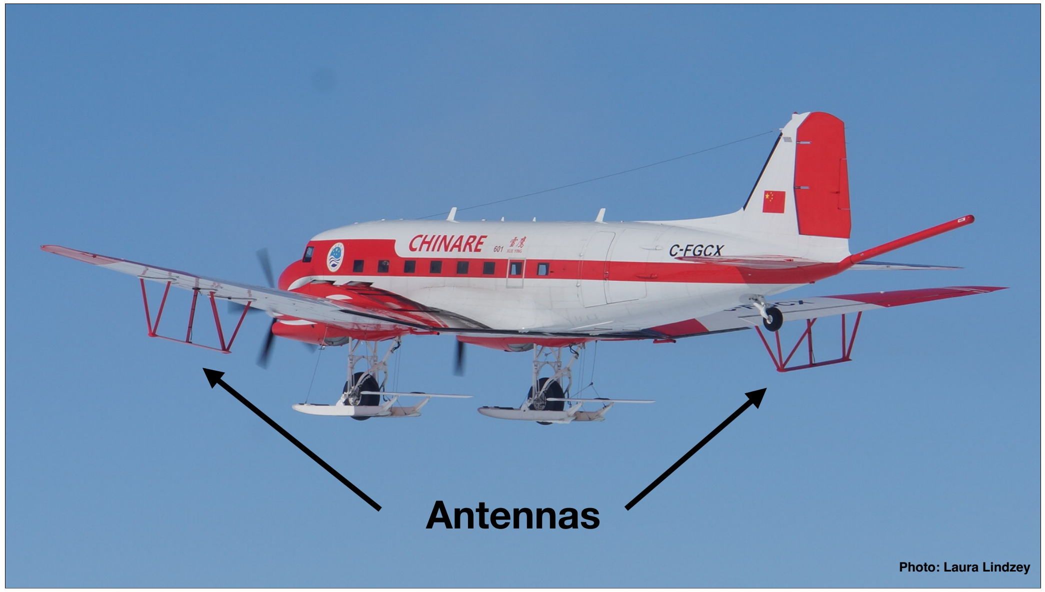
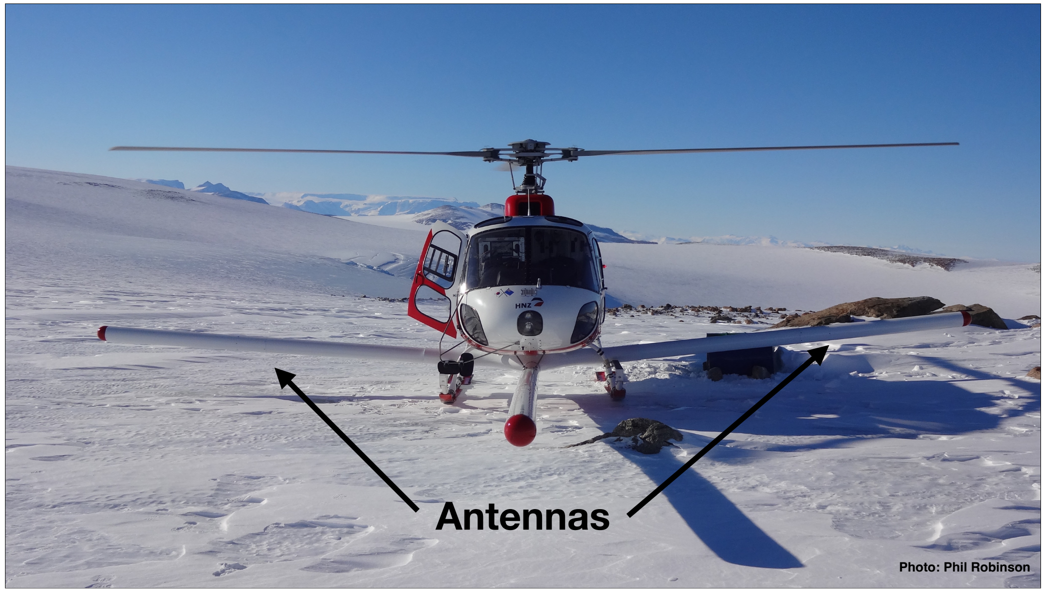
Hi, I’m Laura Lindzey. I’m a research engineer at the University of Washington Applied Physics Laboratory. Most of my job is writing software for underwater robots, but that’s not what I’m talking about today … I’m presenting QIceRadar, which is a tool I’m building to make it easier for scientists to explore ice penetrating radar data.

This talk will be a case study of a scientific community that needs better tools, and one tool I’m working on to help. Before I talk more about building a QGIS plugin and wrangling datasets, I want to talk about the data that it is making available, and how it is used by scientists.



Ice is transparent to electromagnetic energy in the RF range, which means that it's possible to "see" through kilometers of ice. The most efficient way to collect this data is to mount antennas and electronics on an aircraft and fly above the ice sheet. The research group I was part of used aircraft like this updated DC-3 airplane …



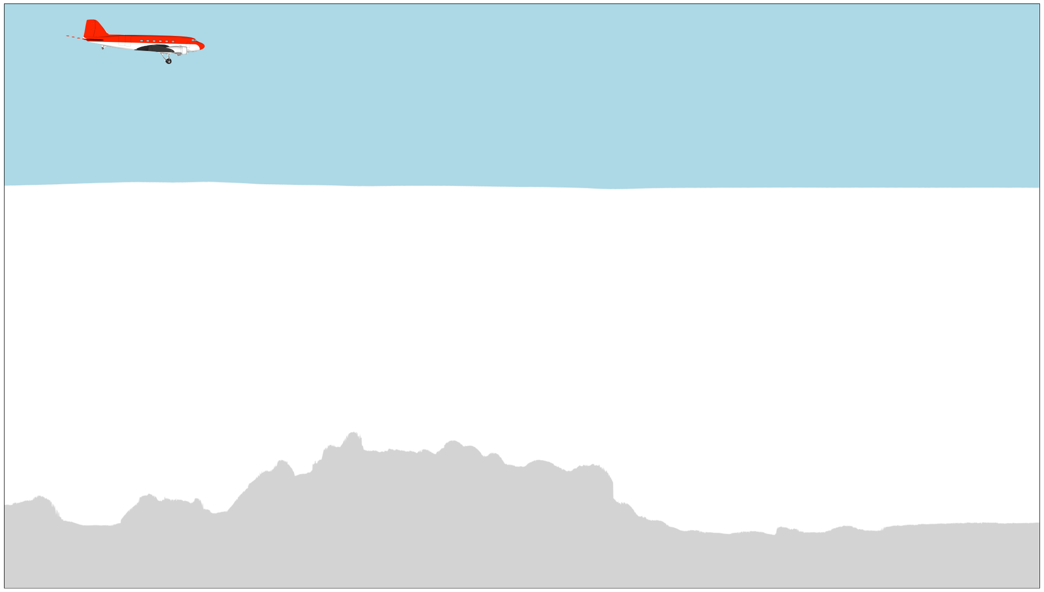
And helicopters, in large area surveys.

For small-area surveys, it's also common to pull a ground-penetrating radar with a skidoo, as in this photo of a survey near South Pole

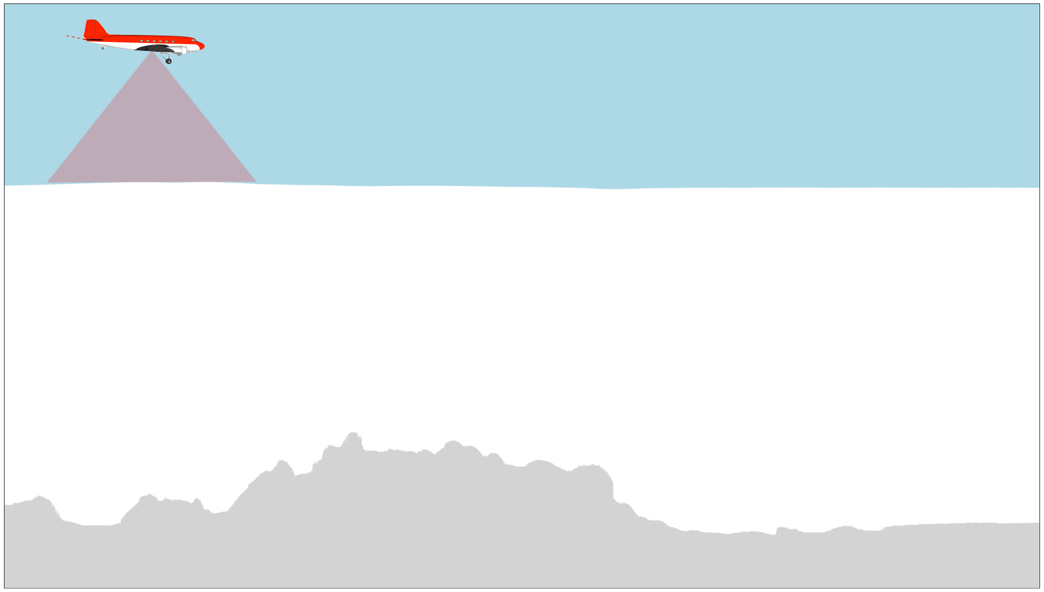
… or even on foot as in this photo of a survey on the Nansen Ice Shelf.

————-

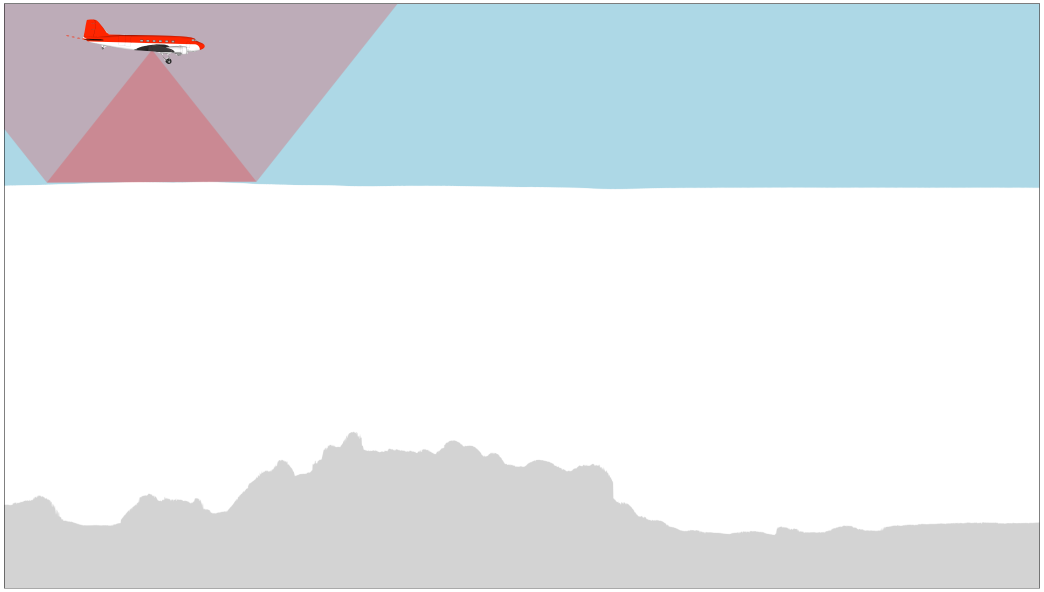
2016 Nansen, provided by Christine Dow



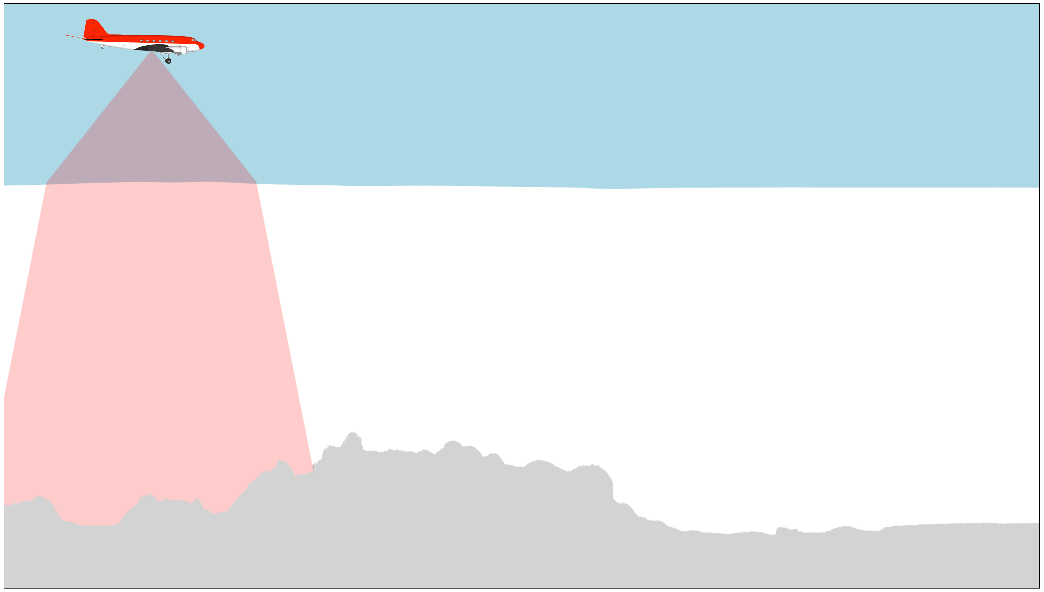
So, how does ice penetrating radar work?
As we fly above the ice sheet …



