sentinel-3 SLSTR active fire detection and FRP product: Pre-launch algorithm development and performance evaluation using MODIS and ASTER datasets. *Remote Sensing of Environment*, *120*, 236–254. https://doi.org/10.1016/j.rse.2011.09.033

Wotton, B. M., Flannigan, M. D., & Marshall, G. A. (2017). Potential climate change impacts on fire intensity and key wildfire suppression thresholds in Canada. *Environmental Research Letters*, *12*(9), 095003. https://doi.org/10.1088/1748-9326/aa7e6e

YUAN, M. (1997). Use of knowledge acquisition to build wildfire representation in Geographical Information Systems. *International Journal of Geographical Information Science*, *11*(8), 723–746. https://doi.org/10.1080/136588197242059

8. Appendix

8.1. GitHub

It is planned to provide access to code via GitHub at

https://github.com/weatherfire/Master-Thesis

8.2. Scripts and Code

VBA, Power Queries and Python were used in this thesis.

A word on Python:

`Python 3.10.4` was used and managed via Anaconda `conda 4.12.0`.

From Anaconda Prompt, the first module installed: Geopandas requires

```
(ENVIRONMENT) C:\PATH>conda config --env --add channels conda-forge
(ENVIRONMENT) C:\PATH>conda config --env --set channel_priority strict
(ENVIRONMENT) C:\PATH>conda install python=3 geopandas
```

Installation of other imported modules is straightforward via

```
conda install -c conda-forge MODULENAME
```

PyCharm (https://www.jetbrains.com/pycharm/) and Jupyter Notebooks (https://jupyter.org/) were used for development and documentation here. The latter can be installed via coda as well. A .bat file can be created as "desktop starter" with the following (example) content:

```
call C:\anaconda3\Scripts\activate ENVIRONMENT
jupyter notebook
```

Python code converted from Jupyter Notebooks is available here while it is recommended to visit the original at GitHub.

`pandoc jupyter_file.ipynb -s -o new_word_file.docx` is used to append readable and copyable Jupyter Notebook content here (except for Aircraft_ovr_fires_to_trajectories.ipynb).

8.2.1. Relating 4.1.4, The Study Area as FIRs

The study area gets retrieved by Get_FIRs_as_WKT.ipynb as Code 1, with omitted code parts (~60 pages WKT) indicated by […].

A script to be run in ArcGIS Pro to gather WKT from FIR feature class

Printing feature classes in the current gdb helps what to choose in the next step

```
for fc in arcpy.ListFeatureClasses():
 print(fc)
```

```
_25811702_2022_01_24_22_36_14
_25811702_2022_01_24_22_36_14_MultipleRingBuffer
FIRs_NA_3line_FeatureToPolyg
FIRs_FeatureToPolyg
FIRs_Boundary
FIRs_FeatureToPolyg_Dissolve
```

Iterating through the feature class enables for either one (Boundary of all FIRs) or multiple WKT outputs

```
 for row in arcpy.da.SearchCursor("FIRs_FeatureToPolyg_Dissolve", ["OID@",
"SHAPE@WKT"]):
  # Print feature ID
  print("FIRs_FeatureToPolyg_Dissolve", " Feature {}:".format(row[0]))
  # Print geometry as WKT
  print (row[1])
```

```
FIRs_FeatureToPolyg_Dissolve  Feature 1:
MULTIPOLYGON (((-95.146720886059995 49.380138397369194, […], -95.146720886
059995 49.380138397369194)))
```

```
 for row in arcpy.da.SearchCursor("FIRs_FeatureToPolyg", ["OID@", "SHAPE@W
KT"]):
  # Print feature ID
  print("FIRs_FeatureToPolyg", " Feature {}:".format(row[0]))
  # Print geometry as WKT
  print (row[1])
```

```
FIRs_FeatureToPolyg  Feature 1:
MULTIPOLYGON (((-91.308334351049439 31.912502288949554, […], -91.308334351
049439 31.912502288949554)))
FIRs_FeatureToPolyg  Feature 2:
MULTIPOLYGON (((-95.366569519467703 35.871002197293649, […], -95.366569519
467703 35.871002197293649)))
FIRs_FeatureToPolyg  Feature 3:
MULTIPOLYGON (((-101.74999999980014 36.500000000000057, […], -101.74999999
980014 36.500000000000057)))
FIRs_FeatureToPolyg  Feature 4:
MULTIPOLYGON (((-113.71666717524988 37.799999236575559, […], -113.71666717
524988 37.799999236575559)))
FIRs_FeatureToPolyg  Feature 5:
MULTIPOLYGON (((-89.750000000399666 40.000000000199918, […], -89.750000000
399666 40.000000000199918)))
FIRs_FeatureToPolyg  Feature 6:
```

```
MULTIPOLYGON (((-121.25000000039967 41.000000000000057, […], -121.25000000
039967 41.000000000000057)))
FIRs_FeatureToPolyg  Feature 7:
MULTIPOLYGON (((-104.2500000001998 45.116668701099741, […], -104.250000000
1998 45.116668701099741)))
FIRs_FeatureToPolyg  Feature 8:

MULTIPOLYGON (((-114.32132720998271 49.000835418220049, […], -114.32132720
998271 49.000835418220049)))

FIRs_FeatureToPolyg  Feature 9:
MULTIPOLYGON (((-122.20121765177316 49.002441405946172, […], -122.20121765
177316 49.002441405946172)))

FIRs_FeatureToPolyg  Feature 10:
MULTIPOLYGON (((-95.150306701592967 49.382999420486897, […], -95.150306701
592967 49.382999420486897)))
```

Code 1 Get_FIRs_as_WKT.ipynb to be run in ArcGIS Pro to gather WKT from FIR feature class

8.2.2. Relating 4.2.1, From Text to GeoJSON

Downloaded Excel files are turned to clean TFR lists by VBA (Code 2). After running VBA, Power Queries used to “concatenate” the per-day-lists to per-FIR-results (Code 3, Code 4). The Jupyter Notebook Fire_NOTAM_to_spatial.ipynb (Code 5) contains everything else to turn the TFR texts from Excel to a GeoJSON.

```
 Sub Clean_TFR_List_RunOnAllFilesInFolder()


    'Prerequisite: Set a reference to Microsoft Scripting Runtime by using
    'Tools > References in the Visual Basic Editor (Alt+F11)

    Dim strFolderName As String, eApp As Excel.Application, strFileName As String
    Dim wb As Workbook, ws As Worksheet, currWs As Worksheet, currWb As Workbook

    Dim objFileDialog As Object: Set objFileDialog =
Application.FileDialog(msoFileDialogFolderPicker)

    'variables for FileSystemObject (FSO) loop
    Dim objFSO As FileSystemObject
    Dim objFolder As Folder
    Dim objFile As File

    'start from this macro containing Workbook
    Set currWb = ActiveWorkbook: Set currWs = ActiveSheet

    'Select dialog: Folder in which all files are stored; start in path of this
macro containing Workbook
    objFileDialog.Title = "Select a folder"
    objFileDialog.InitialFileName = currWb.Path
    If objFileDialog.Show = -1 Then
      strFolderName = objFileDialog.SelectedItems(1)
    End If


    'Create an instance of the FSO
    Set objFSO = CreateObject("Scripting.FileSystemObject")

    'Get the selected folder from dialog to obj
    Set objFolder = objFSO.GetFolder(strFolderName)
```

```
    'If selected folder does not contain files, exit the sub
    If objFolder.Files.Count = 0 Then
        MsgBox "No files were found...", vbExclamation
        Exit Sub
    End If


    'Create a separate Excel process that is invisibile
    Set eApp = New Excel.Application:  eApp.Visible = False

    'No "Do While" with Dir possible as Dir content is subject to change,
    'so NO strFileName = Dir(strFolderName & "\*.xls")
    'and NO Do While strFileName <> ""
    'BUT: Loop through each file in the folder using For / FSO
    For Each objFile In objFolder.Files
           'Update status bar to indicate progress
           Application.StatusBar = "Processing " & strFolderName

           'Open new Workbook to contain cleaned data
           Set wb = eApp.Workbooks.Open(strFolderName + "\" + objFile.Name)

           ' Core task to clean TFR list
           ' Compose a filename from location and date, clear top 4 lines, save
as... and log what has been saved
           ' From this Excel Workbook: Path to logfile and data storage:
/K***/***_YYYY_MM_DD.xls

           ' Keyboard Shortcut: Can be defined personally via GUI

           'Create variables from target (= to be cleaned) Workbook top line
           'That range contains 3 letter location and date of TFRs being
valid/requested
            Dim strFilenameLoc As String
            strFilenameLoc = wb.ActiveSheet.Range("A1").Value

            Dim strFilenameDat As String
            strFilenameDat = wb.ActiveSheet.Range("A1").Value

            'Set the later path to data storage
            Dim strPath As String

            'Fetch datetime for logfile
            Dim strLogtime As String
            strLogtime = Now

            'Get 3 letter location behind '
            strFilenameLoc = Right(Left(strFilenameLoc, (InStr(strFilenameLoc, "'")
+ 3)), 3)

            'Start search for date (of TFRs being valid/requested) behind the
location
            strFilenameDat = Replace(strFilenameDat, "'" + strFilenameLoc + "'",
"")
            strFilenameDat = Right(Left(strFilenameDat, (InStr(strFilenameDat, "'")
+ 10)), 10)

            'Create path when filenames are properly set
            strPath = ThisWorkbook.Path + "\K" + strFilenameLoc

            'Write location and datetime to a log
            Call Module1.Txt_Append(ThisWorkbook.Path + "\VBA_Log.txt", strLogtime
+ " : " + strFilenameLoc + ", " + strFilenameDat)

            'clean row 1-4, select top left cell of target wb
            wb.ActiveSheet.Rows("1:4").Delete
            wb.ActiveSheet.Range("A1").Select
```

```
            'Make directory K*** if none exists for the current file
            If Len(Dir(strPath, vbDirectory)) = 0 Then
               MkDir strPath
               MsgBox "Directory Created Successfully : " & vbCrLf & strPath,
vbInformation, "VBA MkDir Function"
            End If

            'Save target wb
            wb.SaveAs fileName:=strPath + "\" + strFilenameLoc + "_" +
strFilenameDat + ".xls" _
            , FileFormat:=xlExcel8, Password:="", WriteResPassword:="", _
            ReadOnlyRecommended:=False, CreateBackup:=False

            'Close opened worbook w/o saving
            wb.Close SaveChanges:=False
            Debug.Print "Processed "; strFolderName + "\" + objFile.Name

    'End of loop
    Next objFile

    'Quit invisible Excel process
    eApp.Quit
    Set eApp = Nothing

    'Clear statusbar and inform of macro completion
    Application.StatusBar = ""
    MsgBox "Completed executing macro on all workbooks"

End Sub
'-------------------------------------------------------------------------------
-----
' Procedure : Txt_Append
' Author    : Daniel Pineault, CARDA Consultants Inc.
' Website   : http://www.cardaconsultants.com
' Purpose   : Output Data to an external file (*.txt or other format)
'               If the file does not exist already it will be created automatically
'               ***Do not forget about access' DoCmd.OutputTo Method for
'               exporting objects (queries, report,...)***
' Copyright : The following is release as Attribution-ShareAlike 4.0 International
'             (CC BY-SA 4.0) - https://creativecommons.org/licenses/by-sa/4.0/
'
' Input Variables:
' ~~~~~~~~~~~~~~~~
' sFile     : Name of the file that the text is to be output to including the full
path
' sText     : Text to be output to the file
'
' Usage:
' ~~~~~~
' Call Txt_Append("C:\temp\text.txt", "This is a new appended line of text.")
'
' Revision History:
' Rev       Date(yyyy/mm/dd)        Description
'
*******************************************************************************
***
' 1         2011-06-16              Initial Public Release
' 2         2018-02-24              Updated Copyright
'                                   Updated error handler
'-------------------------------------------------------------------------------
-----
Function Txt_Append(sFile As String, sText As String)
    On Error GoTo Err_Handler
    Dim iFileNumber           As Integer

    iFileNumber = FreeFile                  ' Get unused file number
    Open sFile For Append As #iFileNumber   ' Connect to the file
```

```
    Print #iFileNumber, sText                ' Append our string
    Close #iFileNumber                       ' Close the file

Exit_Err_Handler:
    Exit Function

Err_Handler:
    MsgBox "The following error has occurred" & vbCrLf & vbCrLf & _
           "Error Number: " & Err.Number & vbCrLf & _
           "Error Source: Txt_Append" & vbCrLf & _
           "Error Description: " & Err.Description & _
           Switch(Erl = 0, "", Erl <> 0, vbCrLf & "Line No: " & Erl) _
           , vbOKOnly + vbCritical, "An Error has Occurred!"
    GoTo Exit_Err_Handler
End Function
```

Code 2 VBA to acquire clean TFR lists from downloaded Excel files

```
let
    Source = Folder.Files("D:\UNIGIS\MASTER\CreatedData\TFR\KZLC"),
    #"Filtered Hidden Files1" = Table.SelectRows(Source, each
[Attributes]?[Hidden]? <> true),
    #"Invoke Custom Function1" = Table.AddColumn(#"Filtered Hidden Files1",
"Transform File", each #"Transform File"([Content])),
    #"Renamed Columns1" = Table.RenameColumns(#"Invoke Custom Function1", {"Name",
"Source.Name"}),
    #"Removed Other Columns1" = Table.SelectColumns(#"Renamed Columns1",
{"Source.Name", "Transform File"}),
    #"Expanded Table Column1" = Table.ExpandTableColumn(#"Removed Other Columns1",
"Transform File", Table.ColumnNames(#"Transform File"(#"Sample File"))),
    #"Changed Type" = Table.TransformColumnTypes(#"Expanded Table
Column1",{{"Source.Name", type text}, {"Location", type text}, {"NOTAM #", type
text}, {"Class", type text}, {"Issue Date (UTC)", type text}, {"Effective Date
(UTC)", type text}, {"Cancel Date (UTC)", type text}, {"Expiration Date (UTC)",
type text}, {"NOTAM Condition or LTA Subject", type text}}),
    #"Removed Duplicates" = Table.Distinct(#"Changed Type", {"NOTAM #"}),
    #"Replaced Value" = Table.ReplaceValue(#"Removed
Duplicates",".xls","",Replacer.ReplaceText,{"Source.Name"}),
    #"Renamed Columns" = Table.RenameColumns(#"Replaced Value",{{"Source.Name",
"Source"}})
in
    #"Renamed Columns"
```

Code 3 Power Queries code to be run when columns are recognized correctly. The Source path will need to get changed if reproduced!

```
let
    Source = Folder.Files("D:\UNIGIS\MASTER\CreatedData\TFR\KZLC"),
    #"Filtered Hidden Files1" = Table.SelectRows(Source, each
[Attributes]?[Hidden]? <> true),
    #"Invoke Custom Function1" = Table.AddColumn(#"Filtered Hidden Files1",
"Transform File", each #"Transform File"([Content])),
    #"Renamed Columns1" = Table.RenameColumns(#"Invoke Custom Function1", {"Name",
"Source.Name"}),
    #"Removed Other Columns1" = Table.SelectColumns(#"Renamed Columns1",
{"Source.Name", "Transform File"}),
    #"Expanded Table Column1" = Table.ExpandTableColumn(#"Removed Other Columns1",
"Transform File", Table.ColumnNames(#"Transform File"(#"Sample File"))),
    #"Changed Type" = Table.TransformColumnTypes(#"Expanded Table
Column1",{{"Source.Name", type text}, {"Column1", type text}, {"Column2", type
text}, {"Column3", type text}, {"Column4", type text}, {"Column5", type text},
{"Column6", type text}, {"Column7", type text}, {"Column8", type text}}),
    #"Removed Duplicates" = Table.Distinct(#"Changed Type", {"Column2"}),
    #"Promoted Headers" = Table.PromoteHeaders(#"Removed Duplicates",
[PromoteAllScalars=true]),
    #"Changed Type1" = Table.TransformColumnTypes(#"Promoted Headers",{{"ZLC_2021-
08-01.xls", type text}, {"Location", type text}, {"NOTAM #", type text}, {"Class",
type text}, {"Issue Date (UTC)", type text}, {"Effective Date (UTC)", type text},
{"Cancel Date (UTC)", type text}, {"Expiration Date (UTC)", type text}, {"NOTAM
Condition or LTA Subject", type text}}),
    #"Renamed Columns" = Table.RenameColumns(#"Changed Type1",{{"ZLC_2021-08-
01.xls", "Source"}}),
    #"Replaced Value" = Table.ReplaceValue(#"Renamed
Columns",".xls","",Replacer.ReplaceText,{"Source"})
in
    #"Replaced Value"
```

Code 4 Power Queries code to be run when columns are not recognized correctly. The Source path AND "ZLC" location indicator occurrences will need to get changed if reproduced!

A Python 3 Script to extract coordinates and radius (if present) from a TFR list and turn them into GeoJSON

```
import pandas as pd
import re
import geopandas as gpd
from shapely.geometry import Point, Polygon, LineString
```

ZDV : AS4 NM RADIUS, ZHU FIREFIGHTING <-are no wild fire related TFR but NTL DEFENCE AIRSPACE, ZKC has no FIRE FIGHTING at all

```
##specify file location and name
#3 letter location indicator, reused within all created files
#possible values: ZAB ZDV ZFW ZHU ZKC ZLA ZLC ZMP ZOA ZSE
tfr =r"ZSE"
#path and filename
path=r"D:\UNIGIS\MASTER\CreatedData\TFR\\"
tfr_list = tfr + r"_2021_All_Aug-Oct_revisited.xlsx"

#read the Excel Workbook
df = pd.read_excel(path + tfr_list)

df
```

```
             Source Location NOTAM #     Class Issue Date (UTC)  \
0    ZSE_2021-08-01      ZSE  1/3680  Airspace  07/22/2021 1205
1    ZSE_2021-08-01      ZSE  1/6739  Airspace  07/10/2021 0110
2    ZSE_2021-08-01      ZSE  1/3581  Airspace  07/22/2021 0057
3    ZSE_2021-08-01      ZSE  1/7262  Airspace  07/12/2021 0114
4    ZSE_2021-08-01      ZSE  1/2358  Airspace  07/20/2021 0513
..              ...      ...     ...       ...              ...
253  ZSE_2021-10-04      ZSE  1/7554  Airspace  10/04/2021 1448
254  ZSE_2021-10-07      ZSE  1/1515  Airspace  10/07/2021 2152
255  ZSE_2021-10-17      ZSE  1/4201  Airspace  10/17/2021 1216
256  ZSE_2021-10-17      ZSE  1/4257  Airspace  10/17/2021 1527
257  ZSE_2021-10-31      ZSE  1/0057  Airspace  10/31/2021 1603

    Effective Date (UTC) Cancel Date (UTC) Expiration Date (UTC)  \
0        07/22/2021 1400   08/19/2021 0523       09/22/2021 0500
1        07/10/2021 1400   08/17/2021 0450       09/10/2021 0500
2        07/22/2021 1500   08/16/2021 0147       09/22/2021 0330
3        07/12/2021 1300   08/15/2021 1012       09/12/2021 0500
4        07/20/2021 1400   08/12/2021 0510       09/20/2021 0500
..                   ...              ...                   ...
253      10/04/2021 1500   10/15/2021 1441       10/18/2021 0500
254      10/07/2021 2230   10/08/2021 2059       10/14/2021 1423
255      10/17/2021 1500   10/20/2021 0202       11/17/2021 0300
256      10/17/2021 1530   10/17/2021 2107       10/18/2021 0500
257      10/31/2021 1600   10/31/2021 2210       11/01/2021 0030

                        NOTAM Condition or LTA Subject
0    !FDC 1/3680 ZSE MN..AIRSPACE 7NM N THOMPSON FA...
1    !FDC 1/6739 ZSE MT..AIRSPACE 9NM S OF TROY, MT...
2    !FDC 1/3581 ZSE ID..AIRSPACE 20NM SE OF LEWIST...
3    !FDC 1/7262 ZSE ID..AIRSPACE 5NM NE OF KELLOGG...
```

```
4    !FDC 1/2358 ZSE MT..AIRSPACE 5NM N OF TROY, MT...
..                                                 ...
253  !FDC 1/7554 ZSE OR..AIRSPACE 24NM E OF ROSEBUR...
254  !FDC 1/1515 ZSE OR..AIRSPACE 27NM W OF SUNRIVE...
255  !FDC 1/4201 ZSE CA..AIRSPACE 35NM EAST OF CRES...
256  !FDC 1/4257 ZSE OR..AIRSPACE AURORA, OR..TEMPO...
257  !FDC 1/0057 ZSE OR..AIRSPACE CENTERVILLE, WA.....

[258 rows x 9 columns]
```

```
def get_coordinates(row):
#     # find all substrings with 6 digits before N and 7 digits plus W = 8
    sub_coords = "\w{6}N\w{8}"
    coordinates = re.findall(sub_coords,row["NOTAM Condition or LTA Subjec
t"])
    coords_list = []
    for coord in coordinates:
        #replaces old coordinates = row["NOTAM Condition or LTA Subject"].
findall(sub_coords)
        #north
        deg_lat = coord[:2]
        min_lat = coord[2:4]
        sec_lat = coord[4:6]
        dd_lat = float(deg_lat) + float(min_lat)/60 + float(sec_lat)/(60*6
0)

        #west
        deg_lon = coord[7:10]
        min_lon = coord[10:12]
        sec_lon = coord[12:14]
        dd_lon = -1*(float(deg_lon) + float(min_lon)/60 + float(sec_lon)/(
60*60))

        coords_list.append((dd_lon,dd_lat))

    return coords_list

df["Coordinates"]=df.apply(get_coordinates,axis=1)

df["Coordinates"]
```

```
0             [(-115.26666666666667, 47.68333333333333)]
1                                  [(-115.9675, 48.34)]
2      [(-116.82194444444444, 46.18055555555555), (-1...
3      [(-116.21361111111112, 47.68722222222222), (-1...
4      [(-116.03888888888889, 48.638888888888886), (-...
                             ...
253    [(-122.58944444444444, 43.63333333333333), (-1...
254          [(-121.75416666666666, 43.916666666666664)]
255    [(-123.38333333333334, 41.78333333333333), (-1...
256                    [(-120.65, 43.800555555555555)]
257                                [(-120.85, 45.7625)]
Name: Coordinates, Length: 258, dtype: object
```

```
def get_geometry(row):
    #transform float coordinates from get_coordinates(row)
    coords = row["Coordinates"]
    #find points
    if len(coords)==1:
        geom = Point(coords[0][0],coords[0][1])
    elif len(coords)==2:
        geom = LineString(coords)
    elif len(coords)>2:
        geom = Polygon(coords)
    else:
        geom = None

    return geom
df["geometry"] = df.apply(get_geometry,axis=1)
```

```
D:\anaconda3\envs\master_env\lib\site-packages\pandas\core\dtypes\cast.py:
122: ShapelyDeprecationWarning: The array interface is deprecated and will
no longer work in Shapely 2.0. Convert the '.coords' to a numpy array inst
ead.
  arr = construct_1d_object_array_from_listlike(values)
```

```
df["geometry"]
```

```
0          POINT (-115.26666666666667 47.68333333333333)
1                                POINT (-115.9675 48.34)
2      POLYGON ((-116.82194444444444 46.1805555555555...
3      POLYGON ((-116.21361111111112 47.6872222222222...
4      POLYGON ((-116.03888888888889 48.6388888888888...
                              ...
253    POLYGON ((-122.58944444444444 43.6333333333333...
254       POINT (-121.75416666666666 43.916666666666664)
255    POLYGON ((-123.38333333333334 41.7833333333333...
256                  POINT (-120.65 43.800555555555555)
257                              POINT (-120.85 45.7625)
Name: geometry, Length: 258, dtype: object
```

```
def get_radius(row):
    #find radius
    sub_pt_radius = "\w{1,}\\.\w{1,}NM RADIUS" ## if fraction like 1.5NM R
ADIUS is given
    sub_radius = "\w{1,}NM\sRADIUS"
    sub_wr_radius = "\w{1,}NM\nRADIUS"
    sub_spNM_radius = "\w{1,} NM RADIUS"

    #check for decimal fraction, needs to be done first, otherwise 1.5NM w
ould result in 5NM
    radius = re.findall(sub_pt_radius,row["NOTAM Condition or LTA Subject"
])

    #when there is no decimal fraction, look for most common case
    if len(radius)==0:
        radius = re.findall(sub_radius,row["NOTAM Condition or LTA Subject
"])
    else:
```

```
        radius = radius

    #when there is a line wrap btn NM and radius, its len is still 0 inste
ad of 1:
    if len(radius)==0:
        radius = re.findall(sub_wr_radius,row["NOTAM Condition or LTA Subj
ect"])
    else:
        radius = radius

    #when there is a space btn number and NM, its len is still 0 instead o
f 1:
    if len(radius)==0:
        radius = re.findall(sub_spNM_radius,row["NOTAM Condition or LTA Su
bject"])
    else:
        radius = radius

    #if len(radius)!=0:
        #if str(radius[0]).isdigit()==False:
        #if not all([str(i).isdigit() for i in radius]):
        #if not all(chr.isdigit() for chr in radius[0]):
           # radius = "NoNumber"
        #else:
           #radius = radius

    #so far it is 5NM RADIUS or 12NM RADIUS or 2 NM RADIUS or 1.5NM RADIUS
, so cut down to the numbers
    for chars in radius:
        #one digit
        if len(chars)==10:
            radius = chars[0]
        #two digits
        elif len(chars)==11:
            radius = chars[:2]
        #fraction
        elif len(chars)==12:
            radius = chars[:3]
        #fraction and tens (not known if any)
        elif len(chars)==13:
            radius = chars[:4]
        #everything else is erroneous:
        else:
            radius = "NotParsable"

#         if radius.isdigit()==False:
#             radius = radius+"isNoNumber"
#         else:
#             radius = radius

    return radius
df["Radius"]=df.apply(get_radius,axis=1)
```

```
#look at ALL rows
pd.set_option('display.max_rows', None)

print(df["Radius"])

0       6
1       5
2      []
3      []
4      []
[…]
253    []
254     5
255    []
256     4
257     3
Name: Radius, dtype: object

#with Radius being a list that causes issues during JSON export,it needs t
o get changed:
df["Radius"] =df["Radius"].astype('string')

#single numbers are needed as radius instead of list residuals
def convert_radius(row):
    radius = row["Radius"]
    if radius == "[]":
        radius = 0
    return radius
df["Radius"]=df.apply(convert_radius,axis=1)

#string has still [] as values which cannot be converted to number format,
so

df["Radius"] =df["Radius"].astype('float')

#turn radius from NM to m for buffer
def radius_to_m(row):
    radius_m = row["Radius"]*1852
    return radius_m
df["Radius_m"]=df.apply(radius_to_m,axis=1)

print(df["Radius_m"])

0      11112.0
1       9260.0
2          0.0
3          0.0
4          0.0
[…]

253        0.0
254     9260.0
255        0.0
256     7408.0
257     5556.0
Name: Radius_m, dtype: float64
```

According to National Wildfire Coordination Group (2018, p. 106), Keyphrase for aerial firefighting is TO PROVIDE A SAFE ENVIRONMENT FOR WILDLAND FIRE FIGHTING AVIATION OPERATIONS. PURSUANT TO 14 CFR SECTION 91.137(A)(2) TEMPORARY FLIGHT RESTRICTIONS ARE IN EFFECT. So it is decided to serch for variations of the term "FIRE FIGHTING" to identify relevant TFRs. Edit: Using "FIREFIGHTING" as a single word was an erroneus assumption, as this keyword is used in other types of TFR to allow firefighting aircraft.

```
#get reason of the TFR / whether it was firefighting
def get_reason(row):
#     # find all substrings with FIRE FIGHTING
    keyword = "FIRE FIGHTING"
    wr_keyword = "FIRE\nFIGHTING"
#     single_keyword = "FIREFIGHTING"

    #find most common case
    reason = re.findall(keyword,row["NOTAM Condition or LTA Subject"])

    #find with linewrap
    if len(reason)==0:
        reason = re.findall(wr_keyword,row["NOTAM Condition or LTA Subject
"])
    else:
        reason = reason

    #find keyword written a one word
#     if len(reason)==0:
#         reason = re.findall(single_keyword,row["NOTAM Condition or LTA S
ubject"])
#     else:
#         reason = reason

    return reason


df["Reason"]=df.apply(get_reason,axis=1)

df["Reason"]
```

```
0       [FIRE FIGHTING]
1      [FIRE\nFIGHTING]
2       [FIRE FIGHTING]
3      [FIRE\nFIGHTING]
4       [FIRE FIGHTING]
[]…
253     [FIRE FIGHTING]
254     [FIRE FIGHTING]
255     [FIRE FIGHTING]
256                  []
257                  []
Name: Reason, dtype: object
```

For some of the NOTAMs, the journey ends with the following step. Those where firefighting is not the reason or where no geometry could be parsed are rejected. They are stored to Excel Workbooks to enable for manual review whether the above parsing was sufficient.

```
# prepare dataframe for manual sanity checks in Excel
#"Reason" contains still lists so .str.len() checks for content/emptiness
df_no_fire = df[(df["Reason"].str.len() == 0)]
df_no_fire.to_excel(tfr+"_no_fire.xlsx")
```

The reason to purge the non-wildfire TFRs this late is, that it might be of interest to relate them to hotspot clusters as well (in a further research).

```
# dataframe to proceed with: "Reason" shall not be empty
df = df[(df["Reason"].str.len() != 0)]
```

Then it is time to create a geodataframe from the dataframe containing only fire fighting related TFRs.

```
gdf = gpd.GeoDataFrame(df, crs="EPSG:4326", geometry=df["geometry"])

# prepare geodataframe for manual sanity checks in Excel
gdf_no_geom = gdf[gdf["geometry"]==None]
gdf_no_geom.to_excel(tfr+"_no_geom.xlsx")

# geodataframe to proceed with: Only with geometry
gdf = gdf[gdf["geometry"]!=None]

gdf
```

```
             Source Location NOTAM #     Class Issue Date (UTC)  \
0    ZSE_2021-08-01      ZSE  1/3680  Airspace  07/22/2021 1205
1    ZSE_2021-08-01      ZSE  1/6739  Airspace  07/10/2021 0110
2    ZSE_2021-08-01      ZSE  1/3581  Airspace  07/22/2021 0057
3    ZSE_2021-08-01      ZSE  1/7262  Airspace  07/12/2021 0114
4    ZSE_2021-08-01      ZSE  1/2358  Airspace  07/20/2021 0513
[…]
251  ZSE_2021-09-30      ZSE  1/5525  Airspace  09/30/2021 0032
252  ZSE_2021-10-03      ZSE  1/7181  Airspace  10/03/2021 0416
253  ZSE_2021-10-04      ZSE  1/7554  Airspace  10/04/2021 1448
254  ZSE_2021-10-07      ZSE  1/1515  Airspace  10/07/2021 2152
255  ZSE_2021-10-17      ZSE  1/4201  Airspace  10/17/2021 1216

    Effective Date (UTC) Cancel Date (UTC) Expiration Date (UTC)  \
0        07/22/2021 1400   08/19/2021 0523       09/22/2021 0500
1        07/10/2021 1400   08/17/2021 0450       09/10/2021 0500
2        07/22/2021 1500   08/16/2021 0147       09/22/2021 0330
3        07/12/2021 1300   08/15/2021 1012       09/12/2021 0500
4        07/20/2021 1400   08/12/2021 0510       09/20/2021 0500
[…]
251      09/30/2021 1500   09/30/2021 0050       10/10/2021 0300
252      10/03/2021 1400   10/04/2021 1459       10/17/2021 0500
253      10/04/2021 1500   10/15/2021 1441       10/18/2021 0500
254      10/07/2021 2230   10/08/2021 2059       10/14/2021 1423
255      10/17/2021 1500   10/20/2021 0202       11/17/2021 0300

                        NOTAM Condition or LTA Subject  \
0    !FDC 1/3680 ZSE MN..AIRSPACE 7NM N THOMPSON FA...
1    !FDC 1/6739 ZSE MT..AIRSPACE 9NM S OF TROY, MT...
2    !FDC 1/3581 ZSE ID..AIRSPACE 20NM SE OF LEWIST...
```

```
3    !FDC 1/7262 ZSE ID..AIRSPACE 5NM NE OF KELLOGG...
4    !FDC 1/2358 ZSE MT..AIRSPACE 5NM N OF TROY, MT...
[…]
251  !FDC 1/5525 ZSE OR..AIRSPACE 12NM NE OF DETROI...
252  !FDC 1/7181 ZSE OR..AIRSPACE 24NM E OF ROSEBUR...
253  !FDC 1/7554 ZSE OR..AIRSPACE 24NM E OF ROSEBUR...
254  !FDC 1/1515 ZSE OR..AIRSPACE 27NM W OF SUNRIVE...
255  !FDC 1/4201 ZSE CA..AIRSPACE 35NM EAST OF CRES...

                                          Coordinates  \
0           [(-115.26666666666667, 47.68333333333333)]
1                               [(-115.9675, 48.34)]
2    [(-116.82194444444444, 46.18055555555555), (-1...
3    [(-116.21361111111112, 47.68722222222222), (-1...
4    [(-116.03888888888889, 48.638888888888886), (-...
[…]
251                           [(-122.0375, 44.875)]
252  [(-122.60833333333333, 43.63333333333333), (-1...
253  [(-122.58944444444444, 43.63333333333333), (-1...
254       [(-121.75416666666666, 43.916666666666664)]
255  [(-123.38333333333334, 41.78333333333333), (-1...

                                             geometry  Radius  Radius_m
\
0                          POINT (-115.26667 47.68333)     6.0   11112.0
1                          POINT (-115.96750 48.34000)     5.0    9260.0
2    POLYGON ((-116.82194 46.18056, -116.73333 46.1...     0.0       0.0
3    POLYGON ((-116.21361 47.68722, -115.82139 47.7...     0.0       0.0
4    POLYGON ((-116.03889 48.63889, -115.79444 48.6...     0.0       0.0
[…]
251                        POINT (-122.03750 44.87500)     6.0   11112.0
252  POLYGON ((-122.60833 43.63333, -122.46667 43.6...     0.0       0.0
253  POLYGON ((-122.58944 43.63333, -122.46667 43.6...     0.0       0.0
254                        POINT (-121.75417 43.91667)     5.0    9260.0
255  POLYGON ((-123.38333 41.78333, -123.10000 41.7...     0.0       0.0

               Reason
0     [FIRE FIGHTING]
1    [FIRE\nFIGHTING]
2     [FIRE FIGHTING]
3    [FIRE\nFIGHTING]
4     [FIRE FIGHTING]
[…]
251   [FIRE FIGHTING]
252   [FIRE FIGHTING]
253   [FIRE FIGHTING]
254   [FIRE FIGHTING]
255   [FIRE FIGHTING]
```

```
#add possibility to check in an Excel Workbook
gdf.to_excel(tfr+"_excel_check_fire_TFRs.xlsx")
```

List columns "Coordinates" and "Reason" are no longer needed (and would only disturb GeoJSON creation) and become omitted.

```
#geodataframe shall use these columns
gdf= gdf[["Source","Location", "NOTAM #", "Issue Date (UTC)", "Effective D
ate (UTC)", "Cancel Date (UTC)", "Expiration Date (UTC)", "NOTAM Condition
or LTA Subject","Radius","Radius_m","geometry"]]

#prepare buffer
gdf_buffered = gdf.copy()
gdf_buffered = gdf_buffered.to_crs("EPSG:2163")

#do buffer by radius: SHOULD be 0 for polygons, do it for all like this is
FAST
gdf_buffered["geometry"] = gdf_buffered.buffer(gdf["Radius_m"], resolution
=16)

#create geojson of gdf, points and polygons
#gdf.to_file(filename='gdf_first.geojson', driver='GeoJSON')

#create geojson of gdf_buffered, just polygons
#back to WGS84 for geojson creation
gdf_buffered = gdf_buffered.to_crs("EPSG:4326")
gdf_buffered.to_file(filename= tfr+'_fire_TFRs.geojson', driver='GeoJSON')

D:\anaconda3\envs\master_env\lib\site-packages\geopandas\io\file.py:362: F
utureWarning: pandas.Int64Index is deprecated and will be removed from pan
das in a future version. Use pandas.Index with the appropriate dtype inste
ad.
  pd.Int64Index,

#shape file of course only works with uniform geometries
#gdf_buffered.to_file('tfr+'fire_TFRs.shp', driver='ESRI Shapefile')

#plot result into new window
%matplotlib qt
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
world = world.to_crs("EPSG:4326")
ax = world[world.continent == 'North America'].plot(
    color='white', edgecolor='black')
gdf_buffered.plot(ax=ax, color='red')

<AxesSubplot:>
```

Code 5 Jupyter Notebook "Fire_NOTAM_to_spatial.ipynb"

8.2.3. Relating 4.2.3, Exploring the Datasets

Adding TFRs from GeoJSON is achieved by Add_TFRs_from_geojson.ipynb shown by Code 6.

