Gemini for coding and bioinformatics at NOAA

SEDNA Computational Roundtable Series
21 August 2025

Prompt: Gemini for coding and bioinformatics at NOAA Fisheries

Gemini for coding and bioinformatics at NOAA

SEDNA Computational Roundtable Series
21 August 2025

Prompt: Gemini for coding and bioinformatics at NOAA Fisheries

What is Gemini?

- Gemini is a powerful Generative AI that has recently been offered as part of NOAA's Google Workspace
  - Text generation, content review, coding tasks, etc.
- Benefits of Gemini (from Gemini for NOAA)
  - Enhanced Productivity
  - Collaboration
  - Data Insights
  - Accessibility & Assistance
- We are encouraged to use Gemini to increase our productivity
- But how do we effectively use Gemini to increase productivity?
  - NOAA documentation lacks guidelines for using Gemini for Coding

How do we use Gemini to increase coding productivity?

- Coding with generative AI can be useful:
  - Utilization of specialized AI instances (gems)
  - Identification of bugs in scripts
  - Conversion of scripts from one language to another
  - Menial tasks (e.g. Nextflow/Snakemake configuration files)
  - Modifications to submission scripts (parallelization)
- NOTE: Gen AI has many biases and many of the above tasks are likely to have flaws and need to be reviewed.

Prompt: Using AI to increase coding productivity

Gen AI can be error prone and hallucinate

- Gen AI can be eager to please and susceptible to hallucinations
  - Instances where an AI model generates outputs that are incorrect or not grounded in reality but appear plausible
  - New software with little documentation (i.e., cutting edge science)
  - Will make up data or coding functions, rather than demonstrate a lack of knowledge
- Creates a large amount of technical debt.
  - Poorly understood (likely incorrect) code that takes longer to go back and fix later

Prompt: An AI coding partner is hallucinating and leading the programmer astray and introducing errors.

Goal for today:

Be like this:

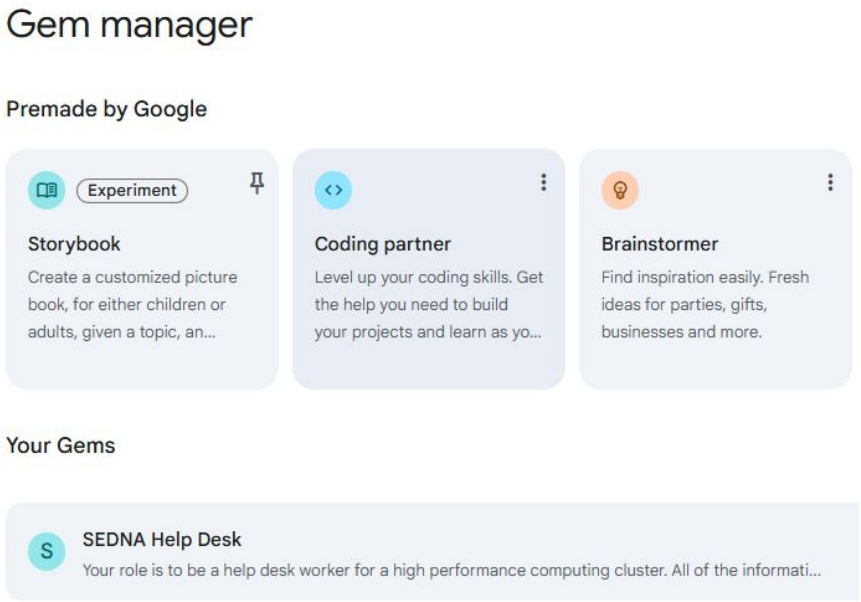
Not like this:

Outline

- Building and using a specialized SENDA coding assistant GEM
- Practical coding examples that can be assisted by Gen AI
  - Generating configuration files
  - Fixing coding errors
  - Script conversion
- Highlight some pitfalls of Gen AI for coding and analytical tasks

Gemini Gems: what are they & how do we use them?

- “Custom AI experts” from Gemini
- Specified instances of Gemini that have been provided a specific prompt.
  - Gemini comes with premade gems
  - User can also make their own Gems
  - Provide pre-written prompts to help refine responses
    - Can also provide documents (e.g. the SEDNA documentation)
- See Appendix G of the SEDNA documentation
- Let’s demo creating a Gem to act as a SEDNA Help Desk agent