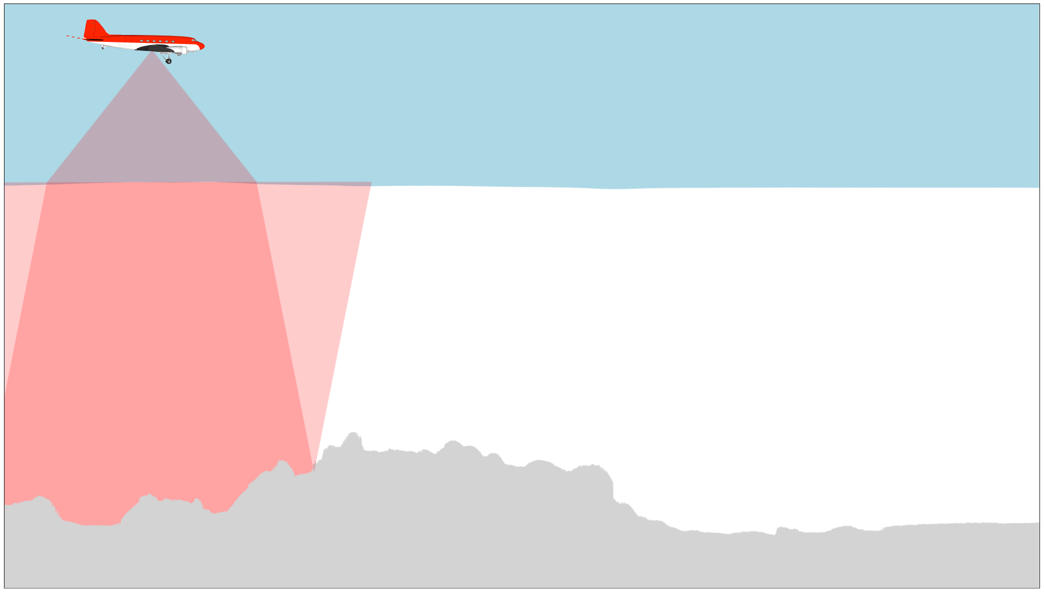
The radar transmits pulses of energy



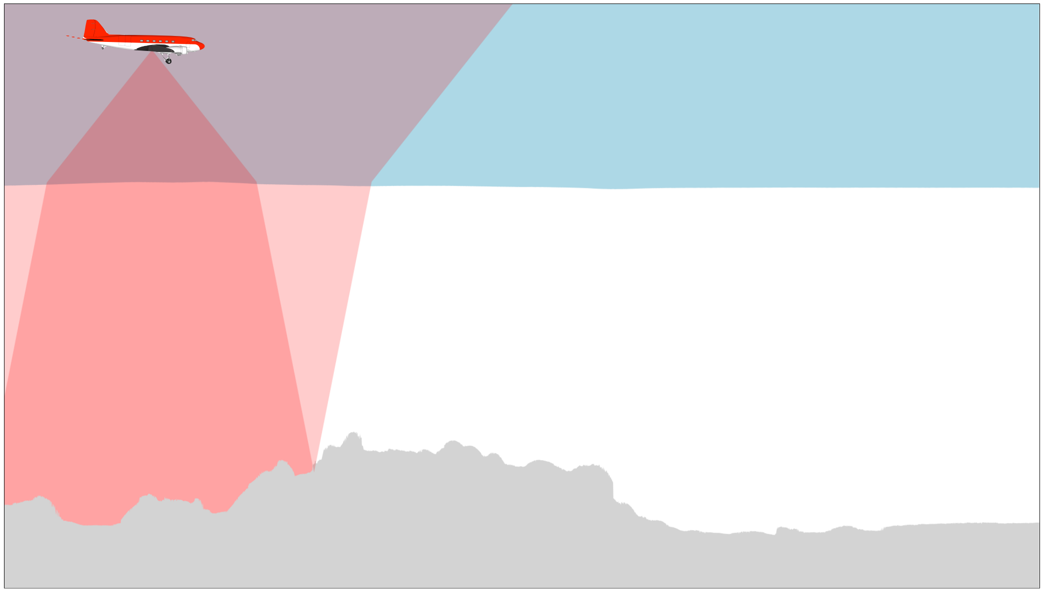
some of it bounces off the ice surface



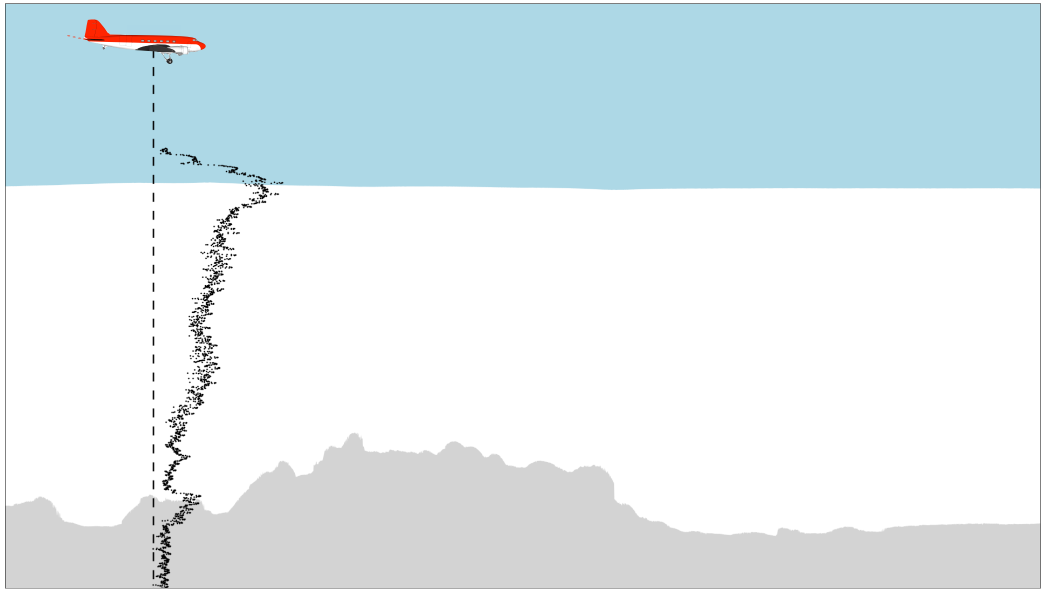
and some is transmitted through the ice.



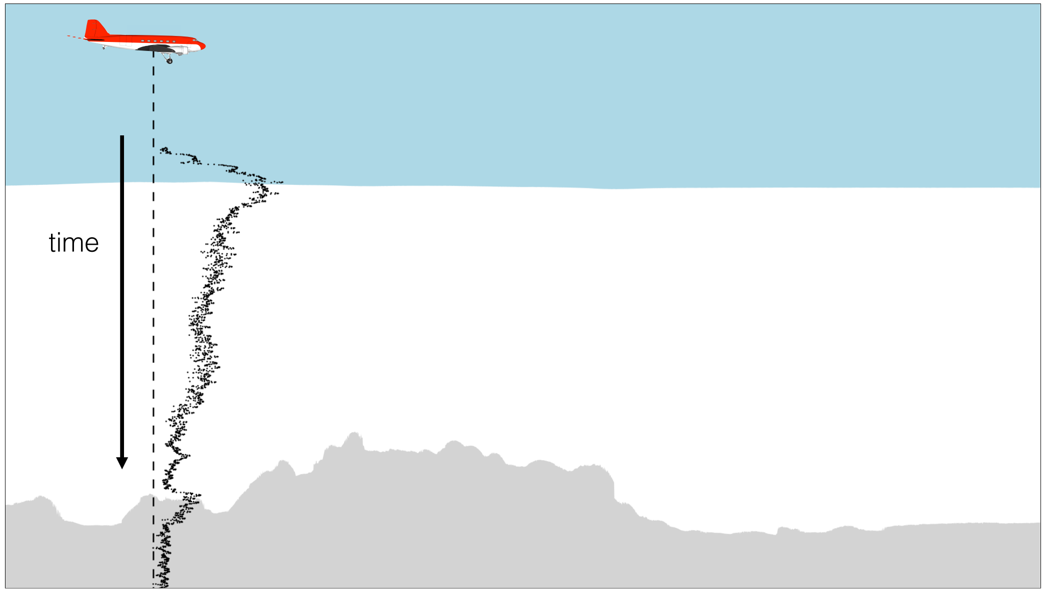
some of the transmitted energy is reflected from the bed….



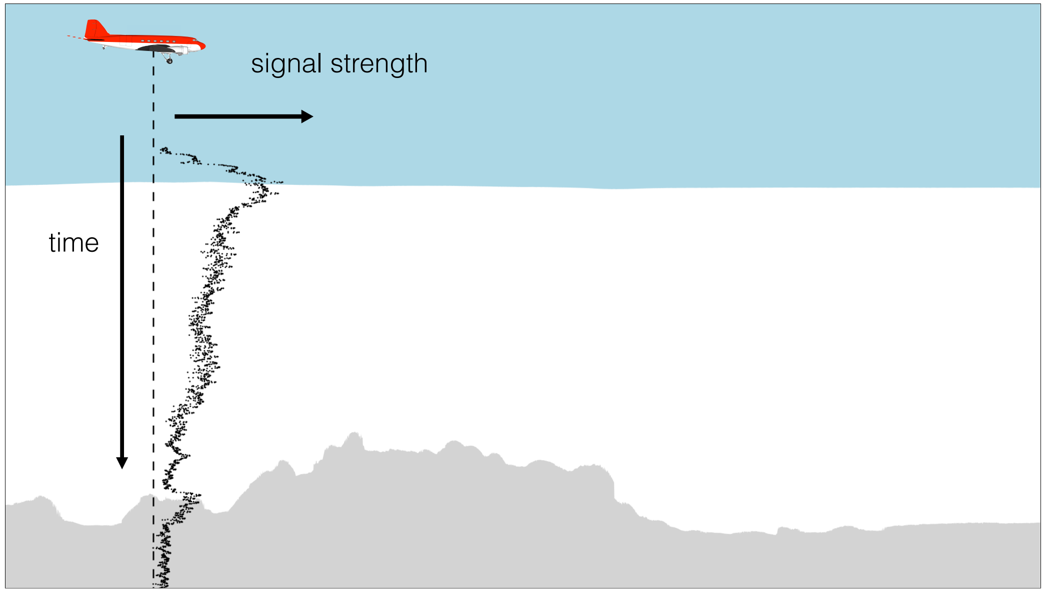
And refracted through the surface.
If you record the time-varying intensity of the returned energy from a single pulse, as measured at the antennas



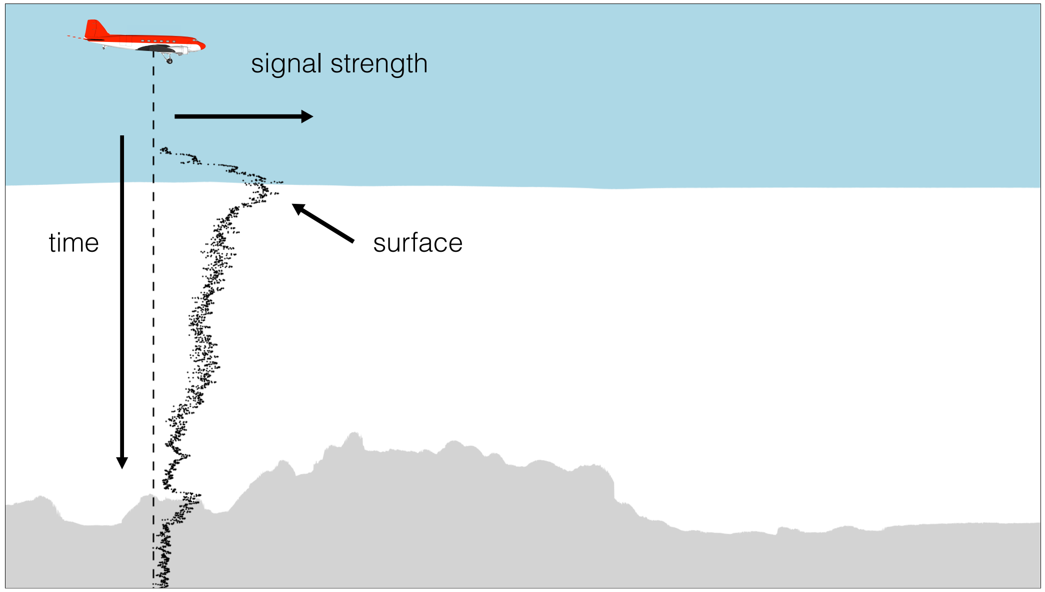
You get a trace like this.



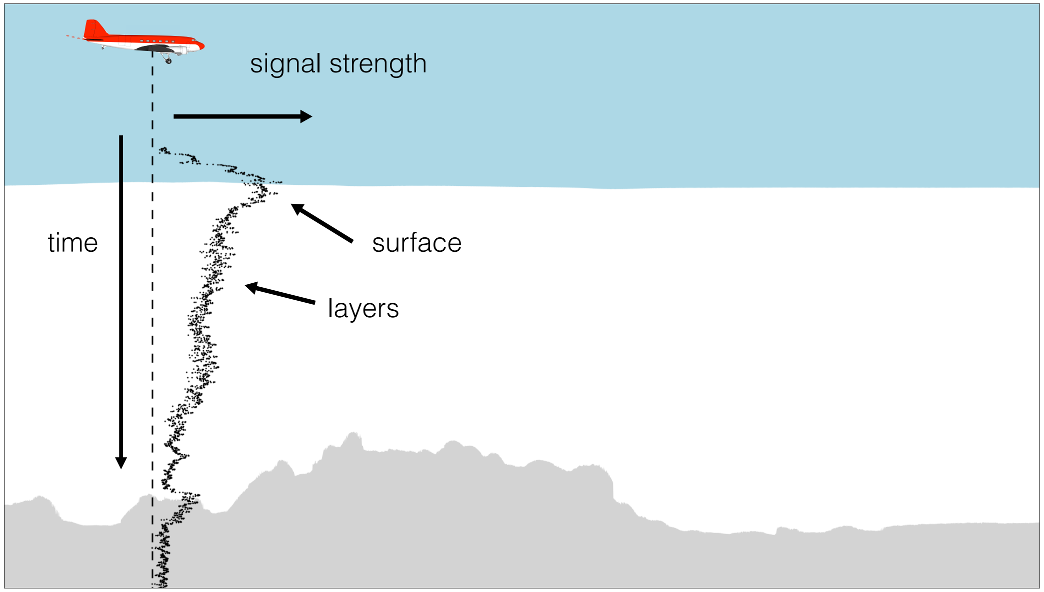
* The y axis is time since transmission, which corresponds to distance beneath the airplane



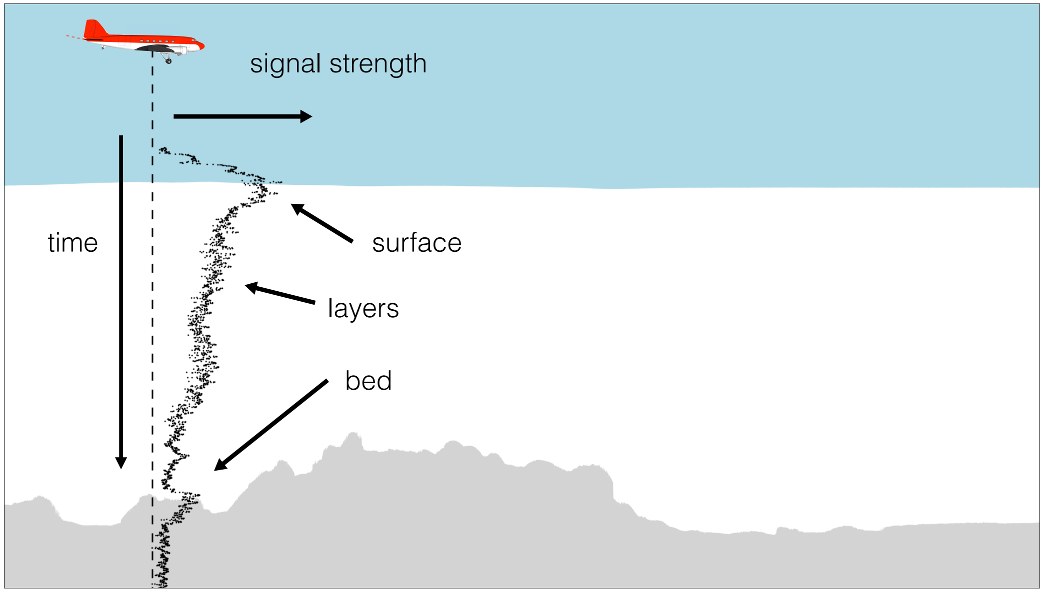
* the x axis is strength of field measured at that time



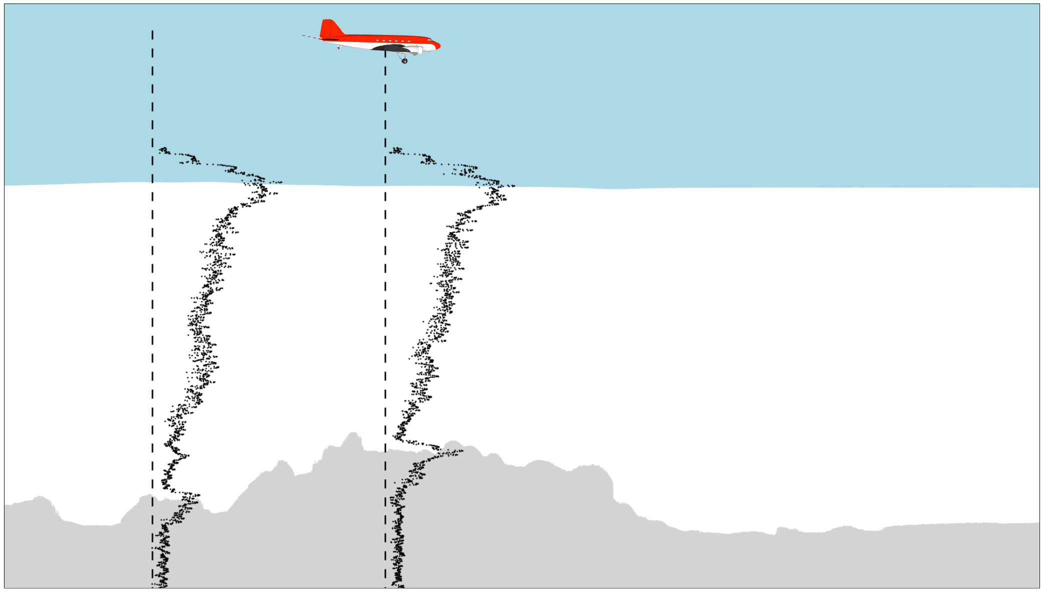
* so this peak corresponds to the ice/air interface



