Trabajo Final de Máster

Máster en Business Intelligence y Big Data

Estudio sobre la

Salud Infantil en África

Child Health Study in Africa

Pablo Bencomo Mesa

Alejandro Lorenzo Dávila

Santa Cruz de Tenerife, 15 de octubre de 2020

Agradecimientos

A nuestras familias y queridos por acompañarnos a los largo de este año aunque no entendieran exactamente que estábamos haciendo.

A Don Gustavo Marrero por proponernos este proyecto.

A los profesores del máster por confiar en un proyecto diferente.

Resumen

El objetivo de este trabajo es realizar un estudio acerca de la salud infantil en África, que busque unir no solo grandes fuentes de datos, sino técnicas vanguardistas generalmente no usadas hasta la fecha.

El estudio busca encontrar relaciones que puedan explicar la variación de los niveles de salud infantil en el continente. Este estudio pretende servir de base a futuras investigaciones, así como de ayuda a organismos que su labor se fundamente en este ámbito.

En este trabajo se analizan los efectos de los choques climáticos sobre la salud infantil medidos a través de las variables altura para la edad y peso para la edad en niños menores de cinco años, en los países africanos. Con este objetivo, fusionamos los datos de la encuesta demográfica y de salud con datos meteorológicos del Laboratorio de Investigación del Sistema Terrestre utilizando coordenadas GPS. Nuestra muestra está formada por unos 650000 mil niños distribuidos en 31 países. Construimos shocks climáticos a partir de un histórico de datos de (temperatura y precipitaciones). Utilizamos la econometría espacial para analizar el impacto de estos choques climáticos en la salud infantil y medir el potencial efecto de recuperación y el papel de la malaria como canal de transmisión.

Palabras clave: Salud Infantil, África, Shocks climáticos, Shocks de malaria, NDVI, Big Data.

Abstract

The objective of this work is to carry out a study about child health in Africa, which seeks to unite not only large sources of data, but also avant-garde techniques generally not used to date.

The study seeks to find relationships that can explain the variation in children's health levels on the continent. This study aims to serve as a basis for future research, as well as to help organizations that their work is based on this area.

This paper analyzes the effects of climatic shocks on children's health, measured as height for age and weight for age less than 5 years, in African countries. To this end, we merged the demographic and health survey data with meteorological data from the Earth System Research Laboratory using GPS coordinates. Our sample is made up of about 650,000 thousand children distributed in 31 countries. We construct climatic shocks from a historical data of (temperature and rainfall). We use spatial econometrics to analyze the impact of these climatic shocks on children's health and measure the potential recovery effect and the role of malaria as a transmission channel.

Keywords: Children's Health, Africa, Climate shocks, Malaria shocks, NDVI, Big Data.

Índice general

Índice de tablas y figuras

Capítulo 1 Introducción

Este capítulo presenta el origen del proyecto, algunos de los estudios realizados en el pasado con temática similar o relacionados, así como el origen de los datos y la hipótesis inicial general del estudio.

Motivación

Este proyecto surge de la unión de un problema moral, como resulta ser el conocimiento de las necesidades que sufren muchos países y la necesidad de aportar al mismo, con la posibilidad de acceder a datos de primera calidad acerca de los niveles de salud en África.

Un tiempo atrás la Universidad de La Laguna adquirió un gran volumen de datos sobre la salud infantil en África. En 2019 estos datos de la mano de Don Gustavo Marrero Díaz fueron cedidos a Pablo Bencomo y por transitividad a Alejandro Lorenzo para ser usados en un proyecto.

Al valorar que esta temática tiene un carácter social importante y que puede generar un impacto en estos países, decidimos aceptar la propuesta del estudio.

Antecedentes

Mientras que la malaria afecta a miles de personas en todo el continente africano, los niños menores de 5 años y las embarazadas, son los grupos más vulnerables, dado sus bajos niveles de inmunidad contra la misma. Alrededor de 500 millones de personas están afectadas de malaria en todo el mundo; se estima que 75% de los enfermos son niños menores de 5 años y mujeres; cada día mueren 3 000 niños y más de un millón de personas cada año. La mayoría de estas muertes ocurre en niños menores de 5 años y mujeres embarazadas en África Subsahariana. Nuestro objetivo es buscar variables que expliquen estos datos de una forma directa como pueden ser los shocks climáticos y de otra forma que subyace en los niveles de inmunidad de la población como es la productividad del suelo que a la postre es el único recurso de alimentación y salud del que disponen.

Actualmente muchos estudios han abordado diversos frentes para entender un poco mejor las causas, pero no se habían juntado anteriormente tantos frentes en un solo estudio.

Literatura previa

La nueva literatura sobre economía climática (Dell et al., 2014) analiza el impacto del clima y choques climáticos sobre el capital humano, la salud de adultos y niños (Maccini & amp; Yang, 2009), y sobre la mortalidad infantil (Kudamatsu et al., 2012) a nivel individual.

Los canales de transmisión más importantes están relacionados con el impacto a través de enfermedades epidémicas, principalmente la malaria (Tanser et al., 2003).

Además, el impacto del clima en la salud humana es mayor en países con menos desarrollo. El perfil de estos países suele tener un predominio de áreas rurales y actividades más relacionadas con eventos climáticos, como la calidad y cantidad de la cosecha, y la transmisión de enfermedades infecciosas como la malaria. Así, (Barrios et al.,2010) analiza el efecto de lluvias sobre el crecimiento económico africano; (Dell et col., 2012) estudian el impacto de los shocks climáticos en el crecimiento económico en varios países en desarrollo; Henderson y col. (2014) analiza el impacto de los cambios en el clima y la urbanización en el África subsahariana; Jina Hsiang (2014) estudian el impacto de fenómenos meteorológicos extremos como ciclones en el futuro.

Paralelamente, otro grupo de estudios analiza la influencia del clima en la economía, política o urbanización. Por ejemplo, (Henderson et al. 2014) estudió las consecuencias de la

variabilidad climática del África subsahariana sobre los procesos de urbanización y transformación del sector rural. Harari y La Ferrara (2012) encuentran que existe una relación significativa entre las perturbaciones climáticas que producen bajos rendimientos agrícolas y probabilidad de existencia de conflicto civil. Dell y col. (2012) también presentó evidencia de que ciertos conflictos tienen lugar en ambientes cálidos. Además, estos autores también señalan que en los países pobres la temperatura afecta la producción industrial, especialmente a través de la productividad laboral. (Barrios et al., 2010) asocia tendencias en la lluvia con un desempeño económico deficiente en África subsahariana: menos lluvia significa menos crecimiento, actuando a través del sector agrícola o hidroeléctrico.

Una literatura más reciente analiza el efecto del clima en la salud. La principal hipótesis aquí es que ciertas condiciones adversas durante el embarazo y los primeros años de la vida puede causar problemas de salud y reducir el capital humano y la productividad en el futuro (Maccini y Yang, 2009; Chatterji y otros, 2014). Así, por ejemplo, circunstancias como la mala nutrición durante el embarazo puede afectar al desarrollo cognitivo de los niños. Hay dos explicaciones que ponderan con más fuerza: primero, la hipótesis de Barker, que afirma que el nacimiento con bajo peso (como medida antropométrica de la salud del niño) puede conducir a enfermedades crónicas en la edad adulta, y esto afectaría la productividad futura; en segundo lugar, los episodios de malnutrición durante el embarazo afectan al desarrollo cognitivo del recién nacido. La nueva literatura destaca el papel del clima en la salud o la mortalidad infantil. Por ejemplo, Maccini y Yang (2009) se centran en los efectos de la lluvia durante el primer año de la vida sobre los resultados futuros individuales en las zonas rurales de Indonesia. Un análisis similar es realizado por Dreesen (2014) para las zonas rurales de Bangladesh.

Sobre los Anexos

Este trabajo cuenta con una componente de programación elevada. La mayor parte del trabajo de procesamiento y preprocesamiento del que hablaremos posteriormente se encuentra reflejado en estos Anexos adjuntos al final del documento.

Origen de los datos

Encuesta DHS

La encuesta de DHS con la que vamos a trabajar recopila datos primarios utilizando varios tipos de cuestionarios. Se utiliza un cuestionario familiar para recopilar información sobre las características de la unidad de vivienda del hogar y datos relacionados con la altura y peso para mujeres y niños en el hogar. También se usa para identificar a los miembros del hogar que son elegible para una entrevista individual.

En la mayoría de las encuestas de DHS, los individuos elegibles incluyen mujeres en edad reproductiva (15-49) y hombres (15-59). En algunos países solo se entrevista a mujeres. Los cuestionarios individuales incluyen información sobre fertilidad, planificación familiar y salud materno-infantil.

Se trata de un archivo de datos de recodificación con un formato estandarizado, con la misma estructura en todos los países. Esta estandarización está destinada a facilitar las comparaciones entre encuestas.

El DHS también recopila datos utilizando otros tipos de encuestas y cuestionarios. En ellos se incluyen encuestas de educación, proveedores de servicios de salud, comunidades, gastos de salud en el hogar y otros. Estos datos están disponibles, pero no hay definiciones de recodificación para ellos.

Este documento contiene dos partes. La primera parte es una discusión general del archivo de recodificación, incluido la justificación de la recodificación; descripción de la estructura física en la que está disponible el archivo de recodificación; codificación de estándares utilizados en el archivo de datos; ubicación de la información de identificación; uso de códigos para fechas. La segunda parte proporciona una descripción de cada variable en el archivo de datos, dando información adicional que nos ayuda a entender la variable en cuestión.

Justificación de la recodificación

Los datos individuales se transforman en un conjunto de datos de recodificación estandarizado por varias razones:

- Primero, se imputan las fechas de varios eventos clave, ya que gran parte del análisis de los datos se basa en estos eventos y sus fechas son a menudo incompletas o faltantes.

- Segundo, las variables recogidas en el cuestionario original están en una forma conveniente para su recolección, pero no siempre para análisis. A menudo se hace la misma pregunta en varios lugares, pero a diferentes encuestados. En el archivo de recodificación, estas variables se combinan y se crean una de fácil uso para el análisis.

- Tercero, ciertos índices, particularmente los índices antropométricos de los datos de altura y peso se calculan a partir de los datos incluidos en el archivo de recodificación.

- Finalmente, y lo más importante, los datos en el archivo de recodificación están en un formato estandarizado permitiendo una fácil comparación de datos entre países.

El enfoque del DHS para crear archivos de datos de recodificación individuales estandarizados para cada país es parte de la política de DHS para hacer que los datos sean accesibles, proporcionando al analista los datos en la forma más conveniente para análisis. Este enfoque, si bien proporciona un fácil acceso a los datos, no está exento de dificultades. DHS sugiere que los analistas se familiaricen con los cuestionarios utilizados en las encuestas.

Los cuestionarios utilizados en un país, aunque contienen esencialmente la misma información, pueden ser diferentes (en muchos sentidos) de los utilizados en otro país. Al crear los archivos de datos de recodificación individuales estandarizados estas diferencias requieren una consideración especial y la estandarización total obviamente no es posible.

La recodificación del archivo de datos está estructurada en dos partes, secciones estándar y secciones específicas del país. Las secciones estándar contienen las mismas variables en las mismas posiciones para todos los países. Las secciones específicas del país contienen todas las variables específicas y, por lo tanto, no están estandarizadas entre países.

Estructura del archivo de datos

El archivo de datos de recodificación está disponible en dos estructuras diferentes; la estructura a utilizar depende del hardware y las necesidades de software del analista:

Cada registro del archivo de datos representa un caso (encuestado), con todas las variables ubicadas uno tras otro en el mismo registro. Se colocan las secciones repetidas del archivo de recodificación uno tras otro en el registro. Cada variable en una sección repetitiva se coloca inmediatamente después de la variable anterior de la misma ocurrencia, de modo que todas las variables de ocurrencia 1 preceder a todas las variables para la ocurrencia 2 de una sección. Por ejemplo, en el historial de nacimientos BIDX, BORD, B0, B1, etc. para la primera aparición aparece seguido de la segunda aparición de BIDX, BORD, B0, B1, etc.

El tamaño total del archivo de datos es en promedio de aproximadamente 40 millones de bytes, dependiendo del tamaño de la muestra, y los archivos más grandes tienen más de 380 M bytes de tamaño. El archivo está diseñado para usuarios que utilizan paquetes estadísticos que solo admiten estructuras de datos que contiene un número fijo de registros por caso como SPSS, SAS o STATA

Estándares de codificación

Se utilizan códigos especiales en todo el archivo de datos para ciertas respuestas. Se presenta el esquema general de codificación a continuación.

Los códigos dados se aplican a variables de 4 dígitos, 3 dígitos, 2 dígitos y 1 dígito, respectivamente. Si hay otras respuestas especiales a preguntas, estas están codificadas en orden decreciente a partir de estos códigos especiales, es decir, 9996, 996, 96, 6; 9995, 995, 95, 5; etc.

9999, 999, 99, 9 Esta pregunta debería haber sido respondida por el encuestado, pero el cuestionario no contenía información para esta variable (datos faltantes).

9998, 998, 98, 8 El encuestado respondió "No sé" a esta pregunta.

9997, 997, 97, 7 La respuesta a esta pregunta fue inconsistente con otras respuestas en el cuestionario y se pensó que esta respuesta probablemente fue un error. La respuesta se cambió a este código para evitar más problemas debido a la inconsistencia de información.

La variable EN BLANCO tampoco es aplicable para este encuestado porque la pregunta no fue preguntada en un país en particular o porque no se hizo la pregunta de esto encuestado debido al patrón de flujo u omisión del cuestionario.

En ciertas preguntas, se utiliza un esquema de codificación de dos dígitos en el que el primer dígito, que representa la codificación principal categoría, es estándar, pero el segundo dígito es específico del país. Esto se aplica a preguntas como las relacionadas a la fuente de agua, los servicios sanitarios y la fuente de anticoncepción. Por ejemplo, para la fuente de anticoncepción las principales categorías son:

1 Sector público

2 Sector médico privado

3 Otro sector privado

4 Otro

El esquema de codificación para V326 (última fuente de anticoncepción para usuarios actuales de métodos modernos) podría usar códigos como:

11 Hospital del gobierno

12 Centro de salud del gobierno

...

21 Hospital privado o clínica

22 Médico privado

...

31 Tienda

Identificación de encuesta

Para cada encuesta hay un código alfabético de identificación de país de dos caracteres, Los códigos de país son los siguientes:

Afghanistan	AF	Haiti	HT	Niger	NI
Angola	AO	Honduras	HN	Nigeria	NG
Armenia	AM	India	IA	Pakistan	PK
Azerbaijan	AZ	Indonesia	ID	Peru	PE
Bangladesh	BD	Jordan	JO	Rwanda	RW
Benin	BJ	Kenya	KE	Senegal	SN
Burundi	BU	Kyrgyz Republic	KY	South Africa	ZA
Cambodia	KH	Lao People's Dem. Rep.	LA	Swaziland	SZ
Colombia	CO	Lesotho	LS	Tajikistan	TJ
Congo (Brazzaville)	CG	Liberia	LB	Tanzania	TZ
Congo Dem. Rep.	CD	Madagascar	MD	Timor-Leste	TP
Cote d'Ivoire	CI	Malawi	MW	Uganda	UG
Egypt	EG	Mali	ML	Yemen	YE
Ethiopia	ET	Mauritania	MR	Zambia	ZM
Gabon	GA	Mozambique	MZ	Zimbabwe	ZW
Gambia	GM	Namibia	NM		
Guinea	GN	Nepal	NP		

Figura 1 Códigos de Países

Cuestionario (Modelo)

Se utilizaron dos cuestionarios básicos durante las encuestas de DHS, el cuestionario modelo "A" para países de alta prevalencia de anticonceptivos y cuestionario modelo "B" para países con baja prevalencia de anticonceptivos. Los dos los cuestionarios contienen básicamente la misma información, aunque el cuestionario modelo "A" contiene un calendario detallado de eventos en los cinco años anteriores a la entrevista, mientras que el cuestionario modelo "B" contiene una serie más simple de preguntas.

En la sección de descripción variable que sigue, la columna etiquetada "Modelo" indica en qué cuestionario se hace la pregunta. Una "A" indica que la variable se refiere a una pregunta formulada solo en países que utilizaron un cuestionario modelo "A" y una "B" indican que la variable se relaciona con una pregunta formulada solo en países que utilizó el cuestionario modelo "B". Si la columna está en blanco, entonces se hace la pregunta en ambos, cuestionarios Modelo "A" y cuestionarios modelo "B". Si la columna contiene una "X", la pregunta no se incluye en ninguno de los dos modelos de cuestionario.

Si la columna contiene "MM", las preguntas provienen del módulo de mortalidad materna. Si la columna contiene "FG", las preguntas provienen del módulo de corte genital femenino.

Secciones y ocurrencias

El archivo de datos se divide en varias secciones lógicas. Estas secciones se traducen directamente en registros para las estructuras de datos jerárquicos. Las secciones lógicas están diseñadas para mapear las secciones de los cuestionarios modelo, aunque algunas secciones del cuestionario modelo se dividen en más de una sección en el archivo de datos de recodificación. Algunas de estas secciones son secciones repetitivas o de ocurrencia múltiple, mientras que otras son secciones de ocurrencia única. Las secciones individuales contienen variables simples de respuesta única. Se utilizan varias secciones para representar conjuntos de preguntas que

se repiten para varios eventos. El historial de nacimientos es un ejemplo de una sección múltiple, donde se hacen preguntas relacionadas con los niños para cada niño, y cada niño tiene una entrada en el historial de nacimientos. Cada entrada en la sección múltiple se conoce como una ocurrencia de la sección. En los archivos de datos jerárquicos, cada aparición de la sección ocupa un registro separado. Se utilizan varias secciones para conjuntos de preguntas donde el número de ocurrencias puede variar. Por el contrario, los grupos de preguntas para las cuales hay un número fijo de ocurrencias se mantienen en un grupo. Un grupo es similar a una sección múltiple, pero se almacena en un único registro para archivos jerárquicos. Además, las variables individuales también se pueden incluir en una sección que contiene un grupo. En el archivo de recodificación, la tabla de anticonceptivos (REC31) se almacena como un grupo que contiene 20 entradas, una para cada método anticonceptivo.

Descripciones de secciones y variables

La siguiente descripción de la sección ofrece un resumen de las secciones del archivo de recodificación y los tipos de información que contienen La descripción se basa en los archivos jerárquicos. La descripción de la sección da el nombre de la sección, el código de sección utilizado para identificar la sección en el archivo de datos, la longitud del registro para esa sección, la clase de sección (S para single y M para múltiple), el número mínimo y máximo de ocurrencias de la sección en cada caso, y la etiqueta de la sección.

La descripción de la sección es seguida por descripciones de variables. Las descripciones variables proporcionan información adicional, información de fondo relacionada con cada variable.

Sección y Descripción de Variable – Hogar

Level Label / Record Label	Level Name / Record Name	Type / Value	Req	Max	Rec Len
HOUSEHOLD	HOUSEHOLD				
Household's basic data	RECH0	H00	Yes	1	121
Household Schedule	RECH1	H01	No	90	59
Household Characteristics	RECH2	H02	No	1	146
Survey specific Household variables	RECH3	H03	No	1	18
Survey specific Household Schedule variables	RECH4	H04	No	90	20
Women Height/Weight/Hemoglobin	RECH5	H05	No	20	117
Children Height/Weight/Hemoglobin	RECH6	H06	No	20	133
Men Height/Weight/Hemoglobin	RECHMA	HMA	?	20	116
Malaria: by Mosquito Bed Net	RECHML	HML	?	7	43
Malaria: by Household Member	RECHMH	HMH	?	90	50

Figura 2 Variable Hogar DHS

La identificación de casos **HHID** identifica de manera única a cada hogar. En la mayoría de las encuestas esto se construye concatenando el número de grupo o punto de muestra y el número de hogar, pero en algunas encuestas este puede ser el número de cuestionario tomado de la primera página del cuestionario.

Código de país alfabético **HV000** para identificar la encuesta de la que se recopilaron los datos. El código se basa en un código estándar internacional. Esta variable tiene 3 caracteres de longitud, con el tercer carácter que indica el formato del archivo de recodificación utilizado para esta encuesta. Para todas las encuestas en DHS siguiendo este estándar, este código será 6. Por ejemplo: DR6 es el de República Dominicana, HT6 es Haití y KH6 es Camboya.

El número de clúster **HV001** es el número que identifica el punto de muestra utilizado durante el trabajo de campo. Esta variable puede ser un compuesto de varias variables en el cuestionario. Si es así, las variables no estándar se incluyen en RECH3 como variables específicas del país.

HV002 Número de hogar es el número que identifica al hogar dentro del grupo. En algunos casos, esta variable puede ser la combinación del número de vivienda y número de hogar dentro de la vivienda. En estos casos, el número de vivienda se incluye como variable específica del país.

El número de línea del encuestado **HV003** es el número de línea en el horario familiar de la persona respondiendo a las preguntas formuladas en el cuestionario del hogar. Si nadie en el hogar estaba disponible para la entrevista, esta variable está codificada como 00.

La unidad de área definitiva **HV004** es un número asignado a cada punto de muestra para identificar el área final unidades utilizadas en la recopilación de datos. Esta variable suele ser la misma que el número de clúster, pero puede ser una variable numerada secuencialmente para muestras con una estructura más complicada.

HV005 El peso de la muestra es una variable de 8 dígitos con 6 decimales implícitos. Para usar la muestra dividirlo por 1000000 antes de aplicar el factor de ponderación. Todos los pesos de muestra son normalizados de modo que el número ponderado de casos sea idéntico al número no ponderado de hogares cuando usan el conjunto de datos completo sin selección. Esta variable debe usarse para ponderar todas las tabulaciones producidas usando el archivo de datos. Para muestras de auto ponderación esta variable es igual a 1000000.

HV006 Mes de entrevista

HV007 Año de entrevista

HV009 El número total de miembros del hogar indica el número de entradas que se encuentran en **RECH1**.

HV010 El número total de mujeres elegibles indica el número de mujeres encontradas elegibles para la encuesta individual en el horario del hogar. Los criterios de elegibilidad son generalmente: femenino, edades entre 15 y 49 años. En algunos países, los criterios de elegibilidad restringen la encuesta a mujeres casadas.

HV012 El número total de miembros del hogar que suele vivir en el hogar.

HV014 Número de niños residentes en el hogar y menores de 5 años. Los niños visitantes no se incluyen.

HV016 Día de entrevista

HV024 Región de residencia en la que reside el hogar. Los códigos son específicos del país.

HV025 Tipo de lugar de residencia donde reside el hogar, ya sea urbano o rural.

HV026 El tamaño del lugar de residencia es el tipo de lugar en el que reside el hogar. Las áreas urbanas se clasifican en grandes ciudades (capitales y ciudades con más de 1 millón de habitantes), pequeñas ciudades (población de más de 50,000) y pueblos (otras áreas urbanas).

HV035 Número de niños menores de cinco años elegibles para estatura y peso.

HV040 Altitud en metros. Se utiliza para ajustar la medición de anemia por altitud.

HV044 Hogar seleccionado para el módulo de violencia doméstica.

HV101 Relación con el jefe de familia

HV103 Si el miembro es un miembro del hogar de facto, es decir, si el miembro durmió en la casa la noche anterior.

HV106 Nivel más alto de educación al que asistió el miembro del hogar. Esta es una variable estandarizada proporcionando nivel de educación en las siguientes categorías: Sin educación, Primaria, Secundaria, y más alto. Cualquier miembro por debajo del límite de edad inferior para las preguntas de educación está clasificado en la categoría "Sin educación".

HV108 Educación en años individuales.

HV110 Si el miembro del hogar todavía está en la escuela. Todos los miembros de edad igual o mayor al límite superior (generalmente 25 años) para esta pregunta o para quienes no han asistido a la escuela están codificados 0 (no en la escuela).

HV111 Si la madre del miembro del hogar todavía está viva.

HV113 Si el padre del miembro del hogar todavía está vivo.

HV116 Si el miembro del hogar está en la actualidad, anteriormente o nunca casado (o vivió con un compañero). Actualmente casado incluye mujeres casadas y mujeres que viven con una pareja, y anteriormente casada incluye viudas, divorciadas, mujeres separadas y mujeres que han vivido con una pareja, pero ahora no vive con una pareja. En países donde la única pregunta se refiere a si el miembro del hogar alguna vez se casó, las respuestas se codifican como un 2 para siempre casado y 0 para nunca casado.

HV117 Elegibilidad del miembro del hogar para la encuesta individual de mujeres. En la mayoría de las encuestas, tanto de facto como no de facto las mujeres son entrevistadas, sin embargo, las mujeres se incluyen en la recodificación individual solo si eran elegibles para la entrevista y eran miembros de facto del hogar.

HV122 Nivel educativo asistido durante el año escolar actual.

HV129 Estado de asistencia escolar.

0, Nunca asistió. Niños sin educación.

1 Entró en la escuela. Niños que no asistieron a la escuela el año anterior pero son actualmente matriculados.

2 Avanzado. Niños en un nivel actual que es más alto que el año anterior

3 Repitiendo. Niños que están en el mismo nivel que el año anterior o en un nivel menos que el año anterior.

4 Abandono. Niños que estaban en la escuela el año anterior pero que actualmente no asisten al colegio.

5 Abandonó la escuela hace más de 2 años. Niños que actualmente no asisten a la escuela y no asistieron el año anterior.

8 No sé

HA1 Edad de la mujer en años.

HA4 Percentil de altura / edad

Variables de altura / peso / hemoglobina de los niños

HC1 Edad en meses

HC2 Peso en kilogramos

HC3 Altura en centímetros

HC4 Percentil de altura / edad

HC7 Percentil de peso / edad

HC15 Altura: acostado o de pie

HC16 Día de nacimiento del niño

HC17 Fecha medida (día)

HC18 Fecha medida (mes)

HC19 Fecha medida (año)

HC27 Sexo del niño

HC30 Mes de nacimiento del niño

HC31 Año de nacimiento del niño

Fumar tabaco

MV463A Cigarrillos

MV463C Mascar tabaco.

MV463E Específico del país

MV463Z Nada

Transmisión de la tuberculosis

BASE: Alguna vez has oído hablar de la tuberculosis (MV474 = 1).

MV474A Aire al toser o estornudar.

MV474B Compartiendo utensilios.

MV474C Tocando a una persona con TB.

MV474D La tuberculosis se propaga a través de los alimentos.

MV474E Contacto sexual

Picaduras de mosquito MV474F

MV474G Específico del país

MV474X Otro

MV474Z No lo sé.

Matrimonio

MV501 Estado civil actual del encuestado

MV502 Si el encuestado está en la actualidad, anteriormente o nunca casado (o vivió con una pareja). Actualmente casados incluye hombres casados y hombres que viven con una pareja, y anteriormente casados incluye hombres viudos, divorciados, separados y hombres que han vivido con una pareja.

MV504 Si la esposa o la pareja vive con el encuestado.

MV505 El número de esposas que tiene actualmente el encuestado.

Primer matrimonio o unión

MV507 Mes del inicio del primer matrimonio o unión.

MV508 Año de inicio del primer matrimonio o unión.

Datos climáticos

Nuestra fuente de datos incluye un histórico de datos climatológicos de precipitación y temperatura georreferenciados desde principios de siglo pasado. Para su obtención nos remitimos al portal Global Weather Data for SWAT.

Los datos climáticos los obtenemos de la web http://www.climatologylab.org/terraclimate.html que proporciona información climática mensual y balance hídrico para toda la superficie terrestre desde 1958-2019. El conjunto de datos que podemos extraer se puede clasificar en dos categorías, por un lado las **Variables Climáticas Primarias** (temperatura máxima, temperatura mínima, presión de vapor, acumulación de precipitación, radiación de onda corta en la superficie descendente y velocidad del viento), por otro lado tenemos las variables derivadas (la evapotranspiración de referencia (ASCE Penman-Montieth), la escorrentía, la evapotranspiración real, déficit del clima agua, humedad del suelo, equivalente agua de la nieve, índice Palmer de severidad de la sequía y déficit de presión de vapor).

En el anexo adjuntamos el resto del documento que explica desde cómo se obtienen los datos hasta la creación del dataset

Estos datos tendrán una importancia altamente notable en la creación de la mayor parte de los datos generados.

NDVI

VER APÉNDICE 4 PARA MÁS INFORMACIÓN.

Figura 3 Logotipo FAO

A través de la FAO (que se la agencia de las Naciones Unidas que lidera el esfuerzo internacional para poner fin al hambre intentando garantizar los alimentos al mismo tiempo que intenta garantizar el acceso regular a alimentos suficientes y de buena calidad para llevar una vida activa y sana), obtenemos el NDVI que utilizaremos en nuestro proyecto como variable explicativa en una primera fase de estudio del nivel de malaria y en una fase de los niveles de salud infantil.

El uso de sensores remotos en la investigación ha permitido realizar análisis a escalas que anteriormente resultaba muy difícil llevar a cabo. En la actualidad se usan imágenes de satélite para analizar el comportamiento promedio y la tendencia sostenida del NDVI por pixel en cualquier región y cualquier periodo de tiempo.

El NDVI (Índice de Vegetación de Diferencia Normalizada) es un indicador de biomasa sintética activa, en términos más sencillos, es un indicador de la salud de la vegetación. Es uno

de los índices más utilizados en la teledetección desde su introducción en la década de los 70. Concretamente en la Agricultura de precisión es la industria que más hace uso de las ventajas de este indicador por la precisión de los resultados y la frecuencia con la que se obtienen.

El siguiente paso es entender qué es un índice de vegetación y para ello nos vamos a apoyar en un único concepto competencia de la óptica física que es la **reflectancia.** Para hacerlo más sencillo nos apoyaremos en el gráfico que mostramos inmediatamente a continuación en el que se representa los dos posibles escenarios cuando un haz de luz incide sobre una superficie. El primero de esos escenarios es cuando el haz de luz se desvía con el mismo ángulo (respecto a la normal) y por el mismo medio por el que incidió, eso es lo que se conoce como reflexión y el segundo escenario es cuando el haz de luz atraviesa la superficie (pasando a otro medio) con un ángulo diferente al ángulo incidente, este fenómeno es lo que se conoce como refracción.

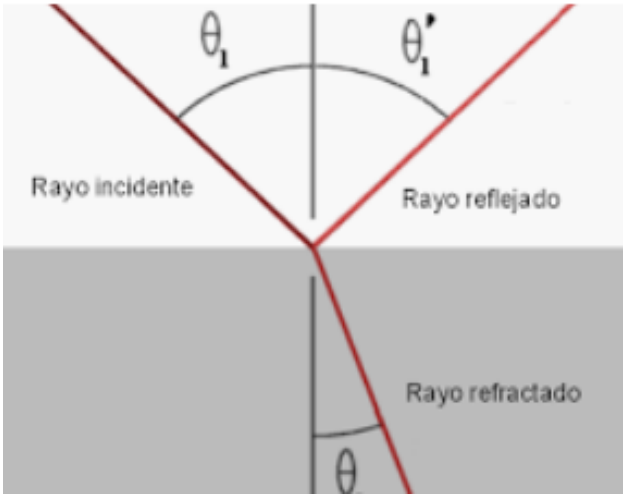


Figura 4 Gráfico sobre la reflectancia

Entendiendo estas dos definiciones estamos en disposición de definir lo qué es la reflectancia. La reflectancia es una proporción de la energía total incidente sobre un material que es reflejada.

Un índice de vegetación, como dijimos hace un momento tienen que ver con el fenómeno físico de la reflectancia (medido a través de sensores) con la particularidad que ahora ese material de superficie incidente son plantas o árboles.

Cómo se calcula

Su cálculo responde a una expresión muy sencilla:

$$NDVI = \frac{NIR - Red}{NIR + Red}$$

Donde NIR es la luz infrarroja cercana y Red es la luz roja visible.

La siguiente pregunta que deberíamos hacernos es *¿de dónde sacamos esta información?* A través de **Landsat 7** que es el séptimo de un grupo de satélites lanzados por Estados Unidos en el programa *LandSat* el 15 de abril de 1999. Su objetivo principal es actualizar la base de datos de imágenes de todo el planeta Tierra sin nubes. A través de **SAGA** (*Sistema para automatizar análisis geo-científicos*) que es un software GIS (sistema de información georreferenciada) de código abierto que se emplea para la edición de datos espaciales. SAGA se encuentra dentro de la barra herramientas de QGIS.

Figura 5 Ubicación herramienta SAGA en QGIS

Cómo funciona

Es sencillo de entender, para ello sólo tenemos que pensar en un haz de luz que choca con una planta/árbol y si fuéramos capaces de descomponer ese haz de luz en un espectro visible, esto es, ver la composición de colores de ese haz, nos quedaríamos sólo con el color rojo. Luego haríamos un cociente muy sencillo entre la cantidad de luz roja visible absorbida y la luz infrarroja cercana que es reflejada.

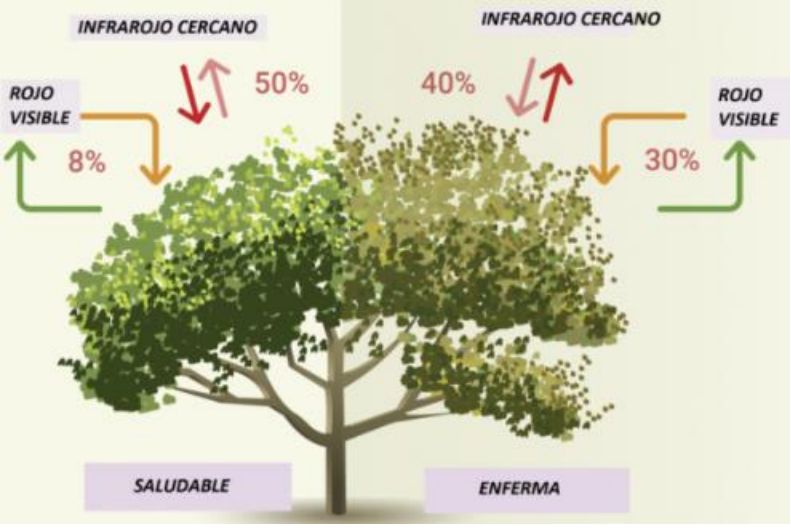


Figura 6 Reflección luz en plantas sanas/enfermas

Interpretación de valores del NDVI

Los valores que toma el NDVI al tratarse de un índice normalizado van desde el -1 hasta el 1. Los valores negativos corresponden a áreas con superficies de agua, estructuras artificiales, rocas, nubes, nieve; el suelo desnudo generalmente cae dentro del rango de 0.1 a 0.2; y las plantas siempre tendrán valores positivos entre 0.2 y 1. El rango de vegetación sana y densa debería estar por encima de 0.5, y la vegetación dispersa probablemente caerá dentro de 0.2 a 0.5. Sin embargo, es solo una regla general y siempre debe tener en cuenta la temporada, el tipo de planta y las peculiaridades del lugar. En la mayoría de los casos, los valores de NDVI entre 0.2 y 0.4 corresponden a áreas con vegetación escasa; la vegetación moderada tiende a variar entre 0.4 y 0.6; cualquier cosa por encima de 0.6 indica la mayor densidad posible de hojas verdes.

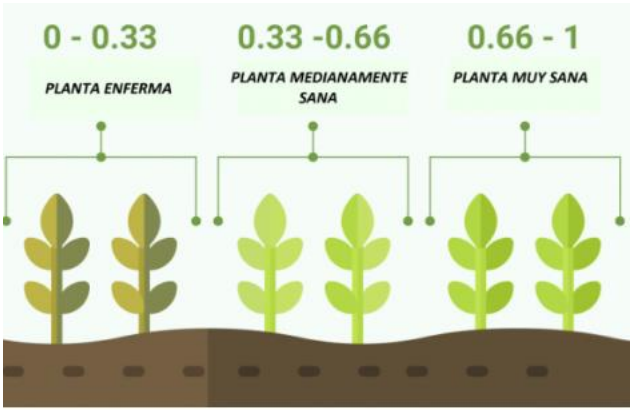


Figura 7 Valores de NDVI

Índice de Evapotranspiración

La **evapotranspiración** se define como la pérdida de humedad de una superficie por evaporación directa junto con la pérdida de agua por transpiración de la vegetación. Se expresa en milímetros por unidad de tiempo.

Transiciones:

- **1º Modelo:** Explicación del NDVI en función a coordenadas geográficas, tiempo y altura.
- **2º Modelo:** NDVI-Malaria Ecology Index
- **3º Modelo:** Evapotranspiración-NDVI
- **4º Modelo:** Evapotranspiración-Malaria Ecology Index
- **5º Modelo:** Evapotranspiración-NDVI-Malaria
- **6º Modelo:** Shocks Climáticos-NDVI-Evapotranspiración-SALUD INFANTIL

Hipótesis

A la hora de plantear este estudio se han formulado una serie de hipótesis a priori que motivan el avance del proyecto y sirven como marcador de hitos:

1. Suponemos que los niveles de calidad y desarrollo del suelo (NDVI) se puede explicar en mayor medida a través de los niveles de evapotranspiración (SPEI).
2. Al disponer de un histórico de datos climáticos creemos que mediante la elaboración de **shocks de temperatura y shocks de precipitaciones** seremos capaces de explicar en mayor medida los niveles de productividad del suelo. (*)
3. Pensamos que los shocks climáticos de precipitación son mejores predictores que sus homólogos de temperatura. En caso de que esto no fuera cierto construiremos shocks climáticos cruzados y verificaremos que son mejores regresores que los shocks tratados individualmente.
4. Creemos que los **shocks de malaria** presentan una correlación muy alta tanto con los niveles de productividad del suelo como con el nivel de evaporación y transpiración del mismo.
5. Los indicadores de desarrollo y calidad del suelo tienen un poder explicativo de los niveles de salud infantil en la misma proporción que lo tienen los shocks de malaria.
6. La probabilidad de que un niño menor de cinco años localizado en una zona de malaria tenga niveles de salud con valores en el intervalo del cuartil más bajo es superior a 70%.
7. Pensamos que una mujer que sufre malaria en el primer mes de embarazo tiene una repercusión más desfavorable en los niveles de salud infantil que sufriéndolo en el segundo o tercer trimestre de embarazo.
8. Más del 75% de los niños que en los dos primeros años padecen malaria registran datos de salud muy por debajo del nivel medio.(**)

(*) Un shock de temperatura lo construimos a partir de datos históricos de temperatura. En primer lugar creamos un Z_{score} a partir de los datos de temperatura que tenemos:

$$z = \frac{x_i - \bar{x}}{DT}$$

Donde x_i = El valor de la temperatura para una latitud-longitud en un mes (por ejemplo Enero) de un año (por ejemplo 1995).

$\bar{x}$ = Es el valor medio de la temperatura en los últimos veinte años correspondientes al mes de Enero.

DT = Es la desviación típica con los mismos datos con los que calculamos la media.

A partir de este momento consideraremos un shock positivo de temperatura aquel que cumpla:

Z > 1

Como el valor de Z puede ser negativo, también tendremos shocks negativos, en ese caso:

Z < 1

Para considerar la existencia de condiciones climáticas extremas crearemos $z(\pm 2)$ y un $z(\pm 3)$ donde el valor del estadístico supera en dos y hasta tres veces el valor de la desviación.

(**) Los shocks de malaria los construiremos a partir de criterios que establece la OMS, concretamente son cuatro:

- La precipitación mensual promedio durante los últimos 3 meses es de al menos 60 mm.
- La precipitación en al menos uno de los últimos 3 meses es de al menos 80 mm.
- Ningún mes en los últimos 12 meses tiene una temperatura promedio inferior a 5ºC
- La temperatura promedio en los últimos 3 meses supera los 19.5ºC + SD (temperatura mensual en los últimos 12 meses).

Capítulo 2 Desarrollo

Este capítulo busca describir de manera general los pasos realizados en el proyecto. Como este proyecto tiene una alta componente de programación, se hará referencia a los apéndices según corresponda.

Conceptos básicos

Aquí se recogen los términos más usados a lo largo del estudio con una definición aproximada y fácil de entender.

Inteligencia Artificial (IA)

Es una subdisciplina del campo de la informática, formada por algoritmos y técnicas que busca crear máquinas o programas que presenten las mismas cualidades que un ser humano o superiores.

Se considera que existen dos tipos:

- Débil: aquellas que realizan una tarea o un conjunto limitado de tareas con éxito. Existen muchos casos de estos.
- Fuerte: aquellas que son capaces de aplicarse a una gran variedad de problemas y de dominios diferentes. Actualmente no existe ningún caso de este grupo.

Dentro del conjunto de técnicas y algoritmos más populares de los últimos años encontramos el Machine Learning y el Deep Learning.

Por su parte el Big Data es transversal a la Inteligencia Artificial por usar las mismas técnicas, servir de fuente de datos y usar los datos generados por la propia IA.

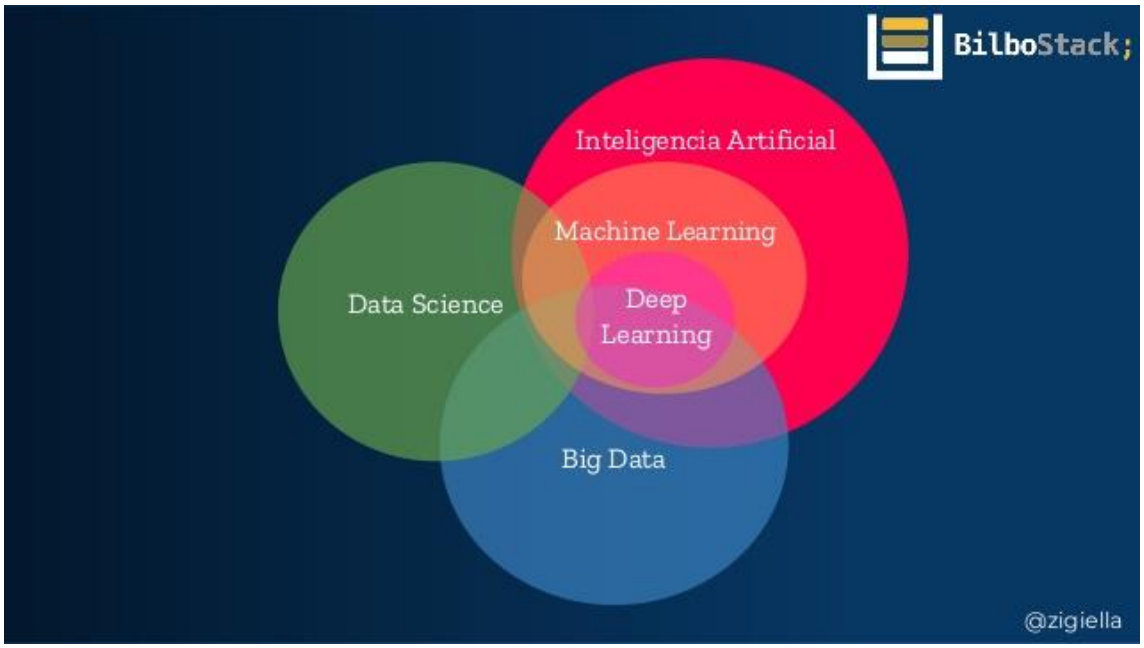


Figura 8 Gráfico de relaciones entre conceptos asociados a la IA

Big Data

Como su nombre indica, se trata de un gran volumen de datos. Dentro de este término se engloban las técnicas asociadas que permiten extraer conocimiento y rendimiento de los datos con los que se cuenta. Estas técnicas combinan tanto métodos tradicionales como

técnicas de Machine Learning y Deep Learning entre otros, que debido al elevado tamaño o complejidad de los datos, resultan en mejores resultados que las técnicas clásicas.

El Big Data se refiere tanto al proceso de almacenaje de datos, como al proceso de análisis de los mismos.

Machine Learning

Ó Aprendizaje Automático, es una disciplina de la Inteligencia Artificial que genera algoritmos que buscan que las máquinas adquieran capacidad de aprendizaje. Entendido este como generar conocimiento a través de un conjunto de experiencias.

Está relacionado íntimamente con el resto de categorías.

Deep Learning

Ó Aprendizaje Profundo, se trata de un subconjunto del Machine Learning basado en el uso de Redes Neuronales. Las cuales simulan la forma en la que los seres humanos aprenden, gracias al concepto de Neurona. Estas están organizadas por capas, e interconectadas bajo diferentes patrones. Mediante el paso de los datos por las diferentes neuronas, el modelo es capaz de detectar patrones y aprender a clasificar o predecir los datos.

Paralelismo

Para poder realizar este conjunto de técnicas y algoritmos de alta complejidad y cómputo con equipos domésticos, se han aplicado diferentes técnicas de paralelismo en el proyecto. El paralelismo permite ejecutar en una máquina varias tareas de manera simultánea con el objetivo (en nuestro caso) de ahorrar tiempo.

Malaria

La malaria es una enfermedad que se produce por un parásito. El parásito se transmite a los seres humanos a través de las picaduras de los mosquitos infectados. Aquellas personas que tienen malaria suelen sentirse muy enfermas, con fiebre alta y escalofríos. Cada año, cerca de 210 millones de personas se infectan con malaria, y aproximadamente 440 000 mueren a causa de la enfermedad. La mayoría de las personas que mueren por la enfermedad son niños pequeños de África.

NDVI

El Índice de Vegetación de Diferencia Normalizada es un indicador simple de biomasa fotosintéticamente activa o, en términos simples, un cálculo de la salud de la vegetación.

Shock

Se trata de un dato calculado que satisface una serie de condiciones. Por ejemplo que en los últimos 3 meses haya llovido más de N cantidad de Litros/metro cuadrado.

Generalmente toma valores de 0 o 1.

Herramientas

El estudio realizado es posible llamarlo un estudio de Big Data, dado el amplio número de registros con los que contamos. Al ser esto así, el número de herramientas es pequeños, puesto que solo necesitaremos herramientas para acceder a los datos, herramientas que actúen como transformadores de los datos, y herramientas para representar los resultados obtenidos:

Excel

Se trata de un Software de la compañía *Microsoft Corp*. Contiene una gran cantidad de herramientas orientadas a la manipulación de hojas de cálculo. Usaremos este software para tener una visión rápida y general de los datos, pero no para manipularlos.

Stata

Se trata de un Software de la compañía *StataCorps*. Es una herramienta estadística que cuenta con un lenguaje de programación propio. Es un software muy potente que nos servirá para acceder a las columnas de interés del *.dta* asociado con los datos DHS.

Python

Se trata de un Lenguaje de programación muy popular en las últimas décadas. Junto con *R* son los dos lenguajes más usados para el tratamiento de datos en el mundo de la *Ciencia de Datos*. Este lenguaje será el pilar del proyecto, puesto que servirá para hacer las transformaciones de los datos, así como generar representaciones simples, hacer *testing*, generar modelos de *Machine Learning*, etc.

QGIS

Se trata de un Software OpenSource de la Open Source Geospatial Foundation (OSGeo). Se trata de un Sistema de Información Geográfica (SIG), que permite la representación de datos con índole geográfica. Será usado para hacer gran parte de las representaciones del proyecto.

Primeros pasos

Pasaremos a describir como a partir de los datos nombrados con anterioridad, se han aplicado transformaciones y filtros a los mismos para limpiar estos y generar nuevos datos que ayuden a los modelos.

También describiremos como se han generado los modelos.

A modo resumen de la arquitectura del proyecto tenemos esta infografía:

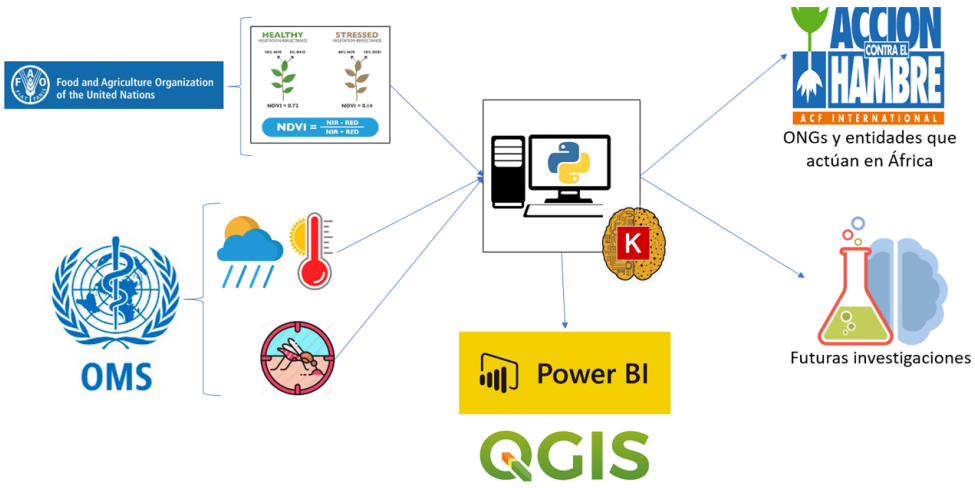


Figura 9 Arquitectura del proyecto

En la figura se observan los diferentes tipos de datos, su procesamiento y análisis y su posterior uso por las entidades competentes.

Aquí presentamos también el modelo asociado al proyecto:

<table>
<tr><td rowspan="2">Key Partnerships

Universidad de La Laguna</td><td>Key Activities

Análisis de los datos

Creación de nuevas columnas

Creación de modelos

Análisis de resultados</td><td rowspan="2">Value Propositions

Encontrar patrones que determinen las causas y posibles mejoras al ecosistema africano en el ámbito de la salud infantil

Nueva forma de abordar el problema

Técnicas de vanguardia. Análisis de causas y no de consecuencias</td><td>Customer Relationships

----</td><td rowspan="2">Customer Segments

Gobiernos y ONGs con intereses en ayudar a África

OMS

Estudios salud africana</td></tr>
<tr><td>Key Resources

Datos ULL

Datos meteorología + malaria

Recursos Humanos

Recursos tecnológicos</td><td>Channels

Medium.com

Towardsdatascience.com</td></tr>
<tr><td colspan="2">Cost Structure

No existe coste económico pero si tiempo dedicado</td><td colspan="3">Revenue Steams

----</td></tr>
</table>

Figura 10 Modelo del proyecto

Creación de variables y análisis

Este apartado busca plasmar el contexto de los datos usados, así como las transformaciones que se les han realizado a los mismos para obtener un mayor valor de ellos, generando en el proceso los distintos shocks, que serán evaluados por los modelos.

Datos del DHS y del clima

Los datos sobre la salud infantil (niños de 0 a 5 años) se obtienen de la Encuestas demográficas y de salud (DHS). De esta base de datos, también recogemos información relacionada con los niveles de educación, ingresos, ciertas características de lugar y fecha de nacimiento de los niños.

La información climática (temperatura y precipitaciones) se extrae de la "Terrestrial Air Temperature & Precipitacion: 1900-2019, Gridded Monthly Time Series, versión 3.01". Los datos climáticos contienen registros mensuales de ambas variables (temperatura media y cantidad total de precipitaciones) en la superficie de la Tierra. Los datos se presentan como una cuadrícula que cubre toda la superficie de la Tierra limitada por líneas de latitud y longitud. La resolución de estos datos es de 0.5x0.5 grados. Cada uno de estos cuadrados se identifica por un par de coordenadas geográficas de latitud y longitud, correspondiente a la posición de su centro. Además, cada uno de los cuadrados identificado se le asigna un valor de temperatura y precipitación promedio.

Ambas bases de datos (DHS e información climática) se fusionan utilizando coordenadas GPS.

En la encuesta inicial el número total de observaciones alcanza la cifra de 678.648 niños. Alrededor del 10% de los niños muere durante los primeros cinco años de vida. Así, nuestra muestra de niños vivos es de 619.130 observaciones. Además, centrándose en las variables de salud infantil de la EDS, la muestra se reduce a 355.136 para la altura y la edad, y 350.359 para el peso y la altura.

	Num. Obs	Mean	Std.	Min	Max	Obs. <-2	Obs. <-3
Height-per-age	355136	-1.508	1.761	-6.0	6.0	138358	65852
Weight-per-age	350359	-1.051	1.301	-6.0	5.0	75915	24944
Weight-per-Height	350359	-0.265	1.444	-5.0	5.0	37355	14002

Figura 11 Tabla: Child Health (antropometría) estadística descriptiva

	<1 year	[1 2) years	[2 3) years	[3 4) years	[4 5] years	Total
Height-per-age	82911	77947	71436	63797	59045	355136
Weight-per-age	81924	77035	70461	62856	58083	350359
Weight-per-Height	81924	77035	70461	62856	58083	350359

Figura 12 Tabla: Distribución por edades de los niños

Figura 13 DHS clústeres África-subsahariana

Antecedentes Malaria y creación de shocks

VER ANEXOS 2 Y 3 PARA MÁS INFORMACIÓN

Es interesante comenzar esta sección con una visión generalizada y simplificada de la epidemiología, inmunología y muerte infantil.

En esta visión general, nos centramos en la malaria durante el embarazo como el canal en el que el clima afecta a la supervivencia de los niños. A continuación, describimos nuestro índice para medir las condiciones climáticas adecuadas para la infección de la malaria, y utilizarlo para clasificar nuestros diferentes grupos de DHS en diferentes zonas de riesgo de malaria.

Según Black (2010) el paludismo y la mortalidad infantil causaron alrededor del 16% de las muertes de niños menores de cinco años en África, mientras que Snow y otros (1999) estiman que alrededor del 75% de la muerte estimada por paludismo en el África subsahariana en 1995 se compone de niños menores de cinco años. Sin embargo, se sabe que los lactantes tienen una sensibilidad reducida al paludismo durante los primeros meses de vida (Maegraith 1984). Se sabe que el paludismo en el embarazo aumenta la probabilidad de bajo peso al nacer, un importante factor de riesgo de muerte infantil (McCormick 1985). Guyatt y Snow (2001) muestran que el riesgo de bajo peso al nacer se duplica si la madre del bebé es infectada con malaria durante el embarazo, y que el 5,7% de las muertes infantiles en África podría atribuirse al bajo peso al nacer inducido por el paludismo materno.

El paludismo es una enfermedad que necesita un vector (mosquito) para propagarse entre los humanos. La supervivencia y la reproducción de estos vectores y parásitos están fuertemente relacionadas con las condiciones climáticas. En un estudio sobre la transmisión del paludismo en África, Tanser y otros (2003) establecieron un grupo de condiciones relacionadas con la influencia del clima en la transmisión de la malaria. Estas condiciones también fueron utilizadas por Kudamatsu y otros (2012), en su estudio del clima y la mortalidad infantil en África. En particular, estas condiciones son:

1- La precipitación mensual promedio durante los últimos 3 meses es de al menos 60 mm.

2- Las precipitaciones en al menos uno de los últimos 3 meses son de al menos 80 mm.

3- Ningún mes en los últimos 12 meses tiene una temperatura media inferior a 5ºC.

4- La temperatura media en los últimos 3 meses supera los 19,5ºC+SD (mensual temperatura en los últimos 12 meses).

Si un mes satisface las cuatro condiciones, establecemos ese mes como "uno" o lo que es lo mismo, “mes adecuado", y si sólo una de las condiciones falla, fijamos ese mes como un "cero" o “no es un mes adecuado para el paludismo". Como Kudamatsu y otros (2012) afirman, las condiciones 1 y 2 asegurar la existencia de sitios de reproducción y la suficiente humedad del suelo para el vector; la condición 3 es necesaria para la supervivencia del vector, porque las bajas temperaturas impiden la supervivencia del vector. La condición 4 permite que el parásito se vuelva infeccioso en el interior el cuerpo del vector y permite la transmisión efectiva de persona a persona.

Usando la serie de tiempo meteorológico asignada a cada nacimiento en cada grupo de DHS, clasificamos todos meses. Luego, contamos los "meses aptos para el paludismo" en los mismos períodos de tiempo utilizados en la construcción de los choques climáticos. Como resultado, tenemos un grupo de variables que cuentan los "meses aptos para el paludismo" en el primer, segundo y tercer trimestre de embarazo, y en cada año de vida de cero a sesenta meses.

	Longitud	Latitud	Year	Mes	Prep	Temp	msm1	msm2	msm3	msm4	msm_1	msm_2	msm_3	msm_4	sumatorio	Malaria_Shocks
114251	3.75	7.75	2017.0	Agosto	96.0	25.0	True	True	True	True	1	1	1	1	4	Malaria
512997	14.25	9.25	2017.0	Agosto	129.0	26.0	True	True	True	True	1	1	1	1	4	Malaria
269287	10.25	6.25	2017.0	Noviembre	140.5	21.2	True	True	True	True	1	1	1	1	4	Malaria
550546	3.75	10.75	2017.0	Julio	201.4	26.9	False	True	True	True	0	1	1	1	3	No_Malaria
550543	7.75	12.75	2017.0	Junio	147.5	30.4	False	False	True	True	0	0	1	1	2	No_Malaria

Figura 14 Tabla: Clusterización de los Shocks de Malaria

Shocks-climáticos

Los datos climáticos de temperatura y precipitaciones están organizados en un formato de cuadrícula en base a la latitud y la longitud, estos están georreferenciados desde 1900 hasta 2017.

Hemos elegido como período de influencia el plazo veinte años antes del año de la encuesta DHS.

En un paso previo, fusionamos los registros completos de temperatura y precipitación a cada cúmulo de DHS minimizando la distancia entre las coordenadas GPS que identifican el clima y los datos del DHS. Además, asignamos esta información a todos los niños de la DHS clusters. Como resultado, cada niño tiene el registro más cercano de series temporales climáticas asignado.

A continuación, siguiendo un proceso similar al de Maccini y Yang (2009), Andalón, Azevedo et al (2014) y Rosales (2013), para cada país y encuesta, y para cada nacimiento, hemos construido variables de z-scores para cada mes, como la desviación del registro a largo plazo de temperatura y precipitación dividido por la desviación estándar mensual a largo plazo.

El procedimiento para calcular los shocks climáticos es el siguiente:

- Creamos una función que tome cuatro valores de entrada: latitud, longitud, año y mes. Y devuelva el valor de la temperatura y precipitaciones para el clúster en concreto.
- Creamos una segunda función que en este caso va a depender de cinco valores, los mismos que en la función anterior, más el número de años atrás que queremos para obtener datos de temperatura de ese clúster.
    - Para cada clúster guardamos un histórico de veinte datos de temperatura y precipitaciones.
    - Sobre ese conjunto de datos calculamos la media y la desviación típica para quedarnos tan sólo con aquellos valores que estén en el intervalo de la media más/menos cuatro desviaciones. Todo esto es muy útil de cara a no considerar desde un principio valores extremos (outliers).
- Volvemos a crear otra función que tiene como valores de entrada las variables de un registro de nuestra tabla de datos, para que nos devuelva una media aritmética de los valores de la temperatura en los últimos veinte años, así como su desviación.
- Generamos en nuestra tabla de datos una nueva variable que representa la media de la temperatura y precipitaciones para cada clúster en cada momento de tiempo (referida a los últimos veinte años) y hacemos lo mismo con la desviación.
- Ejecutamos este proceso para todo el dataframe. (Aquí es donde nos damos cuenta de que es necesario paralelizar el procesamiento).
- Creamos el estadístico (Zscore) donde el numerador vendrá explicado como la diferencia del valor de la temperatura en el momento actual menos el valor promedio de los últimos veinte años. El denominador será el valor de la desviación de cada clúster. (Del mismo modo con las precipitaciones).
- En un principio generamos 10 tipos de shocks diferentes (como puede apreciarse en el código que adjuntamos) sólo haremos uso en nuestro estudio de los shocks ST+1, ST-1, SP+1, SP-1 y de los cruzados STP++1, STP+-1, STP-+1, STP−1.
- Finalmente, como el estudio de la productividad del suelo nos interesa hacerlo con datos anuales creamos una nueva tabla de datos donde acumulamos los datos mensuales en un solo dato anual.

	Latitud	Longitud	Year	SSP+1_2	SSP+2_3	SSP+3	SSP+1	SSP+2	SSP-1_2)	SSP-2_3	SSP-3	SSP-1	SSP-2
0	32.75	-16.75	1986	2.0	0.0	1.0	3.0	1.0	1.0	0.0	0.0	1.0	0.0
1	28.25	-16.75	1986	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2	21.75	-16.75	1986	0.0	0.0	2.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0
3	21.25	-16.75	1986	0.0	2.0	0.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0
4	15.25	-16.75	1986	1.0	0.0	0.0	1.0	0.0	3.0	0.0	0.0	3.0	0.0

Figura 15 Tabla: Shocks anuales de precipitación

Posibles errores en los datos

Dada la naturaleza de los datos, encontramos cierta incertidumbre en algunos aspectos de los mismos:

Mientras que el nacimiento y la muerte de los hijos son ciertamente definitorios de los eventos, no podemos descartar el error de medición (tal vez más sobre el año que el mes de nacimiento o muerte), ya que las encuestas son retrospectivas. Sin embargo, nuestros resultados no cambian significativamente cuando consideramos los nacimientos reportados 10 años antes de la encuesta. Si los eventos más cercanos en el tiempo son más fáciles recordado, esto sugiere que el error de medición debido a una memoria imperfecta no es un problema importante.

Otro motivo de preocupación es la migración de las madres, de manera que los lugares en el momento de la encuesta pueden diferir del lugar de nacimiento de los niños. Sin embargo, los nacimientos que se producen antes de que las madres se trasladen al lugar de la encuesta, no afecta significativamente a los resultados. Por lo tanto, el posible sesgo de usar datos meteorológicos mal ubicados parece ser pequeño.

Modelos y ventajas

VER APÉNDICE 5, 6 Y 7 PARA MÁS INFORMACIÓN

Dentro del conjunto de modelos que se han realizado, cabe destacar dos de ellos.

Hay un primer modelo para el estudio de la productividad del suelo y un segundo modelo donde se intenta explicar los niveles de salud infantil.

En ambos casos empezamos con un modelo exploratorio de datos (ADA). Por un lado estudiamos la variable objetivo que en el primer caso son los cambios en el NDVI (como indicador de productividad del suelo) y en el segundo caso la variable 'alt_edad'.

La utilización de los modelos de Machine Learning genera una ventaja frente a los modelos de regresión tradicionales ya que en este caso podemos hacer combinaciones no lineales entre variables del grado que deseemos. A tenor de los resultados obtenidos, en este caso, no mejora los niveles explicativos de nuestro target.

Hemos recurrido a un análisis exploratorio complejo por dos motivos, el primero de ellos es porque se cuenta con muchas variables y el segundo es porque tenemos variables categóricas y numéricas. La complejidad en nuestro análisis es que hemos analizado las variables numéricas para ver cuáles interesa categorizar para posteriormente volverlas a convertir en numéricas pero con un rango menor de valores.

Para no trabajar con un modelo en bruto (donde consideramos todas las variables por igual) en el ADA tratamos de analizar la significancia de las variables discriminando entre variables categóricas y numéricas. En el primer grupo intentamos llegar a una convergencia en

los resultados por una vía gráfica y por otra estadística donde utilizamos el p-valor de Pearson para estudiar la significancia.

El segundo grupo, referido a las variables numéricas lo trabajamos directamente a través del p-valor sin contraste gráfico.

Una ventaja importante que tiene trabajar con modelos de machine learning frente a los modelos econométricos tradicionales está en el tratamiento de variables categóricas. En nuestro caso hacemos un tratamiento 'a la carta' donde las variables numéricas con pocos valores únicos las categorizamos en pocas etiquetas para posteriormente volverlas a convertir en numéricas, consiguiéndose un nivel de simplicidad de las variables mayor.

Otra de las ventajas que proporciona en Machine Learning con respecto a los modelos (OLS) es el tratamiento de los outliers. De manera que realizamos un estudio para comprobar la existencia de outliers. Igualmente hay un tratamiento gráfico y numérico del problema. En el caso gráfico nos permite ver que la existencia de valores atípicos no es muy grande y que además sigue la tendencia de la mayoría de los datos por lo que su eliminación no sería lo más correcto. Con la opción de preprocesamiento de los datos podemos escoger la opción de RobustScaler que nos permite trabajar con valores atípicos.

El preprocesamiento de variables lo terminamos con un escalado de las variables ya que se mueven en diferentes rangos y unidades (además, algunas de ellas expresan longitud y es razón suficiente para este procedimiento).

El último paso que hacemos es la selección de variables que también a través de las técnicas de Machine Learning se pueden hacer utilizando varios criterios y en nuestro caso, incluso, hemos considerado varios simultáneamente.

Por un lado tenemos el criterio del p-valor de Pearson conjuntamente con regularización Ridge (consiguiendo un resultado muy parecido al del método OLS, y por otro hemos utilizado el método de selección de variables basado en el gradiente descendente donde el resultado diverge de los dos casos anteriores siendo bastante peor.

Capítulo 3 Resultados

En este capítulo se abordarán algunos de los resultados obtenidos en el estudio, así como las conclusiones obtenidas del mismo. También tendremos en cuenta las líneas de investigación futuras y proyectos que se beneficien de este estudio, que es en donde recae el auténtico valor del mismo.

DAFO

Este apartado busca adaptar el análisis DAFO al modelo planteado en nuestro proyecto. Al tratarse de un modelo social, un estudio, y no de un modelo económico al uso, se modificará la estructura tradicional DAFO para adaptarse a este.

Debilidades

Los principales impedimentos que nos hemos ido encontrando se encuentran en el apartado del procesamiento de los datos. Estos al tratarse en ocasiones de cientos de miles de registros con decenas de columnas, requieren de una potencia de procesamiento, y de una preparación previa de los mismos muy alta. Esto implica no solo el uso de máquinas con altas capacidades, sino del estudio de los mismos con el fin de llegar a la algoritmia que permita el cálculo en una cantidad de tiempo asumible.

Ejemplificando lo anterior, el cálculo de la precipitación y temperatura para n meses anteriores por fuerza bruta, tomaría tiempos virtualmente infinitos; Si a este cálculo se le aplica una lógica básica mediante las propiedades de las tablas, podemos alcanzar tiempos de entre 12 y 36 días en ejecución; Este tiempo es reducible mediante técnicas de paralelización. En una máquina con 12 cores podemos reducir el tiempo tomado a entre 1 y 3 días para el procesado. Finalmente, mediante la escritura de algoritmos más complejos basados en generación de tablas hash, el tiempo de procesamiento usando paralelismo, se puede reducir a entre 35 minutos y 2 horas, pudiendo bajar hasta los 20 minutos para *n's* pequeñas.

Como consecuencia de necesitar realizar todo este proceso para reducir tiempos, la comprobación de errores en cálculos, y las múltiples pruebas han tomado mucho tiempo. Del mismo modo, al aumentar la complejidad de los algoritmos para poder calcular estos en un menor tiempo, aumenta la especialización de los mismos, minimizando la probabilidad de poder reutilizar código de manera más eficiente.

Como podrán suponer, este proceso depende completamente de haber realizado un buen sistema de pruebas que garantice que los resultados obtenidos sean correctos, puesto que un fallo en los pasos de depuración de los datos puede desencadenar una inconsistencia de los modelos a posteriori. Sin embargo el modelo debería ser capaz de soportar cierto grado de incertidumbre al contar no solo con columnas carentes de relación entre ellas, sino columnas interrelacionadas en la que los fallos individuales queden compensados por las columnas correctas. De igual forma, regularizaciones como Ridge y Lasso deberían poder detectar valores carentes de importancia y restarles peso en el modelo.

Amenazas

La amenaza estrella en este año y los posteriores, de obligatorio nombramiento, es el Covid-19. Esta enfermedad ha alterado y se prevé que siga alterando patrones por todo el mundo.

En nuestro estudio, el popularmente conocido Coronavirus, puede afectar a la salud de las familias por toda África, modificando la frecuencia que estas van a comprar, minimizando el número de agricultores, reduciendo el número de ayudas que los países más desarrollados pueden enviar, etc. Y por consecuente creando una variación en la salud infantil, que no solo no sería predecible por un modelo perfectamente entrenado, sino que no permitiría que en un

futuro próximo se usaran datos de estos años, puesto que los anteriores al Covid-19 no reflejarían la realidad de la situación actual y próxima, y debido a la alta importancia del histórico para poder realizar nuestros modelos.

Si la situación actual acabara súbitamente y se pudiera volver en un periodo corto a la "normalidad", o si se prolongara la situación sin grandes variaciones por al menos 5 años, y se realizaran otras encuestas en el futuro, se podría seguir usando el modelo actualizando los datos.

Consideramos que es sabio no lanzar afirmaciones vacías, por lo que no nos atreveríamos a teorizar que el modelo se comportaría de manera correcta en situaciones fuera de las nombradas en el anterior párrafo.

Fortalezas

Al tratarse de un estudio que agrupa varios frentes de acción en el estudio de la salud infantil en África, así como ser pionero en el uso de técnicas de estadística avanzadas, Machine Learning y Deep Learning, podemos abarcar estos datos que mediante métodos más tradicionales no podrían ser analizados como se ha visto ya.

Oportunidades

Del mismo modo que nuestro trabajo se apoya en muchos autores y profesionales, este mismo podría servir de motivante y apoyo para estudios posteriores que cuenten con más personal y presupuesto, sentando las bases de la investigación en las hipótesis lanzadas en este proyecto.

Así mismo, simplemente este acercamiento a las causas de los niveles de salud infantil en África puede suponer un ahorro enorme a organizaciones que busque ayudar en el continente, pues es posible no solo paliar los efectos, sino encontrar las causas y atacar el problema de raíz.

Resultados

La muestra total se divide por la edad del niño, y estimamos (1) utilizando diferentes submuestras: niños con menos de un año, entre uno y dos años, dos a tres, tres a cuatro y cuatro a cinco. Estudiamos de esta manera si los efectos del clima en la salud infantil tienen efectos diferentes según la edad, centrándonos especialmente en si las edades más tempranas de la vida son más vulnerables y si el efecto desaparece con la edad (efecto de recuperación).

Uno de los objetivos finales del trabajo es analizar si los efectos se limitan a los primeros años de vida o, por el contrario, los choques experimentados en edades tempranas son significativos para la altura y el peso una vez alcanzados, por ejemplo, a los 5 años. También queremos comprobar si esta transmisión depende del tipo de shock (temperatura, precipitaciones o malaria) y del nivel educativo de la madre, que como factor mediador relevante para prevenir o corregir los shocks negativos para la salud del niño.

La categoría omitida para el Índice de Riqueza es "la más pobre", y para la educación de la madre es "sin educación", por lo que todos los demás coeficientes se refieren a estas categorías omitidas. Obtenemos los resultados esperados: tanto la riqueza como la educación de las madres afectan positivamente el estado de salud (talla para la edad) del niño menor de 5 años. En ambos casos, los coeficientes son siempre positivos y significativos, y el coeficiente estimado aumenta con el nivel de riqueza y educación. En segundo lugar, el orden entre los hermanos es generalmente significativo, pero especialmente para la segunda y tercera posición cuyos coeficientes son positivos. Más allá de esa posición, las diferencias con respecto a ser el primero no son significativas. Además, los coeficientes de ser gemelo o más son siempre negativos y significativos, que es un resultado esperado. El mes de nacimiento también es significativo (aunque no se muestra en la tabla), probablemente por su relación con el clima

(este control debe interpretarse como un efecto fijo mensual). En promedio, nacer en una zona rural tiene un efecto negativo en la estatura del niño, lo que coincide con la literatura existente.

In-utero (temp., last 3), +	0.00981**	Rural (Dummy)	-0.104***
	(2.44)		(-6.33)
In-utero (temp., last 6), +	-0.00713*	Wealth 2	0.0718***
	(-1.75)		(6.54)
In-utero (temp., last 9), +	0.00682	Wealth 3	0.134***
	(1.60)		(11.12)
In-utero (temp., last 3), -	-0.0115*	Wealth 4	0.269***
	(-1.71)		(20.54)
In-utero (temp., last 6), -	0.00354	Wealth 5	0.553***
	(0.53)		(32.92)
In-utero (temp., last 9), -	-0.00216	Primary	0.0428***
	(-0.33)		(4.58)
In-utero (rain, last 3), +	-0.00524	Secondary-higher	0.209***
	(-0.92)		(17.66)
In-utero (rain, last 6), +	-0.0203***	Gender (child)	0.188***
	(-3.72)		(31.43)
In-utero (rain, last 9), +	-0.00543	Child (mult-born)	-0.651***
	(-0.98)		(-29.37)
In-utero (rain, last 3), -	-0.0198***		
	(-3.01)		
In-utero (rain, last 6), -	-0.00269		
	(-0.41)		
In-utero (rain, last 9), -	0.00880		
	(1.28)		
Temp. Shock (12m), +	-0.0236***		
	(-8.94)		
Temp. Shock (12m), -	-0.0261***		
	(-7.34)		
Rain Shock (12m), +	-0.0328***		
	(-9.16)		
Rain Shock (12m), -	-0.0121***		
	(-2.74)		
Observations	320641		
R2	0.210		

Standard deviation (robust, clustered) in parenthesis: significant (***: 1%; **: 5%; *: 10%).