For 4.2.3.2 the state vectors are converted to GeoJSON format by Aircraft_States_to_GeoJSON.ipynb (Code 7). Moving Pandas trajectories are built by Aircraft_ovr_fires_to_trajectories.ipynb ().

A script to be run in ArcGIS Pro to import TFRs via JSON To Features

List with FIR location indicators is set up. Possible values in this research: ZAB ZDV ZFW (ZHU, ZKC turn out having no fire TFRs) ZLA ZLC ZMP ZOA ZSE

```
list = ["ZAB", "ZDV", "ZFW", "ZLA", "ZLC", "ZMP", "ZOA", "ZSE"]
```

With a function, parameters are set and JSON To Features is called to add a TFR. jsonPath and gdbPath need to become adjusted to reproduce.

```
def add_TFRs(indicator):
    jsonPath = r"D:\UNIGIS\MASTER\Scripts\\"
    gdbPath = r"C:\Users\someNAME\Documents\ArcGIS\Projects\somePROJECT\someGDB.gdb\\"

    jsonFile = jsonPath + indicator + "_fire_TFRs.geojson"
    featureClass = gdbPath + indicator + "_TFRs"

    arcpy.JSONToFeatures_conversion(jsonFile, featureClass)
```

With a loop the function to add TFRs is called

```
#loop to iterate through list
for i in range(0, len(list)):
    add_TFRs(list[i])
```

Code 6 Add_TFRs_from_geojson.ipynb

Script to turn aircraft state vector .csv to .geojson

```
import pandas as pd
import geopandas as gpd
import shapely.wkt

#read a dataframe from current directory
df = pd.read_csv('aircraft_states_north_america_20210801_20211101.csv')

#turn dataframe into geodataframe reading coordinates from df
geometry = df['coordinates'].map(shapely.wkt.loads)
df = df.drop('coordinates', axis=1)
gdf = gpd.GeoDataFrame(df, crs="EPSG:4326", geometry=geometry)

#write into .geojson into current directory
gdf.to_file('aircraft_states_north_america_20210801_20211101.geojson', driver='GeoJSON')
```

Code 7 Aircraft_States_to_GeoJSON.ipynb

Turn points from aircraft over fires to trajectories

```
import geopandas as gpd
import movingpandas as mpd
from datetime import timedelta
```

```
INFO: Missing optional dependencies. To use the trajectory smoother classes please
install Stone Soup (see https://stonesoup.readthedocs.io/en/latest/#installation).
```

```
#read the point file
gdf = gpd.read_file('aircraft_over_fires_fmArcGIS.geojson')
```

```
#check gdf which timestamp column can be set as t=
gdf['timestamp_Converted']
```

```
0        1627779216000
1        1627779283000
2        1627779287000
3        1627779300000
4        1627779316000
            ...
75814    1635522884000
75815    1635462337000
75816    1635462359000
75817    1635462368000
75818    1635462388000
Name: timestamp_Converted, Length: 75819, dtype: int64
```

```
#create MovingPandas trajectory collection
#to record movement of single aircraft, 'aircraft_id' is used here
#as t timestamp do work converted timestamp from ArcGIS,
'timestamp_Converted'  as well as the original one 'timestamp',
#where the original one 'timestamp' maintains the time in UTC
collection = mpd.TrajectoryCollection(gdf, 'aircraft_id', t='timestamp')
```

```
#create a gdf containing trajectories from collection
gdf_traj = collection.to_traj_gdf(wkt=False)
```

```
gdf_traj
```

	traj_id	start_t	end_t	geometry	length	direction
0	a02862	2021-08-01 01:19:23	2021-10-16 16:05:49	LINESTRING (-121.10220 39.91300, -121.10920 39...	1.002747e+07	150.657437
1	a0956b	2021-08-06 18:18:07	2021-10-13 23:29:58	LINESTRING (-121.51430 44.58320, -121.13210 46...	6.297902e+06	173.665738
2	a09922	2021-08-14 21:32:03	2021-08-31 01:49:44	LINESTRING (-111.73000 40.73220, -111.71550 40...	6.284843e+05	200.934580
3	a0b782	2021-08-03 17:08:34	2021-09-14 17:31:29	LINESTRING (-121.89210 44.36120, -121.89560 44...	1.261385e+05	73.214255
4	a0bef0	2021-08-10 22:05:39	2021-09-06 02:34:34	LINESTRING (-120.95530 46.91770, -120.97130 46...	2.259979e+06	184.235095

	traj_id	start_t	end_t	geometry	length	direction
5	a0c65e	2021-08-30 03:53:37	2021-09-13 06:48:29	LINESTRING (-120.12110 38.58380, -120.13630 38...	9.985536e+06	257.735492
6	a0f3ed	2021-08-25 22:07:43	2021-08-25 22:07:53	LINESTRING (-123.11360 45.27530, -123.10360 45...	7.846834e+02	89.996448
7	a0f927	2021-08-03 23:15:13	2021-09-18 01:12:07	LINESTRING (-121.83250 44.33680, -121.87410 44...	3.266750e+06	341.170917
8	a0fb5b	2021-09-10 21:41:15	2021-09-26 18:43:45	LINESTRING (-120.61540 38.56450, -120.62700 38...	1.499370e+05	315.431379
9	a0ff12	2021-08-02 22:10:48	2021-08-05 01:58:24	LINESTRING (-121.87880 44.34920, -121.88940 44...	4.026835e+05	42.676306
10	a1a588	2021-10-13 17:07:01	2021-10-17 17:25:18	LINESTRING (-120.18760 34.47170, -120.16920 34...	6.171132e+04	342.408073
11	a282ed	2021-08-20 21:49:44	2021-10-17 12:36:42	LINESTRING (-120.69850 38.58720, -120.67020 38...	3.821931e+06	171.810827
12	a2fd4d	2021-08-02 15:18:56	2021-09-13 01:23:39	LINESTRING (-121.32770 40.33380, -121.32940 40...	7.577057e+06	161.396908
13	a30104	2021-10-11 17:12:21	2021-10-16 00:46:15	LINESTRING (-121.36230 40.30180, -121.39370 40...	5.633138e+05	184.120264
14	a3bf4f	2021-08-11 21:49:34	2021-08-11 21:52:03	LINESTRING (-122.39900 47.12280, -122.39360 47...	6.104609e+03	107.110417
15	a3ca74	2021-09-12 00:23:02	2021-09-12 02:20:17	LINESTRING (-122.93570 45.88810, -122.96300 45...	4.913755e+04	246.660409
16	a40442	2021-08-03 23:08:59	2021-10-14 04:41:04	LINESTRING (-117.76140 34.20350, -117.75660 34...	7.814506e+05	241.064461
17	a42299	2021-08-04 01:41:34	2021-10-17 21:30:02	LINESTRING (-121.40270 39.92620, -121.40230 39...	4.731289e+06	159.276370
18	a492b8	2021-10-18 15:52:45	2021-10-18 18:41:19	LINESTRING (-114.34420 48.16330, -114.35030 48...	2.791873e+05	120.651492
19	a4966f	2021-10-18 16:25:54	2021-10-18 18:41:32	LINESTRING (-114.32540 48.16800, -114.33270 48...	1.812986e+05	181.951886
20	a4acf2	2021-08-16 19:16:44	2021-09-12 23:24:11	LINESTRING (-118.60240 36.25330, -118.59250 36...	7.074507e+06	202.099986
21	a4b0a9	2021-08-22 19:22:58	2021-09-26 20:56:23	LINESTRING (-120.47310 38.52980, -120.49440 38...	1.272255e+07	159.648604
22	a4b460	2021-08-01 00:53:36	2021-09-09 17:17:40	LINESTRING (-121.21740 39.88280, -121.14460 39...	2.211258e+06	343.485205
23	a4b817	2021-08-04 21:53:57	2021-09-09 00:12:16	LINESTRING (-120.95280 39.08800, -121.02390 39...	4.858450e+06	130.911888
24	a4c6f3	2021-09-01 01:22:04	2021-10-02 22:33:58	LINESTRING (-121.45800 40.44020, -122.25720 39...	7.243697e+05	153.866718

	traj_id	start_t	end_t	geometry	length	direction
25	a4c98b	2021-08-03 23:09:39	2021-09-24 12:26:24	LINESTRING (-117.75340 34.20050, -117.75030 34...	9.631617e+04	240.691375
26	a4caaa	2021-08-04 21:30:56	2021-10-27 21:46:30	LINESTRING (-120.99420 39.08700, -120.99420 39...	5.393790e+06	86.293064
27	a4e34d	2021-08-18 20:37:59	2021-09-23 00:44:09	LINESTRING (-122.61040 38.91860, -122.60920 38...	4.620350e+06	169.572258
28	a4e704	2021-08-25 22:35:51	2021-10-16 18:53:43	LINESTRING (-120.47920 38.06080, -120.45560 38...	2.919233e+06	223.403402
29	a4ffa7	2021-08-03 21:48:51	2021-09-01 01:28:09	LINESTRING (-121.39860 39.96570, -121.41750 39...	5.360443e+05	20.000313
30	a5035e	2021-08-22 22:01:39	2021-10-16 01:36:45	LINESTRING (-120.46460 38.52900, -120.47240 38...	1.951424e+06	213.828551
31	a50acc	2021-08-22 22:41:32	2021-10-11 21:27:55	LINESTRING (-120.55970 38.52710, -120.56530 38...	2.169616e+06	265.636251
32	a50e83	2021-08-04 21:32:46	2021-09-05 23:52:51	LINESTRING (-120.99600 39.10270, -120.97310 39...	1.940371e+06	185.752648
33	a5123a	2021-08-04 01:25:08	2021-09-06 00:00:36	LINESTRING (-121.41000 39.97380, -121.41140 39...	8.480343e+05	164.418335
34	a515f1	2021-08-25 21:14:40	2021-08-28 23:36:35	LINESTRING (-117.40850 34.10340, -117.41120 34...	5.334256e+05	171.208530
35	a519a8	2021-10-28 23:05:37	2021-10-28 23:06:28	LINESTRING (-122.46510 38.58820, -122.49040 38...	5.372565e+03	259.862686
36	a51d5f	2021-08-25 21:47:54	2021-10-29 15:54:44	LINESTRING (-117.40350 34.11210, -117.41340 34...	4.265345e+06	323.899072
37	a5236f	2021-09-06 00:54:06	2021-10-17 21:42:18	LINESTRING (-120.23830 38.58700, -120.23270 38...	6.583527e+06	175.156592
38	a52726	2021-08-16 23:32:49	2021-10-01 01:34:43	LINESTRING (-118.55810 36.25140, -118.55450 36...	3.059477e+06	309.004251
39	a5324b	2021-08-04 01:30:29	2021-08-04 03:12:39	LINESTRING (-121.41080 39.99130, -121.45900 40...	2.343662e+04	349.953738
40	a53602	2021-08-03 21:43:10	2021-09-17 23:18:36	LINESTRING (-117.64470 34.18530, -117.65800 34...	1.651191e+06	299.393610
41	a539b9	2021-08-03 22:44:04	2021-10-01 01:34:04	LINESTRING (-117.73040 34.25670, -117.72580 34...	3.551684e+06	336.503492
42	a53d70	2021-08-03 22:20:14	2021-09-17 22:56:28	LINESTRING (-117.64110 34.20630, -117.66530 34...	2.243968e+06	300.129053
43	a54127	2021-08-24 01:58:48	2021-10-16 18:34:40	LINESTRING (-120.67330 38.62770, -120.66270 38...	8.860666e+05	207.672263
44	a5525c	2021-09-26 15:41:04	2021-09-29 17:13:09	LINESTRING (-121.79060 39.47960, -121.78390 39...	3.662780e+04	154.013136

	traj_id	start_t	end_t	geometry	length	direction
45	a559ca	2021-08-25 21:29:28	2021-10-16 01:37:14	LINESTRING (-117.40000 34.19150, -117.40690 34...	1.800788e+06	309.871990
46	a55d81	2021-08-26 21:03:48	2021-10-16 18:40:38	LINESTRING (-120.39400 37.95540, -120.40680 37...	1.395105e+06	228.687886
47	a568a6	2021-08-12 00:20:21	2021-08-31 01:24:05	LINESTRING (-121.28690 39.39760, -121.29770 39...	2.234014e+05	345.970158
48	a5726d	2021-08-04 03:14:56	2021-08-12 01:56:49	LINESTRING (-121.45790 40.19510, -121.29500 39...	1.159681e+05	177.721672
49	a5c16b	2021-08-03 01:36:38	2021-10-02 19:11:35	LINESTRING (-122.84970 40.13060, -122.83940 40...	1.375791e+07	132.384464
50	a606a2	2021-08-03 21:42:23	2021-08-29 20:54:48	LINESTRING (-117.69120 34.21860, -117.69810 34...	7.504047e+05	99.847375
51	a61069	2021-08-12 02:31:04	2021-08-12 02:58:12	LINESTRING (-121.29760 39.40590, -121.29630 39...	3.725013e+02	162.572323
52	a61420	2021-08-29 16:22:14	2021-08-29 19:19:23	LINESTRING (-117.31650 33.55340, -117.32090 33...	1.583213e+05	184.435860
53	a622fc	2021-10-14 19:12:15	2021-10-14 19:59:05	LINESTRING (-121.59480 37.87290, -121.59270 37...	4.435463e+04	326.960015
54	a7d27a	2021-08-02 19:37:15	2021-10-14 18:27:07	LINESTRING (-122.73350 40.36620, -122.71820 40...	9.791285e+06	160.587649
55	a8401a	2021-08-11 02:08:55	2021-09-02 20:06:33	LINESTRING (-121.13500 46.88830, -121.12850 46...	2.449000e+06	84.648001
56	aade7f	2021-08-14 23:40:09	2021-09-17 23:34:40	LINESTRING (-121.52360 40.07420, -121.49750 40...	1.231389e+07	156.713496
57	ab4ea1	2021-08-03 22:52:00	2021-09-24 21:18:24	LINESTRING (-117.71900 34.17460, -117.71420 34...	4.027793e+05	246.842754
58	c01aeb	2021-08-16 01:45:42	2021-09-04 17:02:19	LINESTRING (-117.57890 47.89770, -117.59080 47...	2.702104e+05	270.914236
59	c044b4	2021-08-17 05:16:44	2021-08-17 05:18:28	LINESTRING (-120.85000 47.81790, -120.86380 47...	1.227591e+04	300.195364

```
#create geojson of gdf_traj
gdf_traj.to_file(filename='gdf_traj_orgTS.geojson', driver='GeoJSON')
```

```
D:\anaconda3\envs\master_env\lib\site-packages\geopandas\io\file.py:362: F
utureWarning: pandas.Int64Index is deprecated and will be removed from pan
das in a future version. Use pandas.Index with the appropriate dtype inste
ad.
  pd.Int64Index,
```

```
#split trajectories inti single flights of the aircraft, that currently
equals the trajectory id
split_collection =
mpd.ObservationGapSplitter(collection).split(gap=timedelta(hours=1))
#create a gdf containing trajectories from collection
```

```
gdf_split_traj = split_collection.to_traj_gdf(wkt=False)
gdf_split_traj
```

	traj_id	start_t	end_t	geometry	length	direction
0	a02862_0	2021-08-01 01:19:23	2021-08-01 01:27:32	LINESTRING (-121.10220 39.91300, -121.10920 39...	8685.843986	39.598238
1	a02862_2	2021-08-18 00:49:23	2021-08-18 02:25:29	LINESTRING (-120.70470 38.60880, -120.70120 38...	227848.633706	42.060865
2	a02862_3	2021-08-18 16:21:45	2021-08-18 18:50:32	LINESTRING (-120.68580 38.66870, -120.67300 38...	140454.607426	59.589732
3	a02862_4	2021-08-18 20:32:18	2021-08-18 23:10:47	LINESTRING (-120.64180 38.73780, -120.61470 38...	219142.055673	196.493291
4	a02862_5	2021-08-19 00:18:44	2021-08-19 02:29:19	LINESTRING (-120.64740 38.73800, -120.63540 38...	230519.373323	199.050468
...	...	...	...	...	...	...
636	ab4ea1_1	2021-08-25 23:14:40	2021-08-25 23:56:29	LINESTRING (-117.49380 34.15960, -117.48480 34...	92901.243372	85.183535
637	ab4ea1_2	2021-08-26 01:03:10	2021-08-26 02:03:19	LINESTRING (-117.38040 34.14600, -117.38470 34...	125874.243672	280.121528
638	ab4ea1_3	2021-09-24 20:56:08	2021-09-24 21:18:24	LINESTRING (-118.14250 34.06280, -118.13680 34...	48010.499296	186.557200
639	c01aeb_0	2021-08-16 01:45:42	2021-08-16 02:36:10	LINESTRING (-117.57890 47.89770, -117.59080 47...	20018.906053	277.678108
640	c044b4_0	2021-08-17 05:16:44	2021-08-17 05:18:28	LINESTRING (-120.85000 47.81790, -120.86380 47...	12275.905919	300.195364

641 rows × 6 columns

```
#create geojson of gdf_split_traj
gdf_split_traj.to_file(filename='gdf_split_traj_orgTS.geojson',
driver='GeoJSON')
```

```
D:\anaconda3\envs\master_env\lib\site-packages\geopandas\io\file.py:362: F
utureWarning: pandas.Int64Index is deprecated and will be removed from pan
das in a future version. Use pandas.Index with the appropriate dtype inste
ad.
  pd.Int64Index,
```

End of script

Code 8 Aircraft_ovr_fires_to_trajectories.ipynb

8.2.4. Relating 4.2.4, The Time Gap between Fire Detection and TFR Issue

Get_Detection-Issue-Time_Gap.ipynb (Code 9) models the first TFR for a fire cluster and calculates timespans. Get_Detection-Issue-Time_Gap_v2.ipynb is an alternative without API usage (visit GitHub as of 8.1).

A script to get current fire detection times per TFR from Ororatech API

```
import requests
import json
import geopandas as gpd
import pandas as pd
import datetime as dt
from datetime import timedelta
import sys

#provide credentials for API access, obtain from Ororatech and insert here
APIkey = 'someSuperLongApiKey'
```

Get the current fires

A single hotspot is only contained in the API response, if its center is within the bounding coordinates. So it is no clean solution to search for fires using TFR bounding boxes. Instead, fire clusters and TFR polygons need to get joined first. Then, the boundary box of the fire cluster can be used to spatially limit the area, where a time based search for hotspots being active before TFR issue time can be performed to calculate time between fire detection and TFR being issued.

As it turned out that ARTCCs issue TFRs beyond their FIR´s boundary, the entire fire cluster dataset has to be joined each time and will be read therefor. 'minutes' variable determines, how long it shall be looked for fire clusters before TFR issue time.

```
##specify file location and name
#3 letter location indicator, reused within all created files
#possible values: ZAB ZDV ZFW ZLA ZLC ZMP ZOA ZSE; not ZKC and ZHU having
no wildfire TFRs
tfr =r"ZDV"
#paths and filenames
fire_path = r"D:\UNIGIS\MASTER\DownloadedData\WFS\\"
tfr_path = r"D:\UNIGIS\MASTER\Scripts\\"
out_path = tfr_path

fire_file = "wfs-area-export_FIRs_Boundary_08-102021_con_pt5.geojson"
tfr_file = tfr + r"_fire_TFRs.geojson"

#create geodataframes
gdf_fires = gpd.read_file(fire_path + fire_file)
gdf_tfrs = gpd.read_file(tfr_path + tfr_file)

#set minutes_bf as search period to look at before TFR issue date
minutes_bf = 1440
#set pos_delta as search period to look at after(!) TFR issue date
pos_delta = 360
#compose minutes for API request
minutes = minutes_bf + pos_delta

#set start date of observed datasets
obs_start = pd.to_datetime('2021-08-01')

#no spatial output is generated here
#result lists filename
```

```
 # for Anaconda / excel output xltw module is required: conda install -c a
naconda xlwt
result_list = out_path+tfr+'_afterDetectionTimes_list.csv'

print('Fires: '+str(len(gdf_fires.index)))
print('TFRs: '+str(len(gdf_tfrs.index)))
```

```
Fires: 2926
TFRs: 28
```

Purge old TFRs issued before observation time that can hardly match fire data:

```
gdf_tfrs = gdf_tfrs[pd.to_datetime(gdf_tfrs["Issue Date (UTC)"]) >= obs_st
art]
```

Remove old fires already burning before observation time frame (and that would otherwise most probably get an accidentally intersecting younger TFR joined)

```
#sort out ongoing fires
#need to localize obs_start to compare to UTC format date
gdf_fires = gdf_fires[pd.to_datetime(gdf_fires["oldest_acquisition"]) >= o
bs_start.tz_localize('UTC') - timedelta(minutes = minutes_bf)]

print('Fires: '+str(len(gdf_fires.index)))
print('TFRs: '+str(len(gdf_tfrs.index)))
```

```
Fires: 2804
TFRs: 27
```

Join fire clusters and TFRs

As the fire clusters used so far represent the largest extent of the fires within the observed time period (August to October 2021), without any temporal relation yet, the same fire may be tied to multiple TFRs. For fires close to each other, issuing one large contiguous TFR is allowed, so the same TFR might contain multiple fires as well. So the resulting geodataframe can contain a multiple of rows compared to the origin.

Geopandas sjoin is "one to many" automatically. To look for intersect is expected to deliver many fire clusters!

```
#perform join of gdfs
gdf_intersect = gpd.sjoin(gdf_fires, gdf_tfrs, how='inner', predicate='int
ersects')

# change the global options that Geopandas inherits from if more shall be
displayed
# pd.set_option('display.max_columns',None)
# pd.set_option('display.max_rows',None)
#gdf_intersect

#If it is ever needed to join a join result again, index columns need to g
et renamend
#gdf_intersect.rename(columns = {'index_right':'old_index_right'}, inplace
= True)
```

This is one theoretical possible "exit point": If nothing intersects, then the resulting geodataframe has 0 rows.

Number of intersections

```
print(len(gdf_intersect.index))
```

```
19
```

```
# exit script with an empty result file if intersect geodataframe has 0 ro
ws
if len(gdf_intersect.index) ==0:
    gdf_intersect.to_csv(result_list, columns=['id'], header=['No data fro
m '+tfr])
    sys.exit(0)
```

ZDV : 20

Tasks to fill payload to get current fires:

To create a payload per fire per TFR, each time

A) bounding box coordinates and

B) a time period represented as (end) 'date' and duration in 'minutes' are needed. Thus, minutes are specified globally above.

A) Bounding Box

The fire clusters intersecting any TFR are those to look for.

Bounding box coordinates are created and then added to the geodataframe.

```
# Geopands´ .bounds delivers coordinates of the boundary boxes
gdf_bbox = gdf_intersect.bounds
```

```
gdf_bbox.head(2)
```

```
           minx      miny        maxx       maxy
107 -105.994748  45.03517 -105.914179  45.093576
107 -105.994748  45.03517 -105.914179  45.093576
```

```
#just use pd.concat / axis=1 to append boundary box coordinates
gdf_intersect = pd.concat([gdf_intersect, gdf_bbox], axis=1)
```

```
#check attached columns, if needed
gdf_intersect.head(2)
```

```
           id     age          area  \
107  19806634  125497  2.007181e+07
107  19806634  125497  2.007181e+07

                                          centroid  num_fires  confide
nce  \
107  {'latitude': 45.060425, 'longitude': -105.961138}        140
1.0
107  {'latitude': 45.060425, 'longitude': -105.961138}        140
1.0

              newest_detection           oldest_detection  \
107  2021-08-05T23:17:12+00:00  2021-08-02T03:49:31+00:00
```

```
107  2021-08-05T23:17:12+00:00  2021-08-02T03:49:31+00:00

            newest_acquisition         oldest_acquisition  ...  \
107  2021-08-05T20:22:41+00:00  2021-08-02T03:09:51+00:00  ...
107  2021-08-05T20:22:41+00:00  2021-08-02T03:09:51+00:00  ...

    Effective Date (UTC)  Cancel Date (UTC) Expiration Date (UTC)  \
107      08/05/2021 1400    08/07/2021 0427       09/30/2021 0400
107      08/05/2021 0015               None       08/05/2021 0400

                        NOTAM Condition or LTA Subject Radius Radius_m  \
107  !FDC 1/1635 ZDV MT..AIRSPACE 31NM SW OF BROADU...    7.0  12964.0
107  !FDC 1/1585 ZDV CANCELLED BY FDC 1/1659 ON 08/...    7.0  12964.0

           minx      miny        maxx       maxy
107 -105.994748  45.03517 -105.914179  45.093576
107 -105.994748  45.03517 -105.914179  45.093576

[2 rows x 26 columns]
```

B) Get date(time) for the API request

The date from when the API goes back needs to get acquired as follows: 'date': '2021-08-16-0200' A problem is that a TFR might have been issuded based on ground information while EO has not discovered a fire yet. So a TFR dedicated to a fire cluster might have become issued also hours before the cluster appears within Ororatech´s data.