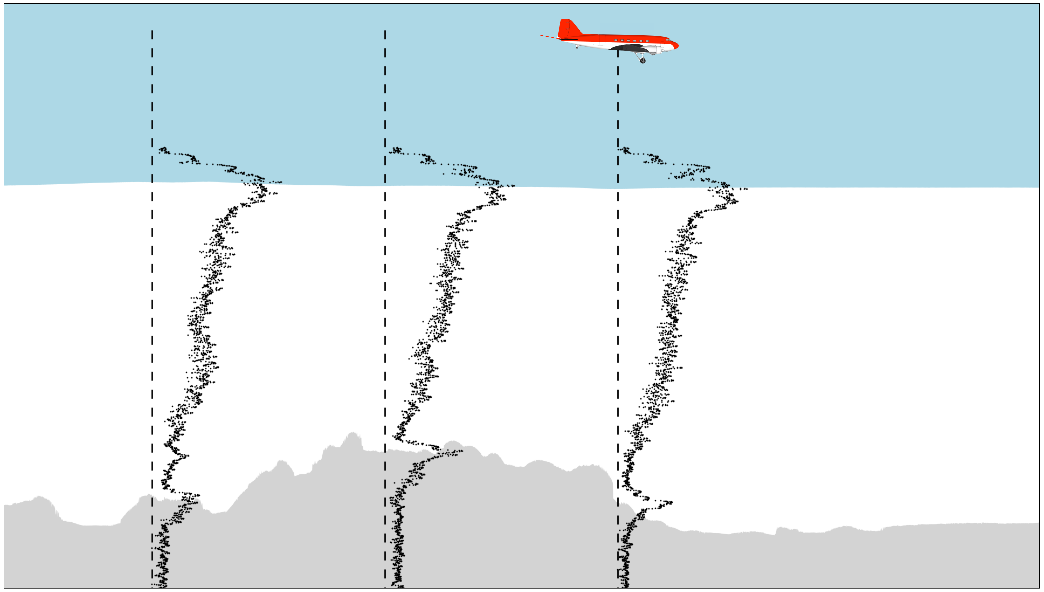
* these wiggles are layers within the ice



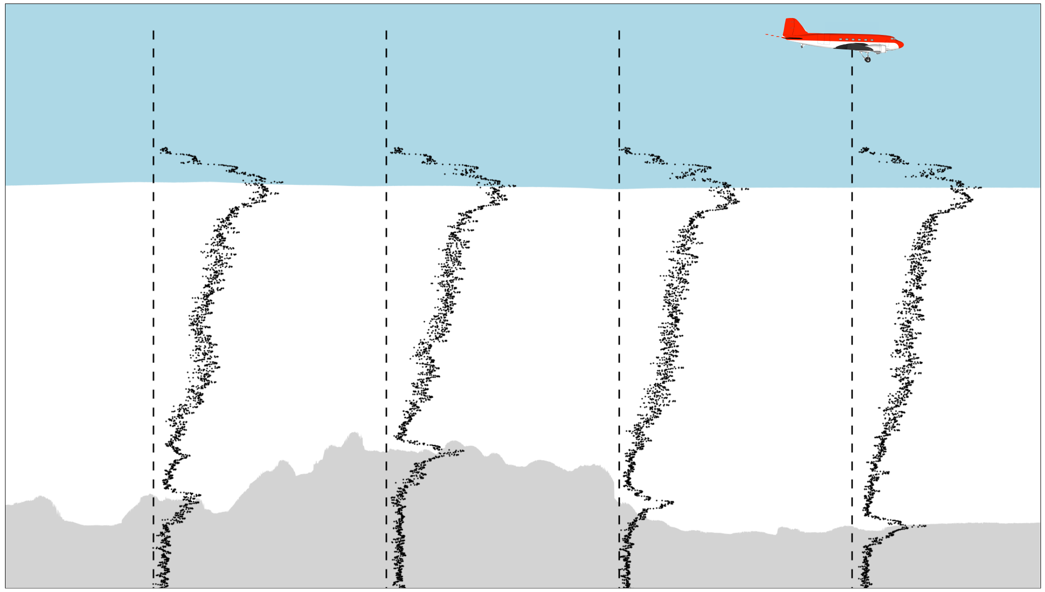
* this smaller peak is the ice/rock interface



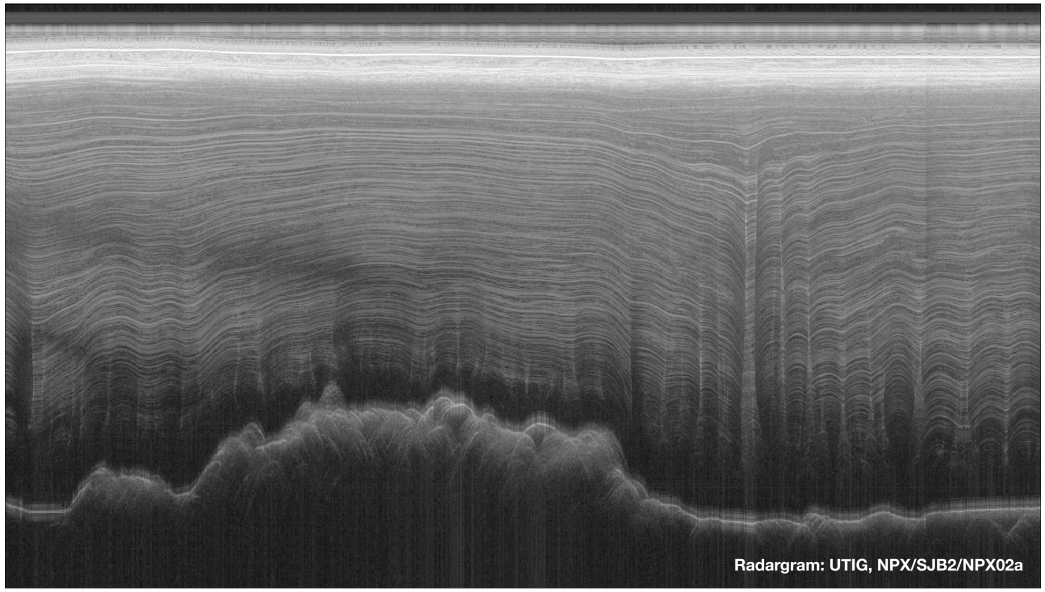
As the airplane flies along, the radar pulses thousands of times a second.



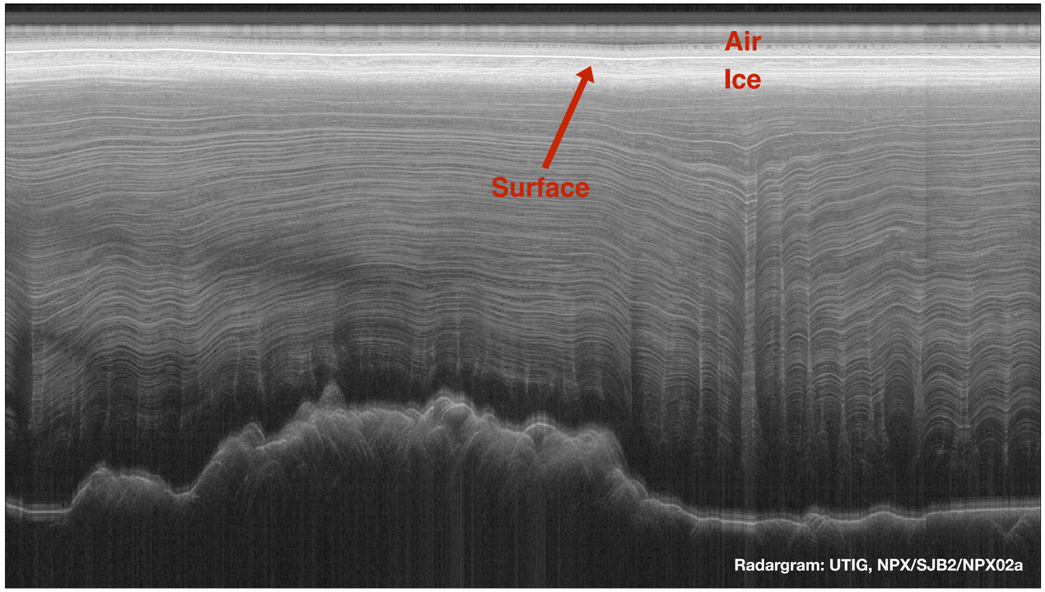
....



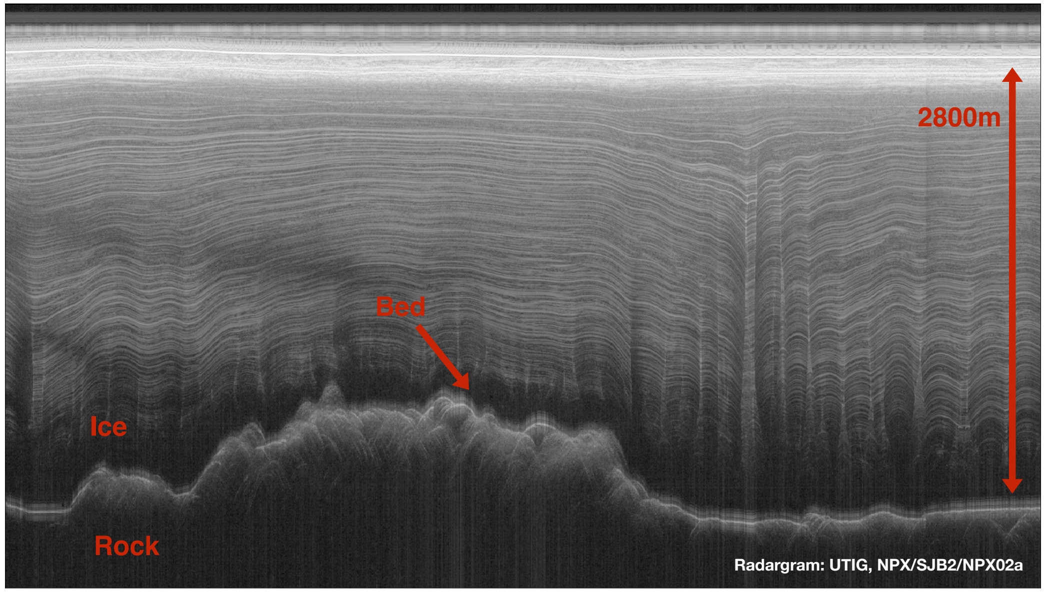
We combine all those pulses in an image like



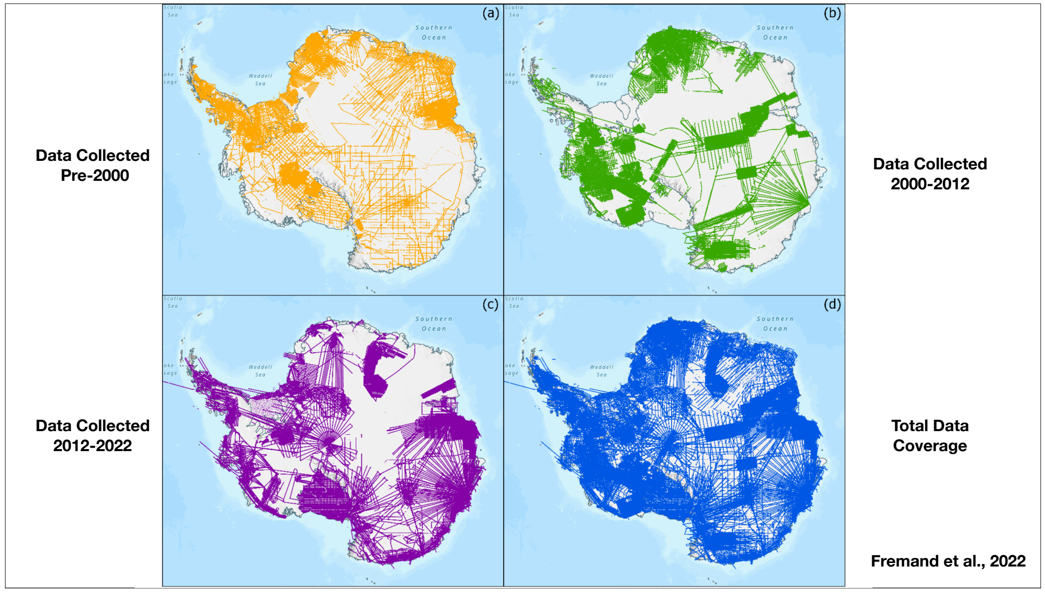
…this, which can be thought of as a 2D cross-section of the ice sheet.
Each column of pixels in this image corresponds to the returned intensities from a single pulse of energy.



The top bright line is the surface.



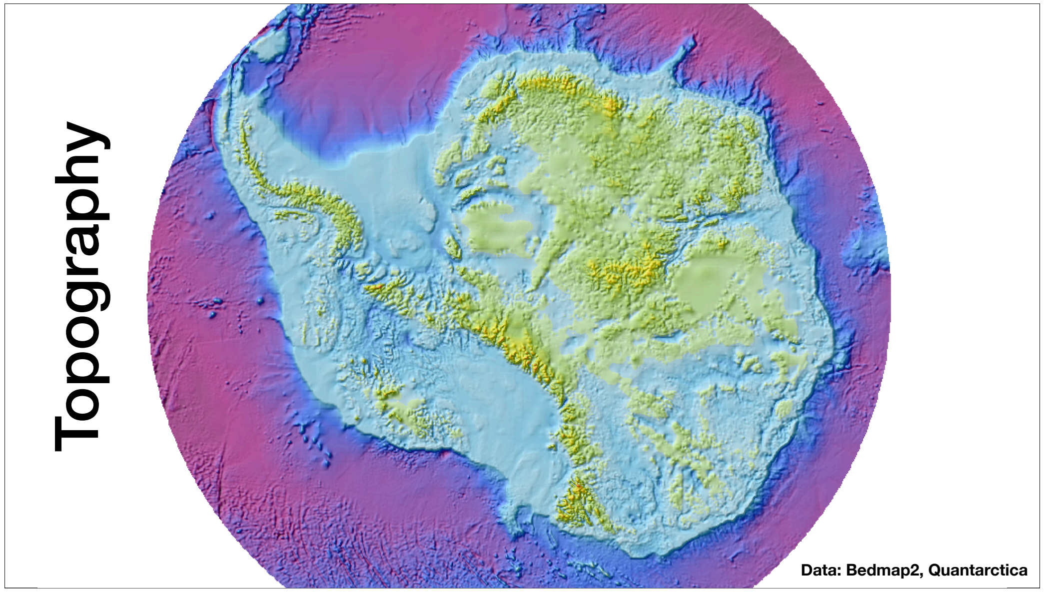
and The brighter reflection at the bottom is the ice/rock interface.
since ice is almost transparent to radar energy, we can easily see through several kilometers.



Over the last 6 decades, a lot of this data has been collected around Antarctica, as shown in this figure from a paper about BEDMAP3, which is a compilation of all the radar depth measurements collected in Antarctica.

However, while I say “a lot of data” … this data volume is measured in Terabytes, not Petabytes like some of y’all are used to =)

And, since I assume most of y'all don't spend as much time looking at maps of Antarctica as I do, I just wanted to give you a sense of the scale of the continent.