Figura 16 Tabla: Altura por edad, shocks de precipitaciones y temperatura

Distinguiendo por edades, analizamos si el posible impacto de los choques climáticos (o Malaria) ocurridos en torno al mes-año del nacimiento del niño se reduce con los años o, por el contrario, se mantiene. Recordemos que las variables antropométricas del niño se miden en el momento en que se realiza la encuesta, mientras que los choques climáticos se miden en el momento del nacimiento y durante su vida.

La Figura 17 incluye resultados cuando se incluyen los choques climáticos generales (de temperatura y precipitaciones).

	<1 year	[1 2) years	[2 3) years	[3 4) years	[4 5] years		<1 year	[1 2) years	[2 3) years	[3 4) years	[4 5] years
In-utero (temp., last 3), +	-0.0148	-0.00732	-0.000177	0.00371	0.0254**	Temp. Shock (24m), +		-0.0301***	-0.0109*	0.00730	0.00537
	(-1.24)	(-0.68)	(-0.02)	(0.30)	(2.30)			(-3.36)	(-1.88)	(1.19)	(0.95)
In-utero (temp., last 6), +	-0.0247**	-0.00578	-0.000634	0.0189*	0.00971	Temp. Shock (24m), -		-0.0287**	-0.00661	0.00161	-0.00150
	(-2.25)	(-0.54)	(-0.06)	(1.70)	(0.85)			(-2.02)	(-0.64)	(0.18)	(-0.14)
In-utero (temp., last 9), +	0.000630	0.0117	0.00720	0.0292**	0.00719	Rain Shock (24m), +		-0.0296**	-0.0114	0	0.00218
	(0.06)	(1.07)	(0.71)	(2.53)	(0.68)			(-2.36)	(-1.27)	(.)	(0.22)
In-utero (temp., last 3), -	-0.0165	-0.00633	0.0159	-0.0258	0.00926	Rain Shock (24m), -		-0.00716	-0.00662	0.0131	0.00799
	(-0.85)	(-0.40)	(0.85)	(-1.42)	(0.52)			(-0.50)	(-0.59)	(1.15)	(0.73)
In-utero (temp., last 6), -	0.0263	0.0157	0.0154	-0.0425**	0.0156	Temp. Shock (36m), +			-0.0150*	-0.0132**	0.00826
	(1.45)	(0.96)	(0.84)	(-2.38)	(0.95)				(-1.80)	(-2.28)	(0.57)
In-utero (temp., last 9), -	0.00187	-0.00514	-0.0118	0.00822	-0.0105	Temp. Shock (36m), -			-0.00604	0.0271**	0.0117
	(0.10)	(-0.31)	(-0.62)	(0.18)	(-0.62)				(-0.43)	(2.42)	(1.21)
In-utero (rain, last 3), +	-0.0127	-0.00677	-0.00415	0.00298	0.00979	Rain Shock (36m), +			-0.0101	-2.06e-15***	-0.000554
	(-0.83)	(-0.47)	(-0.27)	(0.22)	(0.69)				(-0.86)	(-79.43)	(-0.06)
In-utero (rain, last 6), +	-0.0484***	-0.0141	-0.0148	-0.0141	-0.0292**	Rain Shock (36m), -			0.000638	-0.0124	0.0117
	(-3.17)	(-0.95)	(-1.06)	(-1.01)	(-2.16)				(0.04)	(-1.08)	(1.01)
In-utero (rain, last 9), +	-0.0498***	0.00600	0.0169	-0.0112	-0.000635	Temp. Shock (48m), +				-0.0106	-0.00328
	(-3.28)	(0.43)	(1.19)	(-0.77)	(-0.05)					(-1.26)	(-0.59)
In-utero (rain, last 3), -	0.0189	0.00205	0.00131	0.00250	0.0119	Temp. Shock (48m), -				0.0259**	0.0185*
	(1.01)	(0.12)	(0.07)	(0.14)	(0.72)					(1.97)	(1.73)
In-utero (rain, last 6), -	0.0304*	0.0350**	0.0285*	0.00401	0.00826	Rain Shock (48m), +				0	-0.00547
	(1.68)	(2.00)	(1.67)	(0.23)	(0.48)					(.)	(-0.57)
In-utero (rain, last 9), -	0.0597***	0.00660	-0.0287*	-0.0193	-0.00732	Rain Shock (48m), -				-0.0226	-0.00472
	(3.35)	(0.38)	(-1.65)	(-1.11)	(-0.43)					(-1.47)	(-0.42)
Temp. Shock (12m), +	-0.0532***	-0.00463	0.00744	0.00837	0.00682	Temp. Shock (60m), +					-0.00692
	(-5.48)	(-0.72)	(1.24)	(1.44)	(1.06)						(-0.89)
Temp. Shock (12m), -	-0.0576***	0.00307	0.0204**	-0.00748	-0.00181	Temp. Shock (60m), -					0.00846
	(-4.00)	(0.31)	(2.15)	(-0.67)	(-0.18)						(0.70)
Rain Shock (12m), +	-0.0266**	-0.00378	-0.00759	0.00798	0.000223	Rain Shock (60m), +					-0.00278
	(-2.10)	(-0.37)	(-0.84)	(0.77)	(0.02)						(-0.24)
Rain Shock (12m), -	0.0315**	0.0139	0.00964	0.00436	0.0172	Rain Shock (60m), -					0.00530
	(2.03)	(1.28)	(0.82)	(0.36)	(1.49)						(0.34)
Num. Obs	73744	70011	63992	58539	54355						
R2	0.204	0.230	0.232	0.244	0.263						

Standard deviation (robust, clustered) in parenthesis: significant (***: 1%; **: 5%; *: 10%).

Figura 17 Tabla: Altura por edad, shocks de precipitación y temperatura. Distribución por edad del niño.

La Figura 18 muestra los resultados cuando los choques climáticos generales son sustituidos por los choques de la malaria.

En el útero, los choques de malaria afectan de manera positiva y significativa a la talla para la edad cuando se considera la muestra completa. La omisión de otras variables climáticas relacionadas con la calidad y cantidad de la cosecha, por ejemplo, puede afectar este signo. Por el contrario, su coeficiente es negativo y significativo para niños menores de un año. Además, los choques contemporáneos de malaria (es decir, ocurridos durante el año de vida) son siempre negativos y muy significativos para los niños entre 0 y 2 años, y también entre 4 y 5 años.

	All sample	<1 year	[1 2) years	[2 3) years	[3 4) years	[4 5] years
In-utero (Malaria, last 3)	0.00551	-0.0284**	0.000729	0.0157	0.0174	0.0124
	(1.23)	(-2.32)	(0.06)	(1.41)	(1.47)	(1.14)
In-utero (Malaria, last 6)	0.00866***	-0.0000328	-0.00410	0.00644	-0.00568	0.00294
	(2.60)	(-0.00)	(-0.49)	(0.80)	(-0.68)	(0.36)
In-utero (Malaria, last 9)	0.0173***	0.0147	0.00520	-0.0160	0.00935	0.00571
	(3.94)	(1.24)	(0.50)	(-1.54)	(0.85)	(0.58)
Malaria Shock (12m)	-0.0534***	-0.0187**	0.00856	-0.0187**	-0.00277	0.000898
	(-18.95)	(-2.48)	(0.88)	(-2.00)	(-0.31)	(0.09)
Malaria Shock (24m)			-0.0319***	0.0112	-0.00190	0.00227
			(-4.51)	(1.13)	(-0.20)	(0.27)
Malaria Shock (36m)				-0.00602	-0.00671	-0.00803
				(-0.89)	(-0.65)	(-0.88)
Malaria Shock (48m)					-0.00786	-0.00843
					(-1.06)	(-0.92)
Malaria Shock (60m)						-0.0112*
						(-1.69)
Observations	320641	73744	70011	63992	58539	54355
R2	0.210	0.202	0.230	0.232	0.243	0.263

Figura 18 Tabla 3: Altura por edad y shocks de precipitación y temperatura distribuidos por edad del niño.

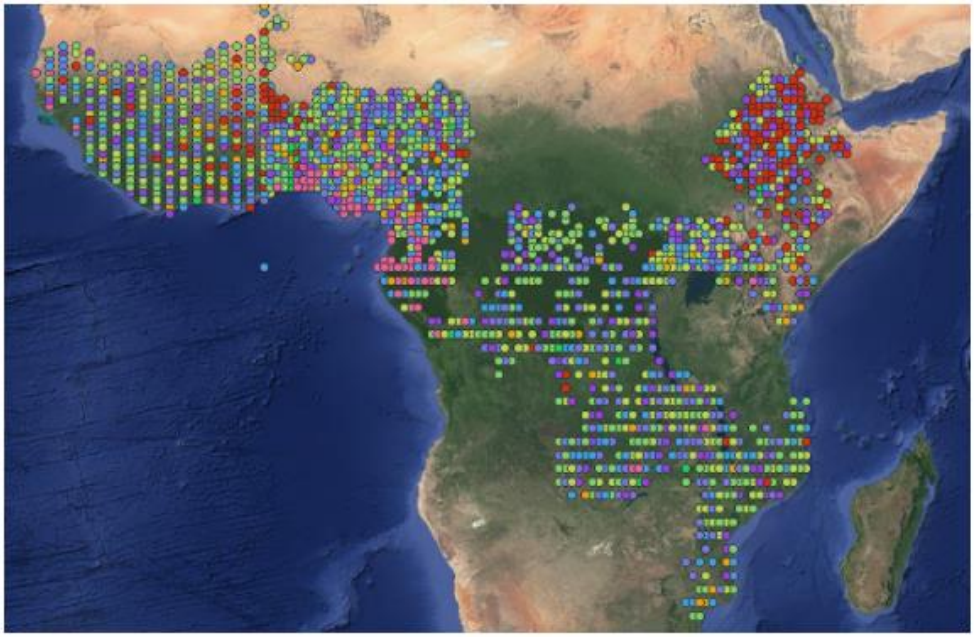
La Figura 19 muestra los resultados estimados para un modelo que incluye todas las variables, los choques climáticos generales y los choques de malaria. Incluir en el mismo modelo la malaria y los shocks climáticos generales refuerzan el impacto negativo de los shocks de malaria en la salud. Quizás, eso se debe a que ahora los choques de temperatura y precipitación están capturando parcialmente su impacto positivo a través de la cosecha. En general, los resultados son básicamente consistentes con los que se encuentran en los cuadros 4 y 5. Si bien los choques intrauterinos parecen desaparecer con la edad, los choques ocurridos durante el último año de vida tienen efectos negativos en la salud infantil hasta los 5 años. Sin embargo, estos choques son más importantes para los niños más pequeños.

	<1 year	[1 2) years	[2 3) years	[3 4) years	[4 5] years		<1 year	[1 2) years	[2 3) years	[3 4) years	[4 5] years
In-utero (temp., last 3), +	-0.0164	-0.00561	0.00373	0.00270	0.0268**	Temp. Shock (24m), +		-0.0288***	-0.00979*	0.00715	0.00495
	(-1.37)	(-0.52)	(0.36)	(0.22)	(2.42)			(-3.17)	(-1.68)	(1.16)	(0.87)
In-utero (temp., last 6), +	-0.0251**	-0.00630	0.00290	0.0190*	0.0114	Temp. Shock (24m), -		-0.0297**	-0.00563	0.00121	-0.00112
	(-2.28)	(-0.58)	(0.28)	(1.71)	(0.99)			(-2.08)	(-0.54)	(0.13)	(-0.11)
In-utero (temp., last 9), +	0.00107	0.00889	0.00408	0.0291**	0.00761	Rain Shock (24m), +		-0.0188	-0.0140	0	0.00239
	(0.10)	(0.82)	(0.40)	(2.51)	(0.72)			(-1.47)	(-1.54)	(.)	(0.24)
In-utero (temp., last 3), -	-0.0160	-0.00724	0.0164	-0.0257	0.0106	Rain Shock (24m), -		0.000538	-0.00492	0.0128	0.00896
	(-0.82)	(-0.45)	(0.87)	(-1.41)	(0.59)			(0.04)	(-0.43)	(1.10)	(0.80)
In-utero (temp., last 6), -	0.0251	0.0172	0.0208	-0.0419**	0.0178	Temp. Shock (36m), +			-0.0169**	-0.0128**	0.00277
	(1.39)	(1.05)	(1.12)	(-2.35)	(1.08)				(-2.00)	(-2.21)	(0.48)
In-utero (temp., last 9), -	0.000757	-0.00578	-0.0104	0.00182	-0.0112	Temp. Shock (36m), -			-0.00778	0.0272**	0.0108
	(0.04)	(-0.35)	(-0.54)	(0.10)	(-0.65)				(-0.54)	(2.41)	(1.11)
In-utero (rain, last 3), +	-0.00817	-0.0105	-0.00609	0.000598	0.00710	Rain Shock (36m), +			-0.00718	-2.07e-15***	0.000983
	(-0.53)	(-0.72)	(-0.39)	(0.04)	(0.50)				(-0.59)	(-78.00)	(0.11)
In-utero (rain, last 6), +	-0.0461***	-0.0144	-0.0172	-0.0148	-0.0319**	Rain Shock (36m), -			0.00491	-0.0138	0.0108
	(-3.02)	(-0.96)	(-1.21)	(-1.04)	(-2.35)				(0.33)	(-1.20)	(0.92)
In-utero (rain, last 9), +	-0.0500***	0.00576	0.0210	-0.0117	-0.00253	Temp. Shock (48m), +				-0.0106	-0.00249
	(-3.31)	(0.41)	(1.45)	(-0.80)	(-0.20)					(-1.24)	(-0.44)
In-utero (rain, last 3), -	0.0231	0.00139	-0.00594	0.00235	0.0118	Temp. Shock (48m), -				0.0254*	0.0192*
	(1.23)	(0.08)	(-0.33)	(0.13)	(0.70)					(1.93)	(1.79)
In-utero (rain, last 6), -	0.0321*	0.0295*	0.0231	0.00295	0.00289	Rain Shock (48m), +				0	-0.00423
	(1.77)	(1.68)	(1.34)	(0.16)	(0.17)					(.)	(-0.44)
In-utero (rain, last 9), -	0.0561***	0.00913	-0.0245	-0.0185	-0.00857	Rain Shock (48m), -				-0.0210	-0.00694
	(3.12)	(0.53)	(-1.41)	(-1.05)	(-0.51)					(-1.36)	(-0.62)
Temp. Shock (12m), +	-0.0513***	-0.00357	0.00616	0.00831	0.00657	Temp. Shock (60m), +					-0.00711
	(-5.21)	(-0.55)	(1.03)	(1.42)	(1.01)						(-0.90)
Temp. Shock (12m), -	-0.0563***	0.00570	0.0192**	-0.00814	-0.00206	Temp. Shock (60m), -					0.00791
	(-3.91)	(0.57)	(2.02)	(-0.73)	(-0.21)						(0.66)
Rain Shock (12m), +	-0.0238*	-0.00487	-0.00440	0.00907	-0.000413	Rain Shock (60m), +					0.00324
	(-1.84)	(-0.48)	(-0.48)	(0.87)	(-0.04)						(0.27)
Rain Shock (12m), -	0.0318**	0.0146	0.00657	0.00343	0.0159	Rain Shock (60m), -					0.0101
	(2.04)	(1.35)	(0.55)	(0.28)	(1.36)						(0.64)
In-utero (Malaria, last 3)	-0.0195*	0.00390	0.0169	0.0139	0.0141	Malaria Shock (12m)	-0.0116	0.0102	-0.0181*	-0.00641	0.000907
	(-1.63)	(0.34)	(1.48)	(1.14)	(1.27)		(-1.51)	(1.04)	(-1.88)	(-0.71)	(0.09)
In-utero (Malaria, last 6)	0.00120	-0.00407	0.00608	0.000984	0.00685	Malaria Shock (24m)		-0.0271***	0.0114	-0.000184	0.00185
	(0.13)	(-0.47)	(0.73)	(0.11)	(0.80)			(-3.72)	(1.14)	(-0.02)	(0.20)
In-utero (Malaria, last 9)	0.0162	0.00385	-0.0182*	0.0141	0.00853	Malaria Shock (36m)			-0.00253	-0.00744	-0.00683
	(1.39)	(0.37)	(-1.71)	(1.27)	(0.84)				(-0.36)	(-0.72)	(-0.72)
Num. Obs	73744	70011	63992	58539	54355	Malaria Shock (48m)				-0.00643	-0.00916
R2	0.204	0.230	0.233	0.244	0.264					(-0.86)	(-0.97)
						Malaria Shock (60m)					-0.0124*
											(-1.82)

Figura 19 Tabla 4: Altura por edad, shocks de precipitación, temperatura, Malaria distribuidos por edad del niño.

Conclusiones

Se han encontrado patrones en los datos y se han realizados diversas tablas que agrupan los coeficientes de estas variables. Estos datos a continuación deberán ser tomados por los organismos, filtrar aquellos datos que manejen, y estudiar las variaciones que supone variar estas métricas.



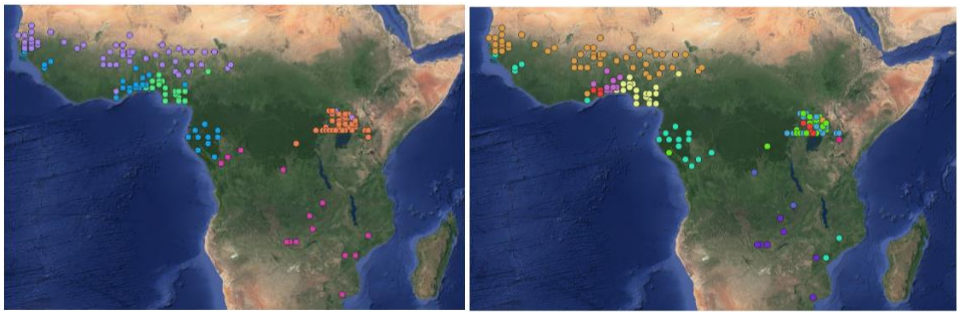


Figura 20 Clusterización de los niños basándonos en las métricas usadas

	<1 year	[1 2) years	[2 3) years	[3 4) years	[4 5] years		<1 year	[1 2) years	[2 3) years	[3 4) years	[4 5] years
In-utero (temp., last 3), +	-0.0148	-0.00732	-0.000177	0.00371	0.0254**	Temp. Shock (24m), +		-0.0301***	-0.0109*	0.00730	0.00537
	(-1.24)	(-0.68)	(-0.02)	(0.30)	(2.30)			(-3.36)	(-1.88)	(1.19)	(0.95)
In-utero (temp., last 6), +	-0.0247**	-0.00578	-0.000634	0.0189*	0.00971	Temp. Shock (24m), -		-0.0287**	-0.00661	0.00161	-0.00150
	(-2.25)	(-0.54)	(-0.06)	(1.70)	(0.85)			(-2.02)	(-0.64)	(0.18)	(-0.14)
In-utero (temp., last 9), +	0.000630	0.0117	0.00720	0.0292**	0.00719	Rain Shock (24m), +		-0.0296**	-0.0114	0	0.00218
	(0.06)	(1.07)	(0.71)	(2.53)	(0.68)			(-2.36)	(-1.27)	(.)	(0.22)
In-utero (temp., last 3), -	-0.0165	-0.00633	0.0159	-0.0258	0.00926	Rain Shock (24m), -		-0.00716	-0.00662	0.0131	0.00799
	(-0.85)	(-0.40)	(0.85)	(-1.42)	(0.52)			(-0.50)	(-0.59)	(1.15)	(0.73)
In-utero (temp., last 6), -	0.0263	0.0157	0.0154	-0.0425**	0.0156	Temp. Shock (36m), +			-0.0150*	-0.0132**	0.00326
	(1.45)	(0.96)	(0.84)	(-2.38)	(0.95)				(-1.80)	(-2.28)	(0.57)
In-utero (temp., last 9), -	0.00187	-0.00514	-0.0118	0.00322	-0.0105	Temp. Shock (36m), -			-0.00604	0.0271**	0.0117
	(0.10)	(-0.31)	(-0.62)	(0.18)	(-0.62)				(-0.43)	(2.42)	(1.21)
In-utero (rain, last 3), +	-0.0127	-0.00677	-0.00415	0.00298	0.00979	Rain Shock (36m), +			-0.0101	-2.06e-15***	-0.000554
	(-0.83)	(-0.47)	(-0.27)	(0.22)	(0.69)				(-0.86)	(-79.43)	(-0.06)
In-utero (rain, last 6), +	-0.0484***	-0.0141	-0.0148	-0.0141	-0.0292**	Rain Shock (36m), -			0.000638	-0.0124	0.0117
	(-3.17)	(-0.95)	(-1.06)	(-1.01)	(-2.16)				(0.04)	(-1.08)	(1.01)
In-utero (rain, last 9), +	-0.0498***	0.00600	0.0169	-0.0112	-0.000635	Temp. Shock (48m), +				-0.0106	-0.00328
	(-3.28)	(0.43)	(1.19)	(-0.77)	(-0.05)					(-1.26)	(-0.59)
In-utero (rain, last 3), -	0.0189	0.00205	0.00131	0.00250	0.0119	Temp. Shock (48m), -				0.0259**	0.0185*
	(1.01)	(0.12)	(0.07)	(0.14)	(0.72)					(1.97)	(1.73)
In-utero (rain, last 6), -	0.0304*	0.0350**	0.0285*	0.00401	0.00826	Rain Shock (48m), +				0	-0.00547
	(1.68)	(2.00)	(1.67)	(0.23)	(0.48)					(.)	(-0.57)
In-utero (rain, last 9), -	0.0597***	0.00660	-0.0287*	-0.0193	-0.00732	Rain Shock (48m), -				-0.0226	-0.00472
	(3.35)	(0.38)	(-1.65)	(-1.11)	(-0.43)					(-1.47)	(-0.42)
Temp. Shock (12m), +	-0.0532***	-0.00463	0.00744	0.00837	0.00682	Temp. Shock (60m), +					-0.00692
	(-5.48)	(-0.72)	(1.24)	(1.44)	(1.06)						(-0.89)
Temp. Shock (12m), -	-0.0576***	0.00307	0.0204**	-0.00748	-0.00181	Temp. Shock (60m), -					0.00846
	(-4.00)	(0.31)	(2.15)	(-0.67)	(-0.18)						(0.70)
Rain Shock (12m), +	-0.0266**	-0.00378	-0.00759	0.00793	0.000223	Rain Shock (60m), +					-0.00278
	(-2.10)	(-0.37)	(-0.84)	(0.77)	(0.02)						(-0.24)
Rain Shock (12m), -	0.0315**	0.0139	0.00964	0.00436	0.0172	Rain Shock (60m), -					0.00530
	(2.03)	(1.28)	(0.82)	(0.36)	(1.49)						(0.34)

Figura 21 Tabla: Fragmento de la tabla de resultados: Altura para la edad, shocks de precipitación y temperatura. Distribuido por edad del niño.

Líneas Futuras

Es indiscutible que la cantidad de recursos que se destinan al continente africano son limitados. Este estudio no pretende solucionar el problema, sin embargo puede servir de base para optimizar la distribución de los mismos al comprender mejor algunos de los motivos que causan los bajos niveles de salud infantil.

Del mismo modo, puede servir de inspiración o de base para estudios más específicos propios de cada asociación o gobierno, centrándose este en los recursos y métricas que contemple cada organismo.

Bajo nuestro criterio, cuanto más se lleguen a entender las causas, más sencillo será mejorar los niveles de salud.

El estudio que hemos hecho para intentar encontrar alguna causalidad a los niveles de salud infantil se puede enfocar con otra perspectiva, pero manteniendo un denominador común que es la consideración de los datos climáticos para la generación de shocks. Las líneas futuras de investigación podrían centrarse en los efectos de los 'shocks económicos' que podría tener un enfoque doble, por un lado, se puede plantear la posibilidad de la existencia de un "coste de oportunidad": un impacto negativo en la economía local disminuye la participación en el mercado laboral, lo que hace más atractivo unirse a la rebelión (y es el punto de partida para la consideración de una nueva variable que hasta el momento no habíamos tenido en cuenta,

'conflictos bélicos-guerras civiles'. Por otro lado, el impacto negativo de los shocks económicos implica que la parte del pastel que se va a apropiar es menor reduciendo los incentivos a luchar. En este punto se estudiaría el efecto neto que a priori es ambiguo, dependiendo, entre otras cosas si el shock se produce en un sector intensivo en mano de obra o en capital. En el caso que nos ocupa que es la zona africana, más concretamente la subsahariana, la agricultura es intensiva en mano de obra de modo que los shocks agrícolas negativos deberían conducir a más conflictos. Los shocks económicos también pueden sumarse siendo causa de un empeoramiento de la pobreza y de las desigualdades. En este punto se podría analizar el impacto de las políticas y capacidad del estado a través de las políticas fiscales en función a la evolución de las bases impositivas, acto que condicionaría directamente su postura de debilidad o fortaleza para luchar contra la rebelión. Por otro lado, si los shocks económicos afectan a las infraestructuras, un aumento de los conflictos bélicos puede venir de la mano de grandes dificultades para reprimir insurgentes.

Por otro lado, podríamos hacer un cambio en el tratamiento de los shocks climáticos, es decir, nos centraríamos sólo en las temporadas de crecimiento agrícola permitiéndonos aislar los efectos que son exclusivamente de la agricultura.

En definitiva, con esta línea de investigación podríamos dar respuestas a preguntas como:

- ¿Existe una relación positiva entre los shocks climáticos relevantes para la agricultura y los conflictos en una zona?
- ¿El conflicto exhibe una persistencia alta o baja tanto en tiempo como en espacio?
- ¿El clima fuera de la temporada de crecimiento agrícola tiene algún efecto sobre los conflictos?
- Los efectos secundarios de los conflictos, ¿son más fuertes en lugares fronterizos o entre las etnias?

Capítulo 4 Summary and Conclusions

One of the final objectives of the work is to analyze whether the effects are limited to the first years of life or, on the contrary, the shocks experienced at an early age are significant for height and weight once they are reached, for example, at 5 years.

We also want to check whether this transmission depends on the type of shock (temperature, rainfall or malaria) and the educational level of the mother, which is a relevant mediating factor to prevent or correct negative shocks to the child's health.

Patterns have been found in the data and various tables have been made that group the coefficients of these variables. These data will then be taken by the agencies, filter those data they handle, and study the variations that vary these metrics.

We have include around the document these information (See: Capítulo 3 -> Resultados).

In general we can say that we are happy with the results gotten and specially with all the process to get these ones. This proyect is not ended because we had recopilate the data, preprocess and made models to can clasificate and predict older and new data. But now is where others can use well this study.

Bibliografía

Alderman, H., Hoddinott, J., & Kinsey, B. (2006). Long term consequences of early childhood malnutrition. Oxford Economic Papers, 58(3), 450-474.

Barrios, S., Bertinelli, L., & Stobl, E. (2010). Trends in rainfall and Economic Growth in Africa: A Neglected Cause of the African Growth Tragedy. Review of Economics and Statistics, 92(2), 350-366.

Bleakley, H. (2010). Malaria eradication in the americas: A retrospective analysis of childhood exposure. American Economic Journal: Applied Economics, 2(2), 1-45.

Burgess, R., Deschênes, O., Donaldson, D., & Greenstone, M. (2011). Weather and Death in India. Cambridge, United States: Massachusetts Institute of Technology, Department of Economics.

Chatterji, P., Dohyung, K., & Kahal, L. (2014). Birthweight and Academic Achievement in Childhood. Health economics, 23(9), 1013-1035.

Currie, J., & Vogl, T. (2013). Early-Life Health and Adult Circumstance in Developing Countries. Annual Review of Economics, Annual Reviews, 5(1), 1-36.

Deaton, A. (2013). The Great Escape: Health, Wealth and the Origins of Inequality. Princeton University Press.

Dell, M., Jones, B., & Olken, B. (2012). Temperature Shocks and Economic Growth: Evidence from the Last Half Century. American Economic Journal: Macroeconomics, 4(3), 66-95.

Dell, M., Jones, B., & Olken, B. (2014). What Do We Learn from the Weather? The New Climate–Economy Literature. Journal of Economic Literature, 52(3), 740-798.

Deschênes, O., & Moretti, E. (2009). Extreme Weather Events, Mortality, and Migration. Review of Economics and Statistics, 91(4), 659-681.

Deschênes, O., Greenstone, M., & Guryan, J. (2009). Climate Change and Birth Weight. American Economic Review, 99(2), 211-217.

Dressen, T. (2014). Early Life Rainfall and Later Life Human Capital Outcomes in Bangladesh. San Francisco: University of San Francisco.

Glewwe, P., & King, E. (2001). The Impact of Early Childhood Nutrition and Academic. World Bank Economic Review, 15(1), 81-114.

Gwatkin, D., Rutstein, S., Johnson, K., Sulimat, E., Wagstaff, A., & Amouzou, A. (2007). Socio-economic differences in health, nutrition, and population, within developing countries. Washington DC: The World Bank.

Harari, M., & La Ferrara, E. (2012). Conflict, climate and cells: a disaggregated analysis. IGIER Working Paper No. 461, Innocenzo Gasparini Institute for Economic Research. Milan.

Henderson, J., Storeygard, A., & Deichmann, U. (2014). Is climate change driving urbanization in Africa?

Hoddinott, J., & Kinsey, B. (2001). Child Growth in the Time of Drought. Oxford Bulletin of Economics and Statistics, 63(4), 409-436.

Hsiang, S., & Jina, A. (2014). The Causal Effect of Environmental Catastrophe on Long-Run Economic Growth. NBER Working Paper Series, Working paper No 20352.

Lucas, A. (2010). Malaria Eradication and Educational Attainment: Evidence from Paraguay and Sri Lanka. American Economic Journal: Applied Economics, 2(2), 46–71.

Maccini, S., & Yang, D. (2009). Under the Weather: Health, Schooling, and Economic Consequences of Early-Life Rainfall. American Economic Review, 99(3), 1006- 1026.

Matsuura, K., & Willmott, C. (2012). Terrestrial Air Temperature & Precipitacion: 1900-2010, Gridded Monthly Time Series, versión 3.01. University of Delaware AirTemperatura & Precipitation data. Newark, Delaware: University of Delaware.

National Population Commission of Nigeria (NPC) and ICF Macro. (2009). Nigeria Demographic and Health Survey 2008, Dataset. Abuja, Nigeria: National population Commission and ICF Macro.

National Population Commission of Nigeria (NPC) and ICF Macro. (2009). Nigeria Demographic and Health Survey 2008, Final Report. Abuja, Nigeria: National Population Commission and ICF Macro.

Anexos

A partir de aquí se incluyen los Anexos del proyecto, en ellos se podrá ver el código usado a lo largo del estudio, así como las interacciones con los diversos programas usados.

Dado que su formato es de *Captura de Pantalla*, no se han incluido en el *Índice de Tablas y Figuras*.

Anexo 1 – Países y provincias

Los datos de países, provincias y distritos los hemos obtenido a través de Qgis. En este apéndice explicaré cómo lo hicimos para ponerle nombre de países, provincias y distritos a todas las coordenadas geográficas que teníamos en las tablas de datos climáticas. El problema nos surgió cuando consideramos que podía ser interesante incluir una nueva variable de georreferenciación a nuestro conjunto de datos que podría tener un peso explicativo importante en la parte de productividad del suelo.

El segundo problema se nos presentó cuando en los datos facilitados por la FAO lo que teníamos eran puntos de zonas agrícolas (Crop Áreas) y eso nos obligaba a considerar esa zona en concreta y no toda la provincia y mucho menos generalizar para el país.

```
1 import pandas as pd
```

```
1 df1 = pd.read_excel('alturas_2.xlsx')
```

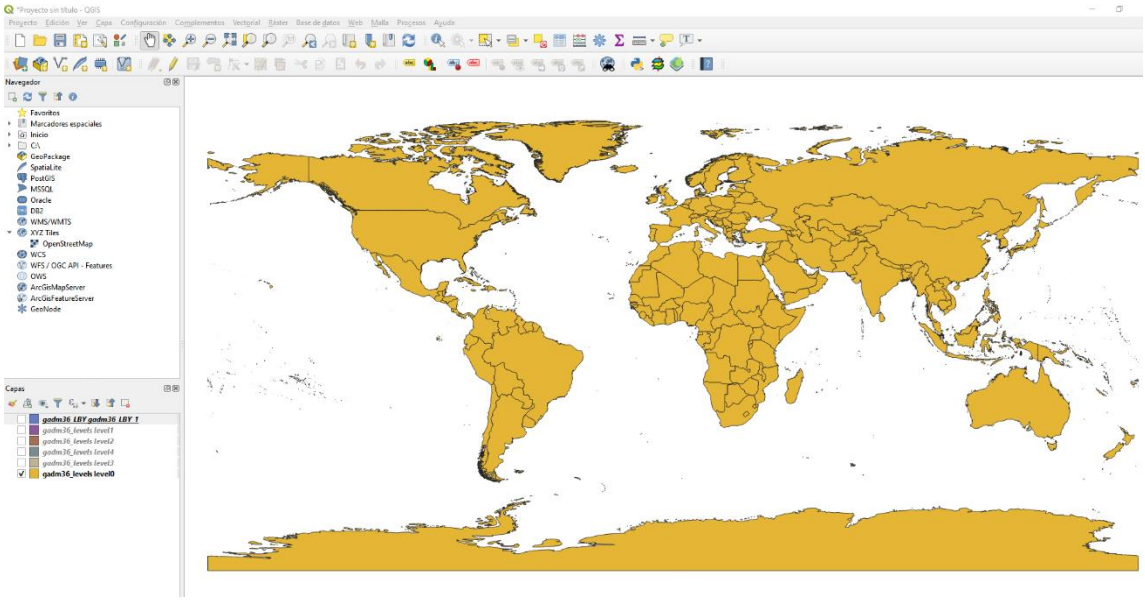
```
1 df1
```

	WLATITUDE	WLONGITUDE	WELEV
0	10.14740	104.6880	-9999
1	10.14740	105.0000	2
2	10.14740	105.3120	2
3	10.14740	105.6250	1
4	10.14740	105.9380	1
...	...	...	...
177570	-9.83521	-77.8125	3230
177571	-9.83521	-78.1250	609
177572	-9.83521	-78.4375	-9999
177573	-9.83521	-78.7500	-9999
177574	-9.83521	-79.0625	-9999

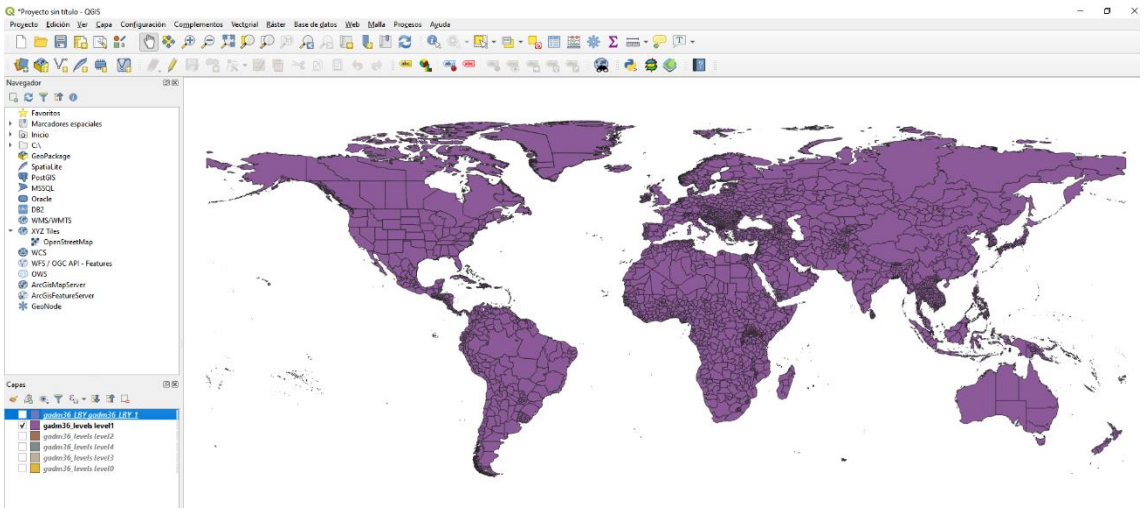
177575 rows × 3 columns

Si observamos, los datos de altura sólo vienen asociados a una referencia geográfica, pero necesitamos saber qué zona es y para ello lo que hicimos fue lo siguiente:

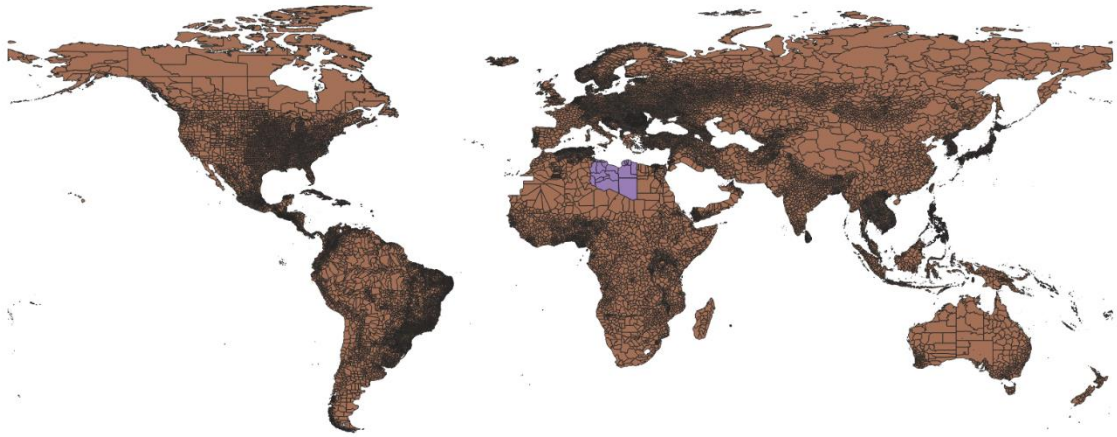
Dado que Qgis es un sistema de información geográfica de software libre que trabaja fundamentalmente con capas, descargamos un archivo ‘shape’ donde se nos facilitaban capas de todo el mapa con distintos niveles de desagregación, desde países hasta distritos como se muestra a continuación



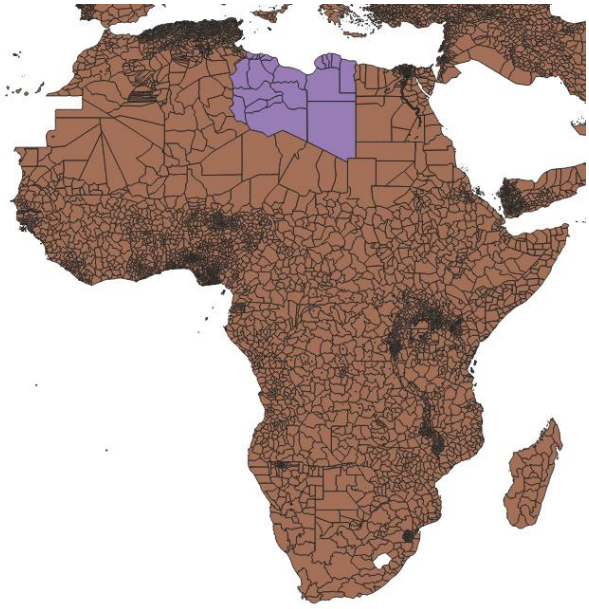
A continuación, mostramos una capa donde aparece la segmentación polinómica por provincias



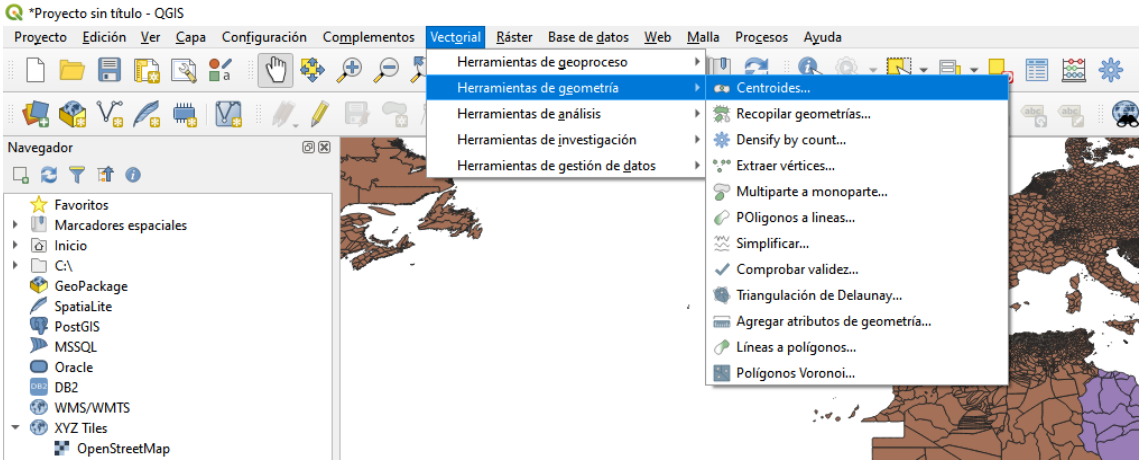
Pero como queremos el máximo nivel de precisión, activamos la capa más desagregada



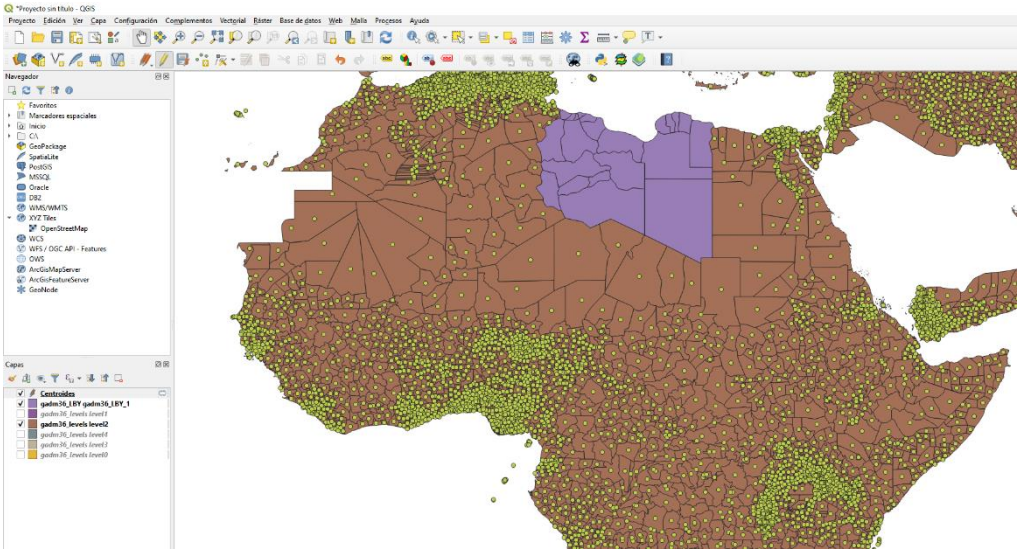
Particularizando para el continente africano



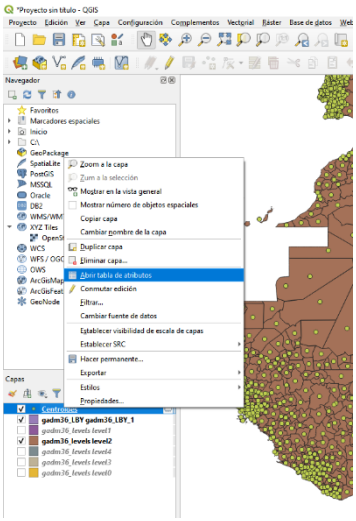
Ahora, nuestro problema está en que necesitamos una referencia geográfica de cada uno de esos polígonos, para ello lo que haremos es dibujar un centroide en cada uno de ellos.



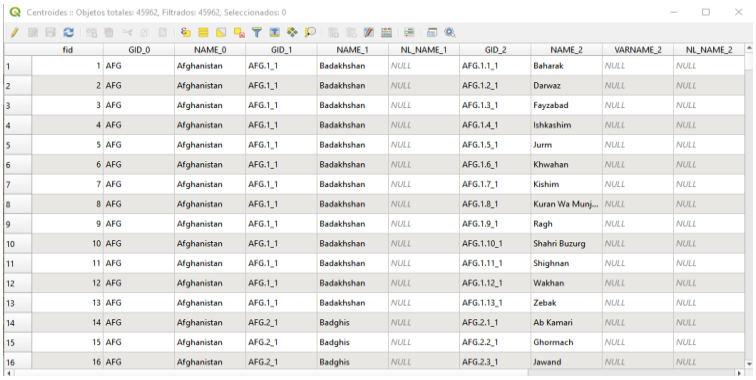
Automáticamente se nos dibujan estos centroides y se nos genera una nueva capa con el mismo nombre.



Podemos remitirnos a esta capa y pedirle un listado de atributos que automáticamente se genera al mismo tiempo que se dibujan los centroides.



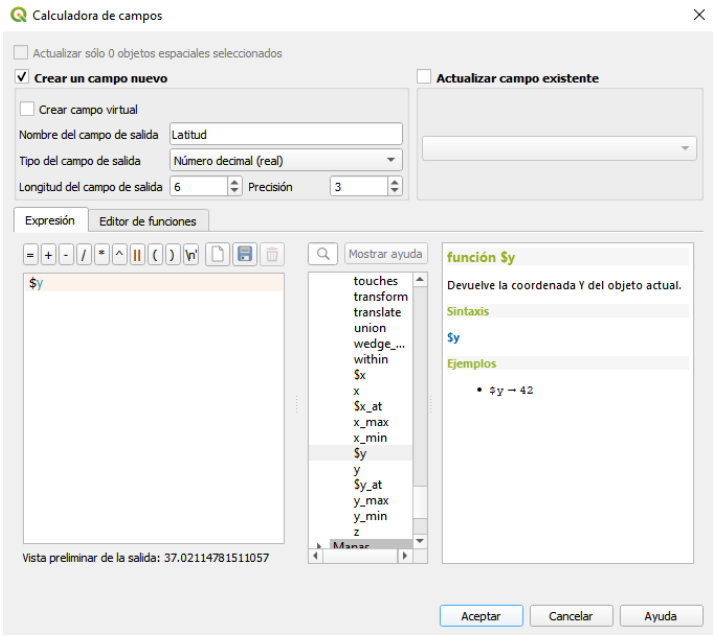
Y automáticamente se nos abre un cuadro de datos con casi toda la información que necesitamos.



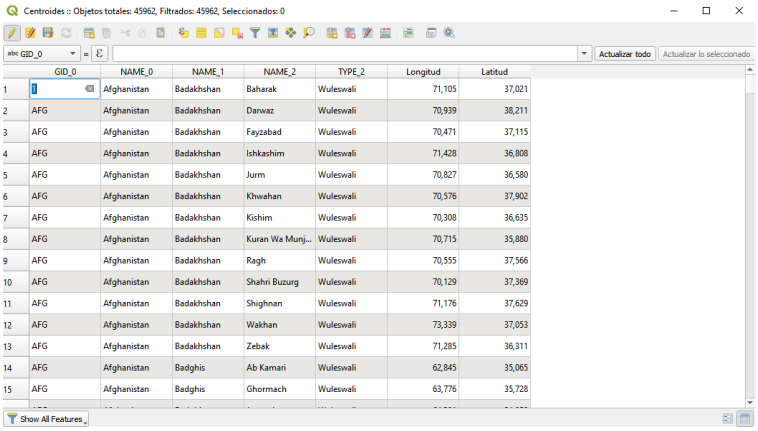
	fid	GID_0	NAME_0	GID_1	NAME_1	NL_NAME_1	GID_2	NAME_2	VARNAME_2	NL_NAME_2
1	1	AFG	Afghanistan	AFG.1_1	Badakhshan	NULL	AFG.1.1_1	Baharak	NULL	NULL
2	2	AFG	Afghanistan	AFG.1_1	Badakhshan	NULL	AFG.1.2_1	Darwaz	NULL	NULL
3	3	AFG	Afghanistan	AFG.1_1	Badakhshan	NULL	AFG.1.3_1	Fayzabad	NULL	NULL
4	4	AFG	Afghanistan	AFG.1_1	Badakhshan	NULL	AFG.1.4_1	Ishkashim	NULL	NULL
5	5	AFG	Afghanistan	AFG.1_1	Badakhshan	NULL	AFG.1.5_1	Jurm	NULL	NULL
6	6	AFG	Afghanistan	AFG.1_1	Badakhshan	NULL	AFG.1.6_1	Khwahan	NULL	NULL
7	7	AFG	Afghanistan	AFG.1_1	Badakhshan	NULL	AFG.1.7_1	Kishim	NULL	NULL
8	8	AFG	Afghanistan	AFG.1_1	Badakhshan	NULL	AFG.1.8_1	Kuran Wa Munj...	NULL	NULL
9	9	AFG	Afghanistan	AFG.1_1	Badakhshan	NULL	AFG.1.9_1	Ragh	NULL	NULL
10	10	AFG	Afghanistan	AFG.1_1	Badakhshan	NULL	AFG.1.10_1	Shahri Buzurg	NULL	NULL
11	11	AFG	Afghanistan	AFG.1_1	Badakhshan	NULL	AFG.1.11_1	Shighnan	NULL	NULL
12	12	AFG	Afghanistan	AFG.1_1	Badakhshan	NULL	AFG.1.12_1	Wakhan	NULL	NULL
13	13	AFG	Afghanistan	AFG.1_1	Badakhshan	NULL	AFG.1.13_1	Zebak	NULL	NULL
14	14	AFG	Afghanistan	AFG.2_1	Badghis	NULL	AFG.2.1_1	Ab Kamari	NULL	NULL
15	15	AFG	Afghanistan	AFG.2_1	Badghis	NULL	AFG.2.2_1	Ghormach	NULL	NULL
16	16	AFG	Afghanistan	AFG.2_1	Badghis	NULL	AFG.2.3_1	Jawand	NULL	NULL

Nos falta algo más, concretamente la georreferenciación de esos centroides que será lo que nos permita hacer la unión con las otras tablas de datos georreferenciadas que tenemos.

Esas coordenadas las generamos a través de la propia calculadora de Qgis como se muestra a continuación:



Y ahora tenemos como resultado la misma tabla con toda la información que necesitamos



	GID_0	NAME_0	NAME_1	NAME_2	TYPE_2	Longitud	Latitud
1	1	Afghanistan	Badakhshan	Baharak	Wuleswali	71,105	37,021
2	AFG	Afghanistan	Badakhshan	Darwaz	Wuleswali	70,939	38,211
3	AFG	Afghanistan	Badakhshan	Fayzabad	Wuleswali	70,471	37,115
4	AFG	Afghanistan	Badakhshan	Ishkashim	Wuleswali	71,428	36,808
5	AFG	Afghanistan	Badakhshan	Jurm	Wuleswali	70,827	36,580
6	AFG	Afghanistan	Badakhshan	Khwahan	Wuleswali	70,576	37,902
7	AFG	Afghanistan	Badakhshan	Kishim	Wuleswali	70,308	36,635
8	AFG	Afghanistan	Badakhshan	Kuran Wa Munj...	Wuleswali	70,715	35,880
9	AFG	Afghanistan	Badakhshan	Ragh	Wuleswali	70,555	37,566
10	AFG	Afghanistan	Badakhshan	Shahri Buzurg	Wuleswali	70,129	37,369
11	AFG	Afghanistan	Badakhshan	Shighnan	Wuleswali	71,176	37,629
12	AFG	Afghanistan	Badakhshan	Wakhan	Wuleswali	73,339	37,053
13	AFG	Afghanistan	Badakhshan	Zebak	Wuleswali	71,285	36,311
14	AFG	Afghanistan	Badghis	Ab Kamari	Wuleswali	62,845	35,065
15	AFG	Afghanistan	Badghis	Ghormach	Wuleswali	63,776	35,728

Finalmente guardamos esta tabla como un archivo (en nuestro caso y por comodidad para posteriormente abrir en Python con extensión .csv) para hacer la unión con la tabla donde teníamos la información de las alturas.

```
import pandas as pd
import numpy as np
```

```
df = pd.read_csv('mundo_distritos_georref.csv')
df
```

	NAME_0	NAME_1	NAME_2	Longitud	Latitud
0	Afghanistan	Badakhshan	Baharak	71.105	37.021
1	Afghanistan	Badakhshan	Darwaz	70.939	38.211
2	Afghanistan	Badakhshan	Fayzabad	70.471	37.115
3	Afghanistan	Badakhshan	Ishkashim	71.428	36.808
4	Afghanistan	Badakhshan	Jurm	70.827	36.580
...	...	...	...	...	...
150298	Zimbabwe	Midlands	Gweru	29.650	-19.460
150299	Zimbabwe	Midlands	Kwekwe	29.561	-18.874
150300	Zimbabwe	Midlands	Mberengwa	30.016	-20.750
150301	Zimbabwe	Midlands	Shurugwi	30.147	-19.750
150302	Zimbabwe	Midlands	Zvishavane	30.078	-20.284

150303 rows × 5 columns

```
df = df.rename(columns={'NAME_0': 'Country',
                        'NAME_1': 'Province',
                        'NAME_2': 'Distrito'})
```

Teniendo en cuenta que las coordenadas de la tabla donde se encuentran las alturas están redondeadas a 0.25 y 0.75 aplicamos previamente una función de redondeo para finalmente hacer la unión.

```
#Aplicamos la funcion redondeo

import math

def redondear(numero):
    decimal, entero  = math.modf(numero)
    decimal = round(decimal,2)
    if decimal > 0.5:
        decimal = 0.75
    else:
        decimal = 0.25
    return entero+decimal

df['Latitud']=df['Latitud'].apply(redondear)
df['Longitud']=df['Longitud'].apply(redondear)
```

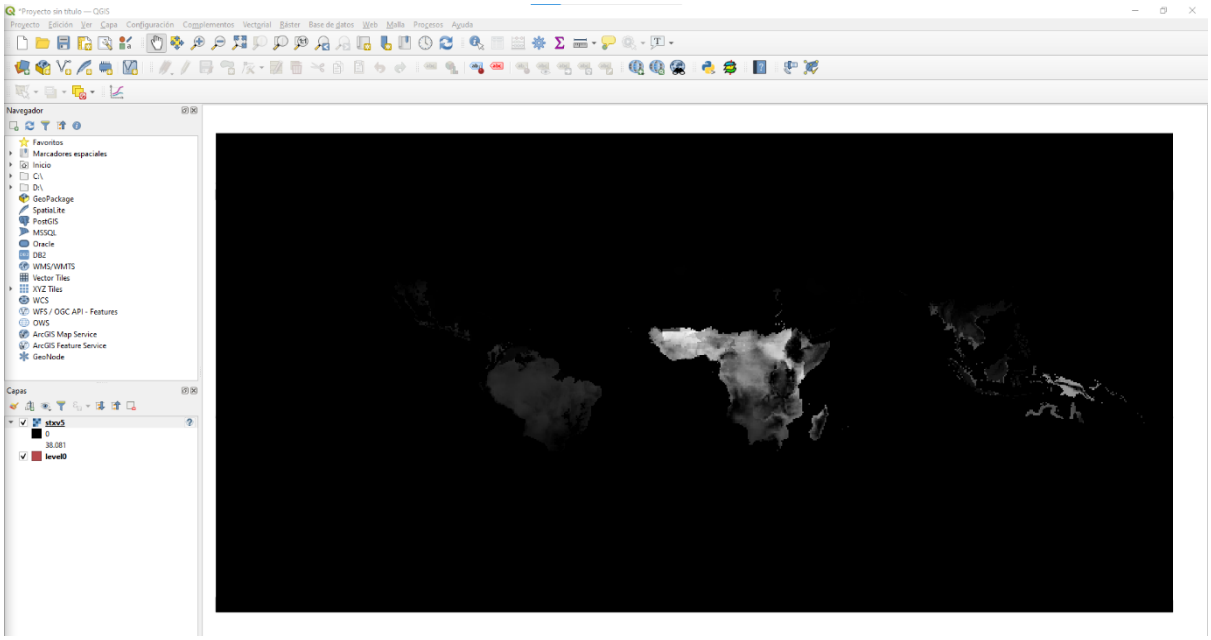
```
df
```

	Country	Province	Distrito	Longitud	Latitud
0	Afghanistan	Badakhshan	Baharak	71.25	37.25
1	Afghanistan	Badakhshan	Darwaz	70.75	38.25
2	Afghanistan	Badakhshan	Fayzabad	70.25	37.25
3	Afghanistan	Badakhshan	Ishkashim	71.25	36.75
4	Afghanistan	Badakhshan	Jurm	70.75	36.75
...	...	...	...	...	...
150298	Zimbabwe	Midlands	Gweru	29.75	-18.75

Anexo 2 – Obtención datos de Malaria

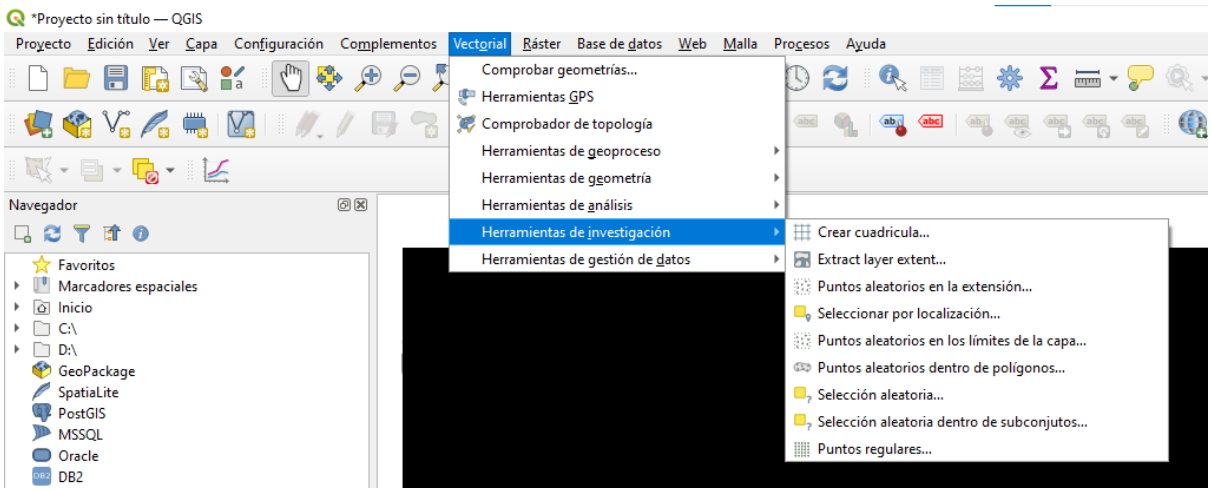
En su forma más simple, un **ráster** consta de una matriz de celdas (o píxeles) organizadas en filas y columnas (o una cuadrícula) en la que cada celda contiene un valor que representa información, en este caso cada una de esas celdas guarda información del Malaria Ecology Index. Para trabajar con imágenes ráster nos apoyamos en Qgis (el programa de georreferenciación de código abierto que utilizamos para dar nombres a cada uno de los clústers en las tablas de con las que trabajamos a lo largo de toda la investigación. En este apéndice voy a contar paso a paso cómo he obtenido la información a partir de una capa ráster.

Empezamos cargando el archivo en QGIS

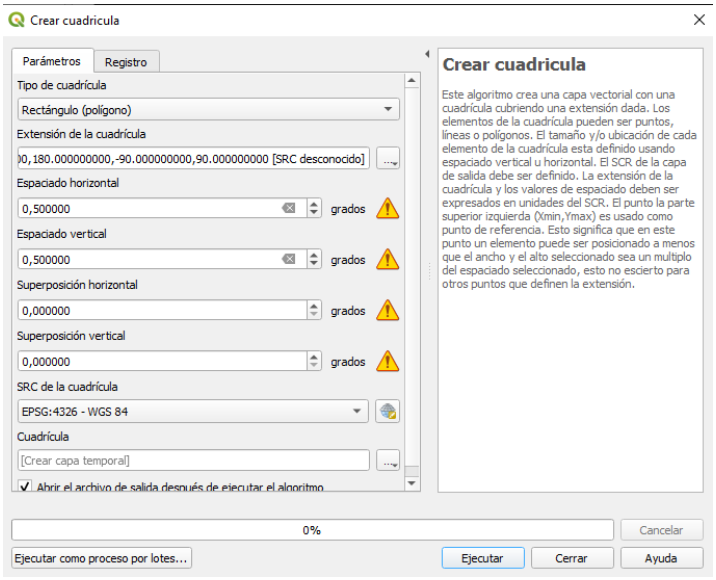


Como puede apreciarse es una imagen con poco detalle en escala de grises.

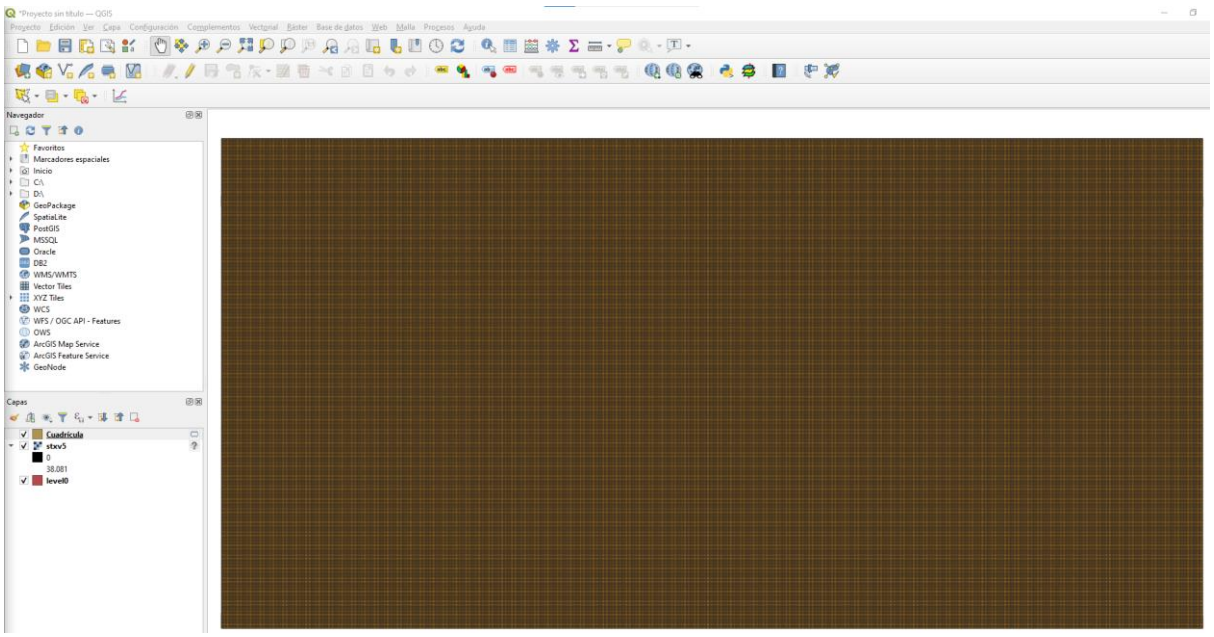
A continuación, queremos obtener para clúster de 0.5*0.5 que es el nivel de detalle que tenemos en el resto de información. Para esto, crearemos una malla artificial de dimensión 0.5*0.5.



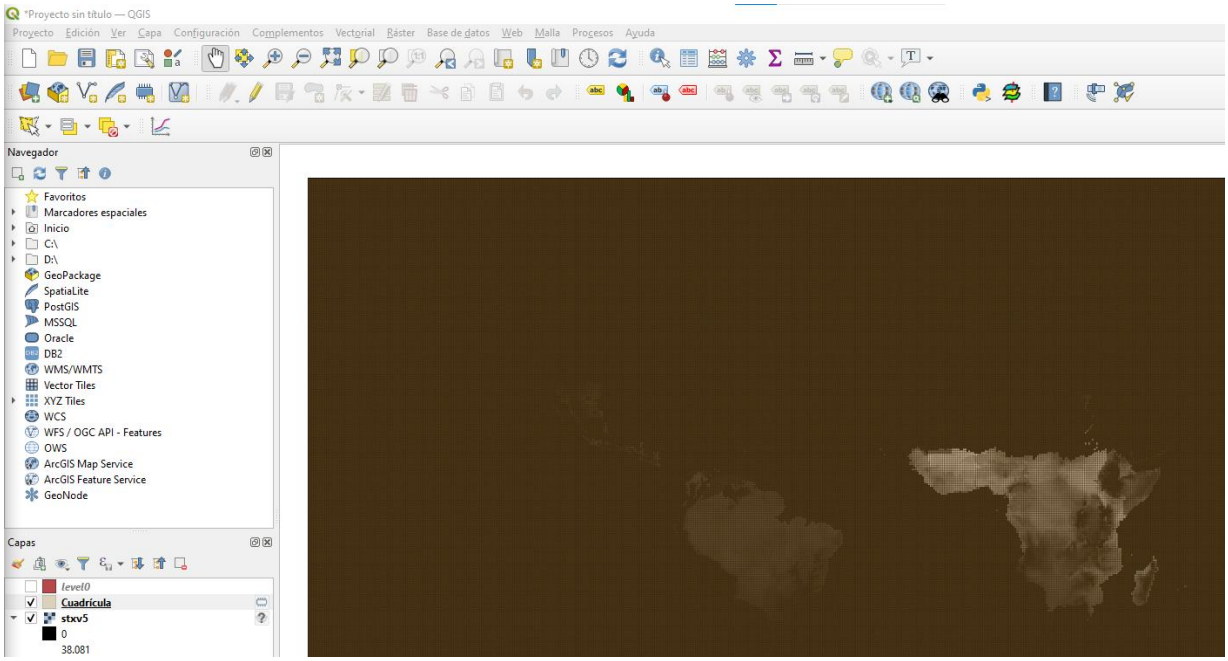
Aparecerá un cuadro de diálogo donde señalaremos sobre que capa queremos construir la malla y la dimensión del cuadro como se muestra en la siguiente figura.

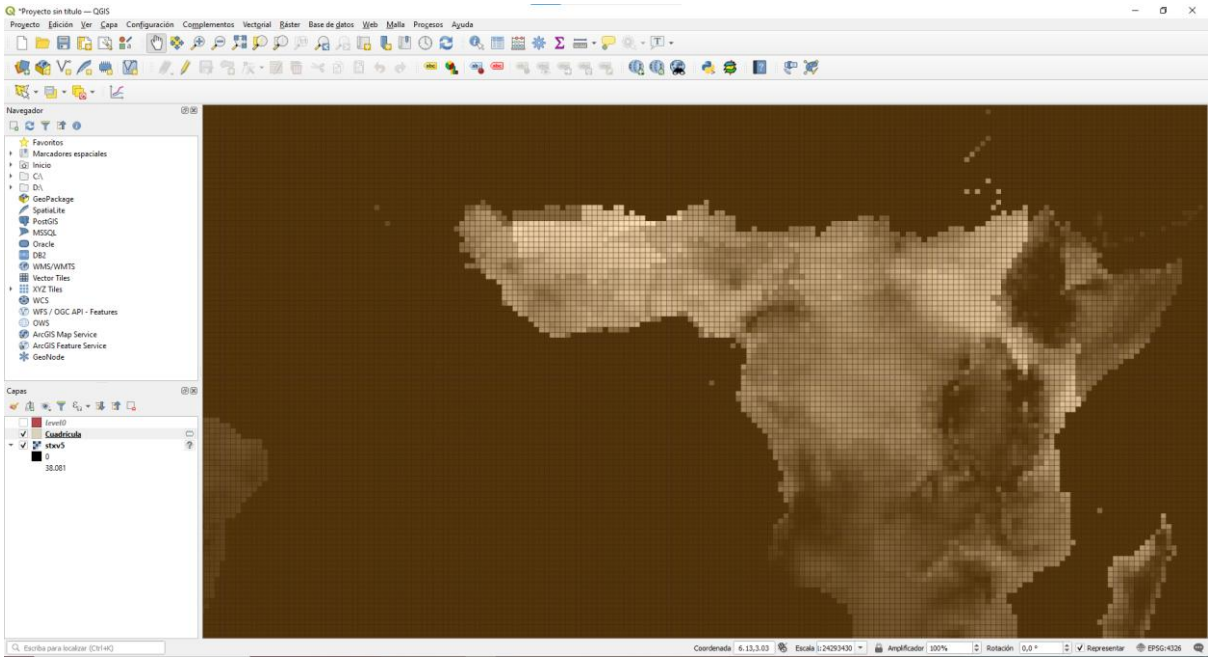


Nos aparecerá algo con la forma

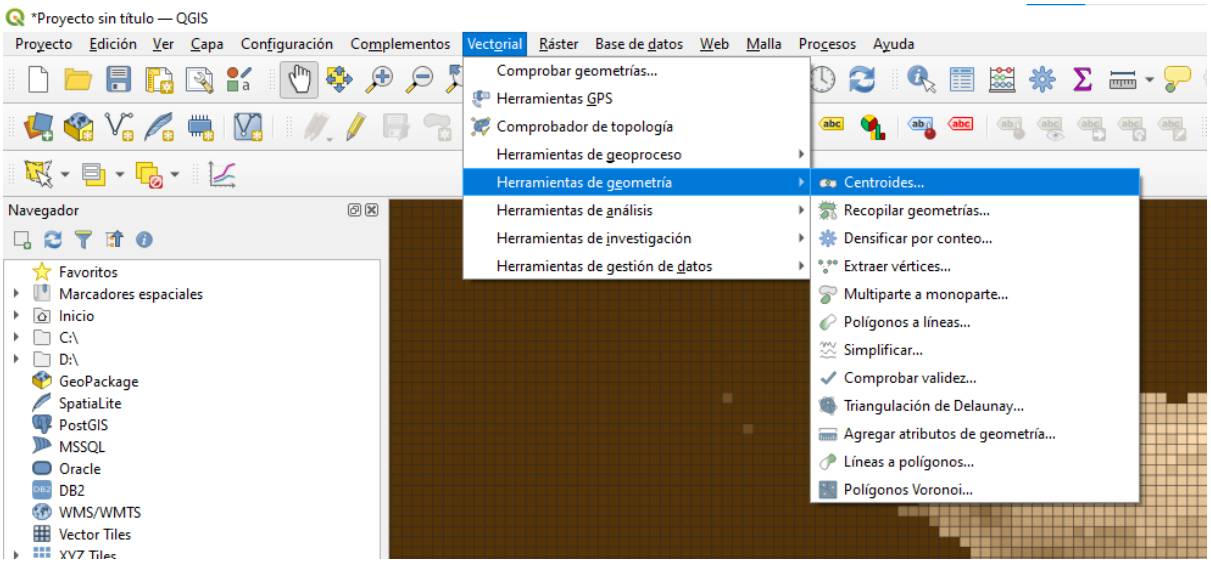


Como parece que no tiene mucho sentido lo que hemos creado le daremos un poco de transparencia y superpondremos la capa ráster para que se entienda lo que hemos conseguido.

