Data: Use the RMNP data from 5

https://goo.gl/1N1pce

https://autogis-site.readthedocs.io/en/latest/

http://geopandas.org/index.html

https://www.earthdatascience.org/courses/earth-analytics-python/

Import the necessary libraries:

In [1]:
```
# Import geopandas library, give it the nickname of gpd
import geopandas as gpd
```

1. Create dataframes from source data and plot them

For RMNP Trails:

2. Display the attribute table, columns, descriptive stats, and crs
3. Plot the trails layer with a figsize of 20, 10 and a green line.
4. Test out and plot a few additional tasks: df.centroid, df.envelope, df.convex_hull and understand what they're showing
5. Return the unique values from the UNIT field.

1. Create the necessary dataframes from the source data:

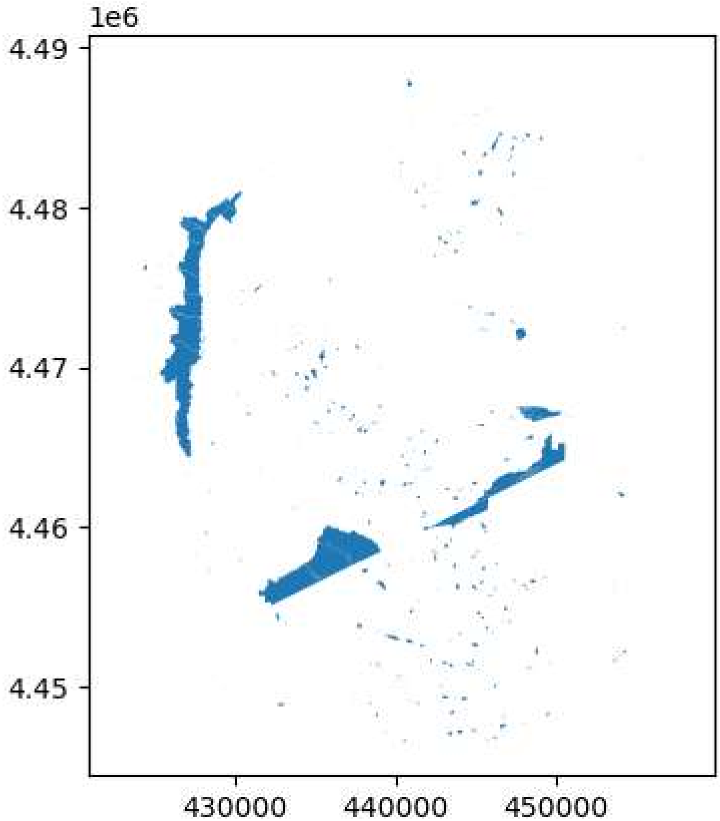
RMNP Lakes
RMNP Trails
ElkData - Elk843cow

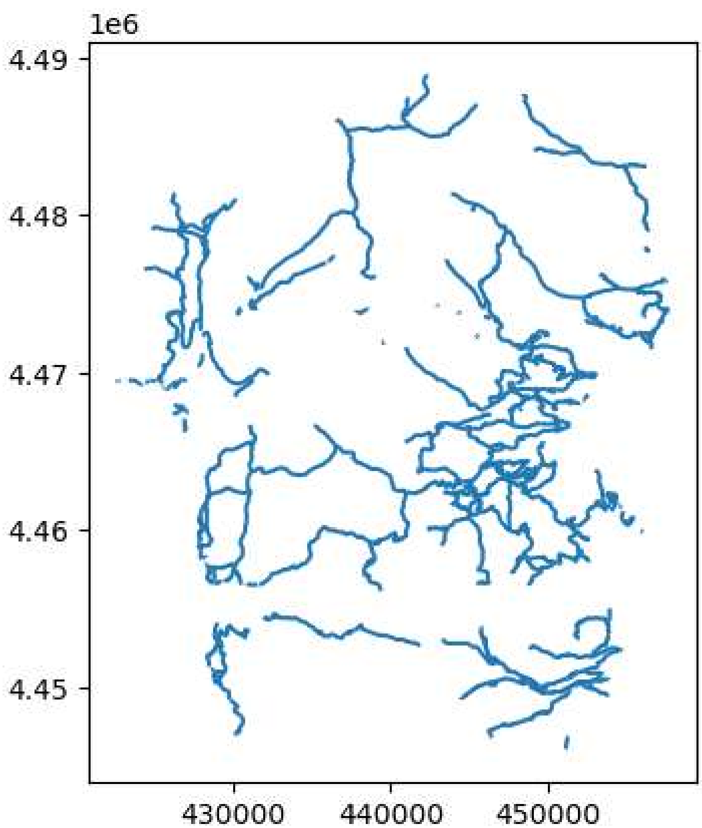
And plot these layers (individually, or better yet, together on one frame)

