Forcing the Retreat: A study of the Tasman Glacier from 2000 to 2025

Isla Fisher and Sara Loxton

GEOG208 Group Project

2025-10-13

Data gathering and Visualisation

Initialise the project and import everything needed for data gathering and visualisation.

```
import ee
import geemap

# Authenticate and initialize Google Earth Engine API
ee.Authenticate()

# Initialises project
ee.Initialize(project='ifi22-geog208')
```

Load all three Landsat images and scales them using the scaling factors from https://www.usgs.gov/faqs/how-do-i-use-a-scale-factor-landsat-level-2-science-products. These images were taken from the earth engine data catelog, Landsat Level 2 collection surface reflectance.

```
# Load the Landsat-7 image from 2000-02-09
image00 = ee.Image("LANDSAT/LE07/C02/T1_L2/LE07_075090_20000209")

# Load the Landsat-5 image from 2011-02-12
image11 = ee.Image("LANDSAT/LT05/C02/T1_L2/LT05_075090_20110215")

# Load the Lansat-8 image from 2025-01-04
image25 = ee.Image("LANDSAT/LC08/C02/T1_L2/LC08_075090_20250104")

# Applies scaling factors of optical bands for Landsat L2 (surface reflectance)
def apply_scale_factors(image):
  optical_bands = image.select('SR_B.').multiply(0.00275).add(-20)
  return image.addBands(optical_bands, None, True)

# Applies scaling function to the images
image00_scaled = apply_scale_factors(image00)
image11_scaled = apply_scale_factors(image11)
image25_scaled = apply_scale_factors(image25)

# Displays image data
image00_scaled
image11_scaled
image25_scaled
```

Creates a clipped rectangle of the Tasman galcier.

```
# Creates a polygon geometry for clipping

# Coordinates of Tasman glacier with some surrounding area
tasman_region = [170.1, -43.75, 170.35, -43.5]
tasman_glacier = ee.Geometry.Rectangle(tasman_region)

# Include only the optical bands in the clipped image
img00_clip = image00_scaled.select(['SR_B1','SR_B2','SR_B3','SR_B4','SR_B5','SR_B7']).clip(tasman_glacier)
img11_clip = image11_scaled.select(['SR_B1','SR_B2','SR_B3','SR_B4','SR_B5','SR_B7']).clip(tasman_glacier)
img25_clip = image25_scaled.select(['SR_B2','SR_B3','SR_B4','SR_B5','SR_B6','SR_B7']).clip(tasman_glacier)

# Display
display('Clipped image retains metadata and band names', img00_clip, img11_clip, img25_clip)
```

Histogram of Pixel Count against Pixel Value

Notes:

For sample00 the estimated max for each band were SR_B1 = 37, SR_B2 = 34, SR_B3 = 34, and SR_B4 = 25.

For sample11 the estimated max for each band were SR_B1 = 31, SR_B2 = 23, SR_B3 = 24, and SR_B4 = 35.

For sample25 the estimated max for each band were SR_B2 = 25, SR_B3 = 28, SR_B4 = 29, and SR_B5 = 35.

An average value was estimated for the different True colour and CIR band combinations. The histogram combined with some trial an error with the visualisation below determined the final min and max values.

Creates a histogram that can be used to see where most of the pixels lie in the dynamic range. The min and max values determined will help to aid in visualisation. Each band from each image was plugged in and the estimated max recorded.

```
# Historgram to look at the pixel distribution of the different Landsat images.
# The same code was used to look at all three satellite bands.
import geemap.chart as chart

# Specify the histogram labels and colours.
options = {
"title": "Histogram of pixel count vs pixel value",
"xlabel": "Pixel value",
"ylabel": "Pixel Count",
"colors": ["#1d6b99"],
}

# Use the clipped image as the sample. Takes 5000 random pixels.
sample00 = img00_clip.sample(tasman_glacier, 5000)
sample11 = img11_clip.sample(tasman_glacier, 5000)
sample25 = img25_clip.sample(tasman_glacier, 5000)

# Displays the histogram for the input sample and spectral band  is specified.
chart.feature_histogram(sample00, "SR_B1", minBucketWidth=0.01, **options)
```

Visualising the images

Notes:

We found that because the Tasman Glacier is covered in sediment, the NDSI did not represent it well. We have chosen to use CIR to visualise the change over time. CIR distinguishes the sediment from nearby vegetation and also highlights the growth of Tasman Lake, which is a result of the retreating Tasman Glacier, better than NDSI and the true colour visualisations.