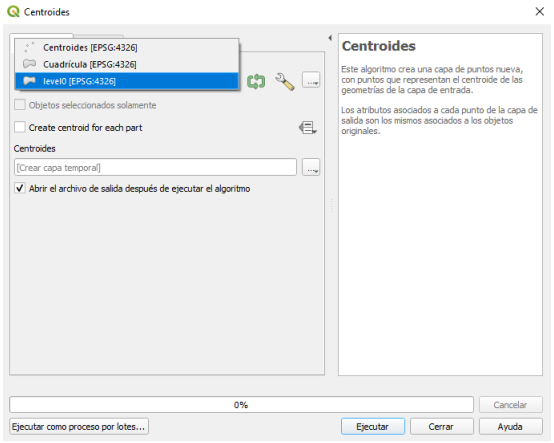




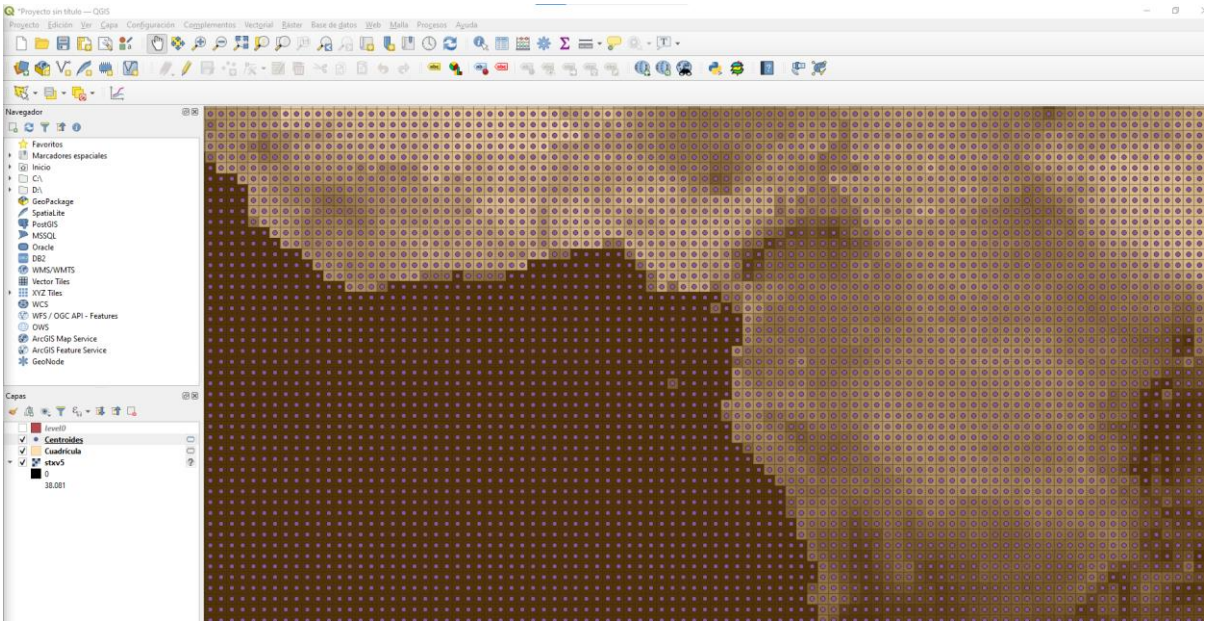
El siguiente paso será crear un punto dentro de cada una de esas celdas para posteriormente pedirle información a la capa ráster en ese punto concreto. De modo que lo más razonable es generar un centroide, un punto en el centro exacto de cada una de esas celdas.



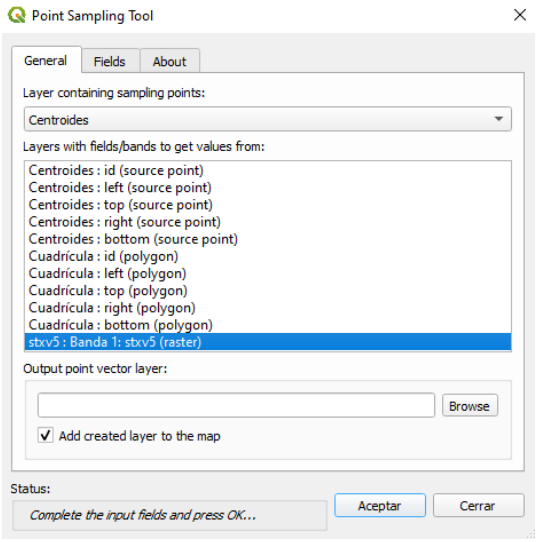
Volverá a aparecer un cuadro de diálogo donde especificamos la capa sobre la que queremos generar los centroides.



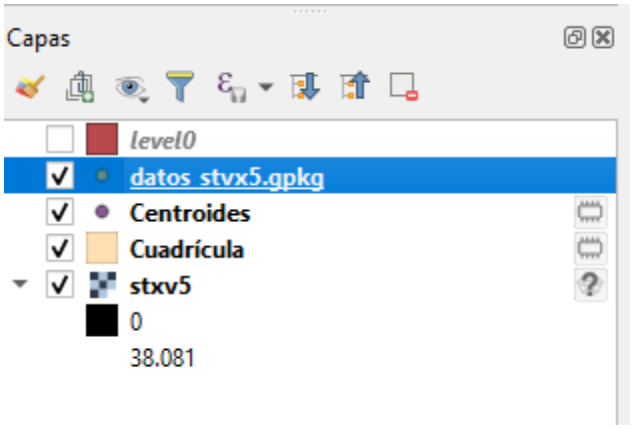
El resultado será



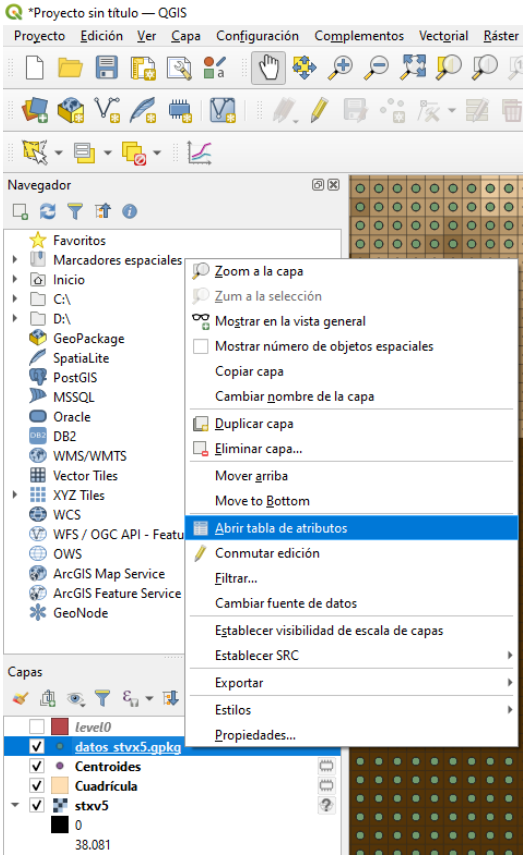
Ahora viene la parte más delicada que es la de pedirle la información que guarda la capa ráster en cada uno de esos puntos que hemos creado, para ello, primero seleccionamos la capa de donde queremos la información



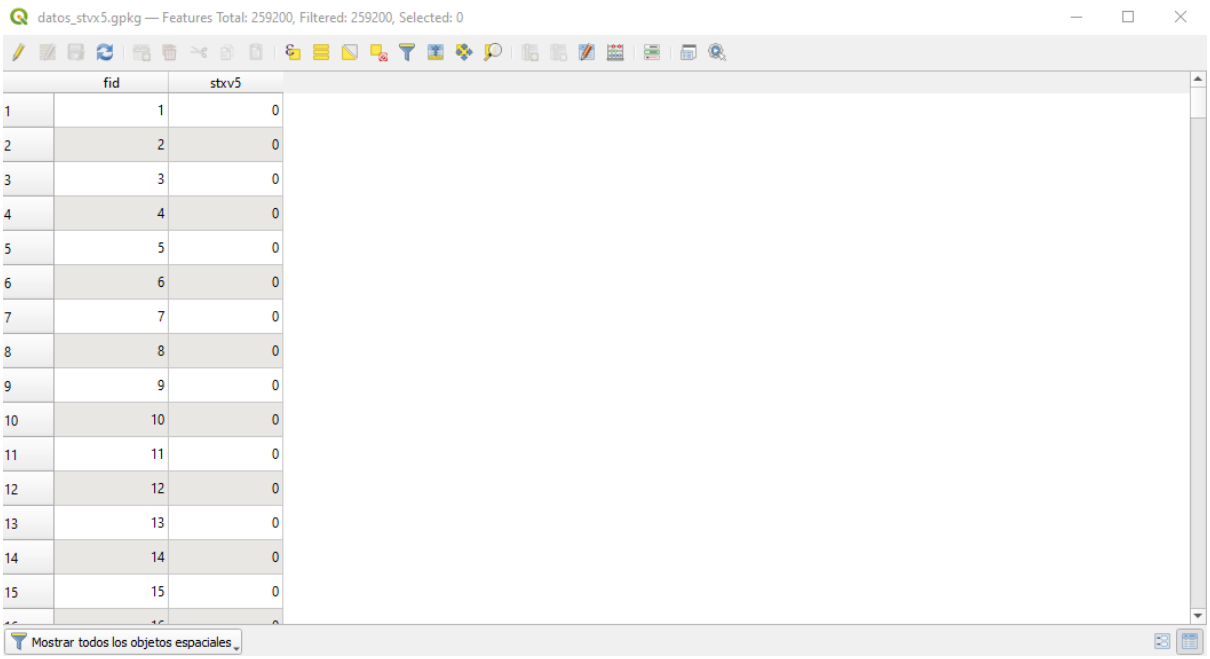
Se observa que se genera una nueva capa con un nombre que nosotros mismos hemos asignado 'datos_stvx5'.



Seleccionamos la capa y con el botón derecho pedimos que nos muestre la tabla de atributos donde debería aparecer la información que estamos buscando.

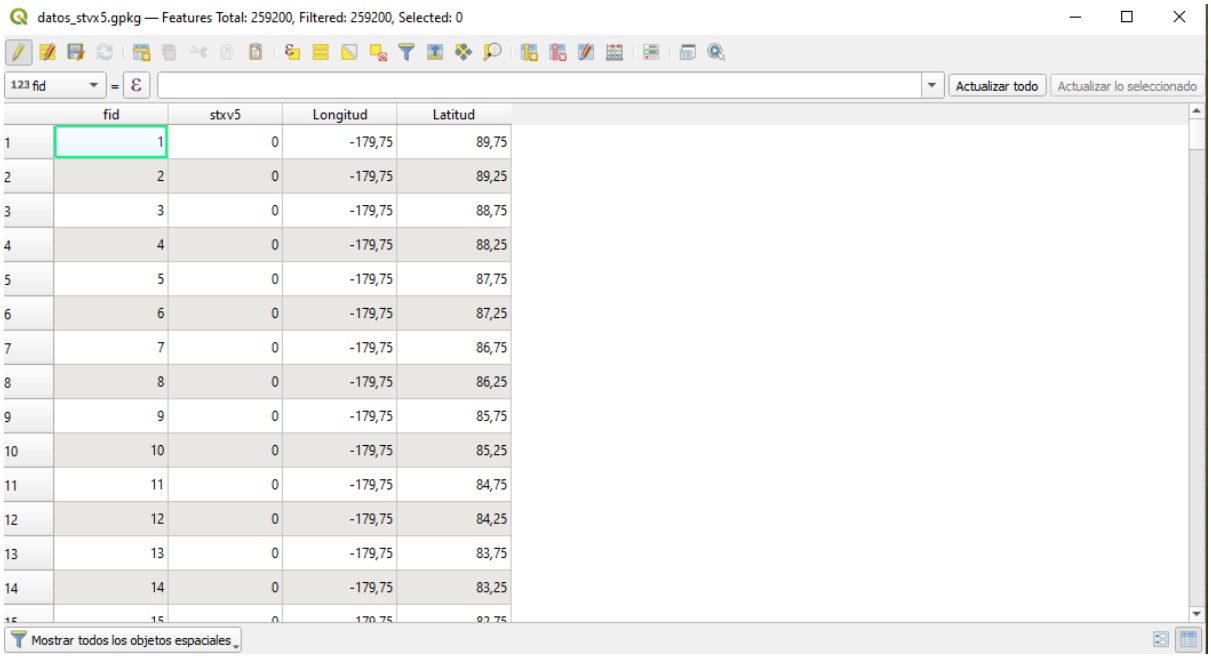


Y el resultado será algo como



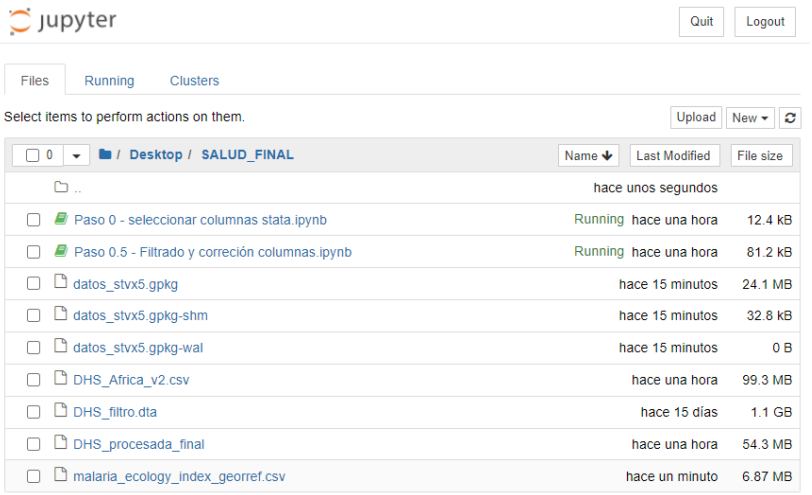
	fid	stxv5
1	1	0
2	2	0
3	3	0
4	4	0
5	5	0
6	6	0
7	7	0
8	8	0
9	9	0
10	10	0
11	11	0
12	12	0
13	13	0
14	14	0
15	15	0

Sólo nos queda un paso más que es el de identificar geográficamente cada uno de esos puntos con una longitud y latitud que nos permita a la postre unirlo con otras tablas. Para ello haremos uso de la calculadora de QGIS.



	fid	stxv5	Longitud	Latitud
1	1	0	-179,75	89,75
2	2	0	-179,75	89,25
3	3	0	-179,75	88,75
4	4	0	-179,75	88,25
5	5	0	-179,75	87,75
6	6	0	-179,75	87,25
7	7	0	-179,75	86,75
8	8	0	-179,75	86,25
9	9	0	-179,75	85,75
10	10	0	-179,75	85,25
11	11	0	-179,75	84,75
12	12	0	-179,75	84,25
13	13	0	-179,75	83,75
14	14	0	-179,75	83,25

Finalmente lo exportamos como un archivo .csv para poderlo abrir en python y generar los merge con otras tablas.



Y efectivamente comprobamos que lo podemos abrir en nuestro notebook de python.

```
1 import pandas as pd
```

```
1 df = pd.read_csv('malaria_ecology_index_georref.csv').drop('fid', axis=1)
```

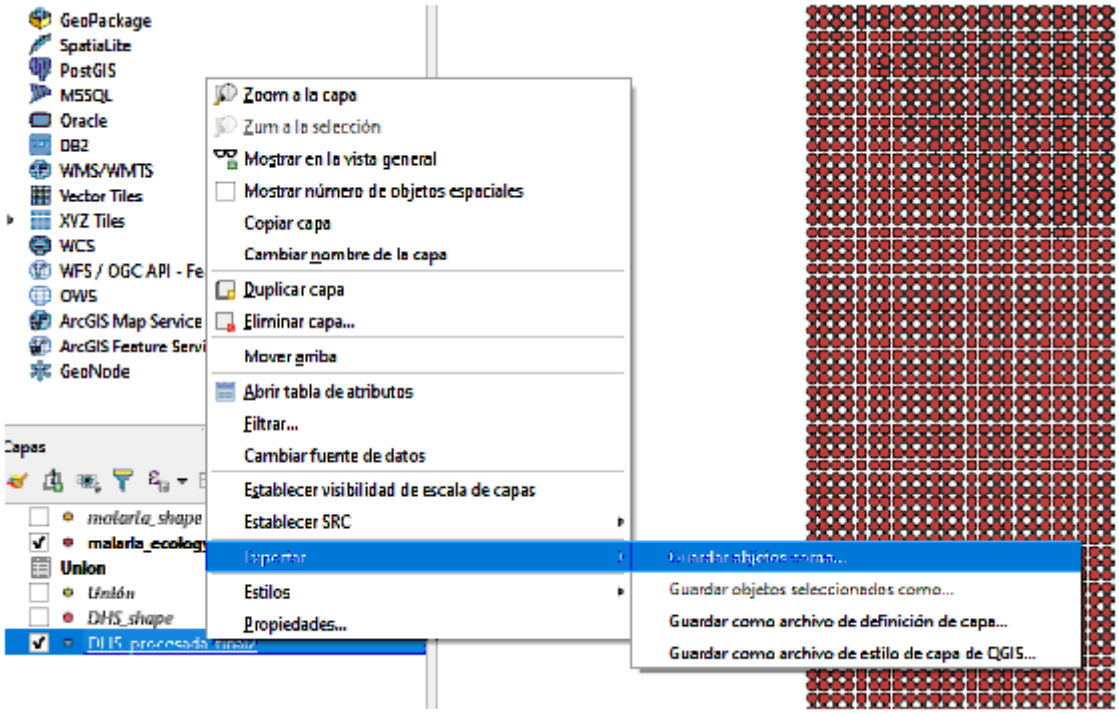
```
1 df
```

	stxv5	Longitud	Latitud
0	0.0	-179.75	89.75
1	0.0	-179.75	89.25
2	0.0	-179.75	88.75

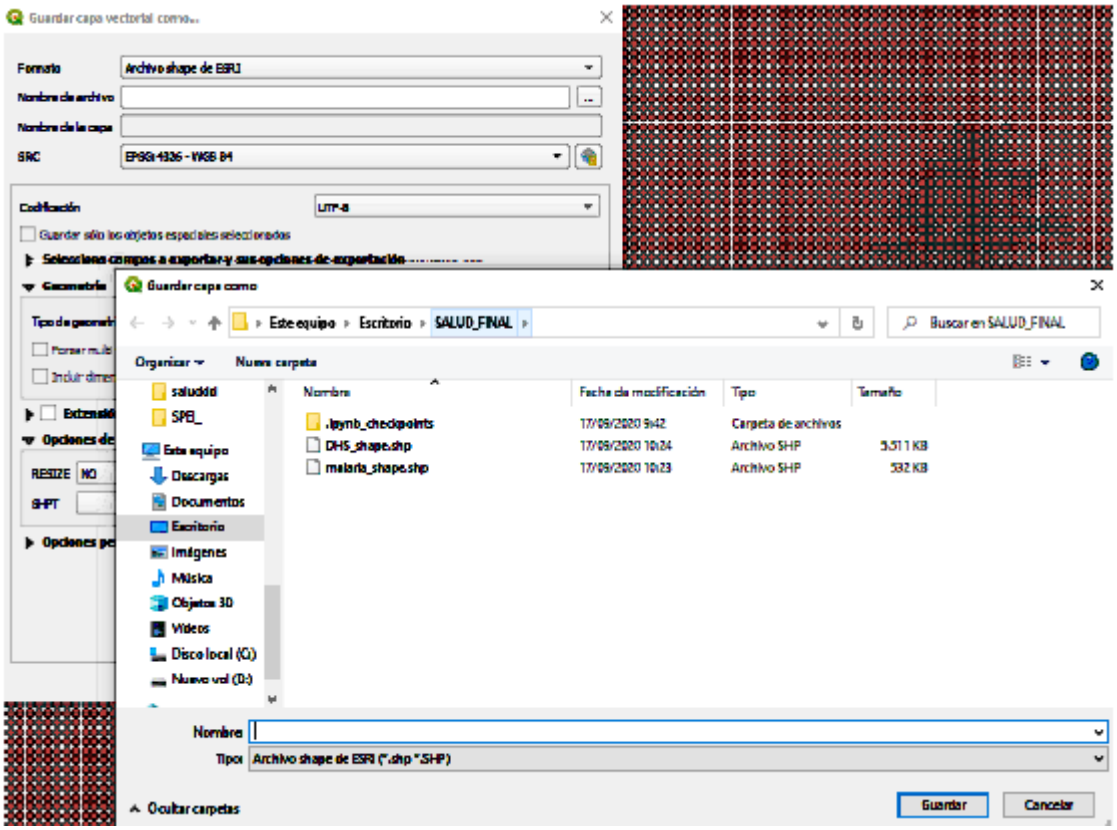
Estamos en un punto donde tenemos dos tablas con los datos que nos interesa para lanzar el modelo de salud infantil, pero para ello debemos hacer un merge perfecto de las

tablas. Una opción es a través de python y su librería 'pandas' y otra es a través de QGIS. A continuación, vamos a explicar este último que es algo más complejo, pero nos hace un merge con total precisión.

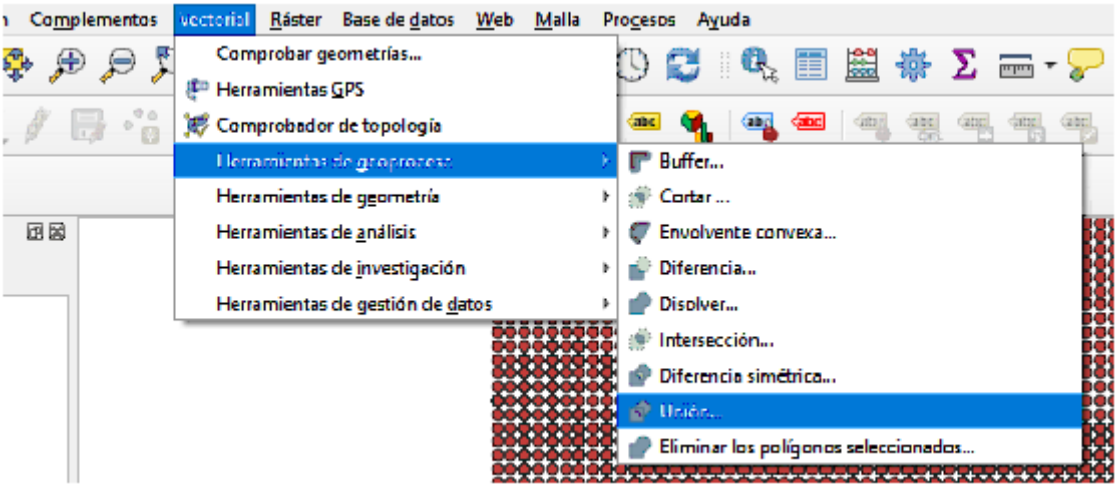
Empezamos abriendo en QGIS los csv que queremos unir, y creamos sobre ellas un archivo shape.



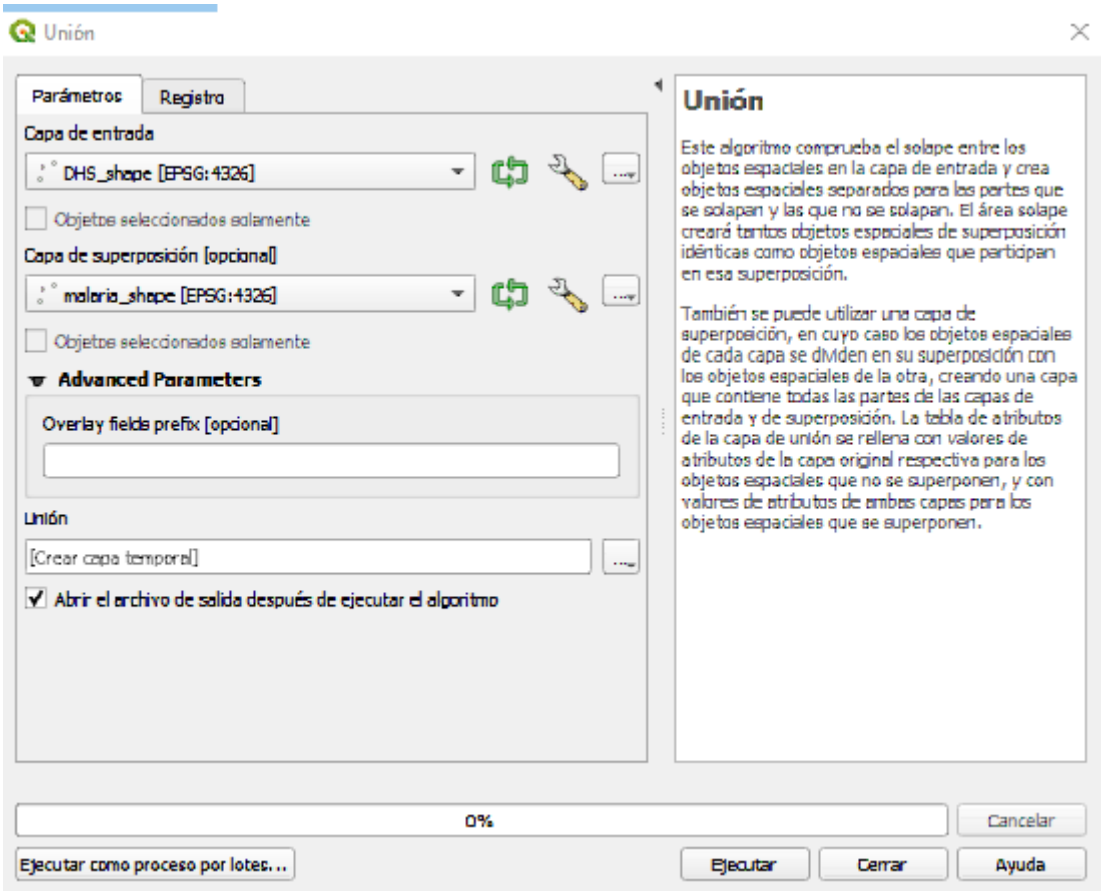
Y después de haber seleccionado como formato 'Archivo shape de ESRI', lo guardamos convenientemente.



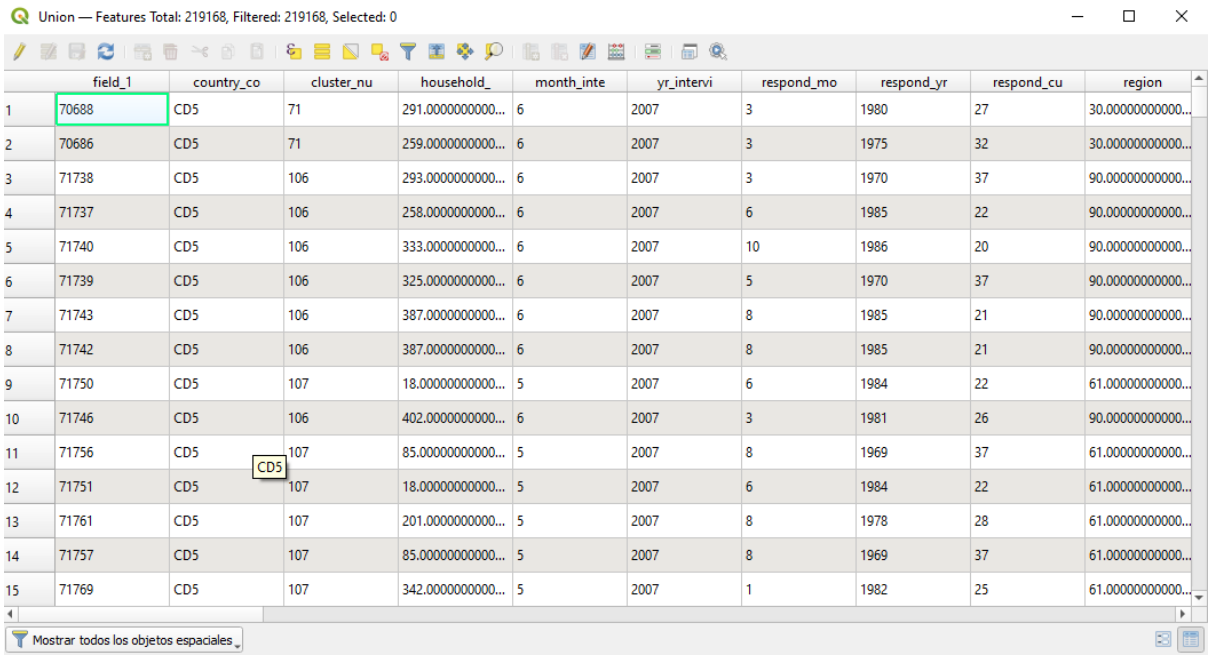
Este proceso lo hacemos para cada una de las tablas que queremos unir. Finalmente tenemos que hacer la unión y para ello nos remitimos al menú vectorial.



Nos aparecerá un cuadro de diálogo donde tendremos que explicar qué capas queremos unir y ejecutamos.



El proceso terminará correctamente si en el cuadro de capas aparece una nueva con el nombre de 'unión'. Para asegurarnos, abrimos la tabla de atributos de esta nueva capa que se ha creado y vemos que están todas las variables de las dos tablas.



	field_1	country_co	cluster_nu	household_	month_inte	yr_intervi	respond_mo	respond_yr	respond_cu	region
1	70688	CD5	71	291.0000000000...	6	2007	3	1980	27	30.0000000000...
2	70686	CD5	71	259.0000000000...	6	2007	3	1975	32	30.0000000000...
3	71738	CD5	106	293.0000000000...	6	2007	3	1970	37	90.0000000000...
4	71737	CD5	106	258.0000000000...	6	2007	6	1985	22	90.0000000000...
5	71740	CD5	106	333.0000000000...	6	2007	10	1986	20	90.0000000000...
6	71739	CD5	106	325.0000000000...	6	2007	5	1970	37	90.0000000000...
7	71743	CD5	106	387.0000000000...	6	2007	8	1985	21	90.0000000000...
8	71742	CD5	106	387.0000000000...	6	2007	8	1985	21	90.0000000000...
9	71750	CD5	107	18.0000000000...	5	2007	6	1984	22	61.0000000000...
10	71746	CD5	106	402.0000000000...	6	2007	3	1981	26	90.0000000000...
11	71756	CD5	107	85.0000000000...	5	2007	8	1969	37	61.0000000000...
12	71751	CD5	107	18.0000000000...	5	2007	6	1984	22	61.0000000000...
13	71761	CD5	107	201.0000000000...	5	2007	8	1978	28	61.0000000000...
14	71757	CD5	107	85.0000000000...	5	2007	8	1969	37	61.0000000000...
15	71769	CD5	107	342.0000000000...	5	2007	1	1982	25	61.0000000000...

Anexo 3 – Generación Shocks de Malaria

Aquí mostramos el código para generar los shocks de malaria. Se trata de un proceso paralelizado porque lanzándolo para procesar de forma normal se extendía a casi 5 días. Este proceso de paralelización lo utilizamos constantemente en el proyecto porque pasamos de trabajar con datos de un solo país a trabajar con los datos para toda África.

```
import pandas as pd
import numpy as np
import copy
from sklearn.utils import shuffle
#from Contador import Contador
```

```
df = pd.read_csv('output.csv').drop(['Unnamed: 0'], axis=1)
```

```
df.head()
```

	Longitud	Latitud	Year	Mes	Prep	Temp	Media_p	Desviacion_p	Media_t	Desviacion_t	...	STP-+ (1_2)	STP-- (1_2)	STP++ (2_1)	STP+- (2_1)	STP-+ (2_1)	STP-- (2_1)	STP+
0	5.25	9.25	1966.0	Julio	185.7	26.7	191.827586	74.604853	26.403448	0.369979	...	0.0	0.0	0.0	0.0	0.0	0.0	0
1	5.25	6.75	2001.0	Junio	184.2	26.2	230.396667	51.168620	26.330000	0.333816	...	0.0	0.0	0.0	0.0	0.0	0.0	0
2	7.75	6.75	1917.0	Septiembre	256.0	25.5	320.188235	61.340380	25.758824	0.574004	...	0.0	0.0	0.0	0.0	0.0	0.0	0
3	4.75	9.25	1933.0	Septiembre	311.4	26.6	260.660000	61.709997	26.666667	0.489444	...	0.0	0.0	0.0	0.0	0.0	0.0	0
4	14.75	12.25	1999.0	Enero	0.0	23.9	0.017241	0.091233	22.846667	1.535520	...	0.0	0.0	0.0	0.0	0.0	0.0	0

5 rows × 52 columns

```
df2 = df[['Longitud', 'Latitud', 'Year', 'Mes', 'Prep', 'Temp']]
```

```
df2.head(10)
```

	Longitud	Latitud	Year	Mes	Prep	Temp
0	5.25	9.25	1966.0	Julio	185.7	26.7
1	5.25	6.75	2001.0	Junio	184.2	26.2
2	7.75	6.75	1917.0	Septiembre	256.0	25.5
3	4.75	9.25	1933.0	Septiembre	311.4	26.6
4	14.75	12.25	1999.0	Enero	0.0	23.9
5	7.25	5.25	1992.0	Mayo	150.8	28.0
6	9.25	6.75	1951.0	Mayo	329.5	26.8
7	13.75	11.25	1956.0	Octubre	32.5	25.9
8	4.75	8.25	2002.0	Mayo	114.9	27.3
9	10.75	6.25	1983.0	Marzo	15.1	19.5

```
df2.shape
```

```
(567216, 6)
```

```python
def get_value_1(lat,lon,mes,anio):
    return df2.loc[(df2['Latitud']==lat) & (df2['Longitud']==lon) & (df2['Mes']==mes) & (df2['Year']==anio)]['Temp']
```

```python
def get_value_2(lat,lon,mes,anio):
    return df2.loc[(df2['Latitud']==lat) & (df2['Longitud']==lon) & (df2['Mes']==mes) & (df2['Year']==anio)]['Prep']
```

```python
def last_n_months_1(year, month, lat, lon, n, df=df2, default=0):
    meses_anios = ()
    mes_actual = month
    anio_actual = year
    correlacion = {
        'Enero': 'Diciembre',
        'Febrero': 'Enero',
        'Marzo': 'Febrero',
        'Abril': 'Marzo',
        'Mayo': 'Abril',
        'Junio': 'Mayo',
        'Julio': 'Junio',
        'Agosto': 'Julio',
        'Septiembre': 'Agosto',
        'Octubre': 'Septiembre',
        'Noviembre': 'Octubre',
        'Diciembre': 'Noviembre'
    }
    contenedor = []
    for i in range(n):
        mes_actual = correlacion[mes_actual]
        if (mes_actual == 'Diciembre'):
            anio_actual -= 1

        try:
            contenedor.append(get_value_1(lat, lon, mes_actual,anio_actual).values[0])
        except IndexError:
            contenedor.append(default)

    return contenedor
```

```python
def last_n_months_2(year, month, lat, lon, n, df=df2, default=0):
    meses_anios = ()
    mes_actual = month
    anio_actual = year
    correlacion = {
        'Enero': 'Diciembre',
        'Febrero': 'Enero',
        'Marzo': 'Febrero',
        'Abril': 'Marzo',
        'Mayo': 'Abril',
        'Junio': 'Mayo',
        'Julio': 'Junio',
        'Agosto': 'Julio',
        'Septiembre': 'Agosto',
        'Octubre': 'Septiembre',
        'Noviembre': 'Octubre',
        'Diciembre': 'Noviembre'
    }
    tupla = []
    for i in range(n):
        mes_actual = correlacion[mes_actual]
        if (mes_actual == 'Diciembre'):
            anio_actual -= 1

        try:
            tupla.append(get_value_2(lat, lon, mes_actual,anio_actual).values[0])
        except IndexError:
            tupla.append(default)

    return tupla
```

```python
## La precipitación mensual promedio durante los últimos 3 meses es de al menos 60 mm.

def prec_promedio_3_meses(row,cant=60,default=60):
    valores = last_n_months_2(row['Year'],row['Mes'],row['Latitud'],row['Longitud'],3,default=default)
    import numpy as np

    aux = 0
    for v in valores:
        if v >= cant:
            aux += 1

    if aux == 3:
        return True
    else:
        return False
```

```python
## La precipitación en al menos uno de los últimos 3 meses es de al menos 80 mm.

def prec_uno_3_meses(row, cant=80,default=0):
    cant = float(cant)
    valores = last_n_months_2(row['Year'], row['Mes'], row['Latitud'],
                              row['Longitud'], 3, default=default)
    import numpy as np
    if float(np.amax(valores)) >= cant:
        return True
    else:
        return False
```

```python
## Ningún mes en los últimos 12 meses tiene una temperatura promedio inferior a 5ºC

def temp_none_12_meses_menor(row, cant=5):
    valores = last_n_months_1(row['Year'], row['Mes'], row['Latitud'],
                              row['Longitud'], 12)
    import numpy as np
    if np.amin(valores) >= cant:
        return True
    else:
        False
```

```python
## La temperatura promedio en los últimos 3 meses supera los 19.5ºC + SD (temperatura mensual en los últimos 12 meses).
cont = []
def temp_3_meses_sd(row, cant=19.5,default=19.5):
    valores = last_n_months_1(row['Year'], row['Mes'], row['Latitud'],
                              row['Longitud'], 3)
    cont.append(valores)

    import numpy as np
    sd = np.std(last_n_months_1(row['Year'], row['Mes'], row['Latitud'], row['Longitud'],12, False,default=default))

    med = np.mean(cont)
    if (med >= cant + sd):
        return True
    else:
        return False
```

```python
def aux_calculo(df_split, n, lis):
    #le = len(df_split)
    #con = Contador(f'paral {n}', le, print_color=True, color_name='rand')

    for index, row in df_split.iterrows():
        aux_1 = prec_promedio_3_meses(row,cant=60,default=60)
        aux_2 = prec_uno_3_meses(row, cant=80,default=0)
        aux_3 = temp_none_12_meses_menor(row, cant=5)
        aux_4 = temp_3_meses_sd(row, cant=19.5,default=19.5)

        df_split.at[index, 'msm1'] = aux_1
        df_split.at[index, 'msm2'] = aux_2
        df_split.at[index, 'msm3'] = aux_3
        df_split.at[index, 'msm4'] = aux_4


        #con.inc()
    #print("ID: ", n, len(df_split))
    lis[n]=df_split
```

Como se trata de proceso paralelizado la otra parte del código se escribe en un notebook diferente. Lo que se hace en esta parte es llamar al dataframe objetivo (sobre el que vamos a coger los datos para hacer los cálculos) e indicarle en cuántas partes lo vamos a fraccionar para que cada uno de los procesadores lógicos de nuestra máquina trabaje de forma independiente sobre cada una de las particiones del dataframe.

```
import pandas as pd
import numpy as np
from multiprocessing import Process, Manager
from sklearn.utils import shuffle
import time
import paralelizacion_malaria as f
```

```
if __name__ == "__main__":
    manager = Manager()
    dict_final = manager.dict()
    df_split = np.array_split(f.df2, 6)
    procesos_v = []
    for n_split, split in enumerate(df_split):
        p_ = Process(target=f.aux_calculo, args=(split, n_split, dict_final))
        p_.start()
        procesos_v.append(p_)
        time.sleep(10)
    for i in procesos_v:
        i.join()
    df_p = pd.concat(dict_final.values())
    df_p.to_csv("Shocks_malaria_paralelizado.csv")
```

```
df_p
```

	Longitud	Latitud	Year	Mes	Prep	Temp	msm1	msm2	msm3	msm4
189072	2.75	11.75	1918.0	Diciembre	2.8	25.2	False	True	True	True
189073	5.25	6.25	1953.0	Diciembre	17.9	26.3	True	True	True	True
189074	9.25	9.25	1986.0	Noviembre	5.4	22.9	True	True	True	True
189075	7.25	9.75	1901.0	Agosto	413.4	24.2	True	True	None	True
189076	14.75	5.75	1999.0	Mayo	140.5	22.7	False	False	True	True

```
df_p['msm1'] = df_p['msm1'].apply(lambda x: 1 if x==True else 0)
```

```
df_p['msm2'] = df_p['msm2'].apply(lambda x: 1 if x==True else 0)
df_p['msm3'] = df_p['msm3'].apply(lambda x: 1 if x==True else 0)
df_p['msm4'] = df_p['msm4'].apply(lambda x: 1 if x==True else 0)
```

```
df_p
```

	Longitud	Latitud	Year	Mes	Prep	Temp	msm1	msm2	msm3	msm4
189072	2.75	11.75	1918.0	Diciembre	2.8	25.2	0	1	1	1
189073	5.25	6.25	1953.0	Diciembre	17.9	26.3	1	1	1	1
189074	9.25	9.25	1986.0	Noviembre	5.4	22.9	1	1	1	1
189075	7.25	9.75	1901.0	Agosto	413.4	24.2	1	1	0	1
189076	14.75	5.75	1999.0	Mayo	140.5	22.7	0	0	1	1

```
df_p['suma'] = df_p['msm1'] + df_p['msm2'] + df_p['msm3'] + df_p['msm4']
```

```
df_p
```

	Longitud	Latitud	Year	Mes	Prep	Temp	msm1	msm2	msm3	msm4	suma
189072	2.75	11.75	1918.0	Diciembre	2.8	25.2	0	1	1	1	3
189073	5.25	6.25	1953.0	Diciembre	17.9	26.3	1	1	1	1	4
189074	9.25	9.25	1986.0	Noviembre	5.4	22.9	1	1	1	1	4
189075	7.25	9.75	1901.0	Agosto	413.4	24.2	1	1	0	1	3
189076	14.75	5.75	1999.0	Mayo	140.5	22.7	0	0	1	1	2

```
import swifter
```

```
df_p['MSM'] = df_p['suma'].swifter.apply(lambda x : 1 if x==4 else 0)
```

```
df_p = df_p.drop(['suma'], axis=1)
```

```
df_p
```

	Longitud	Latitud	Year	Mes	Prep	Temp	msm1	msm2	msm3	msm4	MSM
189072	2.75	11.75	1918.0	Diciembre	2.8	25.2	0	1	1	1	0
189073	5.25	6.25	1953.0	Diciembre	17.9	26.3	1	1	1	1	1
189074	9.25	9.25	1986.0	Noviembre	5.4	22.9	1	1	1	1	1
189075	7.25	9.75	1901.0	Agosto	413.4	24.2	1	1	0	1	0
189076	14.75	5.75	1999.0	Mayo	140.5	22.7	0	0	1	1	0

Finalmente generamos un 'Shocks_malaria_paralelizado.csv' que es el que utilizaremos para general otras tablas de datos.

A continuación, mostramos el código para generar los shocks de precipitación y temperatura. Primero generamos los shocks mensuales y posteriormente, los anuales. El código que se muestra a continuación es para Nigeria, pero cuando escalamos el algoritmo para toda África lo hicimos mediante paralelización.

```
import pandas as pd
import math
```

```
def redondear(numero):
    decimal, entero = math.modf(numero)
    decimal = round(decimal, 2)
    if decimal > 0.5:
        decimal = 0.75
    else:
        decimal = 0.25
    return entero + decimal
```

```
ndvi = pd.read_csv("ndvi.csv", index_col=0).reset_index(drop =True)
ndvi
```

	Year	NDVI	Longitud	Latitud
0	1984	0.245	2.75	11.25
1	1985	0.189	2.75	11.25
2	1986	0.282	2.75	11.25
3	1987	0.232	2.75	11.25
4	1988	0.240	2.75	11.25

```
alturas = pd.read_excel("alturas_2.xlsx")
alturas.rename(columns={'WLATITUDE': 'Latitud', 'WLONGITUDE': 'Longitud', 'WELEV':'Altura'}, inpla
# alturas.drop(alturas[alturas.Altura == -9999].index, inplace=True)
alturas = alturas.drop(alturas[(alturas.Altura == -9999)].index)
alturas['Latitud'] = alturas['Latitud'].apply(redondear)
alturas['Longitud'] = alturas['Longitud'].apply(redondear)
alturas
```

	Latitud	Longitud	Altura
1	10.25	105.25	2
2	10.25	105.25	2
3	10.25	105.75	1
4	10.25	105.75	1
5	10.25	106.25	1

```
# Agrupar duplicados en media
alturas = alturas.groupby(['Latitud', 'Longitud']).mean().reset_index()
```

```
alturas = alturas.query('(-34.75<Latitud<37.25) and (-17.25<Longitud<51.25)')
alturas
```

	Latitud	Longitud	Altura
434	-33.75	18.25	1.000000
435	-33.75	19.25	187.666667
436	-33.75	19.75	368.500000
437	-33.75	20.25	126.400000
438	-33.75	20.75	137.000000
...	...	...	...
19307	36.75	48.75	1418.500000
19308	36.75	49.25	1162.250000
19309	36.75	49.75	1026.000000
19310	36.75	50.25	1230.500000
19311	36.75	50.75	905.000000

8778 rows × 3 columns

```
ndvi = ndvi.query('(-34.75<Latitud<37.25) and (-17.25<Longitud<51.25)')
ndvi
```

	Year	NDVI	Longitud	Latitud
0	1984	0.245	2.75	11.25
1	1985	0.189	2.75	11.25

```
aux = ndvi[ndvi['Year']==1984][['NDVI','Longitud','Latitud']]
aux
```

	NDVI	Longitud	Latitud
0	0.245	2.75	11.25
36	0.250	1.75	10.75
72	0.412	2.25	6.75
108	0.234	2.75	9.75
144	0.152	2.25	8.25

```
aux = ndvi[ndvi['Year']==1984][['NDVI','Longitud','Latitud']]
ndvi = ndvi.merge(aux, left_on=['Latitud','Longitud'], right_on=['Latitud','Longitud'],
          suffixes=('', '_84'))
aux = ndvi[ndvi['Year']==2017][['NDVI','Longitud','Latitud']]
ndvi = ndvi.merge(aux, left_on=['Latitud','Longitud'], right_on=['Latitud','Longitud'],
          suffixes=('', '_2017'))
ndvi
```

	Year	NDVI	Longitud	Latitud	NDVI_84	NDVI_2017
0	1984	0.245	2.75	11.25	0.245	0.339
1	1985	0.189	2.75	11.25	0.245	0.339
2	1986	0.282	2.75	11.25	0.245	0.339
3	1987	0.232	2.75	11.25	0.245	0.339
4	1988	0.240	2.75	11.25	0.245	0.339
...	...	...	...	...	...	...
220351	2015	0.565	-15.75	12.75	0.261	0.637
220352	2016	0.707	-15.75	12.75	0.261	0.637
220353	2017	0.637	-15.75	12.75	0.261	0.637
220354	2018	0.577	-15.75	12.75	0.261	0.637
220355	2019	0.494	-15.75	12.75	0.261	0.637

220356 rows × 6 columns

```
ndvi['C_NDVI']=ndvi['NDVI_2017']-ndvi['NDVI_84']
ndvi
```

```
merge = ndvi.merge(alturas, left_on=['Latitud','Longitud'], right_on=['Latitud','Longitud'])
merge
```

```
merge.to_csv("merge_alturas_ndvi.csv")
```

Cargar precipitaciones

```
prep = pd.read_csv("precipitaciones.csv").drop(['Unnamed: 0'], axis=1)
```

```
prep
```

	Longitud	Latitud	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	Dic
0	-179.75	71.25	0.0	2.3	1.1	3.9	4.9	15.6	11.5	37.0	17.1	29.4	23.1	
1	-179.75	68.75	9.9	6.0	5.7	9.5	9.7	23.2	21.5	53.8	33.6	50.9	44.9	
2	-179.75	68.25	10.7	7.5	6.0	9.8	9.0	24.6	22.1	56.0	33.1	52.2	44.1	
3	-179.75	67.75	15.1	11.7	8.0	11.8	9.0	28.5	26.8	61.2	34.9	54.0	44.4	
4	-179.75	67.25	21.2	16.8	10.7	14.4	9.1	32.7	32.0	66.9	37.2	56.8	46.3	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10123687	179.75	-87.75	0.0	1.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.9	
10123688	179.75	-88.25	0.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
10123689	179.75	-88.75	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
10123690	179.75	-89.25	0.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
10123691	179.75	-89.75	0.0	0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

10123692 rows × 16 columns

```
prep = prep.query('(-34.75<Latitud<37.25) and (-17.25<Longitud<51.25)')
prep.shape
```

```
(1350392, 16)
```

Cargar Temperatura

```
temp = pd.read_csv("temperatura.csv").drop(['Unnamed: 0'], axis=1)
temp
```

	Longitud	Latitud	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	Dic
0	-179.75	71.25	-31.0	-27.5	-27.9	-16.4	-8.9	0.2	1.7	0.9	-0.6	-7.2	-14.9	
1	-179.75	68.75	-33.8	-30.2	-30.6	-17.6	-8.9	0.8	3.0	1.4	-0.9	-8.8	-15.9	
2	-179.75	68.25	-34.3	-30.9	-31.6	-18.3	-9.5	0.8	3.5	1.7	-1.4	-9.7	-16.7	
3	-179.75	67.75	-33.0	-29.8	-30.9	-17.0	-8.0	3.2	6.5	4.2	0.0	-9.1	-16.2	
4	-179.75	67.25	-35.1	-32.3	-33.7	-19.1	-9.6	2.8	6.7	3.7	-1.7	-12.2	-19.3	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
10037893	179.75	-87.75	-26.1	-36.9	-48.3	-55.0	-54.3	-55.5	-56.9	-57.6	-56.4	-45.2	-34.0	
10037894	179.75	-88.25	-26.6	-37.8	-49.5	-56.6	-55.9	-56.7	-58.0	-58.9	-57.4	-46.1	-34.7	
10037895	179.75	-88.75	-26.7	-38.3	-50.1	-57.5	-56.7	-57.4	-58.5	-59.6	-57.8	-46.4	-35.0	
10037896	179.75	-89.25	-26.6	-38.4	-50.6	-57.9	-56.9	-57.7	-58.5	-59.8	-57.8	-46.4	-35.1	
10037897	179.75	-89.75	-26.6	-38.5	-50.8	-58.2	-56.9	-58.1	-58.5	-60.1	-57.8	-46.5	-35.6	

10037898 rows × 16 columns

```
temp = temp.query('(-34.75<Latitud<37.25) and (-17.25<Longitud<51.25)')
temp.shape
```

```
(1338948, 16)
```

Unir tablas

```
merge = prep.merge(temp, left_on=['Latitud', 'Longitud', 'Year'], right_on=['Latitud', 'Longitud',
merge
```

	Longitud	Latitud	Enero_prep	Febrero_prep	Marzo_prep	Abril_prep	Mayo_prep	Junio_prep	Julio_prep	Agosto_pr
0	-16.75	32.75	174.8	222.1	89.7	77.5	31.9	6.3	21.9	2
1	-16.75	28.25	127.1	99.6	24.0	38.1	1.9	1.2	0.4	
2	-16.75	21.75	6.7	13.5	0.0	1.2	0.2	13.6	38.7	
3	-16.75	21.25	5.9	10.7	0.4	1.2	0.8	15.4	43.2	
4	-16.75	15.25	0.0	0.0	0.0	0.4	2.0	48.2	142.8	19
...	...	...	...	...	...	...	...	...	...	
1338943	50.75	10.75	1.5	0.0	0.3	8.1	2.2	5.2	0.1	
1338944	50.75	10.25	0.3	0.0	0.5	11.9	9.1	8.2	0.0	
1338945	50.75	10.25	0.3	0.0	0.5	11.9	9.1	8.2	0.0	
1338946	50.75	9.75	0.2	0.0	0.9	16.1	17.9	14.8	0.0	
1338947	50.75	9.75	0.2	0.0	0.9	16.1	17.9	14.8	0.0	

1338948 rows × 29 columns

```python
def one_col(row):
    lat = row['Latitud']
    lon = row['Longitud']
    year = row['Year']

    meses = ['Enero', 'Febrero','Marzo','Abril','Mayo','Junio','Julio','Agosto','Septiembre','Octu
    kind = ['prep','temp']

    table=[]
    for m in meses:
        table.append([lat,lon,int(year),m,row[f"{m}_{kind[0]}"],row[f"{m}_{kind[1]}"]])
    return table
```

```python
columns_names=['Latitud','Longitud', 'Year', 'Mes', 'Precipitaciones', 'Temperatura']
```

```python
one_col(merge.iloc[1000])+one_col(merge.iloc[1001])
```

```
[[31.75, -5.25, 1901, 'Enero', 43.7, 6.6],
 [31.75, -5.25, 1901, 'Febrero', 45.8, 8.2],
```

```python
aux = merge[10:20]
aux
```

	Longitud	Latitud	Enero_prep	Febrero_prep	Marzo_prep	Abril_prep	Mayo_prep	Junio_prep	Julio_prep	Agosto_prep	S
10	-16.25	22.25	7.8	15.4	0.0	1.4	0.0	11.1	31.6	0.0	
11	-16.25	21.75	6.5	13.0	0.0	1.2	0.3	13.6	38.1	0.0	
12	-16.25	21.25	5.6	11.0	0.6	1.2	1.1	15.4	43.3	0.0	
13	-16.25	20.75	5.2	9.6	0.9	2.1	1.8	18.0	46.4	0.0	
14	-16.25	19.75	4.7	8.9	2.9	2.0	2.5	19.6	51.6	1.8	
15	-16.25	19.25	4.6	9.1	3.7	1.9	2.8	20.8	60.1	13.6	
16	-16.25	16.75	3.6	6.2	1.1	0.4	2.0	37.9	110.0	91.9	
17	-16.25	16.25	2.4	3.5	0.8	0.1	0.6	42.7	123.1	105.4	
18	-16.25	15.75	0.0	0.0	0.0	0.2	1.1	40.8	153.2	156.7	
19	-16.25	15.25	0.0	0.0	0.0	0.6	2.4	41.6	140.6	161.3	

```python
from tqdm import tqdm
tqdm.pandas()
a = merge.progress_apply(one_col, axis=1)
# a = merge.apply(one_col, axis=1)
```

```
c:\users\usuario\pycharmprojects\prueba\venv\lib\site-packages\tqdm\std.py:666: FutureWarning: The Pa
nel class is removed from pandas. Accessing it from the top-level namespace will also be removed in t
he next version
  from pandas import Panel
100%|██████████████████████████████████████████████████████████| 1338948/1338948 [05:10<0
0:00, 4317.75it/s]
```

```
import itertools
b = list(itertools.chain.from_iterable(a))
```

```
c = pd.DataFrame(b,columns =columns_names)
c
```

	Latitud	Longitud	Year	Mes	Precipitaciones	Temperatura
0	32.75	-16.75	1901	Enero	174.8	14.7
1	32.75	-16.75	1901	Febrero	222.1	14.9
2	32.75	-16.75	1901	Marzo	89.7	15.2
3	32.75	-16.75	1901	Abril	77.5	16.2
4	32.75	-16.75	1901	Mayo	31.9	16.0
...	...	...	...	...	...	...
16067371	9.75	50.75	2017	Agosto	0.0	26.8
16067372	9.75	50.75	2017	Septiembre	2.6	26.1
16067373	9.75	50.75	2017	Octubre	17.3	26.9
16067374	9.75	50.75	2017	Noviembre	4.3	26.7
16067375	9.75	50.75	2017	Diciembre	2.4	24.6

16067376 rows × 6 columns

```
merge = c
```

```

```

```
merge.to_csv("prep_temp_africa.csv")
```

```
import pandas as pd
import numpy as np
from tqdm import tqdm
tqdm.pandas()
```

```
c:\users\usuario\pycharmprojects\prueba\venv\lib\site-packages\tqdm\std.py:666: FutureWarning: The Pa
nel class is removed from pandas. Accessing it from the top-level namespace will also be removed in t
he next version
  from pandas import Panel
```

```
df = pd.read_csv('prep_temp_africa.csv').drop(['Unnamed: 0'], axis=1)
df = df.drop_duplicates()
df = df[df['Year']>1985]
```

```
def obtener_valor(df, lon, lat, mes):
    return df.loc[(df['Longitud']==lon) & (df['Latitud']==lat) & (df['Mes']==mes) ][['Year', 'Prec
```

```
#Comprobamos la función
aux = obtener_valor(df, 4.75, 11.75, 'Septiembre')
aux = aux[aux['Year']<10]
```

```python
#A continuación creamos una función que recoja valores n_years atras

def last_n_years(lon, lat, mes, yr, n, df=df, default=None):
    year_actual = yr

    anios = obtener_valor(df, lon, lat, mes)
    filtro_anios = anios[(anios['Year']> year_actual-n) & anios['Year'] < year_actual]

    contenedor_valores_p = filtro_anios['Precipitaciones'].values
    contenedor_valores_t = filtro_anios['Temperatura'].values


    med_0_p = np.mean(contenedor_valores_p)
    sd_0_p = np.std(contenedor_valores_p)

    l_sup_p = med_0_p + 4*sd_0_p
    l_inf_p = med_0_p - 4*sd_0_p

    vector_valores_p =[]
    for v in contenedor_valores_p:
        if (v>l_inf_p) and (v<l_sup_p):
            vector_valores_p.append(v)

    med_0_t = np.mean(contenedor_valores_t)
    sd_0_t = np.std(contenedor_valores_t)

    l_sup_t = med_0_t + 4*sd_0_t
    l_inf_t = med_0_t - 4*sd_0_t

    vector_valores_t =[]
    for v in contenedor_valores_t:
        if (v>l_inf_t) and (v<l_sup_t):
            vector_valores_t.append(v)


    return vector_valores_p, vector_valores_t
```

```python
def calculo(row):
    data_p, data_t = last_n_years(row['Longitud'], row['Latitud'], row['Mes'], row['Year'], 30)
    media_p = np.mean(data_p)
    media_t = np.mean(data_t)
    desv_p = np.std(data_p)
    desv_t = np.std(data_t)
    return media_p, media_t, desv_p, desv_t
```

```
pip install pandarallel
```

```
Requirement already satisfied: pandarallel in c:\users\usuario\pycharmprojects\prueba\venv\lib\site-p
ackages (1.4.8)
Requirement already satisfied: dill in c:\users\usuario\pycharmprojects\prueba\venv\lib\site-packages
(from pandarallel) (0.3.1.1)
Note: you may need to restart the kernel to use updated packages.
```

```python
from pandarallel import pandarallel
pandarallel.initialize(progress_bar=True)
```

```
INFO: Pandarallel will run on 12 workers.
INFO: Pandarallel will use standard multiprocessing data transfer (pipe) to transfer data between the
main process and workers.
```

```python
df['Media_p'],df['Media_t'],df['Desviacion_p'],df['Desviacion_t'] = df.parallel_apply(calculo, axi
```

```
df['Z_prep'] = (df['Precipitaciones'] - df['Media_p']) / df['Desviacion_p']
df['Z_temp'] = (df['Temperatura'] - df['Media_t']) / df['Desviacion_t']
```

Creacion de shocks

```
df['SP+1_2'] = df['Z_prep'].apply(lambda x: 1 if (1<=x<2) else 0)
df['SP+2_3'] = df['Z_prep'].apply(lambda x: 1 if (2<=x<3) else 0)
df['SP+3'] = df['Z_prep'].apply(lambda x: 1 if x>=3 else 0)
df['SP+1']= df['SP+1_2'] + df['SP+2_3'] + df['SP+3']
df['SP+2']= df['SP+2_3'] + df['SP+3']
df['SP-1_2'] = df['Z_prep'].apply(lambda x: 1 if (-2<=x<-1) else 0)
df['SP-2_3'] = df['Z_prep'].apply(lambda x: 1 if (-3<=x<-2) else 0)
df['SP-3'] = df['Z_prep'].apply(lambda x: 1 if x<=-3 else 0)
df['SP-1']= df['SP-1_2'] + df['SP-2_3'] + df['SP-3']
df['SP-2']= df['SP-2_3'] + df['SP-3']
```

```
df['ST+1_2'] = df['Z_temp'].apply(lambda x: 1 if (1<=x<2) else 0)
df['ST+2_3'] = df['Z_temp'].apply(lambda x: 1 if (2<=x<3) else 0)
df['ST+3'] = df['Z_temp'].apply(lambda x: 1 if x>=3 else 0)
df['ST+1']= df['ST+1_2'] + df['ST+2_3'] + df['ST+3']
df['ST+2']= df['ST+2_3'] + df['ST+3']
df['ST-1_2'] = df['Z_temp'].apply(lambda x: 1 if (-2<=x<-1) else 0)
df['ST-2_3'] = df['Z_temp'].apply(lambda x: 1 if (-3<=x<-2) else 0)
df['ST-3'] = df['Z_temp'].apply(lambda x: 1 if x<=-3 else 0)
df['ST-1']= df['ST-1_2'] + df['ST-2_3'] + df['ST-3']
df['ST-2']= df['ST-2_3'] + df['ST-3']
```

```
df
```

```
df.to_csv('Shocks_Prep_Temp_Africa_Mensual.csv')
```

Agrupacion de shocks por años

```
df2 = pd.read_csv('Shocks_Prep_Temp_Africa_Mensual.csv').drop(['Unnamed: 0'], axis=1)
```

```
def get_value(lat, long, yr):
    return df2.loc[(df2['Latitud']==lat) & (df2['Longitud']==long) & (df2['Year']==yr)][['Mes',
        'SP+1_2', 'SP+2_3', 'SP+3', 'SP+1', 'SP+2', 'SP-1_2','SP-2_3', 'SP-3', 'SP-1', 'SP-2',
        'ST+1_2', 'ST+2_3', 'ST+3', 'ST+1', 'ST+2', 'ST-1_2','ST-2_3', 'ST-3', 'ST-1', 'ST-2']]
```

```
vector = []
for index, row in tqdm(df2.iterrows(), total=df2.shape[0]):
    lat = row['Latitud']
    long = row['Longitud']
    mes=row['Mes']
    yr = row['Year']
    values = get_value(lat, long, yr)

    val_1_p = values['SP+1_2'].sum()
    val_2_p = values['SP+2_3'].sum()
    val_3_p = values['SP+3'].sum()
    val_4_p = values['SP+1'].sum()
    val_5_p = values['SP+2'].sum()
    val_6_p = values['SP-1_2'].sum()
    val_7_p = values['SP-2_3'].sum()
    val_8_p = values['SP-3'].sum()
    val_9_p = values['SP-1'].sum()
    val_10_p= values['SP-2'].sum()

    val_1_t = values['ST+1_2'].sum()
    val_2_t = values['ST+2_3'].sum()
    val_3_t = values['ST+3'].sum()
    val_4_t = values['ST+1'].sum()
    val_5_t = values['ST+2'].sum()
    val_6_t = values['ST-1_2'].sum()
    val_7_t = values['ST-2_3'].sum()
    val_8_t = values['ST-3'].sum()
    val_9_t = values['ST-1'].sum()
    val_10_t =values['ST-2'].sum()

    vector.append([lat, long, yr,
                   val_1_p, val_2_p, val_3_p, val_4_p, val_5_p, val_6_p, val_7_p, val_8_p, val_9_p
                   val_1_t, val_2_t, val_3_t, val_4_t, val_5_t, val_6_t, val_7_t, val_8_t, val_9_t
```

```
vector
```

```
df2_anual = pd.DataFrame(vector, columns = ['Latitud', 'Longitud', 'Year',
        'SSP+1_2', 'SSP+2_3', 'SSP+3', 'SSP+1', 'SSP+2', 'SSP-1_2', 'SSP-2_3', 'SSP-3', 'SSP-1', '
        'SST+1_2', 'SST+2_3', 'SST+3', 'SST+1', 'SST+2', 'SST-1_2', 'SST-2_3', 'SST-3', 'SST-1', '
```

```
df2_anual
```

```
df2_final = df2_anual.drop_duplicates()
df2_final.shape
```

```
df2_final.to_csv('Shocks_Prep_Temp_Africa_Anual.csv')
```

Anexo 4 - NDVI

En este apartado intentaremos entender el comportamiento de la variable objetivo 'C_NDVI' desde una perspectiva gráfica. Nos apoyaremos en diagramas de puntos (scatter), en gráficos de barras y líneas y gráficos interactivos que nos permiten seleccionar las variables que deseemos (generalmente esta selección se reduce a shocks, países y años).