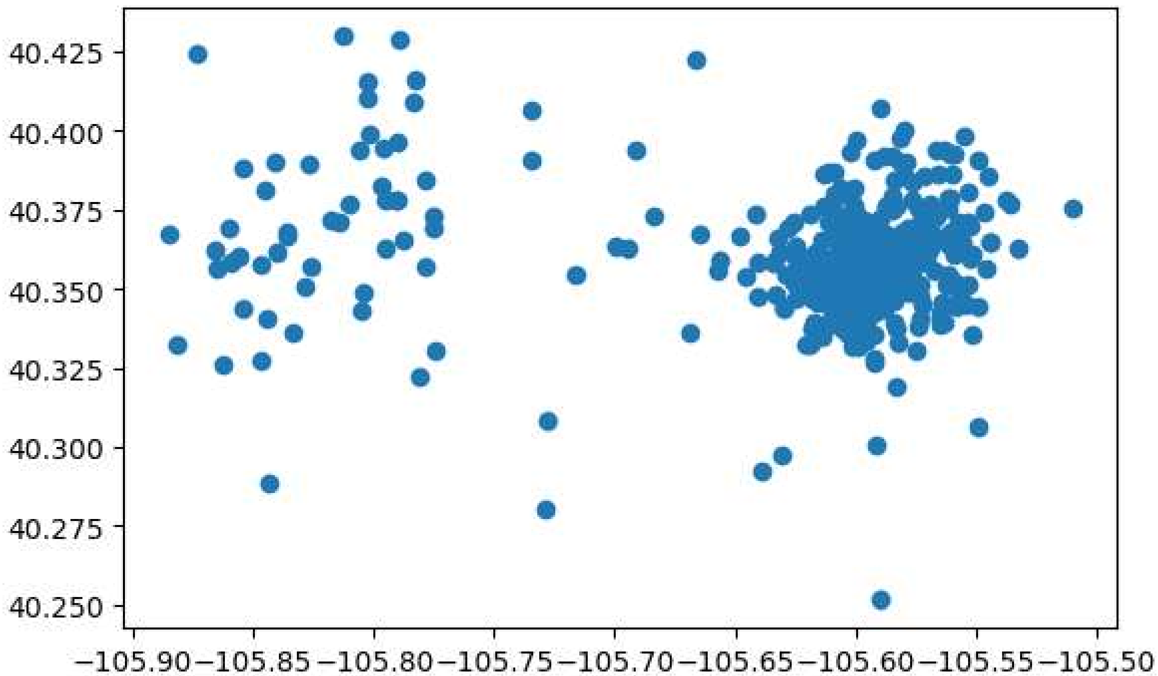
In [3]:
```
# Creating the dataframes
lakesPath = r"C:\Users\xfour\Downloads\RMNPDataLesson4\RMNPDataLesson4\RMNP_Lakes.shp"
trailsPath = r"C:\Users\xfour\Downloads\RMNPDataLesson4\RMNPDataLesson4\RMNP_Trails.sh
elkData = r"C:\Users\xfour\Downloads\RMNPDataLesson4\RMNPDataLesson4\Elk843cow.shp"

lakesDF = gpd.read_file(lakesPath)
trailsDF = gpd.read_file(trailsPath)
elkDF = gpd.read_file(elkData)
```

In [4]:
```
# Visualize the datasets with the needed library and method
lakesDF.plot()
trailsDF.plot()
elkDF.plot()
```

Out[4]: <Axes: >


1e6
4.49
4.48
4.47
4.46
4.45
430000
440000
450000


1e6
4.49
4.48
4.47
4.46
4.45
430000
440000
450000



2. Explore basic properties of your trails dataframe:

Return the attribute table, coordinate reference system, number of features, and column names
Return descriptive stats for the length column

In [9]:

```
# Show the attribute table

trailsDF.head(5)
```

Out[9]:

	SEGNAME	SEGMENT	UNIT	UNITNUM	STANDARD	USERCLASS	HORSE_USE	CDT	MILES
0	COMANCHE PEAK	PO-05	POUDRE	10	E	ALL	Y	None	2.2
1	MIRROR LAKE	PO-04	POUDRE	10	E	F	Y	None	0.8
2	MIRROR LAKE	PO-04	POUDRE	10	E	F	Y	None	0.5
3	MUMMY PASS, LOWER	PO-02	POUDRE	10	E	ALL	Y	None	2.6
4	POUDRE RIVER, LOWER	PO-01	POUDRE	10	E	ALL	Y	None	0.7

In [10]:
```
# How many trail feautures are there?
len(trailsDF)
```

Out[10]: 502

In [6]:
```
# Return the crs
trailsDF.crs
```

Out[6]:
```
<Projected CRS: EPSG:32613>
Name: WGS 84 / UTM zone 13N
Axis Info [cartesian]:
- E[east]: Easting (metre)
- N[north]: Northing (metre)
Area of Use:
- name: Between 108°W and 102°W, northern hemisphere between equator and 84°N, onshor
e and offshore. Canada - Northwest Territories (NWT); Nunavut; Saskatchewan. Mexico.
United States (USA).
- bounds: (-108.0, 0.0, -102.0, 84.0)
Coordinate Operation:
- name: UTM zone 13N
- method: Transverse Mercator
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

In [19]:
```
# Return the column names, as a nice list
column_names = trailsDF.columns.tolist()
```