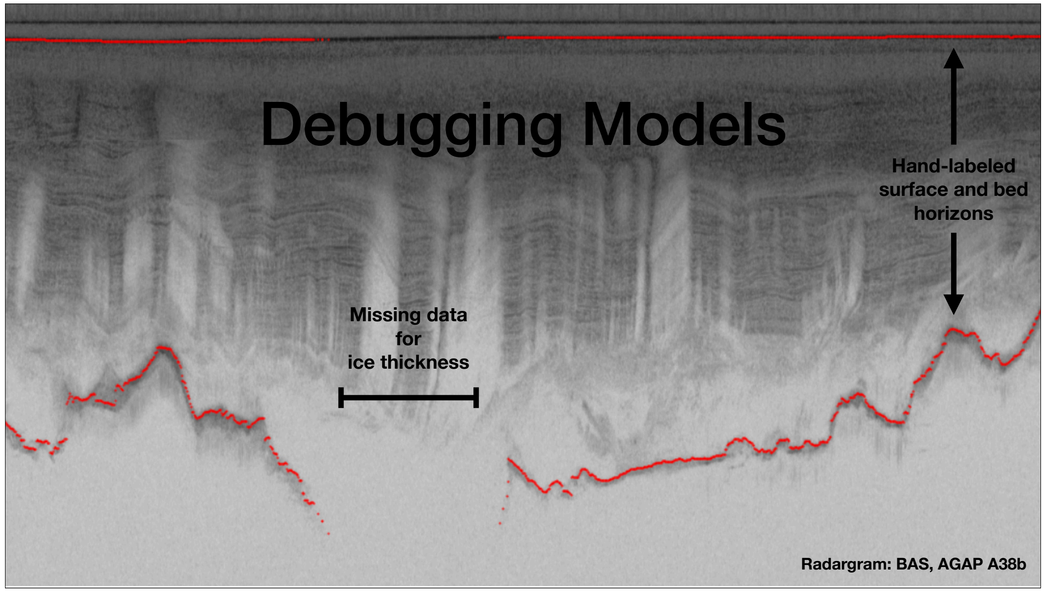


OK, so why do we go to so much trouble to collect this data?

For starters, data about ice thickness from these radargrams are used to build our models of Antarctica's sub-ice topography, which is a fundamental data product for anybody studying the continent.

As a society, we care about this because as warming oceans melt the ice, Antarctica will contribute to sea level rise, and we need to understand how much ice is vulnerable to melting, and on what timescales.

This processed ice thickness data, is easily visualized with existing tools - you can just drag a grid into QGIS … but the raw radargrams are not as easy to use. So, who needs access to the raw radargrams?

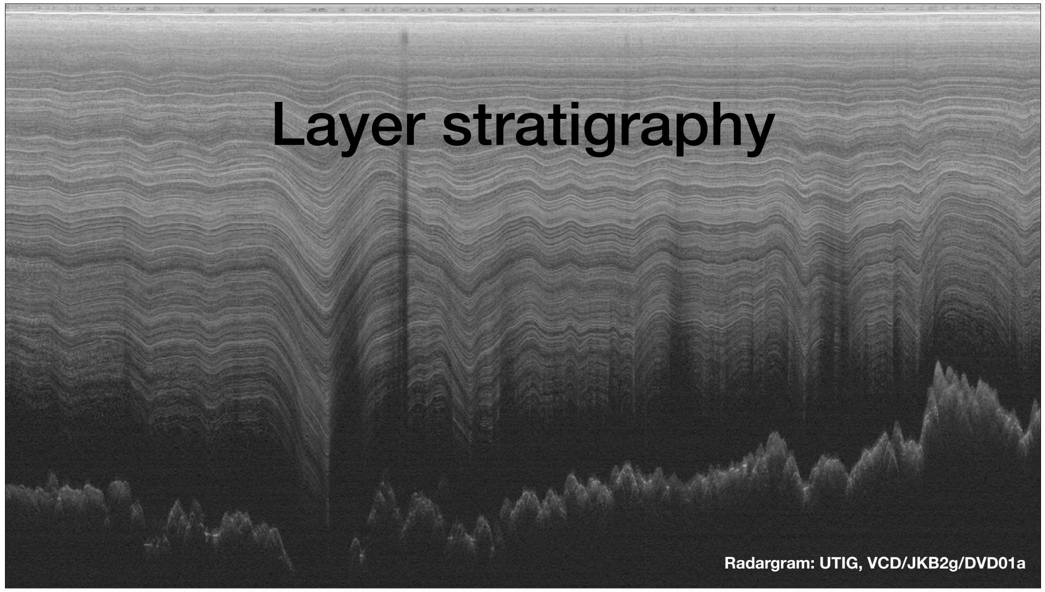


When modelers use gridded maps, the data has necessarily been smoothed and interpolated. So, when a model doesn't converge, it is useful to know if that's due to a flaw in the math, a bug in the code, or to incorrect boundary conditions provided by the data.

For example, here is a radargram from BAS's AGAP campaign. Manually identified surface and bed horizons are shown in red, and we can see a big gap on the left of the image where there were no radar returns from a deep valley. So, in that region, the gridded map will probably smooth over this valley and underestimate ice thickness. So then, ice flow models and water routing models may have problems.

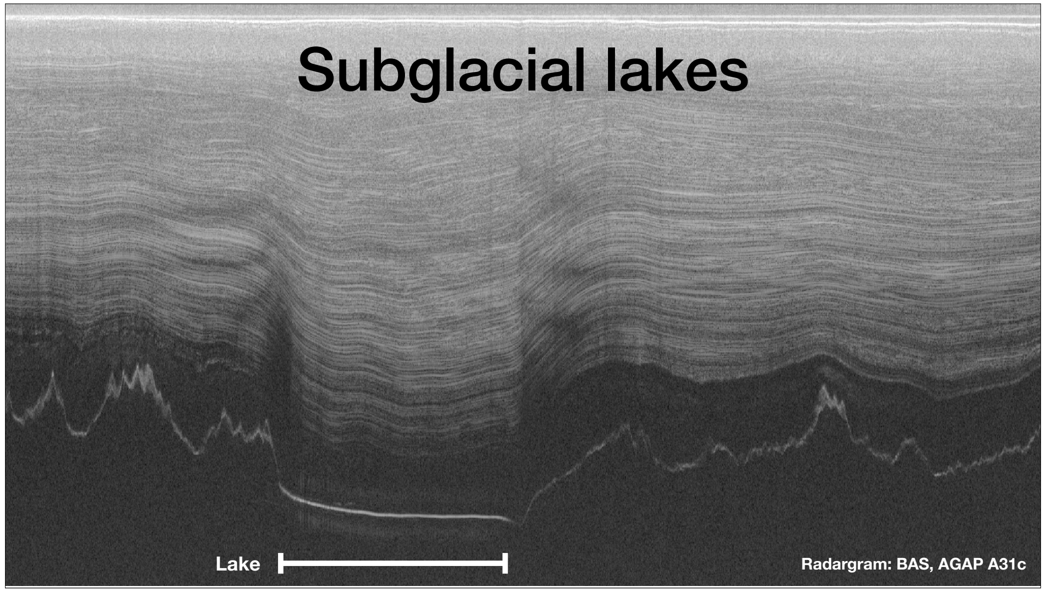
————————-

Bell, Robin E., et al. "Widespread persistent thickening of the East Antarctic Ice Sheet by freezing from the base." Science 331.6024 (2011): 1592-1595.
AGAP A38b



In addition to information about the terrain underneath the ice, radar also reflects slightly from changes in either chemical composition (so various impurities) or the ice's fabric — which is the orientation of the crystals. We assume that those are isochrons — that is, they correspond to ice deposited at the same time. Researchers trace these horizons across many intersecting radar lines and compare them to data from drilled cores in order to get an age-depth map of the ice sheet. Among other things, this is then used in the hunt for old ice.

Multiple countries currently have projects trying to find and drill for million-plus year old ice.
This is motivated by wanting to know more about the earth's climate history.
As snow falls and is compressed into ice, air bubbles are trapped, which can give us a record of atmospheric gases at that time.
——————



We can also use radar to identify subglacial lakes

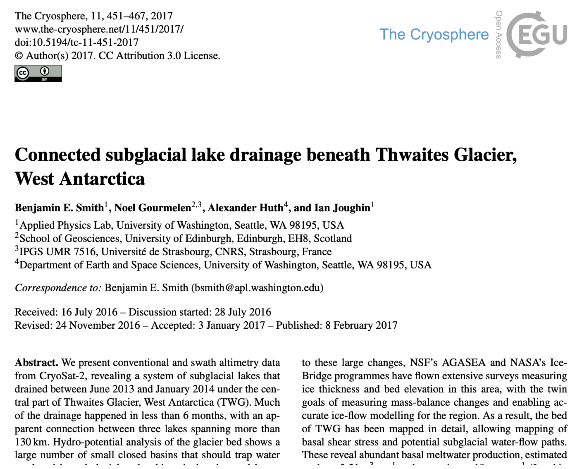
These are interesting targets for biologists and astrobiologists because some of these lakes are ecosystems that have been fully isolated for millions of years. They are also terrestrial analogues for icy moons, which are a primary target in the search for extraterrestrial life.

Subglacial lakes are able to exist because Ice is a pretty good thermal insulator. So … kilometers of ice basically serve as a blanket for the rocks below, and in places, geothermal flux from the core is enough to melt that ice from below. When that meltwater collects into stable subglacial lakes, they show up as these bright, smooth, hydraulically-flat reflectors.

——-

The meltwater also forms drainage networks under the ice, feeding in to the southern ocean.

Current State—Radargram usage



The Cryosphere, 11, 451–467, 2017
www.the-cryosphere.net/11/451/2017/
doi:10.5194/tc-11-451-2017

The Cryosphere
Open Access
EGU

Connected subglacial lake drainage beneath Thwaites Glacier, West Antarctica

Benjamin E. Smith[1], Noel Gourmelen[2,3], Alexander Huth[4], and Ian Joughin[1]

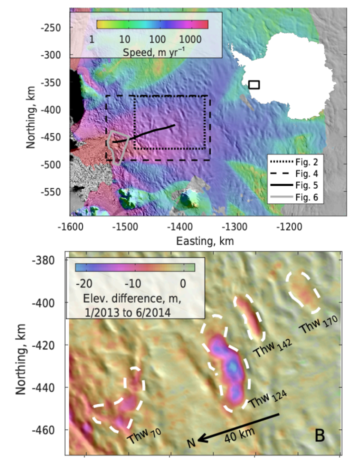
[1]Applied Physics Lab, University of Washington, Seattle, WA 98195, USA
[2]School of Geosciences, University of Edinburgh, Edinburgh, EH8, Scotland
[3]IPGS UMR 7516, Université de Strasbourg, CNRS, Strasbourg, France
[4]Department of Earth and Space Sciences, University of Washington, Seattle, WA 98195, USA

Correspondence to: Benjamin E. Smith (bsmith@apl.washington.edu)

Received: 16 July 2016 – Discussion started: 28 July 2016
Revised: 24 November 2016 – Accepted: 3 January 2017 – Published: 8 February 2017

Abstract. We present conventional and swath altimetry data from CryoSat-2, revealing a system of subglacial lakes that drained between June 2013 and January 2014 under the central part of Thwaites Glacier, West Antarctica (TWG). Much of the drainage happened in less than 6 months, with an apparent connection between three lakes spanning more than 130 km. Hydro-potential analysis of the glacier bed shows a large number of small closed basins that should trap water

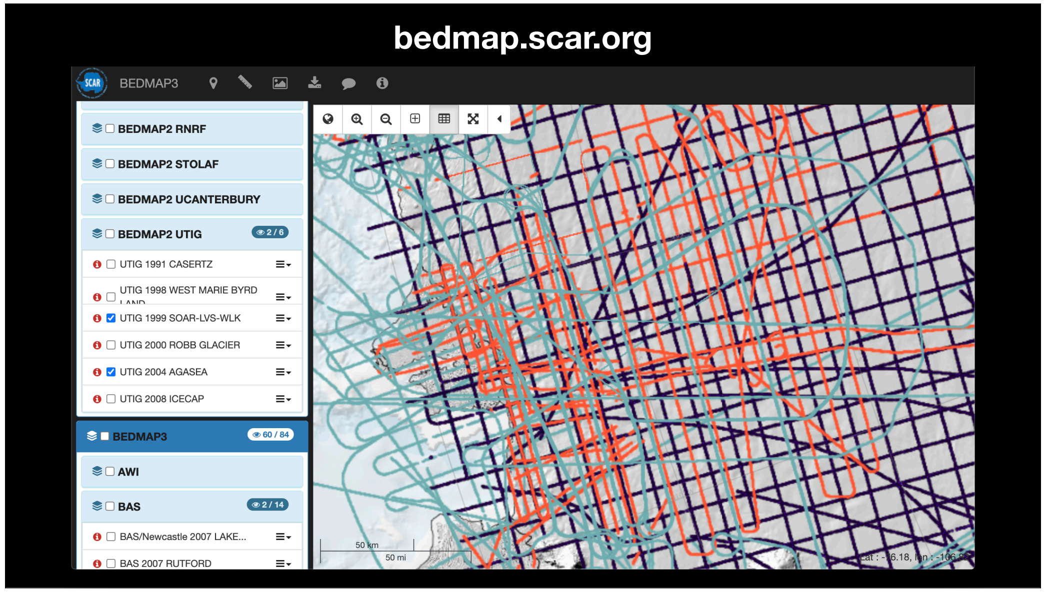
to these large changes, NSF's AGASEA and NASA's Ice-Bridge programmes have flown extensive surveys measuring ice thickness and bed elevation in this area, with the twin goals of measuring mass-balance changes and enabling accurate ice-flow modelling for the region. As a result, the bed of TWG has been mapped in detail, allowing mapping of basal shear stress and potential subglacial water-flow paths. These reveal abundant basal meltwater production, estimated



OK, so that's a sampling of reasons you might want to use this raw data. Right now, it's a lot harder than I think it should be to access these datasets.
One of this project's driving use cases is an early stage grad student—they're reading broadly, in search of a project, and want to look deeper into data relevant to the papers that they're reading.

As a concrete example, let's say the student just finished reading this 2017 paper by Ben Smith et al. presenting data about the surface expression of a set of active subglacial lakes beneath Thwaites Glacier.
Smith infers the presence of filling and draining lakes under the ice based on repeat-track satellite altimetry: these are regions of the ice surface that rise and fall in a way that isn't explained by ice flow dynamics but is consistent with water collecting and draining.

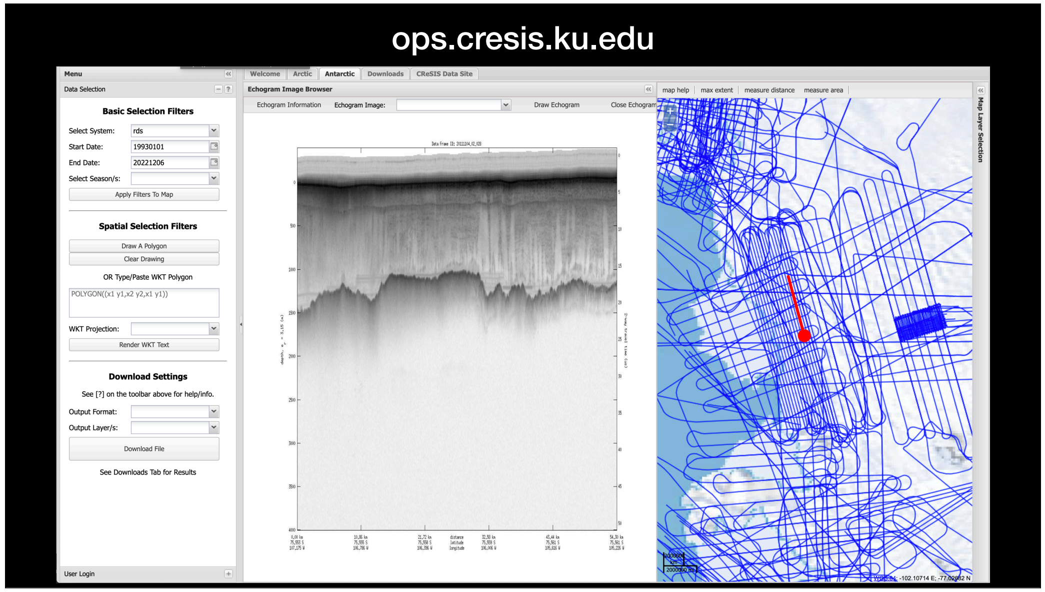
And the student is curious about the basal interface and englacial layers in the vicinity of these features. So … what would it take to indulge that curiosity?



OK, so first, we gotta find the data.

Right now, the SCAR bedmap data portal is probably the best tool for seeing what campaigns collected data in a given region.