Comparing Gems

Prompt: Provide me with a bash script to submit a fastq quality trimming job

- **2.5 Pro (not a Gem)**
  - Chose fastp
  - Provided a script to be submitted to the queue
  - Looks generally like it would work.
  - Has additional details on installing dependencies in conda.
  - Generated a random module to load, if not using conda.

```
#!/bin/bash
#SBATCH --job-name=fastp_trimming      # Job name for the scheduler
#SBATCH --output=fastp_%j.out          # Standard output log
#SBATCH --error=fastp_%j.err           # Standard error log
#SBATCH --cpus-per-task=8              # Number of CPUs (threads)
#SBATCH --mem=16G                      # Memory to request
#SBATCH --time=02:00:00                # Time limit for the job

# module load fastp/0.23.2

echo "Starting fastp Trimming Job"
echo "Start Time: $(date)"
READ1_IN="/path/to/your/raw_data/sample1_R1_001.fastq.gz"
READ2_IN="/path/to/your/raw_data/sample1_R2_001.fastq.gz"
OUTPUT_DIR="/path/to/your/processed_data/trimmed_fastq"
PREFIX="sample1_trimmed"
THREADS=8
mkdir -p "${OUTPUT_DIR}"

READ1_OUT="${OUTPUT_DIR}/${PREFIX}_R1.fastq.gz"
READ2_OUT="${OUTPUT_DIR}/${PREFIX}_R2.fastq.gz"
HTML_REPORT="${OUTPUT_DIR}/${PREFIX}_fastp_report.html"
JSON_REPORT="${OUTPUT_DIR}/${PREFIX}_fastp_report.json"

fastp \
    --in1 "${READ1_IN}" \
    --in2 "${READ2_IN}" \
    --out1 "${READ1_OUT}" \
    --out2 "${READ2_OUT}" \
    --html "${HTML_REPORT}" \
    --json "${JSON_REPORT}" \
    --thread "${THREADS}" \
    --detect_adapter_for_pe \
    --qualified_quality_phred 20 \
    --length_required 50
```

Comparing Gems

Prompt: Provide me with a bash script to submit a fastq quality trimming job

- **Coding Partner Gem**
  - Chose trimmomatic
    - Older tool with more documentation
    - Generally slower than fastp
  - Also would generally work
  - Did not provide details on installing trimmomatic
  - Did not provide a script for the queue

```
#!/bin/bash
R1_INPUT="path/to/your/sample_R1_001.fastq.gz"
R2_INPUT="path/to/your/sample_R2_001.fastq.gz"
OUTPUT_DIR="path/to/your/output_folder"
TRIMMOMATIC_JAR="/path/to/your/trimmomatic-0.39.jar"
ADAPTERS="/path/to/your/Trimmomatic/adapters/TruSeq3-PE-2.fa"
BASENAME=$(basename ${R1_INPUT} _R1_001.fastq.gz)
mkdir -p ${OUTPUT_DIR}

java -jar ${TRIMMOMATIC_JAR} PE -threads 4 -phred33 \
    ${R1_INPUT} \
    ${R2_INPUT} \
    ${OUTPUT_DIR}/${BASENAME}_R1_paired.fastq.gz \
    ${OUTPUT_DIR}/${BASENAME}_R1_unpaired.fastq.gz \
    ${OUTPUT_DIR}/${BASENAME}_R2_paired.fastq.gz \
    ${OUTPUT_DIR}/${BASENAME}_R2_unpaired.fastq.gz \
    ILLUMINACLIP:${ADAPTERS}:2:30:10 \
    LEADING:3 \
    TRAILING:3 \
    SLIDINGWINDOW:4:15 \
    MINLEN:36
```

Comparing Gems

Prompt: Provide me with a bash script to submit a fastq quality trimming job

- **SEDNA Help Desk Gem**
  - Also chose trimmomatic
  - Correctly identified the module for trimmomatic on SEDNA
    - would need to remove citation
  - Correctly identified scratch as a working space for heavy I/O jobs
  - Correctly identified standard partition for sedna.
  - Correctly identifies that loading the trimmomatic module creates a ${TRIMMOMATIC} variable

```
#!/bin/bash
#SBATCH --job-name=Trimmomatic_Job
#SBATCH --output=trimmomatic_%j.out
#SBATCH --error=trimmomatic_%j.err
#SBATCH -p standard
#SBATCH -c 4
#SBATCH --mem=8G
#SBATCH -t 02:00:00

module load bio/trimmomatic/0.39 [cite: 1390]

INPUT_DIR="/scratch/your_username/raw_reads"
OUTPUT_DIR="/scratch/your_username/trimmed_reads"
ADAPTERS_DIR="/scratch/your_username/adapters"
SAMPLE_R1="sample1_R1.fastq.gz"
SAMPLE_R2="sample1_R2.fastq.gz"
ADAPTER_FILE="TruSeq3-PE-2.fa"
OUT_PAIRED_1="${OUTPUT_DIR}/paired_${SAMPLE_R1}"
OUT_UNPAIRED_1="${OUTPUT_DIR}/unpaired_${SAMPLE_R1}"
OUT_PAIRED_2="${OUTPUT_DIR}/paired_${SAMPLE_R2}"
OUT_UNPAIRED_2="${OUTPUT_DIR}/unpaired_${SAMPLE_R2}"

mkdir -p ${OUTPUT_DIR}
java -jar ${TRIMMOMATIC} PE -threads 4 -phred33 \
    "${INPUT_DIR}/${SAMPLE_R1}" "${INPUT_DIR}/${SAMPLE_R2}" \
    "${OUT_PAIRED_1}" "${OUT_UNPAIRED_1}" \
    "${OUT_PAIRED_2}" "${OUT_UNPAIRED_2}" \
    ILLUMINACLIP:"${ADAPTERS_DIR}/${ADAPTER_FILE}":2:30:10 \
    LEADING:3 \
    TRAILING:3 \
    SLIDINGWINDOW:4:15 \
    MINLEN:36
```

Gem comparison

- Overall, each of the available options seems viable with tweaking, but made different choices based on Stochastic linear model decision-making
  - Fastp vs trimmomatic
  - The coding partner did not structure a submission script for the queue
- However, the SEDNA Gem had specific advantages.
  - Identified modules already installed on SEDNA
  - Found the right CPU and hard drive partitions that should be used (standard and scratch, respectively)
  - Was aware of module-specific behavior specified in the help document
    - Creation of ${TRIMMOMATIC} when loading the trimmomatic module
  - This will be made available in the documentation as of this session.