```
print(column_names)
```

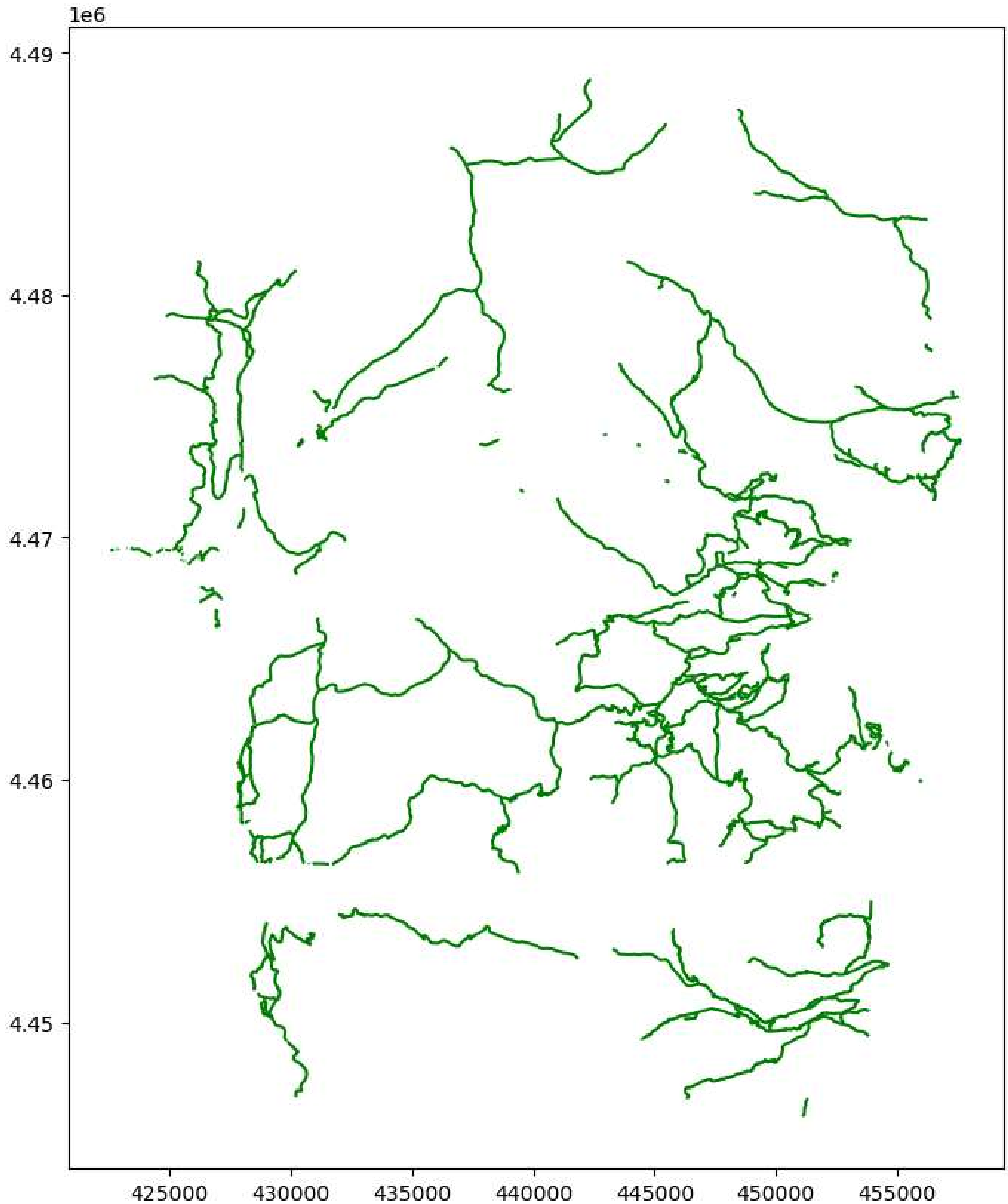
```
['SEGNAME', 'SEGMENT', 'UNIT', 'UNITNUM', 'STANDARD', 'USERCLASS', 'HORSE_USE', 'CD
T', 'MILES', 'FEET', 'SOURCE', 'OLDNAME', 'OLDNUM', 'FMSS', 'FMSSINFO', 'geometry']
```

In [20]:
```
# Get some descriptive statistics for a numeric column with df.fieldname.describe()

trailsDF.MILES.describe()
```

Out[20]:
```
count    502.000000
mean       0.700398
std        0.920188
min        0.000000
25%        0.100000
50%        0.400000
75%        0.900000
max        5.500000
Name: MILES, dtype: float64
```

3. Plot trails with a figure size of 20 by 10 and a green line.

In [46]:
```
import matplotlib.pyplot as plt