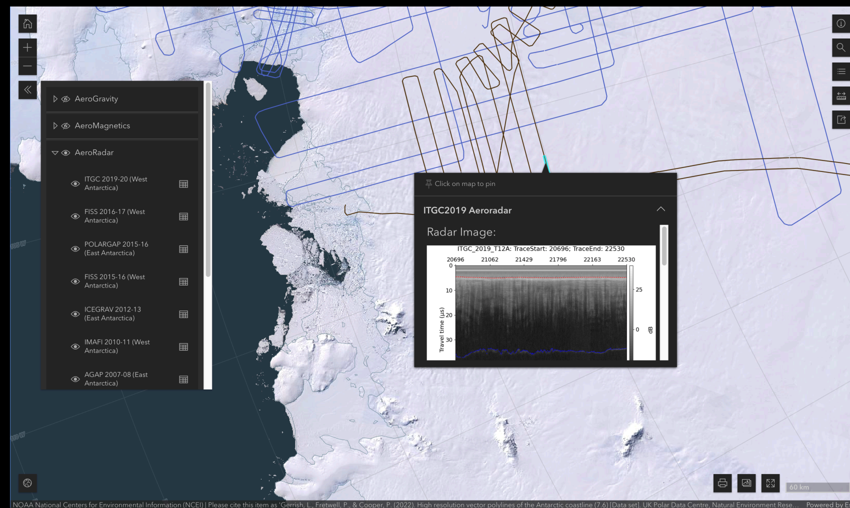
I've zoomed into Thwaites, and there are so many radar lines that they obscure the coastline.
We can see that UTIG, CRESIS, and BAS all have relevant datasets.



To find the CReSIS data, we'll go to Open Polar Server.

Clicking on a line lets you preview the radargram and download it right from the webpage.
However, you have to eyeball where the features of interest are, and which lines cross them; there's no way to upload geometries to their basemap.

https://www.bas.ac.uk/project/nagdp/



BAS recently released a very similar portal.

USAP-DC
U.S. ANTARCTIC PROGRAM DATA CENTER

About Contribute Search Services and Support Other Resources Contact Login

Dataset Information

Report Issue Edit

Radar Sounding Observations of the Amundsen Sea Embayment, 2004-2005

Data DOI: https://doi.org/10.15784/601436

Cite as

Chu, W., Culberg, R., Hilger, A., Jordan, T., Schroeder, D., Seroussi, H., et al. (2021) "Radar Sounding Observations of the Amundsen Sea Embayment, 2004-2005" U.S. Antarctic Program (USAP) Data Center. doi: https://doi.org/10.15784/601436.

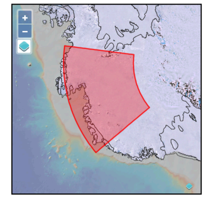
AMD - DIF Record(s)

USAP-1745137_1

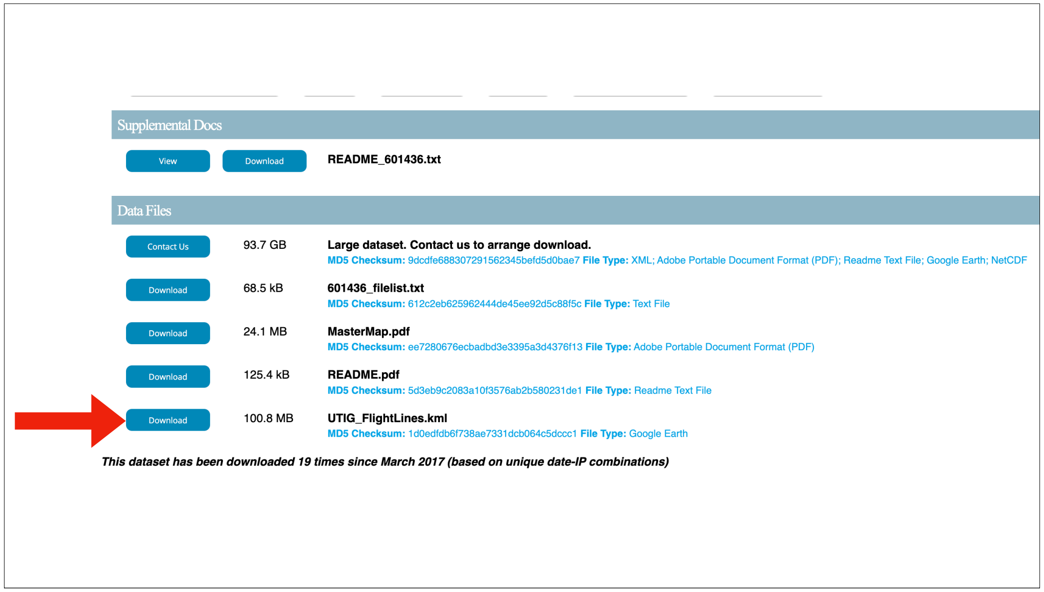
Abstract

The dataset contains radargrams from the 2004-2005 airborne radar sounding surveys on Pine Island Glacier and Thwaites Glacier as part of the BBAS and AGASEA projects. It also includes basal reflectivity and one-way attenuation rates derived from these radargrams. Radar data from the Pine Island Ice Shelf inland to the Bentley Sublglacial Trench were collected by the British Antarctica Survey (BAS) with the Polarimetric-radar Airborne Science INstrument (PASIN) radar sounder, operating at a center frequency of 150 MHz and 15 MHz bandwidth. Data over Thwaites Glacier were collected by the University of Texas Institute for Geophysics (UTIG) High Capability Airborne Radar Sounder (HiCARS) operating at a center frequency of 60 MHz with 15 MHz of bandwidth. Data are provided as 50km segments in NetCDF files, along with kml location files, and pdf files for browsing radargrams images by flight transect. Details of the processing methods are included in the associated README file. The processed data sets (reflectivity and attenuation) are provided as a single NetCDF file for each flight transect. Details of the calibration and processing procedures are provided in Chu, et al (in review).

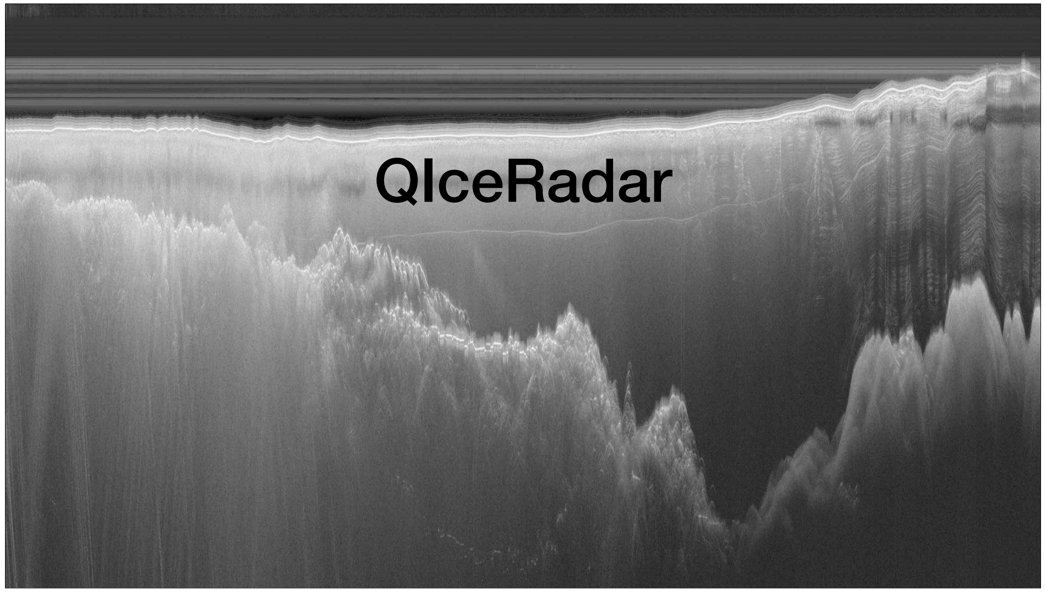
Creator(s): Chu, Winnie; Hilger, Andrew M.; Culberg, Riley; Schroeder, Dustin; Jordan, Thomas M.; Seroussi, Helene; Young, Duncan A.; Vaughan, David G.



And finally, the AGASEA radargrams were released via the USAP Data Center.
It doesn't have a slick web interface…



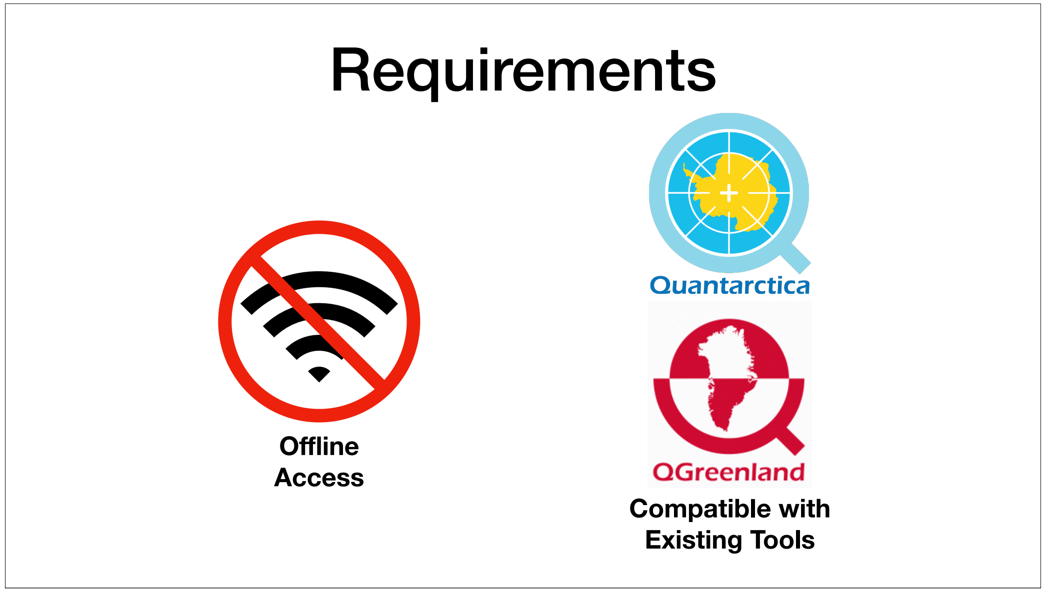
… but if you scroll down, you'll find a KML of all the flight lines.



That’s just finding and downloading the data.
Each file is in a different format, and the student will have to figure out how to:
* plot the radargrams
* Plot their GPS track
* Associate traces on the radargram with points on a map

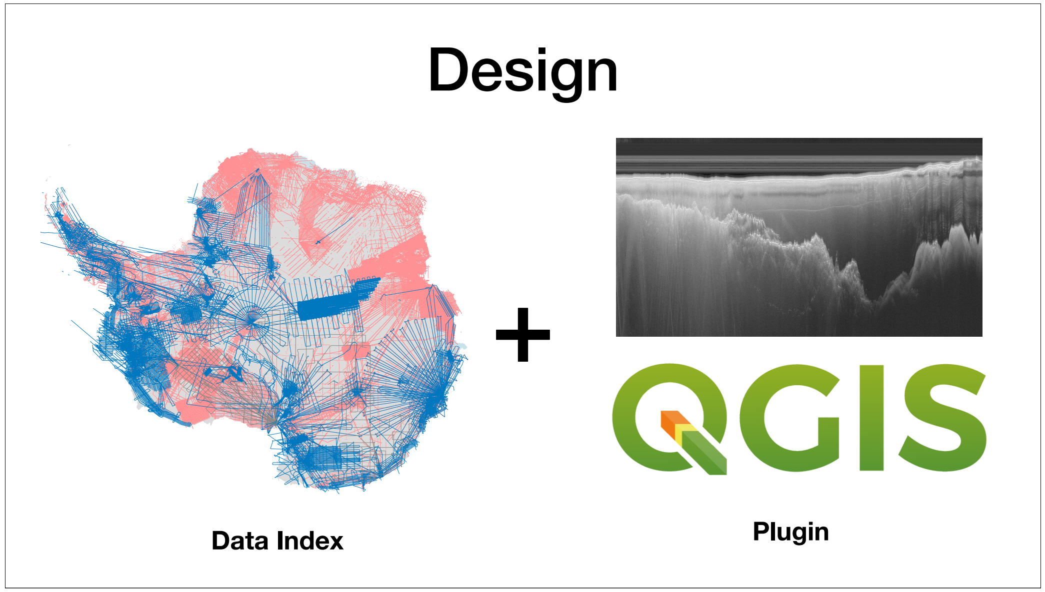
In all, this is hours to days of work for somebody who is somewhat comfortable with Python. I imagine that it would probably be pretty daunting for a non-programmer. It’s not insurmountable, but it’s also not trivial or something that you’d do to indulge idle curiosity, and it’s work that has to be repeated by every researcher.

I think this whole process should be a lot easier, so I proposed QIceRadar.



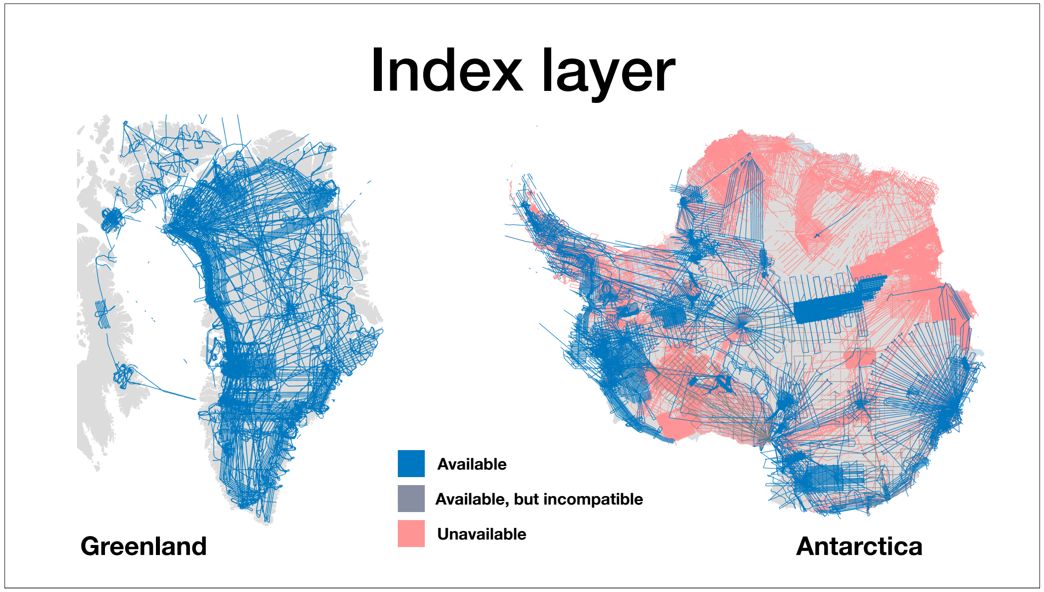
Offline access: a lot of my users work in the field. You’re not going to be using a web interface and streaming a cloud-optimized dataset over a satellite link. Maybe in a few years — but right now, Starlink doesn’t have good polar coverage =) So, beyond initially downloading the data of interest, which you could do before a field season, everything is done on the desktop without needing internet access.

Plays nicely with Quantarctica / Qgreenland — these two projects have done a fantastic job of making nicely styled map-view datasets readily available in QGIS for the polar research community. Thanks to this, many scientists are already using QGIS, so adding a new plugin provides a relatively small barrier to adoption.



the QIceRadar project has two primary components

* First, we have a database, in the form of a geopackage with associated style information. It contains the ground tracks for all known radar transects, along with metadata about who collected them and whether and where the data is available for download.
* Second, there is the QGIS plugin, which is used for viewing the radargrams in geospatial context.



The index layer is a useful product on its own, simply for seeing what data is available where.

This screencap shows all data that I'm aware of. QIceRadar supports Arctic as well as Antarctic data, though I'm mostly talking about Antarctica because it's a harder problem — there's a longer history of exploration, with more different research groups collecting data.

This database will be hosted at the Arctic and Antarctic data centers, once it is finalized and reviewed.