En primer lugar, explicamos cómo generamos los cuadros de datos que vamos a utilizar para este análisis.

```
import pandas as pd
import numpy as np
```

```
pd.set_option('display.max_columns', 60)
```

```
df1 = pd.read_csv('Shocks_Anuales_Africa.csv').drop('Unnamed: 0', axis=1)
df1
```

	Latitud	Longitud	Year	SSP+1_2	SSP+2_3	SSP+3	SSP+1	SSP+2	SSP-1_2)	SSP-2_3	SSP-3	SSP-1	SSP-2	SST+1_2	SST+
0	32.75	-16.75	1986	2.0	0.0	1.0	3.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	
1	28.25	-16.75	1986	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	
2	21.75	-16.75	1986	0.0	0.0	2.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0	1.0	
3	21.25	-16.75	1986	0.0	2.0	0.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0	1.0	
4	15.25	-16.75	1986	1.0	0.0	0.0	1.0	0.0	3.0	0.0	0.0	3.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
366203	11.75	50.75	2003	1.0	1.0	0.0	2.0	1.0	0.0	0.0	0.0	0.0	0.0	2.0	
366204	11.25	50.75	2003	1.0	1.0	0.0	2.0	1.0	0.0	0.0	0.0	0.0	0.0	2.0	
366205	10.75	50.75	2003	1.0	1.0	0.0	2.0	1.0	0.0	0.0	0.0	0.0	0.0	2.0	
366206	10.25	50.75	2003	2.0	1.0	0.0	3.0	1.0	0.0	0.0	0.0	0.0	0.0	2.0	
366207	9.75	50.75	2003	1.0	2.0	0.0	3.0	2.0	0.0	0.0	0.0	0.0	0.0	2.0	

366208 rows × 43 columns

```
seleccion_shocks=['SSP+1', 'SSP+2', 'SSP-1', 'SSP-2', 'SST+1', 'SST+2', 'SST-1', 'SST-2',
                  'SSTP++1', 'SSTP+-1',
                 'SSTP-+1','SSTP--1']
```

```
df1 = df1[['Latitud', 'Longitud', 'Year', 'SSP+1', 'SSP+2', 'SSP-1', 'SSP-2', 'SST+1', 'SST+2', 'S
                  'SSTP++1', 'SSTP+-1',
                 'SSTP-+1','SSTP--1']]
df1
```

	Latitud	Longitud	Year	SSP+1	SSP+2	SSP-1	SSP-2	SST+1	SST+2	SST-1	SST-2	SSTP++1	SSTP+-1	SSTP-+1	SSTP--1
0	32.75	-16.75	1986	3.0	1.0	1.0	0.0	0.0	0.0	5.0	1.0	0.0	0.0	2.0	0.0
1	28.25	-16.75	1986	1.0	0.0	0.0	0.0	2.0	0.0	4.0	1.0	1.0	0.0	0.0	0.0
2	21.75	-16.75	1986	2.0	2.0	0.0	0.0	1.0	0.0	6.0	0.0	0.0	0.0	1.0	0.0
3	21.25	-16.75	1986	2.0	2.0	0.0	0.0	1.0	0.0	5.0	0.0	0.0	0.0	1.0	0.0
4	15.25	-16.75	1986	1.0	0.0	3.0	0.0	0.0	0.0	7.0	3.0	0.0	0.0	1.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
366203	11.75	50.75	2003	2.0	1.0	0.0	0.0	4.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0
366204	11.25	50.75	2003	2.0	1.0	0.0	0.0	4.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0
366205	10.75	50.75	2003	2.0	1.0	0.0	0.0	4.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0
366206	10.25	50.75	2003	3.0	1.0	0.0	0.0	4.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0
366207	9.75	50.75	2003	3.0	2.0	0.0	0.0	4.0	2.0	1.0	0.0	0.0	0.0	0.0	0.0

366208 rows × 15 columns

```
#A continuacion vamos a georreferenciar por paises
```

```
df_geo = pd.read_csv('Pais_Provincia_Distrito_georreferenciado.csv').drop('Unnamed: 0', axis=1)
df_geo
```

	Country	Province	Distrito	Longitud	Latitud
0	Afghanistan	Badakhshan	Baharak	71.25	37.25
1	Afghanistan	Badakhshan	Darwaz	70.75	38.25
2	Afghanistan	Badakhshan	Fayzabad	70.25	37.25
3	Afghanistan	Badakhshan	Ishkashim	71.25	36.75
4	Afghanistan	Badakhshan	Jurm	70.75	36.75
...	...	...	...	...	...
150333	Libya	Tripoli	Tripoli|Shab'iyat Ţarābulus	13.25	32.75
150334	Libya	Wadi al Hayat	Wadi Al Hayaa|Wādī al Ḥayāh	12.75	26.25
150335	Libya	Wadi al Hayat	Wadi Al Hayaa|Wādī al Ḥayāh	13.75	26.25
150336	Libya	Wadi ash Shati'	Wadi Al Shatii|Shati|Wadi ash-Shati'| Wādī aš ...	12.75	27.75
150337	Libya	Wadi ash Shati'	Wadi Al Shatii|Shati|Wadi ash-Shati'| Wādī aš ...	11.25	26.25

150338 rows × 5 columns

```
df_merge2 = pd.merge(df_geo, df1, how='inner', on=['Longitud', 'Latitud'])
df_merge2 = df_merge2.drop_duplicates()
df_merge2 = df_merge2.dropna()
df_merge2
```

	Country	Province	Distrito	Longitud	Latitud	Year	SSP+1	SSP+2	SSP-1	SSP-2	SST+1	SST+2	SST-1	SST-2
0	Angola	Bengo	Ambriz	13.25	-6.75	1986	0.0	0.0	2.0	0.0	0.0	0.0	3.0	0.0
1	Angola	Bengo	Ambriz	13.25	-6.75	1987	4.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0
2	Angola	Bengo	Ambriz	13.25	-6.75	1997	2.0	0.0	2.0	0.0	1.0	0.0	4.0	0.0
3	Angola	Bengo	Ambriz	13.25	-6.75	1998	5.0	2.0	1.0	0.0	8.0	3.0	0.0	0.0
4	Angola	Bengo	Ambriz	13.25	-6.75	2000	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Al considerar tres intervalos de estudio es necesario crear un cuadro de datos donde para cada punto de estudio georreferenciado (clúster) tengamos el valor de la variable al principio de cada uno de los intervalos para posteriormente poder calcular las variaciones. Este cuadro lo hemos creado con el nombre de 'Panel_DATA'. Antes de seguir con el estudio facilitamos el procedimiento de creación de este cuadro de datos.

```
import pandas as pd
import numpy as np
```

```
panel = pd.read_csv('Panel_DATA.csv').drop('Unnamed: 0', axis=1)
panel
```

	Year	NDVI	Longitud	Latitud	NDVI_84	NDVI_95	NDVI_06	NDVI_17
0	1984	0.245	2.75	11.25	0.245	0.324	0.320	0.339
1	1985	0.189	2.75	11.25	0.245	0.324	0.320	0.339
2	1986	0.282	2.75	11.25	0.245	0.324	0.320	0.339
3	1987	0.232	2.75	11.25	0.245	0.324	0.320	0.339
4	1988	0.240	2.75	11.25	0.245	0.324	0.320	0.339
...	...	...	...	...	...	...	...	...
72643	2015	0.565	-15.75	12.75	0.261	0.547	0.548	0.637
72644	2016	0.707	-15.75	12.75	0.261	0.547	0.548	0.637
72645	2017	0.637	-15.75	12.75	0.261	0.547	0.548	0.637
72646	2018	0.577	-15.75	12.75	0.261	0.547	0.548	0.637
72647	2019	0.494	-15.75	12.75	0.261	0.547	0.548	0.637

72648 rows × 8 columns

```
panel['C_84_95']= panel['NDVI_95'] - panel['NDVI_84']
```

```
panel['C_95_06']= panel['NDVI_06'] - panel['NDVI_95']
```

```
panel['C_06_17']= panel['NDVI_17'] - panel['NDVI_06']
```

```
panel
```

	Year	NDVI	Longitud	Latitud	NDVI_84	NDVI_95	NDVI_06	NDVI_17	C_84_95	C_95_06	C_06_17
0	1984	0.245	2.75	11.25	0.245	0.324	0.320	0.339	0.079	-0.004	0.019
1	1985	0.189	2.75	11.25	0.245	0.324	0.320	0.339	0.079	-0.004	0.019
2	1986	0.282	2.75	11.25	0.245	0.324	0.320	0.339	0.079	-0.004	0.019
3	1987	0.232	2.75	11.25	0.245	0.324	0.320	0.339	0.079	-0.004	0.019
4	1988	0.240	2.75	11.25	0.245	0.324	0.320	0.339	0.079	-0.004	0.019
...	...	...	...	...	...	...	...	...	...	...	...

```
panel.to_csv('Panel_Completo_sin_paises.csv')
```

Sobre este cuadro hacemos algunas modificaciones y crearemos más cuadros de datos que utilizaremos en función de las necesidades explicativas que tengamos.

```
corte = panel.drop(['Year', 'NDVI', 'NDVI_84', 'NDVI_95', 'NDVI_06', 'NDVI_17'], axis=1)
```

```
corte
```

	Longitud	Latitud	C_84_95	C_95_06	C_06_17
0	2.75	11.25	0.079	-0.004	0.019
1	2.75	11.25	0.079	-0.004	0.019
2	2.75	11.25	0.079	-0.004	0.019
3	2.75	11.25	0.079	-0.004	0.019
4	2.75	11.25	0.079	-0.004	0.019

```
corte.to_csv('PANEL_DATA_Final.csv')
```

Finalmente completaremos este cuadro con nombres de países y provincias. Para ello recurrimos a otro cuadro generado anteriormente. Éste ha sido creado a través de una herramienta de georreferenciación 'Qgis' ya que no fuimos capaz de encontrar ninguna base de datos georreferenciada con un nivel de desagregación lo suficientemente grande.

```
paises = pd.read_csv('Pais_Provincia_Distrito_georreferenciado.csv')
paises
```

	Unnamed: 0	Country	Province	Distrito	Longitud	Latitud
0	0	Afghanistan	Badakhshan	Baharak	71.25	37.25
1	1	Afghanistan	Badakhshan	Darwaz	70.75	38.25
2	2	Afghanistan	Badakhshan	Fayzabad	70.25	37.25
3	3	Afghanistan	Badakhshan	Ishkashim	71.25	36.75
4	4	Afghanistan	Badakhshan	Jurm	70.75	36.75

```
paises = paises.drop(['Unnamed: 0', 'Province', 'Distrito'], axis=1)
paises
```

```
merge = pd.merge(paises, corte, on=['Latitud','Longitud'])
merge
```

	Country	Longitud	Latitud	C_84_95	C_95_06	C_06_17
0	Angola	13.75	-7.75	0.195	-0.050	-0.060
1	Angola	14.25	-11.75	0.083	-0.044	-0.025
2	Angola	17.25	-11.75	0.049	-0.114	0.035
3	Angola	17.25	-11.75	0.049	-0.114	0.035
4	Angola	14.75	-7.75	0.010	0.012	-0.095
...	...	...	...	...	...	...
1899	Libya	14.25	32.75	0.122	-0.153	0.016
1900	Libya	11.75	32.75	0.030	-0.018	-0.001
1901	Libya	20.75	31.75	0.151	-0.291	0.048
1902	Libya	13.25	32.75	-0.005	-0.029	-0.005
1903	Libya	12.75	27.75	0.051	-0.058	0.020

1904 rows × 6 columns

```
merge.to_csv('Panel_DATA_final_por_paises')
```

Ahora abrimos otro cuadro de datos que utilizaremos es la práctica totalidad de nuestros estudios de productividad del suelo que es el de los 'Shocks_Anuales_África.csv', cuya generación se explica en el apéndice 1.

```
df1 = pd.read_csv('Shocks_Anuales_Africa.csv').drop('Unnamed: 0', axis=1)
df1
```

	Latitud	Longitud	Year	SSP+1_2	SSP+2_3	SSP+3	SSP+1	SSP+2	SSP-1_2)	SSP-2_3	SSP-3	SSP-1	SSP-2	SST+1_2	SST+
0	32.75	-16.75	1986	2.0	0.0	1.0	3.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	
1	28.25	-16.75	1986	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	
2	21.75	-16.75	1986	0.0	0.0	2.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0	1.0	
3	21.25	-16.75	1986	0.0	2.0	0.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0	1.0	
4	15.25	-16.75	1986	1.0	0.0	0.0	1.0	0.0	3.0	0.0	0.0	3.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
366203	11.75	50.75	2003	1.0	1.0	0.0	2.0	1.0	0.0	0.0	0.0	0.0	0.0	2.0	
366204	11.25	50.75	2003	1.0	1.0	0.0	2.0	1.0	0.0	0.0	0.0	0.0	0.0	2.0	
366205	10.75	50.75	2003	1.0	1.0	0.0	2.0	1.0	0.0	0.0	0.0	0.0	0.0	2.0	
366206	10.25	50.75	2003	2.0	1.0	0.0	3.0	1.0	0.0	0.0	0.0	0.0	0.0	2.0	
366207	9.75	50.75	2003	1.0	2.0	0.0	3.0	2.0	0.0	0.0	0.0	0.0	0.0	2.0	

366208 rows × 43 columns

Hacemos una selección de shocks para quedarnos sólo con los que nos interesan.

```
df1 = df1[['Latitud', 'Longitud', 'Year', 'SSP+1', 'SSP+2', 'SSP-1', 'SSP-2', 'SST+1', 'SST+2', 'S
                  'SSTP++1', 'SSTP+-1',
                 'SSTP-+1','SSTP--1']]
df1
```

	Latitud	Longitud	Year	SSP+1	SSP+2	SSP-1	SSP-2	SST+1	SST+2	SST-1	SST-2	SSTP++1	SSTP+-1	SSTP-+1	SSTP--1
0	32.75	-16.75	1986	3.0	1.0	1.0	0.0	0.0	0.0	5.0	1.0	0.0	0.0	2.0	0.0
1	28.25	-16.75	1986	1.0	0.0	0.0	0.0	2.0	0.0	4.0	1.0	1.0	0.0	0.0	0.0
2	21.75	-16.75	1986	2.0	2.0	0.0	0.0	1.0	0.0	6.0	0.0	0.0	0.0	1.0	0.0
3	21.25	-16.75	1986	2.0	2.0	0.0	0.0	1.0	0.0	5.0	0.0	0.0	0.0	1.0	0.0
4	15.25	-16.75	1986	1.0	0.0	3.0	0.0	0.0	0.0	7.0	3.0	0.0	0.0	1.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
366203	11.75	50.75	2003	2.0	1.0	0.0	0.0	4.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0
366204	11.25	50.75	2003	2.0	1.0	0.0	0.0	4.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0
366205	10.75	50.75	2003	2.0	1.0	0.0	0.0	4.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0
366206	10.25	50.75	2003	3.0	1.0	0.0	0.0	4.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0
366207	9.75	50.75	2003	3.0	2.0	0.0	0.0	4.0	2.0	1.0	0.0	0.0	0.0	0.0	0.0

366208 rows × 15 columns

Pasamos de tener 43 variables predictoras a tener 15, pero como veremos en el modelo, basándonos en el criterio de significancia.

```
#A continuacion vamos a georreferenciar por paises
```

```
df_geo = pd.read_csv('Pais_Provincia_Distrito_georreferenciado.csv').drop('Unnamed: 0', axis=1)
df_geo
```

	Country	Province	Distrito	Longitud	Latitud
0	Afghanistan	Badakhshan	Baharak	71.25	37.25
1	Afghanistan	Badakhshan	Darwaz	70.75	38.25

```
df_merge2 = pd.merge(df_geo, df1, how='inner', on=['Longitud', 'Latitud'])
df_merge2 = df_merge2.drop_duplicates()
df_merge2 = df_merge2.dropna()
df_merge2
```

	Country	Province	Distrito	Longitud	Latitud	Year	SSP+1	SSP+2	SSP--1	SSP--2	SST+1	SST+2	SST--1	SST--2
0	Angola	Bengo	Ambriz	13.25	-6.75	1986	0.0	0.0	2.0	0.0	0.0	0.0	3.0	0.0
1	Angola	Bengo	Ambriz	13.25	-6.75	1987	4.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0
2	Angola	Bengo	Ambriz	13.25	-6.75	1997	2.0	0.0	2.0	0.0	1.0	0.0	4.0	0.0

A continuación, procedemos a hacer un merge con el cuadro 'Panel_Data'.

```
df_merge3 = pd.merge(df, df1, how='inner', on=['Longitud', 'Latitud', 'Year'])
df_merge3 = df_merge3.drop_duplicates()
df_merge3 = df_merge3.dropna()
df_merge3
```

	Year	NDVI	Longitud	Latitud	NDVI_84	NDVI_95	NDVI_06	NDVI_17	SSP+1	SSP+2	SSP--1	SSP--2	SST+1	SST+2
0	1986	0.282	2.75	11.25	0.245	0.324	0.320	0.339	0.0	0.0	1.0	0.0	0.0	0.0
1	1987	0.232	2.75	11.25	0.245	0.324	0.320	0.339	0.0	0.0	2.0	0.0	5.0	1.0
2	1988	0.240	2.75	11.25	0.245	0.324	0.320	0.339	0.0	0.0	3.0	0.0	1.0	0.0

```
df_merge3.to_csv('TABLA_para Creacion_VAriacion_shocks.csv')
```

El último paso es crear la misma tabla, pero georreferenciada por países y provincias.

```
df_merge4 = pd.merge(df_geo, df_merge3, how='inner', on=['Longitud', 'Latitud'])
df_merge4 = df_merge4.drop_duplicates()
df_merge4 = df_merge4.dropna()
df_merge4
```

	Country	Province	Distrito	Longitud	Latitud	Year	NDVI	NDVI_84	NDVI_95	NDVI_06	NDVI_17	SSP+1
0	Angola	Bengo	Dande	13.75	-7.75	1986	0.647	0.453	0.648	0.598	0.538	2.0
1	Angola	Bengo	Dande	13.75	-7.75	1987	0.467	0.453	0.648	0.598	0.538	1.0
2	Angola	Bengo	Dande	13.75	-7.75	1988	0.580	0.453	0.648	0.598	0.538	0.0
3	Angola	Bengo	Dande	13.75	-7.75	1989	0.608	0.453	0.648	0.598	0.538	1.0

```
df_merge4.to_csv('TABLA_PAISES_NDVI_SHOCKS_georref.csv')
```

Finalmente crearemos un cuadro de datos donde se tenga como información el NDVI medio por país y año.

```
def suma_por_anios(yr, country):
    return df_merge4.loc[(df_merge4['Year']==yr) & (df_merge4['Country']==country)][['SSP+1', 'SSP
                'SSTP++1', 'SSTP+-1','SSTP-+1','SSTP--1']].sum().values
```

```
suma_por_anios(1996, 'Nigeria')
```

```
array([80., 16., 26.,  2., 15.,  0., 77., 12.,  0.,  0., 22.,  3.])
```

```
def NDVI_medio(yr, country):
    return df_merge4.loc[(df_merge4['Year']==yr) & (df_merge4['Country']==country)]['NDVI'].mean()
```

```
NDVI_medio(1986, 'Nigeria')
```

```
0.3357027027027028
```

```
from tqdm import tqdm
```

```
#iterando para toda la tabla

contenedor_valores1 = []
contenedor_valores2 = []
contenedor_ndvi = []
for index, row in tqdm(df_merge4.iterrows()):
    ctry = row['Country']
    lg = row['Longitud']
    lt = row['Latitud']
    y = row['Year']
    ndvi = row['NDVI']

    val1 = suma_por_anios(y, ctry)
    val2 = NDVI_medio(y, ctry)
    contenedor_valores1.append([ctry, lg, lt, y])
    contenedor_valores2.append(val1)
    contenedor_ndvi.append([y, ctry, val2])
```

```
10618it [00:26, 401.47it/s]
```

```
a_ = pd.DataFrame(contenedor_valores1, columns=['Country', 'Longitud', 'Latitud', 'Year'])
a_.head(2)
```

	Country	Longitud	Latitud	Year
0	Angola	13.75	-7.75	1986
1	Angola	13.75	-7.75	1987

```
b_ = pd.DataFrame(contenedor_valores2, columns=['SSP+1', 'SSP+2', 'SSP-1', 'SSP-2', 'SST+1', 'SST+
                  'SSTP++1', 'SSTP+-1',
                 'SSTP-+1','SSTP--1'])
b_.head(4)
```

	SSP+1	SSP+2	SSP-1	SSP-2	SST+1	SST+2	SST-1	SST-2	SSTP++1	SSTP+-1	SSTP-+1	SSTP--1
0	26.0	6.0	8.0	0.0	0.0	0.0	31.0	1.0	0.0	0.0	7.0	1.0
1	22.0	8.0	20.0	3.0	9.0	0.0	4.0	0.0	1.0	1.0	0.0	0.0
2	18.0	5.0	10.0	0.0	0.0	0.0	47.0	0.0	0.0	0.0	5.0	5.0
3	18.0	9.0	28.0	10.0	3.0	0.0	103.0	7.0	0.0	2.0	14.0	11.0

```
ab_ = pd.concat([a_, b_], axis=1)
```

```
c = pd.DataFrame(contenedor_ndvi, columns=['Year', 'Country', 'NDVI'])
```

```
ab_.shape,        c.shape
```

```
((10618, 16), (10618, 3))
```

```
ab_.to_csv('tabla_paises_shocks_georref_SUMA_POR_YEARS.csv')
```

```
abc = pd.concat([ab_,c], axis=1)
```

```
ab_2 = ab_.drop(['Longitud', 'Latitud'], axis=1)
ab_2
```

	Country	Year	SSP+1	SSP+2	SSP-1	SSP-2	SST+1	SST+2	SST-1	SST-2	SSTP++1	SSTP+-1	SSTP-+1	SSTP--1
0	Angola	1986	26.0	6.0	8.0	0.0	0.0	0.0	31.0	1.0	0.0	0.0	7.0	1.0
1	Angola	1987	22.0	8.0	20.0	3.0	9.0	0.0	4.0	0.0	1.0	1.0	0.0	0.0

```
tabla_ = pd.concat([c, ab_2], axis=1)
```

```
tabla_.head(5)
```

	Year	Country	NDVI	Country	Year	SSP+1	SSP+2	SSP-1	SSP-2	SST+1	SST+2	SST-1	SST-2	SSTP++1	SSTP+-1	SS
0	1986	Angola	0.669929	Angola	1986	26.0	6.0	8.0	0.0	0.0	0.0	31.0	1.0	0.0	0.0	
1	1987	Angola	0.678429	Angola	1987	22.0	8.0	20.0	3.0	9.0	0.0	4.0	0.0	1.0	1.0	
2	1988	Angola	0.710000	Angola	1988	18.0	5.0	10.0	0.0	0.0	0.0	47.0	0.0	0.0	0.0	
3	1989	Angola	0.741500	Angola	1989	18.0	9.0	28.0	10.0	3.0	0.0	103.0	7.0	0.0	2.0	1
4	1990	Angola	0.573429	Angola	1990	38.0	16.0	16.0	1.0	0.0	0.0	38.0	0.0	0.0	0.0	1

```
tabla_.to_csv('TABLA_PAISES_NDVI_SHOCKS_POR_YEAR.csv')
```

En primer lugar, nos interesa comparar cómo evoluciona el número de shocks con el comportamiento del NDVI por país a lo largo del tiempo. Este tipo de análisis es muy importante porque nos ayuda a ver si existe algún tipo de correlación entre ellas de forma contemporánea o con algún desfase de tiempo. Para esto es necesario hacer una reestructuración del cuadro que tenemos inmediatamente en la parte superior, nos interesa crear un agregado anual.

```
#Reestructuracion de la tabla
```

```
lista_sk = ['SSP+1', 'SSP+2', 'SSP-1', 'SSP-2', 'SST+1', 'SST+2', 'SST-1', 'SST-2',
                  'SSTP++1', 'SSTP+-1',
                 'SSTP-+1','SSTP--1']
```

```

```

```
def one_col(row):
    yr = row['Year']
    country = row['Country']
    ndvi = row['NDVI']

    tabla = []
    for s in lista_sk:
        tabla.append([yr, country, ndvi, s, row[s]])
    return tabla
```

```
from tqdm import tqdm
tqdm.pandas()
a = df.progress_apply(one_col, axis=1)
```

```
import itertools
b = list(itertools.chain.from_iterable(a))
```

```
df_ = pd.DataFrame(b, columns=['Year','Country', 'NDVI', 'Shock', 'Val_Sk'])
```

```
df_
```

	Year	Country	NDVI	Shock	Val_Sk
0	1986	Angola	0.669929	SSP+1	26.0
1	1986	Angola	0.669929	SSP+2	6.0
2	1986	Angola	0.669929	SSP-1	8.0

```
lista_years = df_['Year'].unique().tolist()
lista_paises = df_['Country'].unique().tolist()
lista_sk = df_['Shock'].unique().tolist()
```

```
def devolver_valor1(country, sk):
    return df_.loc[(df_['Country']==country) & (df_['Shock']==sk)][['Year', 'Val_Sk']]
```

```
def devolver_ndvi(country, sk):
    return df_.loc[(df_['Country']==country) & (df_['Shock']==sk)][['Year','NDVI']]
```

```
devolver_valor1('Angola', 'SSP+1')
```

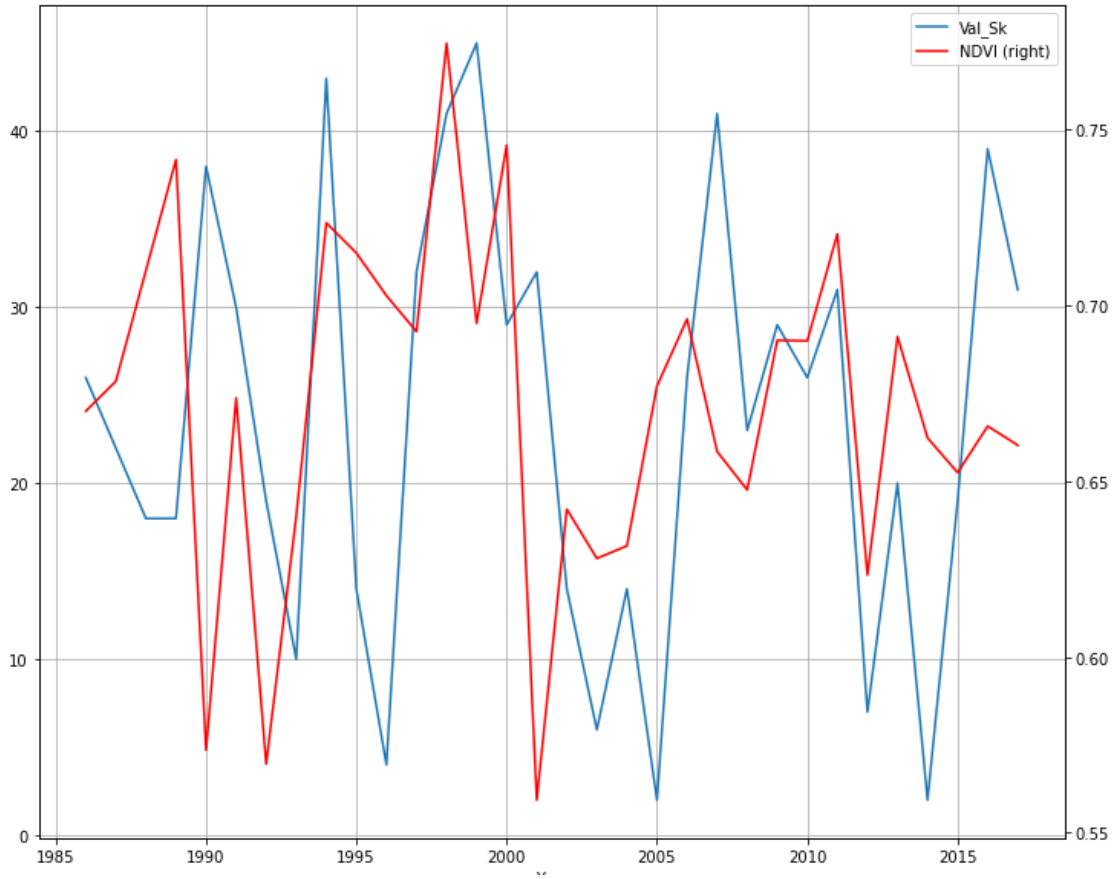
	Year	Val_Sk
0	1986	26.0
12	1987	22.0
24	1988	18.0
36	1989	18.0
48	1990	38.0
60	1991	30.0
72	1992	19.0
84	1993	10.0
96	1994	43.0
108	1995	14.0
120	1996	4.0

```
from ipywidgets import interact
```

```
def dibujo_evolucion(country, sk):
    resultado = df_.loc[(df_['Country']==country) & (df_['Shock']==sk)][['Year', 'Val_Sk']]
    resultado2 = df_.loc[(df_['Country']==country)][['Year','NDVI']]
    resultado = resultado.set_index('Year')
    resultado2 = resultado2.set_index('Year')
    fig, ax = plt.subplots(figsize=(12,10))
    fig.suptitle(f'Número de Shokcs climáticos {sk}--NDVI en {country}', fontsize=25)
    return resultado.plot(kind='line', ax = ax, grid=True), resultado2.plot(kind='line', ax=ax, co
                                                                            secondary_y=True, grid
interact(dibujo_evolucion, country=lista_paises, sk=lista_sk)
```

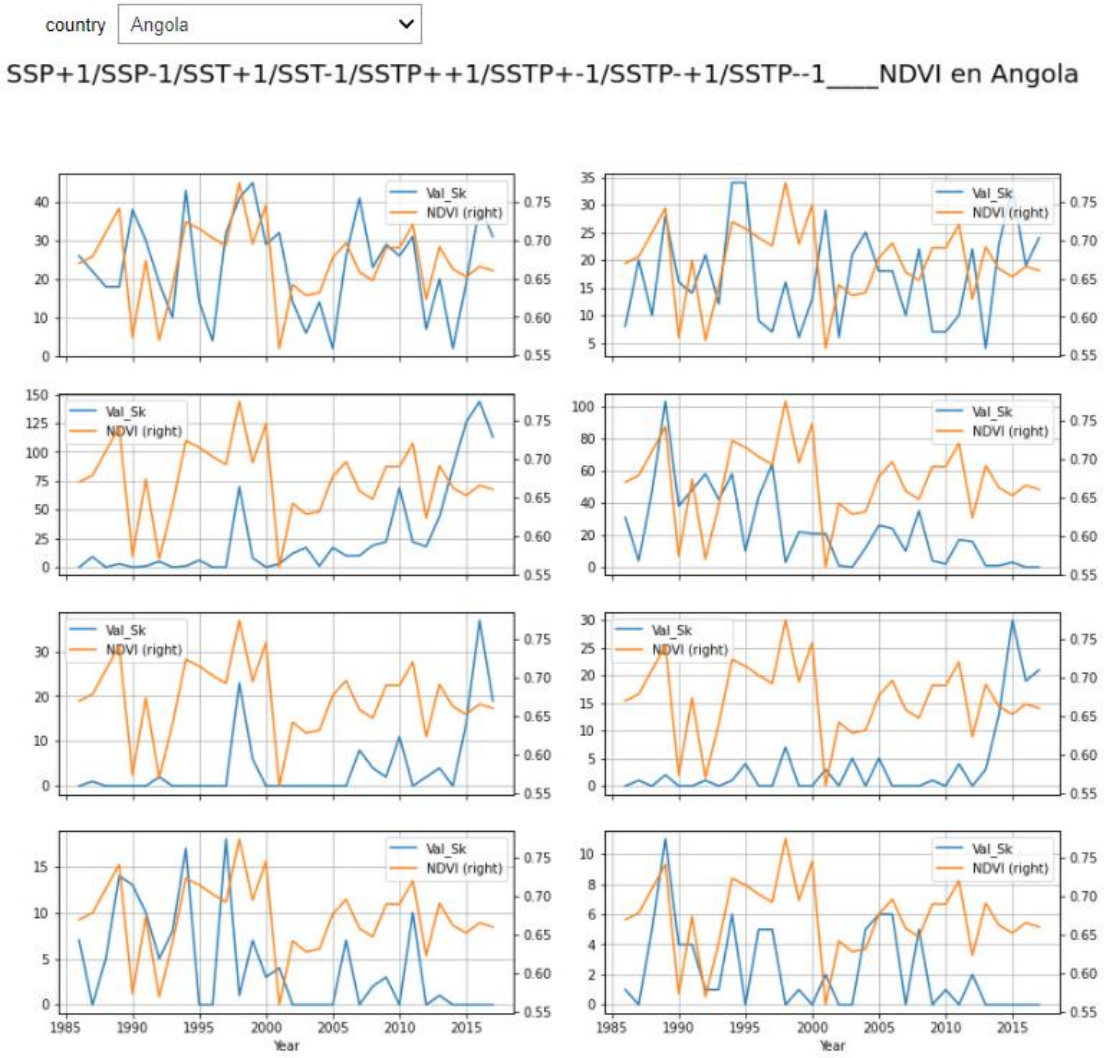
En este gráfico se ve muy bien como existe un comportamiento más o menos parecido entre las dos variables, pero con un desfase temporal de un año, por lo que puede existir una correlación importante entre el $NDVI_t$ y $SSP+1_{t-1}$

Número de Shokcs climáticos SSP+1--NDVI en Angola




```python
def dibujo_evolucion2(country):
    resultado_1 = df_.loc[(df_['Country']==country) & (df_['Shock']=='SSP+1')][['Year', 'Val_Sk']]
    resultado_2 = df_.loc[(df_['Country']==country) & (df_['Shock']=='SSP-1')][['Year', 'Val_Sk']]
    resultado_3 = df_.loc[(df_['Country']==country) & (df_['Shock']=='SST+1')][['Year', 'Val_Sk']]
    resultado_4 = df_.loc[(df_['Country']==country) & (df_['Shock']=='SST-1')][['Year', 'Val_Sk']]

    resultado_5 = df_.loc[(df_['Country']==country) & (df_['Shock']=='SSTP++1')][['Year', 'Val_Sk'
    resultado_6 = df_.loc[(df_['Country']==country) & (df_['Shock']=='SSTP+-1')][['Year', 'Val_Sk'
    resultado_7 = df_.loc[(df_['Country']==country) & (df_['Shock']=='SSTP-+1')][['Year', 'Val_Sk'
    resultado_8 = df_.loc[(df_['Country']==country) & (df_['Shock']=='SSTP--1')][['Year', 'Val_Sk'


    resultado_ndvi = df_.loc[(df_['Country']==country)][['Year','NDVI']]

    resultado_1 = resultado_1.set_index('Year')
    resultado_2 = resultado_2.set_index('Year')
    resultado_3 = resultado_3.set_index('Year')
    resultado_4 = resultado_4.set_index('Year')

    resultado_5 = resultado_5.set_index('Year')
    resultado_6 = resultado_6.set_index('Year')
    resultado_7 = resultado_7.set_index('Year')
    resultado_8 = resultado_8.set_index('Year')

    resultado_ndvi = resultado_ndvi.set_index('Year')

    fig, ax = plt.subplots(4,2, figsize=(14, 12))

    fig.suptitle(f'SSP+1/SSP-1/SST+1/SST-1/SSTP++1/SSTP+-1/SSTP-+1/SSTP--1____NDVI en {country}',

    return resultado_1.plot(kind='line', ax = ax[0][0], grid=True, label=f'SSP+1'), resultado_2.pl

interact(dibujo_evolucion2, country=lista_paises)
```

Este código nos ayuda a ver la comparativa de todos los shocks simultáneamente por país.



A continuación, lo que trataremos de ver es si existe alguna relación entre el tipo de shock y la localización geográfica en donde suceden (por latitud y longitud).

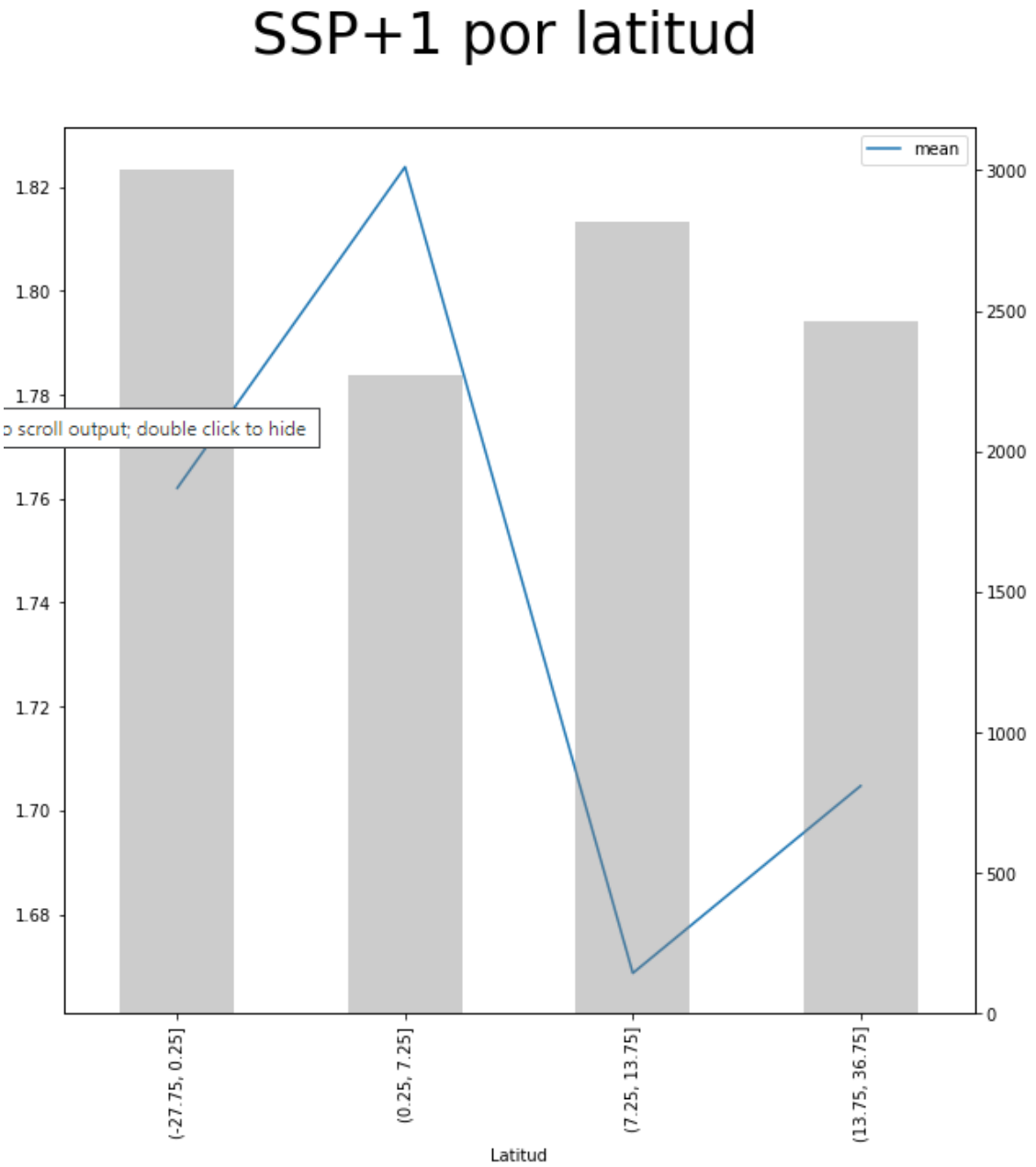
En este caso como necesitamos los datos georreferenciados por latitud y longitud recurrimos a otra tabla cuya generación explicamos en este mismo apéndice.

```
1
2 df2 = pd.read_csv('tabla_paises_ndvi_shocks_georref.csv').drop('Unnamed: 0', axis=1)
3 df2
```

```
1 df2 = df2.drop(['NDVI_84', 'NDVI_95', 'NDVI_06', 'NDVI_17'], axis=1)
2 df2
```

	Country	Longitud	Latitud	Year	NDVI	SSP+1	SSP+2	SSP-1	SSP-2	SST+1	SST+2	SST-1	SST-2	SSTP++1	SSTP+-1
0	Angola	13.75	-7.75	1986	0.647	2.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0
1	Angola	13.75	-7.75	1987	0.467	1.0	1.0	2.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
2	Angola	13.75	-7.75	1988	0.580	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0
3	Angola	13.75	-7.75	1989	0.608	1.0	1.0	1.0	0.0	0.0	0.0	7.0	0.0	0.0	0.0

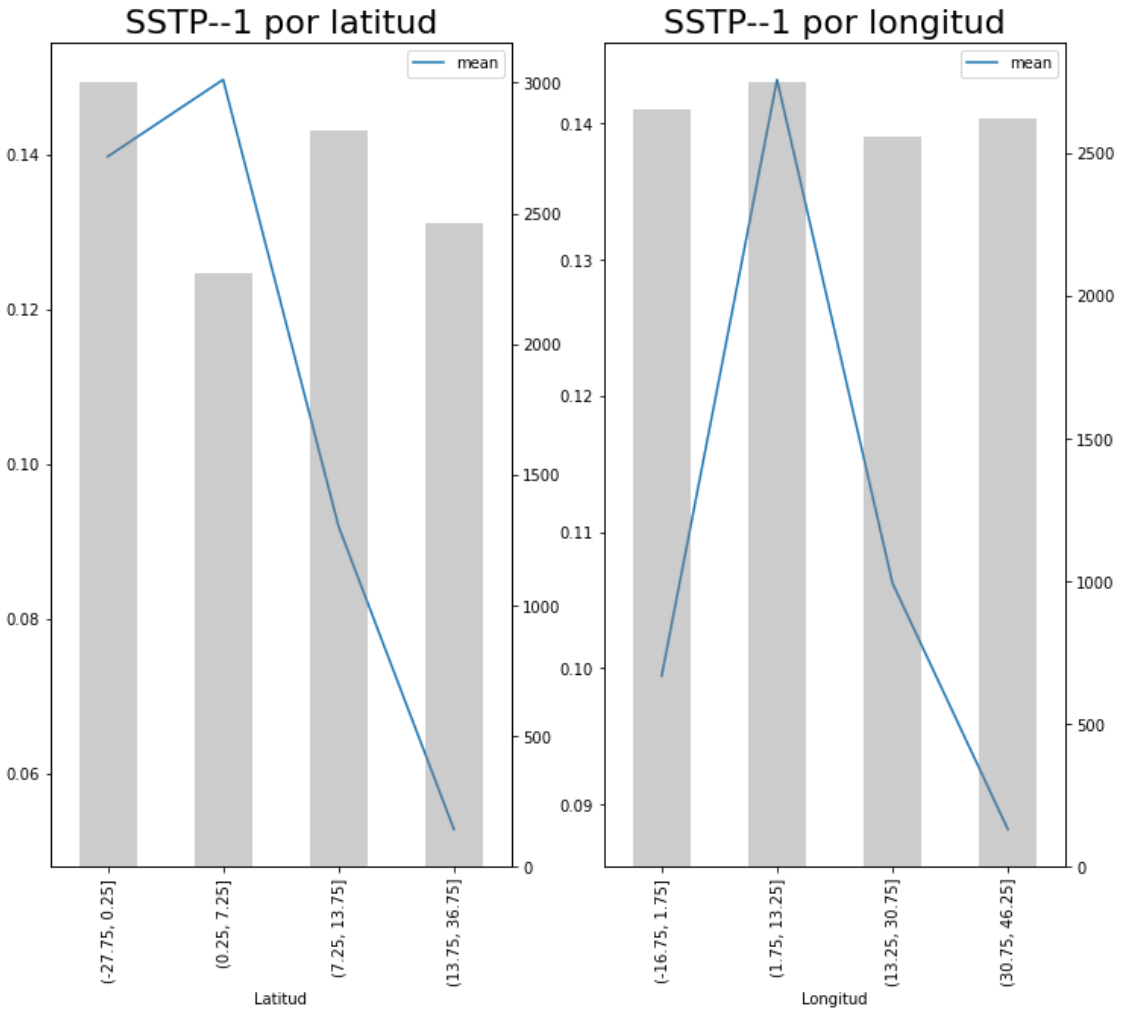
```
intervalos = np.quantile(df2['Latitud'], np.linspace(0,1,5))
grupos = df2.groupby([pd.cut(df2['Latitud'], bins=intervalos)])['SSP+1'].agg(['mean', 'size'])

fig, ax = plt.subplots(figsize=(10,10))
fig.suptitle('SSP+1 por latitud', fontsize=35)

grupos[['mean']].plot(kind='line', ax=ax)
grupos['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True)
```



En este gráfico lo que hemos hecho es dividir la variable 'Latitud' en intervalos, concretamente en cuatro. A partir de ahí lo que hacemos es que nos muestre el valor medio del shock en cada uno de los intervalos.

Para tener una localización más precisa donde cada uno de los shocks se da con más intensidad, lo que haremos es ampliar este gráfico con otra ventana donde nos muestre el comportamiento por latitud.

```
intervalos_lt = np.quantile(df2['Latitud'], np.linspace(0,1,5))
intervalos_lg = np.quantile(df2['Longitud'], np.linspace(0,1,5))

grupos_lt = df2.groupby([pd.cut(df2['Latitud'], bins=intervalos_lt)])['SSTP--1'].agg(['mean', 'siz
grupos_lg = df2.groupby([pd.cut(df2['Longitud'], bins=intervalos_lg)])['SSTP--1'].agg(['mean', 'si

fig, ax = plt.subplots(1,2, figsize=(12,10))
ax[0].set_title('SSTP--1 por latitud', fontsize=22)
ax[1].set_title('SSTP--1 por longitud', fontsize=22)

grupos_lt[['mean']].plot(kind='line', ax=ax[0])
grupos_lt['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[0])

grupos_lg[['mean']].plot(kind='line', ax=ax[1])
grupos_lg['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[1])
```



Finalmente cogemos los intervalos donde se produce el pico en los valores promedios y dibujamos la zona con ayuda de la librería 'folium' de Python.

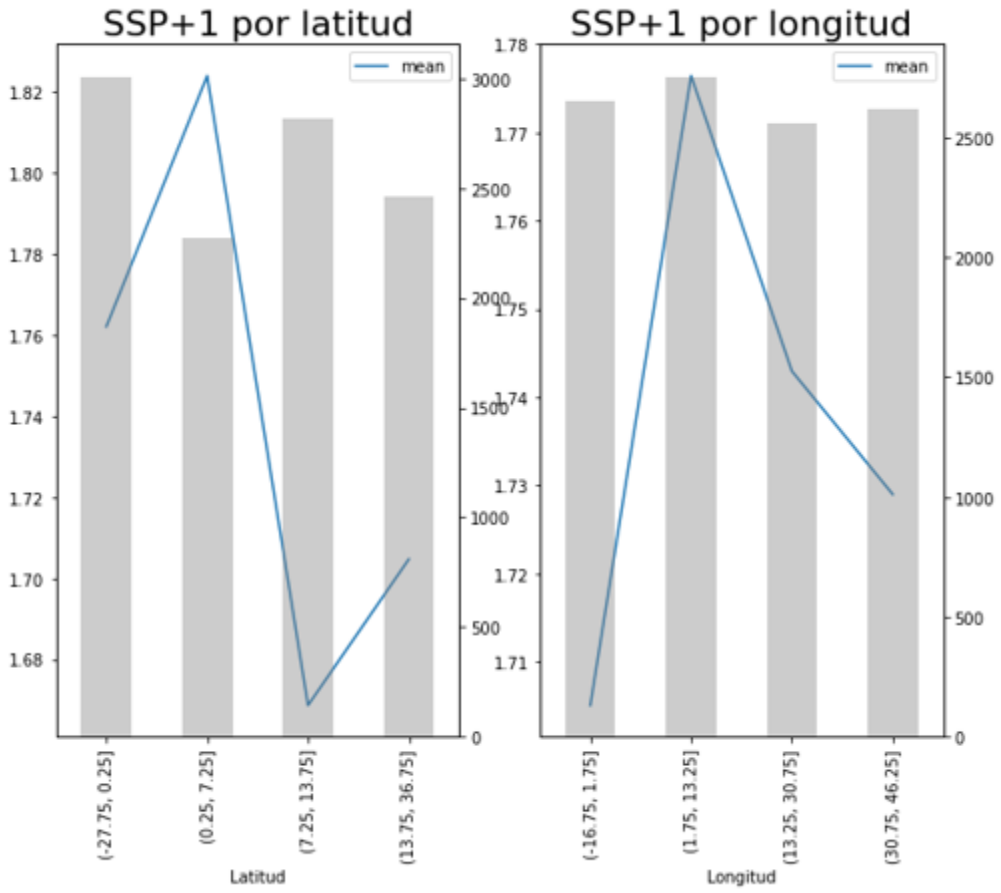
```
import folium
```

```


longitud = 13.25
latitud = 7.25

map_ = folium.Map(location=[latitud, longitud], zoom_start=7)

map_
```

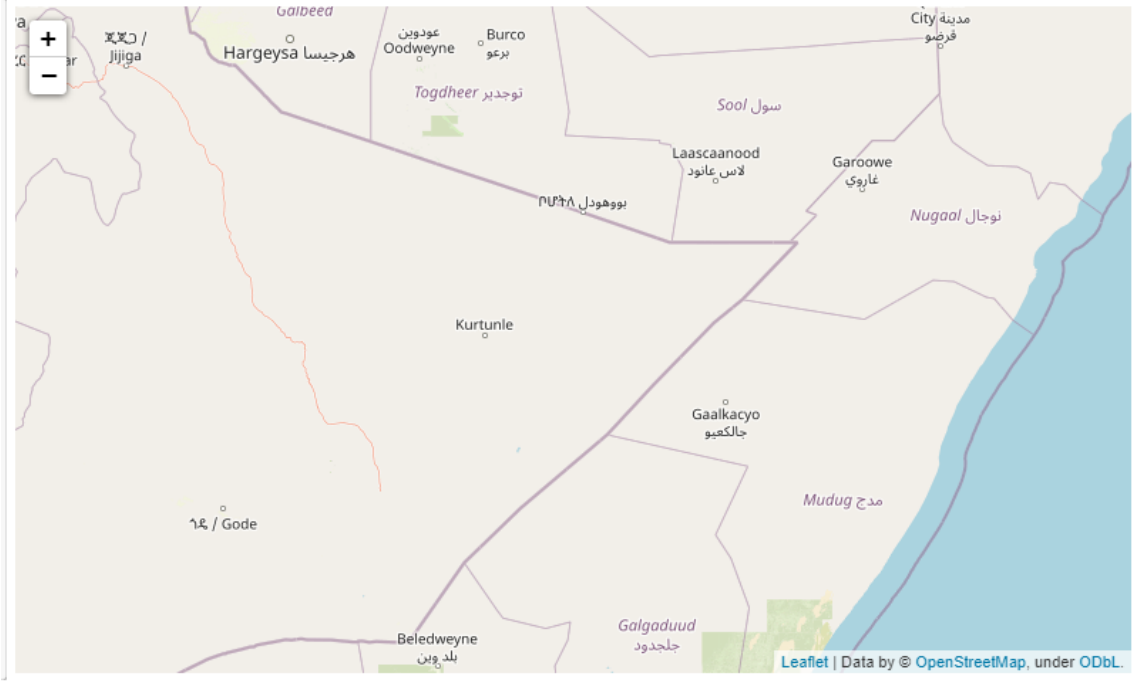
Como este proceso lo queremos repetir para cada uno de los shocks seleccionados, corregimos el código como sigue:

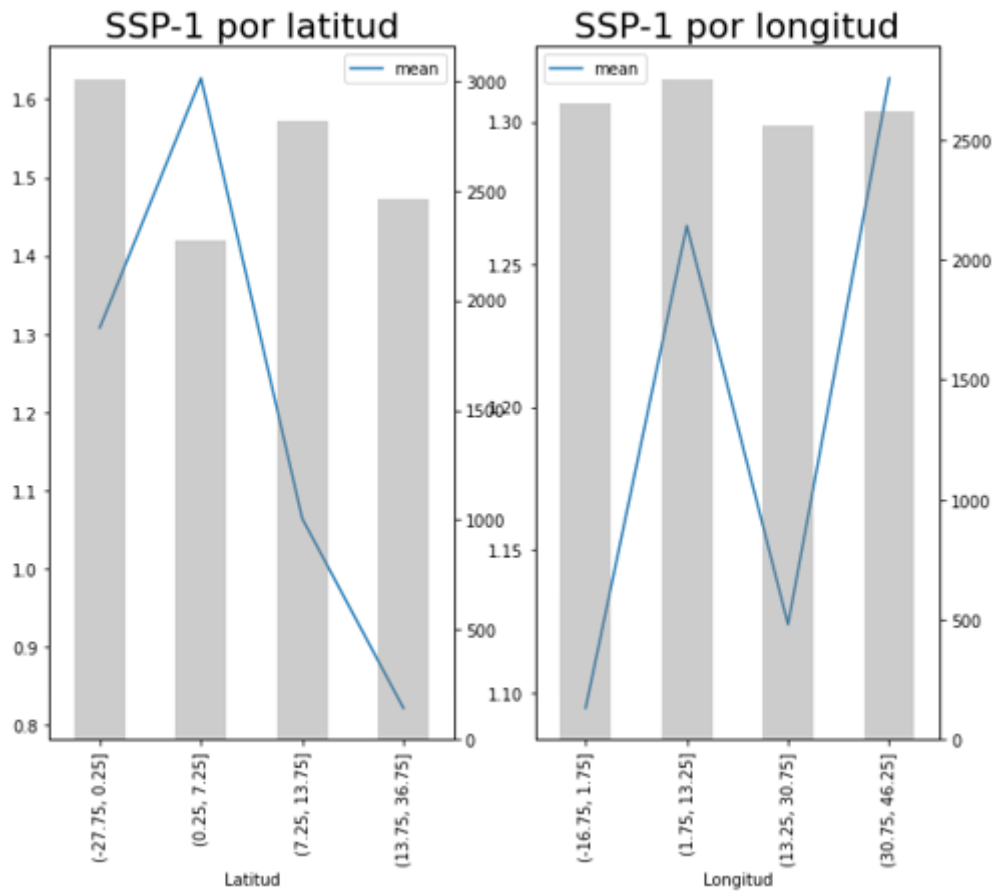
```
lista_sk2 = ['SSP+1', 'SSP-1', 'SST+1', 'SST-1', 'SSTP++1', 'SSTP+-1',
                'SSTP-+1','SSTP--1']
```

```


for sk in lista_sk2:

    intervalos_lt = np.quantile(df2['Latitud'], np.linspace(0,1,5))
    intervalos_lg = np.quantile(df2['Longitud'], np.linspace(0,1,5))

    grupos_lt = df2.groupby([pd.cut(df2['Latitud'], bins=intervalos_lt)])[sk].agg(['mean', 'size']
    grupos_lg = df2.groupby([pd.cut(df2['Longitud'], bins=intervalos_lg)])[sk].agg(['mean', 'size'

    fig, ax = plt.subplots(1,2, figsize=(10,8))
    ax[0].set_title(f'{sk} por latitud', fontsize=22)
    ax[1].set_title(f'{sk} por longitud', fontsize=22)

    grupos_lt[['mean']].plot(kind='line', ax=ax[0])
    grupos_lt['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[0])


    grupos_lg[['mean']].plot(kind='line', ax=ax[1])
    grupos_lg['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[1])
```

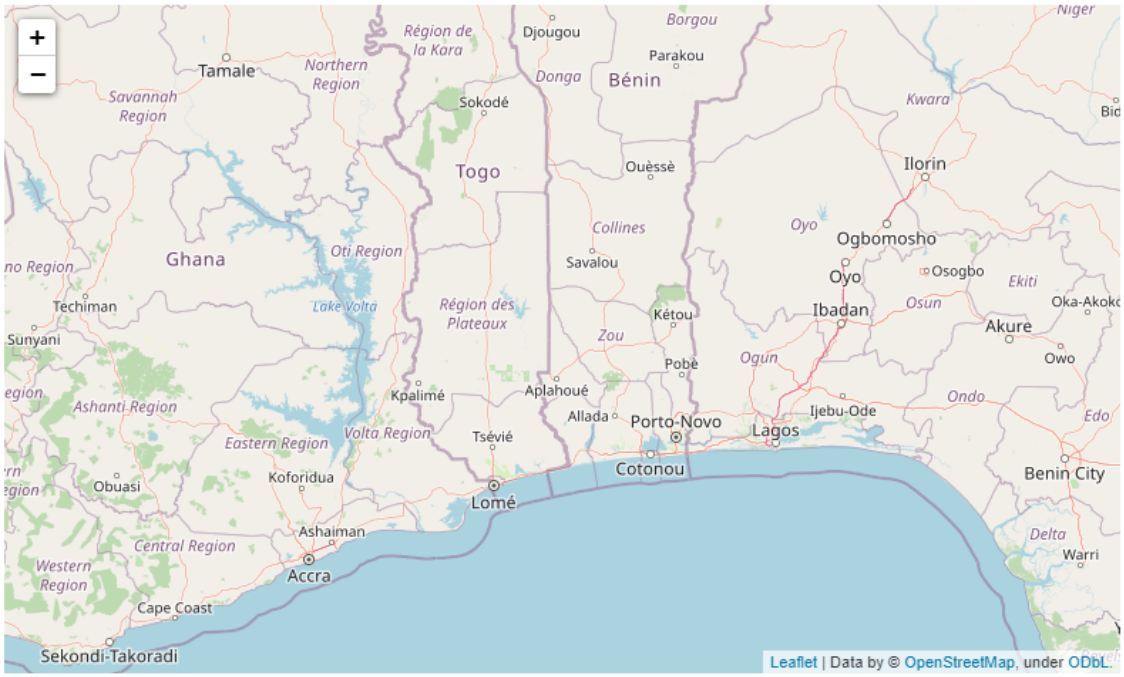


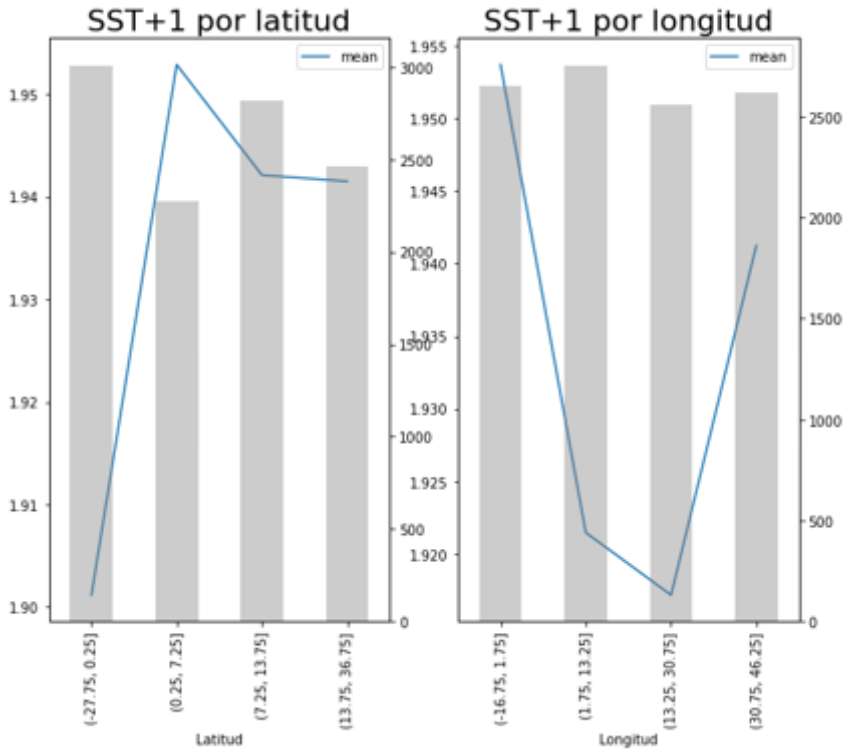

```
longitud = 46.25
latitud = 7.25

map_2 = folium.Map(location=[latitud, longitud], zoom_start=7)

map_2
```



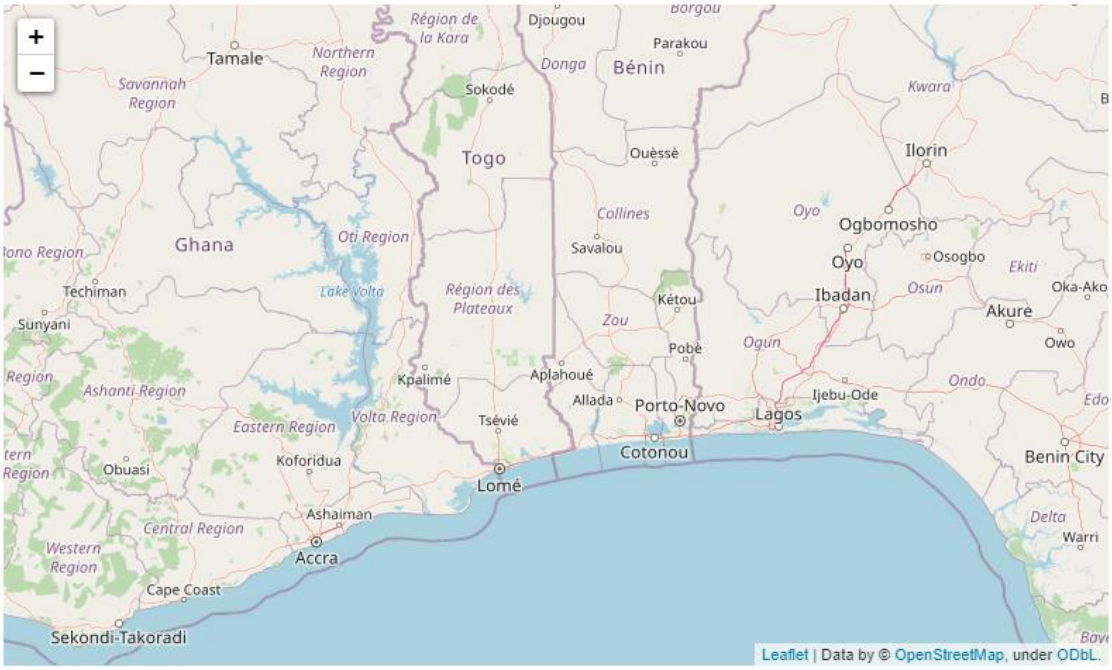


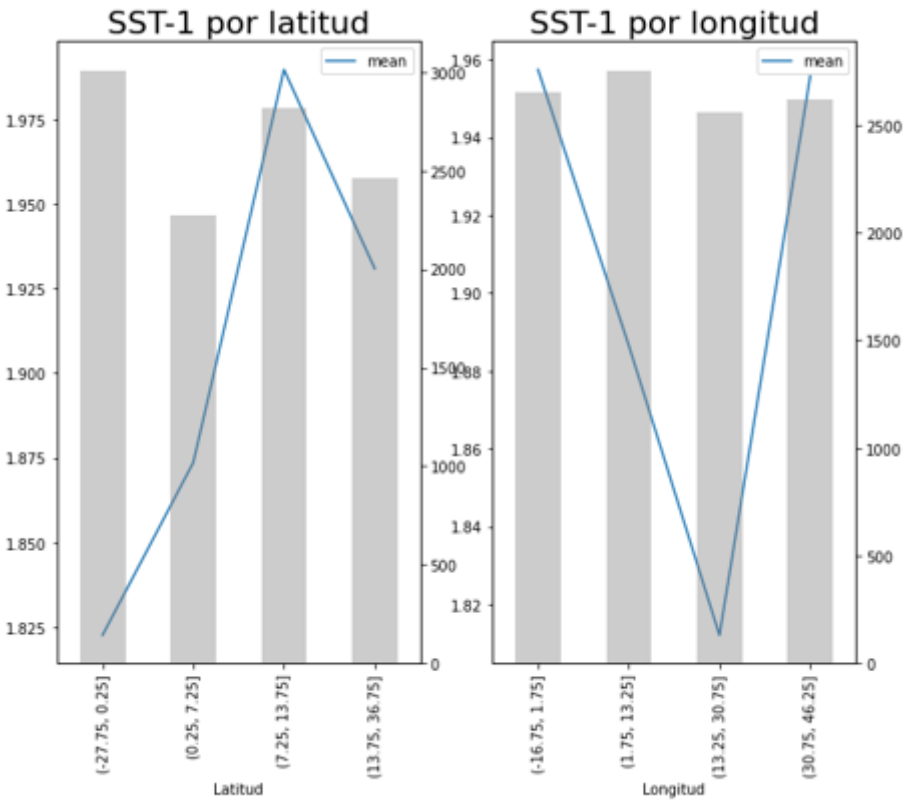

```
1 longitud = 1.75
2 latitud = 7.25
3
4 map_3 = folium.Map(location=[latitud, longitud], zoom_start=7)
5
6 map_3
```






```
longitud = 1.75
latitud = 7.25

map_3 = folium.Map(location=[latitud, longitud], zoom_start=7)

map_3
```

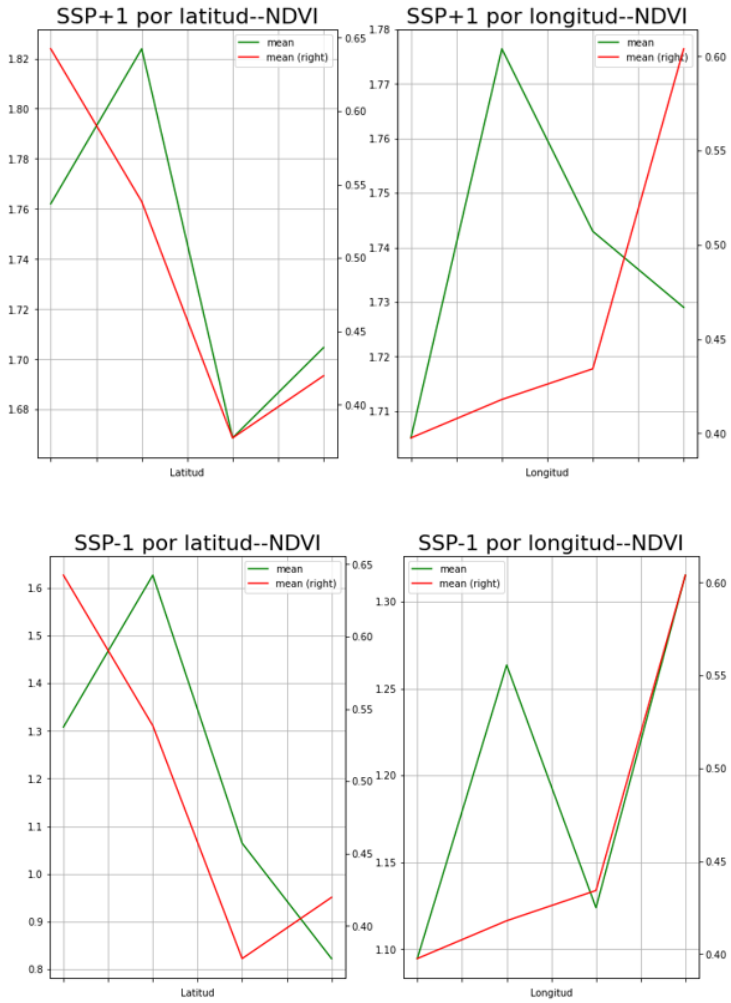


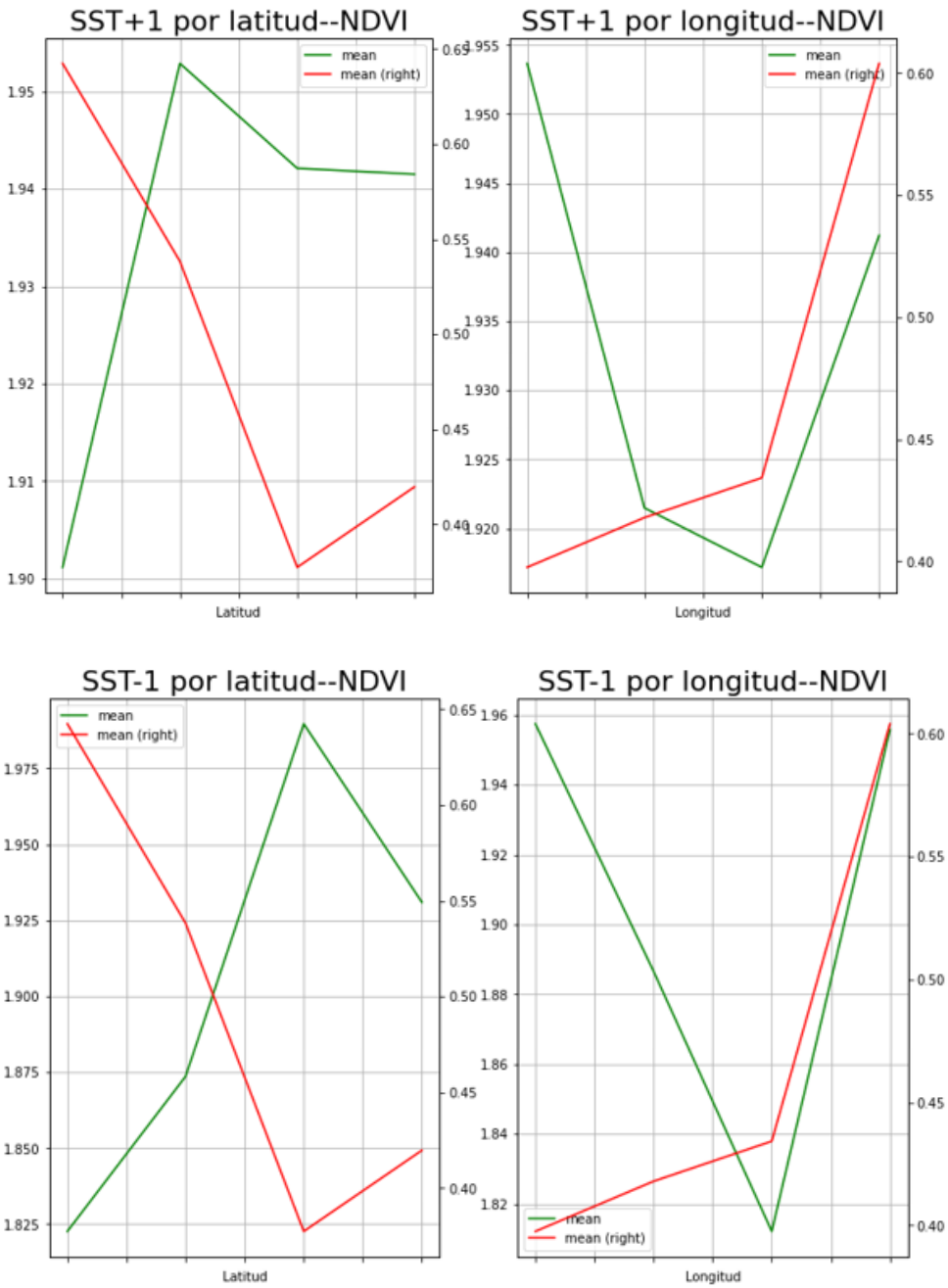



```
1 longitud = 1.75
2 latitud = 13.75
3
4 map_5 = folium.Map(location=[latitud, longitud], zoom_start=7)
5
6 map_5
```

Ahora mostraremos un gráfico muy sencillo que nos ayudará a ver esta correlación de una forma más sencilla.

```python
for sk in lista_sk2:

    intervalos_lt = np.quantile(df2['Latitud'], np.linspace(0,1,5))
    intervalos_lg = np.quantile(df2['Longitud'], np.linspace(0,1,5))

    grupos_lt = df2.groupby([pd.cut(df2['Latitud'], bins=intervalos_lt)])[sk].agg(['mean', 'size']
    grupos_lt_ndvi = df2.groupby([pd.cut(df2['Latitud'], bins=intervalos_lt)])['NDVI'].agg(['mean'

    grupos_lg = df2.groupby([pd.cut(df2['Longitud'], bins=intervalos_lg)])[sk].agg(['mean', 'size'
    grupos_lg_ndvi = df2.groupby([pd.cut(df2['Longitud'], bins=intervalos_lt)])['NDVI'].agg(['mean

    fig, ax = plt.subplots(1,2, figsize=(12,8))
    ax[0].set_title(f'{sk} por latitud--NDVI', fontsize=22)
    ax[1].set_title(f'{sk} por longitud--NDVI', fontsize=22)

    grupos_lt[['mean']].plot(kind='line', color = 'green', ax=ax[0], grid=True)
    grupos_lt_ndvi[['mean']].plot(kind='line', ax=ax[0],color = 'red', secondary_y=True)
    #grupos_lt[['size']].plot(kind='bar', color='grey', alpha=0.4, ax=ax[0])
    ax[0].grid(True)

    grupos_lg[['mean']].plot(kind='line', color = 'green', ax=ax[1], grid=True)
    grupos_lg_ndvi[['mean']].plot(kind='line', ax=ax[1], color = 'red', secondary_y=True)
    #grupos_lg[['size']].plot(kind='bar', color='grey', alpha=0.4, ax=ax[1])
    ax[1].grid(True)
```



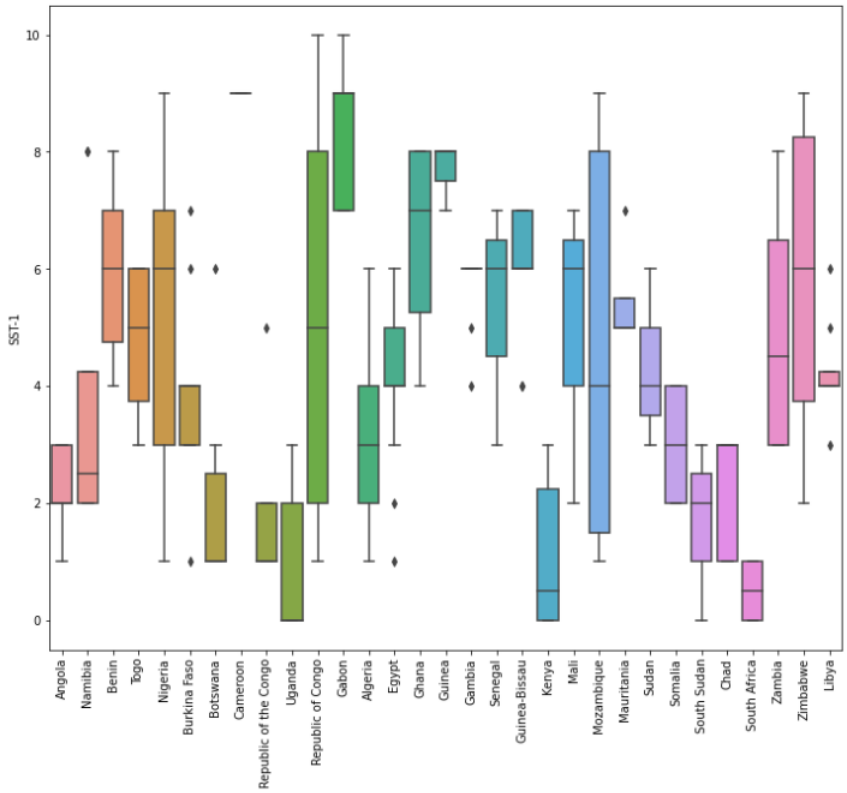


Ahora, por medio de un gráfico interactivo veremos cómo se distribuyen los valores de los Shocks climáticos en cada país en un año concreto. Esto es interesante porque a golpe de vista podemos ver si hay variación en los shocks a lo largo del tiempo. Como hay muchos shocks diferentes lo mostraremos para uno de ellos en cuatro momentos del tiempo que coinciden con los inicios de cada uno de los intervalos que hemos seleccionado.

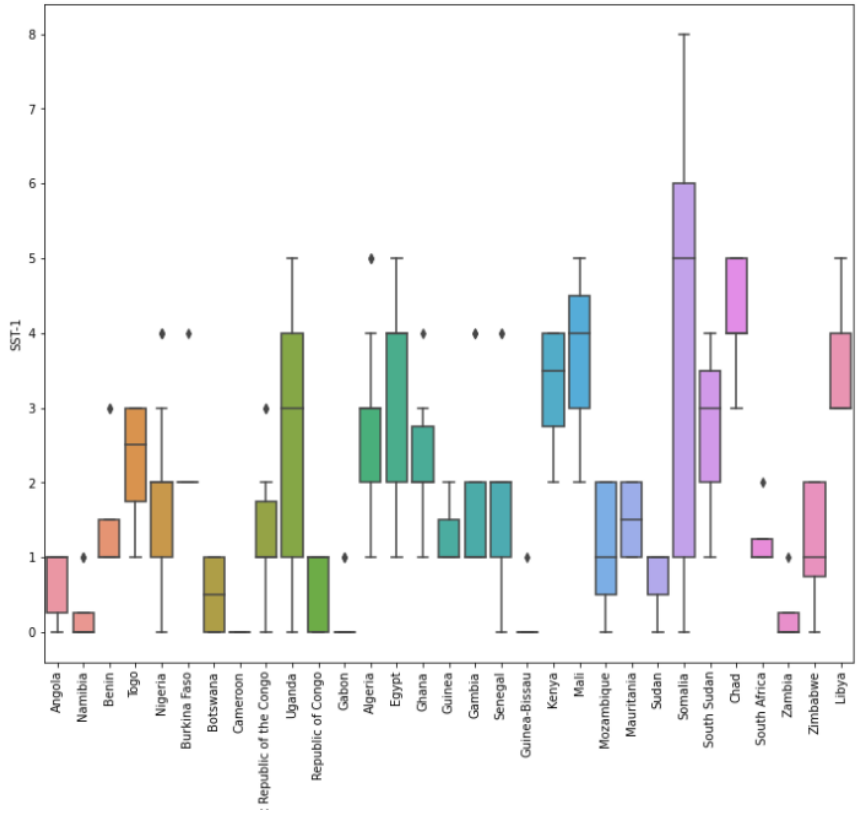
```
from ipywidgets import interact
```

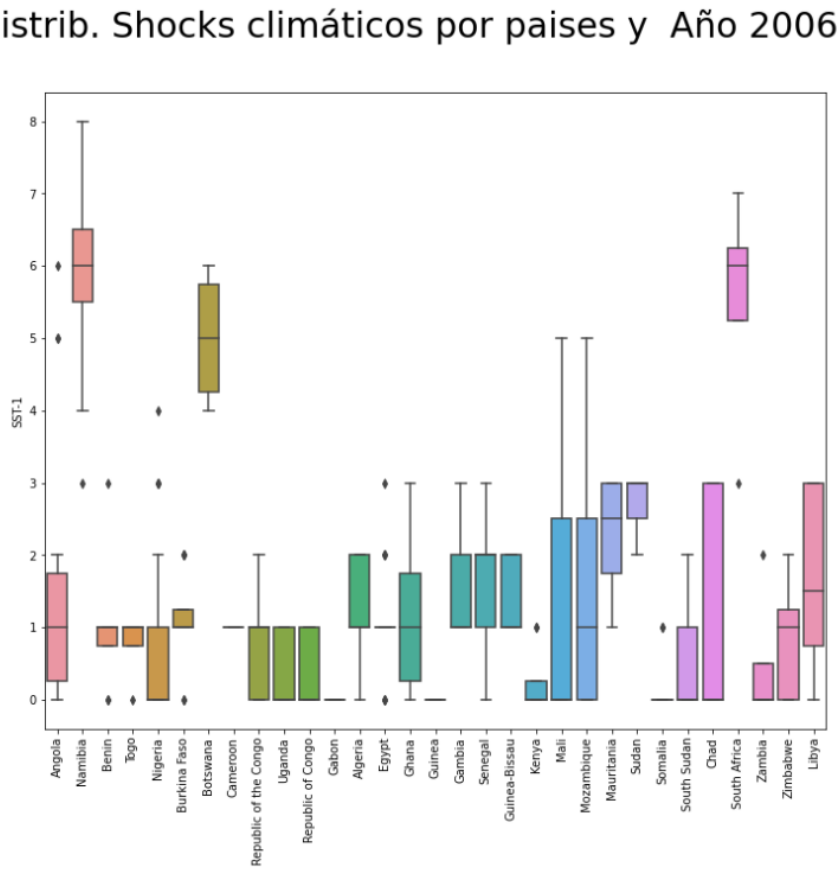
```
def distribucion(year, sk):
    df_filter = df2.loc[df2['Year']==year]

    fig, ax = plt.subplots(figsize=(12,10))
    fig.suptitle(f'Distrib. Shocks climáticos por paises y  Año {year}', fontsize=30)
    plt.xticks(rotation=90)
    return sns.boxplot(df_filter['Country'], df_filter[sk], ax = ax)
interact(distribucion, sk=lista_sk, year=years)
```

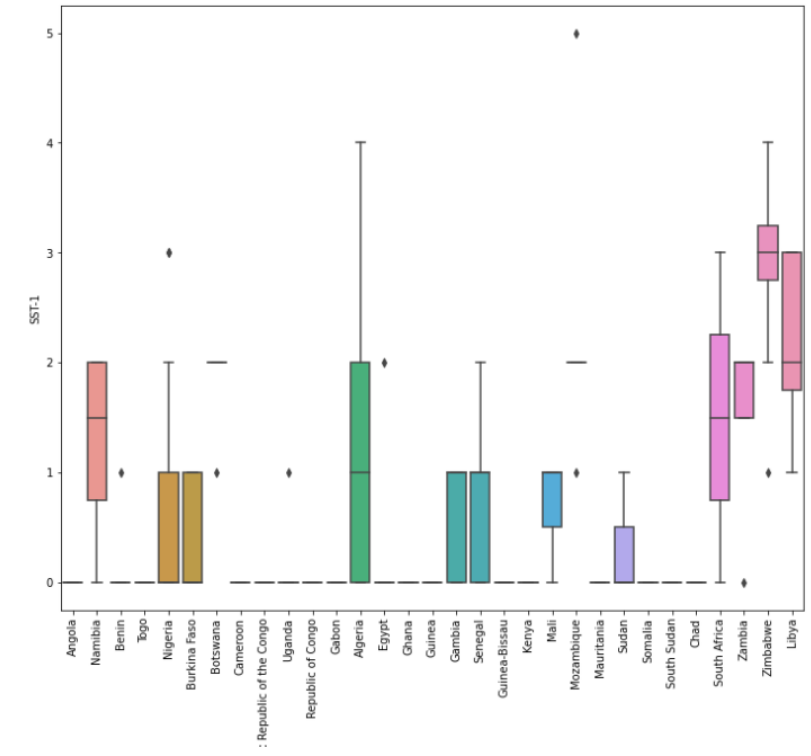
Distrib. Shocks climáticos por paises y Año 1986.0



Distrib. Shocks climáticos por paises y Año 1995







Con este apoyo gráfico se observa que es más que evidente que ha habido un cambio en los comportamientos climáticos a lo largo de estos 33 años de estudio. En el modelo de

regresión que lancemos veremos si estos cambios se manifiestan en los cambios sufridos en el NDVI.