Creates the mapped visualisation of Tasman Glacier with different spectral band combinations and NDSI layers. The min and max values are determined by the avergae of the values determined from the histogram above.

```
# Visualisaations for the different images. With true colour and
# colour infra red (CIR).

# Landsat-7 visuals with bands, min, and max values for image00_clip.
l7_visual_321 = {
    'bands': ['SR_B3', 'SR_B2', 'SR_B1'],
    'min': 0,
    'max': 35,
}

l7_visual_CIR = {
    'bands': ['SR_B4', 'SR_B3', 'SR_B2'],
    'min': 0,
    'max': 30,
}


# Landsat-5 visuals with bands, min, and max values for image11_clip.
l5_visual_321 = {
    'bands': ['SR_B3', 'SR_B2', 'SR_B1'],
    'min': 0,
    'max': 25,
}

l5_visual_CIR = {
    'bands': ['SR_B4', 'SR_B3', 'SR_B2'],
    'min': 0,
    'max': 30,
}


# Landsat-8 (l8) visuals. Need to use different bands than Lansat-7 and
```

```
# Landsat-5 satellites. For image25_clip.
l8_visual_432 = {
    'bands': ['SR_B4', 'SR_B3', 'SR_B2'],
    'min': 0,
    'max': 27,
}

l8_visual_CIR = {
    'bands': ['SR_B5', 'SR_B4', 'SR_B3'],
    'min': 0,
    'max': 30,
}


# Initialising the map of Tasman glacier
map_tasman = geemap.Map(ee_initialize=False)

# Setting the centre of the map to the centre of img00_clip
map_tasman.centerObject(img00_clip, 10)

# Add the visualisation layers to the Tasman glacier map
map_tasman.add_layer(img00_clip, l7_visual_321, '2000 True Color (321)')
map_tasman.add_layer(img00_clip, l7_visual_CIR, '2000 False Color (CIR)')
map_tasman.add_layer(img11_clip, l5_visual_321, '2011 True Color (321)')
map_tasman.add_layer(img11_clip, l5_visual_CIR, '2011 False Color (CIR')
map_tasman.add_layer(img25_clip, l8_visual_432, '2025 True Color (321)')
map_tasman.add_layer(img25_clip, l8_visual_CIR, '2025 False Color (CIR')


# NDSI set up

# NDSI visuals.
ndsi_parameters = {
    'min': -0.5,
    'max': 1,
    'palette': ['brown', 'lightblue', 'white'],
}

# Use Landsat-7 bands 2 (green) and 5 (SWIR1) for NDSI
ndsi_00 = img00_clip.normalizedDifference(['SR_B2', 'SR_B5'])  # Green - SWIR1

# Use Landsat-5 bands 2 (green) and 5 (SWIR1) for NDSI
ndsi_11 = img11_clip.normalizedDifference(['SR_B2', 'SR_B5'])  # Green - SWIR1

# Use Landsat-8 bands 3 (green) and 6 (SWIR1) for NDSI
ndsi_25 = img25_clip.normalizedDifference(['SR_B3', 'SR_B6'])  # Green - SWIR1


# Add the NDSI visualisation layer to the Tasman glacier map
map_tasman.add_layer(ndsi_00, ndsi_parameters, '2000 NDSI')
map_tasman.add_layer(ndsi_11, ndsi_parameters, '2011 NDSI')
map_tasman.add_layer(ndsi_25, ndsi_parameters, '2025 NDSI')


# Set up for the NDSI colour bar
colors = ndsi_parameters["palette"]
vmin = ndsi_parameters["min"]
vmax = ndsi_parameters["max"]


# Adds colour bar to the Tasman Glacier map
map_tasman.add_colorbar_branca(colors=colors, vmin=vmin, vmax=vmax, layer_name="2000 NDSI")


map_tasman
```