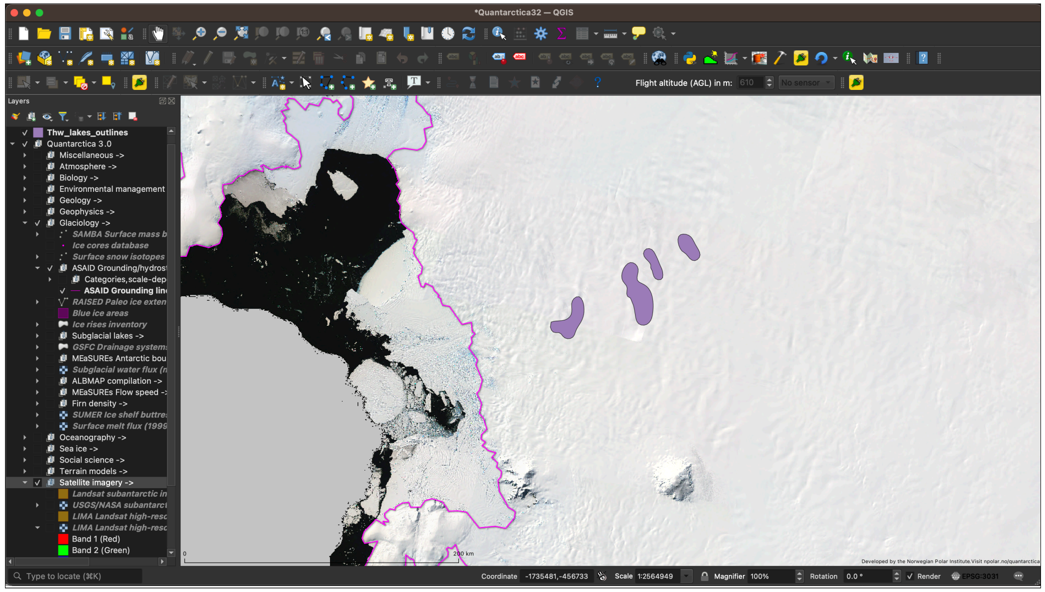
The styling indicates the availability of the data:
* Blue means that it is openly available, I have downloaded it, and I have succeeded at plotting + georeferencing the radargrams.
* Grey means that the data is kind of available, but not in a format that's compatible with this tool. The only grey dataset right now is a survey from the 60's — it was recorded photographically, and while scans of the film are available, there's no way to programmatically associate a scanned image with a map location.
* Finally, pink indicates radargrams that have not been released publicly. The major pink surveys include old US/UK data; as well as all German/Russian/Chinese datasets. I've been talking with the German researchers, and they are actually in the

process of releasing their data, so this map will look very different in a few years.

QGIS Plugin
(scientist POV)

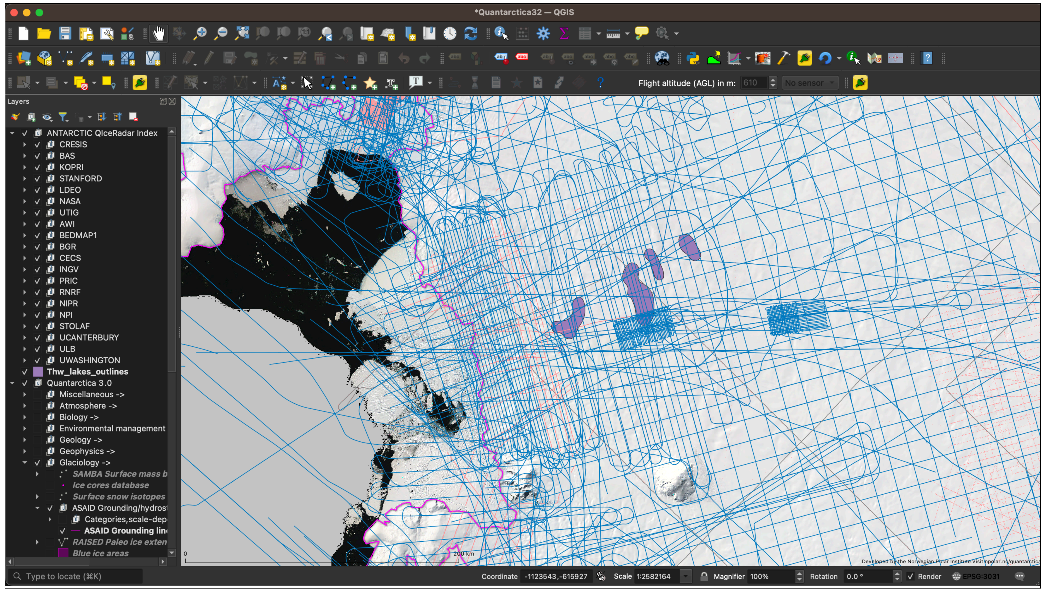
The other half of the QIceRadar project is the QGIS plugin. It uses the database as an index to what's available.

we'll look at it's operation first from a scientist's point of view

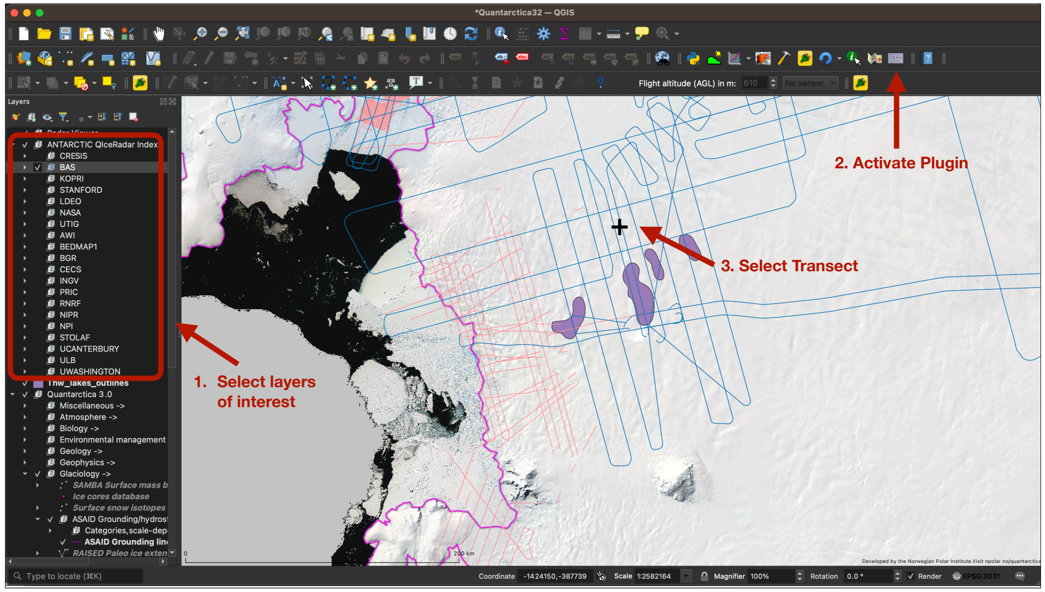


Coming back to our graduate student …
They already have QGIS installed w/ Quantarctica's data layers imported, and have downloaded the Thwaites lake outlines from the Smith paper's supplement

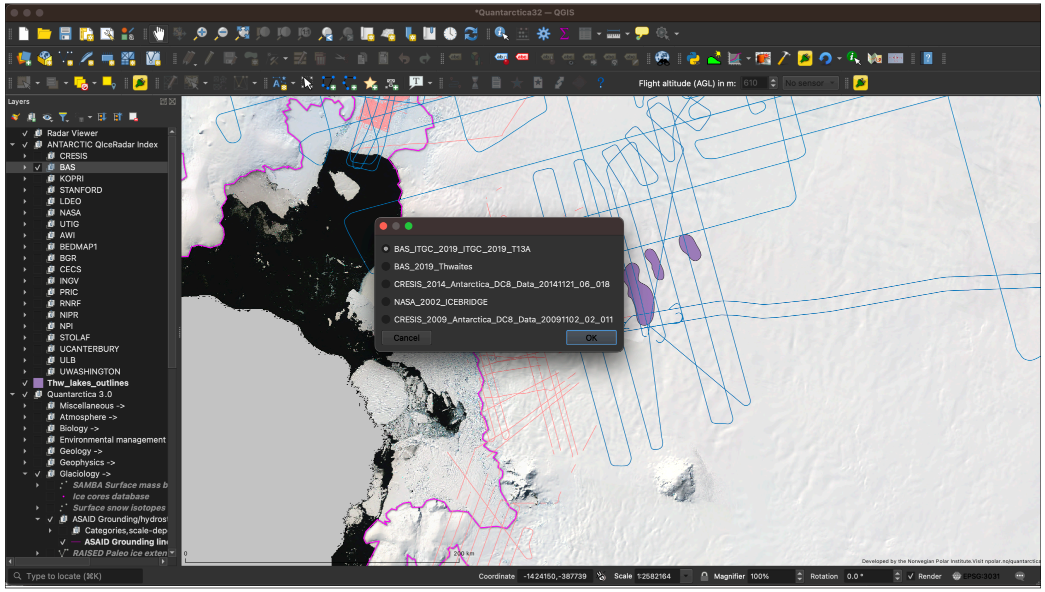
Here, the LIMA satellite mosaic is the basemap, with the ASAID grounding line in magenta, and the proposed lake extents in purple.



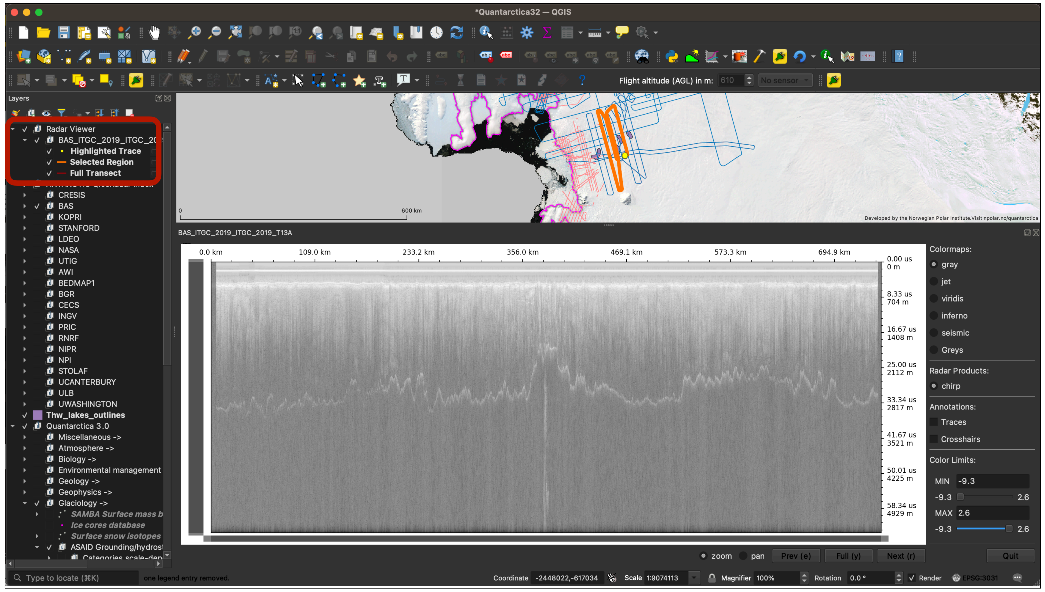
They obtain our index, drag it into QGIS, and install our plugin. There's a LOT of data in this region, so they decide to look at BAS's data first.



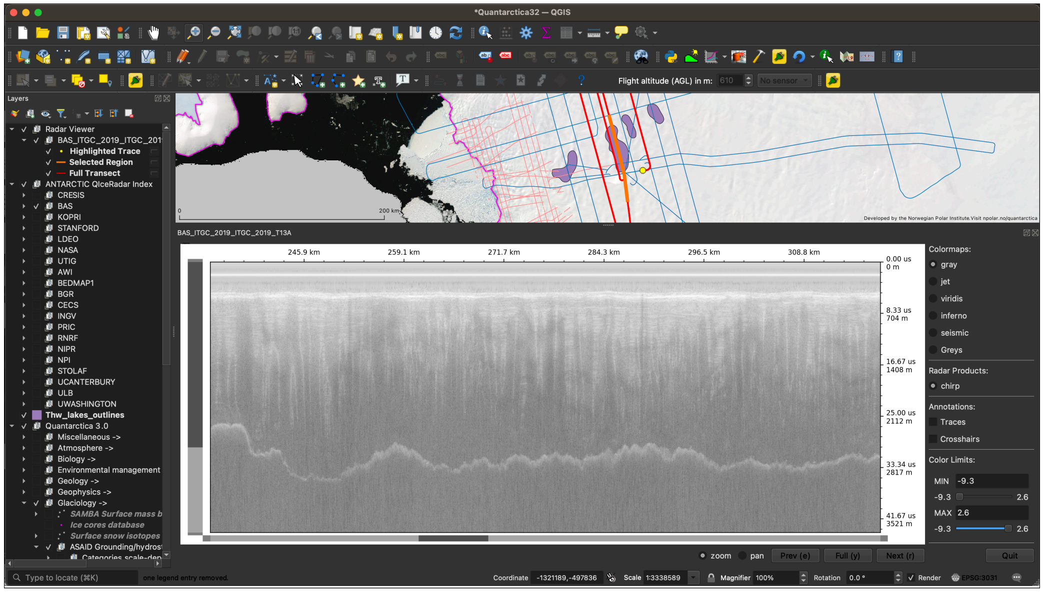
They’ve used the QGIS Layers Toolbar to deactivate all non-BAS layers within the Data Index group.
They activate the plugin by clicking on its icon in the toolbar,
then use the cursor to click on the line of interest



A dialog box pops up with a list of transects near their mouse click;
They choose the one they're interested in

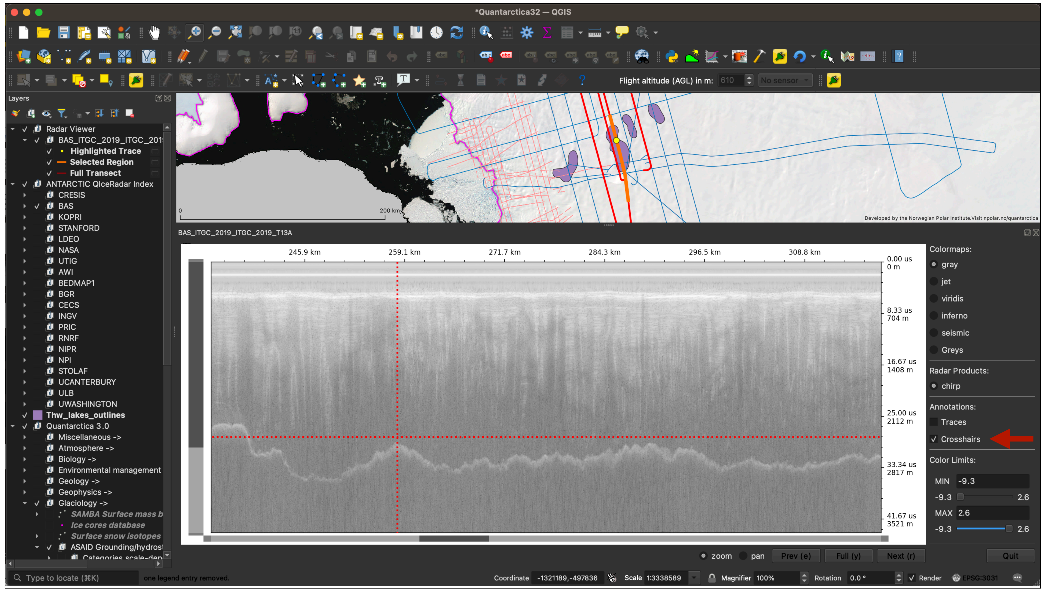


And a radargram pops up, along with a new set of layers in QGIS.
In the map view, they can now see the full flight path in orange, with a yellow dot at the start.