Este mismo análisis lo podemos hacer por país a lo largo del tiempo.

```
#Ahora vamos ha hacer la evolución de las distribución de los shocks a lo largo del tiempo para ca
```

```
paises = df2['Country'].unique().tolist()
```

```
def devolver_shock(pais):
    return df2.loc[df2['Country']==pais][['SSP+1', 'SSP-1', 'SST+1', 'SST-1', 'SSTP++1', 'SSTP+-1'
                  'SSTP-+1','SSTP--1']]
```

```
devolver_shock('Angola')['SSP+1']
```

```
0      2.0
1      1.0
2      0.0
3      1.0
4      4.0
      ...
475    0.0
476    0.0
477    2.0
478    5.0
479    5.0
Name: SSP+1, Length: 448, dtype: float64
```

```
def devolver_year(pais):
    return df2.loc[df2['Country']==pais]['Year']
```

```
devolver_year('Angola')
```

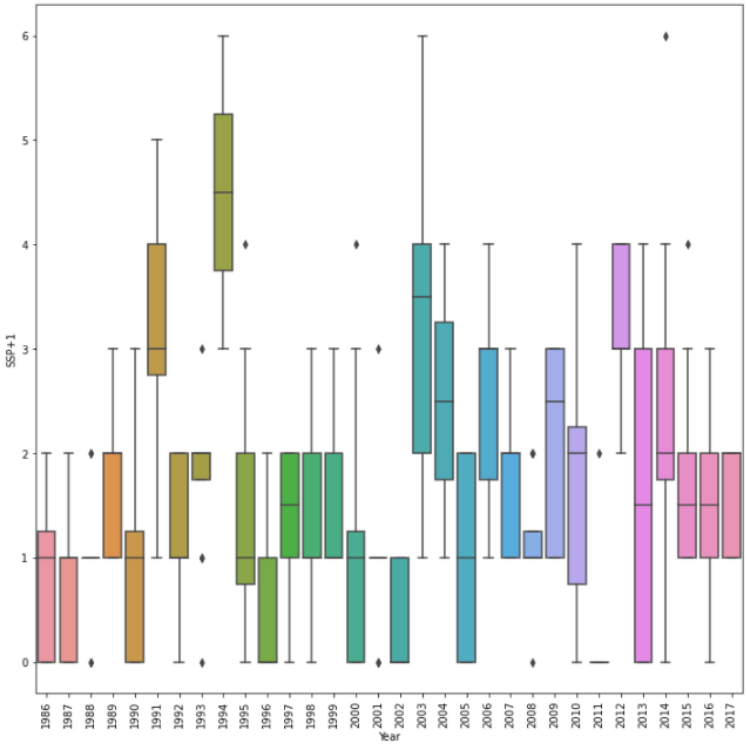
```
0      1986
1      1987
2      1988
3      1989
4      1990
       ...
475    2013
476    2014
477    2015
478    2016
479    2017
Name: Year, Length: 448, dtype: int64
```

```
def distribucion_paises_sk(p, sk):
    fig, ax = plt.subplots(figsize=(12,12))
    fig.suptitle(f'Distribución del {sk} del pais {p}', fontsize=28)
    plt.xticks(rotation=90)

    return sns.boxplot(devolver_year(p), devolver_shock(p)[sk])
interact(distribucion_paises_sk, p=paises, sk=lista_sk2)
```

A continuación, muestro la evolución de un mismo shock en dos países diferentes. Se observa claramente que la evolución del mismo shock es muy distinta, teniendo un comportamiento mucho más estable en Nigeria que en Burkina Faso. De modo que se vuelve a poner de manifiesto que la presencia de shocks no sólo varía con el tiempo sino que también lo hace con las coordenadas geográficas.

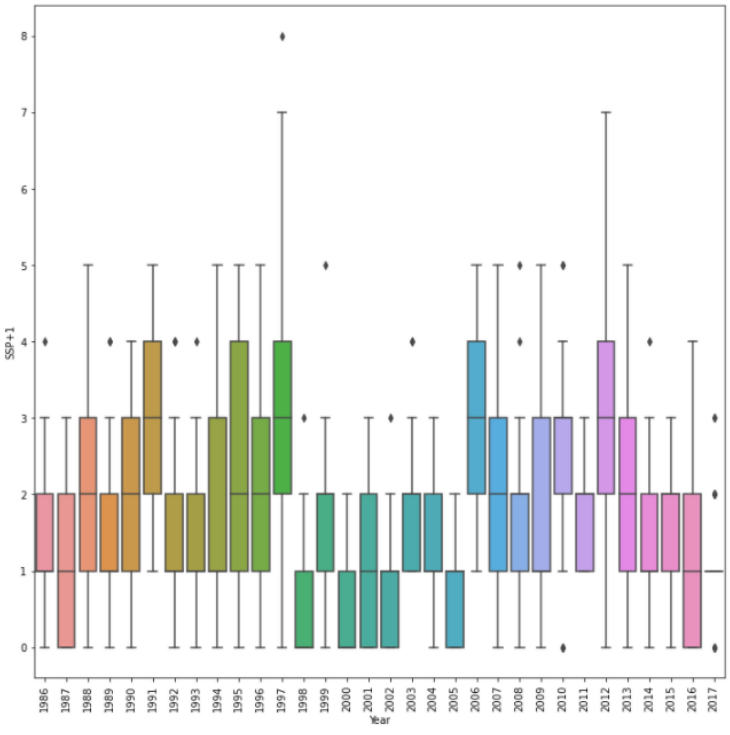
p Burkina Faso
sk SSP+1

Distribución del SSP+1 del pais Burkina Faso



p Nigeria
sk SSP+1

Distribución del SSP+1 del pais Nigeria



A continuación, estudiaremos el comportamiento de nuestra variable objetivo en lo que se refiere al estudio de la productividad del suelo a través de la variable 'C_NDVI'. En este caso nos apoyaremos en la tabla 'Panel_DATA_final_por_paises' que al principio de este apéndice vimos cómo la creamos.

```
df3 = pd.read_csv('Panel_DATA_final_por_paises').drop('Unnamed: 0', axis=1)
```

```
df3
```

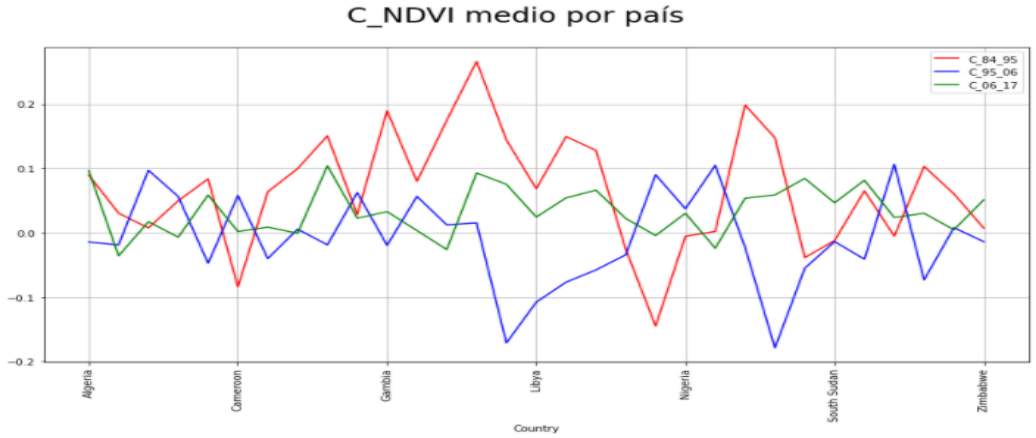
	Country	Longitud	Latitud	C_84_95	C_95_06	C_06_17
0	Angola	13.75	-7.75	0.195	-0.050	-0.060
1	Angola	14.25	-11.75	0.083	-0.044	-0.025
2	Angola	17.25	-11.75	0.049	-0.114	0.035
3	Angola	14.75	-7.75	0.010	0.012	-0.095
4	Angola	15.25	-9.75	0.035	0.074	-0.111
...	...	...	...	...	...	...
293	Libya	14.25	32.75	0.122	-0.153	0.016
294	Libya	11.75	32.75	0.030	-0.018	-0.001
295	Libya	20.75	31.75	0.151	-0.291	0.048
296	Libya	13.25	32.75	-0.005	-0.029	-0.005
297	Libya	12.75	27.75	0.051	-0.058	0.020

298 rows × 6 columns

Este dataset cuenta con 298 registros que son las 'Crop_Areas' o áreas de cultivo para las que la FAO tiene datos de productividad del suelo.

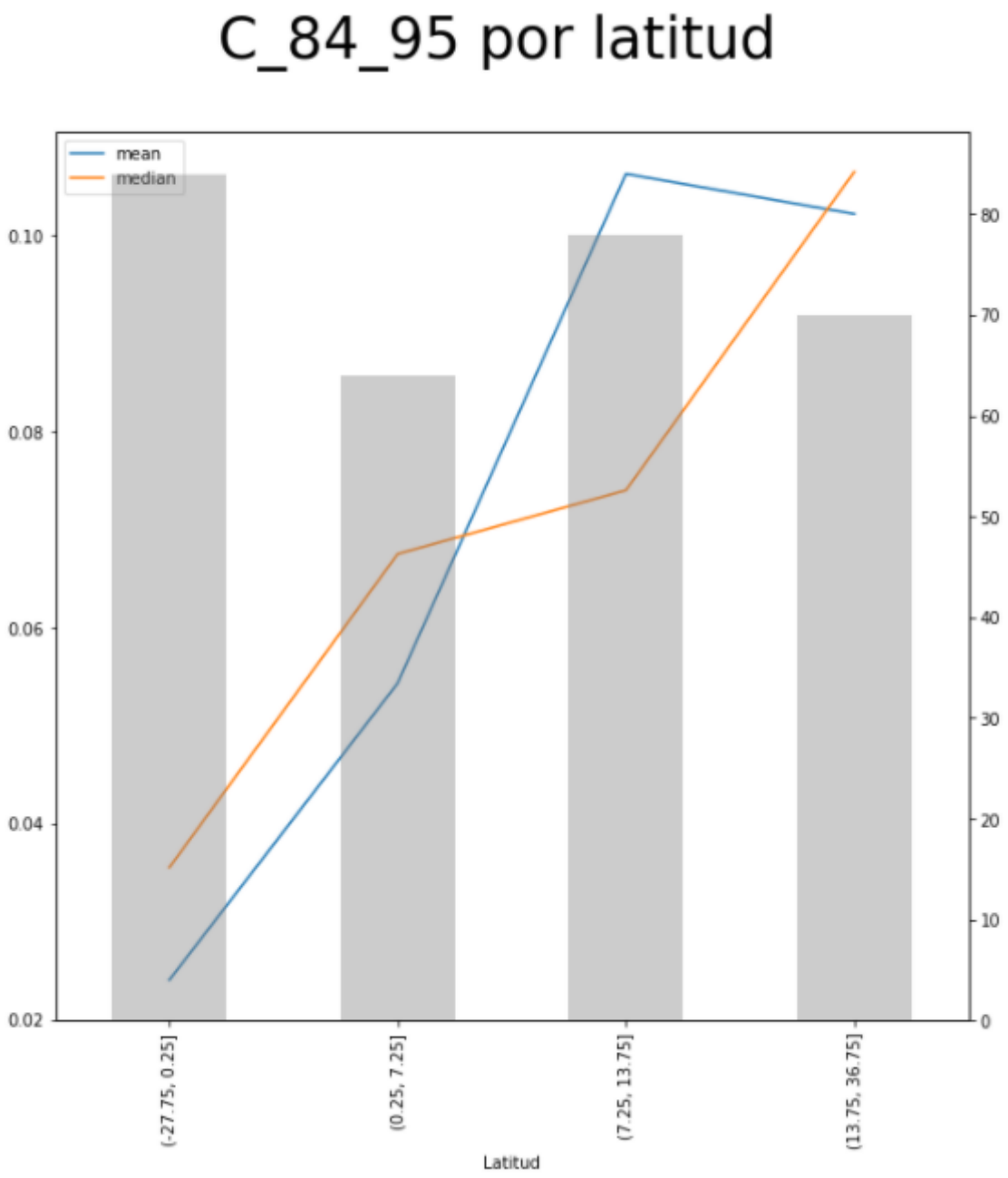
```
fig, ax = plt.subplots(figsize=(15,7))
fig.suptitle("C_NDVI medio por país ", fontsize=25)
df3.groupby(['Country']).mean()['C_84_95'].plot(ax = ax, color='red', legend=True)
df3.groupby(['Country']).mean()['C_95_06'].plot(ax = ax, color='blue', legend=True)
df3.groupby(['Country']).mean()['C_06_17'].plot(ax = ax, color='green', legend=True)

ax.grid(True)
plt.xticks(rotation=90)
```

Mediante estas líneas de código observamos cómo ha sido la evolución en el cambio del NDVI por países en cada uno de los tres intervalos de tiempo que hemos escogido.



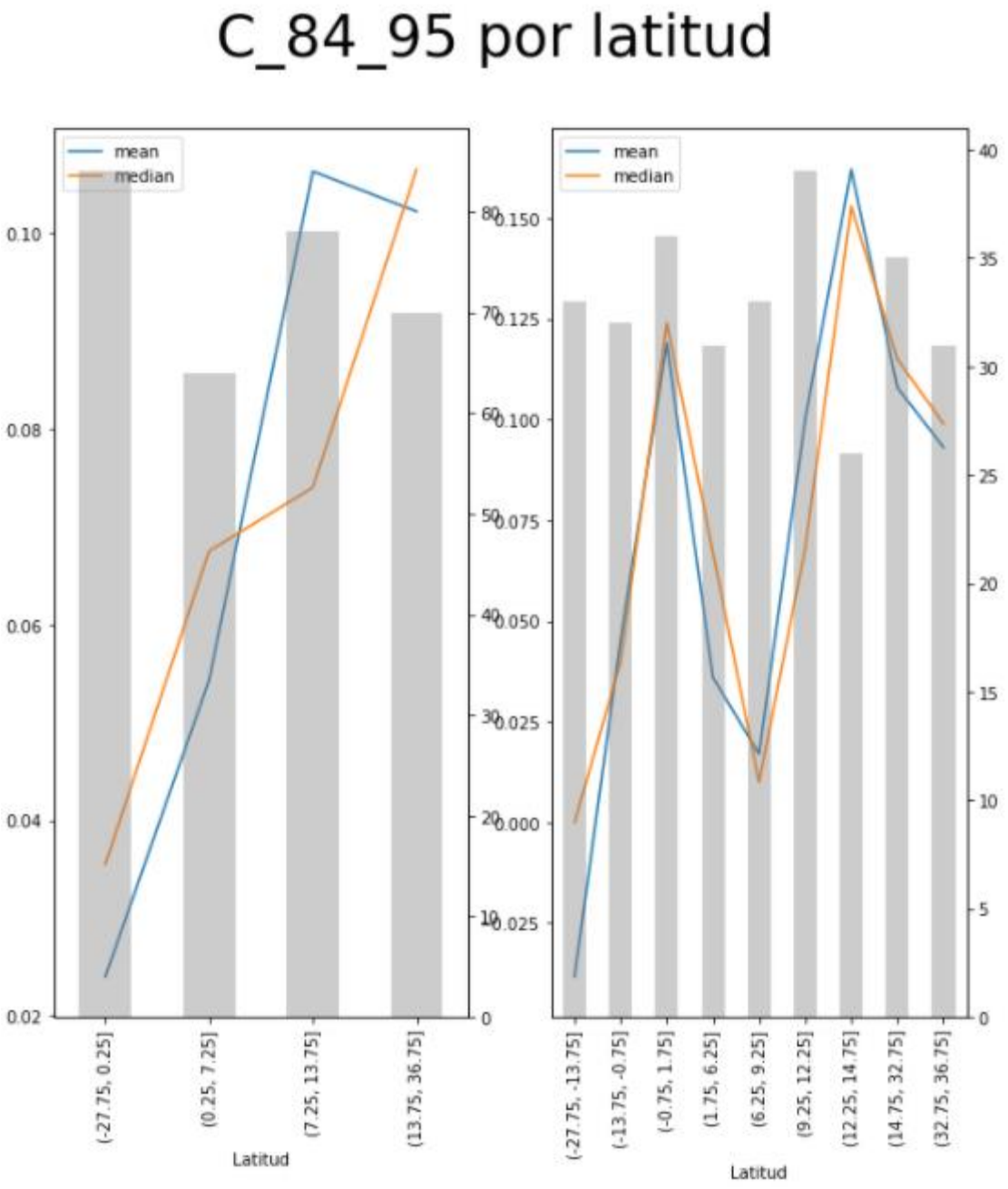
Al igual que hicimos con la variable NDVI, ahora estudiaremos los C_NDVI por intervalo según latitud y longitud. Volvemos a hacer lo mismo que en el caso anterior donde primero definimos unos intervalos y luego calculamos los valores medios del cambio en el NDVI por intervalos. Las barras lo que nos indica es el número de observaciones que tenemos para cada intervalo.

```
intervalos = np.quantile(df3['Latitud'], np.linspace(0,1,5))
grupos = df3.groupby([pd.cut(df3['Latitud'], bins=intervalos)])['C_84_95'].agg(['mean', 'median',

fig, ax = plt.subplots(figsize=(10,10))
fig.suptitle('C_84_95 por latitud', fontsize=35)

grupos[['mean', 'median']].plot(kind='line', ax=ax)
grupos['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True)
```

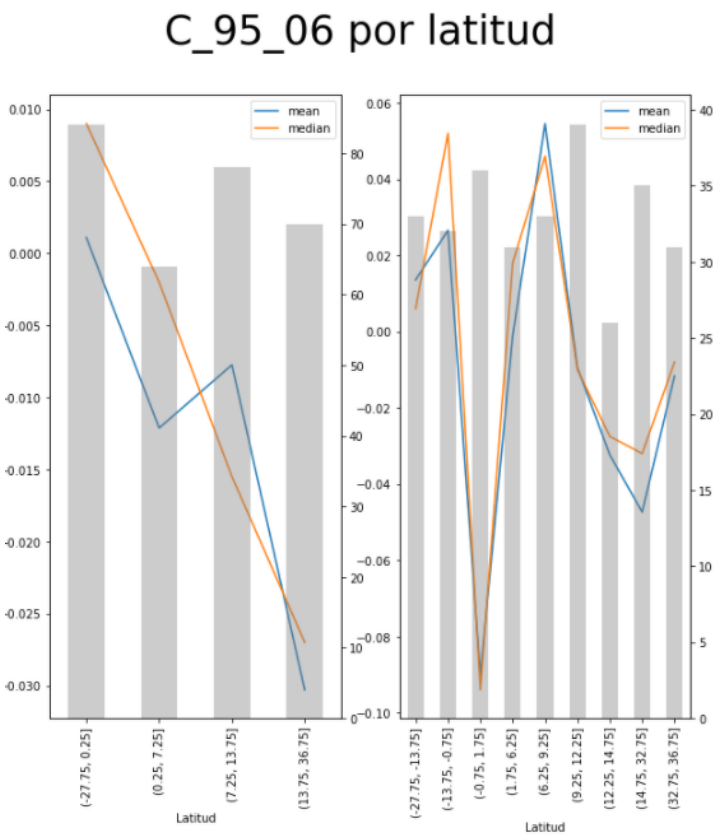


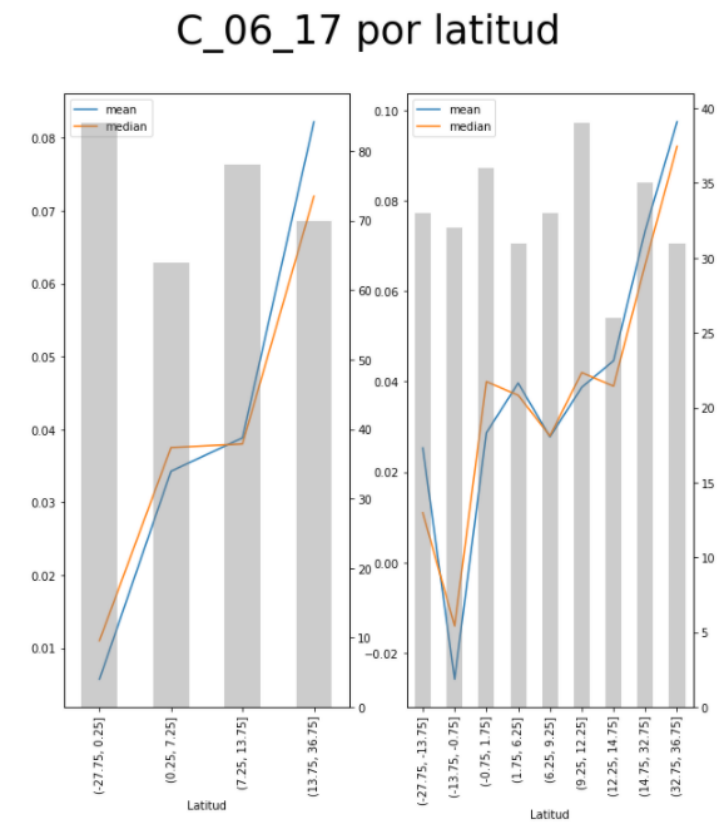
Con este gráfico se observa una tendencia bastante clara, parece que hay una correlación positiva entre la variable cambio y la variable longitud en el primer intervalo de estudio. Pero… qué ocurre si en lugar de construir cuatro intervalos construimos nueve.

```
intervalos = np.quantile(df3['Latitud'], np.linspace(0,1,5))
intervalos2 = np.quantile(df3['Latitud'], np.linspace(0,1,10))

grupos = df3.groupby([pd.cut(df3['Latitud'], bins=intervalos)])['C_84_95'].agg(['mean', 'median',
grupos2 = df3.groupby([pd.cut(df3['Latitud'], bins=intervalos2)])['C_84_95'].agg(['mean', 'median'

fig, ax = plt.subplots(1,2, figsize=(10,10))
fig.suptitle('C_84_95 por latitud', fontsize=35)

grupos[['mean', 'median']].plot(kind='line', ax=ax[0])
grupos['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[0])

grupos2[['mean', 'median']].plot(kind='line', ax=ax[1])
grupos2['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[1])
```



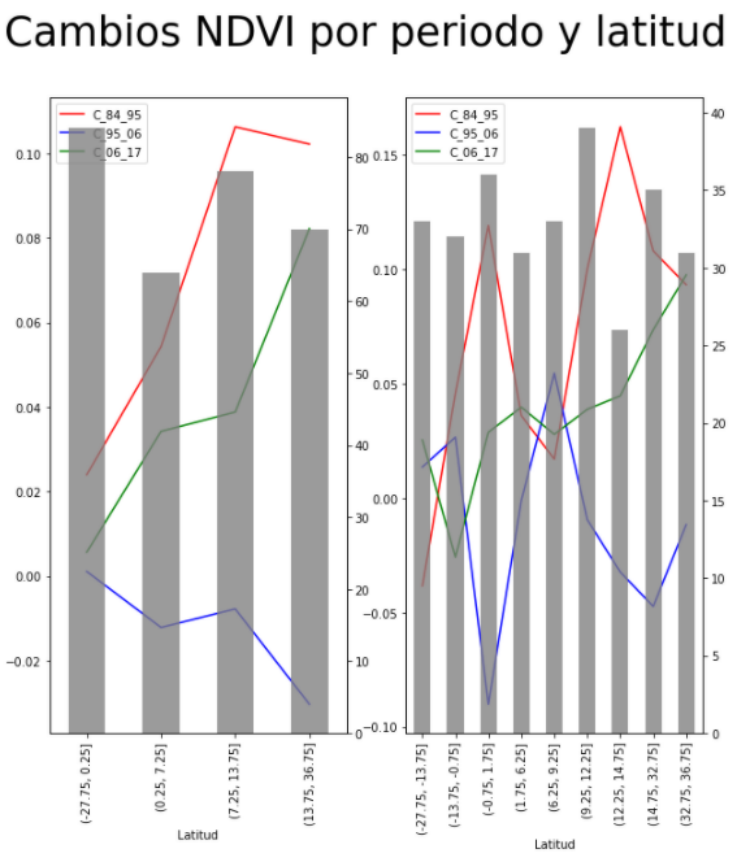
En este caso la tendencia no está tan definida como en el caso de establecer sólo cuatro intervalos. Hacemos lo mismo para los siguientes intervalos:





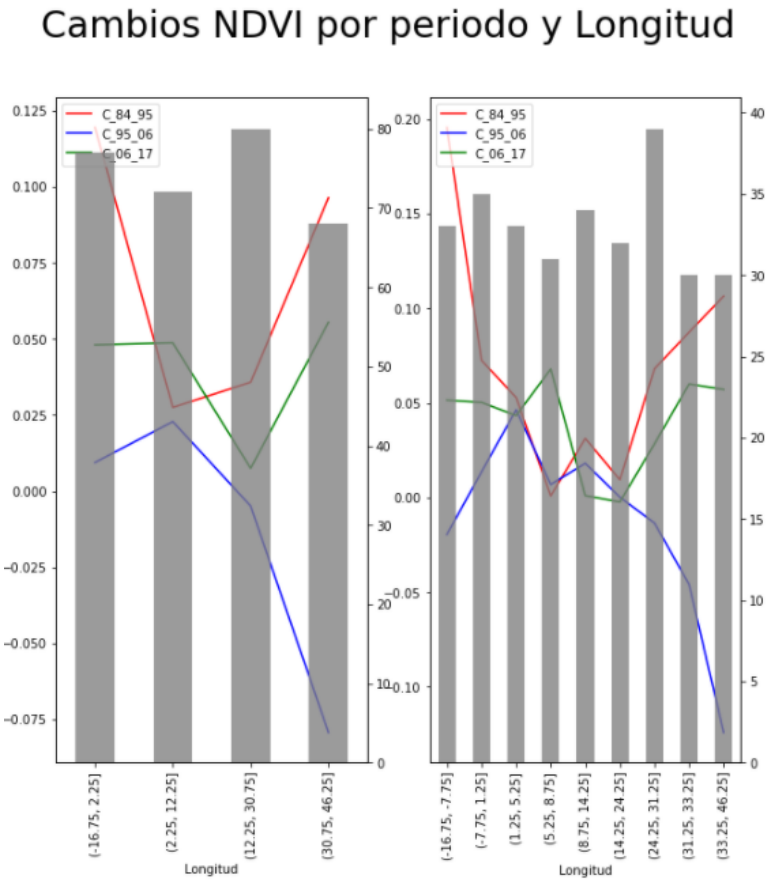
En los otros dos intervalos de estudio ocurre exactamente lo mismo, las tendencias con nueve intervalos no están tan claras como con cuatro. Para compararlos con más comodidad incorporamos todas las evoluciones en un solo gráfico.

```
intervalos = np.quantile(df3['Latitud'], np.linspace(0,1,5))
intervalos2 = np.quantile(df3['Latitud'], np.linspace(0,1,10))


grupos_a = df3.groupby([pd.cut(df3['Latitud'], bins=intervalos)])['C_84_95'].agg(['mean', 'median'
grupos2_a = df3.groupby([pd.cut(df3['Latitud'], bins=intervalos2)])['C_84_95'].agg(['mean', 'media

grupos_b = df3.groupby([pd.cut(df3['Latitud'], bins=intervalos)])['C_95_06'].agg(['mean', 'median'
grupos2_b = df3.groupby([pd.cut(df3['Latitud'], bins=intervalos2)])['C_95_06'].agg(['mean', 'media

grupos_c = df3.groupby([pd.cut(df3['Latitud'], bins=intervalos)])['C_06_17'].agg(['mean', 'median'
grupos2_c = df3.groupby([pd.cut(df3['Latitud'], bins=intervalos2)])['C_06_17'].agg(['mean', 'media

fig, ax = plt.subplots(1,2, figsize=(10,10))
fig.suptitle('Cambios NDVI por periodo y latitud', fontsize=35)

grupos_a['mean'].plot(kind='line', ax=ax[0], color='red', label = "C_84_95")
grupos_a['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[0])
grupos_b['mean'].plot(kind='line', ax=ax[0], color='blue', label = "C_95_06")
grupos_b['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[0])
grupos_c['mean'].plot(kind='line', ax=ax[0], color='green', label = "C_06_17")
grupos_c['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[0])

ax[0].legend(loc=2)


grupos2_a['mean'].plot(kind='line', ax=ax[1], color='red', label = "C_84_95")
grupos2_a['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[1])
grupos2_b['mean'].plot(kind='line', ax=ax[1], color='blue', label = "C_95_06")
grupos2_b['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[1])
grupos2_c['mean'].plot(kind='line', ax=ax[1], color='green', label = "C_06_17")
grupos2_c['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[1])
ax[1].legend(loc=2)
```



Hacemos lo mismo por 'Longitud' y en este caso sólo presentamos en cuadro que contiene las tres evoluciones simultáneamente.

```
intervalos = np.quantile(df3['Longitud'], np.linspace(0,1,5))
intervalos2 = np.quantile(df3['Longitud'], np.linspace(0,1,10))


grupos_a = df3.groupby([pd.cut(df3['Longitud'], bins=intervalos)])['C_84_95'].agg(['mean', 'median
grupos2_a = df3.groupby([pd.cut(df3['Longitud'], bins=intervalos2)])['C_84_95'].agg(['mean', 'medi

grupos_b = df3.groupby([pd.cut(df3['Longitud'], bins=intervalos)])['C_95_06'].agg(['mean', 'median
grupos2_b = df3.groupby([pd.cut(df3['Longitud'], bins=intervalos2)])['C_95_06'].agg(['mean', 'medi

grupos_c = df3.groupby([pd.cut(df3['Longitud'], bins=intervalos)])['C_06_17'].agg(['mean', 'median
grupos2_c = df3.groupby([pd.cut(df3['Longitud'], bins=intervalos2)])['C_06_17'].agg(['mean', 'medi

fig, ax = plt.subplots(1,2, figsize=(10,10))
fig.suptitle('Cambios NDVI por periodo y Longitud', fontsize=30)

grupos_a['mean'].plot(kind='line', ax=ax[0], color='red', label = "C_84_95")
grupos_a['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[0])
grupos_b['mean'].plot(kind='line', ax=ax[0], color='blue', label = "C_95_06")
grupos_b['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[0])
grupos_c['mean'].plot(kind='line', ax=ax[0], color='green', label = "C_06_17")
grupos_c['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[0])

ax[0].legend(loc=2)


grupos2_a['mean'].plot(kind='line', ax=ax[1], color='red', label = "C_84_95")
grupos2_a['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[1])
grupos2_b['mean'].plot(kind='line', ax=ax[1], color='blue', label = "C_95_06")
grupos2_b['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[1])
grupos2_c['mean'].plot(kind='line', ax=ax[1], color='green', label = "C_06_17")
grupos2_c['size'].plot(kind='bar', color='grey', alpha=0.4, secondary_y=True, ax=ax[1])
ax[1].legend(loc=2)
```



Ahora queremos responder a la pregunta: cómo ha sido el crecimiento por término medio en cada uno de los intervalos por países. Para responder a esto hacemos una reestructuración de la tabla anterior como se explica a continuación.

```
df3_cambio_index = df3.set_index('Country')
df3_cambio_index
```

Country	Longitud	Latitud	C_84_95	C_95_06	C_06_17
Angola	13.75	-7.75	0.195	-0.050	-0.060
Angola	14.25	-11.75	0.083	-0.044	-0.025
Angola	17.25	-11.75	0.049	-0.114	0.035
Angola	14.75	-7.75	0.010	0.012	-0.095
Angola	15.25	-9.75	0.035	0.074	-0.111
...	...	...	...	...	...
Libya	14.25	32.75	0.122	-0.153	0.016
Libya	11.75	32.75	0.030	-0.018	-0.001
Libya	20.75	31.75	0.151	-0.291	0.048
Libya	13.25	32.75	-0.005	-0.029	-0.005
Libya	12.75	27.75	0.051	-0.058	0.020

298 rows × 5 columns

```
def devolver_media_crecimiento(country):
    return df3.loc[df3['Country']==country]['C_84_95'].values.mean()
```

```
devolver_media_crecimiento('Angola')
```

```
0.03042857142857144
```

```
contenedor_resultados = []
for index, row in df3.iterrows():
    pais = row['Country']

    val = devolver_media_crecimiento(pais)
    #print(val)
    contenedor_resultados.append(val)
```

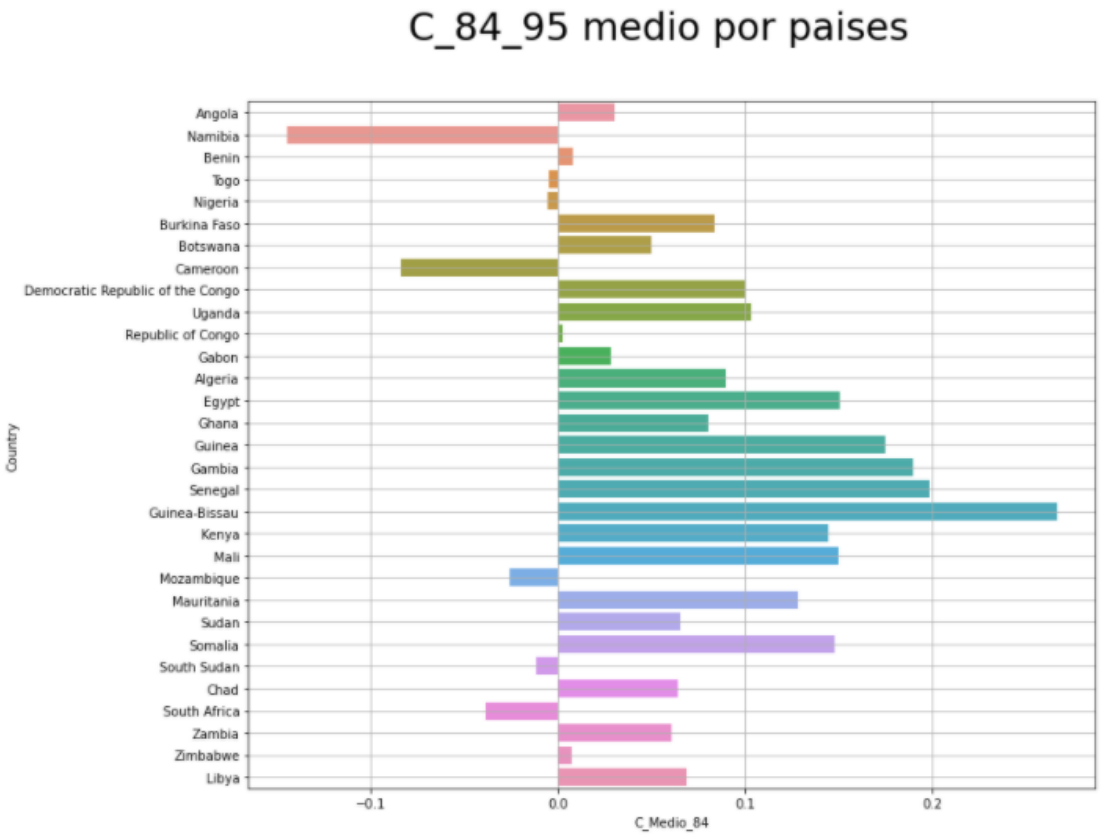
```
df4 = df3
df4['C_Medio_84'] = contenedor_resultados
```

```
df4 = df4.drop(['Latitud', 'Longitud', 'C_84_95', 'C_95_06', 'C_06_17'], axis=1)
df4
```

	Country	C_Medio_84
0	Angola	0.030429
1	Angola	0.030429
2	Angola	0.030429
3	Angola	0.030429
4	Angola	0.030429
...	...	...
293	Libya	0.068500

```
df4 = df4.drop_duplicates()
```

```
fig, ax = plt.subplots(figsize=(12,10))
fig.suptitle('C_84_95 medio por paises', fontsize=30)
sns.barplot(x=df4['C_Medio_84'], y=df4['Country'], ax = ax)
ax.grid(True)
```



Este gráfico es interesante porque nos hacemos una idea inmediata de que los cambios en casi la totalidad de los países fueron positivos. Ahora desarrollaremos un poco el código para entender cuántas variaciones negativas hay por periodo de estudio.

```
def devolver_media_crecimiento2(pais):
    return df3.loc[df3['Country']==pais][['C_84_95', 'C_95_06', 'C_06_17']]
```

```
devolver_media_crecimiento2('Angola')['C_84_95'].mean()
```

```
0.03042857142857144
```

```
contenedor_84_95 = []
contenedor_95_06 = []
contenedor_06_17 = []
contenedor_medias = []

for index, row in df3.iterrows():
    pa = row['Country']

    val1 = devolver_media_crecimiento2(pa)['C_84_95'].mean()
    val2 = devolver_media_crecimiento2(pa)['C_95_06'].mean()
    val3 = devolver_media_crecimiento2(pa)['C_06_17'].mean()

    print(pa, val1, val2, val3)
    contenedor_medias.append([pa, val1, val2, val3])
```

```
df_cambios_medios = pd.DataFrame(contenedor_medias, columns=['Country', 'C_84_95_m', 'C_95_06_m',
df_cambios_medios
```

	Country	C_84_95_m	C_95_06_m	C_06_17_m
0	Angola	0.030429	-0.018786	-0.035929
1	Angola	0.030429	-0.018786	-0.035929

```
#A continuación lo que haremos es un cambio de indice
df_cambios_medios_indice = df_cambios_medios.set_index('Country')
df_cambios_medios_indice
```

Country	C_84_95_m	C_95_06_m	C_06_17_m
Angola	0.030429	-0.018786	-0.035929
Angola	0.030429	-0.018786	-0.035929
Angola	0.030429	-0.018786	-0.035929
Angola	0.030429	-0.018786	-0.035929
Angola	0.030429	-0.018786	-0.035929
...	...	...	...
Libya	0.068500	-0.107750	0.024500
Libya	0.068500	-0.107750	0.024500

```
df_cambios_medios_indice = df_cambios_medios_indice.drop_duplicates()
```

```
df_cambios_medios_indice
```

Country	C_84_95_m	C_95_06_m	C_06_17_m
Angola	0.030429	-0.018786	-0.035929
Namibia	-0.144875	0.090250	-0.004250
Benin	0.007556	0.097222	0.017333
Togo	-0.005200	0.106600	0.023800
Nigeria	-0.005417	0.037306	0.030306
Burkina Faso	0.083833	-0.047417	0.058750
Botswana	0.049800	0.056400	-0.006800
Cameroon	-0.084000	0.058000	0.002000
Democratic Republic of the Congo	0.099750	0.005625	-0.000750
Uganda	0.103250	-0.073156	0.030375
Republic of Congo	0.002143	0.104857	-0.023857
Gabon	0.028400	0.062500	0.022700
Algeria	0.089788	-0.014424	0.096667
Egypt	0.150667	-0.018733	0.104133
Ghana	0.079889	0.056667	0.003556
Guinea	0.174667	0.012333	-0.026000
Gambia	0.189833	-0.019500	0.032833
Senegal	0.198615	-0.022769	0.053769
Guinea-Bissau	0.266333	0.015500	0.093000
Kenya	0.144333	-0.171167	0.075333
Mali	0.149667	-0.076667	0.054333
Mozambique	-0.025714	-0.034143	0.022286
Mauritania	0.128000	-0.057750	0.066250
Sudan	0.065333	-0.041000	0.081667
Somalia	0.147667	-0.178889	0.058889
South Sudan	-0.012000	-0.013667	0.046667
Chad	0.063750	-0.040000	0.008875
South Africa	-0.038500	-0.055000	0.084500
Zambia	0.060500	0.008000	0.004750
Zimbabwe	0.007286	-0.013857	0.051286
Libya	0.068500	-0.107750	0.024500

```
#contar las variaciones negativas
contador_1 = []
contador_2 = []
contador_3 = []
for index, row in df_cambios_medios_indice.iterrows():
    a = row['C_84_95_m']
    b = row['C_95_06_m']
    c = row['C_06_17_m']
    if a < 0:
        contador_1.append(a)
    if b < 0:
        contador_2.append(b)
    if c < 0:
        contador_3.append(c)
```

```
dic = {'C_Negativos_84_95': len(contador_1),
       'C_Negativos_95_06': len(contador_2),
       'C_Negativos_06,17': len(contador_3)}
```

```
df_var_neg = pd.DataFrame.from_dict(dic, orient='index')
```

```
df_var_neg = df_var_neg.rename(columns={0:'Var_Negativas'})
```

```
df_var_neg
```

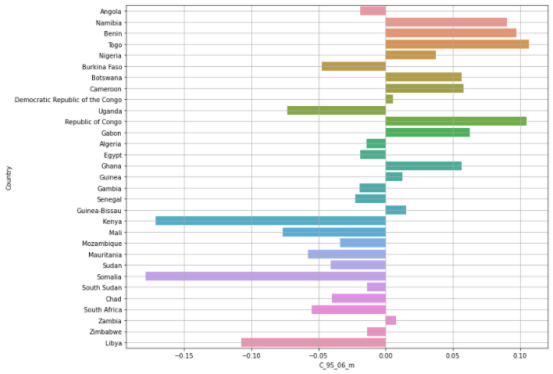
	Var_Negativas
C_Negativos_84_95	7
C_Negativos_95_06	18
C_Negativos_06,17	6

```
df_var_neg['Porcentaje'] = df_var_neg['Var_Negativas'].apply(lambda x: x*100/31)
```

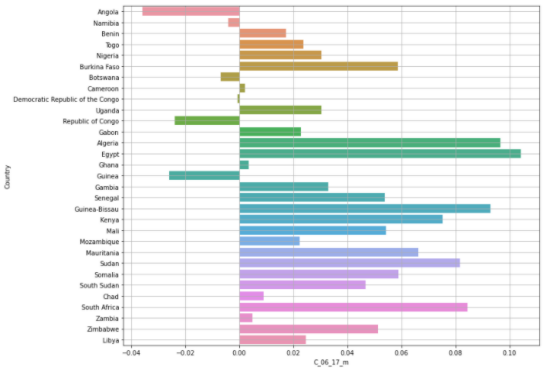
```
df_var_neg
```

	Var_Negativas	Porcentaje
C_Negativos_84_95	7	22.580645
C_Negativos_95_06	18	58.064516
C_Negativos_06,17	6	19.354839

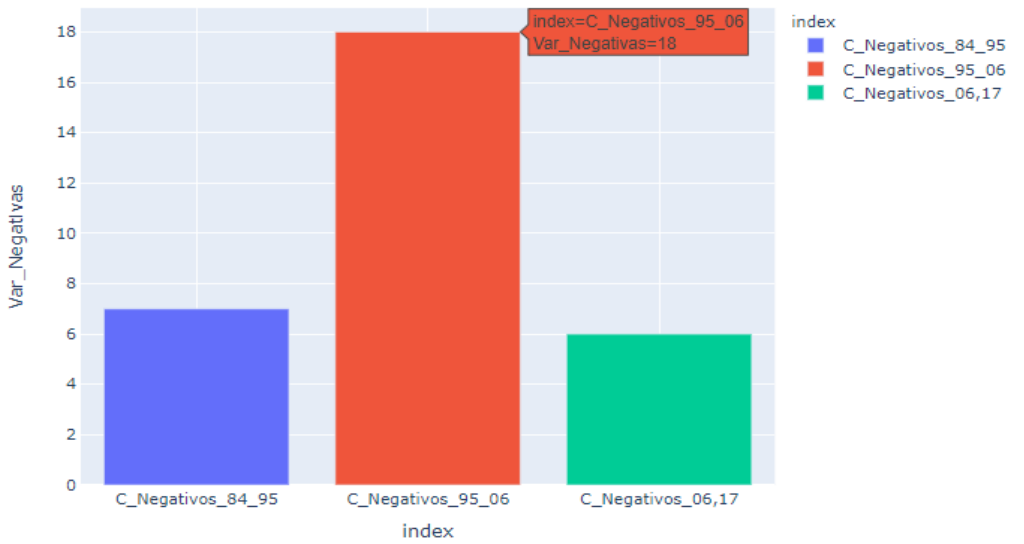
C_95_06 medio por paises



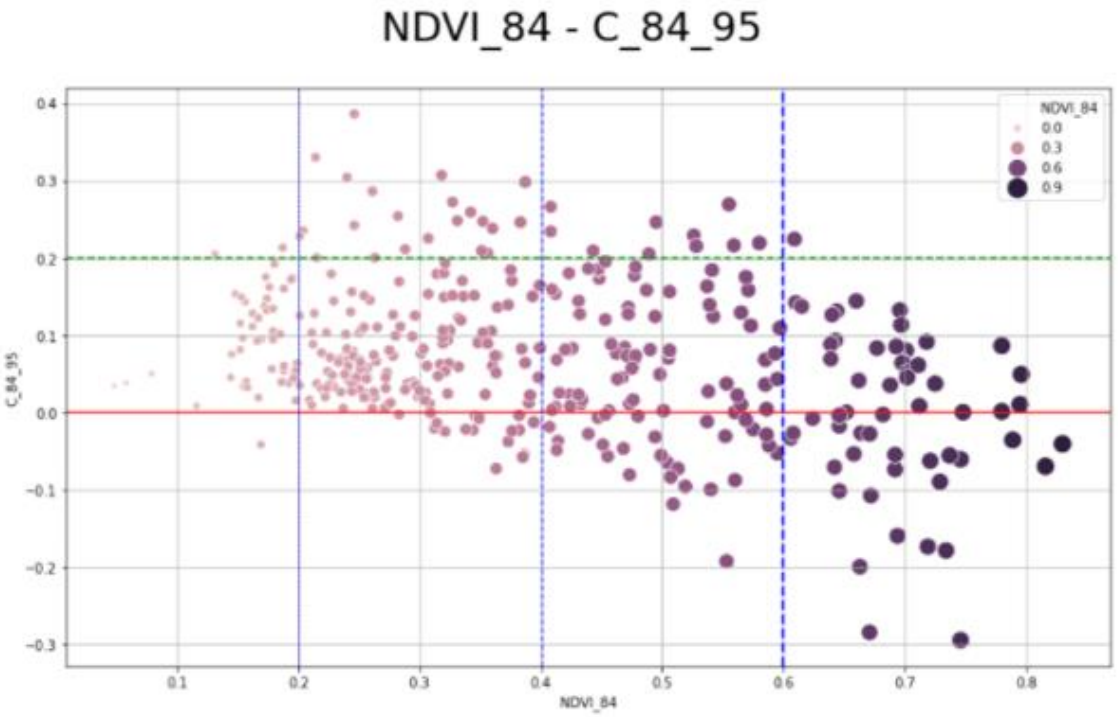
C_06_17 medio por paises



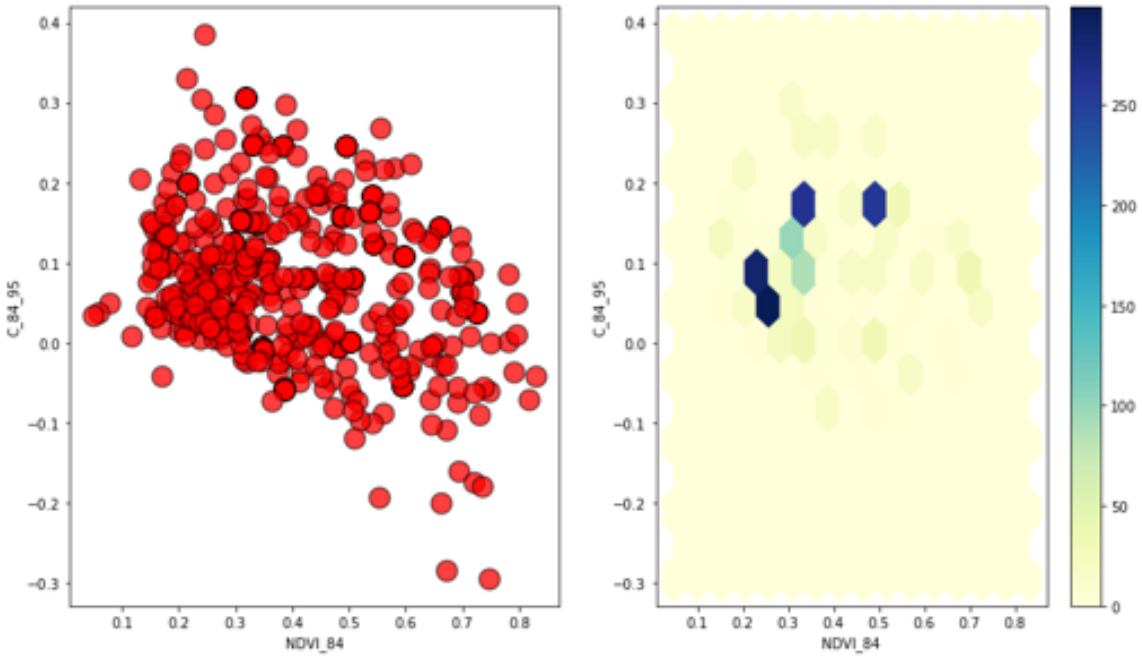
```
# Gráfico de barras
fig = px.bar (df_var_neg, color = df_var_neg.index, x = df_var_neg.index , y = 'Var_Negativas')
fig.show ()
```



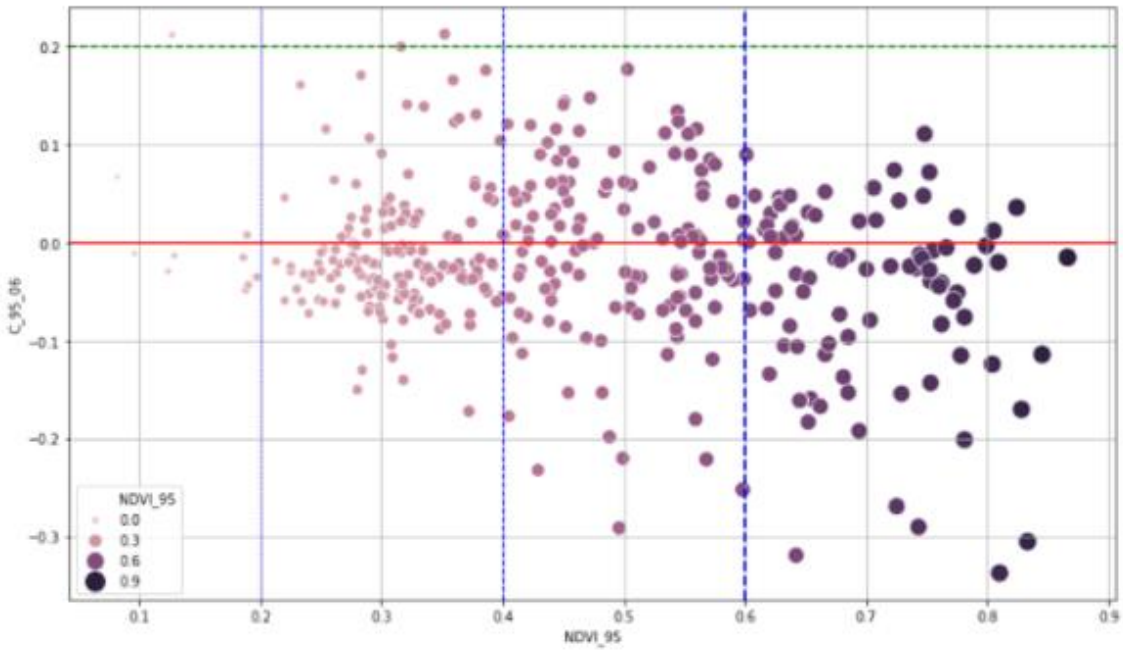
Mediante un gráfico de dispersión de puntos se observa como son las variaciones que sufre el NDVI (en el intervalo de estudio) para los diferentes niveles que presentaba a comienzos del periodo. Teniendo en cuenta que los valores de NDVI entre 0.2 y 0.4 corresponden a áreas con vegetación escasa; la vegetación moderada tiende a variar entre 0.4 y 0.6 y cualquier cosa por encima de 0.6 indica la mayor densidad posible de hojas verdes, en este primer intervalo que va desde 1984 hasta 1995 la mayor concentración de puntos está localizados en el intervalo entre 0.2-0.4.

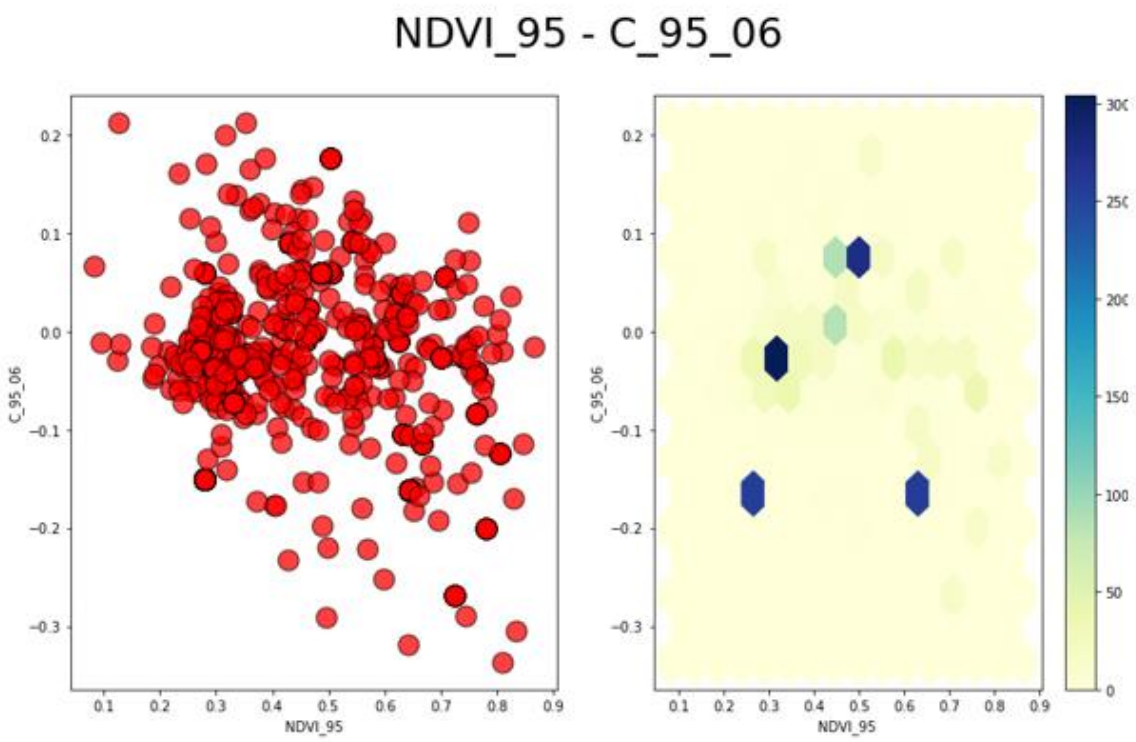


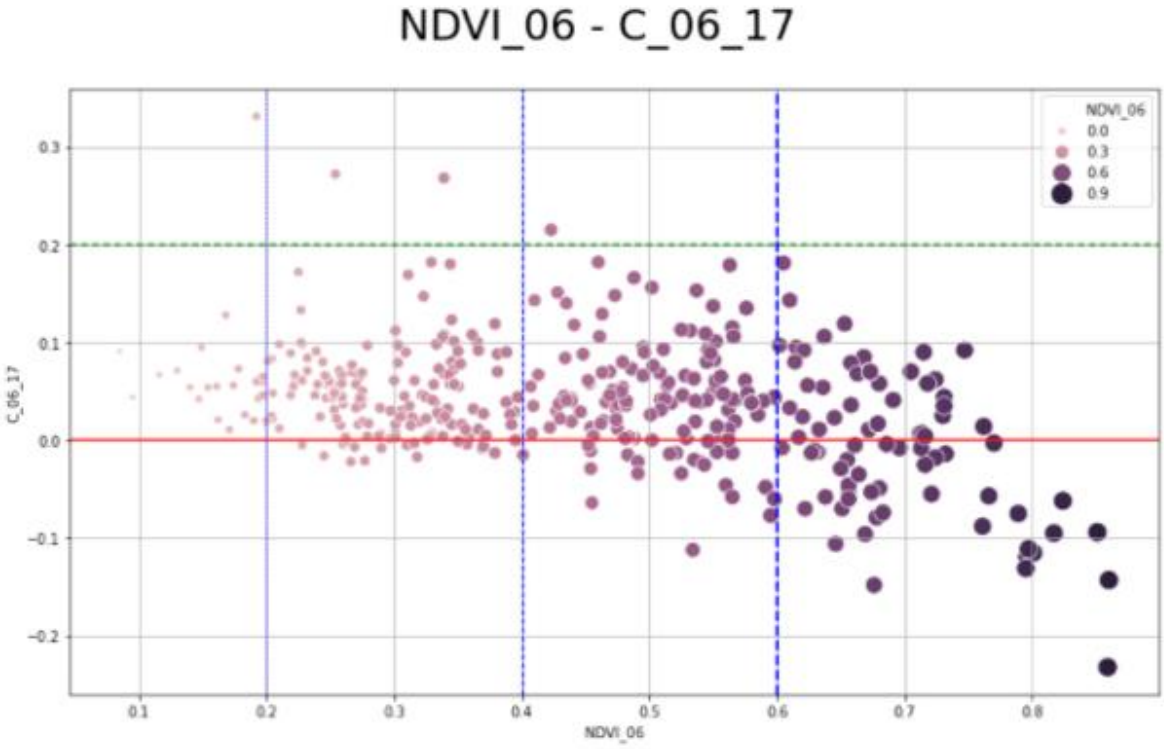
NDVI_84 - C_84_95

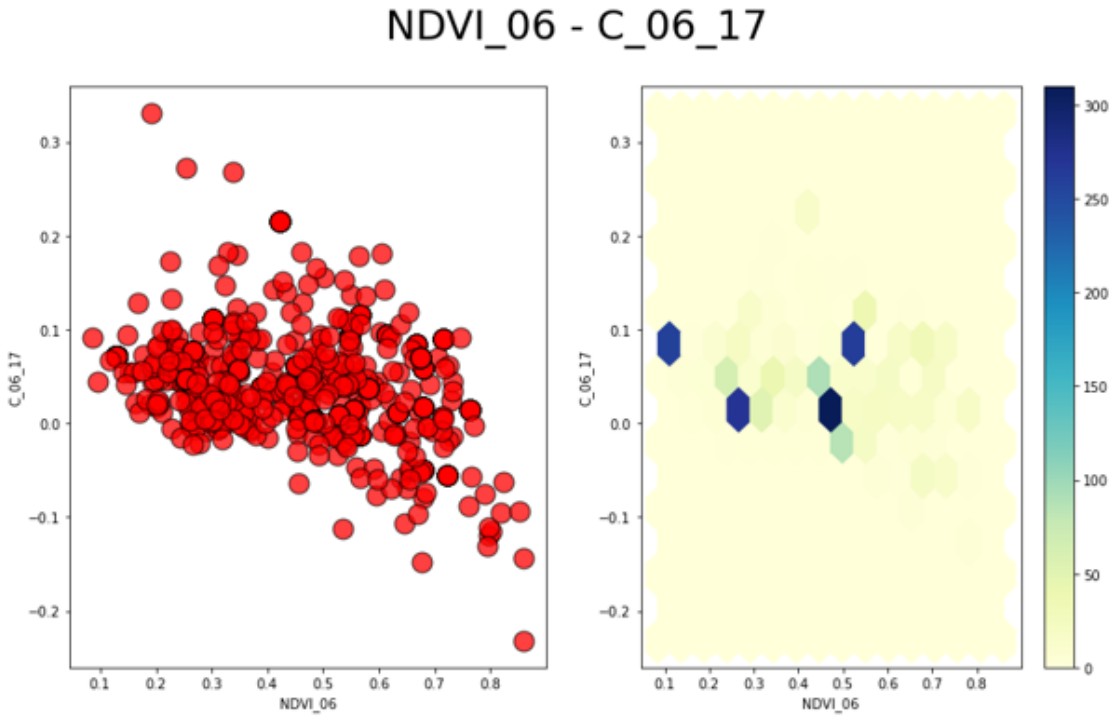


NDVI_95 - C_95_06

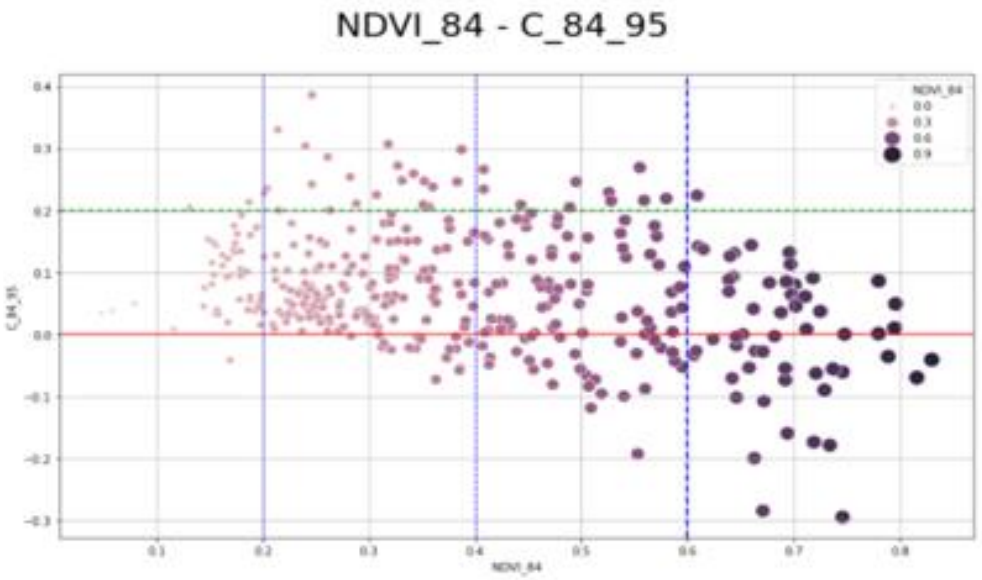


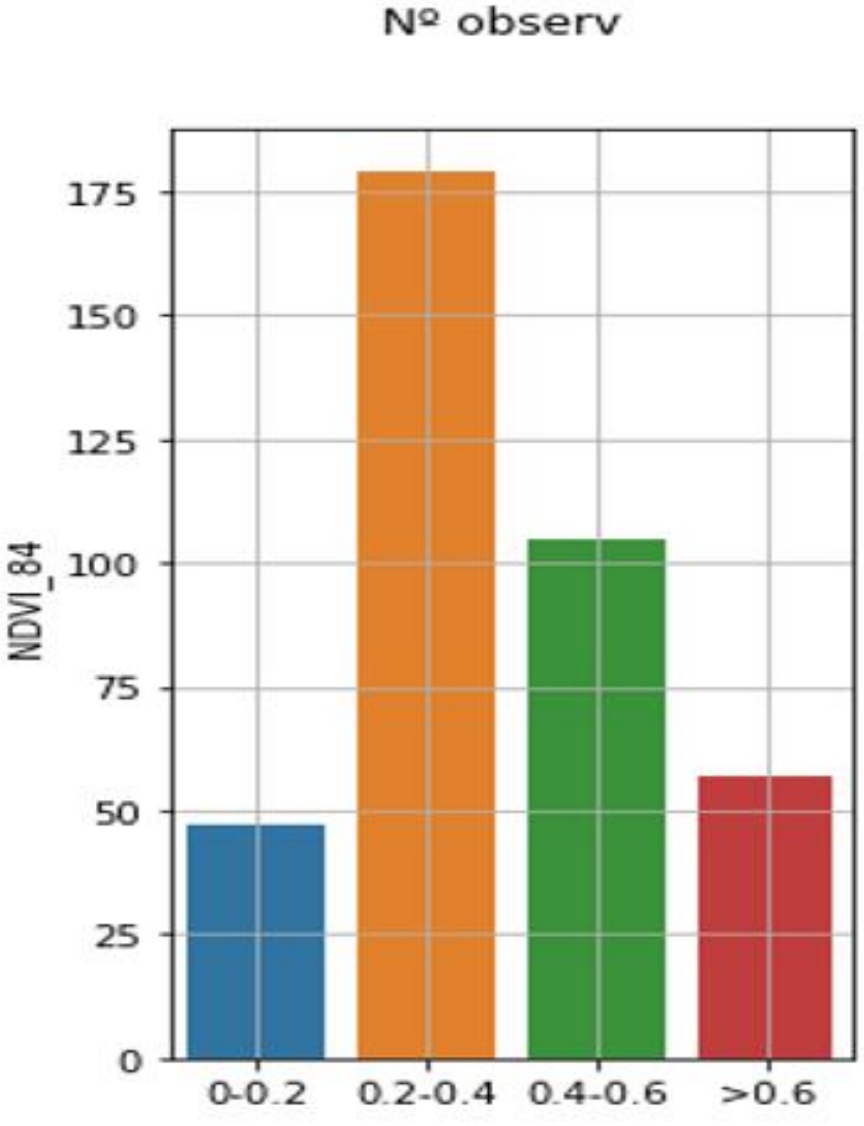

NDVI_95 - C_95_06
C_95_06
NDVI_95


NDVI_06 - C_06_17
NDVI_06
0.0
0.3
0.6
0.9
C_06_17
NDVI_06

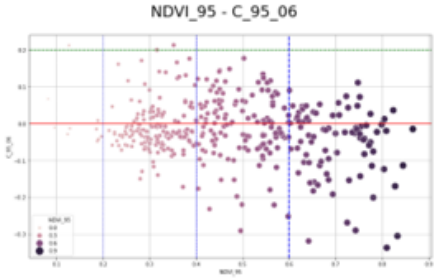


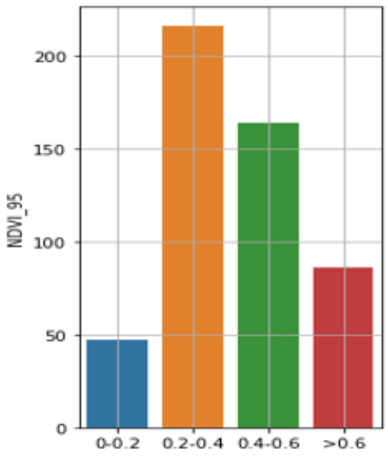
Ahora incorporaremos un diagrama de barras para hacernos una idea de la frecuencia de observaciones que tenemos en cada intervalo.

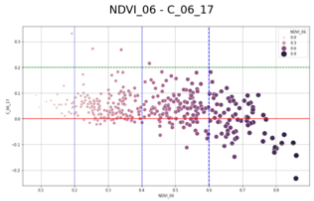

NDVI_84 - C_84_95
NDVI_84
0.0
0.3
0.6
0.9
C_84_95
NDVI_84

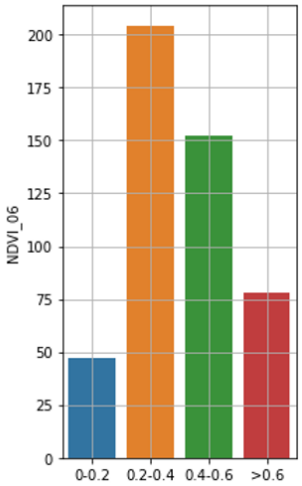

Nº observ
NDVI_84
175
150
125
100
75
50
25
0
0-0.2
0.2-0.4
0.4-0.6
>0.6

De esta composición se deduce que la mayoría de las observaciones están en unos niveles de NDVI que son apropiados para el cultivo. También se observa que las mayores tasas de crecimiento se dan para niveles de NDVI altos y como es lógico estas tasas de crecimiento son negativas. Lo mismo ocurre en el resto de intervalos de estudio.