trailsplot = trailsDF.plot(color='green', figsize=(20, 10))
```

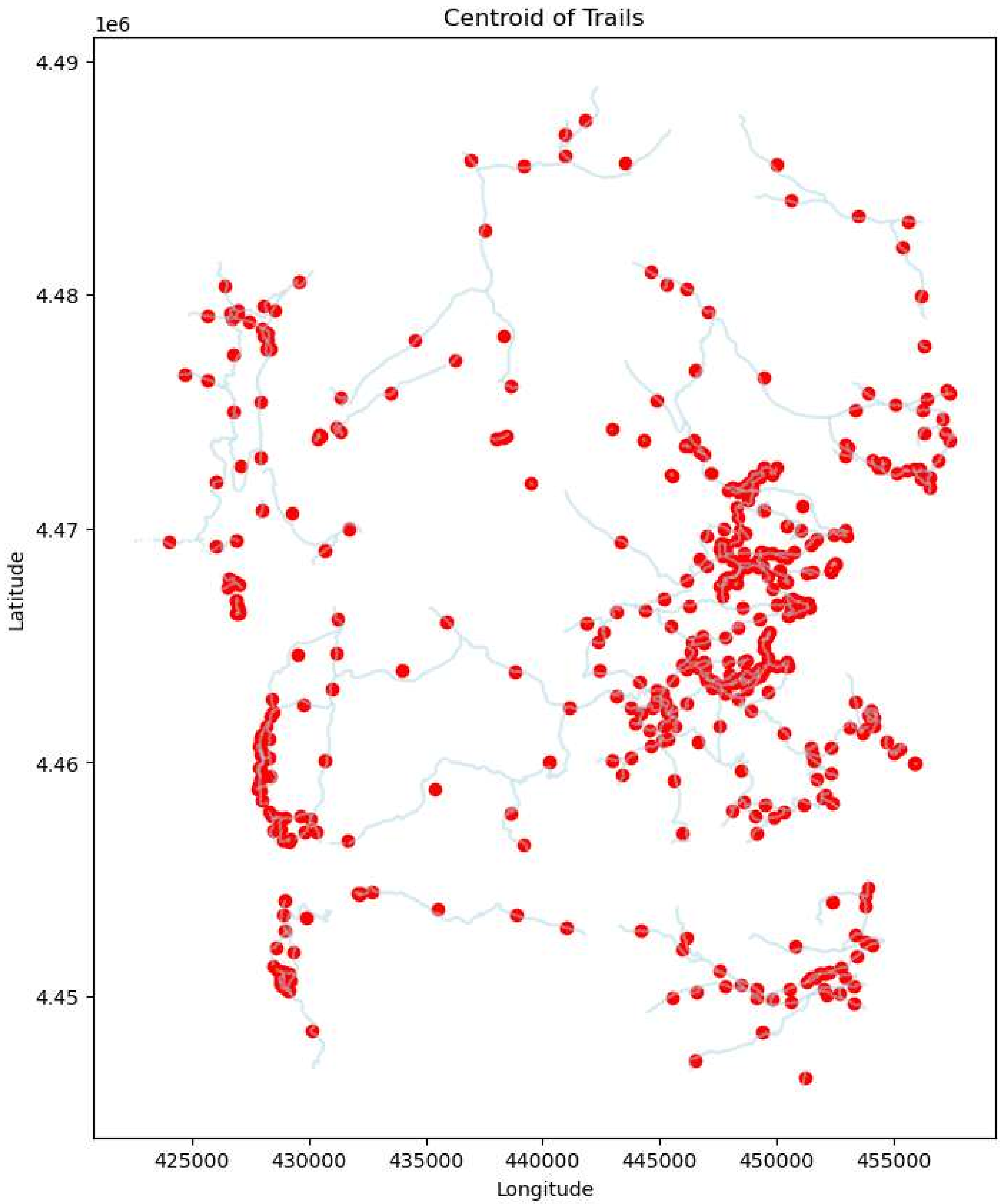


4. Test out and plot geometry operations: centroid, envelope, convex_hull

In [47]:

```
centroids = trailsDF.geometry.centroid

fig, ax = plt.subplots(figsize=(20, 10))
trailsDF.plot(ax=ax, color='lightblue', alpha=0.5)
centroids.plot(ax=ax, color='red', marker='o')

