

```

import json
import os

import pandas as pd
import streamlit as st

# Ruta relativa donde guardaste los archivos descargados
LOCAL_DATA_PATH = os.path.join("data", "iri")

def fetch_iri_data(filename):
    """
    Carga los datos del IRI desde archivos locales (JSON) previamente descargados.
    Evita errores de conexión y SSL.
    """
    # Construir ruta completa
    file_path = os.path.join(LOCAL_DATA_PATH, filename)

    try:
        # Verificar existencia
        if not os.path.exists(file_path):
            # Intento de fallback: buscar en la raíz del proyecto
            if os.path.exists(filename):
                file_path = filename
            else:
                st.error(
                    f"⚠ Archivo no encontrado: `{file_path}`. Por favor sube el archivo a `data/iri/`."
                )
                return None

        # Leer archivo
        with open(file_path, "r", encoding="utf-8") as f:
            data = json.load(f)
            return data

    except json.JSONDecodeError:
        st.error(f"❌ El archivo `{filename}` no es un JSON válido.")
        return None
    except Exception as e:
        st.error(f"❌ Error leyendo archivo `{filename}`: {e}")
        return None

# --- FUNCIONES DE PROCESAMIENTO (SE MANTIENEN IGUAL) ---

```

```

def process_iri_plume(data_json):
    """Procesa el JSON de plumas (modelos spaghetti)"""
    if not data_json or "years" not in data_json:
        return None

    try:
        # Busca el último año disponible en el archivo
        last_year_entry = data_json["years"][-1]
        year = last_year_entry["year"]

        if not last_year_entry["months"]:
            return None
        # Busca el último mes disponible
        last_month_entry = last_year_entry["months"][-1]
        month_idx = last_month_entry["month"]

        models_data = []
        if "models" in last_month_entry:
            for m in last_month_entry["models"]:
                # Limpieza de valores centinela (-999, etc)
                clean_values = [
                    x if x is not None and x > -100 else None for x in m["data"]
                ]
                models_data.append(
                    {"name": m["model"], "type": m["type"], "values": clean_values}
                )

        seasons_base = [
            "DJF",
            "JFM",
            "FMA",
            "MAM",
            "AMJ",
            "MJJ",
            "JJA",
            "JAS",
            "ASO",
            "SON",
            "OND",
            "NDJ",
        ]
        start_idx = (month_idx + 1) % 12
        forecast_seasons = [seasons_base[(start_idx + i) % 12] for i in range(9)]

        return {

```

```

        "year": year,
        "month_idx": month_idx,
        "seasons": forecast_seasons,
        "models": models_data,
    }
except Exception as e:
    st.error(f"Error procesando estructura Plume: {e}")
    return None

def process_iri_probabilities(data_json):
    """Procesa el JSON de probabilidades (barras)"""
    if not data_json or "years" not in data_json:
        return None

    try:
        last_year_entry = data_json["years"][-1]
        if not last_year_entry["months"]:
            return None

        last_month_entry = last_year_entry["months"][-1]

        probs = []
        if "probabilities" in last_month_entry:
            for p in last_month_entry["probabilities"]:
                probs.append(
                    {
                        "Trimestre": p["season"],
                        "La Niña": p["lanina"],
                        "Neutral": p["neutral"],
                        "El Niño": p["elnino"],
                    }
                )

        return pd.DataFrame(probs)
    except Exception as e:
        st.error(f"Error procesando estructura Probabilities: {e}")
        return None

```