FINAL POSTER

Remington Rohel - August 16, 2023

Mission

Be an expert on a project that improves the world around me.

Goal #1: Learn a new programming language (Rust)

-> It's more secure than most languages

-> I am interested in becoming proficient in more languages

Goal #2: Make a 5-year plan

-> I want to be more intentional in planning my life

-> I don't have a good idea of what opportunities are available to me

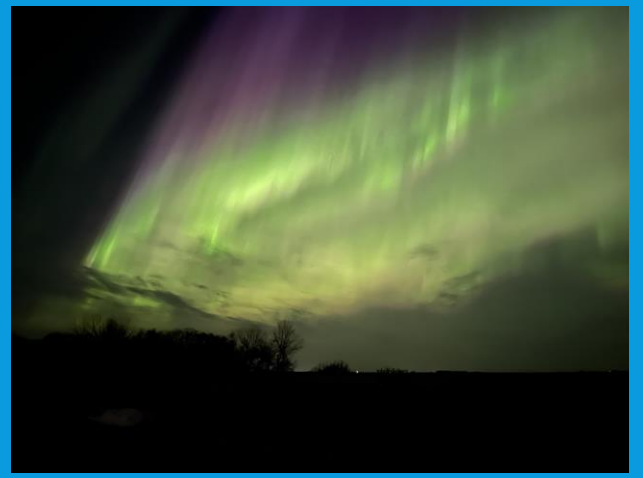
Goal #3: Sleep for 7.5+ hours per night, 5 nights a week

-> Sleep is important for health, and I don't have a consistent schedule

Lessons Learned

I've learned that I thrive in an environment where I can pick a project I am passionate about and become an expert on it. I've also learned that Rust is extremely satisfying to program, and so much nicer to debug than any other language I've used. Lastly, I've learned that I trust my gut to make the right decisions for me.

Aurora chasing instead of sleeping



Snippet of my Rust program

```rust
use backscatter_rs::fitting::fitacf3::fitacf_v3::{fit_rawacf_record, Fitacf3Error};
use backscatter_rs::utils::hdw::HdwInfo;
use chrono::NaiveDateTime;
use clap::Parser;
use dmap::formats::{to_file, DmapRecord, FitacfRecord, RawacfRecord};
use rayon::prelude::*;
use std::fs::File;
use std::path::PathBuf;

pub type BinResult<T, E = Box<dyn std::error::Error + Send + Sync>> = Result<T, E>;

fn main() {
    if let Err(e) = bin_main() {
        eprintln!("error: {e}");
        if let Some(e) = e.source() {
            eprintln!("error: {e}")
        }
        std::process::exit(1);
    }
}

#[derive(Parser, Debug)]
#[command(author, version, about, long_about = None)]
struct Args {
    /// Rawacf file to fit
    #[arg(short, long)]
    infile: PathBuf,

    /// Output fitacf file path
    #[arg(short, long)]
    outfile: PathBuf,
}

fn bin_main() -> BinResult<()> {
    let args = Args::parse();

    let rawacf = File::open(path: args.infile)?;
    let rawacf_records = RawacfRecord::read_records(dmap_data: rawacf)?;

    let rec = &rawacf_records[0];
    let file_datetime = NaiveDateTime::parse_from_str(
        s: format!(
            "{:4}{:02}{:02} {:02}:{:02}:{:02}",
            rec.year, rec.month, rec.day, rec.hour, rec.minute, rec.second
        )
        .as_str(),
        fmt: "%Y%m%d %H:%M:%S",
    )
    .map_err(|_| Fitacf3Error::Message("Unable to interpret record timestamp".to_string()))?;
    let hdw = HdwInfo::new(rec.station_id, file_datetime)
        .map_err(|e| Fitacf3Error::Message(e.details))?;

    // Fit the records!
    let fitacf_records: Vec<FitacfRecord> = rawacf_records
        .par_iter()
        .map(|rec| fit_rawacf_record(record: rec, &hdw).expect(msg: "Unable to fit record"))
        .collect();

    // Write to file
    to_file(path: args.outfile, dmap_records: &fitacf_records)?;
    Ok(())
}
```