ax.set_title('Centroid of Trails')
ax.set_xlabel('Longitude')
```

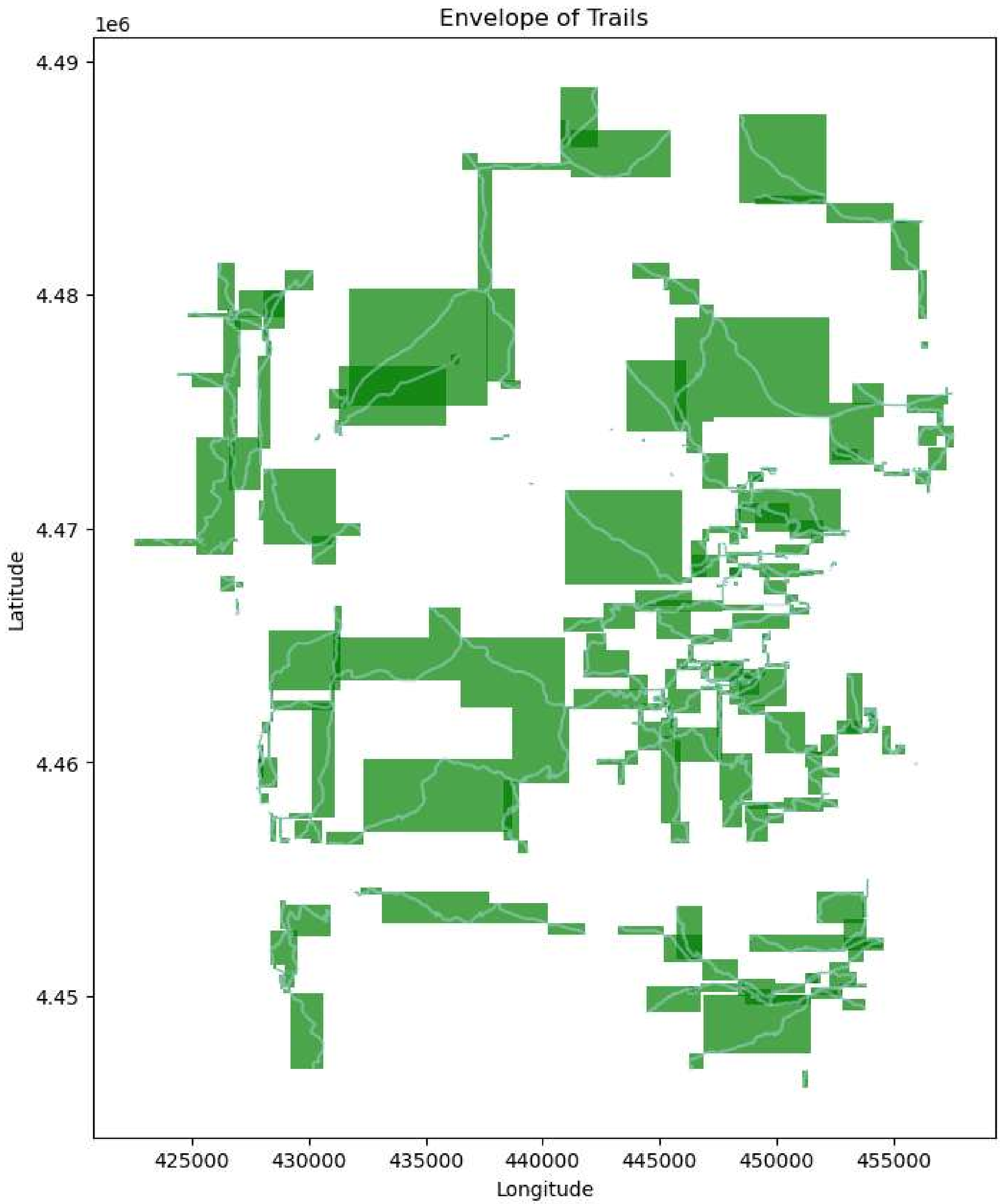
```
ax.set_ylabel('Latitude')
plt.show()
```



In [48]:

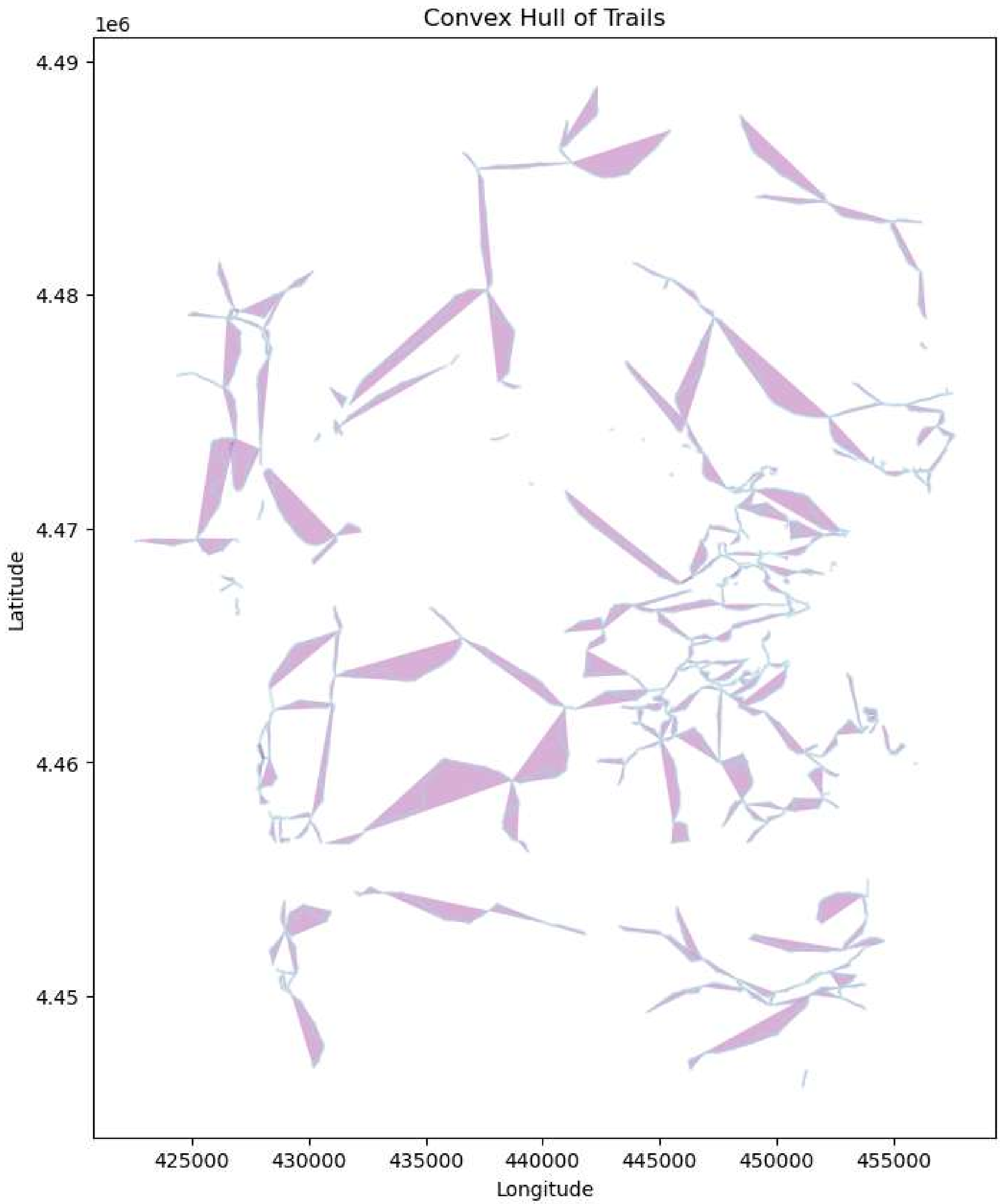
```
envelopes = trailsDF.geometry.envelope