Save CIR images

Assings images with their visualisation parameters to objects. These will be used in the code below when the CIR images are saved as tiff files.

```
# Saves the three images with their spectral bands and visual parameters
# for use in following code.
cir_2000 = img00_clip.visualize(**l7_visual_CIR)
cir_2011 = img11_clip.visualize(**l5_visual_CIR)
cir_2025 = img25_clip.visualize(**l8_visual_CIR)
```

Save 2000, 2011, and 2025 CIR images as tif files. This uses code from the DEM tutorial included in lab 6.

```
# Exports all of the images

# Export cir_2000
geemap.ee_export_image_to_drive(
    image=cir_2000,
    description='cir_tasman_2000',
    fileNamePrefix='cir_tasman_2000',
    scale=10,
    region=tasman_glacier,
    fileFormat='GeoTIFF'
)

# Export cir_2011
geemap.ee_export_image_to_drive(
    image=cir_2011,
    description='cir_tasman_2011',
    fileNamePrefix='cir_tasman_2011',
    scale=10,
    region=tasman_glacier,
    fileFormat='GeoTIFF'
)

# Export cir_2025
geemap.ee_export_image_to_drive(
    image=cir_2025,
    description='cir_tasman_2025',
    fileNamePrefix='cir_tasman_2025',
    scale=10,
    region=tasman_glacier,
    fileFormat='GeoTIFF'
)
```

Image Classification

Attempt at Supervised classification

Notes:

Tasman Glacier is covered in debris which makes image classification more challenging. We attempted to use this training data https://developers.google.com/earth-engine/datasets/catalog/GLIMS_20230607#description to help counter this issue. We were unable to complete the supervised image classification because the three different classes snow, clean ice, and debris covered ice were not accessable. These thre classes were clumped together to form the class glacier (cyan) which is displayed above, and everything else was classes as non-glacier (brown). This was the only display that was sucsessful. We were unable to distinguish Tasman Glacier from the surrounding snow so the image classification was unsucessful. We were not capable of fixing this which meant we lacked data for area change and rate of change for Tasman Glacier and instead used data for all of New Zealands Glacier.

Supervised classification using the Global Land Ice Measurements (GILMs) training data.

```
# Loads the Global Land Ice Measurements (GILMs) training data
glims = ee.FeatureCollection("GLIMS/20230607")
glims_tasman = glims.filterBounds(tasman_glacier)


# Assing all the pixels to 1 class (because there wasnt a way to distinguish
# between them).
label_image = ee.Image(0).byte().paint(glims_tasman, 1).rename('class')


# Stratified sub sample
subsample = img00_clip.addBands(label_image).stratifiedSample(
    numPoints=1000,
    classBand='class',
    region=tasman_glacier,
    scale=30,
    geometries=True
)


# Split into training (80%) and validation (20%)
subsample = subsample.randomColumn()
training_sample = subsample.filter('random <= 0.8')
validation_sample = subsample.filter('random > 0.8')
```

```
# Train Random Forest classifier
trained_classifier = ee.Classifier.smileRandomForest(50).train(
    features=training_sample,
    classProperty='class',
    inputProperties=img00_clip.bandNames()
)


# Get information about the trained classifier.
display('Results of trained classifier', trained_classifier.explain())
```

Visualise the classification

```
# Classify the reflectance image using your trained GLIMS-based classifier
img00_classified = img00_clip.classify(trained_classifier)


## Visualization

# Binary legend for glacier vs non-glacier
legend_keys = [
    "Non-glacier (0)",
    "Glacier (1)"
]

legend_colors = [
    "#8B4513",  # brown for non-glacier
    "#00FFFF"   # cyan for glacier
]

# Visualization parameters for the classified image
classified_vis = {
    'min': 0,
    'max': 1,
    'palette': legend_colors,
}


# Initialising the map of Tasman glacier
m = geemap.Map(ee_initialize=False)

# Setting the centre of the map to the Tasman glacier
m.centerObject(img00_classified, 10)


# Add the classified layer to the map
m.add_layer(img00_classified, classified_vis, 'Glacier vs Non-glacier')

# Add legend
m.add_legend(keys=legend_keys, colors=legend_colors, position="bottomleft")

# Optional layers
# m.add_layer(glims_tasman, {}, 'GLIMS polygons')
# m.add_layer(training_sample, {}, 'Training samples')
# m.add_layer(validation_sample, {}, 'Validation samples')

# Map controls
m.add_layer_control(position='bottomright')
m
```

Attempt at Unsupervised classification using K-means

Notes:

This method of image classification was unsucessful because K means was unable to differentiate between Tasman Glacier and some surrounding mountainous regions. This caused extra pixels to be identified as glacier and displaced the area calculation. The table of areas produced shows an increase in ice volume between 2000 and 2011. This is not correct according to peer reviewed sources, indicating an issue with the classification.