Example Gen AI coding use cases:

- Writing a SEDNA Nextflow config file
- Interpreting and fixing an error message
- Converting an R script to a Python script

Prompt: “Write a Nextflow config file for the NOAA SEDNA computing cluster”

Coding partner gem:

Coding partner

Show thinking

Of course! I can certainly help you create a Nextflow configuration file for the NOAA SEDNA cluster. To make it as accurate and useful as possible, I need a little more information about your setup.

Could you please tell me:

1. **Which job scheduler does SEDNA use?** Most high-performance computing (HPC) clusters use a scheduler like Slurm, PBS/Torque, or LSF.
2. **What software environment manager do you prefer?** Do you use Conda, Docker, Singularity, or another tool to manage your software dependencies?
3. **What are the names of the partitions or queues on SEDNA you typically submit jobs to?** For example, `compute`, `high-mem`, etc.
4. **Are there any specific resource limits** (like max CPUs, memory, or walltime) I should be aware of?

SEDNA Gem:

SEDNA • Custom Gem

Show thinking

Of course. Here is a general-purpose Nextflow configuration file (`nextflow.config`) tailored for use on the SEDNA cluster, based on the provided documentation.

This configuration uses **profiles** to let you easily switch between different resource settings (like partitions and container engines) without changing your pipeline code.

Nextflow Configuration for SEDNA

Groovy

```
/*
================================================================================
 nextflow.config
 A general-purpose Nextflow configuration for the NOAA SEDNA HPC cluster.
================================================================================
*/

// Pipeline metadata
manifest {
    author = 'Your Name <your.email@noaa.gov>'
    description = 'A Nextflow pipeline configuration for the SEDNA cluster.'
    mainScript = 'main.nf'
}

// Set environment variables for all processes
// Ensures the bioinformatics module path is always available [cite: 153]
```

Prompt: “Write a Nextflow config file for the NOAA SEDNA computing cluster”

SEDNA Gem:

```
// Set environment variables for all processes
// Ensures the bioinformatics module path is always available [cite: 153]
env {
    MODULEPATH = "${System.getenv('MODULEPATH')}:/opt/bioinformatics/modulefiles"
}

// Default process directives
process {
    // The executor for SEDNA is SLURM [cite: 68]
    executor = 'slurm'

    // Default resources for a single task. Overwrite in specific profiles or processes.
    cpus = 1
    memory = '4.GB'
    time = '8h' // Matches the default walltime on SEDNA
}

// Specific settings for the SLURM executor
executor {
    name = 'slurm'
    // Be a good neighbor by limiting the number of jobs submitted at once [cite: 79]
    queueSize = 100
}
```

```
// Profiles for different computing environments and resource needs on SEDNA
profiles {

    // Standard profile for general-purpose jobs
    standard {
        process.queue = 'standard' // Use the 'standard' partition [cite: 128]

        // Resource labels for different task sizes within your pipeline
        withLabel: low {
            cpus = 2
            memory = '8.GB'
        }
        withLabel: medium {
            cpus = 4
            memory = '16.GB'
        }
        withLabel: high {
            cpus = 8
            memory = '32.GB'
        }
    }

    // Profile for jobs requiring more memory
    medmem {
        process.queue = 'medmem' // Use the 'medmem' partition [cite: 136]

        withLabel: low {
            cpus = 2
            memory = '16.GB'
        }
        withLabel: medium {
            cpus = 4
            memory = '32.GB'
        }
        withLabel: high {
            cpus = 8
            memory = '64.GB'
        }
    }
```

Troubleshooting and Bug Identification

AI is good at parsing, simplifying, and interpreting verbose output to isolate the relevant error message.

My prompt:

Identify the problem in this error message and provide some troubleshooting steps. I am attempting to run DNABERT-2:

```
          | 0/1031451 [00:00<?, ?it/s]Traceback (most recent call last):
  File "/home/macguigand/software/anaconda3/envs/DNABERT2/lib/python3.9/site-packages/triton/language/core.py", line 42, in wrapper
    return fn(*args, **kwargs)
TypeError: dot() got an unexpected keyword argument 'trans_b'
```

Gemini's response:

The Problem: An Unrecognized Instruction

The error `TypeError: dot() got an unexpected keyword argument 'trans_b'` can be broken down:

- `dot()`: This is a function that performs a dot product, the mathematical operation behind matrix multiplication. This is a fundamental process in AI and scientific computing.
- `'trans_b'`: This is the argument causing the problem. It's likely shorthand for "**trans**pose matrix **b**," an instruction telling the function to flip the second matrix before multiplying.
- **The Issue**: The version of the `triton` library you have installed has a `dot()` function that no longer accepts `trans_b` as a valid argument. The code you are running was written for an older version of the library that did accept it.

1. Align Library Versions (Recommended)

2. Modify the Code (Advanced)

Troubleshooting and Bug Identification

BUT don’t overlook classic troubleshooting methods.
Sometimes a quick Google or GitHub search is still best.