fig, ax = plt.subplots(figsize=(20, 10))
trailsDF.plot(ax=ax, color='lightblue', alpha=0.5)
envelopes.plot(ax=ax, color='green', linestyle='--', alpha=0.7)

ax.set_title('Envelope of Trails')
ax.set_xlabel('Longitude')
```

```
ax.set_ylabel('Latitude')
plt.show()
```



In [49]:
```
convex_hulls = trailsDF.geometry.convex_hull

fig, ax = plt.subplots(figsize=(20, 10))
trailsDF.plot(ax=ax, color='lightblue', alpha=0.5)
convex_hulls.plot(ax=ax, color='purple', alpha=0.3)

ax.set_title('Convex Hull of Trails')
ax.set_xlabel('Longitude')
```

```
ax.set_ylabel('Latitude')
plt.show()
```



5. Return the unique values from the UNIT field, nicely formatted

In [51]:
```
unique_units = trailsDF['UNIT'].unique()


print("Unique 'UNIT' values:")
for unit in unique_units:
    print(unit)
```

```
Unique 'UNIT' values:
POUDRE
NORTH FORK
BLACK CANYON/ROARING RIVER
NEVER SUMMER
TRAIL RIDGE
FRONT RANGE
KAWUNEECHE VALLEY
WEST VALLEYS
LONGS PEAK
WILD BASIN
```

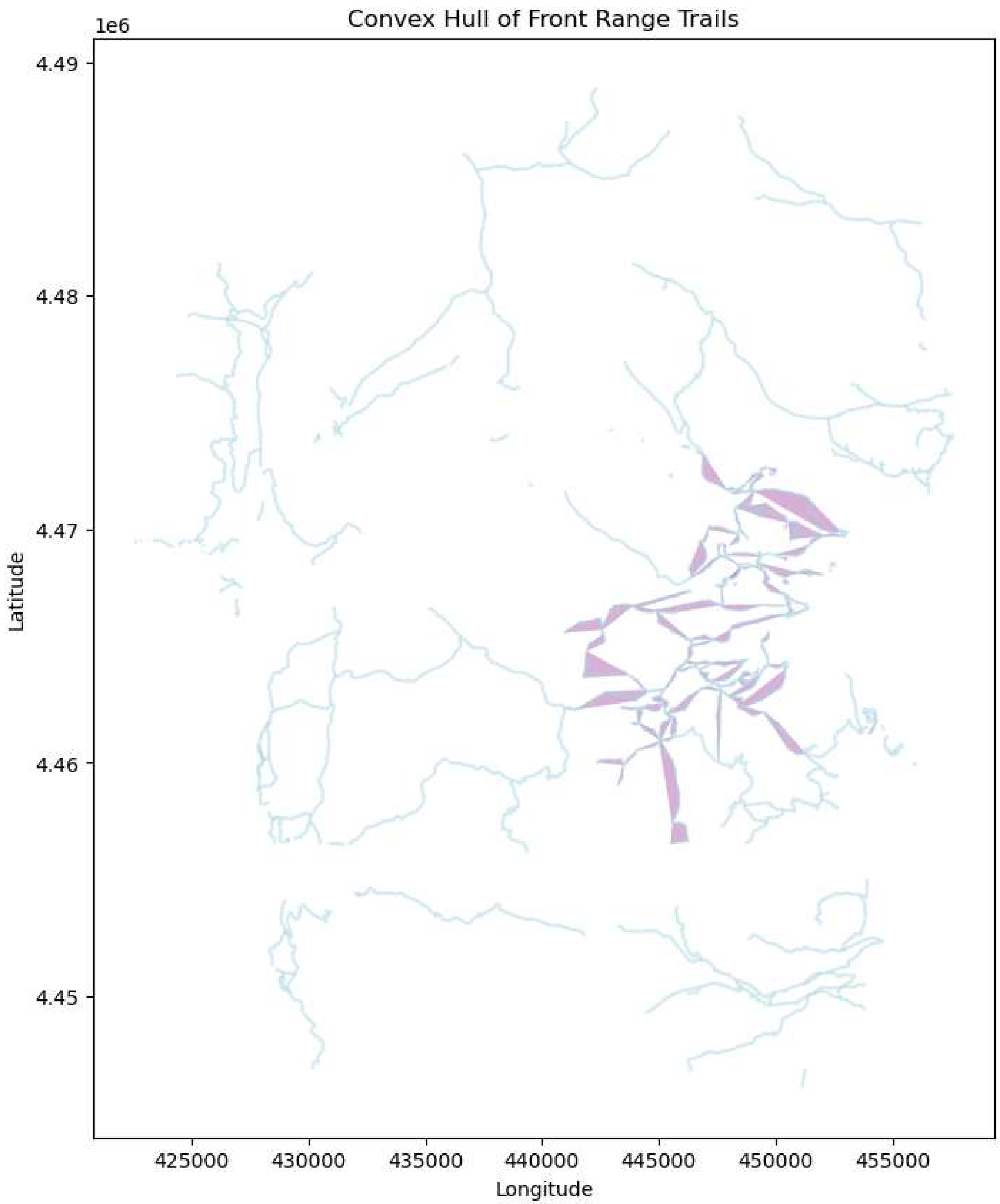
Which Elk 843 points are within 300 meters of a Front Range trail?

1. Get/Select the trails that belong to the FRONT RANGE unit (Use the UNIT field)