The cluster that shows the glacier was found by plugging in every number from 0-15 into the code "glacier_cluster00 = result00.eq(10)" until the pixels of the glacier were highlighted cyan. This was repeated for each year and a note was taken of the glacier cluster.

Number of Glacier pixel cluster for each year:

2000 cluster number - 10

2011 cluster number - 11

2025 cluster number - 12

```
# Tight Tasman Glacier region
tight_tasman = ee.Geometry.Rectangle([170.13, -43.75, 170.31, -43.5])


# Creating tightly clipped images to get cut out surrounding glaciers
img00_tight = img00_clip.clip(tight_tasman)
img11_tight = img11_clip.clip(tight_tasman)
img25_tight = img25_clip.clip(tight_tasman)


# Select optical bands
bands_l5_l7 = ['SR_B1', 'SR_B2', 'SR_B3', 'SR_B4', 'SR_B5', 'SR_B7']
bands_l8 = ['SR_B2', 'SR_B3', 'SR_B4', 'SR_B5', 'SR_B6', 'SR_B7']


# Resample images
resampled_image00 = img00_tight.select(bands_l5_l7).resample('bilinear')
resampled_image11 = img11_tight.select(bands_l5_l7).resample('bilinear')
resampled_image25 = img25_tight.select(bands_l8).resample('bilinear')


# Unsupervised classification (K-means) for resampled_image00
subsample00 = resampled_image00.sample(region=tight_tasman, scale=30, numPixels=50000)
clusterer00 = ee.Clusterer.wekaKMeans(15).train(subsample00)
result00 = resampled_image00.cluster(clusterer00)

# Unsupervised classification (K-means) for resampled_image11
subsample11 = resampled_image11.sample(region=tight_tasman, scale=30, numPixels=50000)
clusterer11 = ee.Clusterer.wekaKMeans(15).train(subsample11)
result11 = resampled_image11.cluster(clusterer11)

# Unsupervised classification (K-means) for resampled_image25
subsample25 = resampled_image25.sample(region=tight_tasman, scale=30, numPixels=50000)
clusterer25 = ee.Clusterer.wekaKMeans(15).train(subsample25)
result25 = resampled_image25.cluster(clusterer25)


# Map results
m = geemap.Map()
m.centerObject(tight_tasman, 10)


# Random colours used for visualisation
m.add_layer(result00.randomVisualizer(), {}, 'Clusters 2000')
m.add_layer(result11.randomVisualizer(), {}, 'Clusters 2011')
m.add_layer(result25.randomVisualizer(), {}, 'Clusters 2025')


# Adds a layer of the pixels identified as glacier for 2000
glacier_cluster00 = result00.eq(10)
m.add_layer(glacier_cluster00.updateMask(glacier_cluster00), {'palette': ['cyan']}, 'Tasman Glacier 2000')

# Adds a layer of the pixels identified as glacier for 2011
glacier_cluster11 = result11.eq(11)
#m.add_layer(glacier_cluster11.updateMask(glacier_cluster11), {'palette': ['cyan']}, 'Tasman Glacier 2011')

# Adds a layer of the pixels identified as glacier for 2025
glacier_cluster25 = result25.eq(12)
#m.add_layer(glacier_cluster25.updateMask(glacier_cluster25), {'palette': ['cyan']}, 'Tasman Glacier 2025')


# Show the map
m
```