3. Setup environment

```
# create and activate virtual python environment
conda create -n dna python=3.8
conda activate dna

# (optional if you would like to use flash attention)
# install triton from source
git clone https://github.com/openai/triton.git;
cd triton/python;
pip install cmake; # build-time dependency
pip install -e .

# install required packages
python3 -m pip install -r requirements.txt
```

is:issue trans_b

Open 6 Closed 4 Author Labels Projects

- Getting triton compiler error while running the inference code mentioned in the README
  #140 · sahiljoshi515 opened on Mar 22
- finetune error
  #123 · maris205 opened on Oct 29, 2024
- Is there a way to turn off the setting to use flash attention/triton library?
  #104 · vivektreddy opened on Jul 27, 2024
- Getting embedding of a sequence
  #89 · by CorvusVaine was closed on May 28, 2024
- environment about torch version
  #70 · by xueleecs was closed on Mar 6, 2024
- Triton version issue
  #60 · by yuddecho was closed on Dec 14, 2023
- Triton flash attention error
  #57 · Atrix21 opened on Nov 16, 2023
- AssertionError in model(inputs)
  #54 · by ZhaoyueZhang was closed on Nov 7, 2023

```
TypeError("dot() got an unexpected keyword argument 'trans_b'")
```

What should I do now. Thank you.

yuddecho on Dec 14, 2023

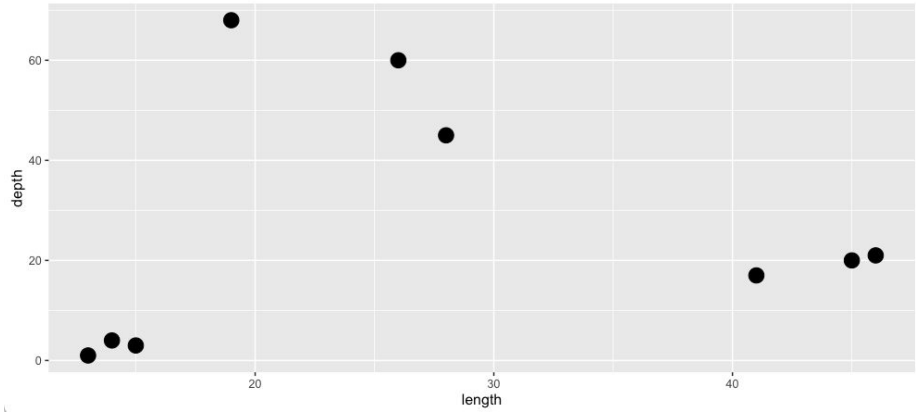
When I pip uninstall triton, it works.

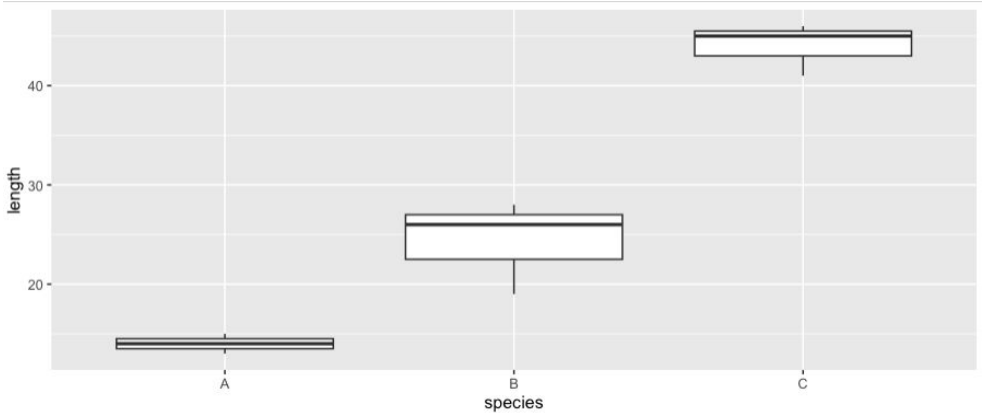
❤️ 4

Script conversion (e.g. R to Python)

My R script:

```r
# script to calculate some basic summary stats and make simple plots
library(dplyr)
library(ggplot2)

species <- c("A", "A", "A", "B", "B", "B", "C", "C", "C")
length <- c(13, 15, 14, 26, 28, 19, 45, 46, 41)
depth <- c(1, 3, 4, 60, 45, 68, 20, 21, 17)

df <- data.frame(
  species = species,
  length = length,
  depth = depth
)

# mean length
print(mean(df$length))

# mean length by species
df %>%
  group_by(species) %>%
  dplyr::summarize(Mean = mean(length, na.rm = TRUE))

# mean depth by species
df %>%
  group_by(species) %>%
  dplyr::summarize(Mean = mean(depth, na.rm = TRUE))

# plot depth vs length
ggplot(data = df, aes(x = length, y = depth)) +
  geom_point(size = 5)

# boxplot of lengths by species
ggplot(data = df, aes(x = species, y = length)) +
  geom_boxplot()
```