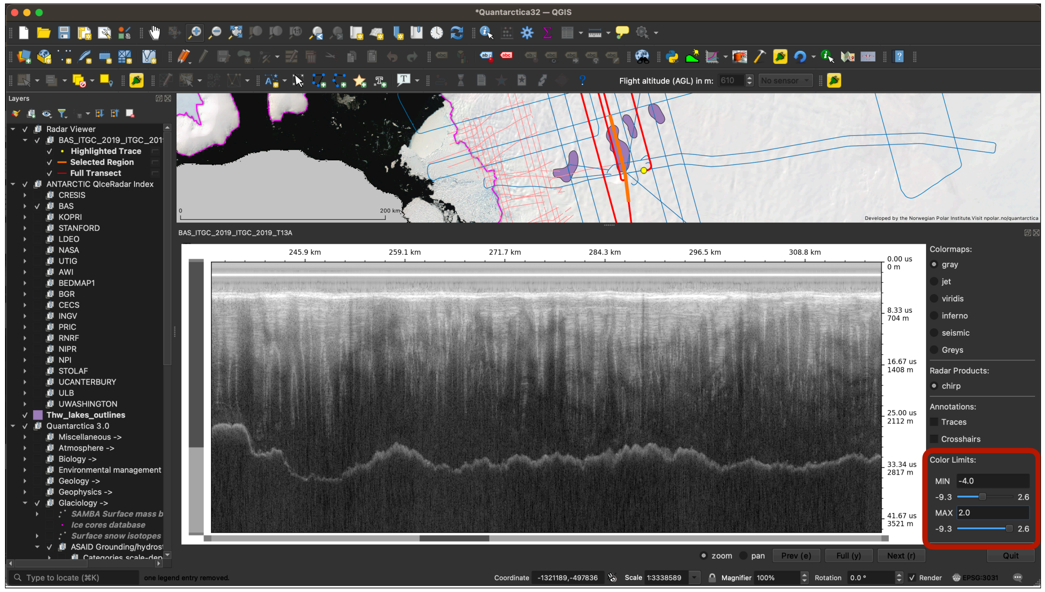


They zoom in to where the radargram crosses the largest active lake.

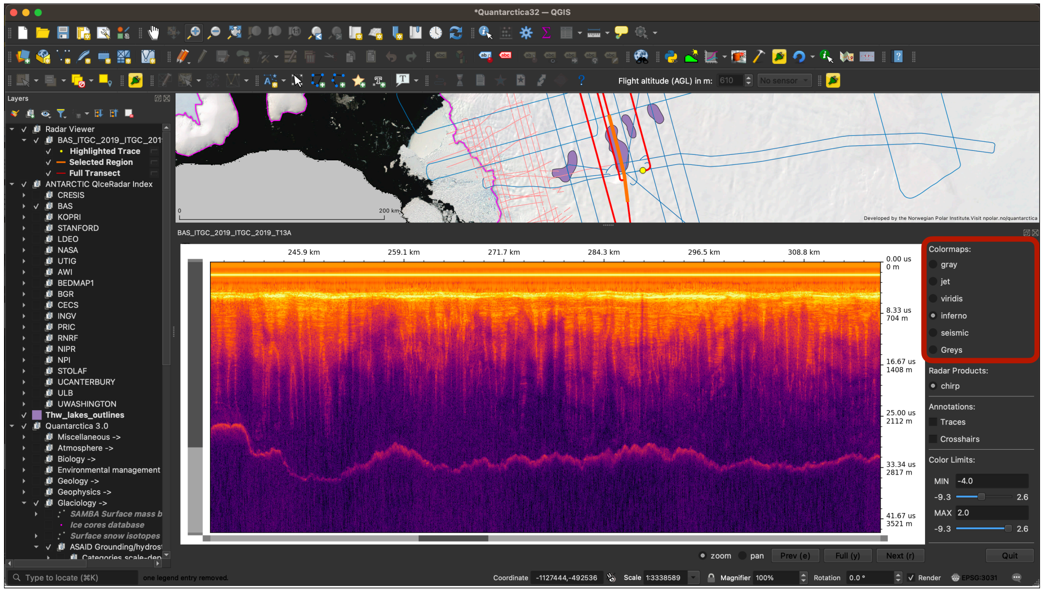
The orange line in the map view now shows the extent of the zoomed-in radargram, while the red line behind it shows the full flight path.



When they activate the cursor tool, crosshairs appear in the radargram.
As the user moves their mouse around the radargram, these crosshairs follow the mouse.
There is also a yellow dot in the map that follows the crosshairs, which enables precise georeferencing of features between the radargram and the map.
This is kind of like the profile tool, if you're familiar with that, but it's displaying a different data source, not just the QGIS layer's data.
Here, we have the crosshairs at the top of a hill, and can see that that's inside the hypothesized lake boundaries.



The radar viewer also supports changing the color limits



And color maps, to aid in interpretation.

QGIS Plugin
(Software/Implementation POV)

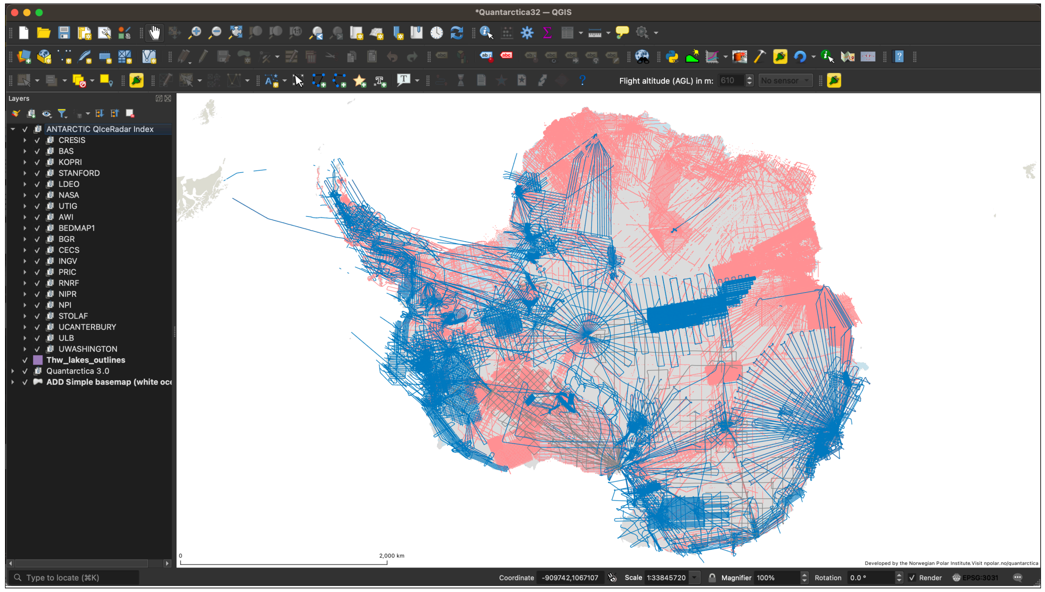
Let’s go through that again, but this time with a focus on implementation details rather than user experience.

- **Ujaval Gandhi's QGIS tutorials: https://spatialthoughts.com/**
- **Plugin Reloader: https://plugins.qgis.org/plugins/plugin_reloader/**

QGIS makes it reasonably easy to implement tools like this; there's a python API to interact with any element on the map, and they also have a python console that lets you experiment with the live map and test code snippets.

I didn't do anything particularly clever or noteworthy =) it was all very straightforward.

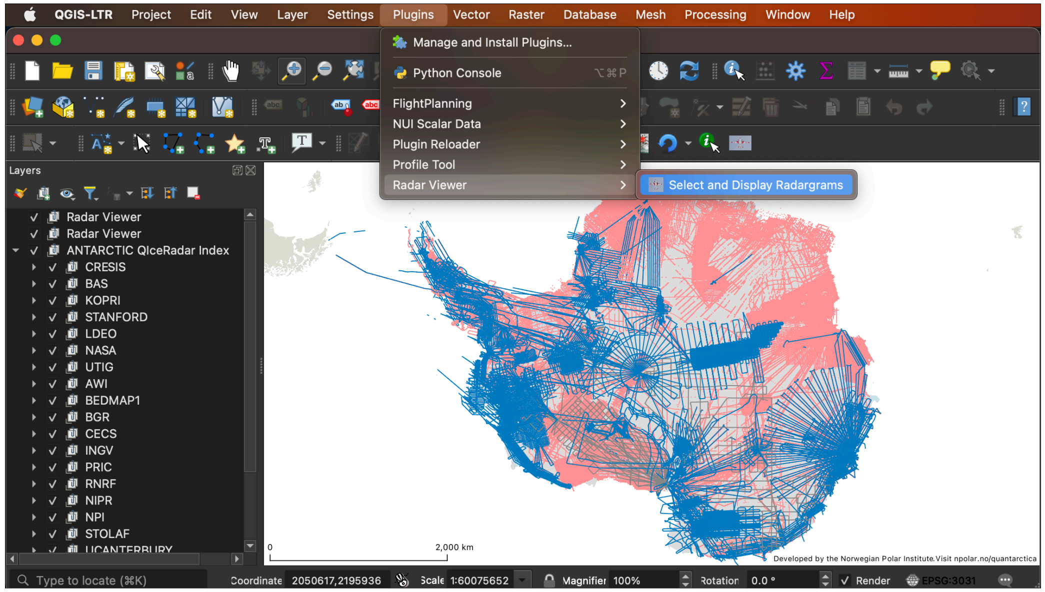
As far as resources go:
* As mentioned in an earlier talk, I found Ujaval Gandhi's tutorials to be incredibly useful.
* And, the plugin reloader plugin is absolutely indispensable for writing tools like this — it allows you to reload a plugin after changing the code without having to restart QGIS.



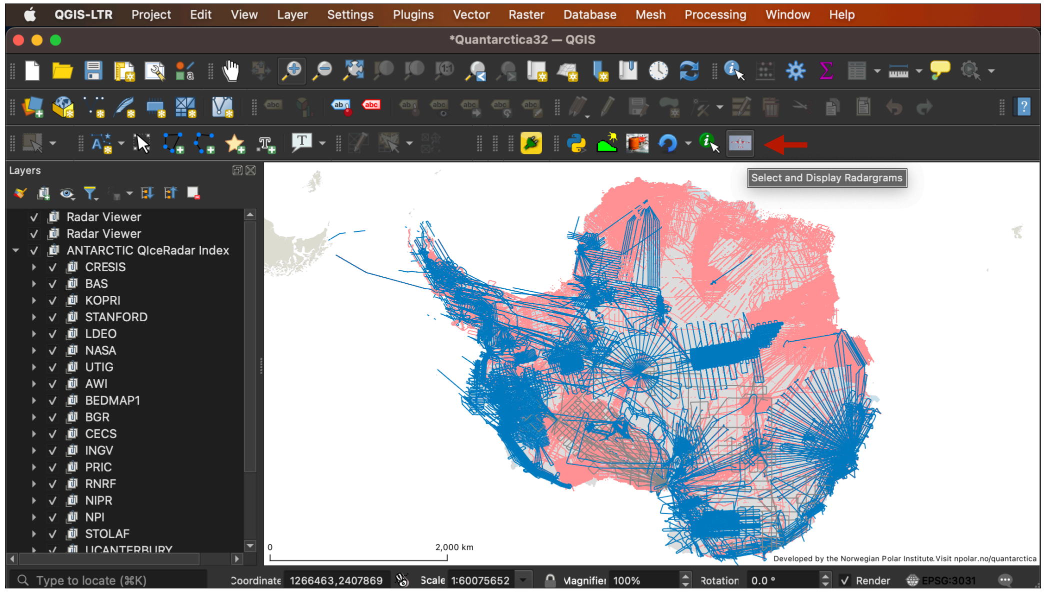
The database was actually the most annoying thing to figure out. I chose to use a geopackage, which is based on SQLite, so it can be queried programmatically as well as having drag-and-drop compatibility with QGIS.

* I wanted to have separate layers for each transect, so the user could style them independently / show and hide them; that ate up way too much memory and crashed QGIS.
* Instead, I wound up creating an overall group for the index, with sub-groups for each data provider. In each data provider group there are layers for individual expeditions, with separate line features for each transect.

* I think it's possible to encode styling in a gpkg file, but when I was working on it last year, it didn't seem fully supported, and it wound up being easier to include a .qlr file that handled styling.



And just as a note, this is a “GUI” plugin (vs. a “processing” plugin).
We want our users to be able to launch it either from the Plugins dropdown menu,



…or by clicking on a toolbar icon.

pyQGIS — Register Plugin

```
action = QtWidgets.QAction(QtGui.QIcon("airplane.png"),
                           "Select and Display Radargrams”,
                           self.iface.mainWindow())

action.triggered.connect(self.run)

self.iface.addPluginToMenu("&Radar Viewer", self.action)
self.iface.addToolBarIcon(self.action)
```

Setting this up is pretty easy.
In addition to a few metadata files to register your plugin you just need to:

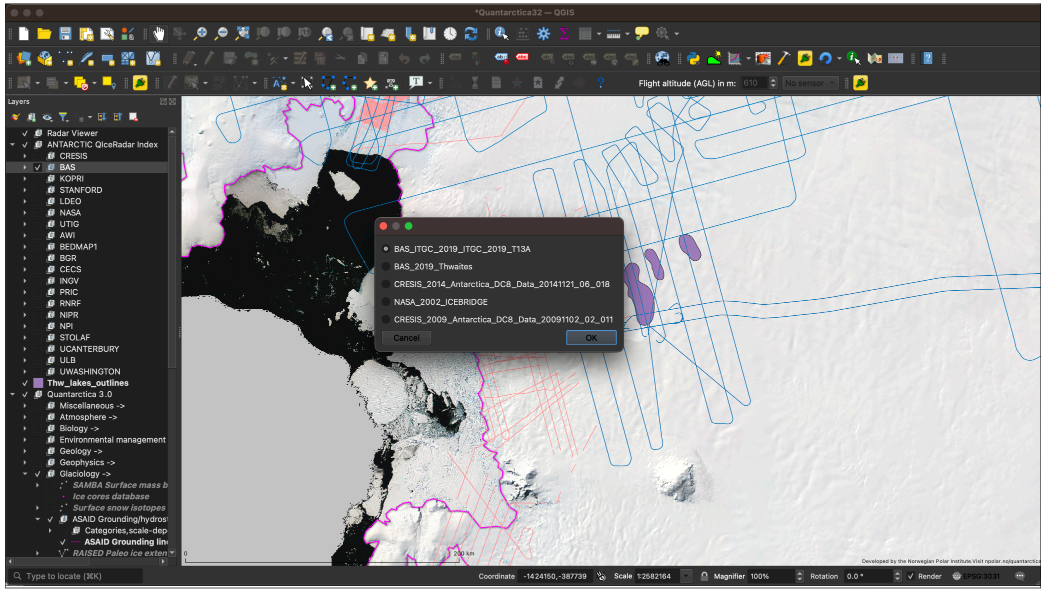
define an action, with an associated icon and text description.
connect our plugin’s “run” to the action.
And then insert it into the plugin menu and toolbar.

```
def run(self):
    if not config_is_valid(self.config):
        cw = RadarViewerConfigurationWidget(
            self.iface, self.config, self.set_config
        )
        cw.run()

    if self.spatial_index is None:
        self.build_spatial_index()

    selection_tool = RadarViewerSelectionTool(
        self.iface.mapCanvas(), self.selected_point_callback
    )
    self.iface.mapCanvas().setMapTool(selection_tool)
```

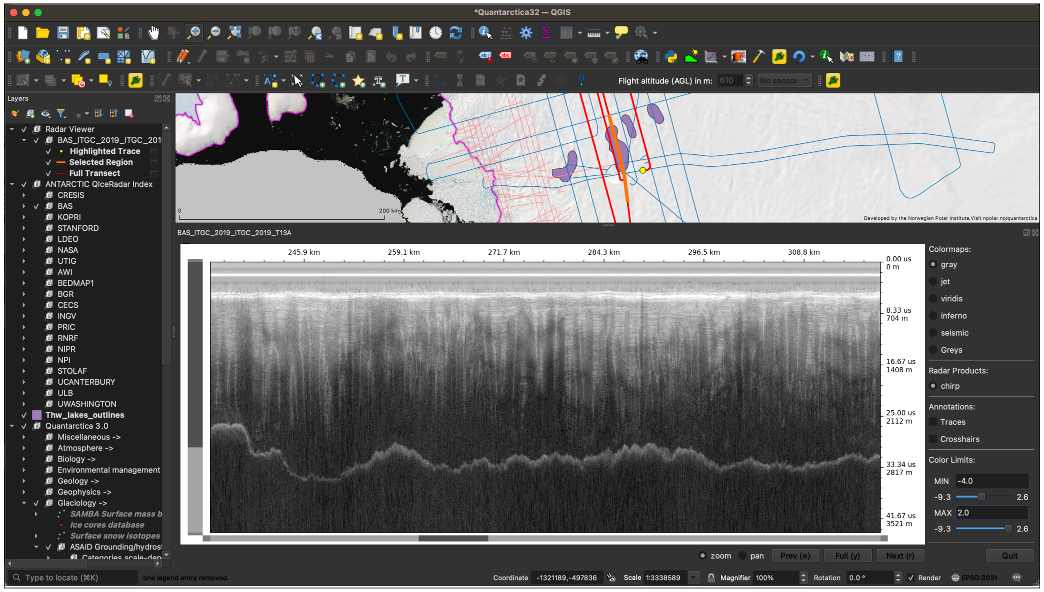
And then for the run method…
On the first load of the plugin, it prompts the user for configuration information — in this case, the root directory where radargrams are stored — using a QDialog widget.
This is saved globally using Qsettings, so on future runs the user won't need to reconfigure the plugin.