Show a list of the number of pixels per cluster and the total number of pixels for that image.

```
# Compute histogram of cluster IDs for 2000 within tight_tasman region
hist_dict00 = result00.reduceRegion(
    reducer=ee.Reducer.frequencyHistogram(),
    geometry=tight_tasman,
    scale=30,
    maxPixels=1e8
).get('cluster')

# Compute histogram of cluster IDs for 2011 within tight_tasman region
```

```
hist_dict11 = result11.reduceRegion(
    reducer=ee.Reducer.frequencyHistogram(),
    geometry=tight_tasman,
    scale=30,
    maxPixels=1e8
).get('cluster')

# Compute histogram of cluster for 2025 IDs within tight_tasman region
hist_dict25 = result25.reduceRegion(
    reducer=ee.Reducer.frequencyHistogram(),
    geometry=tight_tasman,
    scale=30,
    maxPixels=1e8
).get('cluster')


# Get it as a normal Python dictionary
hist00 = ee.Dictionary(hist_dict00).getInfo()
hist11 = ee.Dictionary(hist_dict11).getInfo()
hist25 = ee.Dictionary(hist_dict25).getInfo()


# Total pixels in the image
total_pixels00 = sum(hist00.values())
total_pixels11 = sum(hist11.values())
total_pixels25 = sum(hist25.values())


# Printing a nice formatted list for 2000
print("Cluster pixel counts 2000:\n")
for cluster_id, count in sorted(hist00.items(), key=lambda x: int(x[0])):
    print(f"Cluster {int(cluster_id):2d}: {count:>10,} pixels")

print(f"\nTotal pixels: {total_pixels00:,}\n")


#Printing a nice formatted list for 2011
print("Cluster pixel counts 2011:\n")
for cluster_id, count in sorted(hist11.items(), key=lambda x: int(x[0])):
    print(f"Cluster {int(cluster_id):2d}: {count:>10,} pixels")

print(f"\nTotal pixels: {total_pixels11:,}\n")


#Printing a nice formatted list for 2025
print("Cluster pixel counts 2025:\n")
for cluster_id, count in sorted(hist25.items(), key=lambda x: int(x[0])):
    print(f"Cluster {int(cluster_id):2d}: {count:>10,} pixels")

print(f"\nTotal pixels: {total_pixels25:,}")
```

Table of Tasman Glacier Area for every year

```
# Pixel counts for each year. This was taken from above
pixel_counts = {
    2000: 59528.333333333365, # Cluster 10
    2011: 84538.79215686275,  # Cluster 11
    2025: 70728.74509803922   # Cluster 12
}

# Landsat pixel size in meters and area
pixel_size = 30
pixel_area_m2 = pixel_size ** 2  # 900 m²

# Print table header
print("Year | Glacier Area (km²)")
print("------------------------")

# Compute area for each year
for year, pixels in pixel_counts.items():
    area_km2 = pixels * pixel_area_m2 / 1e6  # convert m² to km²
    print(f"{year} | {area_km2:.2f}")
```

Timelapse

Timelapse of the Tasman Glacier from 2000 to 2025. This will be completed by creating two separate timelapes and then merging them into one final timelapse which will be a mp4 file.

Notes:

Two timelapses are created and then the mp4 from each are merged to create one timelapse. This is because Landsat-8 an Landsat-9 data is unable to be used alongside Landsat-5 and Landsat-7 because of its different spectral band names.

These image ids were taken straight from google earth engine. We filtered through images from all Landsat satellites, going by year, and selecting the images that shows Tasman Glacier the clearest.

We are missing data from 2009 and 2012 because there were no usable images. In all images, Tasman Glacier was covered by cloud.

We avoided using Landsat-7 data past 2003 because of the Scan Line Corrector (SLC) failure.