A TFR might have been issued prior to sat acquisiton from ground knowledge. In cosequence, the API request has to look forward as well.

```
#get (end)date for the API request
def get_apidate(row):

    enddate= row["Issue Date (UTC)"]

    #endate is 'MM/DD/YYYY hhmm' format (s string)
    #turn to  'YYYY-MM-DD-hhmm' format for API. Mind strftime() Directives
    enddate = pd.to_datetime(enddate,errors='coerce') + timedelta(minutes
= pos_delta)
    enddate = enddate.strftime('%Y-%m-%d-%H%M')

    apidate = enddate

    return apidate
gdf_intersect["APIdate"]=gdf_intersect.apply(get_apidate,axis=1)

gdf_intersect["APIdate"].head(2)
```

```
107    2021-08-05-0833
107    2021-08-05-0619
Name: APIdate, dtype: object
```

Perform the API request

A minimum confidence of 0.5 is recommended by Ororatech for analysis of historical data. 'select' delivers additional columns. Here, the oldest detection time is needed for the timespan calculation planned below. 'confidence' must be set to 0.4 as API looks for everything ABOVE and the initial dataset followed Ororatech´s advice to use 0.5 for historical data. Additionally, the type is requested (see 4.2.3.1)

```
def get_clusterPerAPI(row):
    #collect payload content per row
    xmin_pl = str(row["minx"])
    ymin_pl = str(row["miny"])
    xmax_pl = str(row["maxx"])
    ymax_pl = str(row["maxy"])

    date_pl = str(row["APIdate"])

    payload = {'xmin': xmin_pl,
                    'ymin': ymin_pl,
                    'xmax': xmax_pl,
                    'ymax': ymax_pl,
                    'minutes': minutes,
                    'date': date_pl,
                    'confidence': '0.4',
                    'select': ['oldest_detection,oldest_acquisition,types']
,
                    'token': APIkey}

    #the request:
    response = requests.get('https://app.ororatech.com/v1/clusters/',param
s=payload)

    #test request:
    #testpayload = {'xmin': '-117.86', 'ymin': '47.88', 'xmax': '-117.53',
'ymax': '48.00', 'minutes': '360', 'date': '2021-08-16-0200','confidence':
'0.5', 'token': APIkey}
    #response = requests.get('https://app.ororatech.com/v1/clusters/',para
ms=testpayload)

    data = response.json()
    #if data is not None does not help in case of an "empty" json
    #in that case, response.json() = {'type': 'FeatureCollection', 'featur
es': None}
    #would lead to an error trying to create a gdf from features
    if data != {'type': 'FeatureCollection', 'features': None}:
        #response json has its columns=['geometry', 'id', 'num_fires', 'ol
dest_detection', 'oldest_acquisition', 'types']
        gdf_local = gpd.GeoDataFrame.from_features(data)
    else:
        #else prepare an empty gdf for return
        gdf_local = gpd.GeoDataFrame()

    return gdf_local

series_of_gdfs = gdf_intersect.apply(get_clusterPerAPI,axis=1)
list_of_gdfs= series_of_gdfs.tolist()
```

```
#concat returned gdfs; empty ones are not cosidered by default
gdf_current = gpd.GeoDataFrame(pd.concat(list_of_gdfs, ignore_index=True))

#gdf_current is a geodataframe, but crs has still to be specified
gdf_current = gdf_current.set_crs("EPSG:4326", allow_override=True)

gdf_current.head(2)
```

```
                                            geometry        id types  \
0  POLYGON ((-105.98341 45.06118, -105.98321 45.0...  19806634   [1]
1  POLYGON ((-105.98341 45.06118, -105.98321 45.0...  19806634   [1]

   num_fires           oldest_detection         oldest_acquisition
0        140  2021-08-04T22:24:59+00:00  2021-08-02T03:09:51+00:00
1        140  2021-08-04T22:24:59+00:00  2021-08-02T03:09:51+00:00
```

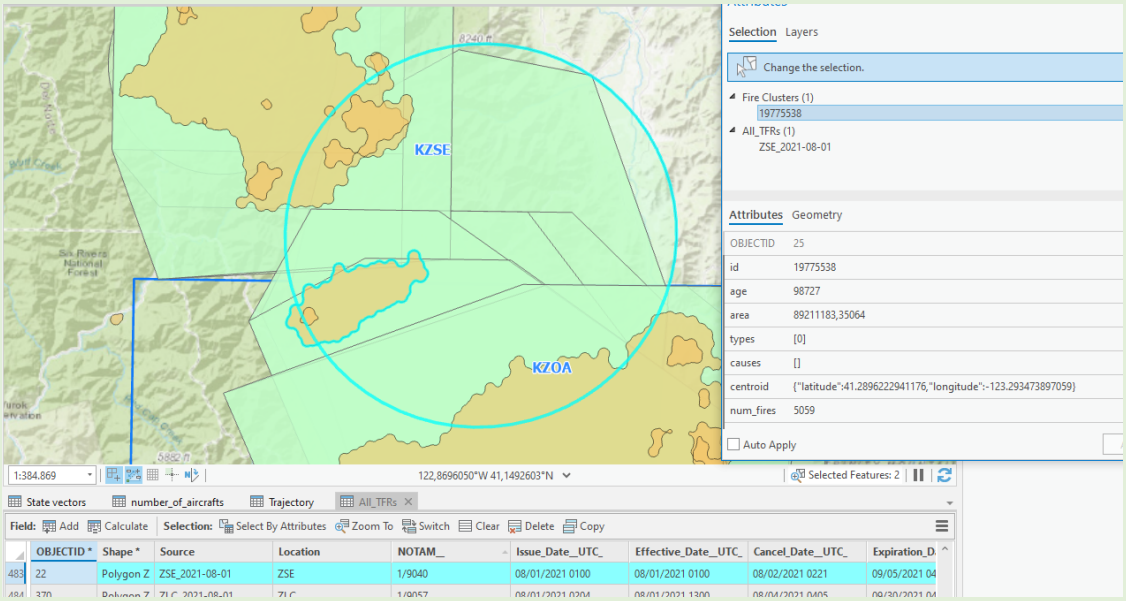
If there is an intersect between TFRs and API request results, then those TFRs should be those which originally "belong" to that fire. However, later and earlier TFRs may be contained here as well (if any).They are taken care of in a minute. First, the results from the API can again contain ongoing fires. Result would be long Timespan Acquisition and possibly trustworthy but meaningless Timespan Detection. Thus, they get sorted out again.

```
#2nd sort out ongoing fires
#need to localize obs_start to compare to UTC format date
gdf_current = gdf_current[pd.to_datetime(gdf_current["oldest_acquisition"]
) >= obs_start.tz_localize('UTC') - timedelta(minutes = minutes_bf)]

#perform 2nd join of gdfs
gdf_current_intersect = gpd.sjoin(gdf_current, gdf_tfrs, how='inner', pred
icate='intersects')

gdf_current_intersect.head(2)
```

```
                                            geometry        id types  \
0  POLYGON ((-105.98341 45.06118, -105.98321 45.0...  19806634   [1]
1  POLYGON ((-105.98341 45.06118, -105.98321 45.0...  19806634   [1]

   num_fires           oldest_detection         oldest_acquisition  \
0        140  2021-08-04T22:24:59+00:00  2021-08-02T03:09:51+00:00
1        140  2021-08-04T22:24:59+00:00  2021-08-02T03:09:51+00:00

   index_right          Source Location NOTAM # Issue Date (UTC)  \
0            1  ZDV_2021-08-05      ZDV  1/1635  08/05/2021 0233
1            1  ZDV_2021-08-05      ZDV  1/1635  08/05/2021 0233

  Effective Date (UTC) Cancel Date (UTC) Expiration Date (UTC)  \
0      08/05/2021 1400   08/07/2021 0427       09/30/2021 0400
1      08/05/2021 1400   08/07/2021 0427       09/30/2021 0400

                      NOTAM Condition or LTA Subject  Radius  Radius_m
0  !FDC 1/1635 ZDV MT..AIRSPACE 31NM SW OF BROADU...     7.0   12964.0
1  !FDC 1/1635 ZDV MT..AIRSPACE 31NM SW OF BROADU...     7.0   12964.0
```

Now it is important to understand what the result of API request and the 2nd intersect can be. If a 3-month-fire-cluster intersected with more TFRs, then different oldest acquisition and detection dates are possible to be contained in the above gdf_current_intersect. Removing duplicates concerning fire cluster id would deliver the first TFR for each fire event then, but this leads to addtional skewed results where a TFR overlaps multiple fire clusters (thus, also a younger fire, which does happen e.g. in ZSE). So, another solution to narrow down results is chosen to perform the calculation for all fires on all TFRs and then sort by the absolute value sum to keep the TFR that was issued with the lowest temporal distance to the fire, assuming that this is the TFR intended for the fire.

id	Fire type	Fire detection	Fire acquisition	NOTAM # from ZSE	Issue Date (UTC)	Timespan Detection	Timespan Acquisition
19775538	[0]	09.08.2021 06:47	31.07.2021 22:16	1/9040	08/01/2021 0100	-11747	283
19779404	[0]	02.09.2021 01:06	01.08.2021 07:10	1/9040	08/01/2021 0100	-45966	-250
19779404	[0]	28.08.2021 06:50	01.08.2021 07:10	1/9040	08/01/2021 0100	-39110	-250
19779404	[0]	28.08.2021 06:50	01.08.2021 07:10	1/9040	08/01/2021 0100	-39110	-250
19779404	[0]	28.08.2021 02:45	01.08.2021 07:10	1/9040	08/01/2021 0100	-38865	-250
19779404	[0]	09.09.2021 06:58	01.08.2021 07:10	1/9040	08/01/2021 0100	-56398	-250
19779404	[0]	09.09.2021 06:58	01.08.2021 07:10	1/9040	08/01/2021 0100	-56398	-250
21714880	[0]	27.09.2021 08:14	20.09.2021 20:51	1/9040	08/01/2021 0100	-82394	-73071
19775538	[0]	01.08.2021 01:08	31.07.2021 22:16	1/9040	08/01/2021 0100	111	283
19775538	[0]	01.08.2021 07:50	31.07.2021 22:16	1/9040	08/01/2021 0100	-290	283
19784784	[0]	01.08.2021 14:41	01.08.2021 12:14	1/9040	08/01/2021 0100	-701	-554
19775538	[0]	01.08.2021 07:50	31.07.2021 22:16	1/9040	08/01/2021 0100	-290	283
19784784	[0]	01.08.2021 14:41	01.08.2021 12:14	1/9040	08/01/2021 0100	-701	-554
19775538	[0]	05.08.2021 00:25	31.07.2021 22:16	1/9040	08/01/2021 0100	-5605	283
19773091	[1]	05.08.2021 00:24	31.07.2021 20:59	1/9040	08/01/2021 0100	-5604	360
19775538	[0]	06.08.2021 07:11	31.07.2021 22:16	1/9040	08/01/2021 0100	-7451	283
19775538	[0]	07.08.2021 00:43	31.07.2021 22:16	1/9040	08/01/2021 0100	-8503	283
19773091	[1]	06.08.2021 07:11	31.07.2021 20:59	1/9040	08/01/2021 0100	-7451	360
19775538	[0]	09.08.2021 14:30	31.07.2021 22:16	1/9040	08/01/2021 0100	-12210	283
19773091	[1]	09.08.2021 06:47	31.07.2021 20:59	1/9040	08/01/2021 0100	-11747	360
20753704	[0]	29.08.2021 00:55	26.08.2021 23:18	1/9040	08/01/2021 0100	-40195	-37218
20753704	[0]	29.08.2021 00:55	26.08.2021 23:18	1/9040	08/01/2021 0100	-40195	-37218
19773091	[1]	28.08.2021 22:38	31.07.2021 20:59	1/9040	08/01/2021 0100	-40058	360
19775538	[0]	20.08.2021 07:18	31.07.2021 22:16	1/9040	08/01/2021 0100	-27618	283
19775538	[0]	20.08.2021 07:18	31.07.2021 22:16	1/9040	08/01/2021 0100	-27618	283
19775538	[0]	20.08.2021 07:18	31.07.2021 22:16	1/9040	08/01/2021 0100	-27618	283
19773091	[1]	20.08.2021 07:18	31.07.2021 20:59	1/9040	08/01/2021 0100	-27618	360
19773091	[1]	02.09.2021 01:06	31.07.2021 20:59	1/9040	08/01/2021 0100	-45966	360
19779404	[0]	31.08.2021 21:33	01.08.2021 07:10	1/9040	08/01/2021 0100	-44313	-250
19779404	[0]	01.09.2021 03:14	01.08.2021 07:10	1/9040	08/01/2021 0100	-44654	-250
21714880	[0]	27.09.2021 08:14	20.09.2021 20:51	1/9040	08/01/2021 0100	-82394	-73071
19773092	[1]	31.07.2021 23:58	31.07.2021 19:05	1/9078	08/01/2021 0410	371	664




```
# set gdf_current_intersect as final gdf_result sort gdf by TFR issue date for an interim overview
gdf_result = gdf_current_intersect.sort_values(["Issue Date (UTC)"])
# gdf_result
```

A timespan between fire detection/acquisition and TFR issue time can be calculated (and turned to minutes). Oldest detection/acquisition time is subtracted from TFR issue date.

```
#calculate timespans column wise
# both times to calculate with are UTC but Python does not know yet
# for 'Issue Date (UTC)', as the format does not indicate by "+00:00"
#pd.to_datetime(gdf_result["Issue Date (UTC)"].tz_localize('UTC') ) is use
d

#.astype(int) returns an integer being sufficient as minute value

#oldest_detection
gdf_result["Calc_time_det"] = pd.to_datetime(gdf_result.loc[:,"Issue Date
(UTC)"]).dt.tz_localize('UTC') - pd.to_datetime(gdf_result["oldest_detecti
on"])
gdf_result["Timespan Detection"] = gdf_result.loc[:,"Calc_time_det"].dt.to
tal_seconds().div(60).astype(int)

#oldest_acquisition
gdf_result["Calc_time_acq"] = pd.to_datetime(gdf_result.loc[:,"Issue Date
(UTC)"]).dt.tz_localize('UTC') - pd.to_datetime(gdf_result["oldest_acquisi
tion"])
gdf_result["Timespan Acquisition"] = gdf_result.loc[:,"Calc_time_acq"].dt.
total_seconds().div(60).astype(int)
```

In FIRs with a lot of overlapping patterns, this leads to a lot of duplicates of fire ids and TFRs

```
#count duplicates (fire id)
gdf_result.duplicated(["id"]).sum()
```

```
35
```

```
#count non-duplicates
(~gdf_result.duplicated(["id"])).sum()
```

```
9
```

```
# Check: duplicates of TFRs
gdf_result.duplicated(["NOTAM #"]).sum()
```

```
26
```

```
#count non-duplicates TFRs
(~gdf_result.duplicated(["NOTAM #"])).sum()
```

```
18
```

To filter for the most trustworthy combination (meaning: To match a fire cluster and the TFR wich was the first one intended for it), the absolute temporal distance of a fire cluster from a TFR issue date needs to be summed up for both, detection and acquisition, then the lowest value must be chosen. This matches the highlighted correct cluster/TFR combination from the ZSE screenshot from excel above.

```
# sum up absolute times
gdf_result["Abs_time"] = gdf_result.loc[:,"Timespan Detection"].abs() + gd
f_result.loc[:,"Timespan Acquisition"].abs()
```

```
gdf_result = gdf_result.sort_values(["Abs_time"])
# gdf_result

# spec 'id' column for dropping duplicates, first occurrence and therefor
smallest Abs_time (after sorting above) is kept by default
gdf_result = gdf_result.drop_duplicates(subset=['id'])
```

If fire clusters for all TFRs have been fetched via API, the gdf_result has the same row count as gdf_tfrs. But most probably, sometimes multiple TFRs had to be issued for the same fire event. So length of gdf is not meaningful.

```
print(len(gdf_tfrs.index))
```

```
27
```

```
print(len(gdf_result.index))
```

```
9
```

Not all previously intersected fire clusters are still contained: Not for all of them a specific TFR might have been intended.

```
# Select fire clusters from first intersect, then check for those not repr
esented:
options = gdf_result["id"]
gdf_fires_not_found = gdf_intersect[( ~gdf_intersect["id"].isin(options) )
]
#print(len(gdf_fires_not_found.index))
print ('Fire clusters not incorporated: '  )
(~gdf_fires_not_found.duplicated(["id"])).sum()
```

```
Fire clusters not incorporated:

1
```

Number of results: ZDV : 9 (as expected from checking In ArcGIS Pro)

```
gdf_result.head(2)
```

```
                                              geometry        id types  \
10  POLYGON ((-103.60934 44.32135, -103.60888 44.3...  20177190   [1]
14  POLYGON ((-103.49522 42.60600, -103.49501 42.6...  21594740   [0]

    num_fires           oldest_detection         oldest_acquisition  \
10         23  2021-08-11T23:25:58+00:00  2021-08-11T21:00:13+00:00
14        781  2021-09-17T03:59:11+00:00  2021-09-16T19:15:05+00:00

    index_right          Source Location NOTAM #  ... Cancel Date (UTC)  \
10           16  ZDV_2021-08-11      ZDV  1/5720  ...              None
14           24  ZDV_2021-09-17      ZDV  1/9881  ...   09/17/2021 1812

   Expiration Date (UTC)                    NOTAM Condition or LTA Subjec
t  \
10       08/12/2021 0400  !FDC 1/5720 ZDV CANCELLED BY FDC 1/5865 ON 08/..
.
14       11/18/2021 0200  !FDC 1/9881 ZDV NE..AIRSPACE 8NM S OF CRAWFORD..
.
```

```
   Radius Radius_m     Calc_time_det  Timespan Detection   Calc_time_acq
\
10    5.0   9260.0   0 days 00:15:02                 15 0 days 02:40:47
14    7.0  12964.0 -1 days +20:39:49               -200 0 days 05:23:55

    Timespan Acquisition Abs_time
10                   160      175
14                   323      523

[2 rows x 22 columns]
```

Finally, compose an output file as csv to list timespan between detection and TFR issue time

```
# write output to file
gdf_result.to_csv(result_list, columns=['id','types', 'oldest_detection','
oldest_acquisition', 'NOTAM #','Issue Date (UTC)', 'Timespan Detection', '
Timespan Acquisition'], header=['id','Fire type', 'Fire detection','Fire a
cquisition', 'NOTAM # from '+tfr,'Issue Date (UTC)', 'Timespan Detection',
'Timespan Acquisition'], index=None, sep=' ', mode='w')
```

This Notebook ends here.

Code 9 Get_Detection-Issue-Time_Gap.ipynb

8.2.5. Relating 4.2.5, Coverage Quality

Get_TFR-exceeding_Fires_from_API.ipynb (Code 10) fetches all cases of fires leaving a TFR. The step to acquire lists of fires and TFRs in question is processed by Get_Events_from_Fires_from_API.ipynb (Code 11). A quick check of the fire cluster types is performed by Get_Types_from_exceeding_Fires.ipynb (Code 12).

A script to get current fires per TFR from Ororatech API

and to find exceeding ones

```
import requests
import json
import geopandas as gpd
import pandas as pd
import sys
```

Get the current fires

```
#provide credentials for API access, obtain from Ororatech and insert here
APIkey = 'someSuperLongApiKey'
```

A single hotspot is only contained in the API response, if its center is within the bounding coordinates. So it is no clean solution to search for fires using TFR bounding boxes. Instead, buffered fire clusters and TFR polygons need to get joined first. Then, the boundary box of the fire cluster can be used to spatially limit the area, where a time based search for hotspots being active within TFR runtime can be performed to detect runaway fires.

As it turned out that ARTCCs issue TFRs beyond their FIR´s boundary, the entire fire cluster dataset has to be joined each time and will be read therefor.

```
##specify file location and name
#3 letter location indicator, reused within all created files
#possible values: ZAB ZDV ZFW ZLA ZLC ZMP ZOA ZSE; not ZKC and ZHU having
no wildfire TFRs
tfr =r"ZDV"
#paths and filenames
fire_path = r"D:\UNIGIS\MASTER\DownloadedData\WFS\\"
tfr_path = r"D:\UNIGIS\MASTER\Scripts\\"
out_path = r"D:\UNIGIS\MASTER\Scripts\Exceeding\\"

fire_file = "wfs-area-export_FIRs_Boundary_08-102021_con_pt5.geojson"
tfr_file = tfr + r"_fire_TFRs.geojson"

#read geojson files
gdf_fires = gpd.read_file(fire_path + fire_file)
gdf_tfrs = gpd.read_file(tfr_path + tfr_file)

#size to buffer fires in m; 1609.344 m = 1 SM, 4828.032 m = 3 SM
buf_size = 4828.032

#output concatenates from these strings as well. Buffer size must be a str
ing for that as well
str_buf_size = str(int(buf_size/1609.344))

#output filename (if any):
outfile = out_path+tfr+'_'+str_buf_size+'SM_runaway_fires.geojson'

#logfile filename (to append some row/feature counts)
logfilename = out_path+'Fires_from_API_log.txt'
```

Number of evaluated TFRs per FIR, ZDV: 28

```
print(len(gdf_tfrs.index))
```

```
28
```

```
# Appending to logfile
with open(logfilename, 'a') as logfile:
    logfile.write('Number of evaluated TFRs for '+tfr+' '+str_buf_size+' S
M: '+str(len(gdf_tfrs.index))+'\n' )
```

Buffer fire clusters

Without any temporal relation yet, the same fire may be tied to multiple TFRs. For fires close to each other, issuing one large contiguous TFR is allowed, so the same TFR might contain multiple fires as well. So the resulting geodataframe can contain a multiple of rows compared to the origin. As the fire clusters used so far represent the largest extent of the fires within the observed time period (August to October 2021), those not leaving their TFR (after the fire cluster got buffered) can be omitted here. This is also done to limit the amount of API requests. But those fires (plus buffer) crossing TFR boundaries (=overlap in shapely terms) do need a closer look considering time.

Now buffer fire clusters by specified amount of statute miles to get prepared for the overlap.

```
#prepare buffer with a metric CRS
gdf_fires_buffered = gdf_fires.copy()
```

```
gdf_fires_buffered = gdf_fires_buffered.to_crs("EPSG:2163")

#perform buffer by amount of statute miles (SM), 1 SM = 1609.344 m
gdf_fires_buffered["geometry"] = gdf_fires_buffered.buffer(buf_size, resol
ution=16)

#turn CRS back
gdf_fires_buffered = gdf_fires_buffered.to_crs("EPSG:4326")

gdf_fires_buffered.head(2)
```

```
         id     age          area  \
0  19778761  131964  1.697506e+07
1  19775166  131967  9.360254e+06

                                          centroid  num_fires  confidenc
e  \
0   {'latitude': 29.051065, 'longitude': -97.286025}         47         0.
8
1  {'latitude': 43.104132, 'longitude': -102.572029}         45         1.
0

            newest_detection           oldest_detection  \
0  2021-08-01T11:07:57+00:00  2021-07-31T23:25:52+00:00
1  2021-08-01T11:04:44+00:00  2021-08-01T03:25:31+00:00

          newest_acquisition         oldest_acquisition  \
0  2021-08-01T08:36:21+00:00  2021-07-31T23:15:05+00:00
1  2021-08-01T08:32:35+00:00  2021-07-31T19:25:18+00:00

                                          geometry
0  POLYGON ((-97.29353 29.11033, -97.28962 29.111...
1  POLYGON ((-102.63812 43.11719, -102.63716 43.1...
```

Join buffered fire clusters and TFRs

Geopandas sjoin is "one to many" automatically. Inner join is needed to limit fire clusters to those actually fulfilling the predicate.

Just to look for intersect is expected to deliver way too many fire clusters (even with unbuffered ones).

```
#perform join of gdfs
#gdf_intersect = gpd.sjoin(gdf_fires, gdf_tfrs, how='inner', predicate='in
tersects')
#gdf_intersect

#If it is ever needed to join a join result again, index columns need to g
et renamend
#gdf_intersect.rename(columns = {'index_right':'old_index_right'}, inplace
= True)

gdf_runfires_bybuff = gpd.sjoin(gdf_fires_buffered, gdf_tfrs, how='inner',
predicate='overlaps')
```

This is one possible "exit point": If nothing overlaps, then the buffered fires are contained within the TFRs and the resulting geodataframe has 0 rows. Exit is then performed after the follwoing 2 log entries.

Within the observed data, this is the case for: ZAB

However, there is an exception for an extreme case, if the fire plus buffer has already spread over the entire TFR, this has to be checked:

```
#check for buffered fire already covering entire TFR
gdf_coveringfires_bybuff = gpd.sjoin(gdf_fires_buffered, gdf_tfrs, how='in
ner', predicate='contains')

gdf_coveringfires_bybuff.head(2)
```

```
           id    age          area  \
862  20866464  87258  2.916087e+07

                                            centroid  num_fires  confidenc
e  \
862  {'latitude': 40.1618, 'longitude': -106.237305}        587         1.
0

              newest_detection           oldest_detection  \
862  2021-09-01T12:22:13+00:00  2021-08-29T22:24:39+00:00

            newest_acquisition         oldest_acquisition  ...  \
862  2021-09-01T09:42:27+00:00  2021-08-29T19:31:23+00:00  ...

             Source  Location NOTAM # Issue Date (UTC) Effective Date (UTC
)  \
862  ZDV_2021-08-30       ZDV  1/8134  08/30/2021 0201      08/30/2021 140
0

    Cancel Date (UTC) Expiration Date (UTC)  \
862   09/07/2021 1405       10/29/2021 0200

                        NOTAM Condition or LTA Subject Radius Radius_m
862  !FDC 1/8134 ZDV CO..AIRSPACE 8NM NE OF KREMLIN...    3.0   5556.0

[1 rows x 22 columns]
```

```
#concatenate dataframes if buffered fires contain an entire TFR
if len(gdf_coveringfires_bybuff.index)>0:
    frames = [gdf_runfires_bybuff, gdf_coveringfires_bybuff]
    gdf_runfires_bybuff = gpd.GeoDataFrame(pd.concat(frames, sort=False))



# change the global options that Geopandas inherits from if more rows/colu
mns shall be displayed
# pd.set_option('display.max_columns',None)
gdf_runfires_bybuff.head(2)
```

```
           id     age          area  \
257  19952554  121182  4.859015e+07
```

```
257  19952554  121182  4.859015e+07

                                              centroid  num_fires  confide
nce  \
257  {'latitude': 41.527944, 'longitude': -103.350363}        635
1.0
257  {'latitude': 41.527944, 'longitude': -103.350363}        635
1.0

              newest_detection           oldest_detection  \
257  2021-08-09T18:11:45+00:00  2021-08-06T04:06:48+00:00
257  2021-08-09T18:11:45+00:00  2021-08-06T04:06:48+00:00

            newest_acquisition         oldest_acquisition  ...  \
257  2021-08-09T02:55:05+00:00  2021-08-06T02:55:05+00:00  ...
257  2021-08-09T02:55:05+00:00  2021-08-06T02:55:05+00:00  ...

             Source  Location NOTAM # Issue Date (UTC) Effective Date (UTC
)  \
257  ZDV_2021-08-06       ZDV  1/2974  08/06/2021 1847      08/06/2021 183
0
257  ZDV_2021-08-07       ZDV  1/3125  08/07/2021 1436      08/07/2021 145
0

    Cancel Date (UTC) Expiration Date (UTC)  \
257              None       08/07/2021 0400
257   08/07/2021 2018       08/08/2021 0300

                        NOTAM Condition or LTA Subject Radius Radius_m
257  !FDC 1/2974 ZDV CANCELLED BY FDC 1/3037 ON 08/...    7.0  12964.0
257  !FDC 1/3125 ZDV NE..AIRSPACE 24NM SSE SCOTTSBL...    7.0  12964.0

[2 rows x 22 columns]
```

Number of cases where fire leaves TFR

```
print(len(gdf_runfires_bybuff.index))
```

```
11
```

ZDV (3 SM): 11

ZDV (1 SM): 5

```
# Appending to logfile
with open(logfilename, 'a') as logfile:
    logfile.write('Number of potential cases where fire leaves TFR for '+t
fr+' '+str_buf_size+' SM: '+str(len(gdf_runfires_bybuff.index))+'\n' )
```

Tasks to fill payload to get current fires:

To create a payload per fire per TFR, each time

A) bounding box coordinates and

B) a time period represented as (end) 'date' and duration in 'minutes' are needed.

A) Bounding Box

The buffered fire clusters overlapping any TFR are potentially those causing a safety threat. The following "selection" shows, how many there are.

```
# Select fire clusters that overlapped while being buffered to get a fire(
row) count to mention.
options = gdf_runfires_bybuff["id"]
gdf_fires_tofetch = gdf_fires[gdf_fires["id"].isin(options)]

print(len(gdf_fires_tofetch.index))
```

```
7
```

Number of fire events leaving TFR

ZDV (3 SM): 7

ZDV (1 SM): 1

```
# Appending to logfile
with open(logfilename, 'a') as logfile:
    logfile.write('Number of potential fire events leaving TFR for '+tfr+'
'+str_buf_size+' SM: '+str(len(gdf_fires_tofetch.index))+'\n' )

# exit script with an empty result file if intersect geodataframe has 0 ro
ws
if len(gdf_runfires_bybuff.index) ==0:
    sys.exit(0)
```

For simplification, the work is continued with the buffered dataset. Bounding box coordinates are added to the geodataframe.

```
# Geopands´ .bounds delivers coordinates of the boundary boxes
gdf_bbox = gdf_runfires_bybuff.bounds

gdf_bbox.head(2)
```

```
           minx       miny       maxx      maxy
257 -103.472792  41.444095 -103.24504  41.60568
257 -103.472792  41.444095 -103.24504  41.60568
```

```
#just use pd.concat / axis=1 to append boundary box coordinates
gdf_runfires_bybuff = pd.concat([gdf_runfires_bybuff, gdf_bbox], axis=1)

#check attached columns, if needed
gdf_runfires_bybuff.head(2)
```

```
           id     age          area  \
257  19952554  121182  4.859015e+07
257  19952554  121182  4.859015e+07

                                              centroid  num_fires  confide
nce  \
257  {'latitude': 41.527944, 'longitude': -103.350363}        635
1.0
257  {'latitude': 41.527944, 'longitude': -103.350363}        635
1.0
```

```
              newest_detection           oldest_detection  \
257  2021-08-09T18:11:45+00:00  2021-08-06T04:06:48+00:00
257  2021-08-09T18:11:45+00:00  2021-08-06T04:06:48+00:00

            newest_acquisition         oldest_acquisition  ...  \
257  2021-08-09T02:55:05+00:00  2021-08-06T02:55:05+00:00  ...
257  2021-08-09T02:55:05+00:00  2021-08-06T02:55:05+00:00  ...

    Effective Date (UTC)  Cancel Date (UTC) Expiration Date (UTC)  \
257      08/06/2021 1830               None       08/07/2021 0400
257      08/07/2021 1450    08/07/2021 2018       08/08/2021 0300

                        NOTAM Condition or LTA Subject Radius Radius_m  \
257  !FDC 1/2974 ZDV CANCELLED BY FDC 1/3037 ON 08/...    7.0  12964.0
257  !FDC 1/3125 ZDV NE..AIRSPACE 24NM SSE SCOTTSBL...    7.0  12964.0

           minx       miny       maxx      maxy
257 -103.472792  41.444095 -103.24504  41.60568
257 -103.472792  41.444095 -103.24504  41.60568

[2 rows x 26 columns]
```

B) (Part 1) Get minutes for the API request

Getting the time values for the request cannot be done on column level as the Cancel Date (UTC) may be empty or even be before the Effective Date (UTC) if there is no longer a threat or a flight planned.

```
#get minutes value for the API request
def get_minutes(row):
    startdate = row["Effective Date (UTC)"]

    enddate= row["Cancel Date (UTC)"]
    if enddate == None:
        enddate= row["Expiration Date (UTC)"]

    duration = pd.to_datetime(enddate, errors='coerce') - pd.to_datetime(s
tartdate, errors='coerce')

    minutes = duration.total_seconds()/60

    minutes = int(minutes)

    return minutes
gdf_runfires_bybuff["Minutes"]=gdf_runfires_bybuff.apply(get_minutes,axis=
1)

#column-wise calculation if it was possible
# gdf_runfires_bybuff["restr_duration"] = pd.to_datetime(gdf_runfires_bybu
ff["Cancel Date (UTC)"], errors='coerce') - pd.to_datetime(gdf_runfires_by
buff["Effective Date (UTC)"], errors='coerce')
# gdf_runfires_bybuff["Minutes"] = gdf_runfires_bybuff["restr_duration"].d
t.total_seconds().div(60)
```

```
gdf_runfires_bybuff["Minutes"].head(2)
```

```
257    570
257    328
Name: Minutes, dtype: int64
```

For the API, 0 or a negative value is invalid: {'message': 'Invalid range for minutes parameter'} Thus, the related rows need to become removed

```
#keep only rows where "Minutes" >= 0
gdf_runfires_bybuff = gdf_runfires_bybuff[gdf_runfires_bybuff["Minutes"]>=0]
```

B) (Part 2) Get date(time) for the API request

The date from when the API goes back needs to get acquired as follows: 'date': '2021-08-16-0200'

```
#get date for the API request
def get_apidate(row):

    enddate= row["Cancel Date (UTC)"]
    if enddate == None:
        enddate= row["Expiration Date (UTC)"]
        #if both enddate columns become read, a SettingWithCopyWarning occurs, which is ok

    #endate is 'MM/DD/YYYY hhmm' format (s string)
    #turn to  'YYYY-MM-DD-hhmm' format for API. Mind strftime() Directives
    enddate = pd.to_datetime(enddate,errors='coerce')
    enddate = enddate.strftime('%Y-%m-%d-%H%M')

    apidate = enddate

    return apidate
gdf_runfires_bybuff["APIdate"]=gdf_runfires_bybuff.apply(get_apidate,axis=1)
```

```
gdf_runfires_bybuff["APIdate"].head(2)
```

```
257    2021-08-07-0400
257    2021-08-07-2018
Name: APIdate, dtype: object
```

Perform the API request

A minimum confidence of 0.5 is recommended by Ororatech for analysis of historical data. So, 'confidence' must be set to 0.4 as the API looks for everything ABOVE.

```
def get_clusterPerAPI(row):
    #collect payload content per row
    xmin_pl = str(row["minx"])
    ymin_pl = str(row["miny"])
    xmax_pl = str(row["maxx"])
    ymax_pl = str(row["maxy"])
    minute_pl = str(row["Minutes"])
    date_pl = str(row["APIdate"])
```

```
    payload = {'xmin': xmin_pl,
                    'ymin': ymin_pl,
                    'xmax': xmax_pl,
                    'ymax': ymax_pl,
                    'minutes': minute_pl,
                    'date': date_pl,
                    'confidence': '0.4',
                    'select': ['oldest_detection,oldest_acquisition,types']
,
                    'token': APIkey}

    #the request:
    response = requests.get('https://app.ororatech.com/v1/clusters/',param
s=payload)

    #test request:
    #testpayload = {'xmin': '-117.86', 'ymin': '47.88', 'xmax': '-117.53',
'ymax': '48.00', 'minutes': '360', 'date': '2021-08-16-0200','confidence':
'0.5', 'token': APIkey}
    #response = requests.get('https://app.ororatech.com/v1/clusters/',para
ms=testpayload)

    data = response.json()
    #if data is not None does not help in case of an "empty" json
    #in that case, response.json() = {'type': 'FeatureCollection', 'featur
es': None}
    #would lead to an error, trying to create agdf from features
    if data != {'type': 'FeatureCollection', 'features': None}:
        #columns=['geometry', 'id', 'num_fires']
        gdf_local = gpd.GeoDataFrame.from_features(data)
    else:
        #else prepare an empty gdf for return
        gdf_local = gpd.GeoDataFrame()

    return gdf_local

series_of_gdfs = gdf_runfires_bybuff.apply(get_clusterPerAPI,axis=1)
list_of_gdfs= series_of_gdfs.tolist()

#concat returned gdfs; empty ones are not cosidered by default
gdf_current = gpd.GeoDataFrame(pd.concat(list_of_gdfs, ignore_index=True))

#gdf_current is a geodataframe, but crs has still to be specified
gdf_current = gdf_current.set_crs("EPSG:4326", allow_override=True)

gdf_current.head(2)
```

```
                                            geometry        id types  \
0  POLYGON ((-103.39699 41.50449, -103.39690 41.5...  19952554   [0]
1  POLYGON ((-103.38383 41.50191, -103.38270 41.5...  19952554   [0]

   num_fires           oldest_detection         oldest_acquisition
```

```
0        635  2021-08-06T22:53:25+00:00  2021-08-06T02:55:05+00:00
1        635  2021-08-07T18:53:00+00:00  2021-08-06T02:55:05+00:00
```

```
# field "types" contains list for Python. They need to be converted to str
ings,
# otherwise .duplicated() or turning to GeoJSON would not work!
gdf_current["types"] = gdf_current["types"].astype('string')
# remove [] to avoid confusion when this is read from GeoJSON again
gdf_current["types"] = gdf_current["types"].str.removeprefix("[")
gdf_current["types"] = gdf_current["types"].str.removesuffix("]")
```

```
gdf_current["types"].head(2)
```

```
0    0
1    0
Name: types, dtype: string
```