Mediante un boxplot estudiamos la distribución de los valores del NDVI por países y año.

```
merge_df
```

	Country	Longitud	Latitud	Year	NDVI	NDVI_84	NDVI_95	NDVI_06	NDVI_17	C_84_95	C_95_06	C_06_17
0	Angola	13.75	-7.75	1984	0.453	0.453	0.648	0.598	0.538	0.195	-0.050	-0.06
1	Angola	13.75	-7.75	1985	0.501	0.453	0.648	0.598	0.538	0.195	-0.050	-0.06
2	Angola	13.75	-7.75	1986	0.647	0.453	0.648	0.598	0.538	0.195	-0.050	-0.06
3	Angola	13.75	-7.75	1987	0.467	0.453	0.648	0.598	0.538	0.195	-0.050	-0.06
4	Angola	13.75	-7.75	1988	0.580	0.453	0.648	0.598	0.538	0.195	-0.050	-0.06
...	...	...	...	...	...	...	...	...	...	...	...	...
1317919	Libya	12.75	27.75	2015	0.219	0.197	0.248	0.190	0.210	0.051	-0.058	0.02
1317920	Libya	12.75	27.75	2016	0.207	0.197	0.248	0.190	0.210	0.051	-0.058	0.02
1317921	Libya	12.75	27.75	2017	0.210	0.197	0.248	0.190	0.210	0.051	-0.058	0.02
1317922	Libya	12.75	27.75	2018	0.200	0.197	0.248	0.190	0.210	0.051	-0.058	0.02
1317923	Libya	12.75	27.75	2019	0.201	0.197	0.248	0.190	0.210	0.051	-0.058	0.02

1317924 rows × 12 columns

```
merge_df2 = merge_df[['Country', 'NDVI_84']]
merge_df2
```

```
merge_df2 = merge_df2.drop_duplicates()
merge_df2
```

	Country	NDVI_84
0	Angola	0.453
36	Angola	0.677
72	Angola	0.796
144	Angola	0.795
288	Angola	0.688
...	...	...
1317168	Libya	0.332
1317240	Libya	0.250
1317276	Libya	0.345
1317312	Libya	0.346
1317888	Libya	0.197

288 rows × 2 columns

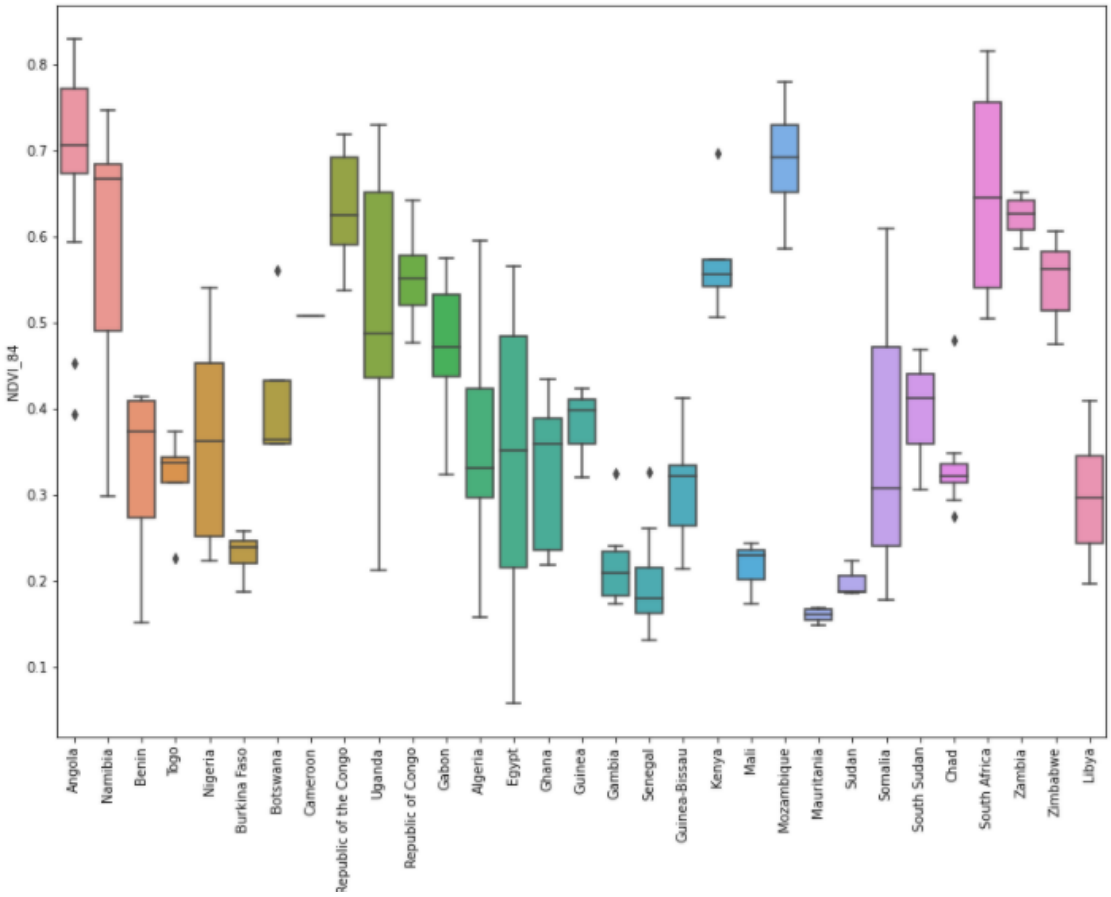
```
merge_df3 = merge_df[['Country', 'NDVI_95']]
merge_df4 = merge_df[['Country', 'NDVI_06']]
merge_df5 = merge_df[['Country', 'NDVI_17']]
```

```
merge_df3 = merge_df3.drop_duplicates()
merge_df4 = merge_df4.drop_duplicates()
merge_df5 = merge_df5.drop_duplicates()
```

```
merge_df5
```

	Country	NDVI_17
0	Angola	0.538
36	Angola	0.691

```
fig, ax = plt.subplots(figsize=(14,10))
fig.suptitle('Distribución NDVI_84 por paises', fontsize=30)
sns.boxplot(merge_df2['Country'], merge_df2['NDVI_84'])
plt.xticks(rotation=90)
```

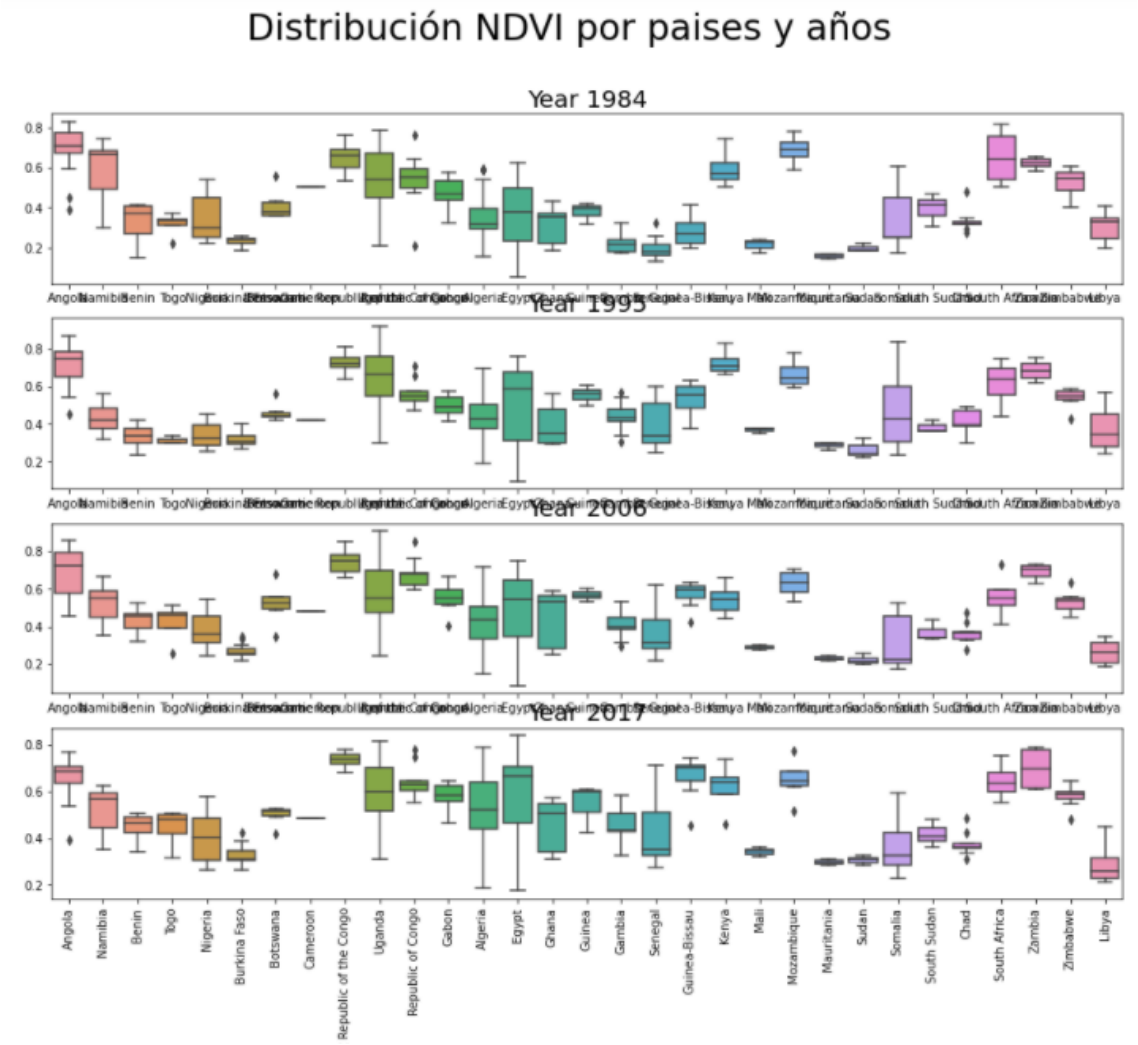



```python
def devolver_pais(yr):
    return merge_df_simply.loc[merge_df_simply['Year']==yr]['Country'].values
```

```python
def devolver_valor_NDVI(yr):
    return merge_df_simply.loc[merge_df_simply['Year']==yr]['NDVI'].values
```

```python
lista_anyos = [1984, 1995, 2006, 2017]
```

```python
fig, ax = plt.subplots(4, 1, figsize=(16,12))
fig.suptitle('Distribución NDVI por paises y años', fontsize=30)

contador = 0
for i in lista_anyos:
    sns.boxplot(devolver_pais(i), devolver_valor_NDVI(i), ax=ax[contador])
    plt.xticks(rotation=90)
    ax[contador].set_title(f'Year {i}', fontsize=20)
    contador += 1
```

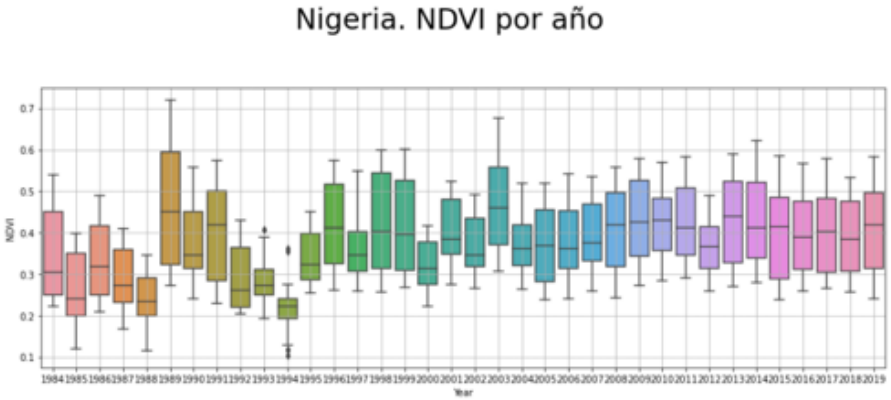



```
#Boxplot de la evolucion de la distribucion del NDVI a lo lardo del tiempo
```

```
def devolver_NDVI(pais):
    return merge_df_simply.loc[merge_df_simply['Country']==pais]['NDVI']
```

```
def devolver_year(pais):
    return merge_df_simply.loc[merge_df_simply['Country']==pais]['Year']
```

```
fig, ax = plt.subplots(figsize=(16,12))
fig.suptitle('Nigeria. NDVI por año', fontsize=30)

sns.boxplot(devolver_year('Nigeria'), devolver_NDVI('Nigeria'))
```



Si queremos acompañar el gráfico de distribución de valores del NDVI con la evolución del NDVI medio por países a lo largo de los años:

```
def devolver_valor_ndvi(pais, yr):
    return merge_df_simply.loc[(merge_df_simply['Country']==pais) & (merge_df_simply['Year']==yr)]
```

```
devolver_valor_ndvi('Nigeria', 1984)
```

```
0.3482972972972972
```

```
co = []
for index, row in merge_df_simply.iterrows():
    c = row['Country']
    y = row['Year']

    val = devolver_valor_ndvi(c, y)
    print(c,y,val)
    co.append([c,y,val])
```

```
Angola 1984 0.6844999999999999
Angola 1985 0.6790714285714285
Angola 1986 0.6699285714285714
Angola 1987 0.6784285714285715
Angola 1988 0.71
Angola 1989 0.7415
Angola 1990 0.5734285714285713
Angola 1991 0.6737142857142857
Angola 1992 0.5694999999999999
Angola 1993 0.6399285714285714
Angola 1994 0.7234999999999999
Angola 1995 0.7149285714285714
Angola 1996 0.7027857142857143
Angola 1997 0.6925714285714287
Angola 1998 0.7746428571428572
Angola 1999 0.6949285714285713
Angola 2000 0.7456428571428573
Angola 2001 0.5592142857142857
Angola 2002 0.6420000000000001
Angola 2003 0.628
```

```
evolucion_df = pd.DataFrame(co, columns=['Country', 'Year', 'NDVI_media'])
```


```
evolucion_df = pd.DataFrame(co, columns=['Country', 'Year', 'NDVI_media'])
```

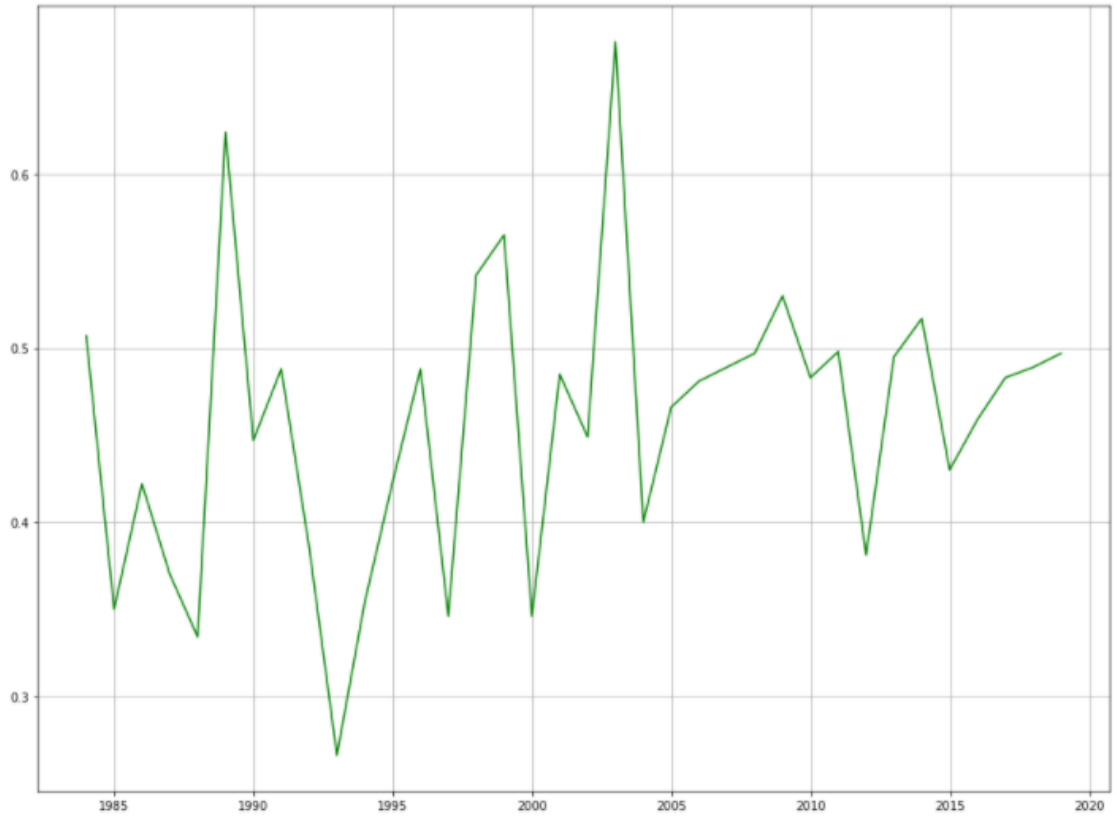

```
evolucion_df
```

	Country	Year	NDVI_media
0	Angola	1984	0.684500
1	Angola	1985	0.679071
2	Angola	1986	0.669929
3	Angola	1987	0.678429

```
filtro_nigeria = evolucion_df.loc[evolucion_df['Country']=='Nigeria']
filtro_nigeria
```

	Country	Year	NDVI_media
864	Nigeria	1984	0.348297
865	Nigeria	1985	0.265730
866	Nigeria	1986	0.335703
867	Nigeria	1987	0.291757
868	Nigeria	1988	0.238000

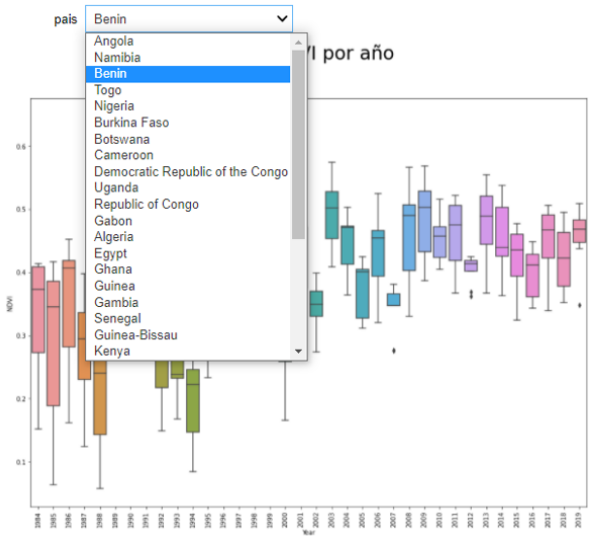
```
fig, ax = plt.subplots(figsize=(16,12))
ax.plot(filtro_nigeria['Year'], filtro_nigeria['NDVI_media'], color='green')
ax.grid(True)
```



De forma interactiva por países:

```
from ipywidgets import interact
```

```
def boxplot_evol_ndvi_paises(pais):
    fig, ax = plt.subplots(figsize=(16,12))
    fig.suptitle(f'{pais}. NDVI por año', fontsize=30)
    plt.xticks(rotation=90)
    return sns.boxplot(devolver_year(pais), devolver_NDVI(pais), ax=ax)

interact(boxplot_evol_ndvi_paises, pais=lista_paises)
```



Generación de Heat_maps.

```
1 import pandas as pd
2 import numpy as np
```

```
1 df_ = pd.read_csv('Panel_DATA.csv').drop('Unnamed: 0', axis=1)
2 df_
```

	Year	NDVI	Longitud	Latitud	NDVI_84	NDVI_95	NDVI_06	NDVI_17
0	1984	0.245	2.75	11.25	0.245	0.324	0.320	0.339
1	1985	0.189	2.75	11.25	0.245	0.324	0.320	0.339
2	1986	0.282	2.75	11.25	0.245	0.324	0.320	0.339
3	1987	0.232	2.75	11.25	0.245	0.324	0.320	0.339
4	1988	0.240	2.75	11.25	0.245	0.324	0.320	0.339
...	...	...	...	...	...	...	...	...
72643	2015	0.565	-15.75	12.75	0.261	0.547	0.548	0.637
72644	2016	0.707	-15.75	12.75	0.261	0.547	0.548	0.637
72645	2017	0.637	-15.75	12.75	0.261	0.547	0.548	0.637
72646	2018	0.577	-15.75	12.75	0.261	0.547	0.548	0.637
72647	2019	0.494	-15.75	12.75	0.261	0.547	0.548	0.637

72648 rows × 8 columns

```
1 df_84 = df_.loc[df_['Year']==1984]
2 df_84
```

	Year	NDVI	Longitud	Latitud	NDVI_84	NDVI_95	NDVI_06	NDVI_17
0	1984	0.245	2.75	11.25	0.245	0.324	0.320	0.339
36	1984	0.250	1.75	10.75	0.250	0.316	0.306	0.336
72	1984	0.412	2.25	6.75	0.412	0.419	0.466	0.505
108	1984	0.234	2.75	9.75	0.234	0.304	0.344	0.401
144	1984	0.152	2.25	8.25	0.152	0.233	0.394	0.423
...	...	...	...	...	...	...	...	...
72468	1984	0.166	-12.75	15.25	0.166	0.258	0.222	0.289
72504	1984	0.152	-14.75	16.25	0.152	0.267	0.273	0.316
72540	1984	0.214	-14.75	12.75	0.214	0.544	0.511	0.604
72576	1984	0.194	-12.75	13.75	0.194	0.366	0.330	0.371
72612	1984	0.261	-15.75	12.75	0.261	0.547	0.548	0.637

2018 rows × 8 columns

```
from folium.plugins import HeatMapWithTime, HeatMap
from folium import Map
```

```
map_ = Map(location=[9,17], tiles='openstreetmap', zoom_start=10)
map_
```

```
data = df_84[['Latitud', 'Longitud', 'NDVI']].values.tolist()
```

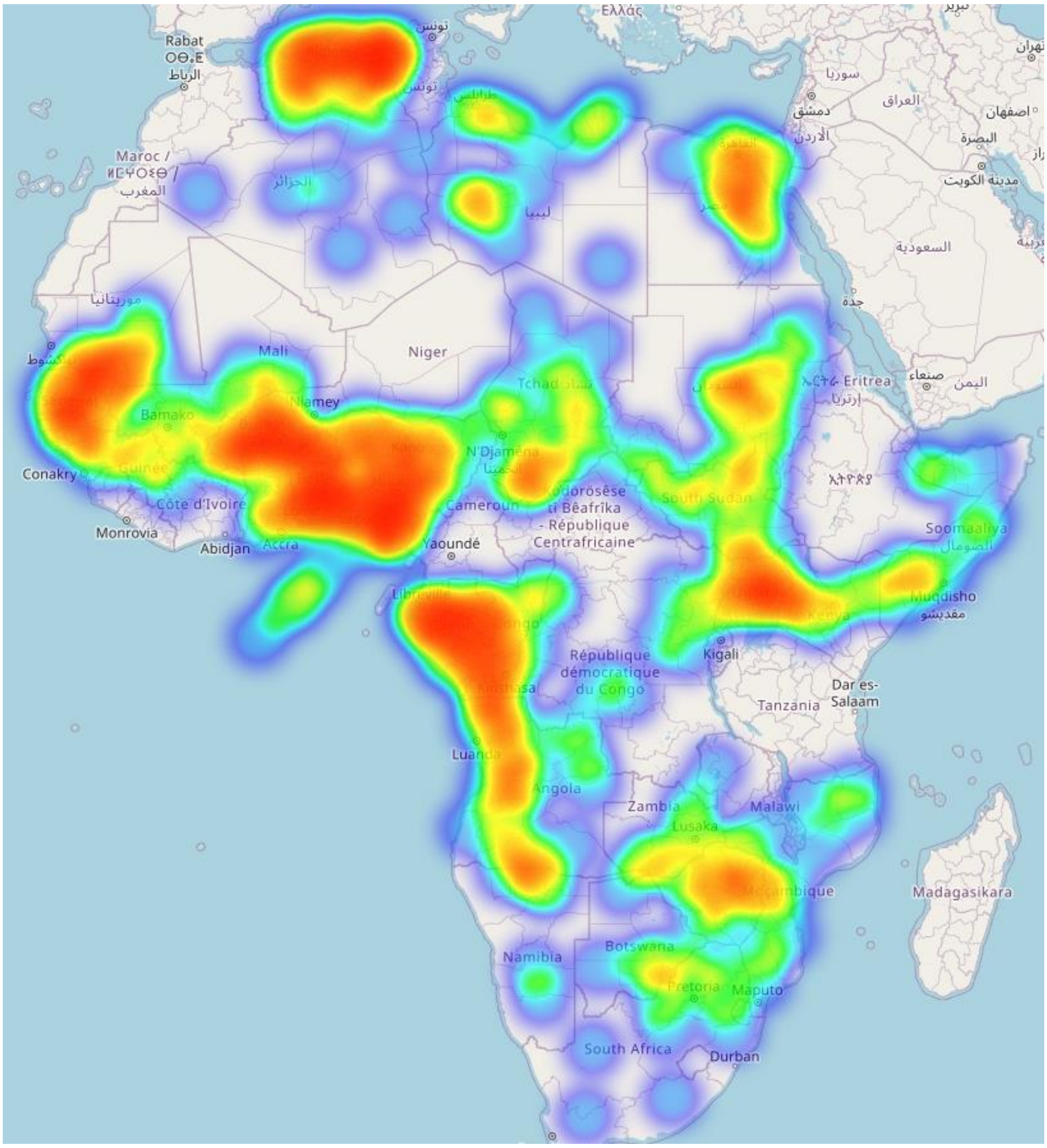
```
heatmap = HeatMap(data)
```

```
heatmap.add_to(map_)
```

```
<folium.plugins.heat_map.HeatMap at 0x1e5b180ac40>
```

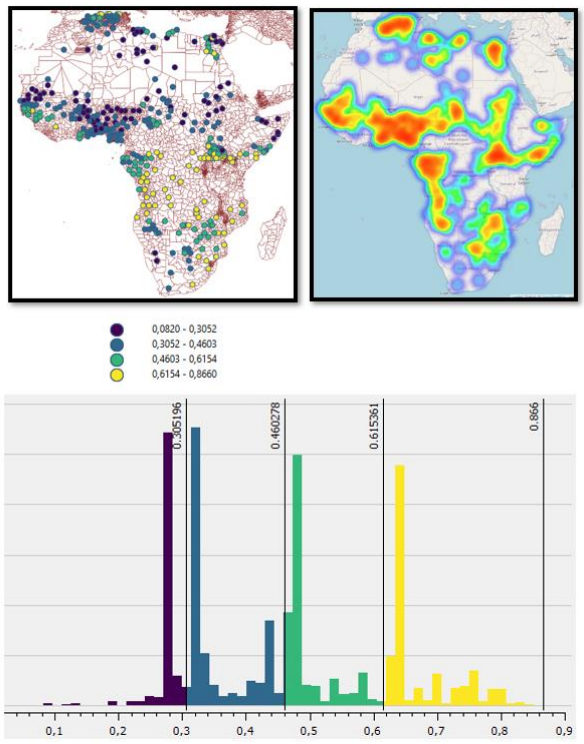
```
map_
```

```
map_.save('C:\\Users\\Usuario\\Desktop\\NDVI_84.html')
```

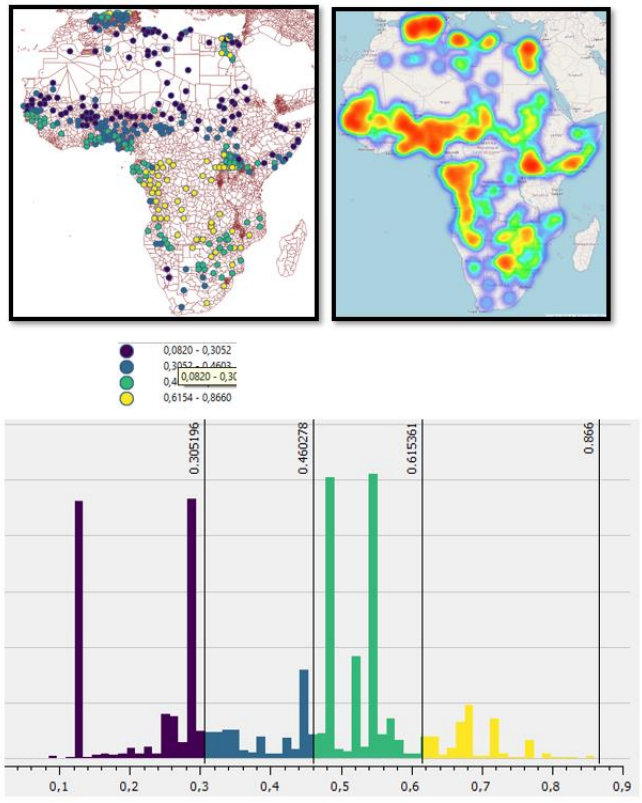


Repetimos el proceso de generación de los mapas de calor para los años 1995, 2006 y 2017 teniendo como resultado

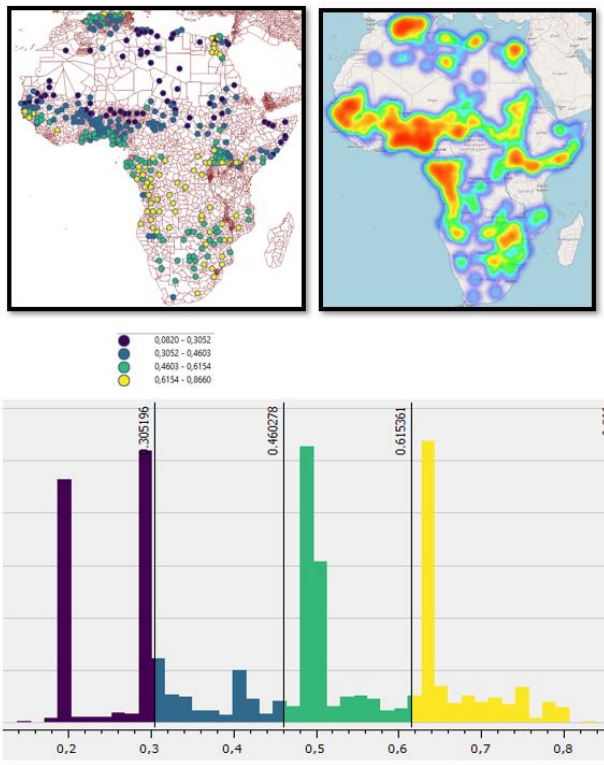
HeatMap NDVI_95



HeatMap NDVI_06



HeatMap NDVI_17



Anexo 5 – Productividad del suelo

En este apéndice desarrollamos paso a paso un modelo de regresión múltiple para intentar explicar el nivel de productividad del suelo agrícola en el continente africano. Para ello haremos uso del dataset 'Cuadro_NDVI_para_REGRESION.csv' que contiene 47 variables predictoras. Como disponemos de un histórico de datos de 33 años consideramos que la variable objetico va a ser el cambio sufrido en el nivel de productividad del suelo medido a través del NDVI.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
df = pd.read_csv('Cuadro_NDVI_para_REGRESION.csv').drop(['Unnamed: 0','NDVI'], axis=1)
df
```

	Latitud	Longitud	Year	Altura	Precip_med	Temp_med	SSP+1_2	SSP+2_3	SSP+3	SSP+1	...	SSTP++ (2_1)	SSTP+- (2_1)	SSTP-+ (2_1)	SSTP- -(2_1)	SSTP++3
0	14.75	-16.75	1984	14.0000	34.6500	25.8833	0.029412	0.000000	0.000000	0.029412	...	0.000000	0.0	0.029412	0.0	0.0
1	14.75	-15.75	1984	29.6667	31.7833	27.5917	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.0	0.000000	0.0	0.0
2	13.75	-15.75	1984	24.0000	52.0000	27.2167	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.0	0.000000	0.0	0.0
3	13.25	-15.75	1984	15.0000	70.7917	26.8167	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.0	0.000000	0.0	0.0
4	12.75	-15.75	1984	18.0000	80.4000	26.6667	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.0	0.000000	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
12123	3.25	46.25	2017	188.0000	33.5333	28.5167	0.029412	0.029412	0.000000	0.058824	...	0.029412	0.0	0.000000	0.0	0.0
12124	5.25	47.25	2017	136.7500	16.6500	29.4417	0.000000	0.000000	0.029412	0.029412	...	0.000000	0.0	0.000000	0.0	0.0
12125	10.25	48.25	2017	1118.5000	12.7667	24.6417	0.000000	0.000000	0.029412	0.029412	...	0.000000	0.0	0.000000	0.0	0.0
12126	6.75	48.25	2017	216.5000	15.8250	28.8917	0.000000	0.000000	0.029412	0.029412	...	0.000000	0.0	0.000000	0.0	0.0
12127	6.25	48.25	2017	162.2500	16.9750	28.8917	0.000000	0.000000	0.029412	0.029412	...	0.000000	0.0	0.000000	0.0	0.0

12128 rows × 48 columns

```
#Establecemos la variable objetivo y las variables predictoras

x = df.drop('C_NDVI', axis=1)
y = df[['C_NDVI']]
```

Nota: A la hora de seleccionar las variables, tanto las predictoras como la variable objetico las construyo como un dataframe de modo que a la hora del preprocesamiento no tengo de redimensionarlas (sobre todo en el caso de la variable y que se quedaría con una única dimensión)

```
#En primer lugar vamos a hacer un preprocesamiento de las variables numéricas

#Estudio de los valores faltantes
```

```
val_faltantes = df.isnull().sum()
val_faltantes
```

```
Latitud       0
Longitud      0
Year          0
Altura        0
Precip_med    0
Temp_med      0
SSP+1_2       0
SSP+2_3       0
SSP+3         0
SSP+1         0
```

También podríamos optar por esta sentencia que además nos facilita el tipo de variable con la que estamos trabajando

```
#2ª Forma de hacerlo
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12128 entries, 0 to 12127
Data columns (total 48 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Latitud      12128 non-null  float64
 1   Longitud     12128 non-null  float64
 2   Year         12128 non-null  int64
 3   Altura       12128 non-null  float64
 4   Precip_med   12128 non-null  float64
 5   Temp_med     12128 non-null  float64

 41  SSTP--(2_1)  12128 non-null  float64
 42  SSTP++3      12128 non-null  float64
 43  SSTP+-3      12128 non-null  float64
 44  SSTP-+3      12128 non-null  float64
 45  SSTP--3      12128 non-null  float64
 46  NDVI_84      12128 non-null  float64
 47  C_NDVI       12128 non-null  float64
dtypes: float64(47), int64(1)
memory usage: 4.4 MB
```

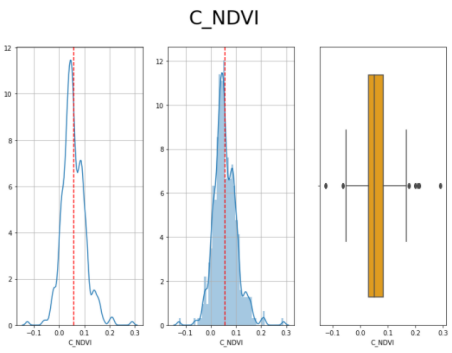
Parece lógico que la tabla esté casi perfecta para trabajar con ella porque es algo que hemos creado previamente nosotros.

Veamos ahora qué forma tiene nuestra variable objetivo.

```
#Estudio de la variable objetivo

desc = df['C_NDVI'].describe()
desc
```

```
count    12128.000000
mean         0.056956
std          0.045458
min         -0.126800
25%          0.029200
50%          0.051500
75%          0.084600
max          0.289700
Name: C_NDVI, dtype: float64
```

```
#Nos apoyaremos en un gráfico para entenderla mejor
fig, ax = plt.subplots(1,3, figsize=(12,8))
fig.suptitle('C_NDVI', fontsize=30)

sns.distplot(df['C_NDVI'], hist=False, ax=ax[0])
ax[0].axvline(desc['mean'], ls='--', color='red')
ax[0].grid(True)

sns.distplot(df['C_NDVI'], hist=True, ax=ax[1])
ax[1].axvline(desc['mean'], ls='--', color='red')
ax[1].grid(True)

sns.boxplot(df['C_NDVI'], color='orange')
```

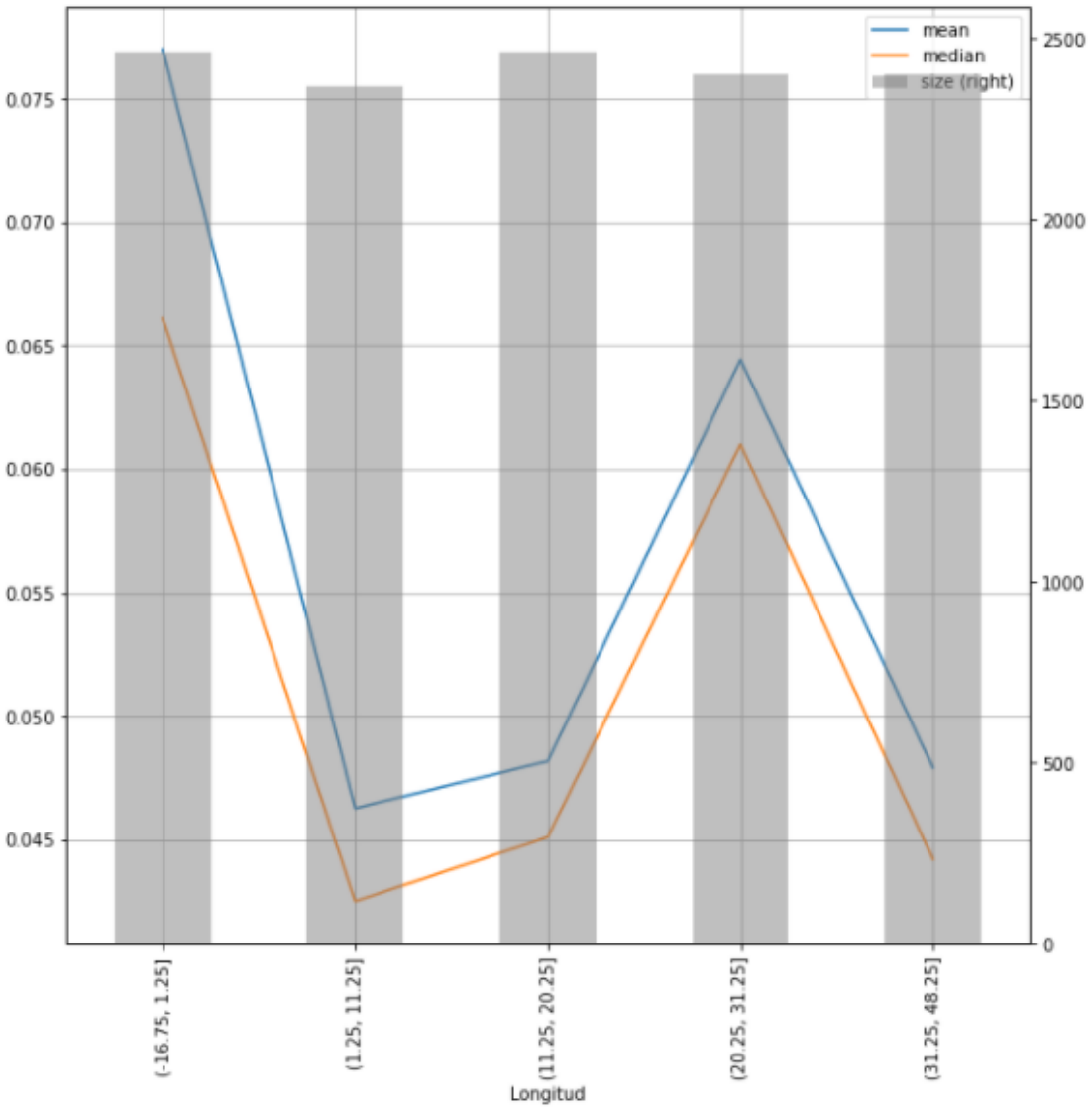



```
#Relación entre las variables numéricas y la variable objetivo
```

```
#En este caso tenemos que todas las variables son numéricas
variables = x.columns.tolist()
variables
```

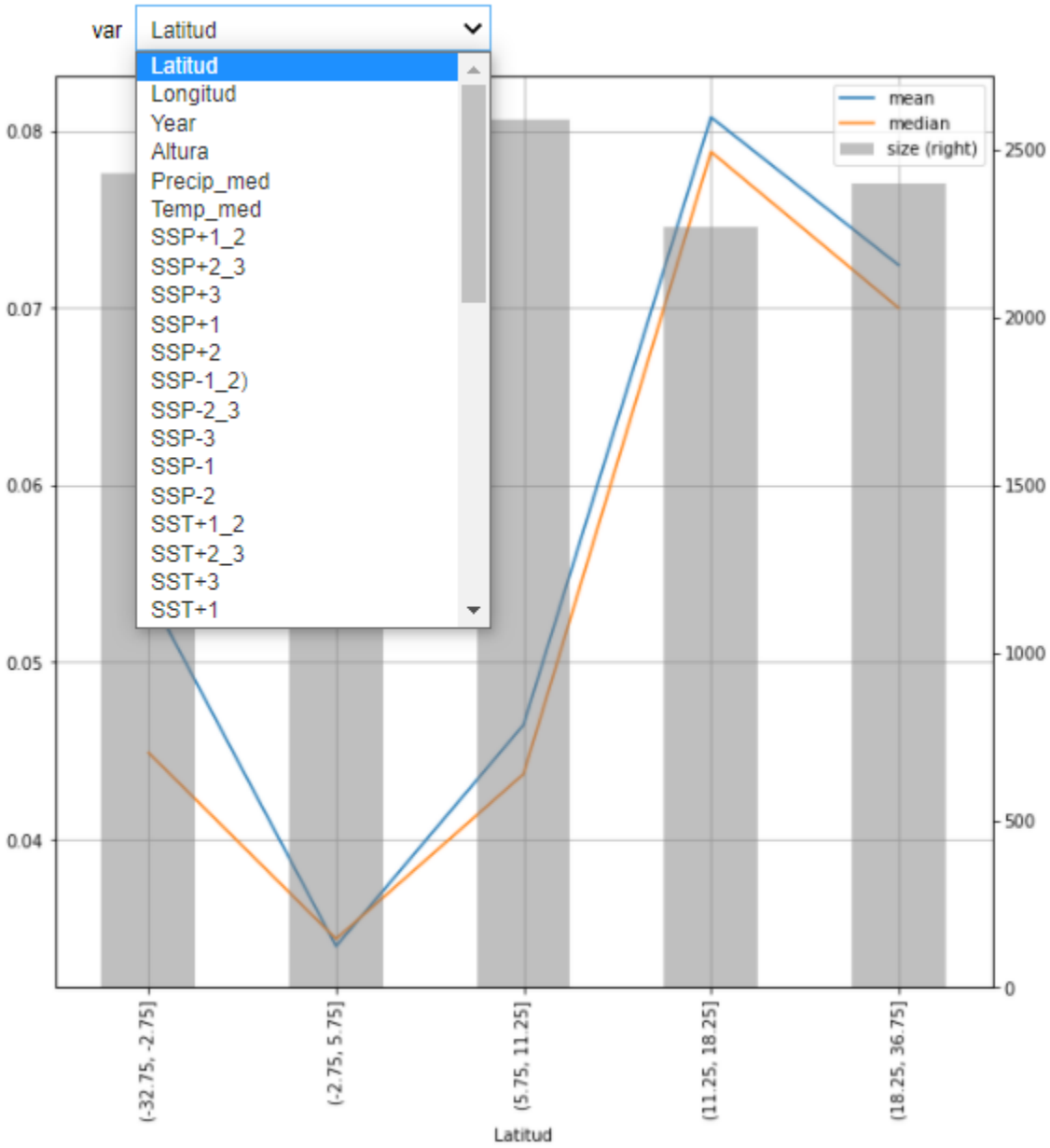
Nos interesa graficar para hacernos una idea de si las variables son o no son significativas. Posteriormente el estudio de la significatividad de las variables la haremos mediante el criterio del p-valor de Pearson, pero esta alternativa gráfica nos ayuda a ver si hay convergencia en los resultados. Si la representación de la media sigue el comportamiento de una recta con pendiente cero, o prácticamente nula a lo largo de su dominio, entonces estaríamos ante una variable poco significativa.

```
#1º lo vemos gráficamente para una única variable

intervalos = np.quantile(df['Longitud'], np.linspace(0,1,6))
grupos = df.groupby([pd.cut(df['Longitud'], duplicates='drop', bins=intervalos)])['C_NDVI'].agg(['mean', 'median', 'size'])

fig, ax = plt.subplots(figsize=(10,10))
grupos[['mean', 'median']].plot(kind='line', ax = ax)
grupos[['size']].plot(kind='bar', color='grey', alpha=0.5, ax=ax, secondary_y=True)
ax.grid(True)
```



Este gráfico se repetiría para cada una de las variables y para hacerlo de una forma más dinámica corregimos el código como sigue:

```
from ipywidgets import interact
def relacion_numericas(var):
    intervalos = np.quantile(df[var], np.linspace(0,1,6))
    grupos = df.groupby([pd.cut(df[var], duplicates='drop', bins=intervalos)])['C_NDVI'].agg(['mea

    fig, ax = plt.subplots(figsize=(10,10))
    return grupos[['mean', 'median']].plot(kind='line', ax = ax), grupos[['size']].plot(kind='bar'
interact(relacion_numericas, var=variables)
```



Utilizando el criterio del p-valor de Pearson

```
import scipy
import statsmodels.formula.api as smf
import statsmodels.api as sm
```

```python
contenedor_significativas = []
contenedor_no_significativas = []
for i in variables:
    coeff, p = scipy.stats.pearsonr(df[i], df['C_NDVI'])
    coeff, p = round(coeff, 3), round(p, 3)
    conclusion = "Significant" if p < 0.05 else "Non-Significant"
    if p < 0.5:
        contenedor_significativas.append(i)
    else:
        contenedor_no_significativas.append(i)

    print(f"La variable {i} tiene un p-valor = {p}, por lo tanto es {conclusion}")
```

```
La variable Latitud tiene un p-valor = 0.0, por lo tanto es Significant
La variable Longitud tiene un p-valor = 0.0, por lo tanto es Significant
La variable Year tiene un p-valor = 1.0, por lo tanto es Non-Significant
La variable Altura tiene un p-valor = 0.0, por lo tanto es Significant
La variable Precip_med tiene un p-valor = 0.0, por lo tanto es Significant
La variable Temp_med tiene un p-valor = 0.141, por lo tanto es Non-Significant
La variable SSP+1_2 tiene un p-valor = 0.0, por lo tanto es Significant
La variable SSP+2_3 tiene un p-valor = 0.691, por lo tanto es Non-Significant
La variable SSP+3 tiene un p-valor = 0.0, por lo tanto es Significant
La variable SSP+1 tiene un p-valor = 0.01, por lo tanto es Significant
La variable SSP+2 tiene un p-valor = 0.0, por lo tanto es Significant
La variable SSP-1_2) tiene un p-valor = 0.0, por lo tanto es Significant
La variable SSP-2_3 tiene un p-valor = 0.0, por lo tanto es Significant
La variable SSP-3 tiene un p-valor = 0.0, por lo tanto es Significant
La variable SSP-1 tiene un p-valor = 0.0, por lo tanto es Significant
La variable SSP-2 tiene un p-valor = 0.0, por lo tanto es Significant
La variable SST+1_2 tiene un p-valor = 0.268, por lo tanto es Non-Significant
La variable SST+2_3 tiene un p-valor = 0.853, por lo tanto es Non-Significant
La variable SST+3 tiene un p-valor = 0.0, por lo tanto es Significant
La variable SST+1 tiene un p-valor = 0.631, por lo tanto es Non-Significant
La variable SST+2 tiene un p-valor = 0.251, por lo tanto es Non-Significant
La variable SST-1_2 tiene un p-valor = 0.103, por lo tanto es Non-Significant
La variable SST-2_3 tiene un p-valor = 0.297, por lo tanto es Non-Significant
La variable SST-3 tiene un p-valor = 0.682, por lo tanto es Non-Significant
La variable SST-1 tiene un p-valor = 0.245, por lo tanto es Non-Significant
La variable SST-2 tiene un p-valor = 0.374, por lo tanto es Non-Significant
La variable SSTP++1 tiene un p-valor = 0.01, por lo tanto es Significant
La variable SSTP+-1 tiene un p-valor = 0.0, por lo tanto es Significant
La variable SSTP-+1 tiene un p-valor = 0.096, por lo tanto es Non-Significant
La variable SSTP--1 tiene un p-valor = 0.0, por lo tanto es Significant
La variable SSTP++(2) tiene un p-valor = 0.015, por lo tanto es Significant
La variable SSTP+-(2) tiene un p-valor = 0.0, por lo tanto es Significant
La variable SSTP-+(2) tiene un p-valor = 0.435, por lo tanto es Non-Significant
La variable SSTP--(2) tiene un p-valor = 0.713, por lo tanto es Non-Significant
La variable SSTP++(1_2) tiene un p-valor = nan, por lo tanto es Non-Significant
La variable SSTP+-(1_2) tiene un p-valor = 0.0, por lo tanto es Significant
La variable SSTP-+(1_2) tiene un p-valor = 0.374, por lo tanto es Non-Significant
La variable SSTP--(1_2) tiene un p-valor = 0.713, por lo tanto es Non-Significant
La variable SSTP++(2_1) tiene un p-valor = 0.0, por lo tanto es Significant
La variable SSTP+-(2_1) tiene un p-valor = 0.001, por lo tanto es Significant
La variable SSTP-+(2_1) tiene un p-valor = 0.238, por lo tanto es Non-Significant
La variable SSTP--(2_1) tiene un p-valor = 0.03, por lo tanto es Significant
La variable SSTP++3 tiene un p-valor = 0.486, por lo tanto es Non-Significant
La variable SSTP+-3 tiene un p-valor = nan, por lo tanto es Non-Significant
La variable SSTP-+3 tiene un p-valor = 0.347, por lo tanto es Non-Significant
La variable SSTP--3 tiene un p-valor = nan, por lo tanto es Non-Significant
La variable NDVI_84 tiene un p-valor = 0.0, por lo tanto es Significant
```

```
1 contenedor_no_significativas
```

```
['Year',
 'SSP+2_3',
 'SST+2_3',
 'SST+1',
 'SST-3',
 'SSTP--(2)',
 'SSTP++(1_2)',
 'SSTP--(1_2)',
 'SSTP+-3',
 'SSTP--3']
```

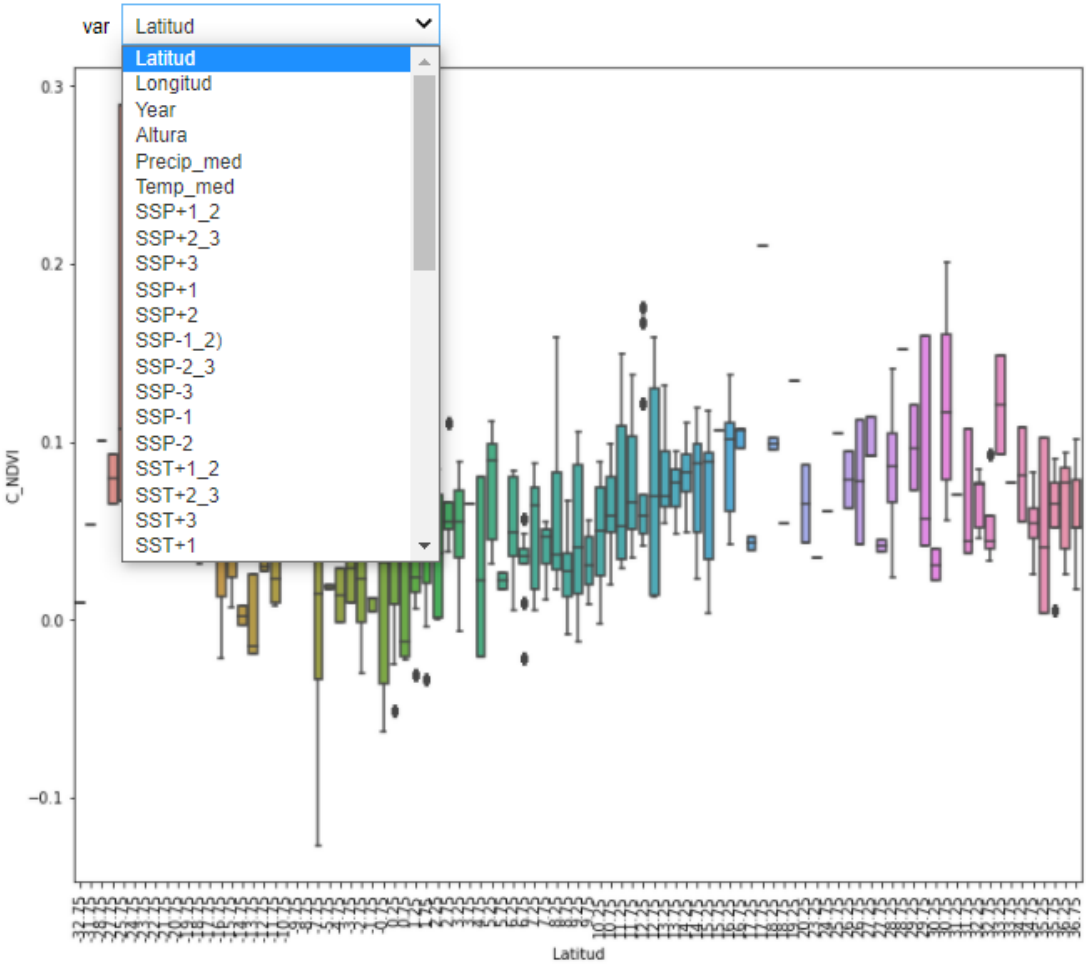
```
1 print(f"Existen {len(contenedor_no_significativas)} variables no significativas y {len(contenedor_
```

```
Existen 10 variables no significativas y 37 variables significativas
```

La segunda parte del análisis exploratorio de datos y preprocesamiento es el estudio de los outliers. Igualmente nos vamos a apoyar en una herramienta gráfica a través de los boxplot y posteriormente utilizaremos el criterio matemático.

Por medio del criterio gráfico consideraremos un 'outliers' los puntos dibujados fuera del diagrama de cajas y bigotes.

```
1 #Estudio de los OUTLIERS
2 def outliers_variables(var):
3     fig, ax= plt.subplots(figsize=(12,10))
4     plt.xticks(rotation=90)
5     return sns.boxplot(df[var], df['C_NDVI'], ax=ax)
6 interact(outliers_variables, var=variables)
```



En este modelo como la presencia de outliers no es tan significativa los voy a corregir mediante el escalado de las variables. En este caso es necesario hacer un escalado de variables porque independientemente de que sean muchas, se mueven en rangos diferentes y lo que es

más importante, están referidas a unidades distintas. Podríamos haber usado un StandardScaler, pero como hay presencia de outliers es más conveniente hacer el escalado a través del RobustScaler.

```
#Preprocesamiento
```

```
#En presencia de outliers es importante hacer un escalado de variables

from sklearn.preprocessing import RobustScaler
#Empezamos escalando las variables predictoras
x_escalado = RobustScaler(quantile_range=(0.25,0.75))
x_rs = x_escalado.fit_transform(x)
```

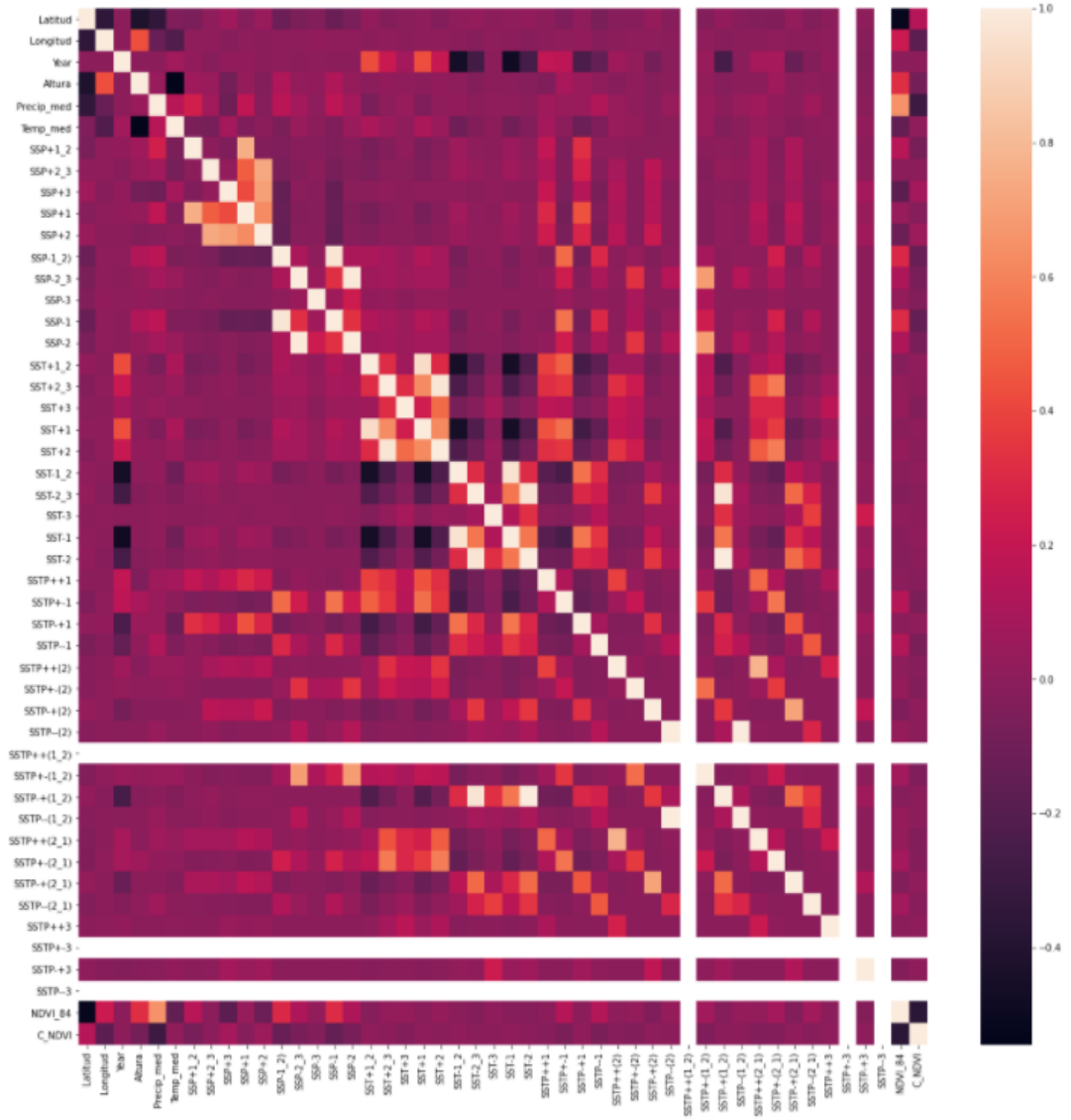
```
#Hacemos lo mismo con la variable y
y_escalado = RobustScaler(quantile_range=(0.25,0.75))
y_rs = y_escalado.fit_transform(y)
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_rs, y_rs, test_size=0.2, random_state=2019)
```

```
#Selección de características
```

```
#En primer lugar vamos a hacer un heat_map
correlacions = df.corr(method='pearson')
fig, ax = plt.subplots(figsize=(20,20))
sns.heatmap(correlacions, annot=False, ax=ax)
```

Para ello vamos a hacernos una idea inicial de cómo es la correlación entre ellas.



```
#A continuación vamos a crear una DF con los cvalores escalados

X = pd.DataFrame(x_rs, columns=x.columns)
X
```

	Latitud	Longitud	Year	Altura	Precip_med	Temp_med	SSP+1_2	SSP+2_3	SSP+3	SSP+1	...	SSTP-(1_2)	SSTP(2
0	1.375	-32.0	-17.5	-4.741553	-255.6000	1.544470	0.000000	0.000000	0.000000	-0.029412	...	0.0	0.000
1	1.375	-31.0	-17.5	-4.554673	-278.5336	4.930946	-0.029412	0.000000	0.000000	-0.058824	...	0.0	0.000
2	1.125	-31.0	-17.5	-4.622268	-116.8000	4.187602	-0.029412	0.000000	0.000000	-0.058824	...	0.0	0.000
3	1.000	-31.0	-17.5	-4.729624	33.5336	3.394702	-0.029412	0.000000	0.000000	-0.058824	...	0.0	0.000
4	0.875	-31.0	-17.5	-4.693839	110.4000	3.097365	-0.029412	0.000000	0.000000	-0.058824	...	0.0	0.000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
12123	-1.500	31.0	15.5	-2.666005	-264.5336	6.764527	0.000000	0.029412	0.000000	0.000000	...	0.0	0.029
12124	-1.000	32.0	15.5	-3.277337	-399.6000	8.598108	-0.029412	0.000000	0.029412	-0.029412	...	0.0	0.000
12125	0.250	33.0	15.5	8.433403	-430.6664	-0.916691	-0.029412	0.000000	0.029412	-0.029412	...	0.0	0.000
12126	-0.625	33.0	15.5	-2.326045	-406.2000	7.507871	-0.029412	0.000000	0.029412	-0.029412	...	0.0	0.000
12127	-0.750	33.0	15.5	-2.973162	-397.0000	7.507871	-0.029412	0.000000	0.029412	-0.029412	...	0.0	0.000

12128 rows × 47 columns

```
Y = pd.DataFrame(y, columns=['C_NDVI'])
Y.head()
```

	C_NDVI
0	0.0990
1	0.1187
2	0.0849
3	0.1157
4	0.1587

A continuación, voy a hacer una selección de variables bastante completa que consiste en aplicar dos métodos simultáneamente, por un lado, utilizaremos el criterio del p-valor que hicimos anteriormente y por otro lado utilizaremos la regularización de (Ridge) con una intensidad de regularización intermedia. Finalmente nos quedaremos con aquellas que hayan sido seleccionadas simultáneamente por los dos criterios. Vamos a verlo paso a paso.

```
on de variables utilizaremos dos métodos, por un lado el p-valor y por otro lado la regularización
ort model_selection, preprocessing, feature_selection, ensemble, linear_model, metrics, decompositi
X.columns



es_p = feature_selection.SelectKBest (score_func = feature_selection.f_regression, k = 47).fit(X,Y)
features = feature_names[selector_variables_p.get_support ()]
```

```
pvalue_selected_features
```

```
Index(['Latitud', 'Longitud', 'Year', 'Altura', 'Precip_med', 'Temp_med',
       'SSP+1_2', 'SSP+2_3', 'SSP+3', 'SSP+1', 'SSP+2', 'SSP-1_2)', 'SSP-2_3',
       'SSP-3', 'SSP-1', 'SSP-2', 'SST+1_2', 'SST+2_3', 'SST+3', 'SST+1',
       'SST+2', 'SST-1_2', 'SST-2_3', 'SST-3', 'SST-1', 'SST-2', 'SSTP++1',
       'SSTP+-1', 'SSTP-+1', 'SSTP--1', 'SSTP++(2)', 'SSTP+-(2)', 'SSTP-+(2)',
       'SSTP--(2)', 'SSTP++(1_2)', 'SSTP+-(1_2)', 'SSTP-+(1_2)', 'SSTP--(1_2)',
       'SSTP++(2_1)', 'SSTP+-(2_1)', 'SSTP-+(2_1)', 'SSTP--(2_1)', 'SSTP++3',
       'SSTP+-3', 'SSTP-+3', 'SSTP--3', 'NDVI_84'],
      dtype='object')
```

```
## Ridge


selector_ridge = feature_selection.SelectFromModel (estimator = linear_model.Ridge (alpha = 10, fi
                                                    max_features = 47) .fit (X, Y)
regularization_selected_features = feature_names [selector_ridge.get_support ()]
```

```
regularization_selected_features
```

```
Index(['SSP+3', 'SSP-2_3', 'SSP-3', 'SSP-1', 'SSP-2', 'SST+2_3', 'SST+3',
       'SST-2_3', 'SST-3', 'SSTP++1', 'SSTP-+1', 'SSTP++(2)', 'SSTP+-(2)',
       'SSTP-+(2)', 'SSTP++(2_1)', 'SSTP--(2_1)'],
      dtype='object')
```

```
#ploteo

#Emopezamos creando un DF con una columna donde sólo aparezcan las variables predictoras
df_features = pd.DataFrame(feature_names, columns=['Feature_Names'])
```

```
#Creamos una segunda columna que llamaremos 'p-valor'
#en ella escribiremos 'p-valor' si la variable se encuentra en la lista 'pvalue_selected_features'

df_features['p-valor'] = df_features['Feature_Names'].apply(lambda x: 'p-valor' if x in pvalue_sel
df_features
```

	Feature_Names	p-valor
0	Latitud	p-valor
1	Longitud	p-valor
2	Year	p-valor
3	Altura	p-valor
4	Precip_med	p-valor
5	Temp_med	p-valor
6	SSP+1_2	p-valor
7	SSP+2_3	p-valor
8	SSP+3	p-valor
9	SSP+1	p-valor
10	SSP+2	p-valor
11	SSP-1_2)	p-valor
12	SSP-2_3	p-valor
13	SSP-3	p-valor

```
#Si en la columna p-valor aparece 'p-valor' queremos que en la nueva columna 'num_1' aparezca un 1

df_features['num_1'] = df_features['p-valor'].apply(lambda x: 1 if x=='p-valor' else 0 )
df_features
```

	Feature_Names	p-valor	num_1
0	Latitud	p-valor	1
1	Longitud	p-valor	1
2	Year	p-valor	1
3	Altura	p-valor	1
4	Precip_med	p-valor	1

```
#Hacemos lo mismo con las características seleccionadas por Ridge
df_features['Ridge'] = df_features['Feature_Names'].apply(lambda x: 'Ridge' if x in regularization
df_features['num_2'] = df_features['Ridge'].apply(lambda x: 1 if x=='Ridge' else 0)
df_features
```

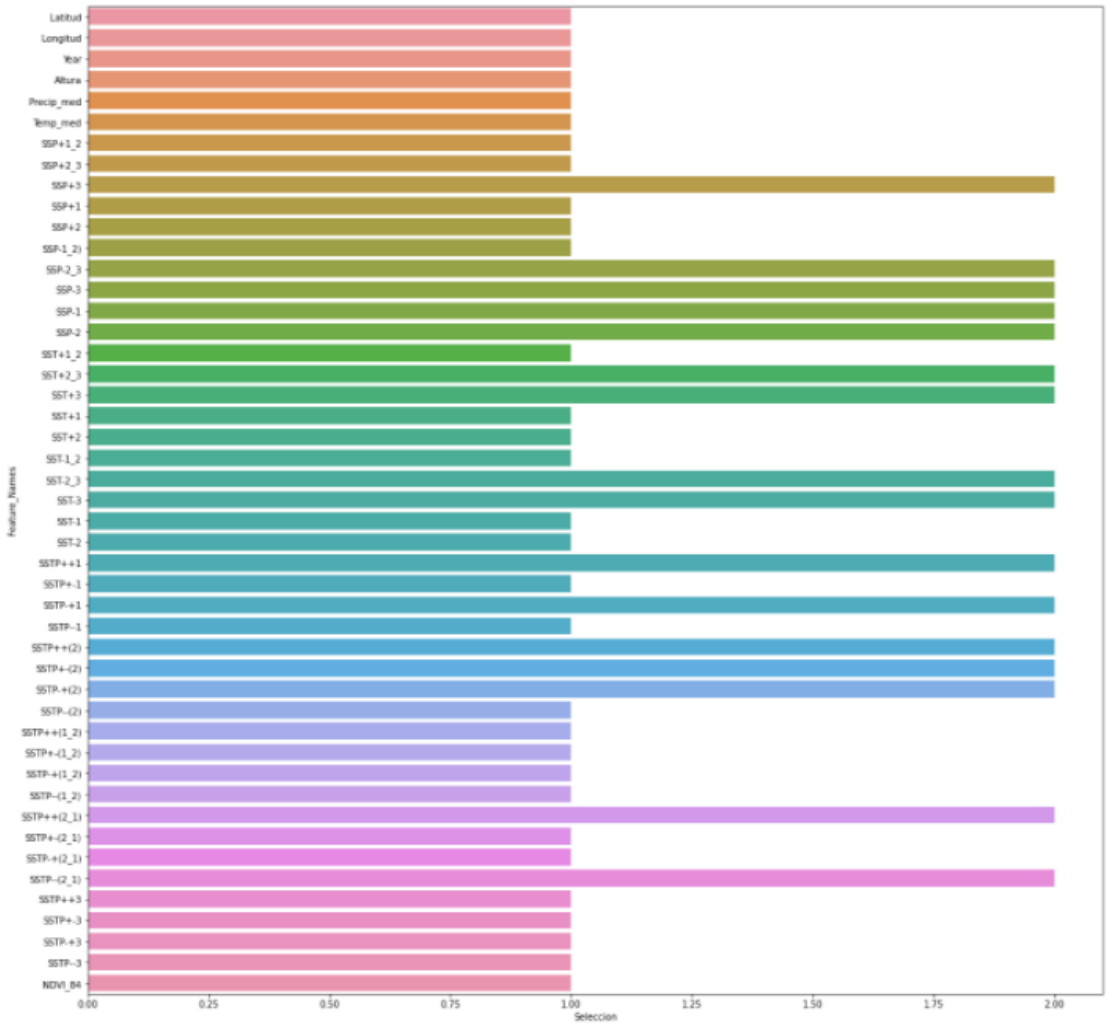
```
#Para saber qué método estamos seleccionando
df_features['Metodo'] = df_features[['p-valor', 'Ridge']].apply(lambda x: (x[0]+" "+x[1]).strip(),
df_features
```

```
df_features['Seleccion'] = df_features['num_1'] + df_features['num_2']
df_features
```

```
df_features["Metodo_2"] = df_features["Metodo"].apply(lambda x: "ambas" if len(x.split()) == 2 els
df_features
```

```
#ploteo
fig, ax = plt.subplots(figsize=(20,20))
sns.barplot(x=df_features['Seleccion'], y=df_features['Feature_Names'], ax = ax, data=df_features.
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2c304ca08e0>
```



Según este criterio doble de selección nos quedaremos con aquellas variables de la figura cuya barra horizontal alcance el valor de dos.

Para hacer más completo nuestro estudio vamos a utilizar otro criterio para la selección de variables, este criterio es muy utilizado en redes neuronales y recibe el nombre de gradiente descendente.

```
#Método de selección de variables basado en el gradiente descendente
```

```
modelo_seleccion = ensemble.GradientBoostingRegressor()
```

```
modelo_seleccion.fit(X,Y)
```

```
variables_importance = modelo_seleccion.feature_importances_
variables_importance
```

```
array([ 3.28735950e-01,  2.32465579e-01,  1.69740867e-18,  1.68597136e-01,
        1.66576990e-02,  2.33857200e-02,  3.94683853e-08, -2.05996198e-19,
        1.81026223e-08,  1.81398868e-07,  1.79614744e-06,  2.62024929e-05,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  3.40632775e-08,  0.00000000e+00,  0.00000000e+00,
       -1.23597719e-20,  0.00000000e+00,  6.59187832e-20,  0.00000000e+00,
        9.55822357e-19,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  2.30129644e-01])
```

```
importancia_df = pd.DataFrame({'Variables':feature_names,
                               'Importancia': variables_importance})
```

```
importancia_df = importancia_df.sort_values(by='Importancia',ascending=False)
```

```
importancia_df['Acumulado'] = importancia_df['Importancia'].cumsum()
importancia_df
```

	Variables	Importancia	Acumulado
0	Latitud	3.260510e-01	0.326051
46	NDVI_84	2.357048e-01	0.561756
1	Longitud	2.320908e-01	0.793847
3	Altura	1.656134e-01	0.959460
5	Temp_med	2.368988e-02	0.983150
4	Precip_med	1.665780e-02	0.999808
9	SSP+1	1.905817e-04	0.999998
10	SSP+2	1.594920e-06	1.000000
8	SSP+3	3.538384e-08	1.000000
21	SST-1_2	2.234892e-08	1.000000
2	Year	5.316633e-18	1.000000
6	SSP+1_2	1.287476e-21	1.000000
36	SSTP-+(1_2)	0.000000e+00	1.000000

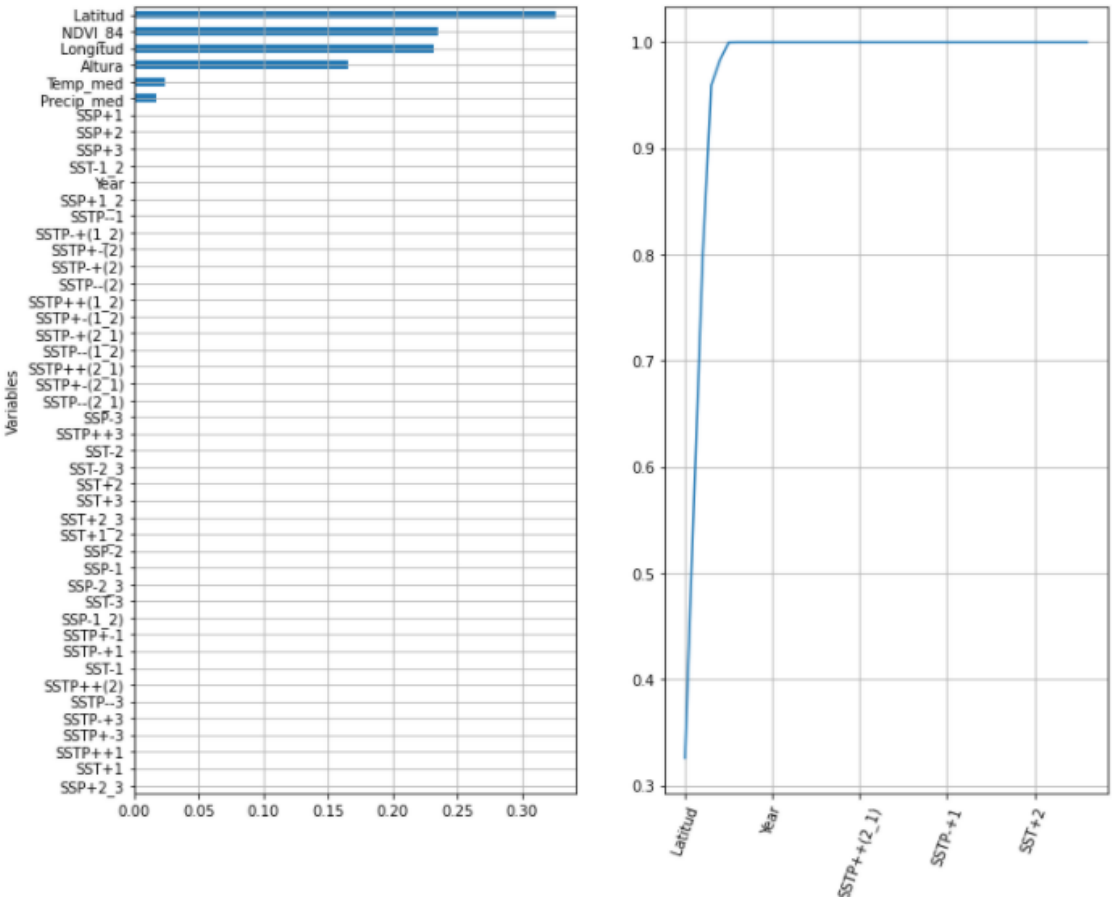
Esta tabla continúa, pero se observa que a partir de la variable SSP+1 la importancia relativa de las variables es prácticamente cero.