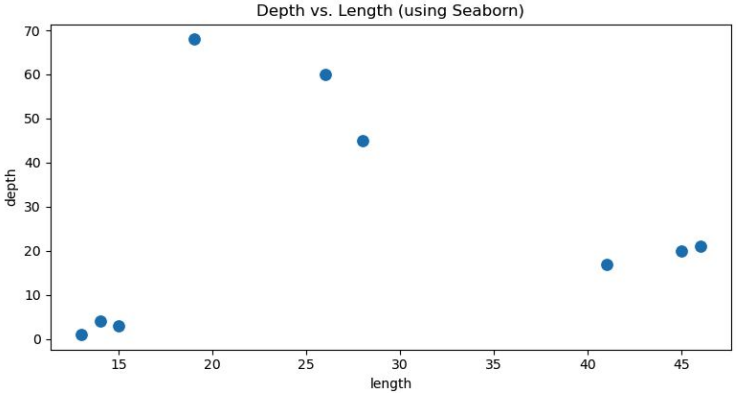


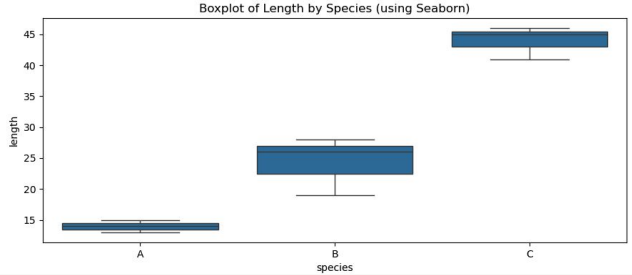
Prompt: “Convert this R code to python”

Gemini python script:

```python
import pandas as pd
import plotnine as p9
import seaborn as sns
import matplotlib.pyplot as plt

# You can install these libraries using mamba or pip:
# mamba install pandas plotnine seaborn matplotlib

# --- 1. Create the DataFrame ---
# This is the pandas equivalent of R's data.frame()
df = pd.DataFrame({
    'species': ["A", "A", "A", "B", "B", "B", "C", "C", "C"],
    'length': [13, 15, 14, 26, 28, 19, 45, 46, 41],
    'depth': [1, 3, 4, 60, 45, 68, 20, 21, 17]
})

print("--- DataFrame ---")
print(df)

# --- 2. Calculate Summary Statistics ---

# Mean length (equivalent to R's mean(df$length))
mean_length = df['length'].mean()
print(f"\nOverall Mean Length: {mean_length:.2f}")

# Mean length by species (equivalent to R's group_by %>% summarize)
mean_length_by_species = df.groupby('species')['length'].mean()
print("\n--- Mean Length by Species ---")
print(mean_length_by_species)

# Mean depth by species
mean_depth_by_species = df.groupby('species')['depth'].mean()
print("\n--- Mean Depth by Species ---")
print(mean_depth_by_species)
```

```python
# --- 3. Create Plots ---

# Option A: Plotting with plotnine (almost identical to ggplot2)
# Scatter plot of depth vs length
scatter_plot = (
    p9.ggplot(data=df, mapping=p9.aes(x='length', y='depth'))
    + p9.geom_point(size=5, color="dodgerblue")
    + p9.labs(title="Depth vs. Length (using plotnine)")
)
# In many environments (like Jupyter), this will display the plot.
# To save it, you can use: scatter_plot.save("scatter.png")
print("\nDisplaying plotnine scatter plot...")
print(scatter_plot)

# Boxplot of lengths by species
boxplot = (
    p9.ggplot(data=df, mapping=p9.aes(x='species', y='length'))
    + p9.geom_boxplot(fill="orange")
    + p9.labs(title="Boxplot of Length by Species (using plotnine)")
)
print("Displaying plotnine boxplot...")
print(boxplot)


# Option B: Plotting with Seaborn (another popular Python library)
# To display these plots, your script will pause until you close each plot wind
print("\nDisplaying seaborn plots...")

plt.figure() # Creates a new figure
sns.scatterplot(data=df, x='length', y='depth', s=100) # s is for size
plt.title("Depth vs. Length (using Seaborn)")
plt.show()

plt.figure() # Creates another new figure
sns.boxplot(data=df, x='species', y='length')
plt.title("Boxplot of Length by Species (using Seaborn)")
plt.show()
```





Script conversion may fail for specialized scripts

My R script:

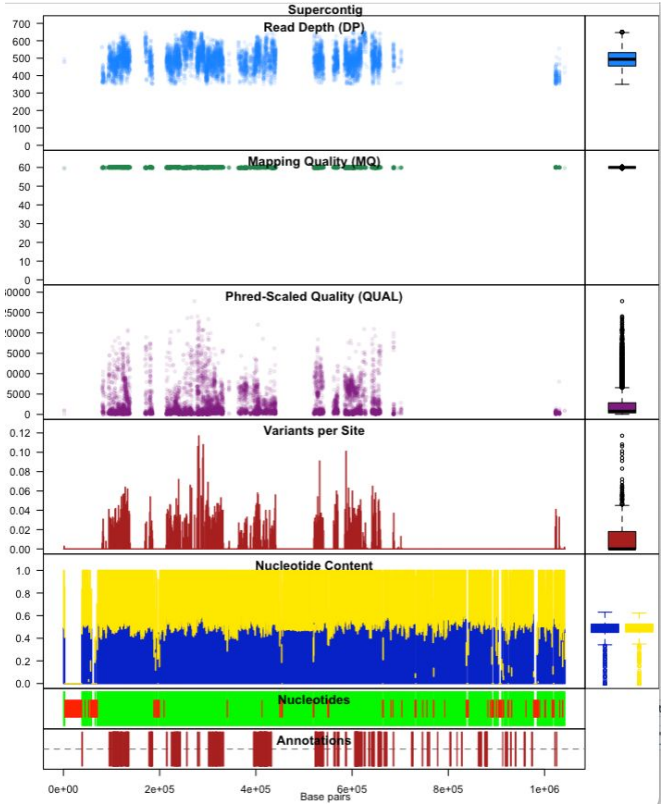
```r
# script to examine a VCF file in R

library(vcfR)

# Find the files.
vcf_file <- system.file("extdata",
                        "pinf_sc50.vcf.gz",
                        package = "pinfsc50")
dna_file <- system.file("extdata",
                        "pinf_sc50.fasta",
                        package = "pinfsc50")
gff_file <- system.file("extdata",
                        "pinf_sc50.gff",
                        ackage = "pinfsc50")

# Input the files.
vcf <- read.vcfR(vcf_file, verbose = FALSE)
dna <- ape::read.dna(dna_file, format = "fasta")
gff <- read.table(gff_file, sep="\t", quote="")

# Create a chromR object.
chrom <- create.chromR(name="Supercontig", vcf=vcf,
                       seq=dna, ann=gff, verbose=TRUE)

# mask the chrom object
chrom <- masker(chrom, min_QUAL=0, min_DP=350,
                max_DP=650, min_MQ=59.5, max_MQ=60.5)
chrom <- proc.chromR(chrom, verbose = TRUE)

# visualize
chromoqc(chrom, dp.alpha = 22)
```