Buffer current fires

The API request results get buffered by specified (variable buf_size) amount of statute miles.

```
#prepare buffer with a metric CRS
gdf_current_buffered = gdf_current.copy()
gdf_current_buffered = gdf_current_buffered.to_crs("EPSG:2163")

#perform buffer by specified amount of statute miles (SM)
gdf_current_buffered["geometry"] = gdf_current_buffered.buffer(buf_size, r
esolution=16)

#turn CRS back
gdf_current_buffered = gdf_current_buffered.to_crs("EPSG:4326")
```

If there is an overlap between TFRs and API request results, then those TFRs can be considered inappropriate from an aerial firefighting perspective.

```
gdf_result = gpd.sjoin(gdf_current_buffered, gdf_tfrs, how='inner', predic
ate='overlaps')
```

```
gdf_result.head(2)
```

```
                                              geometry        id types  \
3  POLYGON ((-107.11377 36.94489, -107.11362 36.9...  20021038     0
3  POLYGON ((-107.11377 36.94489, -107.11362 36.9...  20021038     0

   num_fires           oldest_detection         oldest_acquisition  \
3        939  2021-08-09T23:13:52+00:00  2021-08-07T17:52:20+00:00
3        939  2021-08-09T23:13:52+00:00  2021-08-07T17:52:20+00:00

   index_right          Source Location NOTAM # Issue Date (UTC)  \
3           10  ZDV_2021-08-08      ZDV  1/3260  08/08/2021 1648
3           13  ZDV_2021-08-09      ZDV  1/3434  08/09/2021 0312

  Effective Date (UTC) Cancel Date (UTC) Expiration Date (UTC)  \
3      08/08/2021 1700   08/09/2021 0311       08/10/2021 0300
3      08/09/2021 1300   08/12/2021 1402       09/09/2021 0300
```

```
                      NOTAM Condition or LTA Subject  Radius  Radius_m
3  !FDC 1/3260 ZDV NM..AIRSPACE 47NM NE OF BLOOMF...     5.0    9260.0
3  !FDC 1/3434 ZDV NM..AIRSPACE 47NM NE OF BLOOMF...     5.0    9260.0
```

A TFR is also inappropriate, if it is entirely inside a fire, so the according check from above is reused

```
#check for buffered fire already covering entire TFR
gdf_currentcovering_bybuff = gpd.sjoin(gdf_current_buffered, gdf_tfrs, how
='inner', predicate='contains')

gdf_currentcovering_bybuff.head(2)
```

```
                                          geometry        id types  \
6  POLYGON ((-106.32305 40.16465, -106.32301 40.1...  20866464     0

   num_fires           oldest_detection         oldest_acquisition  \
6        587  2021-08-30T18:51:21+00:00  2021-08-29T19:31:23+00:00

   index_right          Source Location NOTAM # Issue Date (UTC)  \
6           18  ZDV_2021-08-30      ZDV  1/8134  08/30/2021 0201

  Effective Date (UTC) Cancel Date (UTC) Expiration Date (UTC)  \
6      08/30/2021 1400   09/07/2021 1405       10/29/2021 0200

                      NOTAM Condition or LTA Subject  Radius  Radius_m
6  !FDC 1/8134 ZDV CO..AIRSPACE 8NM NE OF KREMLIN...     3.0    5556.0
```

```
#concatenate dataframes if buffered fires contain an entire TFR
if len(gdf_currentcovering_bybuff.index)>0:
    frames = [gdf_result, gdf_currentcovering_bybuff]
    gdf_result = gpd.GeoDataFrame(pd.concat(frames, sort=False))

print(len(gdf_result.index))
```

```
5
```

Number of results: ZDV (3 SM): 11, ZDV (1 SM): 1

```
# Appending to logfile
with open(logfilename, 'a') as logfile:
    logfile.write('Number results for '+tfr+' '+str_buf_size+' SM: '+str(l
en(gdf_result.index))+'\n' )
```

ATTENTION: As joined once again with all TFRs in gdf_tfrs, those TFR cancelled before being effective will be included here! So get_minutes function is applied once more to recognize those. The Number of occurrences gets printed below. To obtain the true count (without duplicates), this must be applied on the input geodataframe gdf_tfrs

```
#apply function get_minutes from above again
gdf_tfrs["Duration_Minutes"]=gdf_tfrs.apply(get_minutes,axis=1)

print(len(gdf_tfrs[gdf_tfrs["Duration_Minutes"]<=0].index))
```

```
1
```

Number of TFR(s) cancelled before becoming effective: ZDV (3 SM): 1 ZDV (1 SM): 0

```
# Appending to logfile
with open(logfilename, 'a') as logfile:
    logfile.write('Number of TFR(s) cancelled before becoming effective fo
r '+tfr+' '+str_buf_size+' SM: '+str(len(gdf_tfrs[gdf_tfrs["Duration_Minut
es"]<=0].index))+'\n' )
```

```
gdf_result.head(2)
```

```
                                             geometry        id types  \
3  POLYGON ((-107.11377 36.94489, -107.11362 36.9...  20021038     0
3  POLYGON ((-107.11377 36.94489, -107.11362 36.9...  20021038     0

   num_fires           oldest_detection         oldest_acquisition  \
3        939  2021-08-09T23:13:52+00:00  2021-08-07T17:52:20+00:00
3        939  2021-08-09T23:13:52+00:00  2021-08-07T17:52:20+00:00

   index_right          Source Location NOTAM # Issue Date (UTC)  \
3           10  ZDV_2021-08-08      ZDV  1/3260  08/08/2021 1648
3           13  ZDV_2021-08-09      ZDV  1/3434  08/09/2021 0312

  Effective Date (UTC) Cancel Date (UTC) Expiration Date (UTC)  \
3      08/08/2021 1700   08/09/2021 0311       08/10/2021 0300
3      08/09/2021 1300   08/12/2021 1402       09/09/2021 0300

                     NOTAM Condition or LTA Subject  Radius  Radius_m
3  !FDC 1/3260 ZDV NM..AIRSPACE 47NM NE OF BLOOMF...     5.0    9260.0
3  !FDC 1/3434 ZDV NM..AIRSPACE 47NM NE OF BLOOMF...     5.0    9260.0
```

Result may contain duplicates, due to multi join and fires overlapping TFRs more than one time. Resulting log gets a count of entirely duplicate rows, the 'Number of TFRs where a fire leaves the TFR' and the 'Number of fire events leaving a TFR'

```
#count duplicates
gdf_result.duplicated().sum()
```

```
0
```

```
#count non-duplicates
(~gdf_result.duplicated()).sum()
```

```
5
```

```
# Appending to logfile
with open(logfilename, 'a') as logfile:
    logfile.write('Number of results without duplicates for '+tfr+' '+str_
buf_size+' SM: '+str((~gdf_result.duplicated()).sum())+'\n' )
```

```
# count non-duplicates of TFRs
(~gdf_result.duplicated(["NOTAM #"])).sum()
```

```
5
```

```
# Appending to logfile
with open(logfilename, 'a') as logfile:
    logfile.write('Number of TFRs where a fire leaves the TFR for '+tfr+'
```

```
'+str_buf_size+' SM: '+str((~gdf_result.duplicated(["NOTAM #"])).sum())+'\
n' )

# count non-duplicates of fires
(~gdf_result.duplicated(["id"])).sum()
```

```
3
```

```
# Appending to logfile
with open(logfilename, 'a') as logfile:
    logfile.write('Number of fire events leaving a TFR for '+tfr+' '+str_b
uf_size+' SM: '+str((~gdf_result.duplicated(["id"])).sum())+'\n'+'\n' )

# Considering certain columns for dropping duplicates, if wanted
#gdf_result.drop_duplicates(subset=['id', 'NOTAM #'])

# Drop entire row duplicates, if wanted
#gdf_result.drop_duplicates()
```

Finally, compose an output file for those fires that had inappropriate TFRs, IF any

```
# GeoJSON output
if len(gdf_result.index)>0:
    gdf_result.to_file(filename= outfile, driver='GeoJSON')
```

```
D:\anaconda3\envs\master_env\lib\site-packages\geopandas\io\file.py:362: F
utureWarning: pandas.Int64Index is deprecated and will be removed from pan
das in a future version. Use pandas.Index with the appropriate dtype inste
ad.
  pd.Int64Index,
```

This Notebook ends here. Follow up analysis is performed Get_Events_from_Fires_from_API to avoid rerunning API requests when this is not necessary.

Code 10 Get_TFR-exceeding_Fires_from_API.ipynb

A Script to gain lists of TFR-exceeding fire clusters that can be revisited

```
import geopandas as gpd
import pandas as pd

##specify file location and name
#3 letter location indicator, reused within all created files
#possible values: ZAB ZDV ZFW ZLA ZLC ZMP ZOA ZSE; not ZKC and ZHU having
no wildfire TFRs, ZAB has no overlapping fires
tfr =r"ZDV"
#paths and filenames
in_path=r"D:\UNIGIS\MASTER\Scripts\Exceeding\\"
out_path=in_path


#size used by prior script to buffer fires in m; 1609.344  m = 1 SM, 4828.
032 m = 3 SM
buf_size = 4828.032

#output concatenates from these strings as well. Buffer size must be a str
ing for that as well
```

```
str_buf_size = str(int(buf_size/1609.344))

#input filename
infile = tfr+'_'+str_buf_size+'SM_runaway_fires.geojson'

#read geojson files
gdf_in = gpd.read_file(in_path + infile)

#output
result_fires = out_path+tfr+'_'+str_buf_size+'SM_result-fires.geojson'
result_TFRs = out_path+tfr+'_'+str_buf_size+'SM_result-TFRs.geojson'

#result lists filename
result_list_fires = out_path+tfr+'_'+str_buf_size+'SM_fires_results_list.c
sv'
result_list_TFRs = out_path+tfr+'_'+str_buf_size+'SM_TFRs_results_list.csv
'
```

Either duplicate fire clusters or TFRs must be dropped to obtain a results list

```
# Considering  column for dropping duplicate fire clusters
gdf_result_fires = gdf_in.drop_duplicates(subset=['id'])

# Considering  column for dropping duplicate TFRs
gdf_result_TFRs = gdf_in.drop_duplicates(subset=['NOTAM #'])

# GeoJSON output fires
gdf_result_fires.to_file(filename= result_fires, driver='GeoJSON')
```

```
D:\anaconda3\envs\master_env\lib\site-packages\geopandas\io\file.py:362: F
utureWarning: pandas.Int64Index is deprecated and will be removed from pan
das in a future version. Use pandas.Index with the appropriate dtype inste
ad.
  pd.Int64Index,
```

```
# GeoJSON output TFRs
gdf_result_TFRs.to_file(filename= result_TFRs, driver='GeoJSON')
```

```
D:\anaconda3\envs\master_env\lib\site-packages\geopandas\io\file.py:362: F
utureWarning: pandas.Int64Index is deprecated and will be removed from pan
das in a future version. Use pandas.Index with the appropriate dtype inste
ad.
  pd.Int64Index,
```

Result lists are written as single files per TFR set issued from one ARTCC

```
gdf_result_fires['id'].to_csv(result_list_fires, header=['Result List of F
ire IDs for '+tfr], index=None, sep=' ', mode='w')

gdf_result_TFRs['NOTAM #'].to_csv(result_list_TFRs, header=['Result List o
f TFRs for '+tfr], index=None, sep=' ', mode='w')
```

This Notebook ends here. A check wheter static/artificial fire types were included can be done by Get_Types_from_exceeding_Fires.ipynb

Code 11 Get_Events_from_Fires_from_API.ipynb

A Script to check types of TFR-exceeding fire clusters for artificial sources

```
import geopandas as gpd
import pandas as pd

##specify file location and name
#3 letter location indicator, reused within all created files
#possible values: ZDV ZFW ZLA ZLC ZMP ZOA ZSE; not ZKC and ZHU having no w
ildfire TFRs, ZAB has no overlapping fires
tfr =r"ZDV"
#paths and filenames
in_path=r"D:\UNIGIS\MASTER\Scripts\Exceeding\\"

out_path=in_path

#size used by prior script to buffer fires in m; 1609.344  m = 1 SM, 4828.
032 m = 3 SM
buf_size = 4828.032

#output concatenates from these strings as well. Buffer size must be a str
ing for that as well
str_buf_size = str(int(buf_size/1609.344))

#input filename
infile = in_path+tfr+'_'+str_buf_size+'SM_result-fires.geojson'
infile_bytfr = in_path+tfr+'_'+str_buf_size+'SM_result-TFRs.geojson'

#read geojson files
gdf_in = gpd.read_file(infile)
gdf_in_bytfr = gpd.read_file(infile_bytfr)

#output
#none, just print(len(gdf_wrtypes.index))
```

Task is to find rows in results sets, that have an unwanted type in the origin data (compare Table 1 from the thesis)

```
# specify a types series
#correct_types = ['0','1','5','6','7','8']
wrong_types = ['2','3','4','9','10','11','12','13','14','15','16','17','18
','19']

# search for wrong types in results
gdf_wrtypes = gdf_in[gdf_in['types'].isin(wrong_types)]

print(len(gdf_wrtypes.index))

0

gdf_wrtypes_tfr = gdf_in_bytfr[gdf_in_bytfr['types'].isin(wrong_types)]

print(len(gdf_wrtypes_tfr.index))

0
```

This Notebook ends here.

Code 12 Get_Types_from_exceeding_Fires.ipynb

8.2.6. Relating 4.2.6, Safety of actual Fire Fighting Aircraft

Get_Dates_Aircraft_and_TFRs.ipynb (Code 13) handles fire related aircraft movements and TFR times.

A script to compare trajectory datetime with TFR times

from an already joined dataset

```
import geopandas as gpd
import pandas as pd
import datetime as dt

#create geodataframe with file from script´s directory
gdf = gpd.read_file('TFRs_on_Trajectories.geojson')


#out files as GeoJSON within scrip directory
file_traj_noTFR = 'traj_not_in_TFR.geojson'
file_trai_not_in_TFR_time = 'traj_not_in_TFR_time.geojson'

gdf.head(2)
```

```
   OBJECTID  Join_Count  TARGET_FID  JOIN_FID   traj_id              start
_t  \
0         1           1           1       251  a02862_0  2021-08-01T01:19:
23
1         2           1           1       255  a02862_0  2021-08-01T01:19:
23

                 end_t       length  direction          Source  ... NOTAM_
_  \
0  2021-08-01T01:27:32  8685.843986  39.598238  ZOA_2021-08-01  ...  1/886
9
1  2021-08-01T01:27:32  8685.843986  39.598238  ZOA_2021-08-03  ...  1/001
4

  Issue_Date__UTC_ Effective_Date__UTC_ Cancel_Date__UTC_  \
0  07/31/2021 0000      07/31/2021 0015   08/03/2021 1350
1  08/03/2021 0201      08/03/2021 1400   08/04/2021 0359

  Expiration_Date__UTC_                     NOTAM_Condition_or_LTA_Subject
\
0       10/01/2021 0015  !FDC 1/8869 ZOA CA..AIRSPACE 24NM E OF CHICO, ...
1       09/03/2021 0400  !FDC 1/0014 ZOA CA..AIRSPACE 24NM E OF CHICO, ...

  Radius  Radius_m  Shape_Length  \
0    0.0       0.0      0.091049
1    0.0       0.0      0.091049

                                             geometry
0  LINESTRING (-121.10220 39.91300, -121.10920 39...
```

```
1  LINESTRING (-121.10220 39.91300, -121.10920 39...

[2 rows x 21 columns]
```

Those aircraft movements having no TFR cover are sorted out

```
# this cannot have duplicates
gdf_noTFR = gdf[gdf["Join_Count"]==0]

# this can have duplicates in case of multiple TFRs being touched
# simultaneously or multiple TFRs being issued in the same place
gdf_hasTFR = gdf[gdf["Join_Count"]!=0]

gdf_noTFR.head(2)
```

```
     OBJECTID  Join_Count  TARGET_FID  JOIN_FID    traj_id  \
889       890           0          71        -1  a0956b_10
934       935           0          74        -1  a0956b_13

                 start_t                 end_t        length   direction  \
889  2021-09-02T22:58:49  2021-09-02T23:02:42  14191.218388   43.931943
934  2021-09-22T22:06:19  2021-09-22T22:14:54  45260.857727  282.981352

    Source  ... NOTAM__ Issue_Date__UTC_ Effective_Date__UTC_  \
889   None  ...    None             None                 None
934   None  ...    None             None                 None

    Cancel_Date__UTC_ Expiration_Date__UTC_ NOTAM_Condition_or_LTA_Subject
\
889              None                  None                            None
934              None                  None                            None

    Radius  Radius_m  Shape_Length  \
889    NaN       NaN      0.147856
934    NaN       NaN      0.471762

                                              geometry
889  LINESTRING (-116.78530 48.28520, -116.76590 48...
934  LINESTRING (-122.32240 38.31120, -122.33190 38...

[2 rows x 21 columns]
```

This many aircraft movements are not covered by TFRs:

```
# row count of gdf_noTFR
print(len(gdf_noTFR.index))
```

```
86
```

```
# just a check: Count non-duplicates (should be 555 here to end up at 555+
86 = 641 aircraft movements)
(~gdf_hasTFR.duplicated(["traj_id"])).sum()
```

```
555
```

A trajectories start_t must be within a TFR´s effective time to be covered for sure. That needs calculation per row

```
gdf_gap = gdf_hasTFR

#get matching TFR
#time formats are all naive but all are known for being UTC
def get_tfrgap(row):

    tfrgap = 'not calculated'

    enddate = row["Cancel_Date__UTC_"]
    if enddate == None:
        enddate = row["Expiration_Date__UTC_"]
        #if both enddate columns become read, a SettingWithCopyWarning occ
urs, which is ok
    startdate = row["Effective_Date__UTC_"]

    traj_startdate= row["start_t"]
    traj_enddate= row["end_t"]

    #convert to datetime to calculate
    enddate_dt = pd.to_datetime(enddate)
    startdate_dt = pd.to_datetime(startdate)
    traj_startdate_dt = pd.to_datetime(traj_startdate)
    traj_enddate_dt = pd.to_datetime(traj_enddate)

    # if trajectory time was within TFR time...
    if startdate_dt <= traj_startdate_dt and enddate_dt >= traj_enddate_dt
:

        #calculate a time gap in minutes
        #this way, a negative value describes TFR being OK, the only case
considered here:
        calcdate= startdate_dt - traj_startdate_dt
        minutes = calcdate.total_seconds()/60
        minutes = int(minutes)
        # need sortable type compared to str 'not calculated':
        minutes = str(minutes)
        tfrgap = minutes

    return tfrgap

gdf_gap["TFRgap"]=gdf_hasTFR.apply(get_tfrgap,axis=1)
```

```
D:\anaconda3\envs\master_env\lib\site-packages\geopandas\geodataframe.py:1
351: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  super().__setitem__(key, value)
```

```
gdf_gap.head(2)
```

```
   OBJECTID  Join_Count  TARGET_FID  JOIN_FID   traj_id              start
_t  \
0         1           1           1       251  a02862_0  2021-08-01T01:19:
23
1         2           1           1       255  a02862_0  2021-08-01T01:19:
23

                 end_t       length  direction          Source  ...  \
0  2021-08-01T01:27:32  8685.843986  39.598238  ZOA_2021-08-01  ...
1  2021-08-01T01:27:32  8685.843986  39.598238  ZOA_2021-08-03  ...

  Issue_Date__UTC_ Effective_Date__UTC_ Cancel_Date__UTC_  \
0  07/31/2021 0000      07/31/2021 0015   08/03/2021 1350
1  08/03/2021 0201      08/03/2021 1400   08/04/2021 0359

  Expiration_Date__UTC_                     NOTAM_Condition_or_LTA_Subject
\
0       10/01/2021 0015  !FDC 1/8869 ZOA CA..AIRSPACE 24NM E OF CHICO, ...
1       09/03/2021 0400  !FDC 1/0014 ZOA CA..AIRSPACE 24NM E OF CHICO, ...

  Radius Radius_m  Shape_Length  \
0    0.0      0.0      0.091049
1    0.0      0.0      0.091049

                                              geometry          TFRgap
0  LINESTRING (-121.10220 39.91300, -121.10920 39...           -1504
1  LINESTRING (-121.10220 39.91300, -121.10920 39...  not calculated

[2 rows x 22 columns]
```

With the time gap available where possible now, those trajectories (trai_id) need to get removed, where another occurrence of it has a value. Not till then the duplicates can get removed to count trajectories and thus aircraft movements that were not under TFR coverage.

```
# change the global options that Geopandas inherits from if more rows/colu
mns shall be displayed
#pd.set_option('display.max_rows',None)

#gdf gets sorted and
#duplicates must be dropped immediately before splitting the sorted gdf
gdf_gap_sorted = gdf_gap.sort_values(["TFRgap"], ascending = True).drop_du
plicates(["traj_id"])

#should be 555 here to end up at 555+86 = 641 aircraft movements,
#unique again but tagged with a minute value for TFRs being OK
print(len(gdf_gap_sorted.index))
```

```
555
```

```
#splitting the sorted gdf
gdf_gap_sorted_hascalc = gdf_gap_sorted[gdf_gap_sorted["TFRgap"]!='not cal
culated']
gdf_gap_sorted_nocalc = gdf_gap_sorted[gdf_gap_sorted["TFRgap"]=='not calc
ulated']
```

```
# Trajectories within TFR timeframe
print('Trajectories within TFR timeframe: Count is '+str(len(gdf_gap_sorte
d_hascalc.index)))

Trajectories within TFR timeframe: Count is 439

# Trajectories out of TFR timeframe
print('Trajectories out of TFR timeframe: Count is '+str(len(gdf_gap_sorte
d_nocalc.index)))

Trajectories out of TFR timeframe: Count is 116

#create geojson of questionable trajectories
 # Trajectories outside of any TFR
gdf_noTFR.to_file(filename = file_traj_noTFR, driver='GeoJSON')
 # Trajectories out of TFR timeframe
gdf_gap_sorted_nocalc.to_file(filename = file_trai_not_in_TFR_time, driver
='GeoJSON')

D:\anaconda3\envs\master_env\lib\site-packages\geopandas\io\file.py:362: F
utureWarning: pandas.Int64Index is deprecated and will be removed from pan
das in a future version. Use pandas.Index with the appropriate dtype inste
ad.
  pd.Int64Index,
D:\anaconda3\envs\master_env\lib\site-packages\geopandas\io\file.py:362: F
utureWarning: pandas.Int64Index is deprecated and will be removed from pan
das in a future version. Use pandas.Index with the appropriate dtype inste
ad.
  pd.Int64Index,
```

By how much TFRs for trajectories out of TFR timeframe were late cannot be said. From the gdf_gap_sorted containing duplicates it is not (always) known, which TFR was intended for that particular flight. So the script ends here.

Code 13 Get_Dates_Aircraft_and_TFRs.ipynb

8.2.7. Relating 4.2.7 Completeness of TFR-Fire-Correspondence

Compare_TFRs_to_Fireclusters.ipynb () handles TFRs that do not intersect a three-month-dataset firecluster.

A script to compare TFRs to fireclusters at different confidence

```
import geopandas as gpd
import pandas as pd

#create geodataframe with files from script´s directory


#paths and filenames
fire_path = r"D:\UNIGIS\MASTER\DownloadedData\WFS\\"
tfr_path=r"D:\UNIGIS\MASTER\Scripts\\"

fire_file = "wfs-area-export_FIRs_Boundary_08-102021_con_pt5.geojson"
# a second GeoJSON containing all fireclusters, also of low confidence, is
needed
fire_file_low_conf = "wfs-area-export_2021-08-01-2021-10-31_FIRs_Boundary_
```

```
08-102021.geojson"

tfr_file = "All_TFRs.geojson"

#read geojson files
gdf_fires_low = gpd.read_file(fire_path + fire_file_low_conf)

gdf_All_TFRs = gpd.read_file(tfr_path + tfr_file)
gdf_fires = gpd.read_file(fire_path + fire_file)

#output filename (if any):
outfile = 'TFRs_without_fires.geojson'

# left join with predicate intersect
gdf_TFR_nofire = gpd.sjoin(gdf_All_TFRs, gdf_fires, how='left', predicate=
'intersects')

# change the global options that Geopandas inherits from if more shall be
displayed
#pd.set_option('display.max_columns',None)
#pd.set_option('display.max_rows',None)

gdf_TFR_nofire = gdf_TFR_nofire[gdf_TFR_nofire["id"].isnull()]

print(len(gdf_TFR_nofire.index))

56

#gdf_TFR_nofire
```

Sort out eventually erroneous TFRs, cancelled before becoming effective, maybe because they were issued in the wrong place

```
gdf_TFR_nofire_err = gdf_TFR_nofire[ pd.to_datetime(gdf_TFR_nofire["Cancel
_Date__UTC_"]) < pd.to_datetime(gdf_TFR_nofire["Effective_Date__UTC_"]) ]

print(len(gdf_TFR_nofire_err.index))

0
```

Sort out TFRs issued before fire cluster timeframe (starting 08/01/2021 0000). Their comparison may suffer from a boundary value problem

```
gdf_TFR_nofire_old = gdf_TFR_nofire[ pd.to_datetime(gdf_TFR_nofire["Issue_
Date__UTC_"]) < pd.to_datetime('08/01/2021 0000') ]

print(len(gdf_TFR_nofire_old.index))

17

gdf_TFR_nofire = gdf_TFR_nofire[ pd.to_datetime(gdf_TFR_nofire["Issue_Date
__UTC_"]) >= pd.to_datetime('08/01/2021 0000') ]
```

The remaining TFRs are now joined on fire clusters with a lower confidence

```
#If it is ever needed to join a join result again, index/ other needed col
umns need to get renamend
gdf_TFR_nofire.rename(columns = {'index_right':'old_index_right'}, inplace
```

```
= True)
gdf_TFR_nofire.rename(columns = {'id':'old_id'}, inplace = True)
# left join with predicate intersect
gdf_TFR_nofire_low = gpd.sjoin(gdf_TFR_nofire, gdf_fires_low, how='left',
predicate='intersects')

#gdf_TFR_nofire_low

gdf_TFR_nofire_low = gdf_TFR_nofire_low[gdf_TFR_nofire_low["id"].isnull()]

print(len(gdf_TFR_nofire_low.index))

3

gdf_TFR_nofire_low

     OBJECTID          Source Location NOTAM__ Issue_Date__UTC_  \
23         24  ZSE_2021-08-01      ZSE  1/9095  08/01/2021 1555
540       541  ZAB_2021-10-20      ZAB  1/5987  10/20/2021 2254
541       542  ZAB_2021-10-21      ZAB  1/6009  10/21/2021 0034

    Effective_Date__UTC_ Cancel_Date__UTC_ Expiration_Date__UTC_  \
23       08/01/2021 1615   08/01/2021 2337       10/01/2021 0500
540      10/20/2021 2245   10/21/2021 0229       10/22/2021 0230
541      10/21/2021 1400   10/22/2021 2316       11/21/2021 0300

                        NOTAM_Condition_or_LTA_Subject  Radius  ...  \
23   !FDC 1/9095 ZSE OR..AIRSPACE 22NM S MEDFORD, O...       3  ...
540  !FDC 1/5987 ZAB AZ..AIRSPACE 23NM SE OF TUCSON...       3  ...
541  !FDC 1/6009 ZAB AZ..AIRSPACE 23NM SE OF TUCSON...       3  ...

     index_right  id  age_right area_right  num_fires_right  confidence_ri
ght  \
23           NaN NaN        NaN        NaN              NaN
NaN
540          NaN NaN        NaN        NaN              NaN
NaN
541          NaN NaN        NaN        NaN              NaN
NaN

     newest_detection_right  oldest_detection_right newest_acquisition_rig
ht  \
23                      NaN                     NaN                      N
aN
540                     NaN                     NaN                      N
aN
541                     NaN                     NaN                      N
aN

     oldest_acquisition_right
23                        NaN
540                       NaN
541                       NaN

[3 rows x 35 columns]
```

```
# GeoJSON output
if len(gdf_TFR_nofire_low.index)>0:
    gdf_TFR_nofire_low.to_file(filename= outfile, driver='GeoJSON')
```

```
D:\anaconda3\envs\master_env\lib\site-packages\geopandas\io\file.py:362: F
utureWarning: pandas.Int64Index is deprecated and will be removed from pan
das in a future version. Use pandas.Index with the appropriate dtype inste
ad.
  pd.Int64Index,
```

Code 14 Compare_TFRs_to_Fireclusters.ipynb

8.3. Tables containing Time Gap between Fire Detection and TFR Issue

Tables listing the calculated timespans for 5.1 are shown here. Whether all fire ids could become tied to a TFR for an FIR is mentioned within the according table caption. Here, the tables are sorted by “Timespan Acquisition”.