Creates a collection of all the images for the first 2000-2011 timelapse and for the second 2012-2025 timelapse.

```
# This first collection invludes Landsat-5 and Landsat-7 data.
# Each have spectral bands SR_B4 = NIR, SR_B3 = Red and SR_B2 = Green.
image_ids_1 = [
    'LANDSAT/LE07/C02/T1_L2/LE07_075090_20000209',
    'LANDSAT/LE07/C02/T1_L2/LE07_075090_20010110',
    'LANDSAT/LE07/C02/T1_L2/LE07_075090_20020129',
    'LANDSAT/LE07/C02/T1_L2/LE07_075090_20030201',
    'LANDSAT/LT05/C02/T1_L2/LT05_075090_20040127',
    'LANDSAT/LT05/C02/T1_L2/LT05_075090_20050113',
    'LANDSAT/LT05/C02/T1_L2/LT05_075090_20060201',
    'LANDSAT/LT05/C02/T1_L2/LT05_075090_20070204',
    'LANDSAT/LT05/C02/T1_L2/LT05_075090_20081223',
    'LANDSAT/LT05/C02/T1_L2/LT05_075090_20100127', # Missing 2009
    'LANDSAT/LT05/C02/T1_L2/LT05_075090_20110215'
 ]


# This second collection invludes Landsat-8 and Landsat-9 data.
# Each have spectral bands SR_B5 = NIR, SR_B4 = Red and SR_B3 = Green.
image_ids_2 = [
    'LANDSAT/LC08/C02/T1_L2/LC08_075090_20130612', # Missing 2012
    'LANDSAT/LC08/C02/T1_L2/LC08_075090_20140207',
    'LANDSAT/LC08/C02/T1_L2/LC08_075090_20150226',
    'LANDSAT/LC08/C02/T1_L2/LC08_075090_20160128',
    'LANDSAT/LC08/C02/T1_L2/LC08_075090_20170215',
    'LANDSAT/LC08/C02/T1_L2/LC08_075090_20180914',
    'LANDSAT/LC08/C02/T1_L2/LC08_075090_20190512',
    'LANDSAT/LC08/C02/T1_L2/LC08_075090_20200208',
    'LANDSAT/LC08/C02/T1_L2/LC08_075090_20210330',
    'LANDSAT/LC09/C02/T1_L2/LC09_075090_20220104',
    'LANDSAT/LC08/C02/T1_L2/LC08_075090_20230115',
    'LANDSAT/LC08/C02/T1_L2/LC08_075090_20240102',
    'LANDSAT/LC08/C02/T1_L2/LC08_075090_20250104'
]


# Applies scaling factors of optical bands.
def apply_scale_factors(image):
  optical_bands = image.select('SR_B.').multiply(0.00275).add(-20)
  return image.addBands(optical_bands, None, True)


# Scales all of the images using the function defined in an ealrier section.
scaled_images_1 = [apply_scale_factors(ee.Image(img_id)) for img_id in image_ids_1]
scaled_images_2 = [apply_scale_factors(ee.Image(img_id)) for img_id in image_ids_2]


# Makes a collection of the scaled images.
collection_1 = ee.ImageCollection(scaled_images_1)
collection_2 = ee.ImageCollection(scaled_images_2)


collection_1
collection_2
```

Installs cartophy and imports os and cartoee.

```
!pip install cartopy

import os
from geemap import cartoee
import matplotlib.pyplot as plt
```

```
%pylab inline
```

Visualises the first image of each of the collections.

```
# Coordinates of Tasman glacier.
lon = 170.205420
lat = -43.622998


# Visual parameters
timelapse_params_1 = {
    "bands": ["SR_B4", "SR_B3", "SR_B2"],
    "min": 0,
    "max": 30
    }

timelapse_params_2 = {
    "bands": ["SR_B5", "SR_B4", "SR_B3"],
    "min": 0,
    "max": 30
    }


# Visualises the first image of collection_1 and collection_2 in CIR.

# First image from each collection.
image_1 = ee.Image(collection_1.first())
image_2 = ee.Image(collection_2.first())

# Initialising the map.
myMap = geemap.Map()

# Add the layers to the map using the visual parameters specified above.
myMap.addLayer(image_1, timelapse_params_1, "1: First image - 2000")
myMap.addLayer(image_2, timelapse_params_2, "2: First image - 2013")

# Set map centre on Tasman Glacier
myMap.setCenter(lon, lat, 12)

# Show the map
myMap
```