   a. Plot the centroid and convex hull of just the FRONT RANGE unit trails
2. Buffer those trails by 300m
3. If the CRS is the same for both trails and lakes, then: Overlay the lakes to the Front Range trail buffer. Try Intersect, Union, and Difference, and compare the outputs.
4. If the CRS is not the same, convert the CRS then perform the overlay

1. Get/Select the trails that belong to the FRONT RANGE unit (Use the UNIT field)

Plot the convex hull of just the FRONT RANGE unit trails

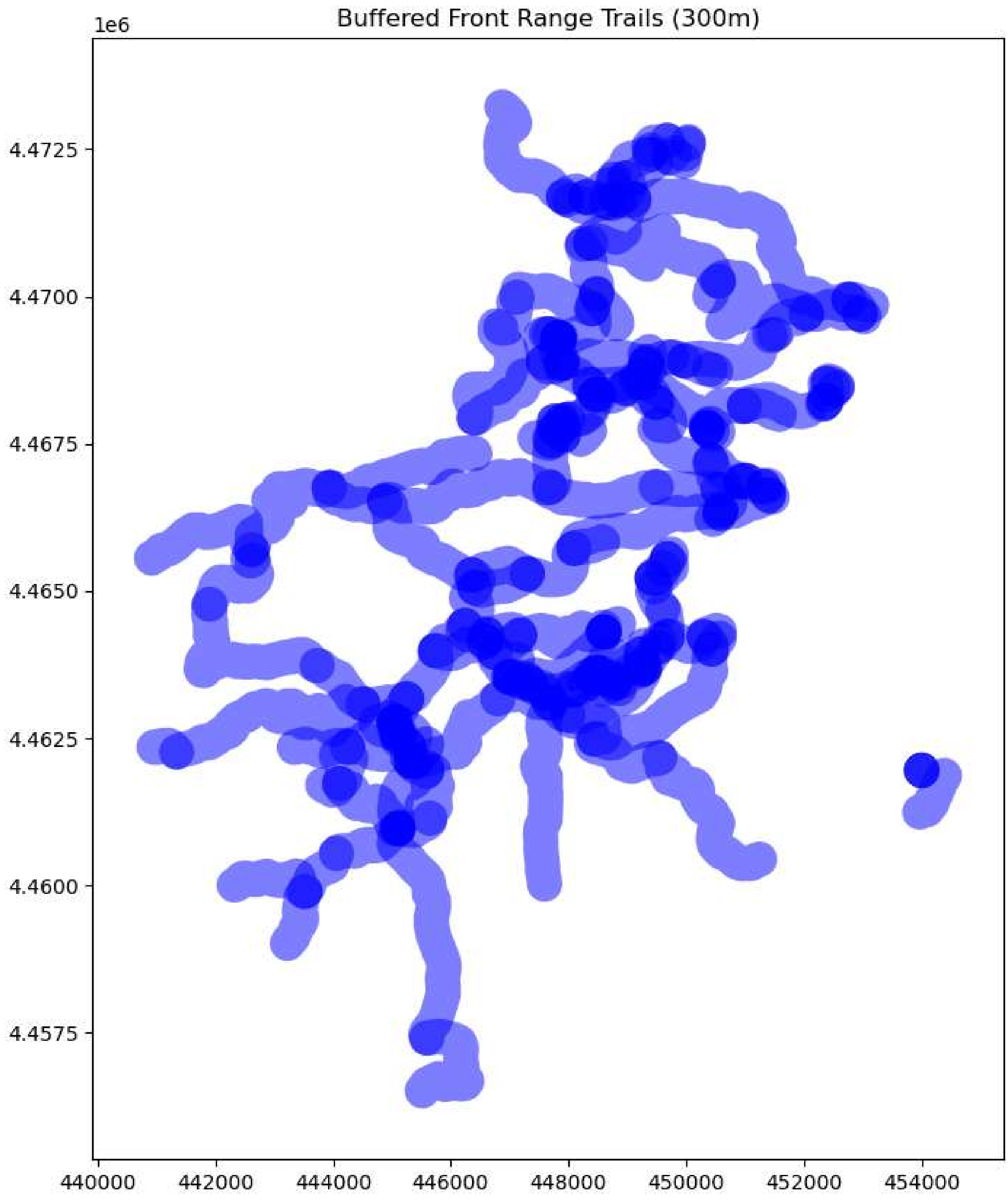
In [72]:
```
front_range_trails = trailsDF[trailsDF['UNIT'] == 'FRONT RANGE']

convex_hulls = front_range_trails.geometry.convex_hull

fig, ax = plt.subplots(figsize=(20, 10))
trailsDF.plot(ax=ax, color='lightblue', alpha=0.5)
convex_hulls.plot(ax=ax, color='purple', alpha=0.3)

ax.set_title('Convex Hull of Front Range Trails')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
plt.show()
```



2. Buffer the selected Front Range trails by 300m, plot the result

In [74]:
```
front_range_buffered = front_range_trails.buffer(300)