```
#ploteamos

#Pero en primer lugar reestructuramos el df
importancia_df = importancia_df.set_index('Variables')
importancia_df
```

Variables	Importancia	Acumulado
Latitud	3.260510e-01	0.326051
NDVI_84	2.357048e-01	0.561756
Longitud	2.320908e-01	0.793847

```
fig, ax = plt.subplots(1,2, figsize=(12,10))
importancia_df['Importancia'].sort_values(ascending=True).plot(kind='barh', ax=ax[0])
ax[0].grid(True)

importancia_df['Acumulado'].plot(kind='line', ax=ax[1])
ax[1].grid(True)
plt.xticks(rotation=70)
```

```
(array([-10.,   0.,  10.,  20.,  30.,  40.,  50.]),
 <a list of 7 Text major ticklabel objects>)
```




```
#Lanzamiento Modelo1
```

```
#Este es un modelo simplificado desechando las variables no relevantes según el criterio del
#gradiente descendente
```

```
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.svm import SVR
```

```
df_simplificado = X[['Latitud', 'NDVI_84', 'Longitud', 'Altura','Temp_med', 'Precip_med']]
df_simplificado.head()
```

	Latitud	NDVI_84	Longitud	Altura	Temp_med	Precip_med
0	1.375	-15.300813	-32.0	-4.741553	1.544470	-255.6000
1	1.375	-16.756098	-31.0	-4.554673	4.930946	-278.5336
2	1.125	-13.617886	-31.0	-4.622268	4.187602	-116.8000
3	1.000	-8.073171	-31.0	-4.729624	3.394702	33.5336
4	0.875	-4.146341	-31.0	-4.693839	3.097365	110.4000

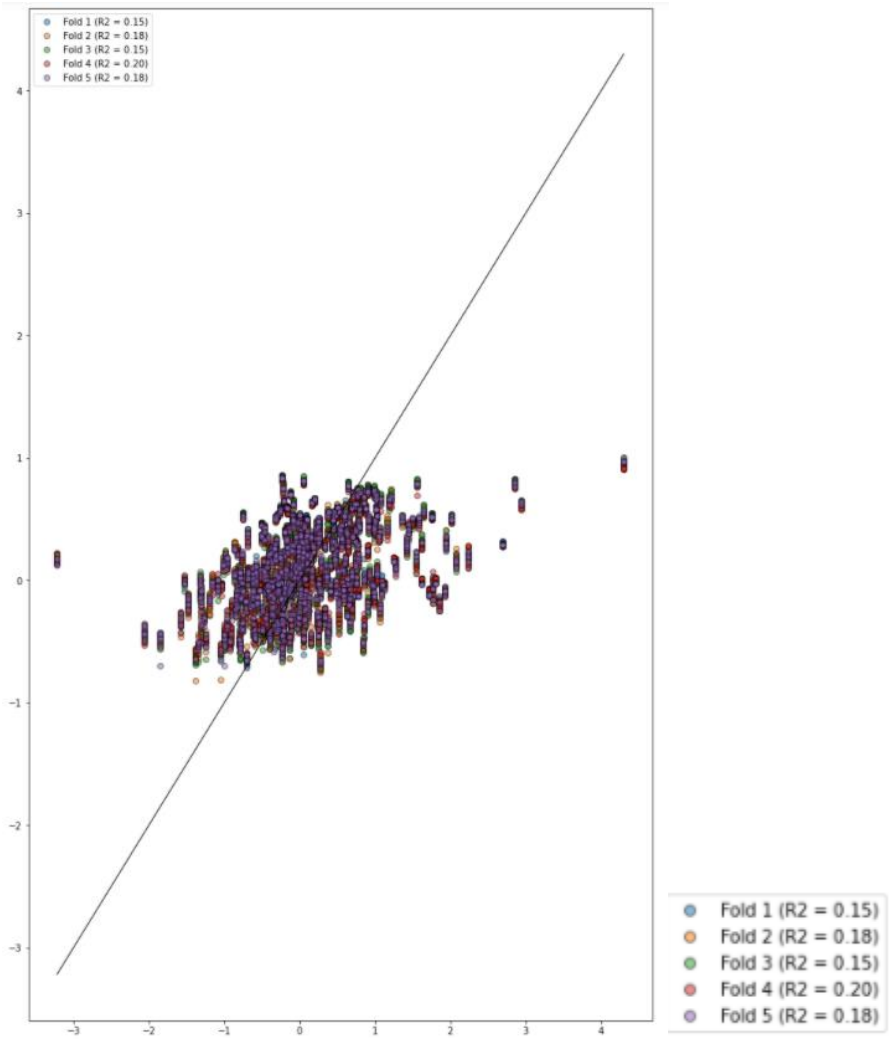
```
#ahora podemos escalar las variables

x_simplificado_escalado = RobustScaler(quantile_range=(25.0, 75.0))
x_simply_std = x_simplificado_escalado.fit_transform(df_simplificado)
```

```
y_simplificado_escalado = RobustScaler(quantile_range=(25.0, 75.0))
y_simply_std = y_simplificado_escalado.fit_transform(Y)
```

```
modelo_lineal_simplificado = LinearRegression()
best_svr = SVR(kernel='rbf')
```

```
from sklearn.model_selection import train_test_split
x_train2, x_test2, y_train2, y_test2 = train_test_split(x_simply_std, y_simply_std, test_size=0.2,
```

```
resultados_modelo_lineal = []
cv = KFold(n_splits=5)
i=1
fig = plt.figure(figsize=(12,20))
for train_index, test_index in cv.split(x_train2, y_train2):
    modelo_lineal_simplificado.fit(x_train2[train_index], y_train2[train_index])
    prediccion = modelo_lineal_simplificado.predict(x_train2[test_index])

    metrica = r2_score(y_train2[test_index],prediccion)
    resultados_modelo_lineal.append(metrica)
    plt.scatter(y_train2[test_index], prediccion, alpha=0.5, edgecolor='black', label='Fold %d (R2
    i+=1
plt.plot([min(y_train2),max(y_train2)], [min(y_train2),max(y_train2)], lw=1, color='black')
plt.legend()
```




```
1 resultados_modelo_lineal
```

```
[0.14760605452413977,
 0.17927588822157592,
 0.1453484733892646,
 0.19920163355040765,
 0.1771302313270694]
```

```
1 np.mean(resultados_modelo_lineal)
```

```
0.16971245620249148
```

A continuación, vamos a lanzar un segundo modelo en el que el criterio de selección de variables fue el de regularización de Ridge juntamente con el de las variables significativas según el p-valor.

```
#Lanzamiento Modelo2

#Este es un modelo simplificado desechando las variables no relevantes según
#p-value y regularización de Ridge
```

```
busqueda = df_features.loc[(df_features['Metodo_2']=='ambas')]
busqueda
```

	Feature_Names	p-valor	num_1	Ridge	num_2	Metodo	Metodo_2
8	SSP+3	p-valor	1	Ridge	1	p-valor Ridge	ambas
9	SSP+1	p-valor	1	Ridge	1	p-valor Ridge	ambas
10	SSP+2	p-valor	1	Ridge	1	p-valor Ridge	ambas
11	SSP-1_2)	p-valor	1	Ridge	1	p-valor Ridge	ambas

```
variables_segindo_modelo = busqueda['Feature_Names'].tolist()
variables_segindo_modelo
```

```
df_simplificado2 = X[['SSP+3',
 'SSP-2_3',
 'SSP-3',
 'SSP-1',
 'SSP-2',
 'SST+2_3',
 'SST+3',
 'SST-2_3',
 'SST-3',
 'SSTP++1',
 'SSTP-+1',
 'SSTP++(2)',
 'SSTP+-(2)',
 'SSTP-+(2)',
 'SSTP++(2_1)',
 'SSTP--(2_1)']]
df_simplificado2.head()
```

	SSP+3	SSP-2_3	SSP-3	SSP-1	SSP-2	SST+2_3	SST+3	SST-2_3	SST-3	SSTP++1	SSTP-+1
0	0.0	0.0	0.0	0.029412	0.0	0.0	0.0	0.088235	0.000000	0.0	0.029412
1	0.0	0.0	0.0	0.058824	0.0	0.0	0.0	0.029412	0.000000	0.0	0.000000
2	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.088235	0.000000	0.0	0.000000
3	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.058824	0.000000	0.0	0.000000
4	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.058824	0.029412	0.0	0.000000

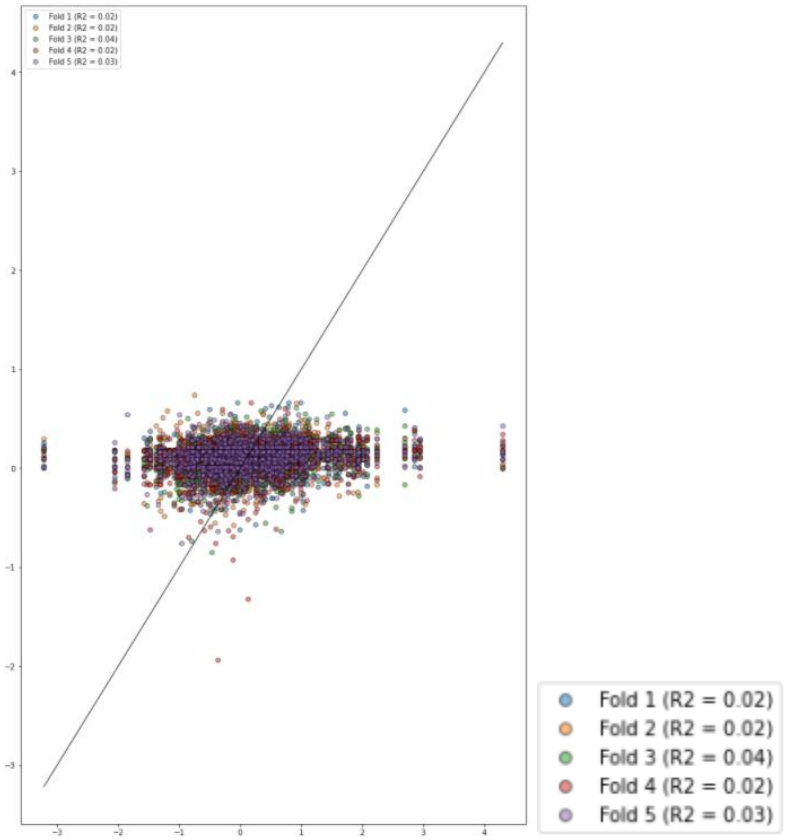
```
x_simplificado_escalado2 = RobustScaler(quantile_range=(25.0, 75.0))
x_simply_std2 = x_simplificado_escalado2.fit_transform(df_simplificado2)
```

```
y_simplificado_escalado2 = RobustScaler(quantile_range=(25.0, 75.0))
y_simply_std2 = y_simplificado_escalado2.fit_transform(Y)
```

```
modelo_lineal_simplificado2 = LinearRegression()
best_svr2 = SVR(kernel='rbf')
```

```
from sklearn.model_selection import train_test_split
x_train3, x_test3, y_train3, y_test3 = train_test_split(x_simply_std2, y_sim
```

```
resultados_modelo_lineal2 = []
cv = KFold(n_splits=5)
i=1
fig = plt.figure(figsize=(12,20))
for train_index, test_index in cv.split(x_train3, y_train3):
    modelo_lineal_simplificado2.fit(x_train3[train_index], y_train3[train_in
    prediccion2 = modelo_lineal_simplificado2.predict(x_train3[test_index])

    metrica2 = r2_score(y_train3[test_index],prediccion2)
    resultados_modelo_lineal2.append(metrica2)
    plt.scatter(y_train3[test_index], prediccion2, alpha=0.5, edgecolor='bla
    i+=1
plt.plot([min(y_train3),max(y_train3)], [min(y_train3),max(y_train3)], lw=1,
plt.legend()
```



```
resultados_modelo_lineal_2
```

```
[0.020012133310201907,
 0.015357175883874086,
 0.035453008942841646,
 0.02492756414986297,
 0.027136350239258977]
```

```
np.mean(resultados_modelo_lineal_2)
```

```
0.024577246505207918
```

A tenor de los resultados queda claro que el primer modelo es capaz de explicar mejor nuestra variable objetivo.

```
coeficient = modelo_lineal_simplificado.coef_.tolist()
np.shape(coeficient)
```

```
(1, 6)
```

```
coeficient = np.reshape(coeficient,(6,))
```

```
coeficient2 = coeficient.tolist()
coeficient2
```

```
[-0.10579685999542635,
 -0.38660152965913575,
 -0.27212415605779616,
 0.04108156519614921,
 -0.04565584902949926,
 -0.19217404089457432]
```

```
variab2 = df_simplificado.columns.tolist()
variab2
```

```
['Latitud', 'NDVI_84', 'Longitud', 'Altura', 'Temp_med', 'Precip_med']
```

```
#Coeficientes_primer_modelo
df_coeficientes1 = pd.DataFrame({'Variables': variab2,
                                 'Coeficientes': coeficient2})
df_coeficientes1
```

	Variables	Coeficientes
0	Latitud	-0.105797
1	NDVI_84	-0.386602
2	Longitud	-0.272124
3	Altura	0.041082
4	Temp_med	-0.045656
5	Precip_med	-0.192174

Modelo de regresión basado en MCO.

A diferencia del modelo anterior vamos a volver a lanzar el modelo pero en esta ocasión no utilizaremos técnicas de preprocesamiento ni análisis exploratorio de datos utilizados en los algoritmos de machine learning.

Como se puede observar el punto de partida es exactamente el mismo que en el apéndice anterior.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

```
df = pd.read_csv('Cuadro_NDVI_para_REGRESION.csv').drop('Unnamed: 0', axis=1)
df
```

	Latitud	Longitud	Year	Altura	Precip_med	Temp_med	NDVI	SSP+1_2	SSP+2_3	SSP+3	...	SSTP++ (2_1)	SSTP+ (2_1
0	14.75	-16.75	1984	14.0000	34.6500	25.8833	0.2414	0.029412	0.000000	0.000000	...	0.000000	0.(
1	14.75	-15.75	1984	29.6667	31.7833	27.5917	0.2235	0.000000	0.000000	0.000000	...	0.000000	0.(
2	13.75	-15.75	1984	24.0000	52.0000	27.2167	0.2621	0.000000	0.000000	0.000000	...	0.000000	0.(
3	13.25	-15.75	1984	15.0000	70.7917	26.8167	0.3303	0.000000	0.000000	0.000000	...	0.000000	0.(
4	12.75	-15.75	1984	18.0000	80.4000	26.6667	0.3786	0.000000	0.000000	0.000000	...	0.000000	0.(
...	...	...	...	...	...	...	...	...	...	...	...	...	..
12123	3.25	46.25	2017	188.0000	33.5333	28.5167	0.3591	0.029412	0.029412	0.000000	...	0.029412	0.(
12124	5.25	47.25	2017	136.7500	16.6500	29.4417	0.2203	0.000000	0.000000	0.029412	...	0.000000	0.(
12125	10.25	48.25	2017	1118.5000	12.7667	24.6417	0.2065	0.000000	0.000000	0.029412	...	0.000000	0.(
12126	6.75	48.25	2017	216.5000	15.8250	28.8917	0.1970	0.000000	0.000000	0.029412	...	0.000000	0.(
12127	6.25	48.25	2017	162.2500	16.9750	28.8917	0.2448	0.000000	0.000000	0.029412	...	0.000000	0.(

12128 rows × 49 columns

Lo realmente interesante es la implementación de un código que nos ahorra hacer la regresión hacia atrás tantas veces como variables no significativas nos aparezcan. Teniendo en cuenta que nuestro punto de partida son 48 variables predictoras y que en caso de aparecer variables no significativas han de ser eliminadas de una en una en orden de mayor p-valor, el proceso puede eternizarse y deja de ser eficiente en casos donde la dimensión de nuestro cuadro de datos sea muy grande. Para que se entienda mejor qué es lo que hemos hecho empezare lanzando el modelo de regresión basado en MCO.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

```
df1 = pd.read_csv('Cuadro_NDVI_para_REGRESION.csv').drop(['Unnamed: 0', 'NDVI'], axis=1)
df1
```

```
XX = df1.iloc[:, :-1].values
XX
```

```
yy = df1.iloc[:,47].values
```

```
XX = np.append(arr = np.ones((12128,1)).astype(int), values=X, axis=1)
```

```
modelo_xx = sm.OLS(yy, XX).fit()
```

```
print(modelo_xx.summary())
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.183
Model:	OLS	Adj. R-squared:	0.181
Method:	Least Squares	F-statistic:	79.78
Date:	Wed, 16 Sep 2020	Prob (F-statistic):	0.00
Time:	16:16:11	Log-Likelihood:	21506.
No. Observations:	12128	AIC:	-4.294e+04
Df Residuals:	12093	BIC:	-4.268e+04
Df Model:	34		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0639	0.048	1.332	0.183	-0.030	0.158
x1	0.0639	0.048	1.332	0.183	-0.030	0.158
x2	-0.0004	3.22e-05	-12.145	0.000	-0.000	-0.000
x3	-0.0006	2.88e-05	-20.454	0.000	-0.001	-0.001
x4	-1.356e-06	4.79e-05	-0.028	0.977	-9.52e-05	9.25e-05
x5	2.85e-06	1.25e-06	2.278	0.023	3.98e-07	5.3e-06
x6	-0.0002	1.21e-05	-12.959	0.000	-0.000	-0.000
x7	-0.0005	0.000	-3.723	0.000	-0.001	-0.000
x8	-0.0117	0.009	-1.265	0.206	-0.030	0.006
x9	-0.0138	0.016	-0.873	0.383	-0.045	0.017
x10	0.0293	0.016	1.793	0.073	-0.003	0.061
x11	0.0037	0.008	0.475	0.635	-0.012	0.019
x12	0.0154	0.009	1.803	0.071	-0.001	0.032
x13	0.0841	0.037	2.290	0.022	0.012	0.156
x14	0.1962	0.110	1.788	0.074	-0.019	0.411
x15	-0.4223	0.171	-2.467	0.014	-0.758	-0.087
x16	-0.1420	0.036	-3.933	0.000	-0.213	-0.071
x17	-0.2261	0.071	-3.178	0.001	-0.366	-0.087
x18	0.0156	0.012	1.359	0.174	-0.007	0.038
x19	0.1182	0.032	3.638	0.000	0.055	0.182
x20	-0.1479	0.047	-3.146	0.002	-0.240	-0.056
x21	-0.0140	0.010	-1.381	0.167	-0.034	0.006
x22	-0.0297	0.019	-1.561	0.119	-0.067	0.008
x23	0.0091	0.011	0.830	0.406	-0.012	0.031
x24	-0.0957	0.043	-2.221	0.026	-0.180	-0.011
x25	0.0888	0.058	1.544	0.123	-0.024	0.202
x26	0.0023	0.010	0.237	0.812	-0.016	0.021
x27	-0.0069	0.018	-0.389	0.697	-0.041	0.028
x28	-0.0312	0.036	-0.855	0.393	-0.103	0.040
x29	0.0206	0.032	0.647	0.517	-0.042	0.083
x30	0.0409	0.027	1.495	0.135	-0.013	0.095
x31	-0.0376	0.047	-0.797	0.426	-0.130	0.055
x32	-0.0402	0.170	-0.236	0.813	-0.374	0.294
x33	-0.4012	0.185	-2.169	0.030	-0.764	-0.039
x34	-0.1639	0.103	-1.589	0.112	-0.366	0.038
x35	0.2660	0.215	1.237	0.216	-0.155	0.687
x36	0	0	nan	nan	0	0
x37	0.1389	0.120	1.161	0.246	-0.096	0.374
x38	-0.0069	0.018	-0.389	0.697	-0.041	0.028
x39	0.2660	0.215	1.237	0.216	-0.155	0.687
x40	-0.2129	0.122	-1.742	0.082	-0.453	0.027
x41	-0.0012	0.061	-0.019	0.985	-0.121	0.119
x42	0.1464	0.081	1.815	0.070	-0.012	0.304
x43	0.2438	0.120	2.030	0.042	0.008	0.479
x44	1.4959	0.844	1.773	0.076	-0.158	3.149
x45	0	0	nan	nan	0	0
x46	-0.3428	0.394	-0.871	0.384	-1.115	0.429
x47	0	0	nan	nan	0	0
x48	-0.0822	0.004	-20.786	0.000	-0.090	-0.074

Omnibus:	1074.789	Durbin-Watson:	1.382
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2638.291
Skew:	0.530	Prob(JB):	0.00
Kurtosis:	5.024	Cond. No.	1.10e+17

Si nos fijamos en la columna del **p>|t|** que es la que hace referencia a los p-valor de cada una de las variables, vemos que hay muchas variables con p-valor por encima del nivel de significación (5%) por lo que son variables que deberíamos eliminar. En ese caso el procedimiento sería fijarnos en aquella que tiene un p-valor mayor y eliminarla. A continuación, volveríamos a lanzar el modelo y haríamos lo mismo hasta que todas las variables cuenten con un p-valor por debajo de 0.05.

Nosotros hemos desarrollado un código que automatiza el procedimiento de eliminación de variables según este criterio y con sólo un lanzamiento tendríamos el resultado esperado.

```python
import statsmodels.api as sm
def backwardElimination(x, sl):
    numVars = len(x[0])
    for i in range(0, numVars):
        regressor_OLS = sm.OLS(y, x.tolist()).fit()
        maxVar = max(regressor_OLS.pvalues).astype(float)
        if maxVar > sl:
            for j in range(0, numVars - i):
                if (regressor_OLS.pvalues[j].astype(float) == maxVar):
                    x = np.delete(x, j, 1)
    regressor_OLS.summary()
    return x

SL = 0.05
X_opt = X[:, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23
X_Modeled = backwardElimination(X_opt, SL)
```

```python
modelo_1 = sm.OLS(y, X_Modeled.tolist()).fit()
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.182
Model:	OLS	Adj. R-squared:	0.181
Method:	Least Squares	F-statistic:	168.4
Date:	Wed, 16 Sep 2020	Prob (F-statistic):	0.00
Time:	16:25:17	Log-Likelihood:	21497.
No. Observations:	12128	AIC:	-4.296e+04
Df Residuals:	12111	BIC:	-4.283e+04
Df Model:	16		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1254	0.004	30.576	0.000	0.117	0.133
x1	-0.0004	3.2e-05	-12.197	0.000	-0.000	-0.000
x2	-0.0006	2.87e-05	-20.551	0.000	-0.001	-0.001
x3	2.905e-06	1.24e-06	2.336	0.019	4.68e-07	5.34e-06
x4	-0.0002	1.17e-05	-13.801	0.000	-0.000	-0.000
x5	-0.0005	0.000	-3.623	0.000	-0.001	-0.000
x6	0.0727	0.035	2.055	0.040	0.003	0.142
x7	0.2317	0.108	2.152	0.031	0.021	0.443
x8	-0.4342	0.171	-2.539	0.011	-0.769	-0.099
x9	-0.1299	0.035	-3.745	0.000	-0.198	-0.062
x10	-0.2025	0.069	-2.937	0.003	-0.338	-0.067
x11	0.0770	0.020	3.772	0.000	0.037	0.117
x12	-0.1713	0.071	-2.415	0.016	-0.310	-0.032
x13	-0.0910	0.025	-3.701	0.000	-0.139	-0.043
x14	0.0646	0.019	3.383	0.001	0.027	0.102
x15	-0.3168	0.161	-1.965	0.049	-0.633	-0.001
x16	-0.2380	0.077	-3.110	0.002	-0.388	-0.088
x17	0.2474	0.097	2.549	0.011	0.057	0.438
x18	0	0	nan	nan	0	0
x19	0	0	nan	nan	0	0
x20	-0.0824	0.004	-21.009	0.000	-0.090	-0.075

Omnibus:	1087.330	Durbin-Watson:	1.377
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2677.372
Skew:	0.535	Prob(JB):	0.00
Kurtosis:	5.038	Cond. No.	inf

Si observamos, el modelo se ha quedado con veinte de las 47 variables y el nivel de precisión es muy parecido al que obtuvimos en el apéndice anterior donde seleccionamos variables basándonos en el criterio del gradiente descendente.

Figura 1. Localización de las Crop_áreas.

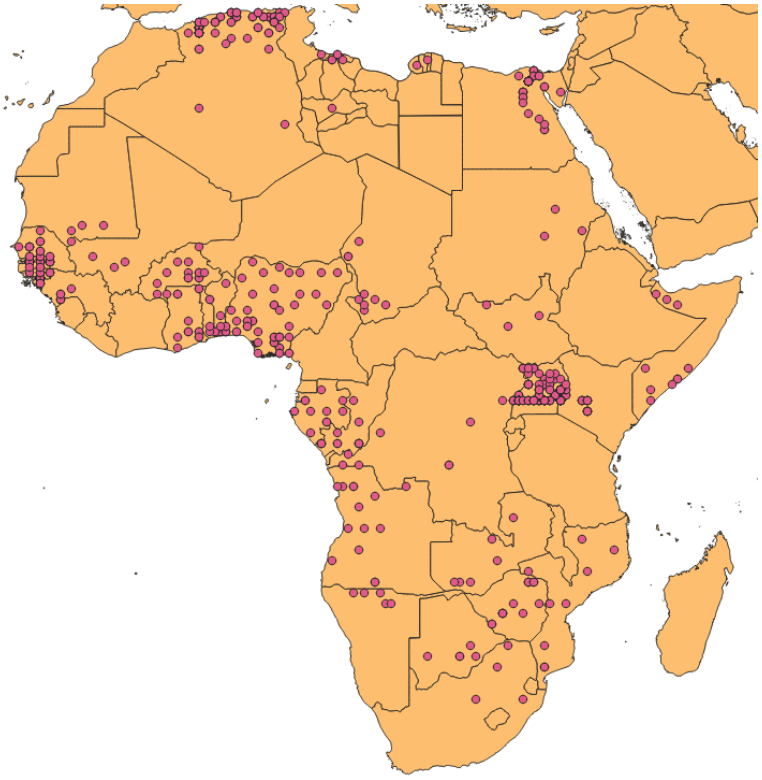


Figura 2. NDVI en las crop_áreas.

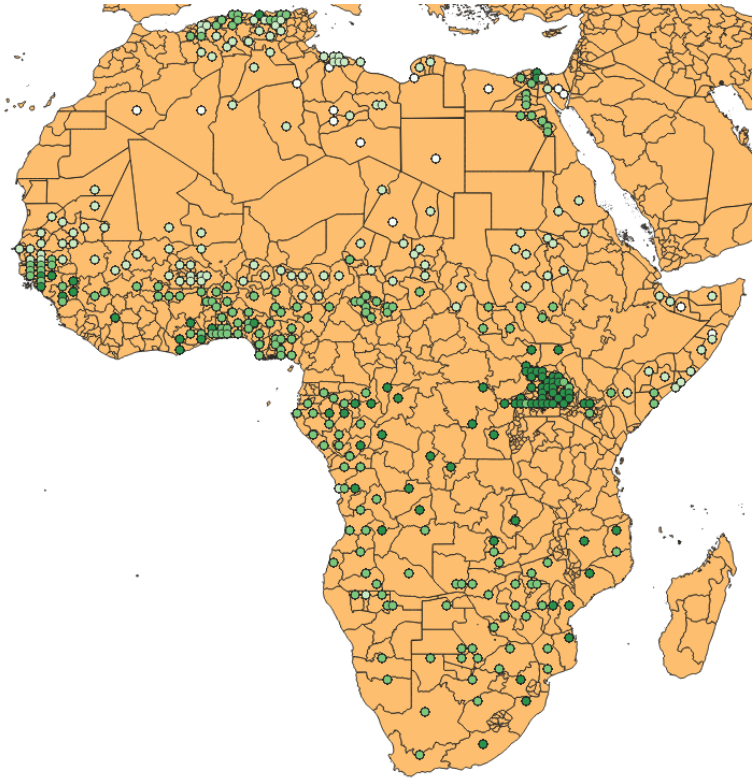


Figura 3. Precipitaciones medias anuales en las crop_áreas.

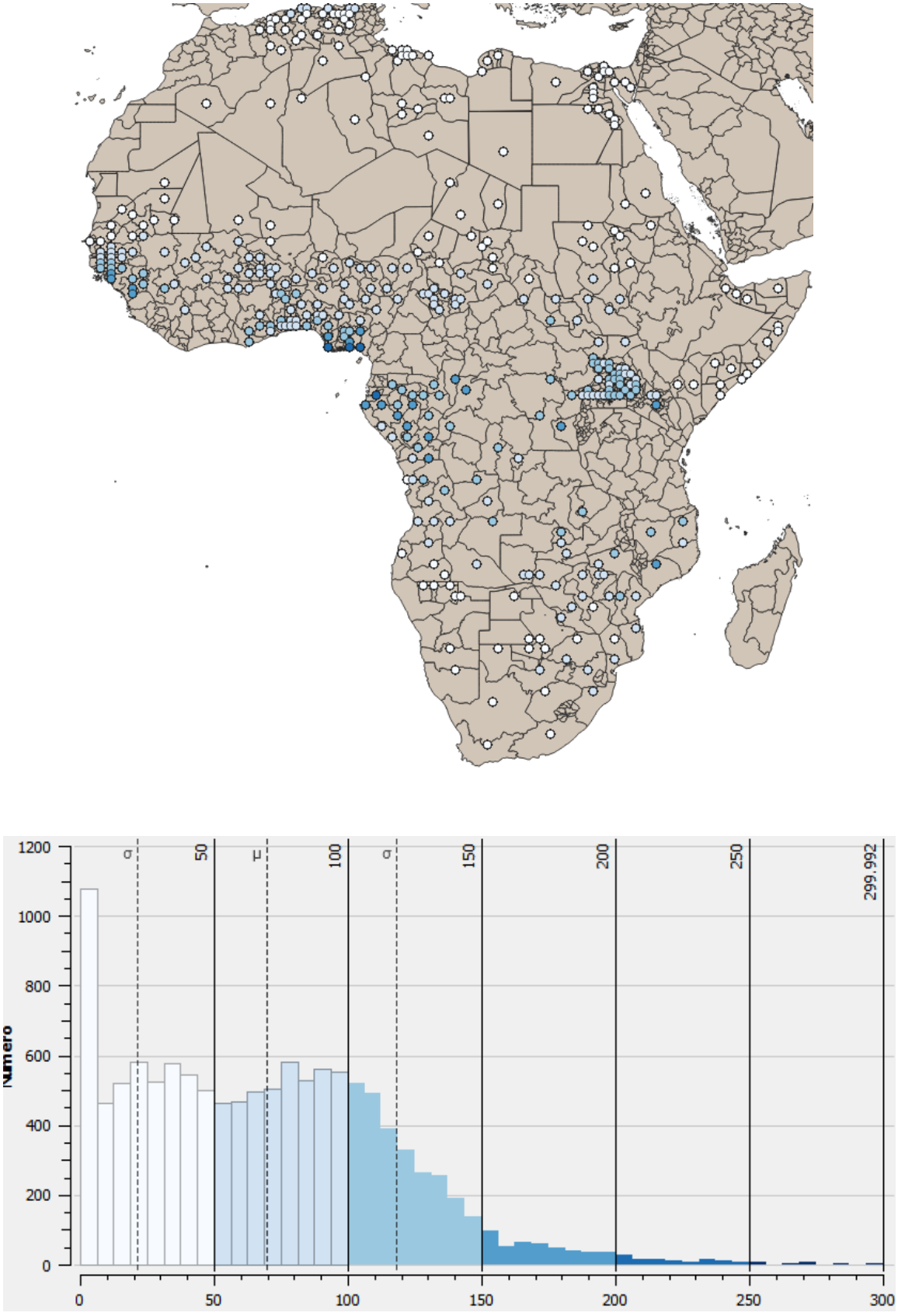
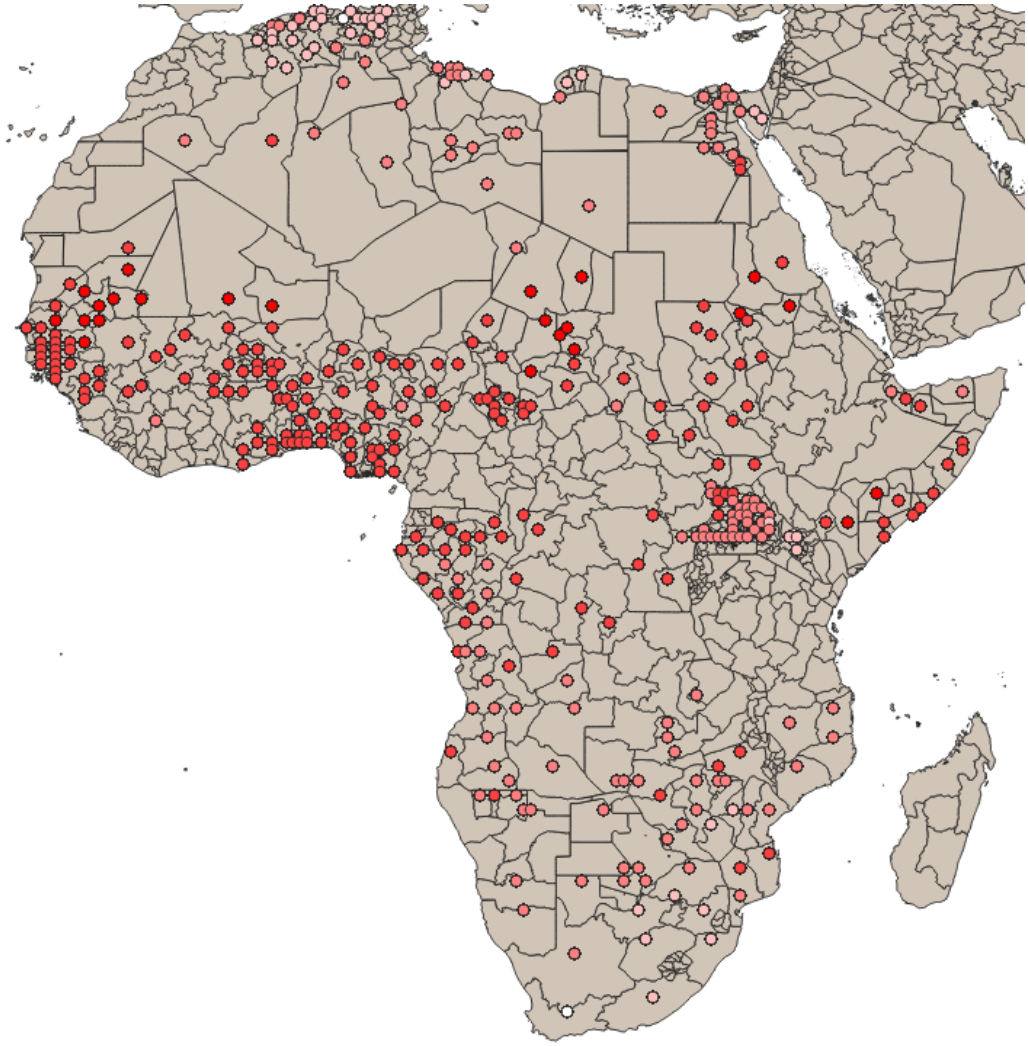
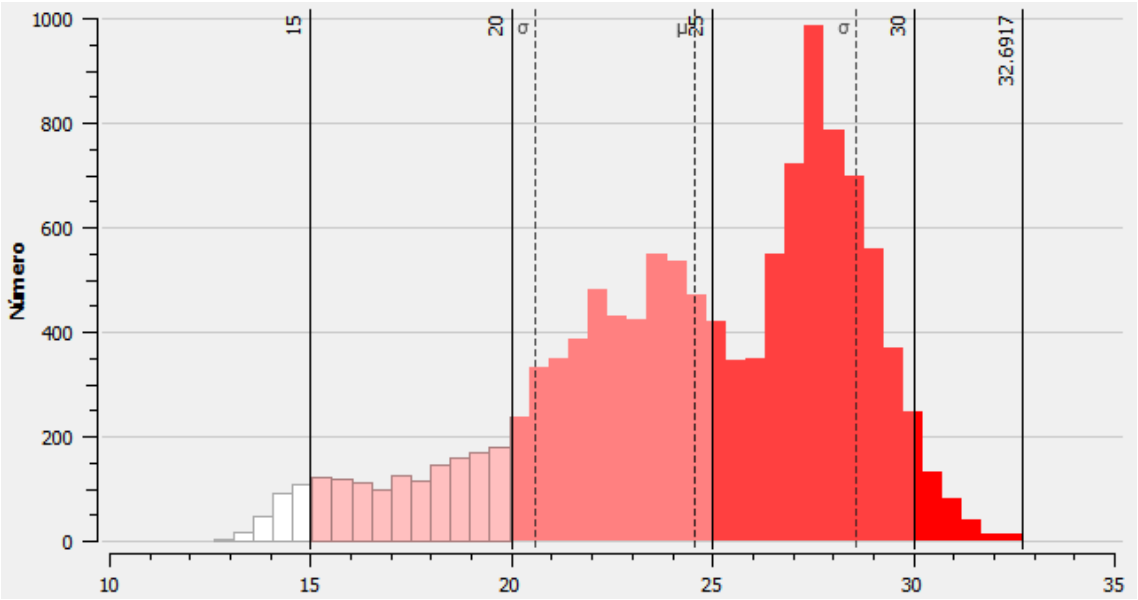


Figura 4. Temperaturas medias anuales en las crop_áreas.




1000
800
600
400
200
0
Número
15
20
σ
25
σ
30
32.6917
10
15
20
25
30
35

Anexo 6 – Salud Infantil

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Cargamos los datos

```
pd.set_option('display.max_columns', 50)
```

```
df = pd.read_csv('tabla_analisis_salud.csv').drop(['Unnamed: 0','field_1'], axis=1)
```

```
df
```

	country_co	cluster_nu	household_	month_inte	yr_intervi	respond_mo	respond_yr	respond_cu	region	type_resid	mother_edu	drinking_w
0	BF4	1.0	76.0	10.0	2003.0	5.0	1968.0	35.0	9.0	rural	noedu	1.0
1	BF4	1.0	98.0	10.0	2003.0	5.0	1972.0	31.0	9.0	rural	noedu	1.0

Análisis Exploratorio de Datos

```
#Comenzamos pidiendo información de las variables que tenemos

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201508 entries, 0 to 201507
Data columns (total 46 columns):
 #   Column      Non-Null Count   Dtype  
---  ------      --------------   -----  
 0   country_co  201508 non-null  object 
 1   cluster_nu  201508 non-null  float64
 2   household_  201508 non-null  float64
 42  field_1_2   201508 non-null  int64  
 43  stxv5       201508 non-null  float64
 44  Longitud_2  201508 non-null  float64
 45  Latitud_2   201508 non-null  float64
dtypes: bool(1), float64(37), int64(1), object(7)
memory usage: 69.4+ MB
```

```
#El siguiente paso que haremos es el de ccategorizar variables
#Todas aquellas variables numericas que tengan menos de 10 datos únicos las vamos a categorizar
```

```
#creamos una lista para almacenar las variables
variables1 = df.columns.tolist()
```

```
#Para hacernos una idea de los valores únicos por variables

from ipywidgets import interact
def valores_unicos(var):
    print(f"La variable {var} presenta {df[var].nunique()} valores unicos")
    return df[var].unique()

interact(valores_unicos, var=variables1)
```

var country_co

```
La variable country_co presenta 34 valores unicos

array(['BF4', 'BF6', 'BJ4', 'BJ6', 'CD5', 'CD6', 'CI6', 'CM4', 'CM6',
       'ET4', 'ET6', 'GA6', 'GH4', 'GH5', 'GN4', 'GN6', 'KE4', 'KE5',
       'LB6', 'ML4', 'ML5', 'ML6', 'MW4', 'MW5', 'MZ6', 'NG5', 'NG6',
       'SL5', 'SL6', 'SN6', 'TG6', 'UG6', 'ZM5', 'ZM6'], dtype=object)
```

```python
#A continuación definimos una función para determinar el tipo de variables en base a sus valores únicos
def definir_tipo(var):
    if (df[var].dtype==object) | (df[var].nunique()<15):
        return 'categorica'
    else:
        return 'numerica'
```

```python
#El resultado del tipo de variable lo vamos a hacer por medio de un diccionario
dic_variables1 = {var : definir_tipo(var) for var in df.columns}
```

```python
#A continuación cada tipo de variable la vamos a almacenar en una lista

contenedor_categoricas = []
contenedor_numericas = []
for clave, valor in dic_variables1.items():
        if valor == 'categorica':
            contenedor_categoricas.append(clave)
        else:
            contenedor_numericas.append(clave)
```

```python
print(f"Exiten {len(contenedor_categoricas)} variables categóricas y {len(contenedor_numericas)} numericas")
```

```
Exiten 15 variables categóricas y 31 numericas
```

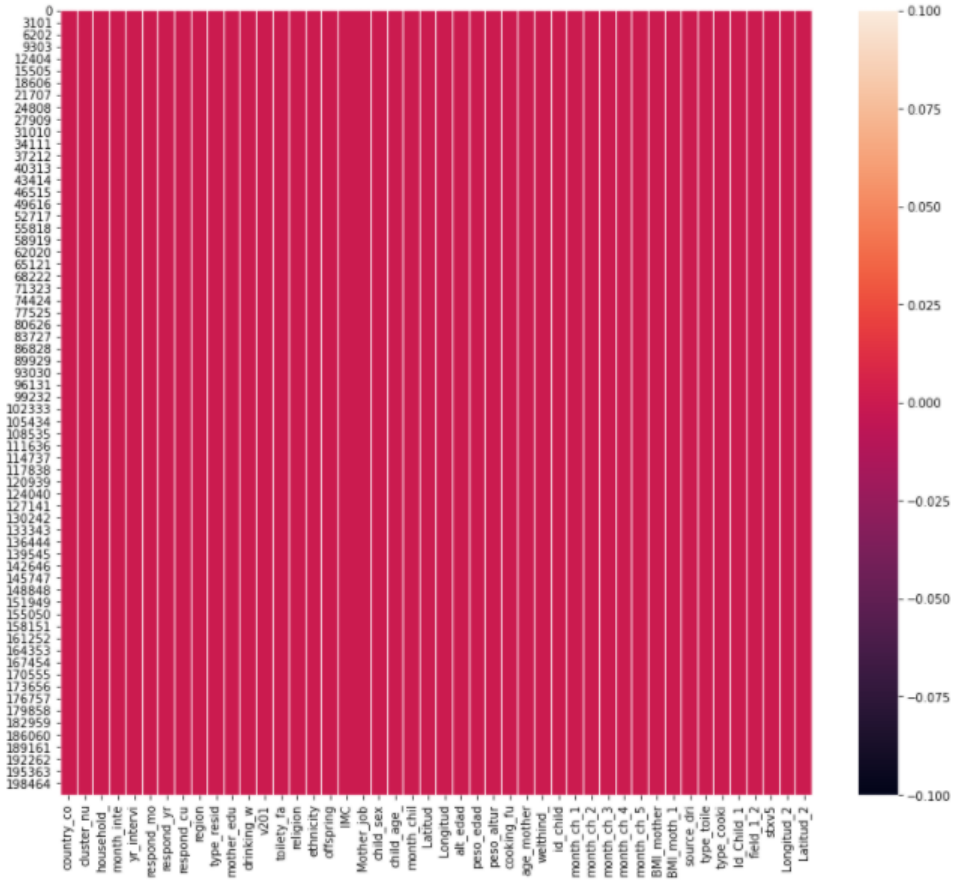
```python
#Estudio de los valores faltantes
#1º forma
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201508 entries, 0 to 201507
Data columns (total 46 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   country_co  201508 non-null  object
 1   cluster_nu  201508 non-null  float64
 2   household_  201508 non-null  float64
 42  field_1_2   201508 non-null  int64
 43  stxv5       201508 non-null  float64
 44  Longitud_2  201508 non-null  float64
 45  Latitud_2   201508 non-null  float64
dtypes: bool(1), float64(37), int64(1), object(7)
memory usage: 69.4+ MB
```

```python
#2º forma. A través de un mapa de calor

#Empezamos creando una matriz de valores booleans
df_faltantes = df.isnull()
#lo que haremos a continuación es sustituir los 'False' por 1 en caso de ser una var numéricas
#y por 2 en caso de ser una variable categórica
for clave, valor in dic_variables1.items():
    if valor =='numericas':
        df_faltantes[clave] = df_faltantes[clave].apply(lambda x: 1 if x is False else 0)
    else:
        df_faltantes[clave] = df_faltantes[clave].apply(lambda x: 0 if x is False else 1)
```

```
fif, ax = plt.subplots(figsize=(14,12))
sns.heatmap(df_faltantes)
```

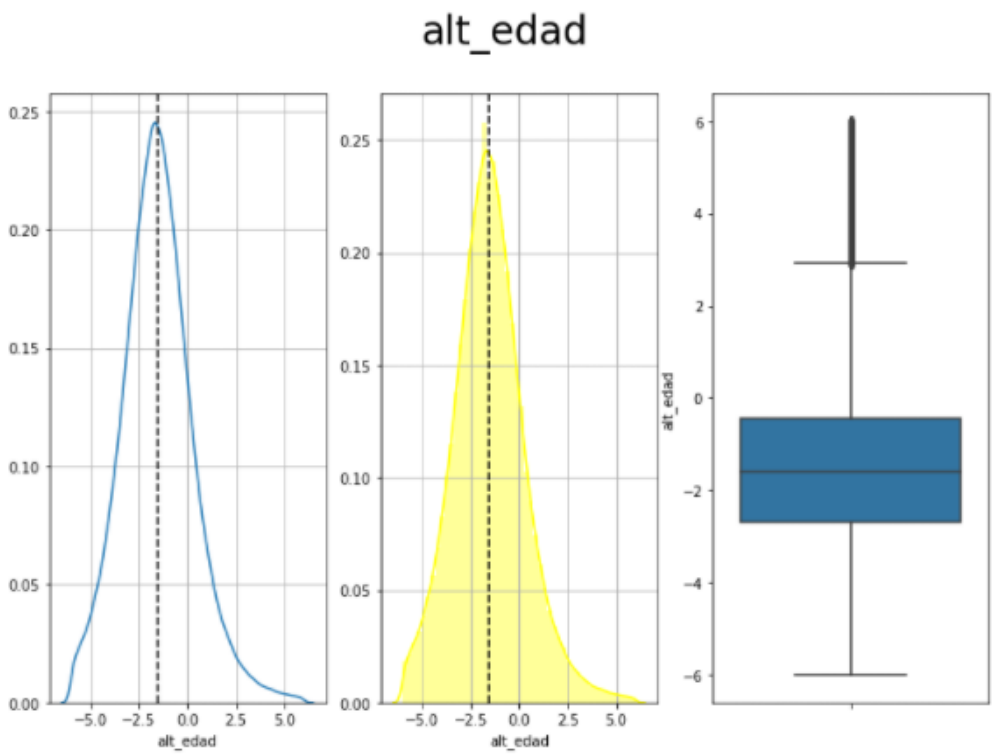
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bf763cd8b0>
```




```
desc = df['alt_edad'].describe()
desc
```

```
count    201508.000000
mean         -1.516701
std           1.832740
min          -6.000000
25%          -2.700000
50%          -1.590000
75%          -0.450000
max           6.000000
Name: alt_edad, dtype: float64
```

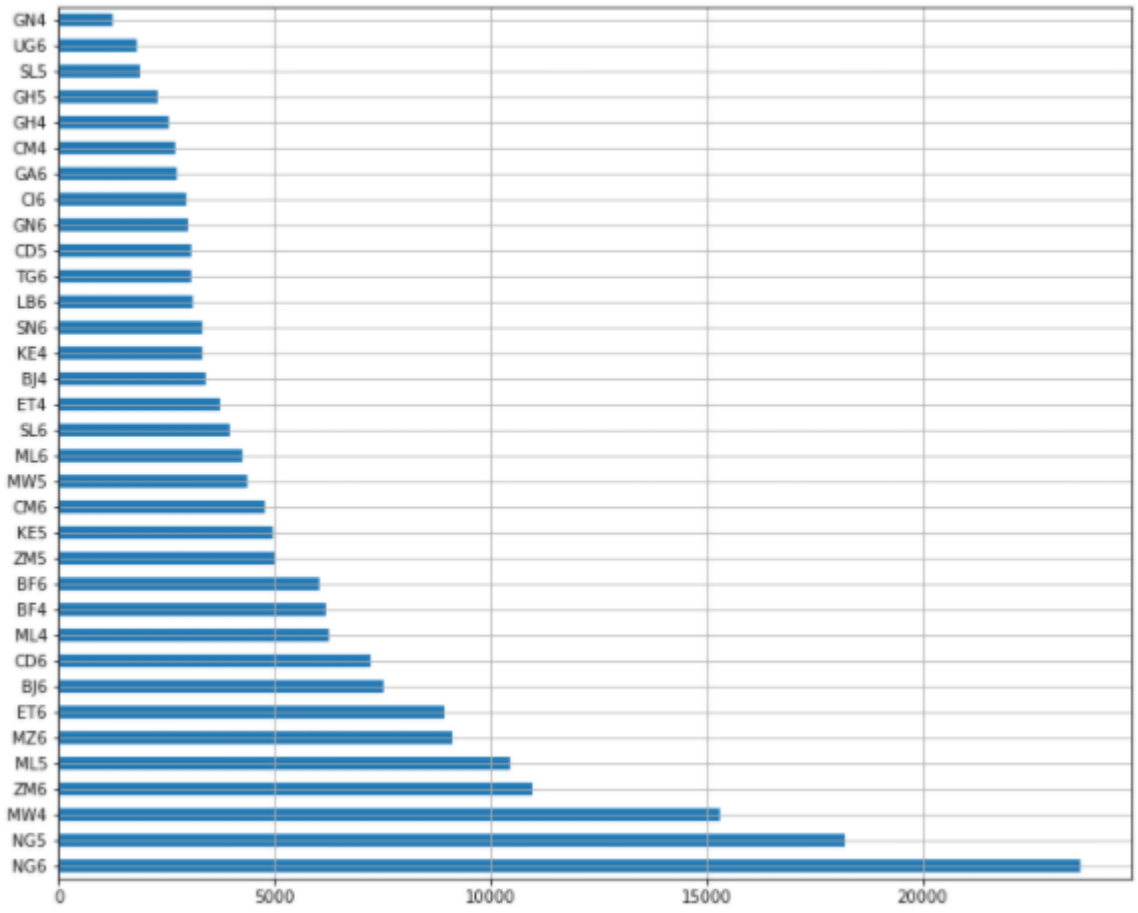
```
#gráficamente

fig, ax = plt.subplots(1, 3, figsize=(12, 8))
fig.suptitle('alt_edad', fontsize=(28))

sns.distplot(df['alt_edad'], hist=False, ax = ax[0])
ax[0].axvline(desc['mean'], ls='--', color = 'black')
ax[0].grid(True)

sns.distplot(df['alt_edad'], hist=True, color='yellow', ax = ax[1])
ax[1].axvline(desc['mean'], ls='--', color = 'black')
ax[1].grid(True)

sns.boxplot(df['alt_edad'], orient='vertival')
```


alt_edad
0.25
0.20
0.15
0.10
0.05
0.00
−5.0
−2.5
0.0
2.5
5.0
alt_edad
6
4
2
0
−2
−4
−6
alt_edad

Estudio de las variables categóricas

```
#De una forma interactiva queremos ver cuántas etiquetas tiene cada variable

def contar_etiquetas(var):
    fig,ax = plt.subplots(figsize=(12,10))
    return df[var].value_counts().sort_values(ascending = False).plot(kind='barh', grid=True)
interact(contar_etiquetas, var = contenedor_categoricas)
```

var country_co



```
#A continuación vamos a representar los 'kde' de cada etiqueta ppor variable
#Si las diferentes funciones de distribución por etiquetas son muy parecidas entre si
#Es muy probable que estemos ante un caso de variable no significativa.

colores = plt.cm.Dark2(range(6))
def kde_categoricas_por_etiquetas(var):
    fig, ax = plt.subplots(figsize=(10,8))

    for i in df[var].unique():
        #k = int(i)
        sns.distplot(df[df[var]==i]['alt_edad'], hist=False)
        ax.axvline(desc['mean'], ls='--', color='black')
interact(kde_categoricas_por_etiquetas, var=contenedor_categoricas)
```

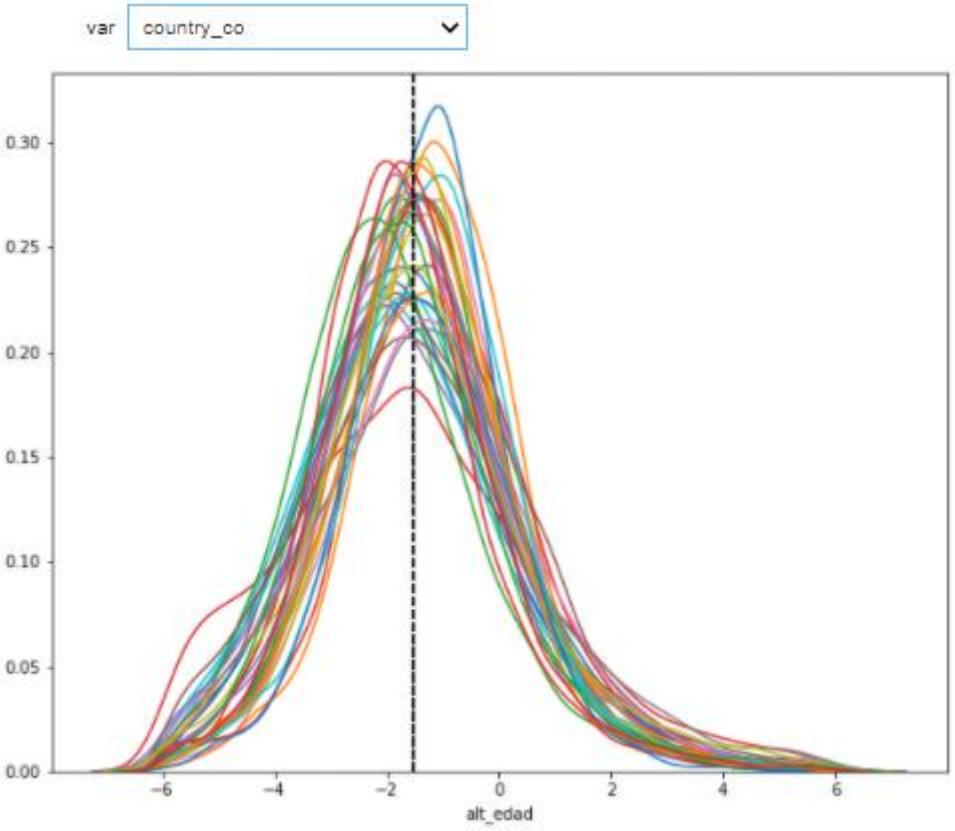


Gráfico de barras acumulativas

```
#Empezamos definiendo los intervalos que queremos construir respecto a los valores de la variable obj.
intervalos = np.quantile(df['alt_edad'], q=np.linspace(0,1,6))
intervalos
```

```
array([-6.  , -2.99, -1.99, -1.17, -0.14,  6.  ])
```

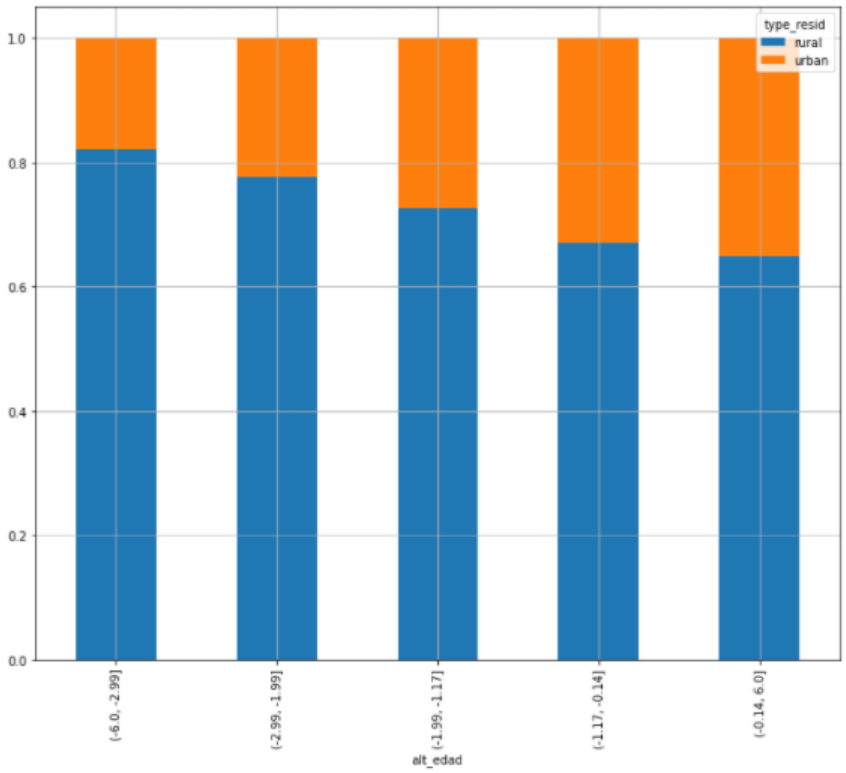
```
#Elegimos una variable y para cada etiqueta de esa variable contamos cuantos valores hay en cada interv
tmp = df.groupby([df['type_resid'], pd.cut(df['alt_edad'], intervalos, duplicates='drop')]).size().unstack().T
tmp
```

```
#El siguiente paso es crear una columna donde se sumen todos los valores que hay por intervalo
tmp['Total'] = tmp.sum(axis=1)
tmp
```

```
#Ahora lo que haremos es poner los valores de la tabla como porcentajes
for i in tmp.drop('Total', axis=1).columns:
    tmp[i] = tmp[i]/tmp['Total']
tmp
```

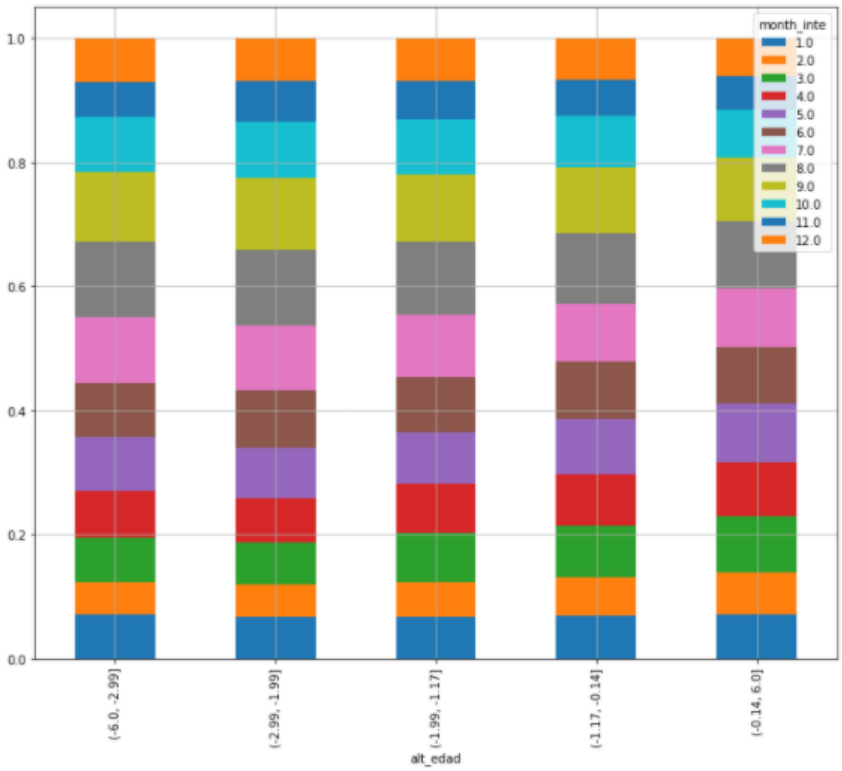
```
#Finalmente ploteamos
fig, ax= plt.subplots(figsize=(12,10))
tmp.drop('Total', axis=1).plot(kind='bar', stacked=True, legend=True, grid=True, ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1bf6734d220>
```



```
#ahora lo haremos de forma interactiva para cualquier variable

def barras_acum(var):
    intervalos = np.quantile(df['alt_edad'], q=np.linspace(0,1,6))
    tmp = df.groupby([df[var], pd.cut(df['alt_edad'], intervalos, duplicates='drop')]).size().unstack().T
    tmp['Total'] = tmp.sum(axis=1)
    for i in tmp.drop('Total', axis=1).columns:
        tmp[i] = tmp[i]/tmp['Total']

    fig, ax= plt.subplots(figsize=(12,10))
    return tmp.drop('Total', axis=1).plot(kind='bar', stacked=True, legend=True, grid=True, ax=ax)
interact(barras_acum, var=contenedor_categoricas)
```

var month_inte



Comprobación matemática para saber si hay o no relacion entre las variables categóricas y la variable objetivo

```
import scipy
import statsmodels.formula.api as smf
import statsmodels.api as sm
```

```
#Empezamos creando un dataframe con las variables categóricas
df_categoricas = df[contenedor_categoricas]
df_categoricas.head()
```

	country_co	month_inte	respond_mo	type_resid	mother_edu	drinking_w	toilety_fa	Mother_job	child_sex	child_age_	cooking_fu	welthind_	source_dri
0	BF4	10.0	5.0	rural	noedu	1.0	1.0	agriculture	False	3.0	1.0	1.0	Improved
1	BF4	10.0	5.0	rural	noedu	1.0	1.0	agriculture	False	1.0	1.0	2.0	Improved

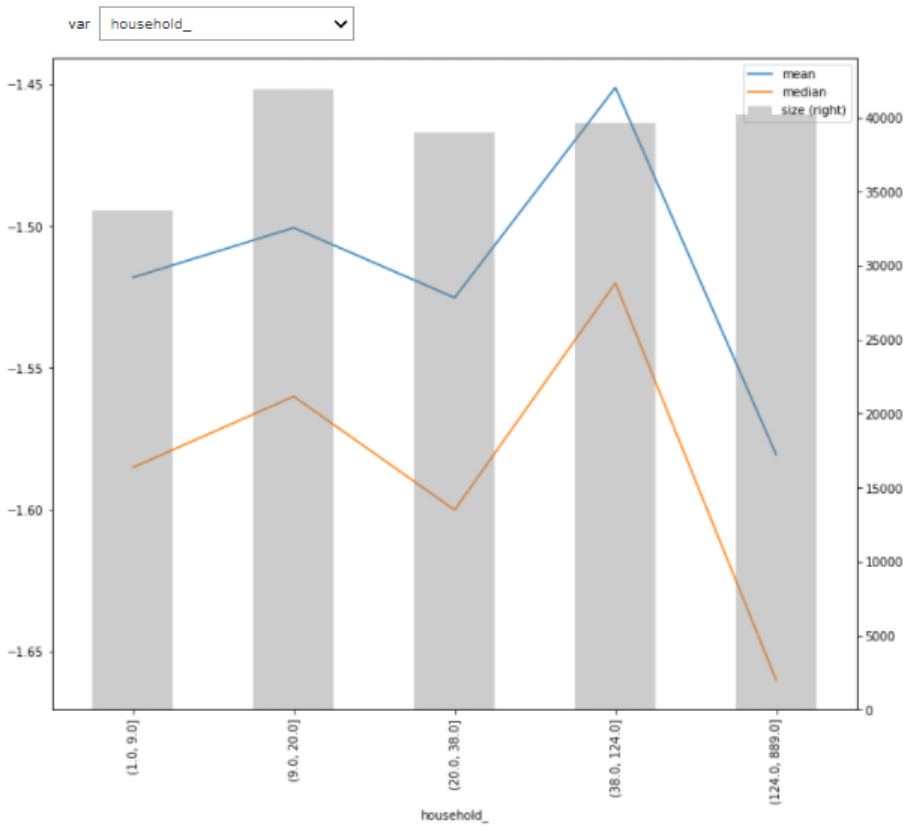
```
#definimos la variable objetivo
objetivo = df['alt_edad']
```

```
df_categoricas['alt_edad'] = objetivo
```

```python
contenedor_correlacionadas = []
contenedor_no_correlacionadas = []
for i in df_categoricas.columns:
    modelo = smf. ols ('alt_edad' + '~' + i, data = df_categoricas) .fit ()
    table = sm.stats.anova_lm (modelo)
    p = table ["PR(>F)"] [0]
    coeff, p = None, round (p, 3)
    conclusion = "esta correlacionada" if p <0.05 else "No esta correlacionada"
    print(f"La variable {i} presenta un p-valor de {p}, por lo tanto {conclusion} con la variable objetivo")
    if p < 0.05:
        contenedor_correlacionadas.append(i)
    else:
        contenedor_no_correlacionadas.append(i)
```

```
La variable country_co presenta un p-valor de 0.0, por lo tanto esta correlacionada con la variable objetivo
La variable month_inte presenta un p-valor de 0.0, por lo tanto esta correlacionada con la variable objetivo
La variable respond_mo presenta un p-valor de 0.001, por lo tanto esta correlacionada con la variable objetivo
La variable type_resid presenta un p-valor de 0.0, por lo tanto esta correlacionada con la variable objetivo
La variable mother_edu presenta un p-valor de 0.0, por lo tanto esta correlacionada con la variable objetivo
La variable drinking_w presenta un p-valor de 0.0, por lo tanto esta correlacionada con la variable objetivo
La variable toilety_fa presenta un p-valor de 0.0, por lo tanto esta correlacionada con la variable objetivo
La variable Mother_job presenta un p-valor de 0.0, por lo tanto esta correlacionada con la variable objetivo
La variable child_sex presenta un p-valor de 0.0, por lo tanto esta correlacionada con la variable objetivo
La variable child_age_ presenta un p-valor de 0.0, por lo tanto esta correlacionada con la variable objetivo
La variable cooking_fu presenta un p-valor de 0.0, por lo tanto esta correlacionada con la variable objetivo
La variable welthind_ presenta un p-valor de 0.0, por lo tanto esta correlacionada con la variable objetivo
La variable source_dri presenta un p-valor de 0.0, por lo tanto esta correlacionada con la variable objetivo
La variable type_toile presenta un p-valor de 0.0, por lo tanto esta correlacionada con la variable objetivo
La variable type_cooki presenta un p-valor de 0.0, por lo tanto esta correlacionada con la variable objetivo
La variable alt_edad presenta un p-valor de 0.0, por lo tanto esta correlacionada con la variable objetivo
```

```python
#Primero estudiaremos esta relacion de forma gráfica

def relacion_numericas(var):
    intervalos = np.quantile(df[var], q=np.linspace(0,1, 6))
    grupos = df.groupby([pd.cut(df[var], intervalos, duplicates='drop')])['alt_edad'].agg(['mean', 'median', 'size'])
    fig, ax = plt.subplots(figsize=(12,10))
    return grupos[['mean', 'median']].plot(kind='line', ax=ax), grupos[['size']].plot(kind='bar', ax=ax,
                                                                                      secondary_y=True,
                                                                                      alpha=0.4, color='grey')
interact(relacion_numericas, var=contenedor_numericas)
```

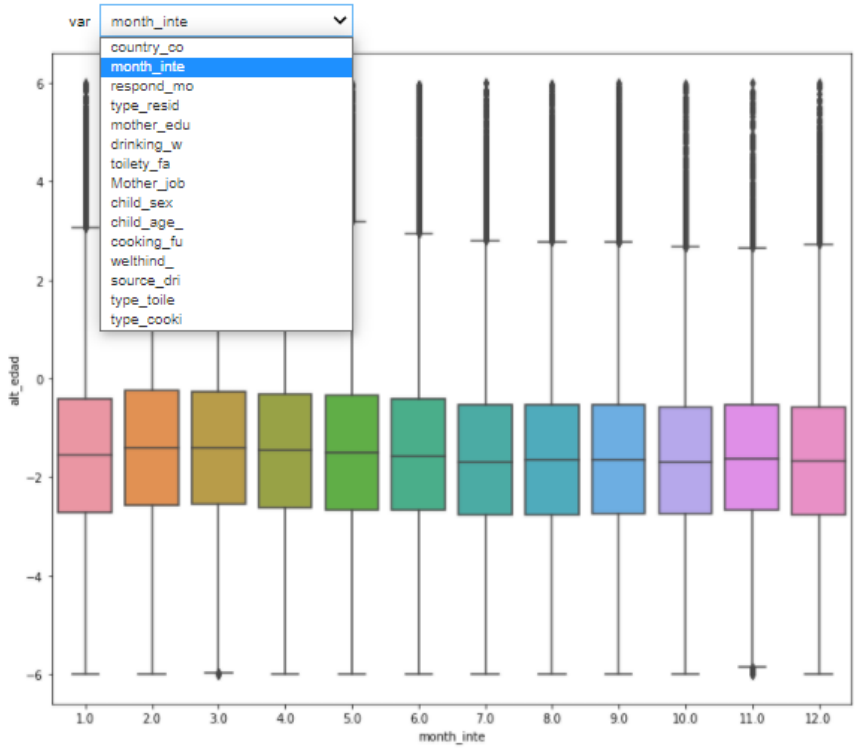


```python
#matemáticamente
for i in contenedor_numericas:
    coeff, p = scipy.stats.pearsonr(df[i], df['alt_edad'])
    coeff, p = round(coeff, 3), round(p, 3)
    conclusion = "Significant" if p < 0.05 else "Non-Significant"

    print(f"La variable {i} tiene un p-valor = {p}, por lo tanto es {conclusion}")
```

```
La variable cluster_nu tiene un p-valor = 0.0, por lo tanto es Significant
La variable household_ tiene un p-valor = 0.0, por lo tanto es Significant
La variable yr_intervi tiene un p-valor = 0.0, por lo tanto es Significant
La variable respond_yr tiene un p-valor = 0.0, por lo tanto es Significant
La variable respond_cu tiene un p-valor = 0.003, por lo tanto es Significant
La variable region tiene un p-valor = 0.001, por lo tanto es Significant
La variable v201 tiene un p-valor = 0.0, por lo tanto es Significant
La variable religion tiene un p-valor = 0.0, por lo tanto es Significant
La variable ethnicity tiene un p-valor = 0.0, por lo tanto es Significant
La variable offspring tiene un p-valor = 0.0, por lo tanto es Significant
La variable IMC tiene un p-valor = 0.0, por lo tanto es Significant
La variable month_chil tiene un p-valor = 0.0, por lo tanto es Significant
La variable Latitud tiene un p-valor = 0.0, por lo tanto es Significant
La variable Longitud tiene un p-valor = 0.0, por lo tanto es Significant
La variable alt_edad tiene un p-valor = 0.0, por lo tanto es Significant
La variable peso_edad tiene un p-valor = 0.0, por lo tanto es Significant
La variable peso_altur tiene un p-valor = 0.0, por lo tanto es Significant
La variable age_mother tiene un p-valor = 0.003, por lo tanto es Significant
La variable id_child tiene un p-valor = 0.004, por lo tanto es Significant
La variable month_ch_1 tiene un p-valor = 0.0, por lo tanto es Significant
La variable month_ch_2 tiene un p-valor = 0.0, por lo tanto es Significant
La variable month_ch_3 tiene un p-valor = 0.0, por lo tanto es Significant
La variable month_ch_4 tiene un p-valor = 0.0, por lo tanto es Significant
La variable month_ch_5 tiene un p-valor = 0.0, por lo tanto es Significant
La variable BMI_mother tiene un p-valor = 0.0, por lo tanto es Significant
La variable BMI_moth_1 tiene un p-valor = 0.0, por lo tanto es Significant
La variable Id_Child_1 tiene un p-valor = 0.0, por lo tanto es Significant
La variable field_1_2 tiene un p-valor = 0.0, por lo tanto es Significant
La variable stxv5 tiene un p-valor = 0.0, por lo tanto es Significant
La variable Longitud_2 tiene un p-valor = 0.0, por lo tanto es Significant
La variable Latitud_2 tiene un p-valor = 0.0, por lo tanto es Significant
```