Table 8 Calculated timespans in minutes since fire detection and acquisition until TFR issue date for FIR Denver (KZDV). 1 fire cluster could not get tied to a TFR (9 clusters are with 27 TFRs being considered).

id	Fire type	Fire detection	Fire acquisition	NOTAM # from ZDV	Issue Date (UTC)	Timespan Detection [min]	Timespan Acquisition [min]
20177190	[1]	11.08.2021 23:25	11.08.2021 21:00	1/5720	08/11/2021 2341	15	160
21594740	[0]	17.09.2021 03:59	16.09.2021 19:15	1/9881	09/17/2021 0039	-200	323
20866464	[0]	29.08.2021 22:24	29.08.2021 19:31	1/8134	08/30/2021 0201	216	389
20110185	[1]	10.08.2021 22:06	10.08.2021 09:05	1/4715	08/10/2021 2143	-23	757
21978529	[0]	28.09.2021 02:56	28.09.2021 01:05	1/4157	09/28/2021 1441	704	815
19952554	[0]	06.08.2021 18:05	06.08.2021 02:55	1/2974	08/06/2021 1847	41	951
21595812	[0]	16.09.2021 23:51	16.09.2021 23:45	1/0214	09/17/2021 1854	1142	1148
20021038	[0]	08.08.2021 04:22	07.08.2021 17:52	1/3260	08/08/2021 1648	745	1375
19806634	[1]	04.08.2021 22:24	02.08.2021 03:09	1/1585	08/05/2021 0019	114	4149

Table 9 Calculated timespans in minutes since fire detection and acquisition until TFR issue date for FIR Ft Worth (KZFW). 0 fire clusters could not get tied to a TFR (1 cluster is with 3 TFRs being considered).

id	Fire type	Fire detection	Fire acquisition	NOTAM # from ZFW	Issue Date (UTC)	Timespan Detection [min]	Timespan Acquisition [min]
22972697	[0]	29.10.2021 03:24	28.10.2021 19:53	1/9732	10/29/2021 2222	1137	1588

Table 10 Calculated timespans in minutes since fire detection and acquisition until TFR issue date for FIR Los Angeles (KZLA). 6 fire clusters could not get tied to a “first” TFR (15 clusters are with 39 TFRs being considered).

id	Fire type	Fire detection	Fire acquisition	NOTAM # from ZLA	Issue Date (UTC)	Timespan Detection [min]	Timespan Acquisition [min]
20441640	[0]	19.08.2021 04:46	19.08.2021 00:05	1/1505	08/19/2021 0122	-204	76
21628656	[0]	17.09.2021 23:21	17.09.2021 20:55	1/0276	09/17/2021 2219	-62	83
20828697	[1]	28.08.2021 22:38	28.08.2021 19:45	1/7865	08/28/2021 2130	-68	104
21400591	[0]	11.09.2021 23:10	11.09.2021 23:05	1/6465	09/12/2021 0053	102	107
20866547	[0]	29.08.2021 22:34	29.08.2021 21:11	1/8106	08/30/2021 0024	109	192
20707731	[0]	25.08.2021 23:06	25.08.2021 20:45	1/5793	08/26/2021 0032	85	226
22238197	[0]	05.10.2021 20:43	05.10.2021 13:25	1/8397	10/05/2021 1743	-180	257
21259974	[0]	08.09.2021 12:30	08.09.2021 10:03	1/3392	09/08/2021 1948	437	584
21259975	[0]	08.09.2021 12:30	08.09.2021 10:03	1/3392	09/08/2021 1948	437	584
22441864	[5]	12.10.2021 05:58	11.10.2021 21:04	1/2241	10/12/2021 1408	489	1023
21629143	[0]	18.09.2021 00:50	17.09.2021 21:55	1/0412	09/18/2021 1730	999	1174
20294555	[0]	16.08.2021 05:01	15.08.2021 09:12	1/8642	08/16/2021 0502	0	1189
21325525	[0]	10.09.2021 05:21	10.09.2021 02:35	1/6340	09/11/2021 0155	1233	1399
20600062	[0]	26.08.2021 12:32	23.08.2021 09:13	1/5793	08/26/2021 0032	-720	3798
21370680	[1]	04.10.2021 23:03	10.09.2021 17:35	1/8397	10/05/2021 1743	1119	36007

Table 11 Calculated timespans in minutes since fire detection and acquisition until TFR issue date for FIR Salt Lake (KZLC). 55 fire clusters could not get tied to a “first” TFR (51 clusters are with 94 TFRs being considered).

id	Fire type	Fire detection	Fire acquisition	NOTAM # from ZLC	Issue Date (UTC)	Timespan Detection [min]	Timespan Acquisition [min]
21637388	[0]	18.09.2021 12:41	18.09.2021 10:13	1/9128	08/16/2021 1925	-47116	-46968
20143507	[0]	11.08.2021 04:03	10.08.2021 23:45	1/4559	08/10/2021 2021	-462	-204
21662781	[0]	19.09.2021 03:53	19.09.2021 03:17	1/0458	09/19/2021 0025	-208	-172
19911782	[0]	05.08.2021 04:26	05.08.2021 03:48	1/1636	08/05/2021 0235	-111	-73
20282034	[1]	14.08.2021 22:17	14.08.2021 19:35	1/8240	08/14/2021 2112	-65	96

id	Fire type	Fire detection	Fire acquisition	NOTAM # from ZLC	Issue Date (UTC)	Timespan Detection [min]	Timespan Acquisition [min]
22210154	[0]	04.10.2021 22:46	04.10.2021 19:58	1/7919	10/04/2021 2256	9	177
20140648	[0]	11.08.2021 02:23	10.08.2021 21:19	1/4779	08/11/2021 0026	-117	186
21159884	[0]	05.09.2021 22:24	05.09.2021 19:51	1/1940	09/05/2021 2319	54	207
19800775	[0]	01.08.2021 19:21	01.08.2021 17:05	1/9211	08/01/2021 2033	71	207
20147669	[0]	11.08.2021 10:37	11.08.2021 07:54	1/5007	08/11/2021 1133	55	218
21978554	[0]	28.09.2021 02:57	27.09.2021 22:05	1/3892	09/28/2021 0312	15	306
21119828	[0]	04.09.2021 23:59	04.09.2021 21:00	1/1740	09/05/2021 0211	131	310
20911975	[0]	30.08.2021 21:52	30.08.2021 21:45	1/8891	08/31/2021 0317	324	331
20247609	[1]	13.08.2021 22:18	13.08.2021 19:33	1/8070	08/14/2021 0126	187	352
20150354	[1]	11.08.2021 12:13	11.08.2021 09:35	1/5286	08/11/2021 1639	265	423
21556189	[0]	15.09.2021 22:15	15.09.2021 19:14	1/9243	09/16/2021 0222	246	427
22209721	[0]	04.10.2021 21:32	04.10.2021 18:12	1/7973	10/05/2021 0358	386	585
20350209	[0]	16.08.2021 18:46	16.08.2021 17:20	1/9450	08/17/2021 0348	541	627
19882731	[0]	04.08.2021 06:28	04.08.2021 05:49	1/1237	08/04/2021 1719	650	690
20362480	[1]	17.08.2021 11:04	17.08.2021 07:35	1/0288	08/17/2021 1907	482	691
20067398	[0]	09.08.2021 10:55	09.08.2021 08:31	1/3834	08/09/2021 2017	561	705
20099614	[1]	10.08.2021 07:15	10.08.2021 04:35	1/4482	08/10/2021 1850	694	854
20140219	[0]	11.08.2021 01:36	10.08.2021 20:29	1/5006	08/11/2021 1129	592	899
21635512	[0]	18.09.2021 11:58	18.09.2021 09:22	1/0478	09/19/2021 0143	824	980
21635505	[0]	18.09.2021 11:58	18.09.2021 09:22	1/0478	09/19/2021 0143	824	980
19745379	[0]	31.07.2021 12:01	31.07.2021 08:50	1/9057	08/01/2021 0204	842	1033
20030384	[1]	08.08.2021 04:22	08.08.2021 02:25	1/3360	08/08/2021 2223	1080	1197
20282666	[1]	15.08.2021 04:19	14.08.2021 20:54	1/8455	08/15/2021 1713	773	1218
19878954	[0]	04.08.2021 05:49	03.08.2021 20:10	1/1237	08/04/2021 1719	689	1268
20065134	[1]	09.08.2021 07:48	09.08.2021 04:50	1/4010	08/10/2021 0235	1126	1304
20356189	[0]	17.08.2021 11:04	16.08.2021 20:17	1/0248	08/17/2021 1846	461	1348
20356196	[1]	17.08.2021 10:24	16.08.2021 20:16	1/0247	08/17/2021 1845	500	1348
20060094	[5]	10.08.2021 01:28	08.08.2021 19:26	1/3834	08/09/2021 2017	-311	1490
21322854	[0]	10.09.2021 08:20	09.09.2021 21:06	1/6335	09/11/2021 0007	946	1620
20257260	[0]	13.08.2021 05:02	12.08.2021 20:41	1/8052	08/14/2021 0013	1150	1651
21119197	[0]	06.09.2021 02:22	04.09.2021 20:10	1/1965	09/06/2021 0245	22	1834
21512433	[0]	15.09.2021 05:17	14.09.2021 19:33	1/9243	09/16/2021 0222	1264	1848
21205055	[0]	07.09.2021 11:10	06.09.2021 17:15	1/2986	09/08/2021 0157	886	1961
22216454	[0]	05.10.2021 21:15	05.10.2021 08:13	1/9447	10/06/2021 1812	1256	2038
20184128	[0]	16.08.2021 05:18	12.08.2021 09:16	1/8052	08/14/2021 0013	-3185	2337
19811564	[0]	03.08.2021 11:05	02.08.2021 09:03	1/0786	08/04/2021 0158	892	2454
20099609	[0]	11.08.2021 12:13	09.08.2021 02:25	1/5006	08/11/2021 1129	-44	3423
21565339	[0]	18.09.2021 11:01	16.09.2021 09:59	1/0458	09/19/2021 0025	803	3745
20150353	[0]	13.08.2021 04:22	11.08.2021 09:34	1/8052	08/14/2021 0013	1190	3758
20958865	[0]	04.09.2021 12:02	01.09.2021 08:51	1/1738	09/05/2021 0210	847	5358
21159882	[1]	09.09.2021 04:57	05.09.2021 18:29	1/4755	09/09/2021 1600	662	5610
20911204	[0]	04.09.2021 05:05	31.08.2021 04:02	1/1740	09/05/2021 0211	1265	7088
20837921	[0]	04.09.2021 05:05	29.08.2021 09:47	1/1740	09/05/2021 0211	1265	9623
20874515	[0]	05.09.2021 05:25	28.08.2021 18:19	1/1738	09/05/2021 0210	-195	10550
19912174	[0]	11.08.2021 12:53	03.08.2021 18:19	1/5005	08/11/2021 1127	-86	11107
20027972	[0]	15.08.2021 23:26	07.08.2021 21:25	1/1968	09/06/2021 0415	30528	42169

Table 12 Calculated timespans in minutes since fire detection and acquisition until TFR issue date for FIR Minneapolis (KZMP). 0 fire clusters could not get tied to a "first" TFR (4 clusters are with 10 TFRs being considered).

id	Fire type	Fire detection	Fire acquisition	NOTAM # from ZMP	Issue Date (UTC)	Timespan Detection [min]	Timespan Acquisition [min]
20319754	[1]	16.08.2021 02:19	15.08.2021 20:35	1/8526	08/15/2021 2326	-173	170
20319753	[0]	16.08.2021 02:19	15.08.2021 20:35	1/8526	08/15/2021 2326	-173	170
20316730	[0]	15.08.2021 20:35	15.08.2021 18:05	1/8533	08/16/2021 0001	205	355
20286067	[0]	15.08.2021 00:11	14.08.2021 21:25	1/8419	08/15/2021 1547	935	1101

Table 13 Calculated timespans in minutes since fire detection and acquisition until TFR issue date for FIR Oakland (KZOA). 29 fire clusters could not get tied to a "first" TFR (50 clusters are with 94 TFRs being considered).

id	Fire type	Fire detection	Fire acquisition	NOTAM # from ZOA	Issue Date (UTC)	Timespan Detection [min]	Timespan Acquisition [min]
20717535	[0]	26.08.2021 23:45	26.08.2021 09:55	1/5900	08/26/2021 0159	-1306	-476
20063924	[0]	06.08.2021 12:11	06.08.2021 09:30	1/2531	08/06/2021 0354	-497	-336
21595202	[0]	17.09.2021 04:34	17.09.2021 03:56	1/9858	09/16/2021 2359	-275	-237
20526911	[0]	21.08.2021 12:26	21.08.2021 09:49	1/2970	08/21/2021 0604	-382	-225
20218430	[0]	13.08.2021 12:17	12.08.2021 06:13	1/5863	08/12/2021 0347	-1950	-146
20710676	[0]	26.08.2021 04:48	26.08.2021 04:04	1/5900	08/26/2021 0159	-169	-125
20793006	[0]	27.08.2021 23:18	27.08.2021 18:49	1/7512	08/27/2021 1855	-263	5
20520282	[0]	21.08.2021 04:57	21.08.2021 04:08	1/2970	08/21/2021 0604	66	115
19917553	[0]	05.08.2021 12:30	05.08.2021 09:49	1/1948	08/05/2021 1326	55	216
20949700	[0]	31.08.2021 23:59	31.08.2021 21:24	1/9576	09/01/2021 0234	154	309
19909479	[0]	05.08.2021 00:03	04.08.2021 21:30	1/1637	08/05/2021 0247	163	316
19876809	[0]	03.08.2021 20:48	03.08.2021 19:54	1/0787	08/04/2021 0212	323	377
19949723	[0]	05.08.2021 23:28	05.08.2021 21:11	1/2531	08/06/2021 0354	265	402
21012977	[0]	02.09.2021 12:58	02.09.2021 11:04	1/0944	09/02/2021 1859	360	474
21160201	[0]	06.09.2021 05:03	05.09.2021 20:40	1/1974	09/06/2021 0536	32	535
20290094	[1]	15.08.2021 06:01	15.08.2021 05:19	1/8489	08/15/2021 1838	756	798
20763669	[0]	27.08.2021 12:14	27.08.2021 05:24	1/7512	08/27/2021 1855	400	810
21295112	[0]	09.09.2021 12:14	09.09.2021 09:43	1/5326	09/09/2021 2356	701	852
19816973	[0]	02.08.2021 12:38	02.08.2021 10:45	1/0014	08/03/2021 0201	802	915
20916324	[0]	31.08.2021 11:44	31.08.2021 09:11	1/9576	09/01/2021 0234	889	1042
20401721	[0]	18.08.2021 08:09	18.08.2021 05:17	1/1448	08/18/2021 2317	907	1079
20035916	[0]	08.08.2021 12:10	08.08.2021 06:16	1/3406	08/09/2021 0047	756	1110
20478906	[0]	20.08.2021 06:59	20.08.2021 06:09	1/2927	08/21/2021 0056	1076	1126
21400590	[0]	12.09.2021 04:37	12.09.2021 04:00	1/6734	09/12/2021 2253	1095	1132
19908855	[0]	05.08.2021 12:30	04.08.2021 19:50	1/2230	08/05/2021 1639	248	1248
22580917	[0]	15.10.2021 23:12	15.10.2021 20:39	1/4154	10/16/2021 1853	1180	1333
20478910	[0]	20.08.2021 06:59	20.08.2021 06:08	1/2957	08/21/2021 0430	1290	1341
20829231	[0]	29.08.2021 00:24	28.08.2021 21:30	1/8078	08/29/2021 2120	1255	1429
21370680	[1]	12.09.2021 04:37	10.09.2021 17:35	1/6390	09/11/2021 1815	-622	1479
20669980	[0]	25.08.2021 07:09	24.08.2021 18:39	1/5787	08/25/2021 2325	975	1725
19804189	[0]	02.08.2021 05:29	01.08.2021 20:46	1/0017	08/03/2021 0211	1241	1764
20226064	[0]	14.08.2021 13:00	13.08.2021 10:39	1/8177	08/14/2021 1823	322	1903
21008961	[0]	03.09.2021 05:26	02.09.2021 10:15	1/1567	09/03/2021 2346	1099	2250
21004654	[0]	03.09.2021 12:23	02.09.2021 09:24	1/1567	09/03/2021 2346	682	2301
20874510	[0]	31.08.2021 11:44	30.08.2021 09:30	1/9568	09/01/2021 0234	889	2463
20866921	[0]	31.08.2021 05:31	29.08.2021 21:12	1/9486	08/31/2021 2349	1097	3036
20353878	[0]	17.08.2021 05:18	15.08.2021 18:59	1/0550	08/18/2021 0329	1330	3389
21131007	[0]	07.09.2021 23:05	05.09.2021 10:08	1/2777	09/07/2021 2052	-133	3523
20358929	[1]	18.08.2021 11:43	16.08.2021 06:09	1/1448	08/18/2021 2317	693	3907
20753704	[0]	28.08.2021 22:55	26.08.2021 21:18	1/8060	08/29/2021 1854	1198	4175
21803467	[1]	23.09.2021 08:23	20.09.2021 18:51	1/2659	09/23/2021 1723	539	4231
19740802	[1]	03.08.2021 19:08	31.07.2021 04:05	1/0427	08/03/2021 1529	-219	5003
20560392	[0]	26.08.2021 00:13	22.08.2021 05:09	1/5900	08/26/2021 0159	105	5569
22238197	[0]	09.10.2021 11:49	05.10.2021 13:25	1/1889	10/09/2021 1506	196	5860
20441640	[0]	23.08.2021 05:00	19.08.2021 00:05	1/4219	08/24/2021 0049	1188	7243
21371987	[0]	19.09.2021 05:34	11.09.2021 09:55	1/0713	09/20/2021 0148	1213	12472
20294555	[0]	24.08.2021 22:35	15.08.2021 09:12	1/4698	08/24/2021 1453	-462	13300
21325525	[0]	22.09.2021 05:32	10.09.2021 02:35	1/2205	09/22/2021 2205	992	18449
19775538	[0]	20.08.2021 05:18	31.07.2021 20:16	1/2927	08/21/2021 0056	1177	29079
19773091	[1]	20.08.2021 06:59	31.07.2021 18:59	1/2927	08/21/2021 0056	1076	29156

Table 14 Calculated timespans in minutes since fire detection and acquisition until TFR issue date for FIR Seattle (KZSE). 103 fire clusters could not get tied to a "first" TFR (111 clusters are with 224 TFRs being considered). It occurs 9 (10 if a 9 minute value is considered as well) times that TFR is issued before knowing about it from satellite data can be possible. Like for 1/3308, this may be due to fires spotting downwind that produce new fire cluster ids

id	Fire type	Fire detection	Fire acquisition	NOTAM # from ZSE	Issue Date (UTC)	Timespan Detection [min]	Timespan Acquisition [min]
20832936	[0]	08.09.2021 06:04	29.08.2021 04:44	1/4930	08/11/2021 0431	-40413	-25933
21476064	[0]	16.09.2021 05:55	14.09.2021 09:48	1/7291	09/14/2021 0008	-3227	-580
19784784	[0]	01.08.2021 12:41	01.08.2021 10:14	1/9040	08/01/2021 0100	-701	-554
20646621	[0]	28.08.2021 23:09	23.08.2021 10:00	1/3308	08/23/2021 0249	-8420	-431
21251076	[0]	08.09.2021 22:31	07.09.2021 20:55	1/2287	09/07/2021 1413	-1938	-402
20477954	[0]	20.08.2021 04:14	20.08.2021 03:39	1/2176	08/20/2021 0020	-234	-199
21322852	[0]	10.09.2021 00:10	09.09.2021 21:06	1/4822	09/09/2021 1750	-380	-196
21725066	[0]	21.09.2021 12:40	21.09.2021 10:06	1/1239	09/21/2021 0856	-224	-70
20213789	[0]	13.08.2021 06:05	13.08.2021 02:35	1/7268	08/13/2021 0230	-215	-5
21325523	[0]	10.09.2021 05:21	10.09.2021 04:42	1/5509	09/10/2021 0440	-41	-2
19952934	[0]	06.08.2021 05:11	06.08.2021 04:20	1/2545	08/06/2021 0430	-41	9
19912173	[1]	05.08.2021 05:28	05.08.2021 04:41	1/1707	08/05/2021 0458	-30	16
19844481	[0]	03.08.2021 05:10	03.08.2021 03:35	1/0029	08/03/2021 0513	2	97
20908577	[1]	30.08.2021 22:23	30.08.2021 21:45	1/8794	08/30/2021 2347	83	121
21401129	[0]	11.09.2021 22:20	11.09.2021 22:15	1/6443	09/12/2021 0018	117	122
21933507	[0]	26.09.2021 20:07	26.09.2021 18:45	1/3312	09/26/2021 2128	80	162
22307054	[0]	07.10.2021 19:47	07.10.2021 18:57	1/1515	10/07/2021 2152	124	174
20176896	[0]	11.08.2021 19:56	11.08.2021 18:52	1/5661	08/11/2021 2149	112	176
20176897	[1]	11.08.2021 22:59	11.08.2021 20:10	1/5729	08/12/2021 0044	104	273
20629444	[0]	24.08.2021 00:18	23.08.2021 21:24	1/4366	08/24/2021 0208	109	283
19775538	[0]	31.07.2021 23:08	31.07.2021 20:16	1/9040	08/01/2021 0100	111	283
20318049	[0]	15.08.2021 23:40	15.08.2021 20:36	1/8595	08/16/2021 0121	100	284
20358787	[0]	16.08.2021 23:53	16.08.2021 23:45	1/9486	08/17/2021 0446	292	300
20042615	[0]	08.08.2021 13:13	08.08.2021 10:31	1/3257	08/08/2021 1604	170	332
19804186	[0]	01.08.2021 23:06	01.08.2021 20:35	1/9278	08/02/2021 0232	206	356
19841939	[0]	03.08.2021 00:10	02.08.2021 21:19	1/0023	08/03/2021 0424	253	424
19949838	[1]	05.08.2021 22:04	05.08.2021 21:12	1/2543	08/06/2021 0418	373	425
19845003	[0]	03.08.2021 06:07	02.08.2021 21:19	1/0025	08/03/2021 0427	-100	427
20035719	[0]	08.08.2021 11:28	08.08.2021 08:50	1/3257	08/08/2021 1604	275	433
20035720	[0]	08.08.2021 11:28	08.08.2021 08:50	1/3257	08/08/2021 1604	275	433
21427428	[0]	12.09.2021 23:16	12.09.2021 20:59	1/6770	09/13/2021 0428	311	448
19841942	[0]	03.08.2021 05:10	02.08.2021 21:19	1/0029	08/03/2021 0513	2	473
21006486	[0]	02.09.2021 12:37	02.09.2021 10:14	1/0914	09/02/2021 1819	341	484
19908817	[0]	04.08.2021 22:24	04.08.2021 19:52	1/1701	08/05/2021 0448	383	535
20148777	[0]	11.08.2021 12:07	11.08.2021 09:35	1/5421	08/11/2021 1911	423	575
21006483	[0]	02.09.2021 12:37	02.09.2021 10:14	1/1133	09/02/2021 2005	447	590
19916066	[0]	04.08.2021 19:51	04.08.2021 18:57	1/1704	08/05/2021 0451	539	593
20150361	[0]	10.08.2021 16:38	09.08.2021 18:39	1/4073	08/10/2021 0437	-721	597
19773092	[1]	31.07.2021 21:58	31.07.2021 17:05	1/9078	08/01/2021 0410	371	664
20954526	[0]	01.09.2021 05:07	01.09.2021 04:29	1/9869	09/01/2021 1633	685	723
21295112	[0]	09.09.2021 12:14	09.09.2021 09:43	1/5289	09/09/2021 2325	670	821
21259999	[0]	08.09.2021 12:30	08.09.2021 10:00	1/3886	09/08/2021 2345	674	824
21437171	[0]	13.09.2021 12:42	13.09.2021 10:07	1/7291	09/14/2021 0008	685	840
19882759	[0]	05.08.2021 04:27	04.08.2021 05:50	1/1480	08/04/2021 2042	-465	891
19952936	[0]	06.08.2021 05:11	06.08.2021 03:25	1/2972	08/06/2021 1834	802	908
21328845	[0]	10.09.2021 08:20	10.09.2021 05:22	1/6261	09/10/2021 2035	734	912
21168941	[0]	06.09.2021 12:24	06.09.2021 09:47	1/2133	09/07/2021 0138	793	950
20828688	[0]	28.08.2021 22:38	28.08.2021 19:52	1/7956	08/29/2021 1207	808	974
20320934	[0]	16.08.2021 07:21	15.08.2021 22:55	1/8925	08/16/2021 1513	471	977
20325699	[0]	16.08.2021 11:41	16.08.2021 08:50	1/9419	08/17/2021 0210	868	1039
20593803	[0]	22.08.2021 23:26	22.08.2021 20:55	1/3648	08/23/2021 1437	910	1061
20478906	[0]	20.08.2021 06:59	20.08.2021 06:09	1/2912	08/21/2021 0020	1040	1090
20027955	[0]	08.08.2021 00:15	07.08.2021 21:26	1/3257	08/08/2021 1604	948	1117
19812473	[0]	03.08.2021 06:07	02.08.2021 09:55	1/0029	08/03/2021 0513	-54	1157
21286336	[0]	08.09.2021 23:03	08.09.2021 20:34	1/4769	09/09/2021 1612	1028	1177
20247609	[1]	14.08.2021 05:41	13.08.2021 19:33	1/8150	08/14/2021 1556	614	1222
21253751	[0]	08.09.2021 04:20	08.09.2021 03:45	1/3937	09/09/2021 0115	1254	1289
19989128	[1]	06.08.2021 22:43	06.08.2021 19:55	1/3139	08/07/2021 1743	1139	1307
19779404	[0]	01.08.2021 05:50	01.08.2021 05:10	1/9324	08/02/2021 0506	1395	1435

id	Fire type	Fire detection	Fire acquisition	NOTAM # from ZSE	Issue Date (UTC)	Timespan Detection [min]	Timespan Acquisition [min]
19844480	[0]	03.08.2021 05:10	03.08.2021 00:55	1/0761	08/04/2021 0100	1189	1444
20804305	[0]	28.08.2021 22:11	28.08.2021 10:06	1/7942	08/29/2021 1028	736	1461
20804299	[0]	29.08.2021 12:17	28.08.2021 10:06	1/7956	08/29/2021 1207	-10	1560
21797908	[0]	23.09.2021 05:50	22.09.2021 22:03	1/2878	09/24/2021 0104	1153	1620
21250885	[0]	07.09.2021 23:07	06.09.2021 18:49	1/2953	09/07/2021 2303	-4	1693
21168945	[0]	06.09.2021 23:41	06.09.2021 09:47	1/2277	09/07/2021 1411	869	1703
21034865	[0]	03.09.2021 12:23	02.09.2021 20:48	1/1585	09/04/2021 0230	846	1781
19909474	[0]	05.08.2021 05:28	04.08.2021 21:33	1/2528	08/06/2021 0323	1314	1789
20000795	[0]	08.08.2021 22:52	07.08.2021 10:01	1/3261	08/08/2021 1653	-359	1851
19773091	[1]	01.08.2021 05:50	31.07.2021 18:59	1/9247	08/02/2021 0206	1215	1866
21471226	[0]	15.09.2021 05:17	14.09.2021 05:00	1/8874	09/15/2021 1725	727	2184
21414211	[0]	14.09.2021 05:39	12.09.2021 09:35	1/7291	09/14/2021 0008	-331	2312
21259984	[0]	09.09.2021 13:09	08.09.2021 10:02	1/5435	09/10/2021 0106	716	2343
19849095	[0]	03.08.2021 02:04	02.08.2021 09:55	1/0760	08/04/2021 0100	1375	2344
21255610	[0]	09.09.2021 12:14	08.09.2021 05:52	1/5270	09/09/2021 2251	636	2459
20042613	[0]	09.08.2021 05:54	08.08.2021 10:32	1/4074	08/10/2021 0439	1364	2526
19886265	[0]	05.08.2021 07:28	04.08.2021 10:07	1/2543	08/06/2021 0418	1249	2530
19912172	[0]	07.08.2021 00:35	05.08.2021 04:41	1/3039	08/07/2021 0416	220	2854
19905324	[1]	06.08.2021 19:36	04.08.2021 18:57	1/3005	08/06/2021 2059	82	3001
19912619	[0]	08.08.2021 06:44	05.08.2021 01:05	1/3040	08/07/2021 0417	-1587	3071
20099608	[0]	11.08.2021 05:49	09.08.2021 18:39	1/5866	08/12/2021 0442	1372	3482
21131007	[0]	07.09.2021 23:05	05.09.2021 10:08	1/2775	09/07/2021 2041	-144	3512
19912171	[0]	06.08.2021 22:43	05.08.2021 04:41	1/3158	08/07/2021 1847	1203	3725
20220811	[0]	15.08.2021 05:28	13.08.2021 09:47	1/8637	08/16/2021 0408	1359	3980
19849606	[0]	05.08.2021 05:28	03.08.2021 09:36	1/2543	08/06/2021 0418	1369	4001
21168939	[0]	08.09.2021 12:30	06.09.2021 09:47	1/4079	09/09/2021 0516	1005	4048
21212938	[0]	09.09.2021 05:43	07.09.2021 04:56	1/5435	09/10/2021 0106	1162	4089
20753704	[0]	28.08.2021 22:55	26.08.2021 21:18	1/8057	08/29/2021 1830	1174	4151
21080530	[0]	06.09.2021 22:02	04.09.2021 08:43	1/2287	09/07/2021 1413	970	4649
19918896	[0]	08.08.2021 11:28	05.08.2021 08:56	1/3256	08/08/2021 1603	274	4746
21833219	[0]	27.09.2021 06:14	24.09.2021 05:58	1/3891	09/28/2021 0310	1256	5591
19908851	[5]	09.08.2021 04:47	04.08.2021 19:52	1/3353	08/08/2021 2036	-491	5803
20707964	[0]	30.08.2021 05:06	25.08.2021 18:59	1/8156	08/30/2021 0407	-59	6307
19849160	[0]	06.08.2021 22:53	03.08.2021 09:34	1/3158	08/07/2021 1847	1193	6312
21034862	[0]	06.09.2021 20:32	02.09.2021 20:49	1/2287	09/07/2021 1413	1060	6803
20764088	[0]	31.08.2021 04:45	27.08.2021 09:34	1/9601	09/01/2021 0337	1371	6842
20144555	[0]	16.08.2021 12:20	11.08.2021 05:05	1/9281	08/16/2021 2330	669	8304
21043347	[0]	08.09.2021 07:40	03.09.2021 09:53	1/4096	09/09/2021 0534	1313	8380
20593648	[0]	29.08.2021 00:24	22.08.2021 20:05	1/7942	08/29/2021 1028	603	9502
21718662	[0]	27.09.2021 06:14	21.09.2021 05:07	1/3891	09/28/2021 0310	1256	9962
20110035	[0]	14.08.2021 05:41	08.08.2021 04:15	1/8337	08/15/2021 0424	1362	10088
20804306	[0]	03.09.2021 05:27	27.08.2021 18:49	1/1528	09/03/2021 2053	925	10203
21714880	[0]	27.09.2021 06:14	20.09.2021 18:51	1/3891	09/28/2021 0310	1256	10578
20068735	[0]	16.08.2021 11:41	09.08.2021 09:21	1/9449	08/17/2021 0346	964	11184
19845194	[0]	03.08.2021 06:49	03.08.2021 06:09	1/4928	08/11/2021 0429	11379	11419
20353878	[0]	23.08.2021 16:57	15.08.2021 18:59	1/4366	08/24/2021 0208	550	11948
21371987	[0]	19.09.2021 05:34	11.09.2021 09:55	1/0714	09/20/2021 0151	1216	12475
20150368	[0]	20.08.2021 11:51	11.08.2021 09:34	1/2952	08/21/2021 0346	954	14051
20837946	[0]	08.09.2021 12:30	29.08.2021 09:48	1/4079	09/09/2021 0516	1005	15567
19878953	[0]	16.08.2021 05:43	03.08.2021 20:11	1/9473	08/17/2021 0434	1370	19222
19916069	[0]	05.08.2021 11:31	05.08.2021 08:56	1/2952	08/21/2021 0346	22574	22729
19740802	[1]	23.08.2021 05:58	31.07.2021 04:05	1/4366	08/24/2021 0208	1209	34442

8.4. Coverage Quality Log

The full log from Get_TFR-exceeding_Fires_from_API.ipynb is appended here. It contains all results for 5.2.

```
Number of evaluated TFRs for ZAB 0 SM: 2
Number of potential cases where fire leaves TFR for ZAB 0 SM: 0
Number of potential fire events leaving TFR for ZAB 0 SM: 0
Number of evaluated TFRs for ZAB 1 SM: 2
Number of potential cases where fire leaves TFR for ZAB 1 SM: 0
Number of potential fire events leaving TFR for ZAB 1 SM: 0
Number of evaluated TFRs for ZAB 3 SM: 2
Number of potential cases where fire leaves TFR for ZAB 3 SM: 0
Number of potential fire events leaving TFR for ZAB 3 SM: 0

Number of evaluated TFRs for ZDV 0 SM: 28
Number of potential cases where fire leaves TFR for ZDV 0 SM: 2
Number of potential fire events leaving TFR for ZDV 0 SM: 2
Number results for ZDV 0 SM: 1
Number of TFR(s) cancelled before becoming effective for ZDV 0 SM: 1
Number of results without duplicates for ZDV 0 SM: 1
Number of TFRs where a fire leaves the TFR for ZDV 0 SM: 1
Number of fire events leaving a TFR for ZDV 0 SM: 1

Number of evaluated TFRs for ZDV 1 SM: 28
Number of potential cases where fire leaves TFR for ZDV 1 SM: 5
Number of potential fire events leaving TFR for ZDV 1 SM: 4
Number results for ZDV 1 SM: 1
Number of TFR(s) cancelled before becoming effective for ZDV 1 SM: 1
Number of results without duplicates for ZDV 1 SM: 1
Number of TFRs where a fire leaves the TFR for ZDV 1 SM: 1
Number of fire events leaving a TFR for ZDV 1 SM: 1

Number of evaluated TFRs for ZDV 3 SM: 28
Number of potential cases where fire leaves TFR for ZDV 3 SM: 11
Number of potential fire events leaving TFR for ZDV 3 SM: 7
Number results for ZDV 3 SM: 5
Number of TFR(s) cancelled before becoming effective for ZDV 3 SM: 1
Number of results without duplicates for ZDV 3 SM: 5
Number of TFRs where a fire leaves the TFR for ZDV 3 SM: 5
Number of fire events leaving a TFR for ZDV 3 SM: 3

Number of evaluated TFRs for ZFW 0 SM: 3
Number of potential cases where fire leaves TFR for ZFW 0 SM: 3
Number of potential fire events leaving TFR for ZFW 0 SM: 1
Number results for ZFW 0 SM: 3
Number of TFR(s) cancelled before becoming effective for ZFW 0 SM: 1
Number of results without duplicates for ZFW 0 SM: 3
Number of TFRs where a fire leaves the TFR for ZFW 0 SM: 3
Number of fire events leaving a TFR for ZFW 0 SM: 1

Number of evaluated TFRs for ZFW 1 SM: 3
Number of potential cases where fire leaves TFR for ZFW 1 SM: 3
Number of potential fire events leaving TFR for ZFW 1 SM: 1
Number results for ZFW 1 SM: 6
Number of TFR(s) cancelled before becoming effective for ZFW 1 SM: 1
Number of results without duplicates for ZFW 1 SM: 6
Number of TFRs where a fire leaves the TFR for ZFW 1 SM: 3
Number of fire events leaving a TFR for ZFW 1 SM: 1

Number of evaluated TFRs for ZFW 3 SM: 3
Number of potential cases where fire leaves TFR for ZFW 3 SM: 3
Number of potential fire events leaving TFR for ZFW 3 SM: 1
```

Number results for ZFW 3 SM: 6
Number of TFR(s) cancelled before becoming effective for ZFW 3 SM: 1
Number of results without duplicates for ZFW 3 SM: 6
Number of TFRs where a fire leaves the TFR for ZFW 3 SM: 3
Number of fire events leaving a TFR for ZFW 3 SM: 1