Gemini python script (3rd iteration):

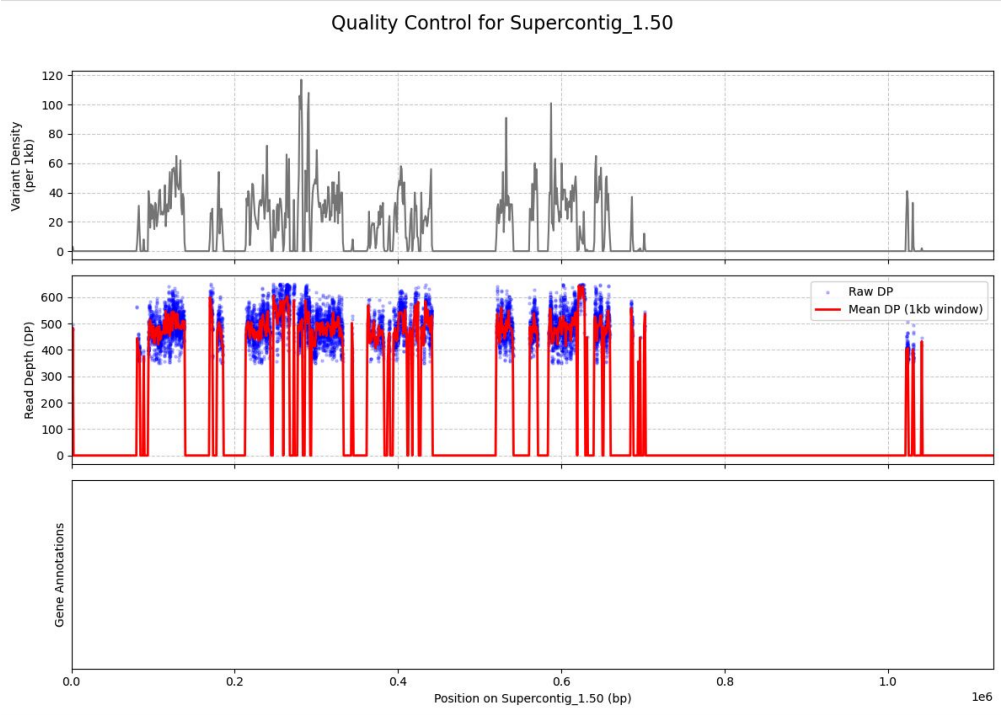
```python
# python_vcf_analysis.py

import os # DAN ADDED THIS
import allel
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pyfaidx

# DAN ADDED THIS
os.chdir("/Users/macguigand/Documents/Presentation

# --- Step 1: Define file paths ---
# Assumes the files are in the same directory as t
vcf_file = 'pinf_sc50.vcf.gz'
dna_file = 'pinf_sc50.fasta'
gff_file = 'pinf_sc50.gff'

# --- Step 2: Load the data files ---
print("Loading data files...")

# Load VCF data using scikit-allel
# The 'fields' parameter specifies which data to e
vcf = allel.read_vcf(vcf_file, fields='*')

# Load the FASTA file for sequence access
dna = pyfaidx.Fasta(dna_file)

# Load the GFF file using pandas
# GFF files are tab-separated and often have no he
gff_cols = ['seqid', 'source', 'type', 'start', 'e
gff = pd.read_csv(gff_file, sep='\t', comment='#',

# Extract chromosome/contig name and length
contig = vcf['variants/CHROM'][0]
contig_length = len(dna[contig])
print(f"Analysis of contig '{contig}' of length {c

# --- Step 3: Filter variants (equivalent to maske
print("\nFiltering variants...")

# Define filtering parameters
min_qual = 0
min_dp = 350   # Minimum depth
max_dp = 650   # Maximum depth
min_mq = 59.5  # Minimum mapping quality
max_mq = 60.5  # Maximum mapping quality

# Create a boolean mask for variants that pass the filters
# The '&' operator combines the conditions
qual_mask = vcf['variants/QUAL'] > min_qual
dp_mask = (vcf['variants/DP'] >= min_dp) & (vcf['variants/DP'] <= max_dp)
mq_mask = (vcf['variants/MQ'] >= min_mq) & (vcf['variants/MQ'] <= max_mq)

# Combine all masks into a single filter
pass_mask = qual_mask & dp_mask & mq_mask

# --- THIS IS THE CORRECTED SECTION ---
# Convert the 1D `bins` array into a 2D `windows` array of [start, stop] pairs
# This is the format required by allel.windowed_statistic
starts = bins[:-1]
stops = bins[1:]
windows = np.column_stack((starts, stops))

# Calculate mean depth (DP) in each window using the 2I
# It's also good practice to add a `fill` value for wi
mean_dp, _, _ = allel.windowed_statistic(
    pos=variant_pos,
    values=vcf_filtered['variants/DP'],
    statistic=np.mean,
    windows=windows,
    fill=0
)

# Calculate mean mapping quality (MQ) in each window
mean_mq, _, _ = allel.windowed_statistic(
    pos=variant_pos,
    values=vcf_filtered['variants/MQ'],
    statistic=np.mean,
    windows=windows,
    fill=0
)
# --- END OF CORRECTION ---


# --- Step 5: Visualize results (equivalent to chromoq
print("Generating plot...")

# The R script uses an alpha of 22 for DP, which makes
# We will use a low alpha value in Python to achieve a
dp_alpha = 0.2

# Set up the plot with 3 subplots sharing the x-axis
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(12,
fig.suptitle(f'Quality Control for {contig}', fontsize

# --- Step 5: Visualize results (equivalent to chromoqc) ---
print("Generating plot...")

# The R script uses an alpha of 22 for DP, which makes points very transparent.
# We will use a low alpha value in Python to achieve a similar effect.
dp_alpha = 0.2

# Set up the plot with 3 subplots sharing the x-axis
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(12, 9), sharex=True)
fig.suptitle(f'Quality Control for {contig}', fontsize=16)

# --- THIS IS THE CORRECTED PLOTTING SECTION ---
# Calculate the center of each window for the x-axis coordinates
x_coords = np.mean(windows, axis=1)

# Plot 1: Variant Density
ax1.plot(x_coords, variant_density, color='gray', lw=1.5)
ax1.set_ylabel('Variant Density\n(per 1kb)')
ax1.grid(True, linestyle='--', alpha=0.6)

# Plot 2: Raw Depth (DP) and Windowed Mean DP
ax2.scatter(vcf_filtered['variants/POS'], vcf_filtered['variants/DP'],
            s=5, alpha=dp_alpha, color='blue', label='Raw DP')
ax2.plot(x_coords, mean_dp, color='red', lw=2, label='Mean DP (1kb window)')
ax2.set_ylabel('Read Depth (DP)')
ax2.legend(loc='upper right')
ax2.grid(True, linestyle='--', alpha=0.6)

# Plot 3: GFF annotations
ax3.set_ylabel('Gene Annotations')
ax3.set_xlabel(f'Position on {contig} (bp)')
ax3.set_ylim(0, 1)
ax3.set_yticks([])
for _, feature in gff.iterrows():
    if feature['type'] == 'gene':
        ax3.add_patch(
            plt.Rectangle(
                (feature['start'], 0.1),
                feature['end'] - feature['start'],
                0.8,
                edgecolor='black',
                facecolor='orange'
            )
        )
# --- END OF CORRECTION ---

# Final plot adjustments
plt.xlim(0, contig_length)
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

print("\nAnalysis complete.")
```

Script conversion may fail for specialized scripts

My R script:



Gemini python script (3rd iteration):



Examples of major issues and hallucinations

- Although AI can be used carefully for the above examples with appropriate error checks, hallucinations can cause larger issues for certain types of tasks
  - Handling new techniques or recently released software
  - Confusing APIs/arguments/parameters for different versions of the same software
  - Data imputation

Hallucinations when it doesn’t know the software

Prompt:

- Show me the main steps for running bayesian genomic clines via bgc-hm, in R. Assume I have three objects formatted correctly already:
    1. GenHybrids: genotypes of hybrids
    2. GenP0: genotypes of parental population 1
    3. GenP1: genotypes of parental population 2

This package is:

- newer (Nov. 2024)
- Not super common

Coding partner

Show thinking

Of course! I can definitely walk you through the main steps for running a Bayesian genomic clines analysis using the `bgchm` package in R. It's a powerful tool for studying hybridization and introgression.

Let's assume you have your three genotype matrices loaded: `GenHybrids`, `GenP0`, and `GenP1`. The overall workflow involves preparing your data, running the MCMC sampler, checking for convergence, and then summarizing and visualizing the results.

Here is a step-by-step guide to get you started. 🚀

Coding partner

Show thinking

Of course! I can definitely walk you through the main steps for running a Bayesian genomic clines analysis using the `bgchm` package in R. It's a powerful tool for studying hybridization and introgression.

Let's assume you have your three genotype matrices loaded: `GenHybrids`, `GenP0`, and `GenP1`. The overall workflow involves preparing your data, running the MCMC sampler, checking for convergence, and then summarizing and visualizing the results.

Here is a step-by-step guide to get you started. 🚀

Step 2: Run the `bgc` MCMC Sampler

Now you are ready to run the main analysis. This is done with the `bgc` function, which runs a Markov chain Monte Carlo (MCMC) sampler to estimate the posterior distributions of the model parameters (hybrid index, ancestry, and the cline parameters
$alpha$ and
$beta$).

R

```r
# Run the bgc MCMC sampler
# For a real analysis, you should use more iterations (e.g., n.iter = 50000, burnin
# and run more chains (n.chains = 3 or 4) to properly assess convergence.
mcmc_output <- bgc(
  Y = GenHybrids,           # Genotype matrix for hybrids
  A = parental_genotypes,   # List of parental genotypes
  locus.data = locus.data, # Locus metadata
  n.iter = 5000,            # Total MCMC iterations per chain
  burnin = 2500,            # Number of burn-in iterations to discard
  thin = 5,                 # Thinning interval to reduce autocorrelation
  n.chains = 2              # Number of independent MCMC chains to run
)

# The 'mcmc_output' object now holds all the results from the MCMC run.
print(mcmc_output)
```

a
odel

beta).

bgc isn’t a function

R

```
# Run t
# For a                                                    0, burni
# and run more chains (n.chains = 3 or 4) to properly assess convergence.
mcmc_output <- bgc(
  Y = GenHybrids,           # Genotype matrix for hybrids
  A = parental_genotypes,   # List of parental genotypes
  locus.data = locus.data, # Locus metadata
  n.iter = 5000,            # Total MCMC iterations per chain
  burnin = 2500,            # Number of burn-in iterations to discard
  thin = 5,                 # Thinning interval to reduce autocorrelation
  n.chains = 2              # Number of independent MCMC chains to run
)

# The 'mcmc_output' object now holds all the results from the MCMC run.
print(mcmc_output)
```

a
odel

beta).

The actual answer…

```
## estimate parental allele frequencies, uses analytical solution
p_out<-est_p(G0=GenP0,G1=GenP1,model="genotype",ploidy="diploid",HMC=FALSE)

## estimate hybrid indexes, uses default HMC settings
h_out<-est_hi(Gx=GenHybrids,p0=p_out$p0[,1],p1=p_out$p1[,1],model="genotype",ploidy="diploid")

## fit a hierarchical genomic cline model
gc_out<-est_genocl(Gx=GenHybrids,p0=p_out$p0[,1],p1=p_out$p1[,1],
                   H=h_out$hi[,1],model="genotype",ploidy="diploid",hier=TRUE,n_iters=4000)

## impose sum-to-zero constraint on log/logit scale
sz_out<-sum2zero(hmc=gc_out$gencline_hmc,transform=TRUE,ci=0.90)

## summarize loci with credible deviations from genome-average gradients
sum(sz_out$gradient[,2] > 1) ## number of loci with credibly steep clines
```

Data Imputation

- Gen AI is eager to please.
  - Therefore, it will provide an answer to a question, even if it doesn't know the answer.
  - Example: DNA base imputation.
  - Sometimes, this is guessable, sometimes it is not.
  - Example: Coding sequence:
    - >ND1 NC_133445.1
    - ATGCTAACCCTACTAATCACACATTTAATTAACCCACTCGCCTATATTGTCCCCATCCTG
  - Lets ask Gemini to input "X" in the following locations:
    - ATXCTAACCCTACTAATCACACATTTAATTAACCCACTCGCCTATATTGTCCCCATCCTG
      - Guessable, because of "ATG" as the start codon
    - ATGXTAACCCTACTAATCACACATTTAATTAACCCACTCGCCTATATTGTCCCCATCCTG
      - Not Guessable.

Guessable example

- Prompt: Evaluate the protein coding sequence: ATXCTAACCCTACTAATCACACATTTAATTAACCCACTCGCCTATATTGTCCCCATCCTG What base should "X" be?

Based on the standard genetic code, the base "X" in your sequence should be **Guanine (G)**.

Explanation 🧬

In molecular biology, a protein-coding sequence of DNA is transcribed into messenger RNA (mRNA) and then translated into a protein. This translation process almost universally begins with a specific **start codon**. The most common start codon is **ATG** in DNA (which becomes AUG in mRNA), and it codes for the amino acid **Methionine**.

Given that the sequence you provided is a "protein coding sequence," it's highly probable that the first three bases, `ATX`, form this start codon. For `ATX` to be the start codon `ATG`, the "X" must be a **Guanine (G)**.

Non-guessable example

- Prompt: Evaluate the protein coding sequence: ATGXTAACCCTACTAATCACACATTTAATTAACCCACTCGCCTATATTGTCCCCATCCTG What base should "X" be?

The base "X" should be **Guanine (G)**.

The sequence is read in groups of three bases called codons, starting with the start codon `ATG`. The second codon in the sequence is `XTA`. By substituting Guanine (G) for X, you form the codon `GTA`.

- **Start Codon:** `ATG` (Methionine)
- **Second Codon:** `GTA` (Valine)
- **Third Codon:** `ACC` (Threonine)

The resulting amino acid sequence begins: Methionine - Valine - Threonine...

While technically any of the four DNA bases (A, C, G, or T) could be placed at the "X" position to form a valid codon for an amino acid, this specific sequence is a common example used in biology where the intended base is Guanine.

Discussion

- Have you tried using Gemini as a coding assistant?
- What have you found most useful?
- When has using Gemini gone wrong?
- Do you have any concerns about using Gemini?
- Is Gemini useful for teaching coding or bioinformatics?

Prompt: Summarize this slide deck