fig, ax = plt.subplots(figsize=(20, 10))


front_range_buffered.plot(ax=ax, color='blue', alpha=0.5)


ax.set_title('Buffered Front Range Trails (300m)')
```

```
plt.show()
```

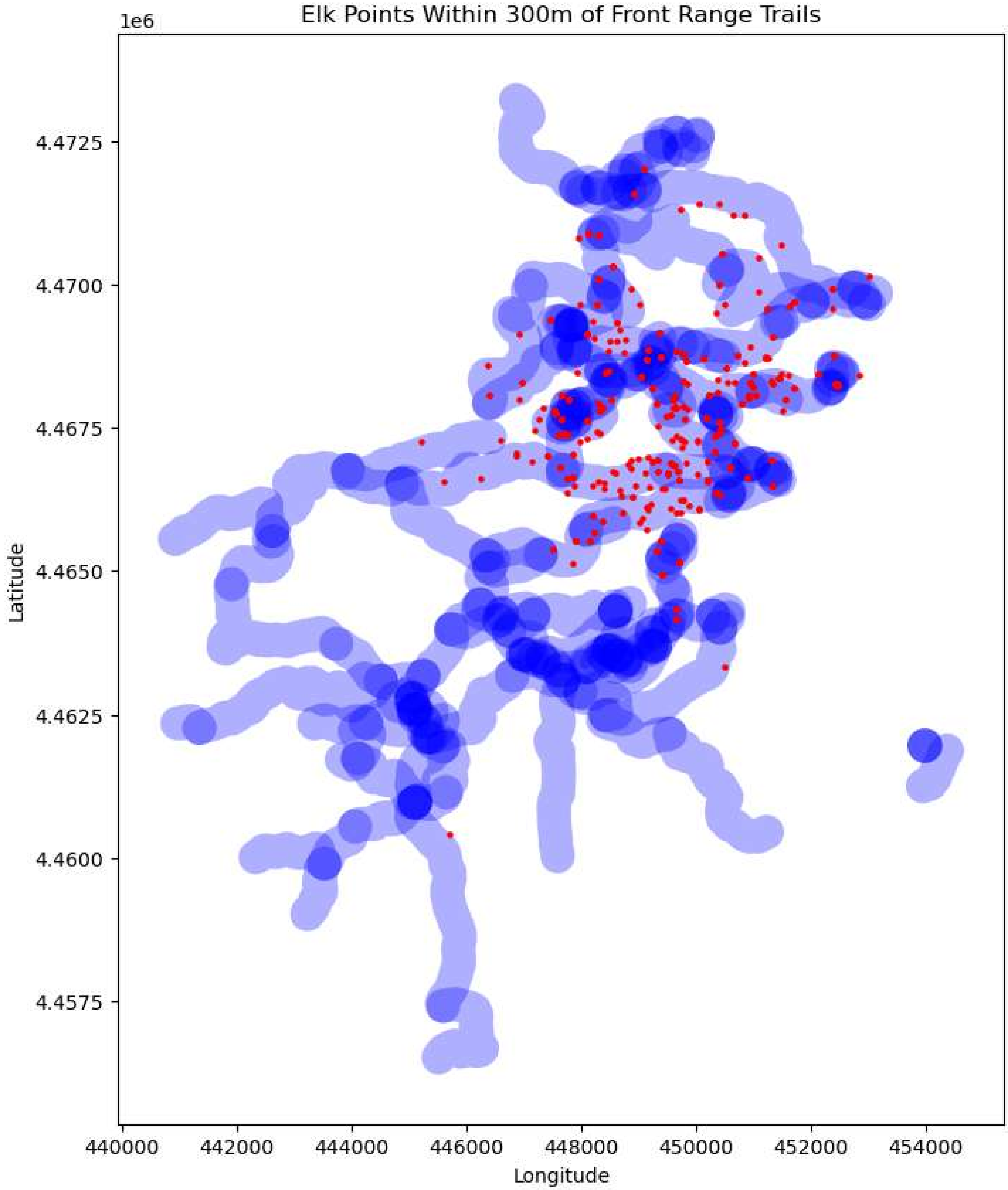


3 / 4. If the CRS's for Elk843cow and Trails match, perform the proper overlay operation to find all the Elk points that are within 300m of a Front Range trail.

If they don't match, convert the CRS, then do the overlay.

In [83]:
```
if elkDF.crs != trailsDF.crs:

    elkDF = elkDF.to_crs(trailsDF.crs)

front_range_buffered = front_range_trails.buffer(300)
```

```
buffered_trails_gdf = gpd.GeoDataFrame(geometry=gpd.GeoSeries(front_range_buffered), c


elk_near_trails = gpd.sjoin(elkDF, buffered_trails_gdf, how='inner', predicate='inters

fig, ax = plt.subplots(figsize=(20, 10))

buffered_trails_gdf.plot(ax=ax, color='blue', alpha=0.3, label='300m Buffer Zone')

elk_near_trails.plot(ax=ax, color='red', marker='o', markersize=5, label='Elk Points N

ax.set_title('Elk Points Within 300m of Front Range Trails')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')


plt.show()
```



Elk Points Within 300m of Front Range Trails
1e6
4.4725
4.4700
4.4675
4.4650
4.4625
4.4600
4.4575
Latitude
440000
442000
444000
446000
448000
450000
452000
454000
Longitude