Number of evaluated TFRs for ZLA 0 SM: 40
Number of potential cases where fire leaves TFR for ZLA 0 SM: 22
Number of potential fire events leaving TFR for ZLA 0 SM: 8
Number results for ZLA 0 SM: 31
Number of TFR(s) cancelled before becoming effective for ZLA 0 SM: 0
Number of results without duplicates for ZLA 0 SM: 31
Number of TFRs where a fire leaves the TFR for ZLA 0 SM: 12
Number of fire events leaving a TFR for ZLA 0 SM: 5

Number of evaluated TFRs for ZLA 1 SM: 40
Number of potential cases where fire leaves TFR for ZLA 1 SM: 34
Number of potential fire events leaving TFR for ZLA 1 SM: 10
Number results for ZLA 1 SM: 99
Number of TFR(s) cancelled before becoming effective for ZLA 1 SM: 0
Number of results without duplicates for ZLA 1 SM: 99
Number of TFRs where a fire leaves the TFR for ZLA 1 SM: 23
Number of fire events leaving a TFR for ZLA 1 SM: 14

Number of evaluated TFRs for ZLA 3 SM: 40
Number of potential cases where fire leaves TFR for ZLA 3 SM: 59
Number of potential fire events leaving TFR for ZLA 3 SM: 15
Number results for ZLA 3 SM: 285
Number of TFR(s) cancelled before becoming effective for ZLA 3 SM: 0
Number of results without duplicates for ZLA 3 SM: 274
Number of TFRs where a fire leaves the TFR for ZLA 3 SM: 37
Number of fire events leaving a TFR for ZLA 3 SM: 32

Number of evaluated TFRs for ZLC 0 SM: 114
Number of potential cases where fire leaves TFR for ZLC 0 SM: 90
Number of potential fire events leaving TFR for ZLC 0 SM: 32
Number results for ZLC 0 SM: 122
Number of TFR(s) cancelled before becoming effective for ZLC 0 SM: 1
Number of results without duplicates for ZLC 0 SM: 111
Number of TFRs where a fire leaves the TFR for ZLC 0 SM: 34
Number of fire events leaving a TFR for ZLC 0 SM: 24

Number of evaluated TFRs for ZLC 1 SM: 114
Number of potential cases where fire leaves TFR for ZLC 1 SM: 136
Number of potential fire events leaving TFR for ZLC 1 SM: 46
Number results for ZLC 1 SM: 334
Number of TFR(s) cancelled before becoming effective for ZLC 1 SM: 1
Number of results without duplicates for ZLC 1 SM: 269
Number of TFRs where a fire leaves the TFR for ZLC 1 SM: 43
Number of fire events leaving a TFR for ZLC 1 SM: 37

Number of evaluated TFRs for ZLC 3 SM: 114
Number of potential cases where fire leaves TFR for ZLC 3 SM: 225
Number of potential fire events leaving TFR for ZLC 3 SM: 77
Number results for ZLC 3 SM: 1058
Number of TFR(s) cancelled before becoming effective for ZLC 3 SM: 1
Number of results without duplicates for ZLC 3 SM: 637
Number of TFRs where a fire leaves the TFR for ZLC 3 SM: 61
Number of fire events leaving a TFR for ZLC 3 SM: 81

Number of evaluated TFRs for ZMP 0 SM: 10
Number of potential cases where fire leaves TFR for ZMP 0 SM: 5
Number of potential fire events leaving TFR for ZMP 0 SM: 1

```
Number results for ZMP 0 SM: 9
Number of TFR(s) cancelled before becoming effective for ZMP 0 SM: 1
Number of results without duplicates for ZMP 0 SM: 9
Number of TFRs where a fire leaves the TFR for ZMP 0 SM: 5
Number of fire events leaving a TFR for ZMP 0 SM: 2

Number of evaluated TFRs for ZMP 1 SM: 10
Number of potential cases where fire leaves TFR for ZMP 1 SM: 7
Number of potential fire events leaving TFR for ZMP 1 SM: 1
Number results for ZMP 1 SM: 13
Number of TFR(s) cancelled before becoming effective for ZMP 1 SM: 1
Number of results without duplicates for ZMP 1 SM: 13
Number of TFRs where a fire leaves the TFR for ZMP 1 SM: 7
Number of fire events leaving a TFR for ZMP 1 SM: 2

Number of evaluated TFRs for ZMP 3 SM: 10
Number of potential cases where fire leaves TFR for ZMP 3 SM: 8
Number of potential fire events leaving TFR for ZMP 3 SM: 2
Number results for ZMP 3 SM: 31
Number of TFR(s) cancelled before becoming effective for ZMP 3 SM: 1
Number of results without duplicates for ZMP 3 SM: 31
Number of TFRs where a fire leaves the TFR for ZMP 3 SM: 7
Number of fire events leaving a TFR for ZMP 3 SM: 2

Number of evaluated TFRs for ZOA 0 SM: 99
Number of potential cases where fire leaves TFR for ZOA 0 SM: 159
Number of potential fire events leaving TFR for ZOA 0 SM: 43
Number results for ZOA 0 SM: 771
Number of TFR(s) cancelled before becoming effective for ZOA 0 SM: 2
Number of results without duplicates for ZOA 0 SM: 737
Number of TFRs where a fire leaves the TFR for ZOA 0 SM: 71
Number of fire events leaving a TFR for ZOA 0 SM: 34

Number of evaluated TFRs for ZOA 1 SM: 99
Number of potential cases where fire leaves TFR for ZOA 1 SM: 250
Number of potential fire events leaving TFR for ZOA 1 SM: 53
Number results for ZOA 1 SM: 1420
Number of TFR(s) cancelled before becoming effective for ZOA 1 SM: 2
Number of results without duplicates for ZOA 1 SM: 1288
Number of TFRs where a fire leaves the TFR for ZOA 1 SM: 89
Number of fire events leaving a TFR for ZOA 1 SM: 92

Number of evaluated TFRs for ZOA 3 SM: 99
Number of potential cases where fire leaves TFR for ZOA 3 SM: 404
Number of potential fire events leaving TFR for ZOA 3 SM: 65
Number results for ZOA 3 SM: 3684
Number of TFR(s) cancelled before becoming effective for ZOA 3 SM: 2
Number of results without duplicates for ZOA 3 SM: 2990
Number of TFRs where a fire leaves the TFR for ZOA 3 SM: 96
Number of fire events leaving a TFR for ZOA 3 SM: 162

Number of evaluated TFRs for ZSE 0 SM: 247
Number of potential cases where fire leaves TFR for ZSE 0 SM: 375
Number of potential fire events leaving TFR for ZSE 0 SM: 110
Number results for ZSE 0 SM: 1161
Number of TFR(s) cancelled before becoming effective for ZSE 0 SM: 7
Number of results without duplicates for ZSE 0 SM: 1047
Number of TFRs where a fire leaves the TFR for ZSE 0 SM: 145
Number of fire events leaving a TFR for ZSE 0 SM: 75

Number of evaluated TFRs for ZSE 1 SM: 247
Number of potential cases where fire leaves TFR for ZSE 1 SM: 618
Number of potential fire events leaving TFR for ZSE 1 SM: 160
```

```
Number results for ZSE 1 SM: 2643
Number of TFR(s) cancelled before becoming effective for ZSE 1 SM: 7
Number of results without duplicates for ZSE 1 SM: 2279
Number of TFRs where a fire leaves the TFR for ZSE 1 SM: 180
Number of fire events leaving a TFR for ZSE 1 SM: 177

Number of evaluated TFRs for ZSE 3 SM: 247
Number of potential cases where fire leaves TFR for ZSE 3 SM: 1116
Number of potential fire events leaving TFR for ZSE 3 SM: 227
Number results for ZSE 3 SM: 8727
Number of TFR(s) cancelled before becoming effective for ZSE 3 SM: 7
Number of results without duplicates for ZSE 3 SM: 6442
Number of TFRs where a fire leaves the TFR for ZSE 3 SM: 208
Number of fire events leaving a TFR for ZSE 3 SM: 298
```

Text 3 ZAB has no fire clusters overlapping TFRs. For all other FIRs, the eight lines per buffer distance (statute miles: SM) provide the following:

The number of TFRs regarded, then, prior to API request of clusters acquired during TFR runtime,

the number of spatially joined polygons (overlaps plus containments) for that buffer distance,

the number of fire events from that,

the total number of spatially joined polygons (overlaps plus containments) from API for that buffer distance (which has duplicates),

a number of TFRs to be considered as well for this because they got cancelled in advance (which may be due to not spatially matching a ground truth fire perimeter anymore),

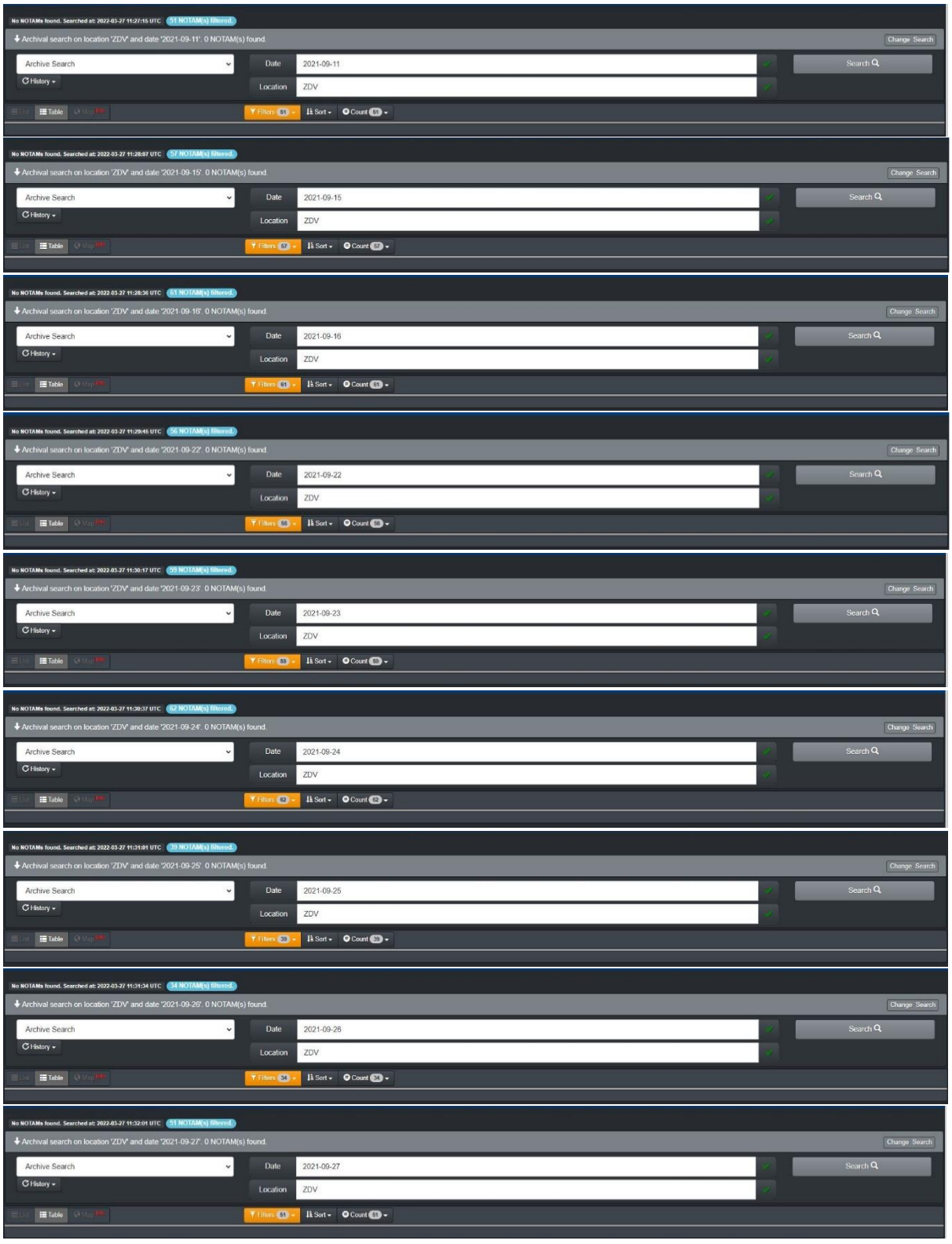
the number of spatially joined polygons (overlaps plus containments) from API for that buffer distance (no duplicated entire rows) and finally

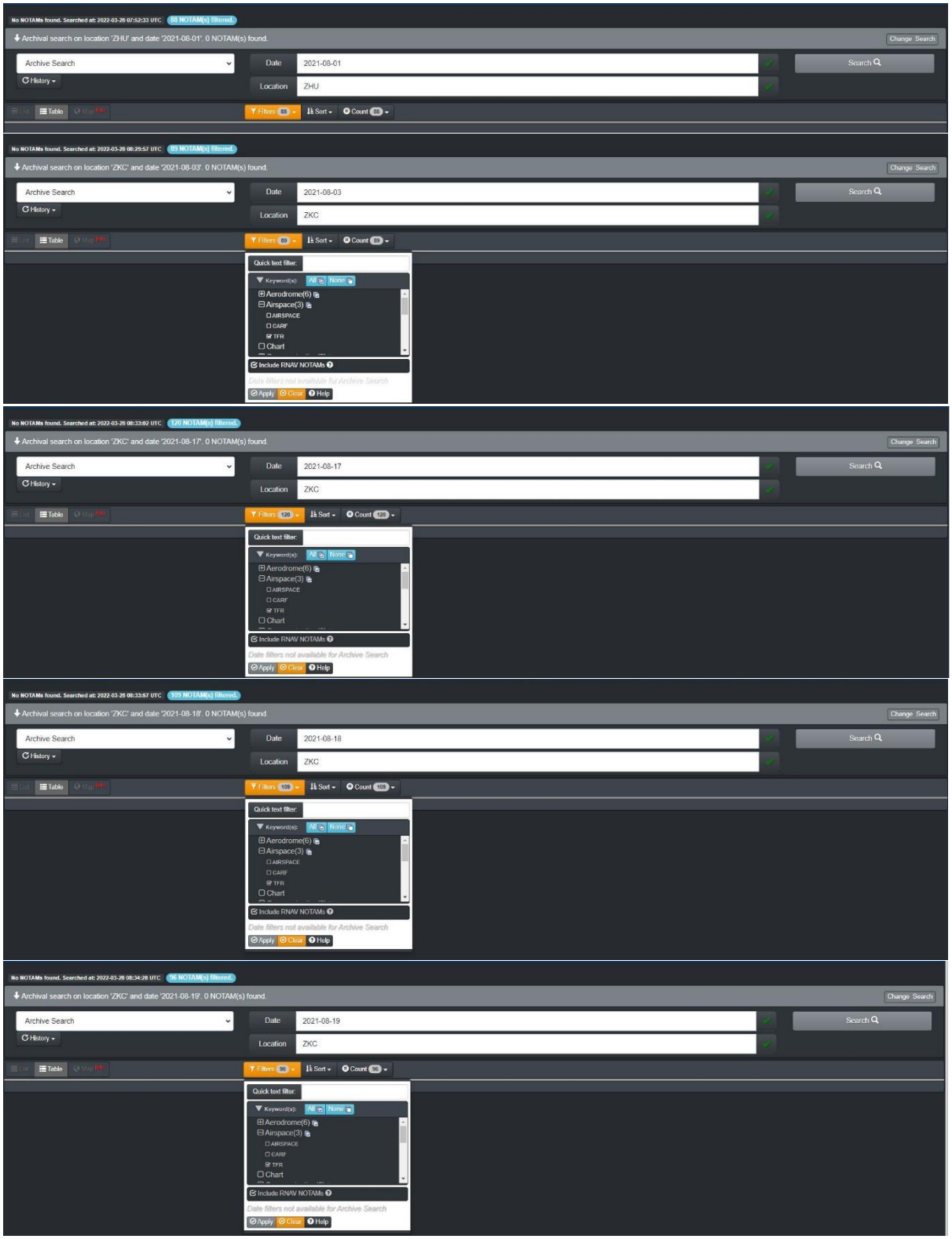
the number of TFRs being overlapped (or contained) by a fire, which has a larger value than the following, if consecutive TFR do not consider fire growth and

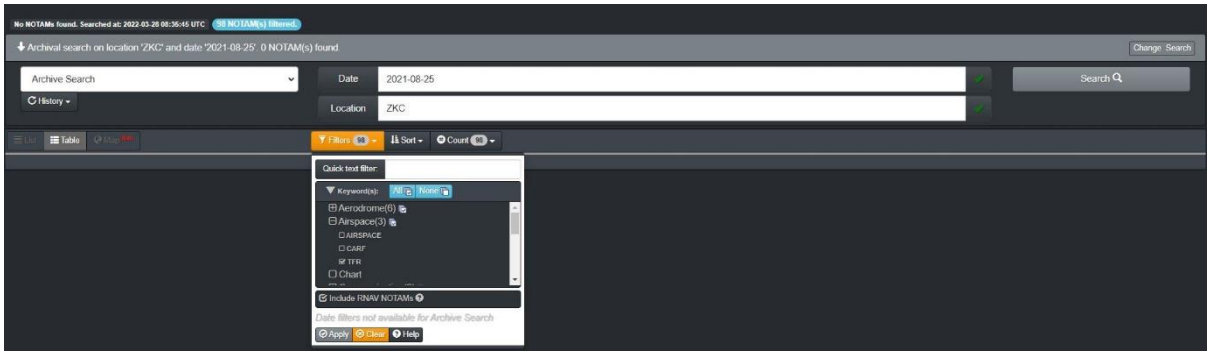
the number of fire clusters that overlap (or contain) a TFR.

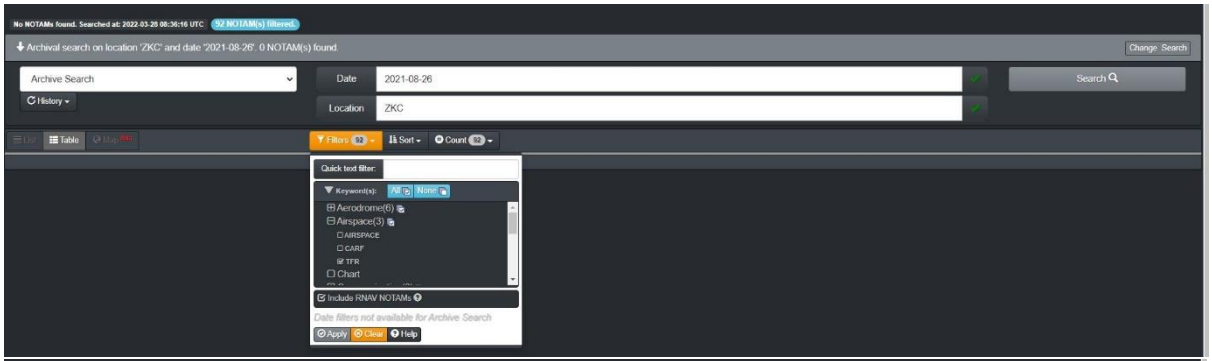
8.5. No-TFR-Data-Documentation

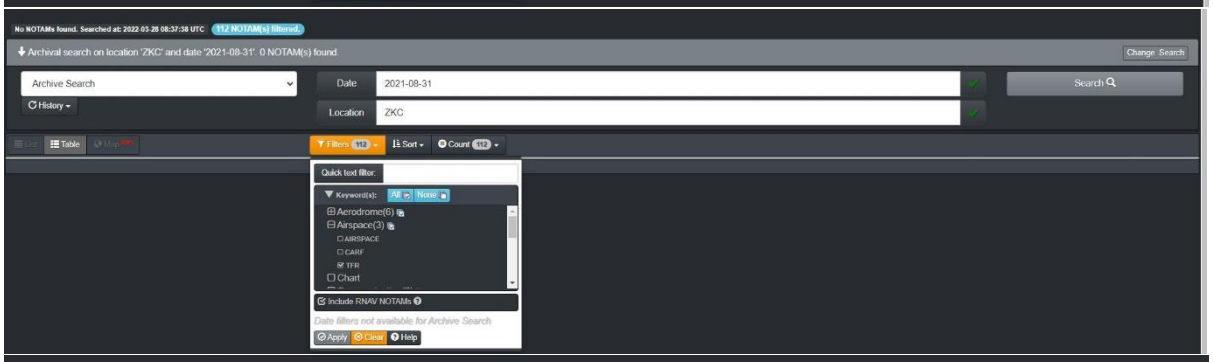
Screenshots of location and date where and when no effective TFR was listed in the NOTAM Archive are provided in this section. Together with the VBA log from 8.6 they prove that fetched TFR data is complete for all 10 FIRs over the observed timespan.

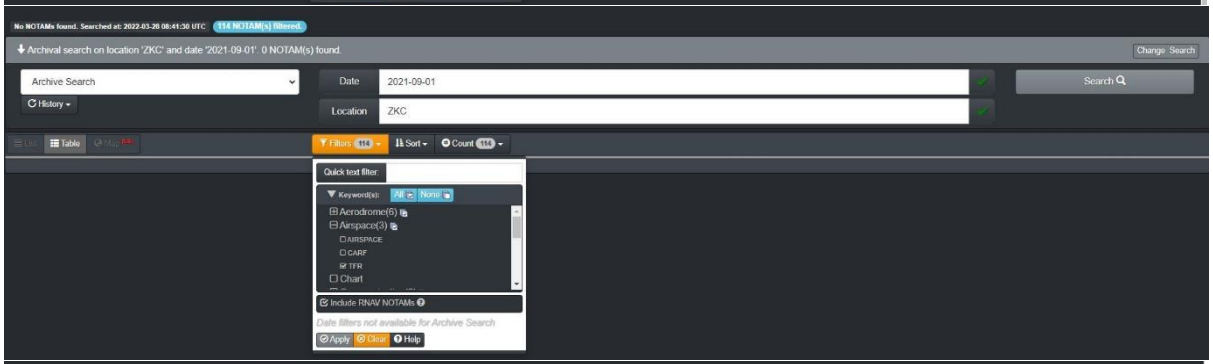


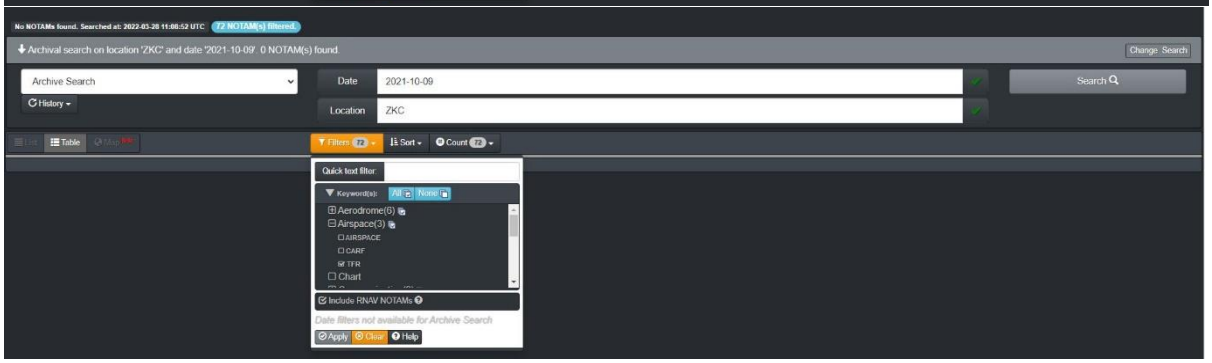

No NOTAMs found. Searched at: 2022-03-26 07:52:33 UTC
88 NOTAM(s) filtered.
Archival search on location 'ZHU' and date '2021-08-01'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-08-01
Location
ZHU
Search
Table
Filters 88
Sort
Count 88
No NOTAMs found. Searched at: 2022-03-26 08:29:57 UTC
89 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-08-03'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-08-03
Location
ZKC
Search
Table
Filters 89
Sort
Count 89
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help
No NOTAMs found. Searched at: 2022-03-26 08:33:02 UTC
120 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-08-17'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-08-17
Location
ZKC
Search
Table
Filters 120
Sort
Count 120
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help
No NOTAMs found. Searched at: 2022-03-26 08:33:57 UTC
109 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-08-18'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-08-18
Location
ZKC
Search
Table
Filters 109
Sort
Count 109
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help
No NOTAMs found. Searched at: 2022-03-26 08:34:28 UTC
96 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-08-19'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-08-19
Location
ZKC
Search
Table
Filters 96
Sort
Count 96
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help

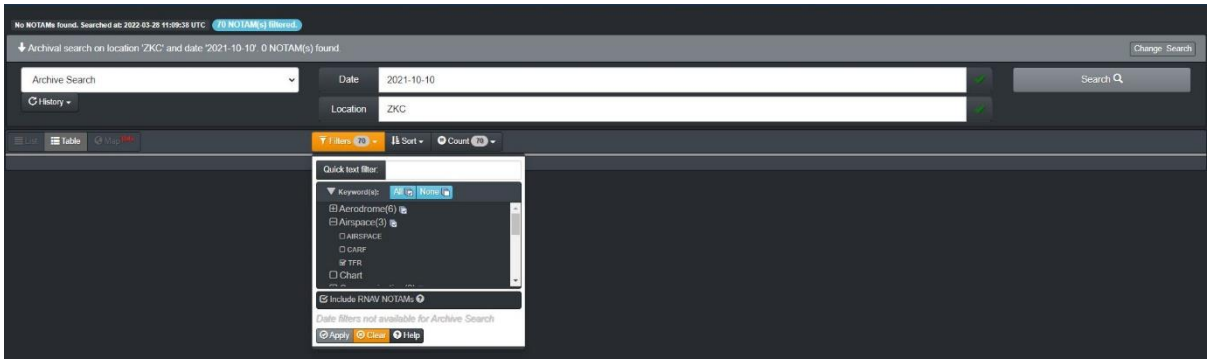

No NOTAMs found. Searched at: 2022-03-28 08:36:45 UTC
98 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-08-25'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-08-25
Location
ZKC
Search
List
Table
Map
Filters 98
Sort
Count 98
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help

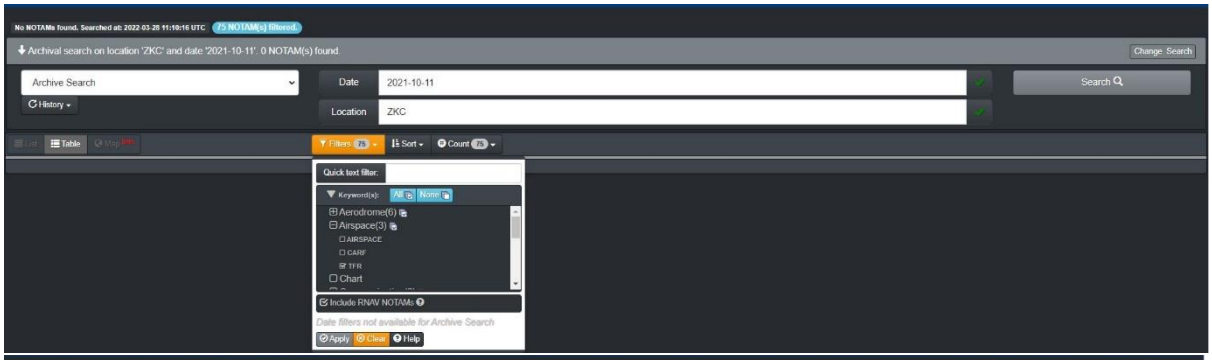

No NOTAMs found. Searched at: 2022-03-28 08:36:16 UTC
92 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-08-26'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-08-26
Location
ZKC
Search
List
Table
Map
Filters 92
Sort
Count 92
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help

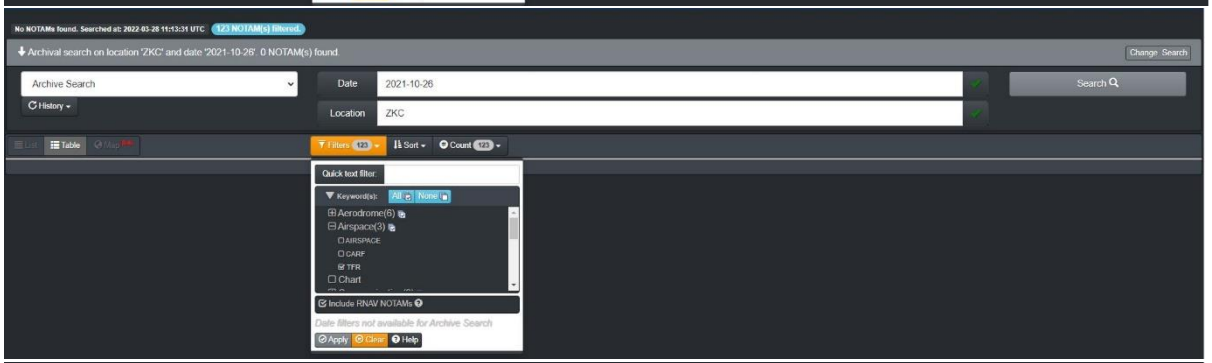

No NOTAMs found. Searched at: 2022-03-28 08:37:38 UTC
112 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-08-31'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-08-31
Location
ZKC
Search
List
Table
Map
Filters 112
Sort
Count 112
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help

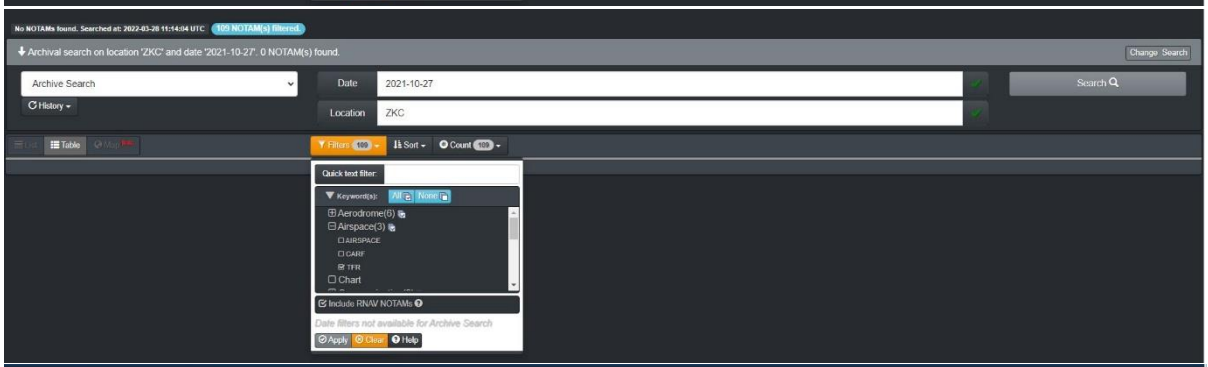

No NOTAMs found. Searched at: 2022-03-28 08:41:30 UTC
114 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-09-01'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-09-01
Location
ZKC
Search
List
Table
Map
Filters 114
Sort
Count 114
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help

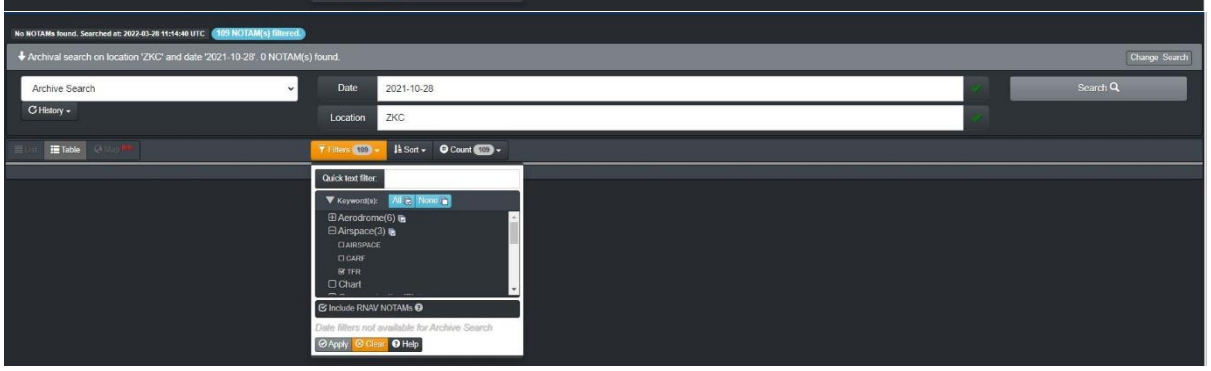

No NOTAMs found. Searched at: 2022-03-28 11:08:52 UTC
72 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-10-09'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-09
Location
ZKC
Search
List
Table
Map
Filters 72
Sort
Count 72
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help


No NOTAMs found. Searched at: 2022-03-28 11:09:38 UTC
70 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-10-10'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-10
Location
ZKC
Search
Table
Filters 70
Sort
Count 70
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help