Then, if this is the first time the plugin has been activated since it was loaded, it builds a spatial index with all of the line segment geometry for the radar surface tracks. (Not just the bounding boxes). This step is a bit slow (~10 seconds). We don't cache its value across sessions, since the database may change.

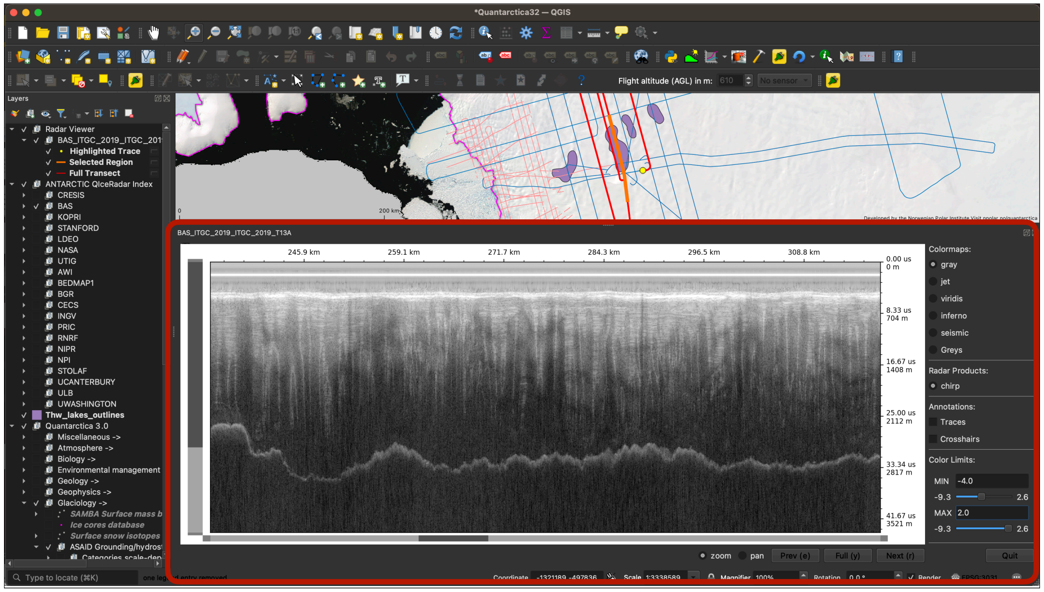
Finally, we set the map tool, which controls what happens when the user clicks on the map with our plugin active. Our implementation uses the previously-generated spatial index to quickly find the transects closest to the mouse click, and then pop up another Qdialog.



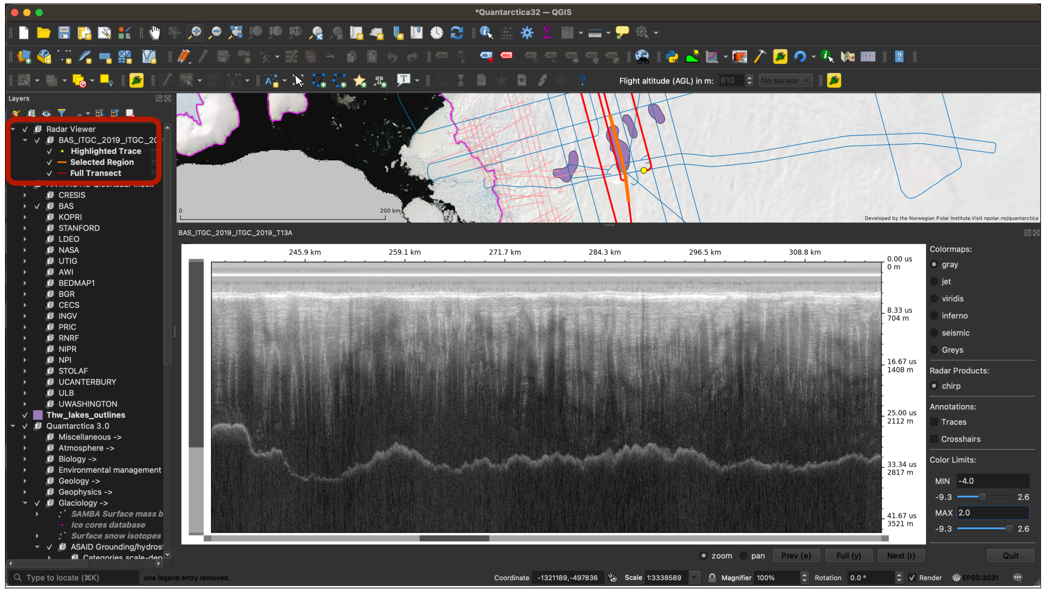
We give the user a choice of nearby transects: there are often repeat lines that totally overlap, because scientists are interested in change over time, so the user needs to be able to specify which one to open.



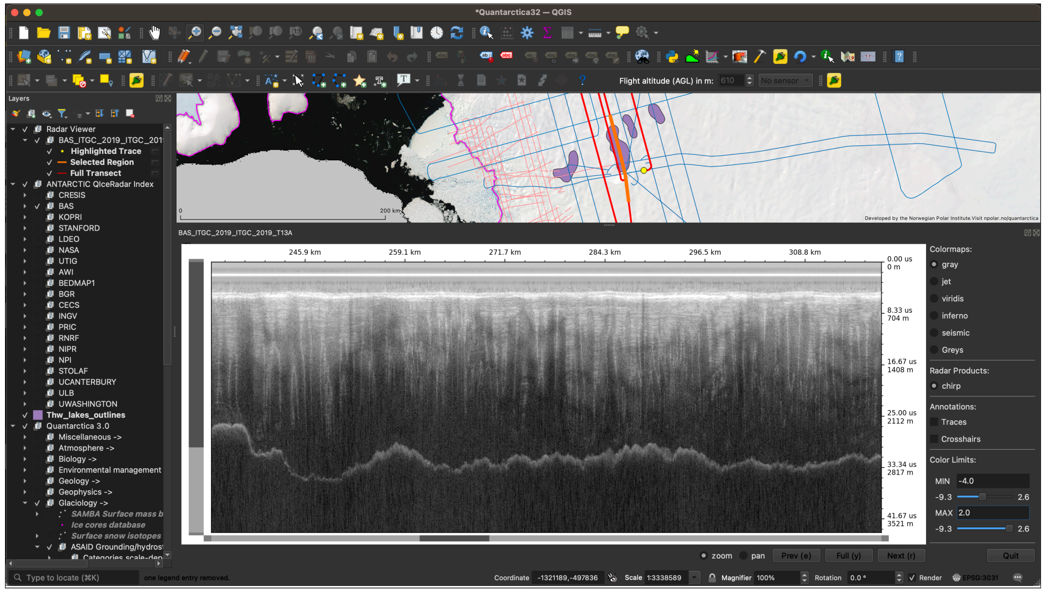
After the transect was selected, two things happen:



* First, the radargram viewer shows up as a DockWidget. It is initialized with the full path to the datafile, as well as the necessary metadata specifying the file format.



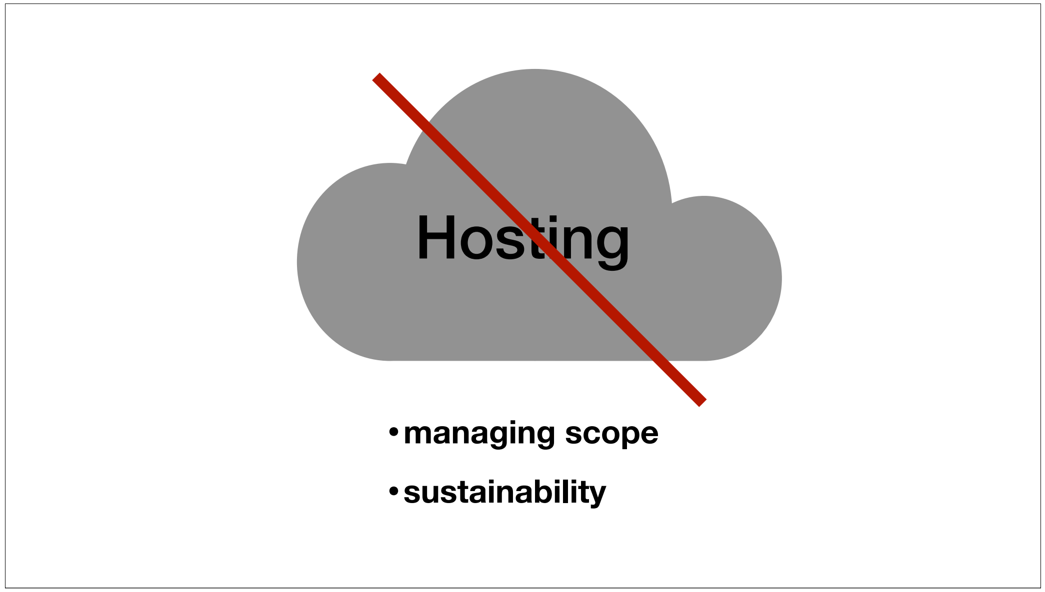
* Second, we add a group for the transect to the QGIS project, and add in-memory layers that programmatically display the full length of the radargram, the currently-displayed section, and the currently-highlighted trace.



I developed the radargram viewer as a standalone PyQt application. Integration with QGIS only required adding 3 callbacks. The callbacks

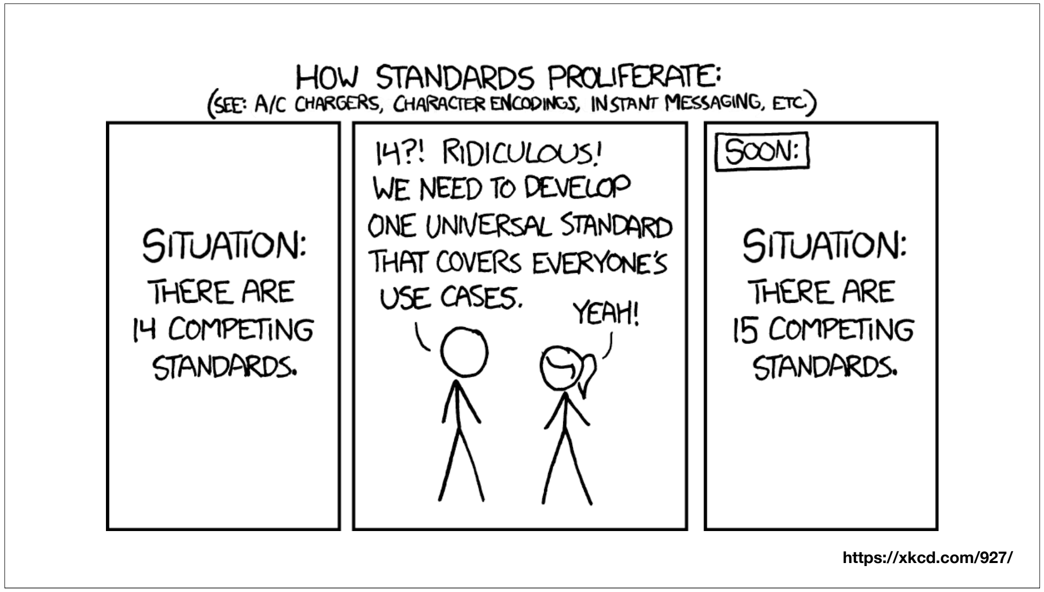
* update the layer showing the current trace
* Update the layer showing the currently-selected region
* clean up after the viewer is closed.

While the software is the most fun part for me, I think the most valuable part may be the data wrangling.
It has also been rather time consuming.
I hope to enable other tools to be built, using this index.



I’ve intentionally chosen to not try to host anybody else’s data — I want whatever national data center the researchers have chosen to stay the canonical source for the data. Partly this is due to scope: this is a small, single-person project, and I want to provide a useful tool that is easy to maintain, and NOT come up with the fanciest most ambitious project; and partly it’s sustainability: I want this project to continue to be useful even if my next grant isn’t funded to pay for ongoing cloud costs in perpetuity.

This means that I’m stuck with whatever download procedure and data formats the providers have chosen. I’m lucky if it’s possible to download the data in megabyte chunks rather than a single multi-gigabyte tar file. I’m also lucky if it’s possible to programmatically download the data — in one case, access involves emailing somebody, having them upload data to google drive, and emailing you the link. So, I can’t fully automate data access for my users, but I can at least provide instructions.



As I mentioned, every institution uses a different format for these data; and many institutions even release different field seasons in different formats.

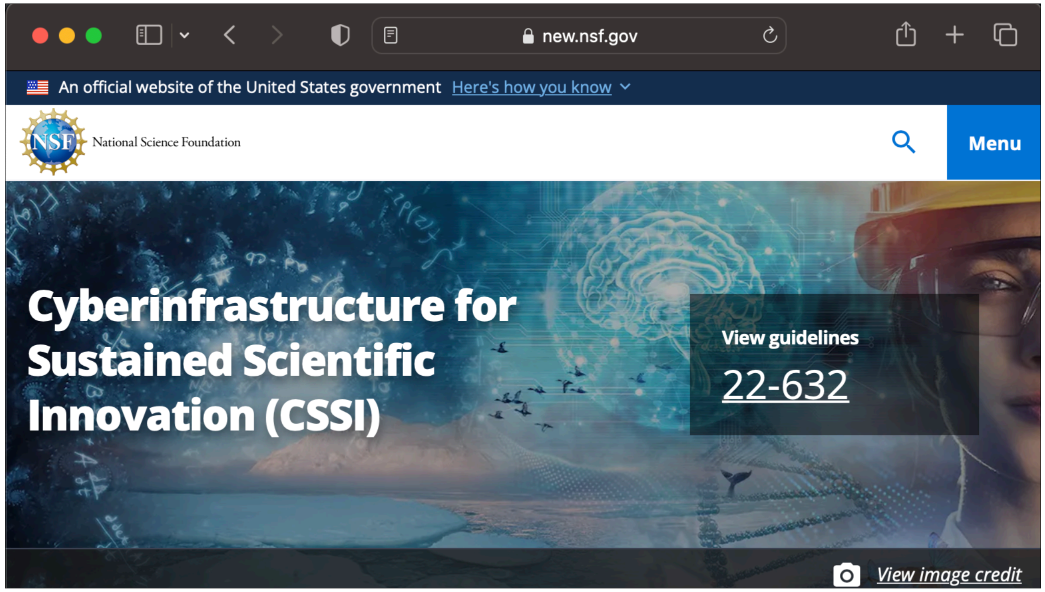
This is partly inherent in the type of data — these are custom instruments, and the people building/operating them are constantly making improvements, and are trying to get the most possible out of them. So rather than using a common format with standardized fields, they use one that reflects the peculiarities of each season.

So, for now, I've just written code that interfaces with all of the formats from the major data providers.

There is a funded project trying to standardize data processing pipelines and data formats across the major US institutions collecting this data; if they succeed, QIceRadar will support their eventual format, in addition to the major legacy ones.

Acknowledgements

This project involves collaborating with a bunch of people, even if they aren't directly contributing code



Finally, this project is funded by the NSF Cyberinfrastructure for Sustained Scientific Innovation program.

I was pleasantly surprised to find that there are mechanisms to fund projects that don't count as research in either engineering or science, but that instead are engineering that enables science.
So, while I wouldn't say that winning an NSF grant is necessarily an *easy* path to funding, it's an option.

Thank you! Any questions?