Defining the specifications for the north arrow and scale bar. For example, the font size, the colour, placement on the image etc.

```
tasman_region = [170.1, -43.75, 170.35, -43.5]


# North arrow parameters
north_arrow_dict = {
    "text": "N",
    "xy": (-0.05, 0.95),
    "arrow_length": 0.15,
    "text_color": "black",
    "arrow_color": "black",
    "fontsize": 20,
    "width": 5,
    "headwidth": 15,
    "ha": "center",
    "va": "center",
}


# add scale bar
scale_bar_dict = {
    "length": 10,
    "xy": (0.75, -0.04),
    "linewidth": 2,
    "fontsize": 15,
    "color": "black",
    "unit": "km",
    "ha": "center",
    "va": "bottom",
}


# Visualise the image with north arrow, scale bar, title ad gridlines.
```

```
# Figure size
fig = plt.figure(figsize=(10, 8))

# Use cartoee to get a map
ax = cartoee.get_map(image_1, region=tasman_region, vis_params=timelapse_params_1)
# Uncomment the code below to get a visualisation of the first image from collection 2.
#ax = cartoee.get_map(image_2, region=tasman_region, vis_params=timelapse_params_2)


# Adds north arrow, scale bar and grid lines.
cartoee.add_north_arrow(ax, **north_arrow_dict)
cartoee.add_scale_bar_lite(ax, **scale_bar_dict)
cartoee.add_gridlines(ax,
                      interval=[0.1, 0.1], # Specified interval
                      linestyle=":"
                      )

# Sets the title for the image.
ax.set_title(label="Tasman Glacier NZ: 2000-2025 \n  Isla Fisher & Sara Loxton \n NIR-Red-Green (CIR)\n Date:",
             fontsize=15)


show()
```

Mounting Google Drive to recieve the two timelapses.

```
from google.colab import drive

# Mount Google Drive.
drive.mount('/content/drive', force_remount=True)

%cd /content/drive/MyDrive

!pwd

!ls
```

Saving timelapse 1

```
# Saving timelapse 1 to Google Drive with visual parameters defined above.
cartoee.get_image_collection_gif(
    ee_ic=collection_1,
    out_dir='tasman_timelapse_2000-2011_folder', # Name of the foler on Google Drive.
    out_gif="tasman_2000-2011.gif",              # name of the gif.
    vis_params=timelapse_params_1,
    region=tasman_region,               # Region is the Tasman region defined earlier.
    fps=1,
    mp4=True,
    grid_interval=(0.1, 0.1),
    plot_title="Tasman Glacier NZ: 2000-2025 \n  Isla Fisher & Sara Loxton \n NIR-Red-Green (CIR)\n Date:",
    date_format="YYYY-MM-dd",
    fig_size=(10, 8),
    dpi_plot=300,
    file_format="png",
    north_arrow_dict=north_arrow_dict, # Using the specifications from above for north arrow.
    scale_bar_dict=scale_bar_dict,     # Using the specifications from above for the scale bar.
    verbose=True,
)
```

Saving timelapse 2

```
# Saving timelapse 2 to Google Drive with visual parameters defined above.
cartoee.get_image_collection_gif(
    ee_ic=collection_2,
    out_dir='tasman_timelapse_2013-2025_folder', # Name of the foler on Google Drive.
    out_gif="tasman_2013-2025.gif",              # name of the gif.
    vis_params=timelapse_params_2,
    region=tasman_region,               # Region is the Tasman region defined earlier.
    fps=1,
    mp4=True,
    grid_interval=(0.1, 0.1),
    plot_title="Tasman Glacier NZ: 2000-2025 \n  Isla Fisher & Sara Loxton \n NIR-Red-Green (CIR)\n Date:",
    date_format="YYYY-MM-dd",
    fig_size=(10, 8),
    dpi_plot=300,
    file_format="png",
```

```
    north_arrow_dict=north_arrow_dict, # Using the specifications from above for north arrow.
    scale_bar_dict=scale_bar_dict,     # Using the specifications from above for the scale bar.
    verbose=True,
)
```

Merges the two mp4 files togther to form one completed timelapse from 2000-2025.

```
from moviepy.editor import VideoFileClip, concatenate_videoclips

# Load your Landsat timelapses as MP4
clip1 = VideoFileClip("/content/drive/MyDrive/tasman_timelapse_2000-2011_folder/tasman_2000-2011.mp4")
clip2 = VideoFileClip("/content/drive/MyDrive/tasman_timelapse_2013-2025_folder/tasman_2013-2025.mp4")

# Concatenate sequentially
final_clip = concatenate_videoclips([clip1, clip2])

# Save final video
final_clip.write_videofile(
    "/content/drive/MyDrive/tasman_2000-2025_final.mp4",
    codec="libx264",
    fps=10
)
```

Digital Elevation Model (DEM) Setup

Code from "3D visualisation of a terrain using ArcGIS Pro, Author: Dr Gorden Jiang" was adapted for the creation of the DEM for this project.