No NOTAMs found. Searched at: 2022-03-28 11:10:16 UTC
75 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-10-11'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-11
Location
ZKC
Search
Table
Filters 75
Sort
Count 75
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help


No NOTAMs found. Searched at: 2022-03-28 11:13:31 UTC
123 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-10-26'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-26
Location
ZKC
Search
Table
Filters 123
Sort
Count 123
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help


No NOTAMs found. Searched at: 2022-03-28 11:14:04 UTC
109 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-10-27'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-27
Location
ZKC
Search
Table
Filters 109
Sort
Count 109
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help


No NOTAMs found. Searched at: 2022-03-28 11:14:40 UTC
109 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-10-28'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-28
Location
ZKC
Search
Table
Filters 109
Sort
Count 109
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help


No NOTAMs found. Searched at: 2022-03-28 11:15:07 UTC
115 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-10-29'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-29
Location
ZKC
Search
List
Table
Map
Filters 115
Sort
Count 115
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help
No NOTAMs found. Searched at: 2022-03-28 11:15:35 UTC
82 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-10-30'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-30
Location
ZKC
Search
List
Table
Map
Filters 82
Sort
Count 82
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help
No NOTAMs found. Searched at: 2022-03-28 11:16:09 UTC
71 NOTAM(s) filtered.
Archival search on location 'ZKC' and date '2021-10-31'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-31
Location
ZKC
Search
List
Table
Map
Filters 71
Sort
Count 71
Quick text filter:
Keyword(s):
All
None
Aerodrome(6)
Airspace(3)
AIRSPACE
CARF
TFR
Chart
Include RNAV NOTAMs
Date filters not available for Archive Search
Apply
Clear
Help
No NOTAMs found. Searched at: 2022-03-25 14:12:19 UTC
92 NOTAM(s) filtered.
Archival search on location 'ZSE' and date '2021-10-21'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-21
Location
ZSE
Search
List
Table
Map
Filters 92
Sort
Count 92
No NOTAMs found. Searched at: 2022-03-25 14:13:49 UTC
92 NOTAM(s) filtered.
Archival search on location 'ZSE' and date '2021-10-22'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-22
Location
ZSE
Search
List
Table
Map
Filters 92
Sort
Count
No NOTAMs found. Searched at: 2022-03-25 14:14:19 UTC
86 NOTAM(s) filtered.
Archival search on location 'ZSE' and date '2021-10-23'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-23
Location
ZSE
Search
List
Table
Map
Filters 86
Sort
Count
No NOTAMs found. Searched at: 2022-03-25 14:14:47 UTC
46 NOTAM(s) filtered.
Archival search on location 'ZSE' and date '2021-10-24'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-24
Location
ZSE
Search
List
Table
Map
Filters 46
Sort
Count


No NOTAMs found. Searched at: 2022-03-25 14:15:11 UTC
47 NOTAM(s) filtered.
Archival search on location 'ZSE' and date '2021-10-25'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-25
Location
ZSE
Search
Table
Filters 47
Sort
Count


No NOTAMs found. Searched at: 2022-03-25 14:15:48 UTC
78 NOTAM(s) filtered.
Archival search on location 'ZSE' and date '2021-10-26'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-26
Location
ZSE
Search
Table
Filters 78
Sort
Count


No NOTAMs found. Searched at: 2022-03-25 14:16:13 UTC
108 NOTAM(s) filtered.
Archival search on location 'ZSE' and date '2021-10-27'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-27
Location
ZSE
Search
Table
Filters 108
Sort
Count


No NOTAMs found. Searched at: 2022-03-25 14:16:37 UTC
129 NOTAM(s) filtered.
Archival search on location 'ZSE' and date '2021-10-28'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-28
Location
ZSE
Search
Table
Filters 129
Sort
Count


No NOTAMs found. Searched at: 2022-03-25 14:17:00 UTC
105 NOTAM(s) filtered.
Archival search on location 'ZSE' and date '2021-10-29'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-29
Location
ZSE
Search
Table
Filters 105
Sort
Count


No NOTAMs found. Searched at: 2022-03-25 14:17:27 UTC
84 NOTAM(s) filtered.
Archival search on location 'ZSE' and date '2021-10-30'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-10-30
Location
ZSE
Search
Table
Filters 84
Sort
Count

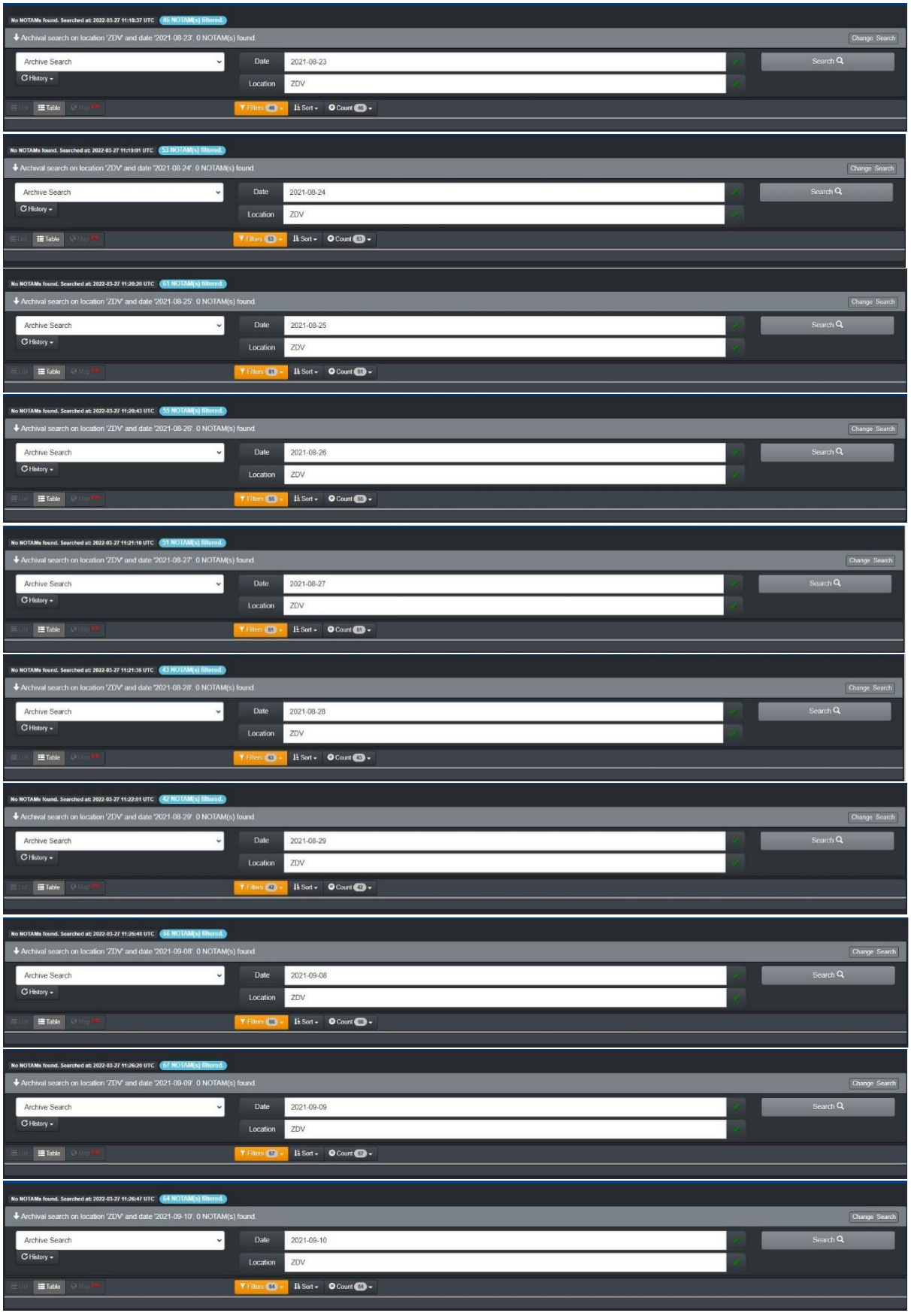

No NOTAMs found. Searched at: 2022-03-27 11:15:45 UTC
67 NOTAM(s) filtered.
Archival search on location 'ZDV' and date '2021-08-18'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-08-18
Location
ZDV
Search
Table
Filters 67
Sort
Count 67


No NOTAMs found. Searched at: 2022-03-27 11:16:46 UTC
64 NOTAM(s) filtered.
Archival search on location 'ZDV' and date '2021-08-19'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-08-19
Location
ZDV
Search
Table
Filters 64
Sort
Count 64


No NOTAMs found. Searched at: 2022-03-27 11:17:15 UTC
59 NOTAM(s) filtered.
Archival search on location 'ZDV' and date '2021-08-20'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-08-20
Location
ZDV
Search
Table
Filters 59
Sort
Count 59


No NOTAMs found. Searched at: 2022-03-27 11:17:41 UTC
43 NOTAM(s) filtered.
Archival search on location 'ZDV' and date '2021-08-21'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-08-21
Location
ZDV
Search
Table
Filters 43
Sort
Count 43


No NOTAMs found. Searched at: 2022-03-27 11:18:06 UTC
39 NOTAM(s) filtered.
Archival search on location 'ZDV' and date '2021-08-22'. 0 NOTAM(s) found.
Change Search
Archive Search
History
Date
2021-08-22
Location
ZDV
Search
Table
Filters 39
Sort
Count 39


No NOTAMs found. Searched at: 2022-03-27 11:18:37 UTC
46 NOTAM(s) filtered.
Archival search on location 'ZDV' and date '2021-08-23'. 0 NOTAM(s) found.
Change Search
Archive Search
Date
2021-08-23
Search
History
Location
ZDV
Table
Filters 46
Sort
Count 46
No NOTAMs found. Searched at: 2022-03-27 11:19:01 UTC
53 NOTAM(s) filtered.
Archival search on location 'ZDV' and date '2021-08-24'. 0 NOTAM(s) found.
Change Search
Archive Search
Date
2021-08-24
Search
History
Location
ZDV
Table
Filters 53
Sort
Count 53
No NOTAMs found. Searched at: 2022-03-27 11:20:20 UTC
61 NOTAM(s) filtered.
Archival search on location 'ZDV' and date '2021-08-25'. 0 NOTAM(s) found.
Change Search
Archive Search
Date
2021-08-25
Search
History
Location
ZDV
Table
Filters 61
Sort
Count 61
No NOTAMs found. Searched at: 2022-03-27 11:20:43 UTC
55 NOTAM(s) filtered.
Archival search on location 'ZDV' and date '2021-08-26'. 0 NOTAM(s) found.
Change Search
Archive Search
Date
2021-08-26
Search
History
Location
ZDV
Table
Filters 55
Sort
Count 55
No NOTAMs found. Searched at: 2022-03-27 11:21:10 UTC
51 NOTAM(s) filtered.
Archival search on location 'ZDV' and date '2021-08-27'. 0 NOTAM(s) found.
Change Search
Archive Search
Date
2021-08-27
Search
History
Location
ZDV
Table
Filters 51
Sort
Count 51
No NOTAMs found. Searched at: 2022-03-27 11:21:35 UTC
43 NOTAM(s) filtered.
Archival search on location 'ZDV' and date '2021-08-28'. 0 NOTAM(s) found.
Change Search
Archive Search
Date
2021-08-28
Search
History
Location
ZDV
Table
Filters 43
Sort
Count 43
No NOTAMs found. Searched at: 2022-03-27 11:22:01 UTC
42 NOTAM(s) filtered.
Archival search on location 'ZDV' and date '2021-08-29'. 0 NOTAM(s) found.
Change Search
Archive Search
Date
2021-08-29
Search
History
Location
ZDV
Table
Filters 42
Sort
Count 42
No NOTAMs found. Searched at: 2022-03-27 11:25:48 UTC
66 NOTAM(s) filtered.
Archival search on location 'ZDV' and date '2021-09-08'. 0 NOTAM(s) found.
Change Search
Archive Search
Date
2021-09-08
Search
History
Location
ZDV
Table
Filters 66
Sort
Count 66
No NOTAMs found. Searched at: 2022-03-27 11:26:20 UTC
67 NOTAM(s) filtered.
Archival search on location 'ZDV' and date '2021-09-09'. 0 NOTAM(s) found.
Change Search
Archive Search
Date
2021-09-09
Search
History
Location
ZDV
Table
Filters 67
Sort
Count 67
No NOTAMs found. Searched at: 2022-03-27 11:26:47 UTC
64 NOTAM(s) filtered.
Archival search on location 'ZDV' and date '2021-09-10'. 0 NOTAM(s) found.
Change Search
Archive Search
Date
2021-09-10
Search
History
Location
ZDV
Table
Filters 64
Sort
Count 64

8.6. VBA Log

What follows is the original VBA log, starting with ZLA without execution date and in the opposite direction back then. Together with the screenshots from 8.5 it proves that fetched TFR data is complete for all 10 FIRs over the observed timespan.

```
ZLA, 2021-09-23
ZLA, 2021-09-22
ZLA, 2021-09-21
ZLA, 2021-09-20
ZLA, 2021-09-19
ZLA, 2021-09-18
ZLA, 2021-09-17
ZLA, 2021-09-16
ZLA, 2021-09-15
ZLA, 2021-09-14
ZLA, 2021-09-13
ZLA, 2021-09-12
ZLA, 2021-09-11
ZLA, 2021-09-10
ZLA, 2021-09-09
ZLA, 2021-09-08
ZLA, 2021-09-07
ZLA, 2021-09-06
ZLA, 2021-09-05
ZLA, 2021-09-04
ZLA, 2021-09-03
ZLA, 2021-09-02
ZLA, 2021-09-01
ZLA, 2021-08-31
ZLA, 2021-08-30
ZLA, 2021-08-29
ZLA, 2021-08-28
ZLA, 2021-08-27
ZLA, 2021-08-26
ZLA, 2021-08-25
ZLA, 2021-08-24
ZLA, 2021-08-23
ZLA, 2021-08-22
ZLA, 2021-08-21
ZLA, 2021-08-20
ZLA, 2021-08-19
ZLA, 2021-08-18
ZLA, 2021-08-17
ZLA, 2021-08-16
ZLA, 2021-08-15
ZLA, 2021-08-14
ZLA, 2021-08-13
ZLA, 2021-08-12
ZLA, 2021-08-11
ZLA, 2021-08-10
ZLA, 2021-08-09
ZLA, 2021-08-08
ZLA, 2021-08-07
ZLA, 2021-08-06
ZLA, 2021-08-05
ZLA, 2021-08-04
ZLA, 2021-08-03
ZLA, 2021-08-02
ZLA, 2021-08-01
25.03.2022 11:50:42 : ZOA, 2021-09-01
25.03.2022 11:51:03 : ZOA, 2021-09-02
25.03.2022 11:51:10 : ZOA, 2021-09-03
25.03.2022 11:51:16 : ZOA, 2021-09-04
25.03.2022 11:51:22 : ZOA, 2021-09-05
25.03.2022 11:51:28 : ZOA, 2021-09-06
25.03.2022 11:51:35 : ZOA, 2021-09-07
25.03.2022 11:51:42 : ZOA, 2021-09-08
25.03.2022 11:51:48 : ZOA, 2021-09-09
25.03.2022 11:51:53 : ZOA, 2021-09-10
25.03.2022 11:52:00 : ZOA, 2021-09-11
25.03.2022 11:52:05 : ZOA, 2021-09-12
25.03.2022 11:52:11 : ZOA, 2021-09-13
25.03.2022 11:52:17 : ZOA, 2021-09-14
25.03.2022 11:52:25 : ZOA, 2021-09-15
25.03.2022 11:52:35 : ZOA, 2021-09-16
25.03.2022 11:52:41 : ZOA, 2021-09-17
25.03.2022 11:52:51 : ZOA, 2021-09-18
25.03.2022 11:52:58 : ZOA, 2021-09-19
25.03.2022 11:53:10 : ZOA, 2021-09-20
25.03.2022 11:53:18 : ZOA, 2021-09-21
25.03.2022 11:53:24 : ZOA, 2021-09-22
25.03.2022 11:53:31 : ZOA, 2021-09-23
25.03.2022 11:53:39 : ZOA, 2021-09-24
25.03.2022 11:53:44 : ZOA, 2021-09-25
25.03.2022 11:53:50 : ZOA, 2021-09-26
25.03.2022 11:53:58 : ZOA, 2021-09-27
25.03.2022 11:54:04 : ZOA, 2021-09-28
25.03.2022 11:54:10 : ZOA, 2021-09-29
25.03.2022 11:54:16 : ZOA, 2021-09-30
25.03.2022 11:54:22 : ZOA, 2021-10-01
25.03.2022 11:54:28 : ZOA, 2021-10-02
25.03.2022 11:54:33 : ZOA, 2021-10-03
25.03.2022 11:54:38 : ZOA, 2021-10-04
25.03.2022 11:54:44 : ZOA, 2021-10-05
25.03.2022 11:54:49 : ZOA, 2021-10-06
25.03.2022 11:54:54 : ZOA, 2021-10-07
25.03.2022 11:55:00 : ZOA, 2021-10-08
25.03.2022 11:55:06 : ZOA, 2021-10-09
25.03.2022 11:55:12 : ZOA, 2021-10-10
25.03.2022 11:55:18 : ZOA, 2021-10-11
25.03.2022 11:55:23 : ZOA, 2021-10-12
25.03.2022 11:55:29 : ZOA, 2021-10-13
25.03.2022 11:55:34 : ZOA, 2021-10-14
25.03.2022 11:55:40 : ZOA, 2021-10-15
25.03.2022 11:55:45 : ZOA, 2021-10-16
25.03.2022 11:55:52 : ZOA, 2021-10-17
25.03.2022 11:55:59 : ZOA, 2021-10-18
25.03.2022 11:56:05 : ZOA, 2021-10-19
25.03.2022 11:56:11 : ZOA, 2021-10-20
25.03.2022 11:56:18 : ZOA, 2021-10-21
25.03.2022 11:56:24 : ZOA, 2021-10-22
25.03.2022 11:56:29 : ZOA, 2021-10-23
25.03.2022 11:56:35 : ZOA, 2021-10-24
25.03.2022 11:56:43 : ZOA, 2021-10-25
25.03.2022 11:56:49 : ZOA, 2021-10-26
25.03.2022 11:56:55 : ZOA, 2021-10-27
25.03.2022 11:57:01 : ZOA, 2021-10-28
25.03.2022 11:57:06 : ZOA, 2021-10-29
25.03.2022 11:57:12 : ZOA, 2021-10-30
25.03.2022 11:57:18 : ZOA, 2021-10-31
25.03.2022 13:22:19 : ZLA, 2021-09-24
25.03.2022 13:22:28 : ZLA, 2021-09-25
25.03.2022 13:22:40 : ZLA, 2021-09-26
25.03.2022 13:22:46 : ZLA, 2021-09-27
25.03.2022 13:23:00 : ZLA, 2021-09-28
25.03.2022 13:24:13 : ZLA, 2021-09-29
25.03.2022 13:24:29 : ZLA, 2021-09-30
25.03.2022 13:24:35 : ZLA, 2021-10-01
25.03.2022 13:24:41 : ZLA, 2021-10-02
25.03.2022 13:25:01 : ZLA, 2021-10-03
25.03.2022 13:25:08 : ZLA, 2021-10-04
25.03.2022 13:25:13 : ZLA, 2021-10-05
25.03.2022 13:25:19 : ZLA, 2021-10-06
25.03.2022 13:25:25 : ZLA, 2021-10-07
25.03.2022 13:25:30 : ZLA, 2021-10-08
```

25.03.2022 13:25:36 : ZLA, 2021-10-09
25.03.2022 14:09:24 : ZLA, 2021-10-10
25.03.2022 14:21:12 : ZLA, 2021-10-11
25.03.2022 14:21:22 : ZLA, 2021-10-12
25.03.2022 14:21:28 : ZLA, 2021-10-13
25.03.2022 14:21:34 : ZLA, 2021-10-14
25.03.2022 14:21:39 : ZLA, 2021-10-15
25.03.2022 14:21:45 : ZLA, 2021-10-16
25.03.2022 14:21:51 : ZLA, 2021-10-17
25.03.2022 14:21:57 : ZLA, 2021-10-18
25.03.2022 14:22:03 : ZLA, 2021-10-19
25.03.2022 14:22:09 : ZLA, 2021-10-20
25.03.2022 14:22:14 : ZLA, 2021-10-21
25.03.2022 14:22:20 : ZLA, 2021-10-22
25.03.2022 14:22:27 : ZLA, 2021-10-23
25.03.2022 14:22:33 : ZLA, 2021-10-24
25.03.2022 14:22:40 : ZLA, 2021-10-25
25.03.2022 14:22:46 : ZLA, 2021-10-26
25.03.2022 14:22:51 : ZLA, 2021-10-27
25.03.2022 14:22:56 : ZLA, 2021-10-28
25.03.2022 14:23:03 : ZLA, 2021-10-29
25.03.2022 14:23:09 : ZLA, 2021-10-30
25.03.2022 14:23:15 : ZLA, 2021-10-31
28.03.2022 22:23:37 : ZSE, 2021-08-01
28.03.2022 22:27:39 : ZSE, 2021-08-02
28.03.2022 22:28:25 : ZSE, 2021-08-03
28.03.2022 22:28:25 : ZSE, 2021-08-04
28.03.2022 22:28:25 : ZSE, 2021-08-05
28.03.2022 22:28:25 : ZSE, 2021-08-06
28.03.2022 22:28:25 : ZSE, 2021-08-07
28.03.2022 22:28:26 : ZSE, 2021-08-08
28.03.2022 22:28:26 : ZSE, 2021-08-09
28.03.2022 22:28:26 : ZSE, 2021-08-10
28.03.2022 22:28:26 : ZSE, 2021-08-11
28.03.2022 22:28:26 : ZSE, 2021-08-12
28.03.2022 22:28:26 : ZSE, 2021-08-13
28.03.2022 22:28:27 : ZSE, 2021-08-14
28.03.2022 22:28:27 : ZSE, 2021-08-15
28.03.2022 22:28:27 : ZSE, 2021-08-16
28.03.2022 22:28:27 : ZSE, 2021-08-17
28.03.2022 22:28:27 : ZSE, 2021-08-18
28.03.2022 22:28:27 : ZSE, 2021-08-19
28.03.2022 22:28:28 : ZSE, 2021-08-20
28.03.2022 22:28:28 : ZSE, 2021-08-21
28.03.2022 22:28:28 : ZSE, 2021-08-22
28.03.2022 22:28:28 : ZSE, 2021-08-23
28.03.2022 22:28:28 : ZSE, 2021-08-24
28.03.2022 22:28:28 : ZSE, 2021-08-25
28.03.2022 22:28:29 : ZSE, 2021-08-26
28.03.2022 22:28:29 : ZSE, 2021-08-27
28.03.2022 22:28:29 : ZSE, 2021-08-28
28.03.2022 22:28:29 : ZSE, 2021-08-29
28.03.2022 22:28:29 : ZSE, 2021-08-30
28.03.2022 22:28:29 : ZSE, 2021-08-31
28.03.2022 22:28:30 : ZSE, 2021-09-01
28.03.2022 22:28:30 : ZSE, 2021-09-02
28.03.2022 22:28:30 : ZSE, 2021-09-03
28.03.2022 22:28:30 : ZSE, 2021-09-04
28.03.2022 22:28:30 : ZSE, 2021-09-05
28.03.2022 22:28:30 : ZSE, 2021-09-06
28.03.2022 22:28:31 : ZSE, 2021-09-07
28.03.2022 22:28:31 : ZSE, 2021-09-08
28.03.2022 22:28:31 : ZSE, 2021-09-09
28.03.2022 22:28:31 : ZSE, 2021-09-10
28.03.2022 22:28:31 : ZSE, 2021-09-11
28.03.2022 22:28:31 : ZSE, 2021-09-12
28.03.2022 22:28:32 : ZSE, 2021-09-13
28.03.2022 22:28:32 : ZSE, 2021-09-14
28.03.2022 22:28:32 : ZSE, 2021-09-15
28.03.2022 22:28:32 : ZSE, 2021-09-16
28.03.2022 22:28:32 : ZSE, 2021-09-17
28.03.2022 22:28:32 : ZSE, 2021-09-18
28.03.2022 22:28:33 : ZSE, 2021-09-19
28.03.2022 22:28:33 : ZSE, 2021-09-20
28.03.2022 22:28:33 : ZSE, 2021-09-21
28.03.2022 22:28:34 : ZSE, 2021-09-22
28.03.2022 22:28:34 : ZSE, 2021-09-23
28.03.2022 22:28:34 : ZSE, 2021-09-24
28.03.2022 22:28:34 : ZSE, 2021-09-25
28.03.2022 22:28:35 : ZSE, 2021-09-26
28.03.2022 22:28:35 : ZSE, 2021-09-27
28.03.2022 22:28:35 : ZSE, 2021-09-28
28.03.2022 22:28:35 : ZSE, 2021-09-29
28.03.2022 22:28:35 : ZSE, 2021-09-30
28.03.2022 22:28:35 : ZSE, 2021-10-01
28.03.2022 22:28:35 : ZSE, 2021-10-02
28.03.2022 22:28:36 : ZSE, 2021-10-03
28.03.2022 22:28:36 : ZSE, 2021-10-04
28.03.2022 22:28:36 : ZSE, 2021-10-05
28.03.2022 22:28:36 : ZSE, 2021-10-06
28.03.2022 22:28:36 : ZSE, 2021-10-07
28.03.2022 22:28:36 : ZSE, 2021-10-08
28.03.2022 22:28:37 : ZSE, 2021-10-09
28.03.2022 22:28:37 : ZSE, 2021-10-10
28.03.2022 22:28:37 : ZSE, 2021-10-11
28.03.2022 22:28:37 : ZSE, 2021-10-12
28.03.2022 22:28:37 : ZSE, 2021-10-13
28.03.2022 22:28:37 : ZSE, 2021-10-14
28.03.2022 22:28:38 : ZSE, 2021-10-15
28.03.2022 22:28:38 : ZSE, 2021-10-16
28.03.2022 22:28:38 : ZSE, 2021-10-17
28.03.2022 22:28:38 : ZSE, 2021-10-18
28.03.2022 22:28:38 : ZSE, 2021-10-19
28.03.2022 22:28:39 : ZSE, 2021-10-20
28.03.2022 22:28:39 : ZSE, 2021-10-31
31.03.2022 13:25:24 : ZLC, 2021-08-01
31.03.2022 13:25:50 : ZLC, 2021-08-02
31.03.2022 13:25:50 : ZLC, 2021-08-03
31.03.2022 13:25:50 : ZLC, 2021-08-04
31.03.2022 13:25:50 : ZLC, 2021-08-05
31.03.2022 13:25:50 : ZLC, 2021-08-06
31.03.2022 13:25:50 : ZLC, 2021-08-07
31.03.2022 13:25:51 : ZLC, 2021-08-08
31.03.2022 13:25:51 : ZLC, 2021-08-09
31.03.2022 13:25:51 : ZLC, 2021-08-10
31.03.2022 13:25:51 : ZLC, 2021-08-11
31.03.2022 13:25:51 : ZLC, 2021-08-12
31.03.2022 13:25:51 : ZLC, 2021-08-13
31.03.2022 13:25:51 : ZLC, 2021-08-14
31.03.2022 13:25:52 : ZLC, 2021-08-15
31.03.2022 13:25:52 : ZLC, 2021-08-16
31.03.2022 13:25:52 : ZLC, 2021-08-17
31.03.2022 13:25:52 : ZLC, 2021-08-18
31.03.2022 13:25:52 : ZLC, 2021-08-19
31.03.2022 13:25:52 : ZLC, 2021-08-20
31.03.2022 13:25:53 : ZLC, 2021-08-21
31.03.2022 13:25:53 : ZLC, 2021-08-22
31.03.2022 13:25:53 : ZLC, 2021-08-23
31.03.2022 13:25:53 : ZLC, 2021-08-24
31.03.2022 13:25:53 : ZLC, 2021-08-25
31.03.2022 13:25:53 : ZLC, 2021-08-26
31.03.2022 13:25:53 : ZLC, 2021-08-27
31.03.2022 13:25:54 : ZLC, 2021-08-28
31.03.2022 13:25:54 : ZLC, 2021-08-29
31.03.2022 13:25:54 : ZLC, 2021-08-30
31.03.2022 13:25:54 : ZLC, 2021-08-31
31.03.2022 13:25:54 : ZLC, 2021-09-01
31.03.2022 13:25:54 : ZLC, 2021-09-02
31.03.2022 13:25:55 : ZLC, 2021-09-03
31.03.2022 13:25:55 : ZLC, 2021-09-04
31.03.2022 13:25:55 : ZLC, 2021-09-05
31.03.2022 13:25:55 : ZLC, 2021-09-06