Ingeniería de características

```python
#A continuación vamos a ver dos cosas simultaneamente, las etiquetas por variables y los outliers

def etiquetas_outiers(var):
    fig, ax = plt.subplots(figsize=(12,10))
    return sns.boxplot(df[var], df['alt_edad'])
interact (etiquetas_outiers, var=contenedor_categoricas)
```



Hay dos variables que contienen muchas etiquetas y esto no es eficiente a la hora de crear un modelo. Lo que haremos es concentrar todas las etiquetas en tres

```
#Empezamos creando un diccionario que refleje lo que hemos dicho

gr = {'respond_mo_low': [1.0,2.0,3.0,4.0],
      'respond_mo_med': [5.0,6.0,7.0,8.0],
      'respond_mo_high': [9.0,10.0,11.0,12.0]}
```

```
#A continuación le daremos la vuelta al diccionario
gr2 = {valor : clave for clave, v in gr.items() for valor in v}
```

```
gr2
```

```
{1.0: 'respond_mo_low',
 2.0: 'respond_mo_low',
 3.0: 'respond_mo_low',
 4.0: 'respond_mo_low',
 5.0: 'respond_mo_med',
 6.0: 'respond_mo_med',
 7.0: 'respond_mo_med',
 8.0: 'respond_mo_med',
 9.0: 'respond_mo_high',
 10.0: 'respond_mo_high',
 11.0: 'respond_mo_high',
 12.0: 'respond_mo_high'}
```

```
contenedor_agrupadas = ['month_inte_agg','respond_mo_agg']
```

```
df_agrupadas = df[['month_inte','respond_mo']]
```

```
df_agrupadas.head(2)
```

	month_inte	respond_mo
0	10.0	5.0
1	10.0	5.0

```
df_agrupadas.columns=contenedor_agrupadas
df_agrupadas.head(2)
```

	month_inte_agg	respond_mo_agg
0	10.0	5.0
1	10.0	5.0

```
for i in contenedor_agrupadas:
    df_agrupadas[i] = df_agrupadas[i].apply(lambda x: gr2[x] if x in gr2.keys() else 'faild')
```

```
df_agrupadas.head()
```

	month_inte_agg	respond_mo_agg
0	respond_mo_high	respond_mo_med
1	respond_mo_high	respond_mo_med
2	respond_mo_high	respond_mo_med
3	respond_mo_high	respond_mo_low
4	respond_mo_high	respond_mo_high

```
df_agrupadas.shape
```

```
(201508, 2)
```

```
#Ahora tendremos que construir el nuevo df
df_new = pd.concat([df, df_agrupadas], axis=1)
```

```
df_new = df_new.drop(['month_inte','respond_mo'], axis=1)
df_new.shape
```

```
(201508, 46)
```

Preprocesamiento

```
#Lo primero que vamos a hacer es crear las variables dummy de las variables categóricas
```

```
df_new.sample()
```

	country_co	cluster_nu	household_	yr_intervi	respond_yr	respond_cu	region	type_resid	mother_edu	drinking_w	v201	toilety_fa	religion	ethnic
154032	NG6	288.0	4.0	2013.0	1978.0	34.0	2.0	rural	noedu	0.0	15.0	0.0	3.0	10

```
nuevo_contenedor_categoricas = []
for i in df_new.columns:
    if df_new[i].dtype!='float64':
        nuevo_contenedor_categoricas.append(i)
```

```
nuevo_contenedor_categoricas
```

```
['country_co',
 'type_resid',
 'mother_edu',
 'Mother_job',
 'child_sex',
 'source_dri',
 'type_toile',
 'type_cooki',
 'field_1_2',
 'month_inte_agg',
 'respond_mo_agg']
```

```
#A continuacion, vamos a crear las dummy de las variables categoricas

dummy = pd.get_dummies(df_new, columns=nuevo_contenedor_categoricas, drop_first=True)
dummy
```

	cluster_nu	household_	yr_intervi	respond_yr	respond_cu	region	drinking_w	v201	toilety_fa	religion	ethnicity	offspring	IMC	child_age_	m
0	1.0	76.0	2003.0	1968.0	35.0	9.0	1.0	20.0	1.0	4.0	4.0	6.0	1981.0	3.0	
1	1.0	98.0	2003.0	1972.0	31.0	9.0	1.0	45.0	1.0	1.0	4.0	5.0	2017.0	1.0	
2	1.0	98.0	2003.0	1972.0	31.0	9.0	1.0	45.0	1.0	1.0	4.0	5.0	2017.0	4.0	
3	1.0	207.0	2003.0	1981.0	22.0	9.0	1.0	30.0	1.0	4.0	4.0	2.0	1834.0	3.0	
4	1.0	232.0	2003.0	1963.0	40.0	9.0	1.0	5.0	1.0	4.0	4.0	8.0	1792.0	2.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
201503	722.0	508.0	2013.0	1991.0	22.0	5.0	1.0	996.0	0.0	2.0	49.0	1.0	2394.0	0.0	
201504	722.0	572.0	2013.0	1990.0	23.0	5.0	1.0	996.0	0.0	2.0	64.0	1.0	1682.0	2.0	
201505	722.0	615.0	2013.0	1991.0	22.0	5.0	1.0	996.0	0.0	2.0	50.0	1.0	3121.0	1.0	
201506	722.0	719.0	2013.0	1978.0	35.0	5.0	1.0	996.0	0.0	2.0	1.0	1.0	1893.0	3.0	
201507	722.0	737.0	2013.0	1989.0	24.0	5.0	1.0	996.0	0.0	2.0	43.0	1.0	2476.0	1.0	

201508 rows × 1869 columns

```
#Separamos la variable objetivo
y = df_preprocessing['alt_edad']
```

```
#Seleccionamos las variables predictoras
x = df_preprocessing.drop(['alt_edad'], axis=1)
```

```
x.shape,     y.shape
```

```
((201508, 1868), (201508,))
```

```
#Para no tener problemas con el escalado de las variables debemos de dotar a la variable y de una dimension adicional
y = y[:, np.newaxis]
```

```
y.shape
```

```
(201508, 1)
```

Escalado de variables

En presencia de outliers es importante hacer un escalado de las variables. Concretamente un RobustScaler En primer lugar veamos de una forma rápida la presencia de outliers

```
from sklearn.preprocessing import RobustScaler
```

```
#Empezamos escalando las variables predictoras
x_escalado = RobustScaler(quantile_range=(25.0, 75.0))
x_rs = x_escalado.fit_transform(x)
```

```
y_escalado = RobustScaler(quantile_range=(25.0, 75.0))
y_rs = y_escalado.fit_transform(y)
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_rs, y_rs, test_size=0.2, random_state=2019)
```

Seleccion de características

```
#A continuación vamos a crear un dataframe con los valores escalados
X = pd.DataFrame(x_rs, columns=x.columns)
```

```
Y = pd.DataFrame(y_rs, columns=['alt_edad'])
```

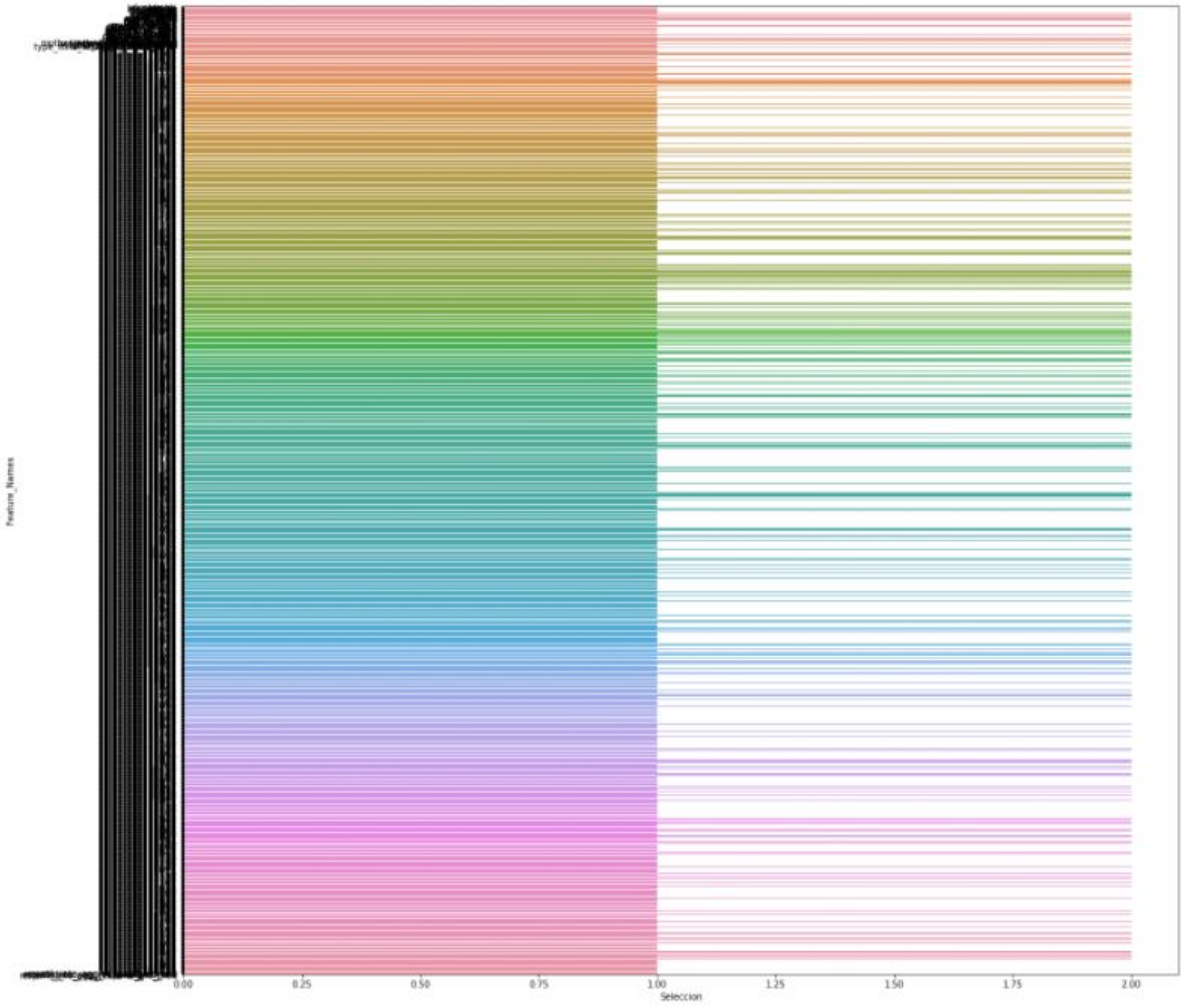
```
#Para la seleccion de variables utilizaremos dos métodos, por un lado el p-valor y por otro lado la regularización de Ridge
from sklearn import model_selection, preprocessing, feature_selection, ensemble, linear_model, metrics, decomposition
feature_names = X.columns


##p-valor
selector_variables_p = feature_selection.SelectKBest (score_func = feature_selection.f_regression, k = 1868).fit(X,Y)
pvalue_selected_features = feature_names[selector_variables_p.get_support ()]
```

```
## Ridge


selector_ridge = feature_selection.SelectFromModel (estimator = linear_model.Ridge (alpha = 100, fit_intercept = True),
                                                    max_features = 1868) .fit (X, Y)
regularization_selected_features = feature_names [selector_ridge.get_support ()]
```

```
#empezamos creando un dadtaframe con una sola columna donde figuren cada una de las var predictoras
df_features = pd.DataFrame(feature_names, columns=['Feature_Names'])
#Creamos una segunda columna que llamaremos 'p-valor'
#en ella escribiremos 'p-valor' si la variable se encuentra en la lista 'pvalue_selected_features'
df_features['p-valor'] = df_features['Feature_Names'].apply(lambda x: 'p-valor' if x in pvalue_selected_features else "")
df_features['num_1'] = df_features['Feature_Names'].apply(lambda x: 1 if x in pvalue_selected_features else 0)
#Hacemos lo mismo con las características seleccionadas por Ridge
df_features['Ridge'] = df_features['Feature_Names'].apply(lambda x: 'Ridge' if x in regularization_selected_features else ""
df_features['num_2'] = df_features['Feature_Names'].apply(lambda x: 1 if x in regularization_selected_features else 0)

df_features["Metodo"] = df_features[["p-valor","Ridge"]].apply(lambda x: (x[0]+" "+x[1]).strip(), axis=1)

df_features['Seleccion'] = df_features['num_1'] + df_features['num_2']

df_features['Seleccion'] = df_features['num_1'] + df_features['num_2']

df_features["metodo_2"] = df_features["Metodo"].apply(lambda x: "ambas" if len(x.split()) == 2 else x)
```

```
fig, ax = plt.subplots(figsize=(20,20))
sns.barplot(x=df_features['Seleccion'], y=df_features['Feature_Names'],data=df_features.sort_values(by='Seleccion', ascendin
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1bf83c5b9d0>
```



Método de selección de variables basado en el gradiente-descendente

```
modelo_seleccion = ensemble.GradientBoostingRegressor()
```

```
modelo_seleccion.fit(X,y_rs.ravel())
```

```
GradientBoostingRegressor()
```

```
variables_importantes = modelo_seleccion.feature_importances_
variables_importantes
```

```
array([0., 0., 0., ..., 0., 0., 0.])
```

```
#Para que se vea mejor crearemos un dataframe

importancia_df = pd.DataFrame({'VARIABLE':feature_names,
                               'IMPORTANCIA':variables_importantes})
importancia_df
```

```
importancia_df = importancia_df.sort_values(by='IMPORTANCIA', ascending=False)
```

```
importancia_df['Acumulado'] = importancia_df['IMPORTANCIA'].cumsum()
importancia_df = importancia_df.set_index('VARIABLE')
importancia_df.head(10)
```

```
importancia_df.head(10)
```

VARIABLE	IMPORTANCIA	Acumulado
peso_edad	0.599275	0.599275
peso_altur	0.380758	0.980033
month_ch_1	0.007766	0.987799
month_ch_2	0.006877	0.994676
month_chil	0.004539	0.999215
month_ch_4	0.000229	0.999444
child_age_	0.000214	0.999659
month_ch_3	0.000172	0.999831
month_ch_5	0.000169	1.000000
cluster_nu	0.000000	1.000000

Modelos

```
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.svm import SVR
```

```
df_simplificado = df_preprocessing[['peso_edad', 'peso_altur', 'month_ch_1', 'month_ch_2', 'month_chil','month_ch_4','child_
df_simplificado.head()
```

	peso_edad	peso_altur	month_ch_1	month_ch_2	month_chil	month_ch_4	child_age_	month_ch_3	alt_edad
0	-2.46	-1.27	1444.0	54872.0	38.0	0.0	3.0	0.0	-2.84
1	-1.94	1.12	361.0	6859.0	19.0	0.0	1.0	0.0	-5.22
2	-1.65	1.38	2500.0	125000.0	50.0	2500.0	4.0	50.0	-4.12
3	-1.91	0.35	1600.0	64000.0	40.0	1600.0	3.0	40.0	-3.50
4	-3.11	-1.43	841.0	24389.0	29.0	0.0	2.0	0.0	-3.83

```
x_simplificado = df_simplificado.drop('alt_edad', axis=1)
y_simplificado = df_simplificado['alt_edad']
```

```
x_simplificado.shape,   y_simplificado.shape
```

```
((201508, 8), (201508,))
```

```
#Antes de proceder al escalado de las cariables procedo a dotarle de una dimensión adicional a la variable 'y'
y_simplificado = y_simplificado[:, np.newaxis]
```

```
#ahora podemos escalar las variables

x_simplificado_escalado = RobustScaler(quantile_range=(25.0, 75.0))
x_simply_std = x_simplificado_escalado.fit_transform(x_simplificado)
```

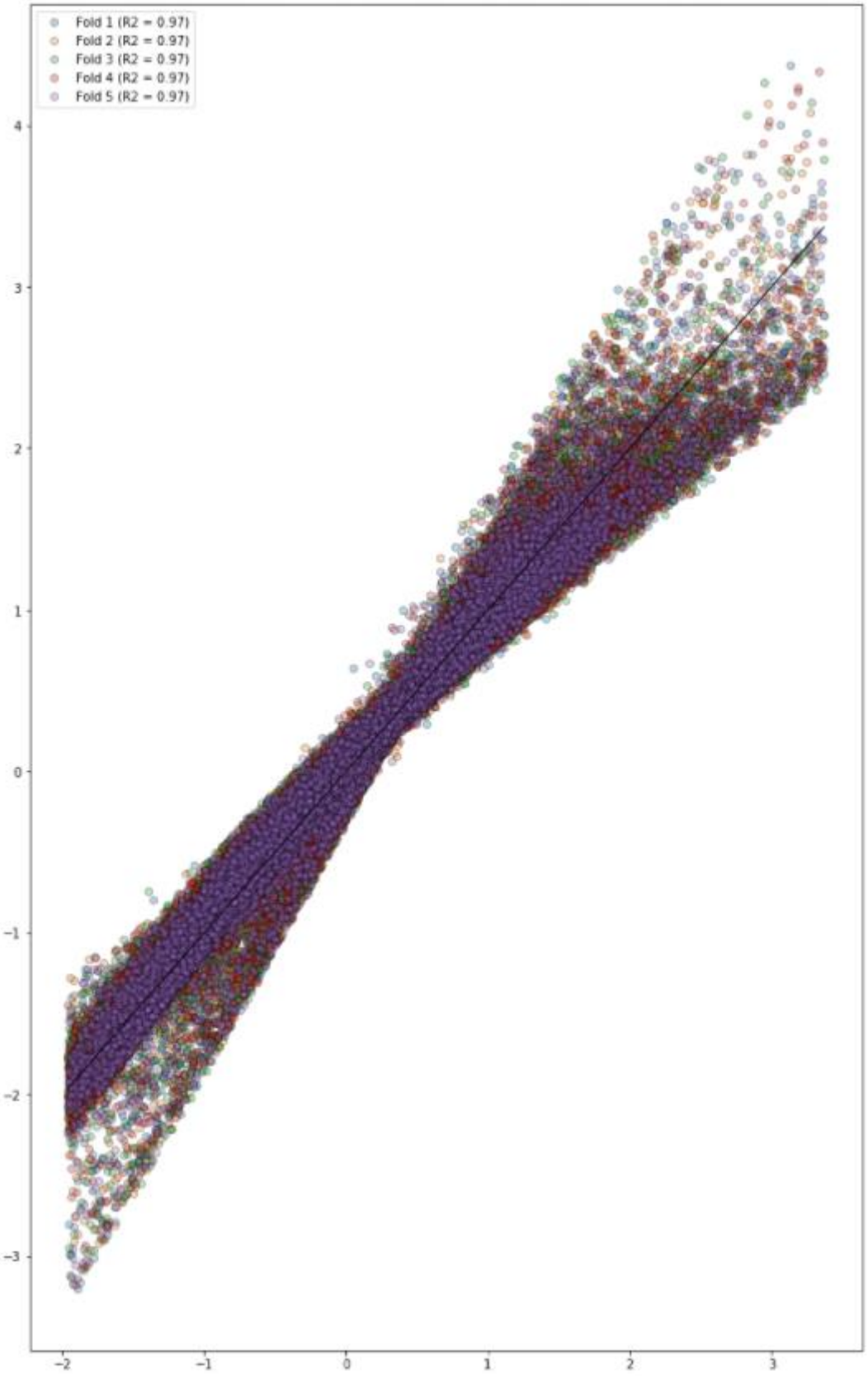
```
y_simplificado_escalado = RobustScaler(quantile_range=(25.0, 75.0))
y_simply_std = y_simplificado_escalado.fit_transform(y_simplificado)
```

```
modelo_lineal_simplificado = LinearRegression()
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_simply_std, y_simply_std, test_size=0.2, random_state=2019)
```

```
# LinearRegression
```

```
resultados_modelo_lineal = []
cv = KFold(n_splits=5)
i=1
fig = plt.figure(figsize=(12,20))
for train_index, test_index in cv.split(x_train, y_train):
    modelo_lineal_simplificado.fit(x_train[train_index], y_train[train_index])
    prediccion = modelo_lineal_simplificado.predict(x_train[test_index])

    metrica = r2_score(y_train[test_index],prediccion)
    resultados_modelo_lineal.append(metrica)
    plt.scatter(y_train[test_index], prediccion, alpha=0.3,edgecolor='black', label='Fold %d (R2 = %0.2f)' % (i,metrica))
    i+=1
plt.plot([min(y_train),max(y_train)], [min(y_train),max(y_train)], lw=1, color='black')
plt.legend()
```


Fold 1 (R2 = 0.97)
Fold 2 (R2 = 0.97)
Fold 3 (R2 = 0.97)
Fold 4 (R2 = 0.97)
Fold 5 (R2 = 0.97)
4
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

```
1  resultados_modelo_lineal
```

```
[0.9706648468077997,
 0.9713155470517392,
 0.9702479923429955,
 0.9713773681362928,
 0.9718370597731305]
```

```
1  np.mean(resultados_modelo_lineal)
```

```
0.9710885628223915
```

```
1  df_coeficientes
```

	Variables	Coeficientes
0	peso_edad	1.221732
1	peso_altur	-0.904638
2	month_ch_1	1.611853
3	month_ch_2	-0.637144
4	month_chil	-0.847137
5	month_ch_4	-0.034422
6	child_age_	-0.070728
7	month_ch_3	0.074729

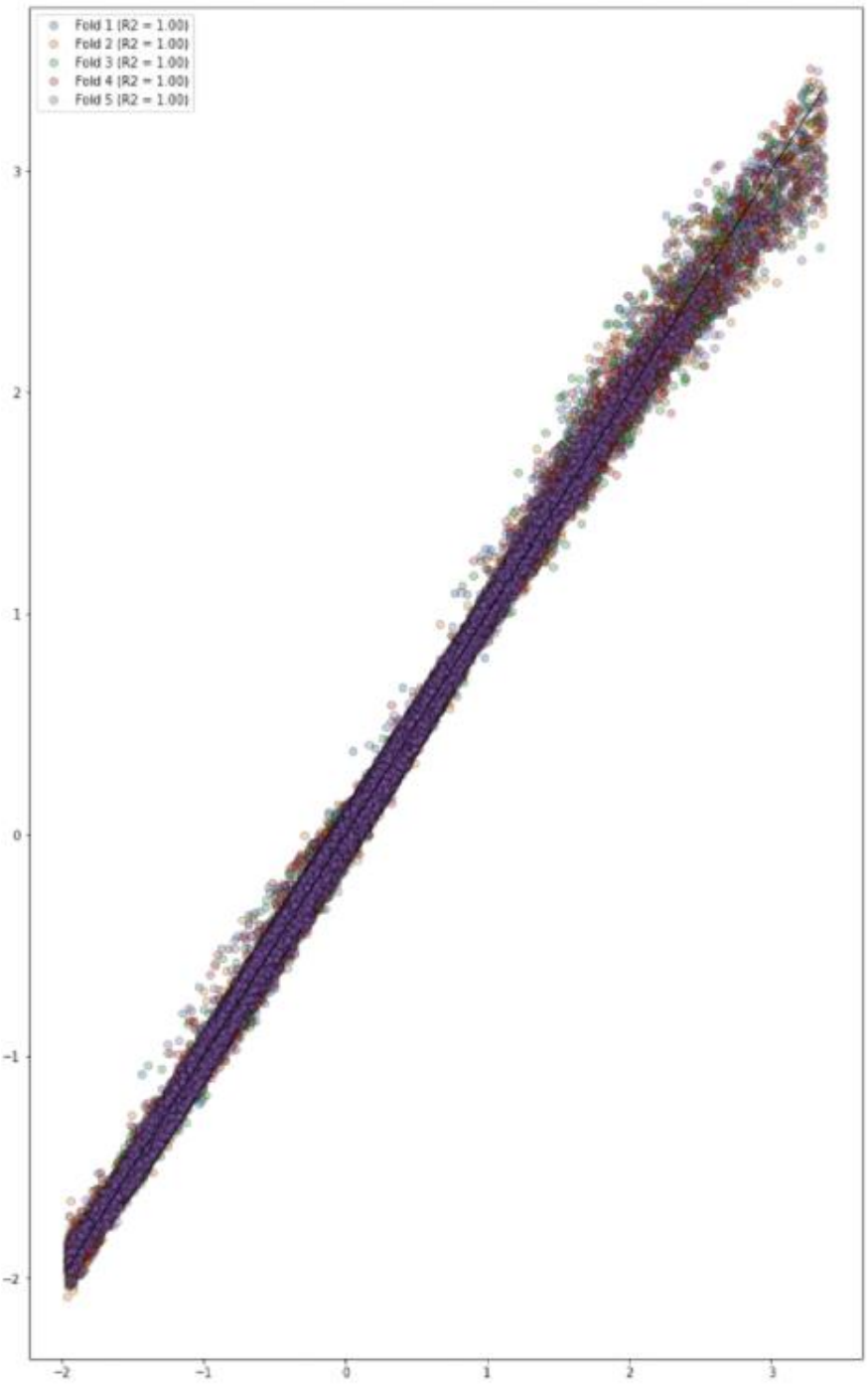
```
1  modelo_lineal_simplificado.intercept_
```

```
array([-0.06402287])
```

```
1  #SVR
```

```
1  best_svr = SVR(kernel='rbf')
```

```
1  resultados_svr = []
2  cv = KFold(n_splits=5)
3  i=1
4  fig = plt.figure(figsize=(12,20))
5  for train_index, test_index in cv.split(x_train, y_train):
6      best_svr.fit(x_train[train_index], y_train[train_index])
7      prediccion = best_svr.predict(x_train[test_index])
8
9      metrica = r2_score(y_train[test_index],prediccion)
10     resultados_svr.append(metrica)
11     plt.scatter(y_train[test_index], prediccion, alpha=0.3,edgecolor='black', label='Fold %d (R2 = %0.2f)' % (i,metrica))
12     i+=1
13 plt.plot([min(y_train),max(y_train)], [min(y_train),max(y_train)], lw=1, color='black')
14 plt.legend()
```




```
1 resultados_svr
```

```
[0.9953306831654054,
 0.9953065524672104,
 0.9952131105044226,
 0.995136669958769,
 0.9952406822876103]
```

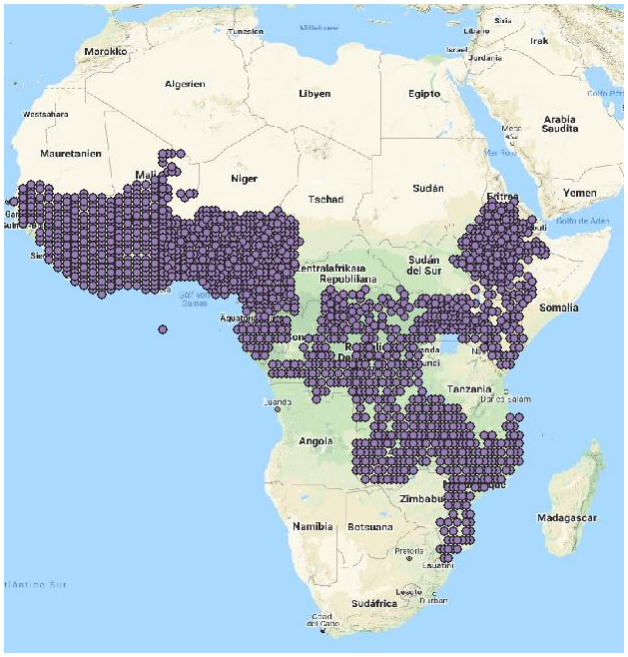
```
1 np.mean(resultados_svr)
```

```
0.9952455396766835
```

Finalmente lanzaremos otro modelo donde estudiaremos los niveles de salud infantil en las zonas donde disponemos de datos del NDVI, que conocemos como 'Crop Áreas' y el efecto que los shocks climáticos hayan podido tener. Para hacernos una idea espacial del estudio adjuntamos a continuación tres mapas donde se localizan las crop áreas, los niños sobre los que tenemos datos y los lugares donde coinciden niños con zonas de crop áreas.

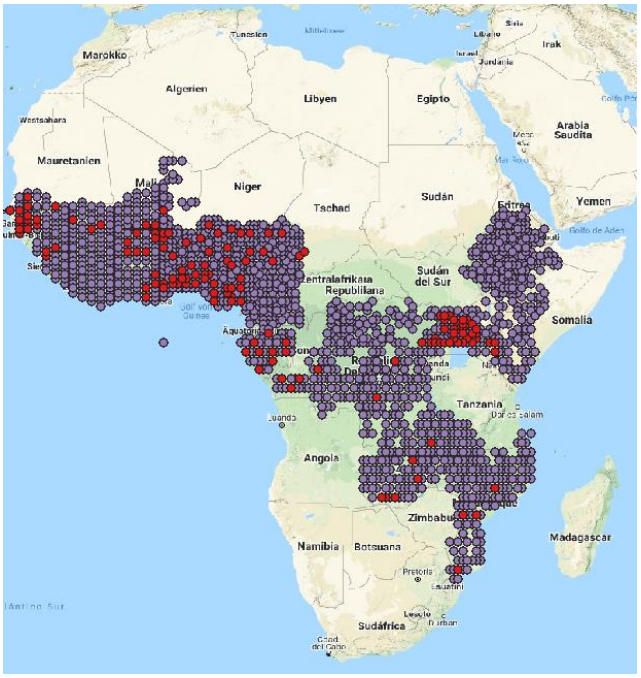
Figura1. Crop_areas

Figura 2. Localización de niños de la encuesta DHS



Finalmente, sobre los puntos de la figura 2 superponemos los que coinciden con crop áreas para construir la figura 3, que son los niños de la encuesta que están localizados en zonas de crop áreas.

Figura 3.



Con este modelo lo que queremos ver es en qué proporción los shocks climáticos son capaces de explicar los niveles de salud infantil. En este caso el cuadro de datos es bastante más reducido que el anterior, pasando de más de doscientos mil registros a unos setenta mil.

Cargamos los datos

```
1 pd.set_option('display.max_columns', 150)
```

```
1 df = pd.read_csv('union_shocks_dhs_qgis.csv').drop(['field_1'], axis=1)
```

```
1 df
```

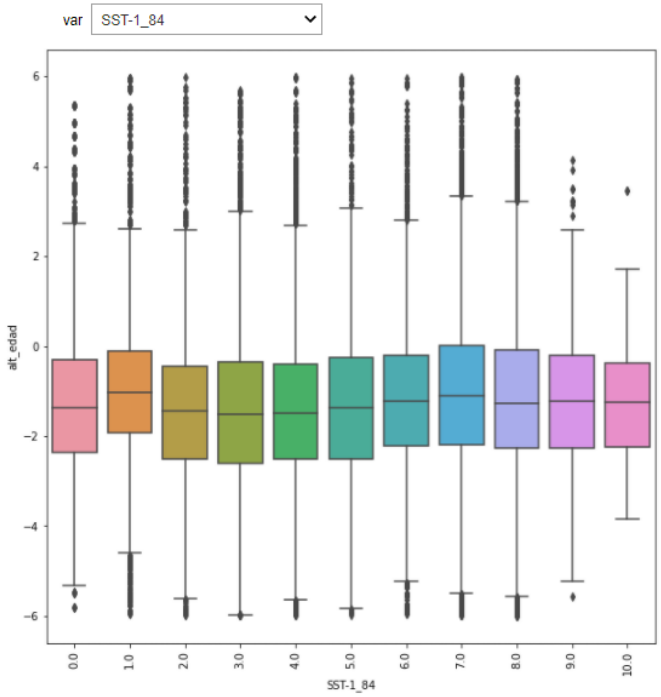
	Country	Province	Longitud	Latitud	Altura	NDVI_84	NDVI_95	NDVI_06	NDVI_17	C_NDVI_85_95	C_NDVI_95_06	C_NDVI_06_17	SPEI_84	SPEI
0	Benin	Alibori	2.75	11.25	267.500	0.245	0.324	0.320	0.339	0.079	-0.004	0.019	17.653620	11.477
1	Benin	Alibori	2.75	11.25	267.500	0.245	0.324	0.320	0.339	0.079	-0.004	0.019	17.653620	11.477
2	Benin	Alibori	2.75	11.25	267.500	0.245	0.324	0.320	0.339	0.079	-0.004	0.019	17.653620	11.477
3	Benin	Alibori	2.75	11.25	267.500	0.245	0.324	0.320	0.339	0.079	-0.004	0.019	17.653620	11.477
4	Benin	Alibori	2.75	11.25	267.500	0.245	0.324	0.320	0.339	0.079	-0.004	0.019	17.653620	11.477
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
71681	Zambia	Southern	25.75	-16.75	1056.875	0.585	0.621	0.627	0.614	0.036	0.006	-0.013	9.084344	11.477
71682	Zambia	Southern	25.75	-16.75	1056.875	0.585	0.621	0.627	0.614	0.036	0.006	-0.013	9.084344	11.477
71683	Zambia	Southern	25.75	-16.75	1056.875	0.585	0.621	0.627	0.614	0.036	0.006	-0.013	9.084344	11.477
71684	Zambia	Southern	25.75	-16.75	1056.875	0.585	0.621	0.627	0.614	0.036	0.006	-0.013	9.084344	11.477
71685	Zambia	Southern	25.75	-16.75	1056.875	0.585	0.621	0.627	0.614	0.036	0.006	-0.013	9.084344	11.477

71686 rows × 102 columns

El análisis exploratorio de datos que haceos es muy parecido al modelo anterior, de modo que pasamos a explicar el proceso de selección de variables que en este caso es un poco más complejo.

Todas aquellas variables numéricas que tienen menos de 15 valores únicos pasamos a categorizarlas, de este modo conseguimos que algunas variables como determinados shocks con más de 10 etiquetas pasen a tener sólo tres.

```
#A continuación vamos a ver el número de etiquetas por variables y la presencia de outliers

def outliers_por_etiquetas(var):
    fig, ax = plt.subplots(figsize=(10,10))
    sns.boxplot(df[var], df['alt_edad'], ax=ax)
    plt.xticks(rotation=90)
interact(outliers_por_etiquetas, var=lista_categoricas)
```



A continuación, explicamos el proceso de categorización.

```
#Empezamos creando un diccionario que refleje lo que hemos dicho

gr_shocks = {'shocks_bajos': [0.0,1.0,2.0,3.0],
     'shocks_medios': [4.0,5.0,6.0,7.0]}
```

```
gr_shocks_2 = {valor : clave for clave,v in gr_shocks.items() for valor in v}
```

```
contenedor_shocks = ['SSP+1_84', 'SSP-1_84', 'SST-1_84', 'SSTP-+1_84',
       'SSTP--1_84', 'SSP+1_95', 'SSP-1_95', 'SST+1_95', 'SST-1_95',
       'SSP+1_06', 'SSP-1_06', 'SST+1_06', 'SST-1_06', 'SSTP++1_06',
       'SSTP-+1_06', 'SSP+1_17', 'SSP-1_17', 'SST-1_17']
```

```
contenedor_shocks2 = []
for i in contenedor_shocks:
    new_name = (f"{i}_agg")
    contenedor_shocks2.append(new_name)
```

```
df_shocks_ag = df[contenedor_shocks]
```

```
df_shocks_ag.columns=contenedor_shocks2
df_shocks_ag.head(2)
```

```
for i in contenedor_shocks2:
    df_shocks_ag[i] = df_shocks_ag[i].apply(lambda x: gr_shocks_2[x] if x in gr_shocks_2.keys() else 'shocks_altos')
```

```
1 df_shocks_ag.head()
```

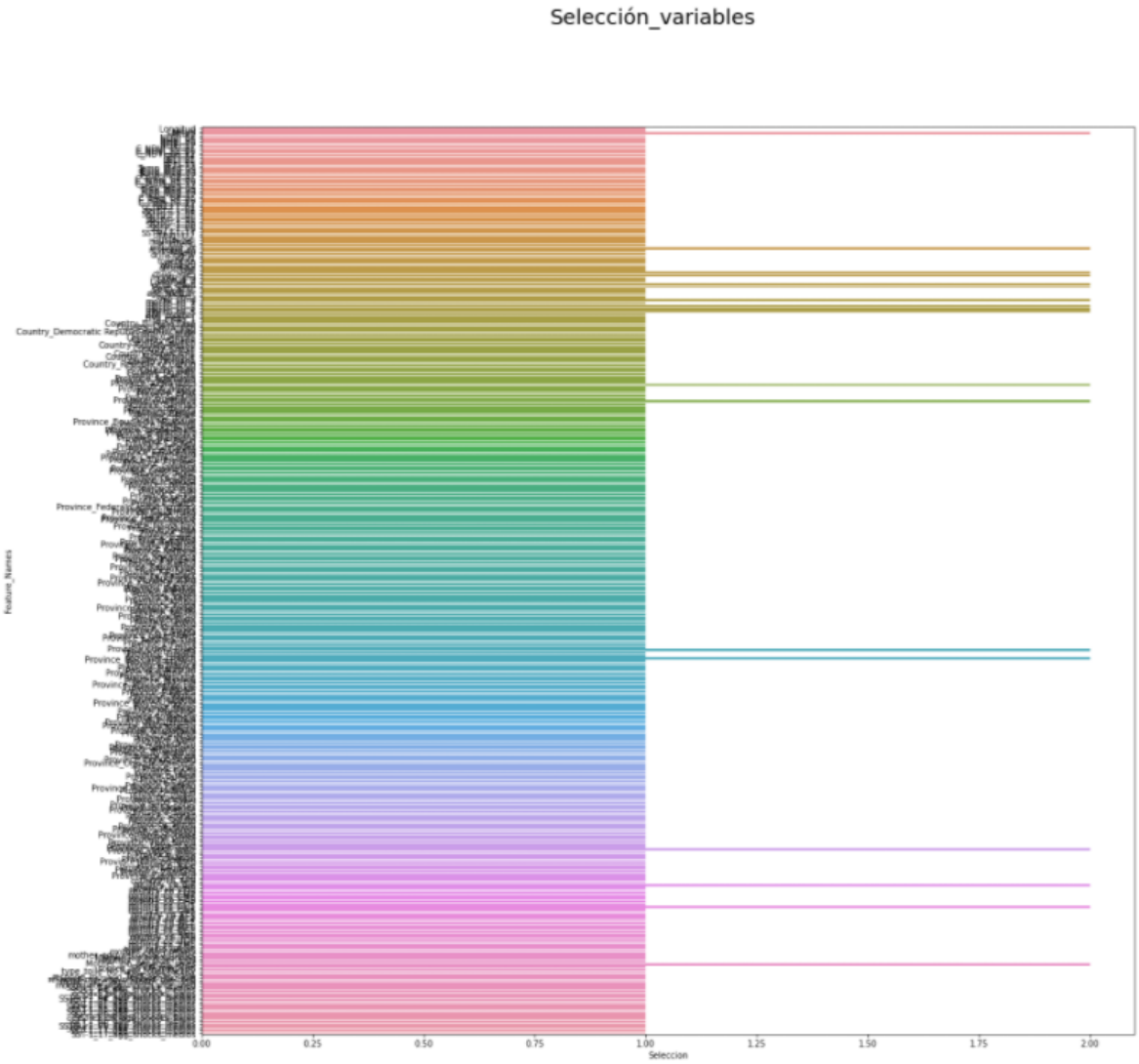
	SSP+1_84_agg	SSP-1_84_agg	SST-1_84_agg	SSTP-+1_84_agg	SSTP--1_84_agg	SSP+1_95_agg	SSP-1_95_agg	SST+1_95_agg	SST-1_95_agg	SSP+1_06_agg	S 1_06_
0	shocks_bajos	shocks_bajos	shocks_medios	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_b
1	shocks_bajos	shocks_bajos	shocks_medios	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_b
2	shocks_bajos	shocks_bajos	shocks_medios	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_b
3	shocks_bajos	shocks_bajos	shocks_medios	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_b
4	shocks_bajos	shocks_bajos	shocks_medios	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_bajos	shocks_b

Repetimos el proceso con todas aquellas variables numéricas con un número de valores únicos pequeño (en nuestro caso hemos considerado menos de quince) y procedemos a la selección de variables. El desarrollo del código es igual que en los modelos anteriores, de modo que pasamos directamente a la exposición de los resultados.

Siguiendo el criterio del gradiente descendente tenemos que:

VARIABLE	IMPORTANCIA	Acumulado
peso_edad	0.608329	0.608329
peso_altur	0.374220	0.982549
month_chil	0.008916	0.991465
month_ch_2	0.004296	0.995760
month_ch_1	0.003897	0.999658
month_ch_5	0.000273	0.999931
month_ch_4	0.000062	0.999993
type_toile_No have toilet facility	0.000006	0.999998
offspring	0.000002	1.000000
Province_Ngounié	0.000000	1.000000

Basándonos en el criterio de los p-valores y la regularización de Ridge (con un alpha=100):



Sólo las barras horizontales que tienen valor igual a dos son las variables que cumplen las dos condiciones. Para que se entiendan mejor que cuáles se trata:

	Feature_Names
2	Altura
44	respond_cu
53	child_age_
54	month_chil
57	peso_edad
58	peso_altur
63	month_ch_1
65	month_ch_3
66	month_ch_4
67	month_ch_5
94	Province_Alibori
100	Province_Banjul
191	Province_Luapula
194	Province_Maccarthy Island
264	Province_Upper West
277	country_co_BJ6
285	country_co_GN4
306	child_sex_True

Se observa que en ambos criterios no se selecciona ninguna variable climática para explicar los niveles de salud infantil.

Anexo 7 – Modelo de Clusterización

En este apéndice lanzamos un modelo de Clusterización con los datos de la tabla DHS con la intención de saber si existe algún tipo de clasificación en bruto a la que podamos sacar algún tipo de lectura. Para ser mas exactos hemos lanzado dos modelos, el primero de ellos con toda la tabla DHS sin ningún tipo de cruce con datos climáticos, shocks, etc. y un segundo modelo con menos registros donde si se cruzan los datos de la tabla DHS con las 'Crop_Áreas'.

A continuación, se presentan el desarrollo de los códigos para ambos casos, así como los resultados implementados con QGIS.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
df = pd.read_csv('DHS_procesada_final2.csv').drop('Unnamed: 0', axis=1)
```

```
df
```

	country_code	cluster_number	household_number	month_interview	yr_interview	respond_month_birth	respond_yr_birth	respond_current_age	region
0	BF4	1	76.0	10	2003	5	1968	35	9.0
1	BF4	1	98.0	10	2003	5	1972	31	9.0
2	BF4	1	98.0	10	2003	5	1972	31	9.0
3	BF4	1	207.0	10	2003	3	1981	22	9.0
4	BF4	1	232.0	10	2003	10	1963	40	9.0
...	...	...	...	...	...	...	...	...	...
201503	ZM6	722	508.0	12	2013	6	1991	22	5.0
201504	ZM6	722	572.0	12	2013	3	1990	23	5.0
201505	ZM6	722	615.0	12	2013	8	1991	22	5.0
201506	ZM6	722	719.0	12	2013	8	1978	35	5.0
201507	ZM6	722	737.0	12	2013	11	1989	24	5.0

201508 rows × 42 columns

```
contenedor_categoricas = []
for i in variables:
    if (df[i].dtype == 'object'):
        contenedor_categoricas.append(i)
```

```
contenedor_categoricas
```

```
['country_code',
 'type_residence',
 'mother_edu',
 'Mother_job',
 'source_drinking_water',
 'type_toilet_facility',
 'type_cooking_fuel']
```

```
df = df.drop(['country_code', 'Id_Child', 'id_child'], axis=1)
```

```
nuevo_contenedor_categóricas = []
for j in df.columns.to_list():
    if (df[j].dtype == 'object'):
        nuevo_contenedor_categóricas.append(j)
```

```
#Ahora queremos ver cuántos valores únicos hay por variable
for k in  nuevo_contenedor_categóricas:
    n_etiquetas = df[k].nunique()
    print(f"La variable {k} tiene {n_etiquetas}")
```

```
La variable type_residence tiene 2
La variable mother_edu tiene 3
La variable Mother_job tiene 4
La variable source_drinking_water tiene 2
La variable type_toilet_facility tiene 2
La variable type_cooking_fuel tiene 2
```

```
for e in  nuevo_contenedor_categóricas:
    etiquetas = df[e].unique()
    print(f"La variable {e} tiene las etiquetas:  {etiquetas}")
```

```
La variable type_residence tiene las etiquetas:  ['rural' 'urban']
La variable mother_edu tiene las etiquetas:  ['noedu' 'secondary_higher' 'primary']
La variable Mother_job tiene las etiquetas:  ['agriculture' 'not_working' 'services_sales' 'other_jobs']
La variable source_drinking_water tiene las etiquetas:  ['Improved' 'Unimproved']
La variable type_toilet_facility tiene las etiquetas:  ['Have toilet facility' 'No have toilet facility']
La variable type_cooking_fuel tiene las etiquetas:  ['Solid' 'Non-solid']
```

```
#El siguiente paso será codificar numericamente estas variables
#Esto forma parte del preprocesamiento
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
contenedor_transformadas = []
for i in nuevo_contenedor_categóricas:
    i_le = le.fit_transform(df[i])
    contenedor_transformadas.append(i_le)
```

```
contenedor_transformadas
```

```
[array([0, 0, 0, ..., 0, 0, 0]),
 array([0, 0, 0, ..., 2, 2, 2]),
 array([0, 0, 0, ..., 1, 2, 2]),
 array([0, 0, 0, ..., 0, 0, 0]),
 array([0, 0, 0, ..., 1, 1, 1]),
 array([1, 1, 1, ..., 0, 0, 0])]
```

```
df_transformadas = pd.DataFrame(contenedor_transformadas)
df_transformadas
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	...	201478	201479	201480	201481	2014
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	...	1	1	1	1	
1	0	0	0	0	0	0	0	0	0	0	0	0	2	0	2	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	...	2	2	2	2	
2	0	0	0	0	0	0	0	0	0	0	0	0	1	1	3	0	1	0	3	3	3	3	2	2	3	0	0	0	0	0	...	1	3	3	1	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	...	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	...	1	1	1	1	
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	...	1	0	0	0	

6 rows × 201508 columns

```
df_transformadas_transpuesta = df_transformadas.T
df_transformadas_transpuesta
```

	0	1	2	3	4	5
0	0	0	0	0	0	1
1	0	0	0	0	0	1
2	0	0	0	0	0	1
3	0	0	0	0	0	1

```
nuevo_contenedor_categóricas
```

```
['type_residence',
 'mother_edu',
 'Mother_job',
 'source_drinking_water',
 'type_toilet_facility',
 'type_cooking_fuel']
```

```
df_transformadas_transpuesta.columns=[nuevo_contenedor_categóricas]
df_transformadas_transpuesta
```

	type_residence	mother_edu	Mother_job	source_drinking_water	type_toilet_facility	type_cooking_fuel
0	0	0	0	0	0	1
1	0	0	0	0	0	1
			0	0	0	1

click to scroll output; double click to hide

```
new_df = pd.concat([df, df_transformadas_transpuesta], axis=1)
new_df
```

	cluster_number	household_number	month_interview	yr_interview	respond_month_birth	respond_yr_birth	respond_current_age	region	type_residenc
0	1	76.0	10	2003	5	1968	35	9.0	rura
1	1	98.0	10	2003	5	1972	31	9.0	rura
2	1	98.0	10	2003	5	1972	31	9.0	rura
3	1	207.0	10	2003	3	1981	22	9.0	rura
4	1	232.0	10	2003	10	1963	40	9.0	rura

```
new_df = new_df.drop(['type_residence',
 'mother_edu',
 'Mother_job',
 'source_drinking_water',
 'type_toilet_facility',
 'type_cooking_fuel'], axis=1)
```

```
new_df.shape
```

```
(201508, 39)
```

```
#Es conveniente que comencemos haciendo un escalado de las variables
from sklearn.preprocessing import StandardScaler
```

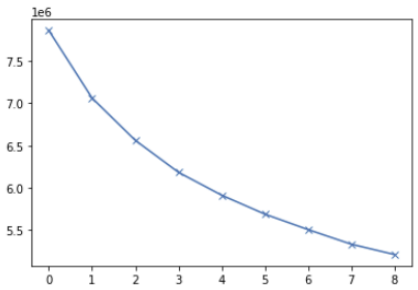
```
sc = StandardScaler()
```

```
x_std = sc.fit_transform(new_df)
```

```
#En principio vamos a hacer un problema de clusterizacion sin considerar ninguna variable objetivo
from sklearn.cluster import KMeans
```

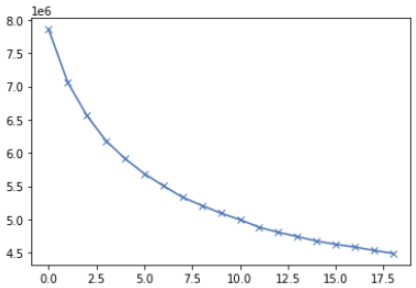
```
#antes de lanzar el modelo vamos a ver cuál es el número optimo de vencindarios que podemos crear
contenedor = []
for i in range(1,10):
    k_vecinos_aux = KMeans(n_clusters=i, random_state=0)
    k_vecinos_aux.fit(x_std)
    contenedor.append(k_vecinos_aux.inertia_)

plt.plot(contenedor, 'x-')
```

```
[<matplotlib.lines.Line2D at 0x2f74c199520>]
```




```
contenedor = []
for i in range(1,20):
    k_vecinos_aux = KMeans(n_clusters=i, random_state=0)
    k_vecinos_aux.fit(x_std)
    contenedor.append(k_vecinos_aux.inertia_)

plt.plot(contenedor, 'x-')
```

```
[<matplotlib.lines.Line2D at 0x2f74c23d490>]
```




```
k_means = KMeans(n_clusters=10)
```

```
k_means.fit(x_std)
```

```
KMeans(n_clusters=10)
```

```
#preguntamos por los centroides
k_means.cluster_centers_
```

```
array([[-1.67054426e-01, -7.97500415e-02,  2.25630028e-02,
         2.62586589e-02,  3.30951667e-02,  5.02421657e-01,
        -5.74595341e-01, -7.06851083e-02,  9.34532249e-01,
         2.95285015e-02, -6.69516560e-01, -1.97150512e-02,
        -8.59206912e-02, -5.16223320e-01, -1.48482181e-01,
        -3.89774820e-02, -5.85328037e-01, -5.88200269e-01,
        -1.86661497e-01,  1.08380641e-01,  1.36913785e-01,
         1.53435998e-01,  3.21536998e-02,  2.72238055e-01,
        -5.73894803e-01,  4.98155595e-01, -6.12993122e-01,
        -6.00543590e-01, -3.08112851e-01, -3.77055154e-01,
        -3.89245307e-01, -1.48482181e-01, -1.65703971e-01,
         3.87956365e-01,  3.04796994e-01,  1.22483434e-01,
        -9.34532249e-01,  6.69516560e-01,  2.72238055e-01],
       [-2.13277789e-01,  1.09360593e-01,  7.30599851e-02,
        -2.23090750e-01, -7.36791649e-03, -1.27055076e+00,
         1.35158255e+00, -8.82542774e-02, -7.63504093e-02,
        -1.55039077e-01,  2.34678831e-01, -6.86144985e-02,
        -8.41486815e-02,  1.25959582e+00, -2.23921092e-01,
        -3.51884439e-02, -3.79865324e-01, -3.88703062e-01,
         1.93975932e-01, -5.36998055e-02, -6.87589431e-02,
```

Tenemos 10 referencias pero es imposible representarlas en un espacio multidimensional tan grande

```
#Lo que si nos interresa es que a cada registro se le asocie una etiqueta
etiquetas = k_means.labels_
```

```
etiquetas
```

```
array([1, 6, 4, ..., 5, 5, 5])
```

```
np.shape(etiquetas)
```

```
(201508,)
```

```
new_df['TAG']=etiquetas
```

```
new_df['TAG']=etiquetas
new_df.head(3)
```

	cluster_number	household_number	month_interview	yr_interview	respond_month_birth	respond_yr_birth	respond_current_age	region	drinking_water	v2(
0	1	76.0	10	2003	5	1968	35	9.0	1	20
1	1	98.0	10	2003	5	1972	31	9.0	1	45
2	1	98.0	10	2003	5	1972	31	9.0	1	45

```
#comprobamos que el número de etiquetas coincida con el número de clusters que decidimos crear
new_df['TAG'].nunique()
```

```
10
```

```
new_df.to_csv('clusterizacion_salud_nuevo_DHS_procesada.csv')
```

```
#Queremos saber cuántos datos hay registrados en cada cluster
puntos_por_cluster = new_df[['Longitud', 'Latitud']]
```

```
prueba = []
for index, row in puntos_por_cluster.iterrows():
    var1 = row['Longitud']
    var2 = row['Latitud']

    union = f"{var1}_{var2}"
    prueba.append(union)
```

```
puntos_por_cluster['merge'] = prueba
```

```
<ipython-input-42-1d7f3c9820de>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  puntos_por_cluster['merge'] = prueba
```

```
puntos_por_cluster['merge'].nunique()
```

```
1788
```

```
child_by_cluster = puntos_por_cluster['merge'].value_counts()
```

```
dff = pd.DataFrame(child_by_cluster)
```

	#childs
35.25_-14.75	3068
35.25_-15.75	1582
-6.75_12.75	1568
33.75_-12.75	1567

```
dff['#childs'].sum()
```

```
201508
```

```
clusters_df = pd.DataFrame(k_means.cluster_centers_)
```

```
clusters_df
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	-0.167054	-0.079750	0.022563	0.026259	0.033095	0.502422	-0.574595	-0.070685	0.934532	0.029529	-0.669517	-0.019715	-0.085921	-0.516223	-0.148
1	-0.213278	0.109361	0.073060	-0.223091	-0.007368	-1.270551	1.351583	-0.088254	-0.076350	-0.155039	0.234679	-0.068614	-0.084149	1.259596	-0.223
2	4.688394	-0.499852	0.630017	-1.865406	-0.066102	-0.872577	-0.148478	-0.384201	-0.149952	-0.385214	-0.317834	-0.070418	-0.427233	-0.174454	-0.099
3	-0.202546	-0.016945	-0.028451	0.314356	-0.032530	0.660765	-0.571699	-0.155421	-1.069909	0.130646	-0.696998	-0.093486	-0.035356	-0.413545	-0.243
4	-0.149611	0.040841	0.010421	0.027116	0.000707	-0.167125	0.214020	-0.083298	-0.056063	-0.066645	0.055313	0.027073	-0.024901	0.221034	-0.113
5	0.139247	-0.118473	-0.338953	0.444993	0.105057	0.211696	0.019811	-0.028415	0.650269	0.693793	-0.548711	-0.131653	0.595875	-0.386041	0.700
6	-0.195178	0.080339	-0.004270	-0.039101	-0.042884	0.440635	-0.541232	-0.039978	-0.224318	-0.244661	1.434724	0.155182	0.055155	-0.427274	-0.341
7	-0.447318	0.909436	-0.224343	-0.310718	0.071110	-0.165450	-0.015990	8.361573	-0.283768	-0.287285	-0.383247	-0.148650	-0.399993	0.106674	-0.131
8	-0.183348	0.049283	0.024063	0.031667	-0.011227	-0.171810	0.223462	-0.092462	-0.063857	-0.075185	0.069356	0.020872	-0.024114	0.227068	-0.171
9	-0.118503	-0.153869	-0.154600	0.303867	0.044192	-0.258451	0.491251	-0.046558	0.350463	0.436997	-0.436270	0.108915	0.092398	0.289958	2.380

```
variab = new_df.columns.to_list()
```

```
variab.pop()
```

```
'TAG'
```

```
clusters_df.columns=[variab]
```

```
clusters_df
```

	cluster_number	household_number	month_interview	yr_interview	respond_month_birth	respond_yr_birth	respond_current_age	region	drinking_water
0	-0.167054	-0.079750	0.022563	0.026259	0.033095	0.502422	-0.574595	-0.070685	0.934532
1	-0.213278	0.109361	0.073060	-0.223091	-0.007368	-1.270551	1.351583	-0.088254	-0.076350
2	4.688394	-0.499852	0.630017	-1.865406	-0.066102	-0.872577	-0.148478	-0.384201	-0.149952
3	-0.202546	-0.016945	-0.028451	0.314356	-0.032530	0.660765	-0.571699	-0.155421	-1.069909
4	-0.149611	0.040841	0.010421	0.027116	0.000707	-0.167125	0.214020	-0.083298	-0.056063
5	0.139247	-0.118473	-0.338953	0.444993	0.105057	0.211696	0.019811	-0.028415	0.650269
6	-0.195178	0.080339	-0.004270	-0.039101	-0.042884	0.440635	-0.541232	-0.039978	-0.224318
7	-0.447318	0.909436	-0.224343	-0.310718	0.071110	-0.165450	-0.015990	8.361573	-0.283768
8	-0.183348	0.049283	0.024063	0.031667	-0.011227	-0.171810	0.223462	-0.092462	-0.063857
9	-0.118503	-0.153869	-0.154600	0.303867	0.044192	-0.258451	0.491251	-0.046558	0.350463

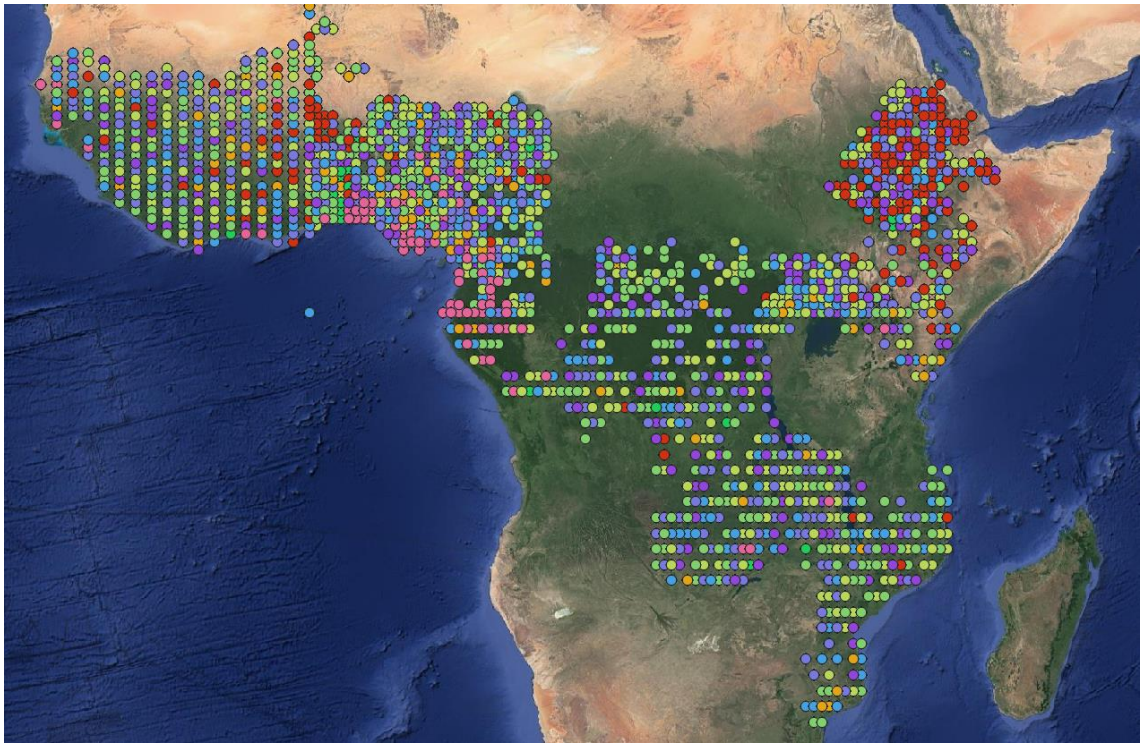
```
sc.inverse_transform(clusters_df)
clusters_df2 = pd.DataFrame(sc.inverse_transform(clusters_df))
```

```
clusters_df2.columns=[variab]
clusters_df2
```

	cluster_number	household_number	month_interview	yr_interview	respond_month_birth	respond_yr_birth	respond_current_age	region	drinking_water
0	290.183257	85.421011	6.717929	2008.517241	6.017809	1983.088728	25.044222	4.809777	9.999370e-01
1	267.617129	117.017215	6.879143	2007.394543	5.881095	1968.574657	38.461256	4.682516	4.956474e-01
2	2660.600508	15.231183	8.657258	2000.000000	5.682646	1971.832587	28.012395	2.538829	4.589307e-01
3	272.856344	95.914354	6.555063	2009.814403	5.796080	1984.384974	25.064394	4.195997	-1.032507e-14
4	298.699177	105.569077	6.679164	2008.521103	5.908378	1977.607625	30.537423	4.718416	5.057681e-01
5	439.719005	78.951299	5.563771	2010.402598	6.260950	1980.708760	29.184633	5.115961	8.581292e-01
6	276.453532	112.168395	6.632263	2008.222960	5.761096	1982.582929	25.276617	5.032200	4.218324e-01
7	153.359106	250.692129	5.929669	2007.000000	6.146251	1977.621332	28.935259	65.888682	3.921751e-01
8	282.228992	106.979551	6.722717	2008.541594	5.868055	1977.569268	30.603194	4.652034	5.018801e-01
9	313.886090	73.037345	6.152327	2009.767176	6.055302	1976.860002	32.468507	4.984544	7.085679e-01

```
clusters_df2.round(2)
```

	cluster_number	household_number	month_interview	yr_interview	respond_month_birth	respond_yr_birth	respond_current_age	region	drinking_water	v2(
0	290.18	85.42	6.72	2008.52	6.02	1983.09	25.04	4.81	1.00	20
1	267.62	117.02	6.88	2007.39	5.88	1968.57	38.46	4.68	0.50	14
2	2660.60	15.23	8.66	2000.00	5.68	1971.83	28.01	2.54	0.46	5



A continuación, mostraremos los resultaos para el caso donde cruzamos los datos DHS con datos climáticos asociados a las zonas de cultivo. Los resultados a los que llegamos son más explícitos que los anteriores, pero hay que tener en cuenta que en este caso los registros bajan de doscientos mil a setenta mil y por teoría de la ciencia de datos, los modelos de Clusterización pierden eficiencia cuanto mayor es el número de datos.

Es conveniente aclarar que tanto en este mapa como el que pondremos posteriormente no aparecen representados ni los 200000 para el primer caso ni 70000 para el segundo, el motivo es porque hay clústeres donde se registra más de un niño. Eso queda explicado en el código.

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 %matplotlib inline
```

```
1 #Empezamos descargando los datos
2 df = pd.read_csv('union_shocks_dhs_qgis.csv').drop(['field_1'], axis=1)
```

```
C:\Users\Usuario\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3071: DtypeWarning: Columns (1,2,80) have mixed t
ypes.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

```
1 pd.set_option('display.max_columns', 150)
```

```
1 df
```

	Country	Province	Longitud	Latitud	Altura	NDVI_84	NDVI_95	NDVI_06	NDVI_17	C_NDVI_85_95	C_NDVI_95_06	C_NDVI_06_17	SPEI_84	SPEI_95
0	Benin	Alibori	2.75	11.25	267.5	0.245	0.324	0.32	0.339	0.079	-0.004	0.019	17.65362	11.47730
1	Benin	Alibori	2.75	11.25	267.5	0.245	0.324	0.32	0.339	0.079	-0.004	0.019	17.65362	11.47730

```
df = df.dropna()
```

```
df = df.drop_duplicates()
```

```
df
```

	Country	Province	Longitud	Latitud	Altura	NDVI_84	NDVI_95	NDVI_06	NDVI_17	C_NDVI_85_95	C_NDVI_95_06	C_NDVI_06_17	SPEI_84	SPEI_
0	Benin	Alibori	2.75	11.25	267.500	0.245	0.324	0.320	0.339	0.079	-0.004	0.019	17.653620	11.477
1	Benin	Alibori	2.75	11.25	267.500	0.245	0.324	0.320	0.339	0.079	-0.004	0.019	17.653620	11.477

```
df.info()
```

```
variables = df.columns.to_list()
```

```
variables
```

```
contenedor_categoricas = []
for i in variables:
    if (df[i].dtype == 'object'):
        contenedor_categoricas.append(i)
```

```
contenedor_categoricas
```

```
#Podemos eliminar variables como 'Country','Province','country_co' porque
#esa información la tenemos en las variables lat y long
```

```
df = df.drop(['Country','Province','country_co', 'Latitud_2', 'Longitud_2', 'Id_Child_1', 'id_child'], axis=1)
```

```
nuevo_contenedor_categóricas = []
for j in df.columns.to_list():
    if (df[j].dtype == 'object'):
        nuevo_contenedor_categóricas.append(j)
```

```
nuevo_contenedor_categóricas
```

```
#Ahora queremos ver cuántos valores únicos hay por variable
for k in  nuevo_contenedor_categóricas:
    n_etiquetas = df[k].nunique()
    print(f"La variable {k} tiene {n_etiquetas}")
```

```
for e in  nuevo_contenedor_categóricas:
    etiquetas = df[e].unique()
    print(f"La variable {e} tiene las etiquetas:  {etiquetas}")
```

```
#El siguiente paso será codificar numericamente estas variables
#Esto forma parte del preprocesamiento
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
contenedor_transformadas = []
for i in nuevo_contenedor_categóricas:
    i_le = le.fit_transform(df[i])
    contenedor_transformadas.append(i_le)
```

```
contenedor_transformadas
```

```
df_transformadas = pd.DataFrame(contenedor_transformadas)
df_transformadas
```

```
df_transformadas_transpuesta = df_transformadas.T
df_transformadas_transpuesta
```

```
nuevo_contenedor_categóricas
```

```
df_transformadas_transpuesta.columns=[nuevo_contenedor_categóricas]
```

```
df_transformadas_transpuesta
```

```
new_df = pd.concat([df, df_transformadas_transpuesta], axis=1)
new_df
```

```
new_df = new_df.drop(['type_resid',
 'mother_edu',
 'Mother_job',
 'child_sex',
 'source_dri',
 'type_toile',
 'type_cooki'], axis=1)
```

```
new_df.shape
```

En este punto tenemos el dataframe con el que queremos trabajar

```
#Es conveniente que comencemos haciendo un escalado de las variables
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
```

```
x_std = sc.fit_transform(new_df)
```

```
#En principio vamos a hacer un problema de clusterizacion sin considerar ninguna variable objetivo
from sklearn.cluster import KMeans
```

```
#antes de lanzar el modelo vamos a ver cuál es el número optimo de vencindarios que podemos crear
contenedor = []
for i in range(1,10):
    k_vecinos_aux = KMeans(n_clusters=i, random_state=0)
    k_vecinos_aux.fit(x_std)
    contenedor.append(k_vecinos_aux.inertia_)

plt.plot(contenedor, 'x-')
```

```
contenedor = []
for i in range(1,20):
    k_vecinos_aux = KMeans(n_clusters=i, random_state=0)
    k_vecinos_aux.fit(x_std)
    contenedor.append(k_vecinos_aux.inertia_)

plt.plot(contenedor, 'x-')
```

```
contenedor = []
for i in range(1,30):
    k_vecinos_aux = KMeans(n_clusters=i, random_state=0)
    k_vecinos_aux.fit(x_std)
    contenedor.append(k_vecinos_aux.inertia_)

plt.plot(contenedor, 'x-')
```

vamos a considerar n_clusters=10 porque es a partir de ahí donde el ritmo de decrecimiento de la curva de inercia es mas lento

```
k_means = KMeans(n_clusters=10)
```

```
k_means.fit(x_std)
```

```
#preguntamos por los centroides
k_means.cluster_centers_
```

Tenemos 10 referencias pero es imposible representarlas en un espacio multidimensional tan grande

```
#Lo que si nos interresa es que a cada registro se le asocie una etiqueta
etiquetas = k_means.labels_
```

```
etiquetas
```

```
np.shape(etiquetas)
```

```
new_df['TAG']=etiquetas
```

```
new_df
```

```
#comprobamos que el número de etiquetas coincida con el número de clusters que decidimos crear
new_df['TAG'].nunique()
```

```
new_df.to_csv('clusterizacion_salud_nuevo_10.csv')
```

```
#veamos que ocurre si solo consideramos 5 vecindarios

k_means2 = KMeans(n_clusters=5)
```

```
k_means2.fit(x_std)
```

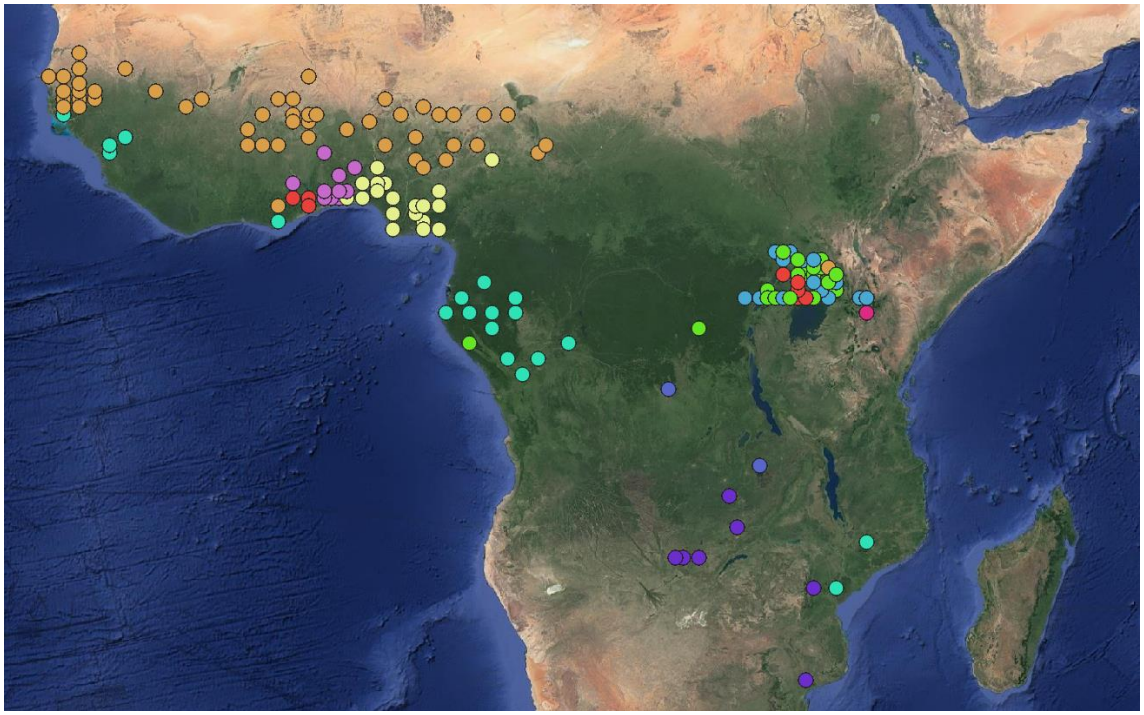
```
etiquetas2 = k_means2.labels_
```

```

```

```
new_df['TAG2']=etiquetas2
```

```
new_df.to_csv('clusterizacion_salud2_5.csv')
```

Si la agrupación la hacemos con diez grupos (‘vecindarios’) podemos sacar alguna conclusión como que existe alguna relación entre los grupos y la longitud. Todavía queda mas clara esta relación si en lugar de 10 cogemos 5 grupos.



Para cinco grupos: