**🌿 NEON Sensitivity Analysis Project Summary**

**🔍 Project Goals**

**Primary Goal**

* Assess the sensitivity of NEON plant cover data to detect a **20% year-to-year change in species abundance** at each site.

**Secondary Goal**

* Evaluate how decreasing the number of sampled plots per site affects detection ability — across each **species × year pair × site** combination.

**📦 Project Context**

* **Model**: GJAM (gjam::gjam)
* **Response**: Species percent cover (continuous, wide format)
* **Predictors**: year (factor), nlcdClass (factor)
* **Model Options**: typeNames = "CA", REDUCT = FALSE
* **Posterior Prediction**: Uses a **custom function** (manual\_posterior\_predict()), not gjamPredict()
* **Pipeline Framework**: Built with {targets} and {future} for reproducibility and parallel scaling

**🧩 Pipeline Functions (Execution Order)**

1. **load\_neon\_data()**
   * Loads and formats NEON vegetation data
   * Filters for samplingImpractical == "OK"
   * Outputs tidy percent cover data per site
2. **fit\_gjam\_model\_test(site\_data)**
   * Fits full-site GJAM model once
   * Predictors: year, nlcdClass
   * Patches internal model elements (e.g., fit$inputs, betaBeta, sigmaSave)
   * Returns: fit, site, xdata, ydata
3. **manual\_posterior\_predict(fit, xnew)**
   * Draws posterior predictions using stored chains (betaBeta, sigmaSave)
   * Returns array: [n\_draws, 2 conditions (baseline/changed), n\_species]
4. **simulate\_change(fit, change\_year = c("2015", "2016"), plot\_index = 1)**
   * (TO BE MODIFIED — see below)
   * Currently uses a single plot’s covariates to simulate baseline vs change
   * Returns posterior array of predicted cover across species
5. **loop\_simulate\_changes(fit, plot\_index = 1)**
   * (TO BE MODIFIED — see below)
   * Loops over all valid year-pairs
   * Calls simulate\_change() for each year pair
   * Returns named list of posterior arrays (e.g., "2015\_2016")
6. **calculate\_detection\_probability(posterior\_array, site\_id, sample\_size, year\_pair)**
   * For each species:
     + Computes posterior difference across years
     + Calculates detection probability (detect\_prob), mean\_diff, CI bounds
   * Returns tidy tibble of results
7. **run\_sample\_size\_sensitivity()**
   * (TO BE MODIFIED — see below)
   * For each sample size:
     + Samples plots
     + Calls full GJAM fit + loop + detection probability functions
   * Records failure/success with fit\_status
   * Returns: tidy tibble with detection outcomes across species × years × sample sizes × replicates

**✅ Immediate Next Steps: Modify Three Functions**

Your insight was exactly right: we **must not refit** the model repeatedly when doing sensitivity to sample size. Instead, we simulate changes in the predictor matrix while using the **same fixed posterior** from the full-site GJAM.

Here’s what to change next:

**1. 🔧 simulate\_change()**

**Current:**

* Uses one plot row (plot\_index) to simulate change via modified year

**Update to:**

* Accept xdata\_subset (subset of xdata with N plots)
* For both change\_year[1] and change\_year[2], clone xdata\_subset and update year
* Bind into xnew with both time conditions
* Run manual\_posterior\_predict(fit, xnew)
* Return posterior array for full set of sampled plots

**2. 🔧 loop\_simulate\_changes()**

**Current:**

* Calls simulate\_change() with same plot\_index for each year pair

**Update to:**

* Accept plot\_subset (vector of row indices or a tibble subset of xdata)
* For each valid year\_pair, call simulate\_change(fit, change\_year, plot\_subset)
* Return named list of posterior arrays (e.g., "2015\_2016" → array)

**3. 🔧 run\_sample\_size\_sensitivity()**

**Current:**

* Repeats GJAM fit for each plot sample (violates fixed posterior requirement)

**Update to:**

* Accept a pre-fitted fit object (from full site model)
* Sample a subset of rows from fit$xdata
* Pass to loop\_simulate\_changes() as the plot\_subset
* Continue with calculate\_detection\_probability() as-is
* Keep fit\_status, sample\_size, and replicate tracking

**✅ Targets-Compatible Design**

Each function:

* Accepts **explicit arguments** (xdata\_subset, plot\_subset, etc.)
* **Returns clean tibbles or arrays**
* Can be embedded into {targets} workflows without file I/O
* Enables use of tar\_map() or branching across sites, sample sizes, or year pairs

**🚀 Ready to Scale**

Once the above three function updates are implemented and tested locally, you’ll be ready to:

* Parallelize across sites using {future}
* Scale to full NEON coverage
* Run this on GCP or another HPC cluster
* Summarize detection thresholds per species × year × site × sample size

Let me know if you'd like:

* A code draft for any of the three functions above
* Help writing tar\_map() across sample sizes
* A GCP deployment checklist

You’re close — this structure finally preserves posterior coherence while allowing realistic simulations of sampling effort 👌.

Ask ChatGPT