31.03.2022 13:25:55 : ZLC, 2021-09-07
31.03.2022 13:25:55 : ZLC, 2021-09-08
31.03.2022 13:25:55 : ZLC, 2021-09-09
31.03.2022 13:25:56 : ZLC, 2021-09-10
31.03.2022 13:25:56 : ZLC, 2021-09-11
31.03.2022 13:25:56 : ZLC, 2021-09-12
31.03.2022 13:25:56 : ZLC, 2021-09-13
31.03.2022 13:25:56 : ZLC, 2021-09-14
31.03.2022 13:25:56 : ZLC, 2021-09-15
31.03.2022 13:25:57 : ZLC, 2021-09-16
31.03.2022 13:25:57 : ZLC, 2021-09-17
31.03.2022 13:25:58 : ZLC, 2021-09-18
31.03.2022 13:25:58 : ZLC, 2021-09-19
31.03.2022 13:25:58 : ZLC, 2021-09-20
31.03.2022 13:25:58 : ZLC, 2021-09-21
31.03.2022 13:25:58 : ZLC, 2021-09-22
31.03.2022 13:25:59 : ZLC, 2021-09-23
31.03.2022 13:25:59 : ZLC, 2021-09-24
31.03.2022 13:25:59 : ZLC, 2021-09-25
31.03.2022 13:25:59 : ZLC, 2021-09-26
31.03.2022 13:25:59 : ZLC, 2021-09-27
31.03.2022 13:25:59 : ZLC, 2021-09-28
31.03.2022 13:25:59 : ZLC, 2021-09-29
31.03.2022 13:26:00 : ZLC, 2021-09-30
31.03.2022 13:26:00 : ZLC, 2021-10-01
31.03.2022 13:26:00 : ZLC, 2021-10-02
31.03.2022 13:26:00 : ZLC, 2021-10-03
31.03.2022 13:26:00 : ZLC, 2021-10-04
31.03.2022 13:26:00 : ZLC, 2021-10-05
31.03.2022 13:26:01 : ZLC, 2021-10-06
31.03.2022 13:26:01 : ZLC, 2021-10-07
31.03.2022 13:26:01 : ZLC, 2021-10-08
31.03.2022 13:26:01 : ZLC, 2021-10-09
31.03.2022 13:26:01 : ZLC, 2021-10-10
31.03.2022 13:26:01 : ZLC, 2021-10-11
31.03.2022 13:26:01 : ZLC, 2021-10-12
31.03.2022 13:26:02 : ZLC, 2021-10-13
31.03.2022 13:26:02 : ZLC, 2021-10-14
31.03.2022 13:26:02 : ZLC, 2021-10-15
31.03.2022 13:26:02 : ZLC, 2021-10-16
31.03.2022 13:26:02 : ZLC, 2021-10-17
31.03.2022 13:26:02 : ZLC, 2021-10-18
31.03.2022 13:26:02 : ZLC, 2021-10-19
31.03.2022 13:26:03 : ZLC, 2021-10-20
31.03.2022 13:26:03 : ZLC, 2021-10-21
31.03.2022 13:26:03 : ZLC, 2021-10-22
31.03.2022 13:26:03 : ZLC, 2021-10-23
31.03.2022 13:26:03 : ZLC, 2021-10-24
31.03.2022 13:26:03 : ZLC, 2021-10-25
31.03.2022 13:26:03 : ZLC, 2021-10-26
31.03.2022 13:26:04 : ZLC, 2021-10-27
31.03.2022 13:26:04 : ZLC, 2021-10-28
31.03.2022 13:26:04 : ZLC, 2021-10-29
31.03.2022 13:26:04 : ZLC, 2021-10-30
31.03.2022 13:26:04 : ZLC, 2021-10-31
31.03.2022 13:32:17 : ZDV, 2021-08-01
31.03.2022 13:32:22 : ZDV, 2021-08-02
31.03.2022 13:32:23 : ZDV, 2021-08-03
31.03.2022 13:32:23 : ZDV, 2021-08-04
31.03.2022 13:32:23 : ZDV, 2021-08-05
31.03.2022 13:32:23 : ZDV, 2021-08-06
31.03.2022 13:32:23 : ZDV, 2021-08-07
31.03.2022 13:32:23 : ZDV, 2021-08-08
31.03.2022 13:32:24 : ZDV, 2021-08-09
31.03.2022 13:32:24 : ZDV, 2021-08-10
31.03.2022 13:32:24 : ZDV, 2021-08-11
31.03.2022 13:32:24 : ZDV, 2021-08-12
31.03.2022 13:32:24 : ZDV, 2021-08-13
31.03.2022 13:32:24 : ZDV, 2021-08-14
31.03.2022 13:32:24 : ZDV, 2021-08-15
31.03.2022 13:32:25 : ZDV, 2021-08-16
31.03.2022 13:32:25 : ZDV, 2021-08-17
31.03.2022 13:32:25 : ZDV, 2021-08-30
31.03.2022 13:32:25 : ZDV, 2021-08-31
31.03.2022 13:32:25 : ZDV, 2021-09-01
31.03.2022 13:32:25 : ZDV, 2021-09-02
31.03.2022 13:32:25 : ZDV, 2021-09-03
31.03.2022 13:32:26 : ZDV, 2021-09-04
31.03.2022 13:32:26 : ZDV, 2021-09-05
31.03.2022 13:32:26 : ZDV, 2021-09-06
31.03.2022 13:32:26 : ZDV, 2021-09-07
31.03.2022 13:32:26 : ZDV, 2021-09-12
31.03.2022 13:32:26 : ZDV, 2021-09-13
31.03.2022 13:32:27 : ZDV, 2021-09-14
31.03.2022 13:32:27 : ZDV, 2021-09-17
31.03.2022 13:32:27 : ZDV, 2021-09-18
31.03.2022 13:32:27 : ZDV, 2021-09-19
31.03.2022 13:32:27 : ZDV, 2021-09-20
31.03.2022 13:32:27 : ZDV, 2021-09-21
31.03.2022 13:32:27 : ZDV, 2021-09-28
31.03.2022 13:32:28 : ZDV, 2021-09-29
31.03.2022 13:32:28 : ZDV, 2021-09-30
31.03.2022 13:32:28 : ZDV, 2021-10-01
31.03.2022 13:32:28 : ZDV, 2021-10-02
31.03.2022 13:32:28 : ZDV, 2021-10-03
31.03.2022 13:32:28 : ZDV, 2021-10-04
31.03.2022 13:32:29 : ZDV, 2021-10-05
31.03.2022 13:32:29 : ZDV, 2021-10-06
31.03.2022 13:32:29 : ZDV, 2021-10-07
31.03.2022 13:32:29 : ZDV, 2021-10-08
31.03.2022 13:32:29 : ZDV, 2021-10-09
31.03.2022 13:32:29 : ZDV, 2021-10-10
31.03.2022 13:32:29 : ZDV, 2021-10-11
31.03.2022 13:32:30 : ZDV, 2021-10-12
31.03.2022 13:32:30 : ZDV, 2021-10-13
31.03.2022 13:32:30 : ZDV, 2021-10-14
31.03.2022 13:32:30 : ZDV, 2021-10-15
31.03.2022 13:32:30 : ZDV, 2021-10-16
31.03.2022 13:32:30 : ZDV, 2021-10-17
31.03.2022 13:32:31 : ZDV, 2021-10-18
31.03.2022 13:32:31 : ZDV, 2021-10-19
31.03.2022 13:32:31 : ZDV, 2021-10-20
31.03.2022 13:32:31 : ZDV, 2021-10-21
31.03.2022 13:32:31 : ZDV, 2021-10-22
31.03.2022 13:32:31 : ZDV, 2021-10-23
31.03.2022 13:32:31 : ZDV, 2021-10-24
31.03.2022 13:32:32 : ZDV, 2021-10-25
31.03.2022 13:32:32 : ZDV, 2021-10-26
31.03.2022 13:32:32 : ZDV, 2021-10-27
31.03.2022 13:32:32 : ZDV, 2021-10-28
31.03.2022 13:32:32 : ZDV, 2021-10-29
31.03.2022 13:32:32 : ZDV, 2021-10-30
31.03.2022 13:32:32 : ZDV, 2021-10-31
31.03.2022 13:35:18 : ZAB, 2021-08-01
31.03.2022 13:35:23 : ZAB, 2021-08-02
31.03.2022 13:35:23 : ZAB, 2021-08-03
31.03.2022 13:35:23 : ZAB, 2021-08-04
31.03.2022 13:35:23 : ZAB, 2021-08-05
31.03.2022 13:35:23 : ZAB, 2021-08-06
31.03.2022 13:35:24 : ZAB, 2021-08-07
31.03.2022 13:35:24 : ZAB, 2021-08-08
31.03.2022 13:35:24 : ZAB, 2021-08-09
31.03.2022 13:35:24 : ZAB, 2021-08-10
31.03.2022 13:35:24 : ZAB, 2021-08-11
31.03.2022 13:35:25 : ZAB, 2021-08-12
31.03.2022 13:35:25 : ZAB, 2021-08-13
31.03.2022 13:35:25 : ZAB, 2021-08-14
31.03.2022 13:35:25 : ZAB, 2021-08-15
31.03.2022 13:35:25 : ZAB, 2021-08-16
31.03.2022 13:35:25 : ZAB, 2021-08-17
31.03.2022 13:35:25 : ZAB, 2021-08-18
31.03.2022 13:35:26 : ZAB, 2021-08-19

31.03.2022 13:35:26 : ZAB, 2021-08-20
31.03.2022 13:35:26 : ZAB, 2021-08-21
31.03.2022 13:35:26 : ZAB, 2021-08-22
31.03.2022 13:35:26 : ZAB, 2021-08-23
31.03.2022 13:35:26 : ZAB, 2021-08-24
31.03.2022 13:35:26 : ZAB, 2021-08-25
31.03.2022 13:35:27 : ZAB, 2021-08-26
31.03.2022 13:35:27 : ZAB, 2021-08-27
31.03.2022 13:35:27 : ZAB, 2021-08-28
31.03.2022 13:35:27 : ZAB, 2021-08-29
31.03.2022 13:35:27 : ZAB, 2021-08-30
31.03.2022 13:35:27 : ZAB, 2021-08-31
31.03.2022 13:35:28 : ZAB, 2021-09-01
31.03.2022 13:35:28 : ZAB, 2021-09-02
31.03.2022 13:35:28 : ZAB, 2021-09-03
31.03.2022 13:35:28 : ZAB, 2021-09-04
31.03.2022 13:35:28 : ZAB, 2021-09-05
31.03.2022 13:35:28 : ZAB, 2021-09-06
31.03.2022 13:35:28 : ZAB, 2021-09-07
31.03.2022 13:35:29 : ZAB, 2021-09-08
31.03.2022 13:35:29 : ZAB, 2021-09-09
31.03.2022 13:35:29 : ZAB, 2021-09-10
31.03.2022 13:35:29 : ZAB, 2021-09-11
31.03.2022 13:35:29 : ZAB, 2021-09-12
31.03.2022 13:35:29 : ZAB, 2021-09-13
31.03.2022 13:35:30 : ZAB, 2021-09-14
31.03.2022 13:35:30 : ZAB, 2021-09-15
31.03.2022 13:35:30 : ZAB, 2021-09-16
31.03.2022 13:35:30 : ZAB, 2021-09-17
31.03.2022 13:35:30 : ZAB, 2021-09-18
31.03.2022 13:35:30 : ZAB, 2021-09-19
31.03.2022 13:35:31 : ZAB, 2021-09-20
31.03.2022 13:35:31 : ZAB, 2021-09-21
31.03.2022 13:35:31 : ZAB, 2021-09-22
31.03.2022 13:35:31 : ZAB, 2021-09-23
31.03.2022 13:35:31 : ZAB, 2021-09-24
31.03.2022 13:35:31 : ZAB, 2021-09-25
31.03.2022 13:35:31 : ZAB, 2021-09-26
31.03.2022 13:35:32 : ZAB, 2021-09-27
31.03.2022 13:35:32 : ZAB, 2021-09-28
31.03.2022 13:35:32 : ZAB, 2021-09-29
31.03.2022 13:35:32 : ZAB, 2021-09-30
31.03.2022 13:35:32 : ZAB, 2021-10-01
31.03.2022 13:35:32 : ZAB, 2021-10-02
31.03.2022 13:35:33 : ZAB, 2021-10-03
31.03.2022 13:35:33 : ZAB, 2021-10-04
31.03.2022 13:35:33 : ZAB, 2021-10-05
31.03.2022 13:35:33 : ZAB, 2021-10-06
31.03.2022 13:35:33 : ZAB, 2021-10-07
31.03.2022 13:35:33 : ZAB, 2021-10-08
31.03.2022 13:35:34 : ZAB, 2021-10-09
31.03.2022 13:35:34 : ZAB, 2021-10-10
31.03.2022 13:35:34 : ZAB, 2021-10-11
31.03.2022 13:35:34 : ZAB, 2021-10-12
31.03.2022 13:35:34 : ZAB, 2021-10-13
31.03.2022 13:35:34 : ZAB, 2021-10-14
31.03.2022 13:35:35 : ZAB, 2021-10-15
31.03.2022 13:35:35 : ZAB, 2021-10-16
31.03.2022 13:35:35 : ZAB, 2021-10-17
31.03.2022 13:35:35 : ZAB, 2021-10-18
31.03.2022 13:35:35 : ZAB, 2021-10-19
31.03.2022 13:35:35 : ZAB, 2021-10-20
31.03.2022 13:35:35 : ZAB, 2021-10-21
31.03.2022 13:35:36 : ZAB, 2021-10-22
31.03.2022 13:35:36 : ZAB, 2021-10-23
31.03.2022 13:35:36 : ZAB, 2021-10-24
31.03.2022 13:35:36 : ZAB, 2021-10-25
31.03.2022 13:35:36 : ZAB, 2021-10-26
31.03.2022 13:35:36 : ZAB, 2021-10-27
31.03.2022 13:35:36 : ZAB, 2021-10-28
31.03.2022 13:35:37 : ZAB, 2021-10-29
31.03.2022 13:35:37 : ZAB, 2021-10-30
31.03.2022 13:35:37 : ZAB, 2021-10-31
31.03.2022 13:36:33 : ZHU, 2021-08-02
31.03.2022 13:36:35 : ZHU, 2021-08-03
31.03.2022 13:36:35 : ZHU, 2021-08-04
31.03.2022 13:36:35 : ZHU, 2021-08-05
31.03.2022 13:36:36 : ZHU, 2021-08-06
31.03.2022 13:36:36 : ZHU, 2021-08-07
31.03.2022 13:36:36 : ZHU, 2021-08-08
31.03.2022 13:36:36 : ZHU, 2021-08-09
31.03.2022 13:36:36 : ZHU, 2021-08-10
31.03.2022 13:36:36 : ZHU, 2021-08-11
31.03.2022 13:36:36 : ZHU, 2021-08-12
31.03.2022 13:36:37 : ZHU, 2021-08-13
31.03.2022 13:36:37 : ZHU, 2021-08-14
31.03.2022 13:36:37 : ZHU, 2021-08-15
31.03.2022 13:36:37 : ZHU, 2021-08-16
31.03.2022 13:36:37 : ZHU, 2021-08-17
31.03.2022 13:36:37 : ZHU, 2021-08-18
31.03.2022 13:36:38 : ZHU, 2021-08-19
31.03.2022 13:36:38 : ZHU, 2021-08-20
31.03.2022 13:36:38 : ZHU, 2021-08-21
31.03.2022 13:36:38 : ZHU, 2021-08-22
31.03.2022 13:36:38 : ZHU, 2021-08-23
31.03.2022 13:36:38 : ZHU, 2021-08-24
31.03.2022 13:36:38 : ZHU, 2021-08-25
31.03.2022 13:36:39 : ZHU, 2021-08-26
31.03.2022 13:36:39 : ZHU, 2021-08-27
31.03.2022 13:36:39 : ZHU, 2021-08-28
31.03.2022 13:36:39 : ZHU, 2021-08-29
31.03.2022 13:36:39 : ZHU, 2021-08-30
31.03.2022 13:36:39 : ZHU, 2021-09-01
31.03.2022 13:36:40 : ZHU, 2021-09-02
31.03.2022 13:36:40 : ZHU, 2021-09-03
31.03.2022 13:36:40 : ZHU, 2021-09-04
31.03.2022 13:36:40 : ZHU, 2021-09-05
31.03.2022 13:36:40 : ZHU, 2021-09-06
31.03.2022 13:36:40 : ZHU, 2021-09-07
31.03.2022 13:36:40 : ZHU, 2021-09-08
31.03.2022 13:36:41 : ZHU, 2021-09-09
31.03.2022 13:36:41 : ZHU, 2021-09-10
31.03.2022 13:36:41 : ZHU, 2021-09-11
31.03.2022 13:36:41 : ZHU, 2021-09-12
31.03.2022 13:36:41 : ZHU, 2021-09-13
31.03.2022 13:36:41 : ZHU, 2021-09-14
31.03.2022 13:36:42 : ZHU, 2021-09-15
31.03.2022 13:36:42 : ZHU, 2021-09-16
31.03.2022 13:36:42 : ZHU, 2021-09-17
31.03.2022 13:36:42 : ZHU, 2021-09-18
31.03.2022 13:36:42 : ZHU, 2021-09-19
31.03.2022 13:36:42 : ZHU, 2021-09-20
31.03.2022 13:36:43 : ZHU, 2021-09-21
31.03.2022 13:36:43 : ZHU, 2021-09-22
31.03.2022 13:36:43 : ZHU, 2021-09-23
31.03.2022 13:36:43 : ZHU, 2021-09-24
31.03.2022 13:36:43 : ZHU, 2021-09-25
31.03.2022 13:36:43 : ZHU, 2021-09-26
31.03.2022 13:36:43 : ZHU, 2021-09-27
31.03.2022 13:36:44 : ZHU, 2021-09-28
31.03.2022 13:36:44 : ZHU, 2021-09-29
31.03.2022 13:36:44 : ZHU, 2021-09-30
31.03.2022 13:36:44 : ZHU, 2021-10-01
31.03.2022 13:36:44 : ZHU, 2021-10-02
31.03.2022 13:36:44 : ZHU, 2021-10-03
31.03.2022 13:36:45 : ZHU, 2021-10-04
31.03.2022 13:36:45 : ZHU, 2021-10-05
31.03.2022 13:36:45 : ZHU, 2021-10-06
31.03.2022 13:36:45 : ZHU, 2021-10-07
31.03.2022 13:36:45 : ZHU, 2021-10-08
31.03.2022 13:36:45 : ZHU, 2021-10-09
31.03.2022 13:36:46 : ZHU, 2021-10-10

31.03.2022 13:36:46 : ZHU, 2021-10-11
31.03.2022 13:36:46 : ZHU, 2021-10-12
31.03.2022 13:36:46 : ZHU, 2021-10-13
31.03.2022 13:36:46 : ZHU, 2021-10-14
31.03.2022 13:36:46 : ZHU, 2021-10-15
31.03.2022 13:36:46 : ZHU, 2021-10-16
31.03.2022 13:36:47 : ZHU, 2021-10-17
31.03.2022 13:36:47 : ZHU, 2021-10-18
31.03.2022 13:36:47 : ZHU, 2021-10-19
31.03.2022 13:36:47 : ZHU, 2021-10-20
31.03.2022 13:36:47 : ZHU, 2021-10-21
31.03.2022 13:36:47 : ZHU, 2021-10-22
31.03.2022 13:36:48 : ZHU, 2021-10-23
31.03.2022 13:36:48 : ZHU, 2021-10-24
31.03.2022 13:36:48 : ZHU, 2021-10-25
31.03.2022 13:36:48 : ZHU, 2021-10-26
31.03.2022 13:36:48 : ZHU, 2021-10-27
31.03.2022 13:36:48 : ZHU, 2021-10-28
31.03.2022 13:36:48 : ZHU, 2021-10-29
31.03.2022 13:36:49 : ZHU, 2021-10-30
31.03.2022 13:36:49 : ZHU, 2021-10-31
31.03.2022 13:37:25 : ZHU, 2021-08-02
31.03.2022 13:38:15 : ZFW, 2021-08-01
31.03.2022 13:38:17 : ZFW, 2021-08-02
31.03.2022 13:38:17 : ZFW, 2021-08-03
31.03.2022 13:38:17 : ZFW, 2021-08-04
31.03.2022 13:38:17 : ZFW, 2021-08-05
31.03.2022 13:38:17 : ZFW, 2021-08-06
31.03.2022 13:38:17 : ZFW, 2021-08-07
31.03.2022 13:38:18 : ZFW, 2021-08-08
31.03.2022 13:38:18 : ZFW, 2021-08-09
31.03.2022 13:38:18 : ZFW, 2021-08-10
31.03.2022 13:38:18 : ZFW, 2021-08-11
31.03.2022 13:38:18 : ZFW, 2021-08-12
31.03.2022 13:38:18 : ZFW, 2021-08-13
31.03.2022 13:38:18 : ZFW, 2021-08-14
31.03.2022 13:38:19 : ZFW, 2021-08-15
31.03.2022 13:38:19 : ZFW, 2021-08-16
31.03.2022 13:38:19 : ZFW, 2021-08-17
31.03.2022 13:38:19 : ZFW, 2021-08-18
31.03.2022 13:38:19 : ZFW, 2021-08-19
31.03.2022 13:38:19 : ZFW, 2021-08-20
31.03.2022 13:38:20 : ZFW, 2021-08-21
31.03.2022 13:38:20 : ZFW, 2021-08-22
31.03.2022 13:38:20 : ZFW, 2021-08-23
31.03.2022 13:38:20 : ZFW, 2021-08-24
31.03.2022 13:38:20 : ZFW, 2021-08-25
31.03.2022 13:38:20 : ZFW, 2021-08-26
31.03.2022 13:38:20 : ZFW, 2021-08-27
31.03.2022 13:38:21 : ZFW, 2021-08-28
31.03.2022 13:38:21 : ZFW, 2021-08-29
31.03.2022 13:38:21 : ZFW, 2021-08-30
31.03.2022 13:38:21 : ZFW, 2021-08-31
31.03.2022 13:38:21 : ZFW, 2021-09-01
31.03.2022 13:38:21 : ZFW, 2021-09-02
31.03.2022 13:38:22 : ZFW, 2021-09-03
31.03.2022 13:38:22 : ZFW, 2021-09-04
31.03.2022 13:38:22 : ZFW, 2021-09-05
31.03.2022 13:38:22 : ZFW, 2021-09-06
31.03.2022 13:38:22 : ZFW, 2021-09-07
31.03.2022 13:38:22 : ZFW, 2021-09-08
31.03.2022 13:38:23 : ZFW, 2021-09-09
31.03.2022 13:38:23 : ZFW, 2021-09-10
31.03.2022 13:38:23 : ZFW, 2021-09-11
31.03.2022 13:38:23 : ZFW, 2021-09-12
31.03.2022 13:38:23 : ZFW, 2021-09-13
31.03.2022 13:38:23 : ZFW, 2021-09-14
31.03.2022 13:38:23 : ZFW, 2021-09-15
31.03.2022 13:38:24 : ZFW, 2021-09-16
31.03.2022 13:38:24 : ZFW, 2021-09-17
31.03.2022 13:38:24 : ZFW, 2021-09-18
31.03.2022 13:38:24 : ZFW, 2021-09-19
31.03.2022 13:38:24 : ZFW, 2021-09-20
31.03.2022 13:38:24 : ZFW, 2021-09-21
31.03.2022 13:38:25 : ZFW, 2021-09-22
31.03.2022 13:38:25 : ZFW, 2021-09-23
31.03.2022 13:38:25 : ZFW, 2021-09-24
31.03.2022 13:38:25 : ZFW, 2021-09-25
31.03.2022 13:38:25 : ZFW, 2021-09-26
31.03.2022 13:38:25 : ZFW, 2021-09-27
31.03.2022 13:38:26 : ZFW, 2021-09-28
31.03.2022 13:38:26 : ZFW, 2021-09-29
31.03.2022 13:38:26 : ZFW, 2021-09-30
31.03.2022 13:38:26 : ZFW, 2021-10-01
31.03.2022 13:38:26 : ZFW, 2021-10-02
31.03.2022 13:38:26 : ZFW, 2021-10-03
31.03.2022 13:38:26 : ZFW, 2021-10-04
31.03.2022 13:38:27 : ZFW, 2021-10-05
31.03.2022 13:38:27 : ZFW, 2021-10-06
31.03.2022 13:38:27 : ZFW, 2021-10-07
31.03.2022 13:38:27 : ZFW, 2021-10-08
31.03.2022 13:38:27 : ZFW, 2021-10-09
31.03.2022 13:38:27 : ZFW, 2021-10-10
31.03.2022 13:38:28 : ZFW, 2021-10-11
31.03.2022 13:38:28 : ZFW, 2021-10-12
31.03.2022 13:38:28 : ZFW, 2021-10-13
31.03.2022 13:38:28 : ZFW, 2021-10-14
31.03.2022 13:38:28 : ZFW, 2021-10-15
31.03.2022 13:38:28 : ZFW, 2021-10-16
31.03.2022 13:38:28 : ZFW, 2021-10-17
31.03.2022 13:38:29 : ZFW, 2021-10-18
31.03.2022 13:38:29 : ZFW, 2021-10-19
31.03.2022 13:38:29 : ZFW, 2021-10-20
31.03.2022 13:38:29 : ZFW, 2021-10-21
31.03.2022 13:38:29 : ZFW, 2021-10-22
31.03.2022 13:38:29 : ZFW, 2021-10-23
31.03.2022 13:38:30 : ZFW, 2021-10-24
31.03.2022 13:38:30 : ZFW, 2021-10-25
31.03.2022 13:38:30 : ZFW, 2021-10-26
31.03.2022 13:38:30 : ZFW, 2021-10-27
31.03.2022 13:38:30 : ZFW, 2021-10-28
31.03.2022 13:38:30 : ZFW, 2021-10-29
31.03.2022 13:38:31 : ZFW, 2021-10-30
31.03.2022 13:38:31 : ZFW, 2021-10-31
31.03.2022 13:38:51 : ZKC, 2021-08-01
31.03.2022 13:38:53 : ZKC, 2021-08-02
31.03.2022 13:38:53 : ZKC, 2021-08-04
31.03.2022 13:38:53 : ZKC, 2021-08-05
31.03.2022 13:38:54 : ZKC, 2021-08-06
31.03.2022 13:38:54 : ZKC, 2021-08-07
31.03.2022 13:38:54 : ZKC, 2021-08-08
31.03.2022 13:38:54 : ZKC, 2021-08-09
31.03.2022 13:38:54 : ZKC, 2021-08-10
31.03.2022 13:38:54 : ZKC, 2021-08-11
31.03.2022 13:38:54 : ZKC, 2021-08-12
31.03.2022 13:38:55 : ZKC, 2021-08-13
31.03.2022 13:38:55 : ZKC, 2021-08-14
31.03.2022 13:38:55 : ZKC, 2021-08-15
31.03.2022 13:38:55 : ZKC, 2021-08-16
31.03.2022 13:38:55 : ZKC, 2021-08-20
31.03.2022 13:38:55 : ZKC, 2021-08-21
31.03.2022 13:38:56 : ZKC, 2021-08-22
31.03.2022 13:38:56 : ZKC, 2021-08-23
31.03.2022 13:38:56 : ZKC, 2021-08-24
31.03.2022 13:38:56 : ZKC, 2021-08-27
31.03.2022 13:38:56 : ZKC, 2021-08-28
31.03.2022 13:38:56 : ZKC, 2021-08-29
31.03.2022 13:38:56 : ZKC, 2021-08-30
31.03.2022 13:38:57 : ZKC, 2021-09-02
31.03.2022 13:38:57 : ZKC, 2021-09-03
31.03.2022 13:38:57 : ZKC, 2021-09-04
31.03.2022 13:38:57 : ZKC, 2021-09-05

31.03.2022 13:38:57 : ZKC, 2021-09-06
31.03.2022 13:38:57 : ZKC, 2021-09-07
31.03.2022 13:38:58 : ZKC, 2021-09-08
31.03.2022 13:38:58 : ZKC, 2021-09-09
31.03.2022 13:38:58 : ZKC, 2021-09-10
31.03.2022 13:38:58 : ZKC, 2021-09-11
31.03.2022 13:38:58 : ZKC, 2021-09-12
31.03.2022 13:38:58 : ZKC, 2021-09-13
31.03.2022 13:38:58 : ZKC, 2021-09-14
31.03.2022 13:38:59 : ZKC, 2021-09-15
31.03.2022 13:38:59 : ZKC, 2021-09-16
31.03.2022 13:38:59 : ZKC, 2021-09-17
31.03.2022 13:38:59 : ZKC, 2021-09-18
31.03.2022 13:38:59 : ZKC, 2021-09-19
31.03.2022 13:38:59 : ZKC, 2021-09-20
31.03.2022 13:39:00 : ZKC, 2021-09-21
31.03.2022 13:39:00 : ZKC, 2021-09-22
31.03.2022 13:39:00 : ZKC, 2021-09-23
31.03.2022 13:39:00 : ZKC, 2021-09-24
31.03.2022 13:39:00 : ZKC, 2021-09-25
31.03.2022 13:39:00 : ZKC, 2021-09-26
31.03.2022 13:39:01 : ZKC, 2021-09-27
31.03.2022 13:39:01 : ZKC, 2021-09-28
31.03.2022 13:39:01 : ZKC, 2021-09-29
31.03.2022 13:39:01 : ZKC, 2021-09-30
31.03.2022 13:39:01 : ZKC, 2021-10-01
31.03.2022 13:39:01 : ZKC, 2021-10-02
31.03.2022 13:39:01 : ZKC, 2021-10-03
31.03.2022 13:39:02 : ZKC, 2021-10-04
31.03.2022 13:39:02 : ZKC, 2021-10-05
31.03.2022 13:39:02 : ZKC, 2021-10-06
31.03.2022 13:39:02 : ZKC, 2021-10-07
31.03.2022 13:39:02 : ZKC, 2021-10-08
31.03.2022 13:39:02 : ZKC, 2021-10-12
31.03.2022 13:39:03 : ZKC, 2021-10-13
31.03.2022 13:39:03 : ZKC, 2021-10-14
31.03.2022 13:39:03 : ZKC, 2021-10-15
31.03.2022 13:39:03 : ZKC, 2021-10-16
31.03.2022 13:39:03 : ZKC, 2021-10-17
31.03.2022 13:39:03 : ZKC, 2021-10-18
31.03.2022 13:39:03 : ZKC, 2021-10-19
31.03.2022 13:39:04 : ZKC, 2021-10-20
31.03.2022 13:39:04 : ZKC, 2021-10-21
31.03.2022 13:39:04 : ZKC, 2021-10-22
31.03.2022 13:39:04 : ZKC, 2021-10-23
31.03.2022 13:39:04 : ZKC, 2021-10-24
31.03.2022 13:39:04 : ZKC, 2021-10-25
31.03.2022 13:40:43 : ZMP, 2021-08-01
31.03.2022 13:40:47 : ZMP, 2021-08-02
31.03.2022 13:40:51 : ZMP, 2021-08-03
31.03.2022 13:40:52 : ZMP, 2021-08-04
31.03.2022 13:40:52 : ZMP, 2021-08-05
31.03.2022 13:40:53 : ZMP, 2021-08-05
31.03.2022 13:40:55 : ZMP, 2021-08-06
31.03.2022 13:40:56 : ZMP, 2021-08-07
31.03.2022 13:40:56 : ZMP, 2021-08-08
31.03.2022 13:40:56 : ZMP, 2021-08-09
31.03.2022 13:40:56 : ZMP, 2021-08-10
31.03.2022 13:40:56 : ZMP, 2021-08-11
31.03.2022 13:40:56 : ZMP, 2021-08-12
31.03.2022 13:40:57 : ZMP, 2021-08-13
31.03.2022 13:40:57 : ZMP, 2021-08-14
31.03.2022 13:40:57 : ZMP, 2021-08-15
31.03.2022 13:40:57 : ZMP, 2021-08-16
31.03.2022 13:40:57 : ZMP, 2021-08-17
31.03.2022 13:40:57 : ZMP, 2021-08-18
31.03.2022 13:40:58 : ZMP, 2021-08-19
31.03.2022 13:40:58 : ZMP, 2021-08-20
31.03.2022 13:40:58 : ZMP, 2021-08-21
31.03.2022 13:40:58 : ZMP, 2021-08-22
31.03.2022 13:40:58 : ZMP, 2021-08-23
31.03.2022 13:40:58 : ZMP, 2021-08-24
31.03.2022 13:40:59 : ZMP, 2021-08-25
31.03.2022 13:40:59 : ZMP, 2021-08-26
31.03.2022 13:40:59 : ZMP, 2021-08-27
31.03.2022 13:40:59 : ZMP, 2021-08-28
31.03.2022 13:40:59 : ZMP, 2021-08-29
31.03.2022 13:41:00 : ZMP, 2021-08-30
31.03.2022 13:41:00 : ZMP, 2021-08-31
31.03.2022 13:41:00 : ZMP, 2021-09-01
31.03.2022 13:41:00 : ZMP, 2021-09-02
31.03.2022 13:41:00 : ZMP, 2021-09-03
31.03.2022 13:41:00 : ZMP, 2021-09-04
31.03.2022 13:41:01 : ZMP, 2021-09-05
31.03.2022 13:41:01 : ZMP, 2021-09-06
31.03.2022 13:41:01 : ZMP, 2021-09-07
31.03.2022 13:41:01 : ZMP, 2021-09-08
31.03.2022 13:41:01 : ZMP, 2021-09-09
31.03.2022 13:41:01 : ZMP, 2021-09-10
31.03.2022 13:41:02 : ZMP, 2021-09-11
31.03.2022 13:41:02 : ZMP, 2021-09-12
31.03.2022 13:41:02 : ZMP, 2021-09-13
31.03.2022 13:41:02 : ZMP, 2021-09-14
31.03.2022 13:41:02 : ZMP, 2021-09-15
31.03.2022 13:41:02 : ZMP, 2021-09-16
31.03.2022 13:41:03 : ZMP, 2021-09-17
31.03.2022 13:41:03 : ZMP, 2021-09-18
31.03.2022 13:41:03 : ZMP, 2021-09-19
31.03.2022 13:41:03 : ZMP, 2021-09-20
31.03.2022 13:41:03 : ZMP, 2021-09-21
31.03.2022 13:41:03 : ZMP, 2021-09-22
31.03.2022 13:41:04 : ZMP, 2021-09-23
31.03.2022 13:41:04 : ZMP, 2021-09-24
31.03.2022 13:41:04 : ZMP, 2021-09-25
31.03.2022 13:41:04 : ZMP, 2021-09-26
31.03.2022 13:41:04 : ZMP, 2021-09-27
31.03.2022 13:41:04 : ZMP, 2021-09-28
31.03.2022 13:41:05 : ZMP, 2021-09-29
31.03.2022 13:41:05 : ZMP, 2021-09-30
31.03.2022 13:41:05 : ZMP, 2021-10-01
31.03.2022 13:41:05 : ZMP, 2021-10-02
31.03.2022 13:41:05 : ZMP, 2021-10-03
31.03.2022 13:41:05 : ZMP, 2021-10-04
31.03.2022 13:41:06 : ZMP, 2021-10-05
31.03.2022 13:41:06 : ZMP, 2021-10-06
31.03.2022 13:41:06 : ZMP, 2021-10-07
31.03.2022 13:41:06 : ZMP, 2021-10-08
31.03.2022 13:41:06 : ZMP, 2021-10-09
31.03.2022 13:41:07 : ZMP, 2021-10-10
31.03.2022 13:41:07 : ZMP, 2021-10-11
31.03.2022 13:41:07 : ZMP, 2021-10-12
31.03.2022 13:41:07 : ZMP, 2021-10-13
31.03.2022 13:41:07 : ZMP, 2021-10-14
31.03.2022 13:41:07 : ZMP, 2021-10-15
31.03.2022 13:41:08 : ZMP, 2021-10-16
31.03.2022 13:41:08 : ZMP, 2021-10-17
31.03.2022 13:41:08 : ZMP, 2021-10-18
31.03.2022 13:41:08 : ZMP, 2021-10-19
31.03.2022 13:41:08 : ZMP, 2021-10-20
31.03.2022 13:41:08 : ZMP, 2021-10-21
31.03.2022 13:41:09 : ZMP, 2021-10-22
31.03.2022 13:41:09 : ZMP, 2021-10-23
31.03.2022 13:41:09 : ZMP, 2021-10-24
31.03.2022 13:41:09 : ZMP, 2021-10-25
31.03.2022 13:41:09 : ZMP, 2021-10-26
31.03.2022 13:41:09 : ZMP, 2021-10-27
31.03.2022 13:41:10 : ZMP, 2021-10-28
31.03.2022 13:41:10 : ZMP, 2021-10-29
31.03.2022 13:41:10 : ZMP, 2021-10-30
31.03.2022 13:41:10 : ZMP, 2021-10-31