Mapping the area to be included in the DEM

```
# Define the coordinates for Tasman Glacier, New Zealand.
# Bounding box: [west, south, east, north]
tasman_region = [170.1, -43.75, 170.35, -43.5]
tasman_glacier = ee.Geometry.Rectangle(tasman_region)

# Calculate the center for map visualization
center_lat = (tasman_region[1] + tasman_region[3]) / 2
center_lon = (tasman_region[0] + tasman_region[2]) / 2

# Optional: Visualize the AOI (requires geemap or folium)
# If using geemap:
m = geemap.Map(center=[center_lat, center_lon], zoom=10)
m.addLayer(tasman_glacier, {'color': 'red'}, 'Tasman Glacier AOI')
m
```

Loads to DEM model to the map above.

```
# Load the SRTM Digital Elevation Model
dem = ee.Image('USGS/SRTMGL1_003').clip(tasman_glacier)

# Optional: Visualize the DEM
# If using geemap (assuming 'm' from previous cell is available):
dem_vis_params = {'min': 0, 'max': 4000, 'palette': ['006633', 'E5FFCC', '662A00', 'FF8000', 'EA0000']}
m.addLayer(dem, dem_vis_params, 'DEM')
# m (run again to display new layer if not already displaying)
```

Loads a true colour iamge from sentinel 2 to the map above.

```
# Load Sentinel-2 Surface Reflectance data
# Filter by date and cloud cover, then take a median composite
# Using COPERNICUS/S2_SR_HARMONIZED for consistency
sentinel2 = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED') \
    .filterDate('2025-01-01', '2025-8-31') \
    .filterBounds(tasman_glacier) \
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 1)) \
    .median() \
    .clip(tasman_glacier)

# Define visualization parameters for Sentinel-2 (Natural Color - B4, B3, B2)
# Sentinel-2 bands are scaled by 10000 for surface reflectance
# So, divide by 10000 to get reflectance values between 0 and 1
# B4 (Red), B3 (Green), B2 (Blue)
rgb_vis_params = {
```

```
    'min': 0,
    'max': 2500, # Max value for visualization (0-10000 original scale for SR)
    'bands': ['B4', 'B3', 'B2']
}

# Optional: Visualize Sentinel-2
# If using geemap (assuming 'm' from previous cell is available):
m.addLayer(sentinel2, rgb_vis_params, 'Sentinel-2 RGB')
# m (run again to display new layer if not already displaying)
```

Defines CIR parameters and adds layer to map above.

```
# Visualise CIR
# Sentinel-2 bands: B8 (NIR), B4 (Red), B3 (Green)
cir = sentinel2.select(['B8', 'B4', 'B3']).clip(tasman_glacier)


# Define visualization parameters for CIR
cir_vis_params = {
    'bands': ['B8', 'B4', 'B3'],
    'min': 0,
    'max': 3000,
}

# Optional: Visualize NDVI
# If using geemap (assuming 'm' from previous cell is available):
m.addLayer(cir, cir_vis_params, '2025 False Colour (CIR)')
# m (run again to display new layer if not already displaying)
```

Export the dem and cir image to google drive.

```
# Export the DEM to the root of Google Drive
print("Exporting DEM to your Google Drive root directory... This may take a few moments.")
geemap.ee_export_image_to_drive(
    dem,
    description='tasman_glacier_dem',
    fileNamePrefix='tasman_glacier_dem',
    scale=30,
    region=tasman_glacier
)

# Export the NDVI to the root of Google Drive
print("\nExporting NDVI to your Google Drive root directory... This may take longer.")
geemap.ee_export_image_to_drive(
    cir,
    description='tasman_glacier_cir',
    fileNamePrefix='tasman_glacier_cir',
    scale=10,
    region=tasman_glacier
)
```