Universidad del Azuay

Facultad de Ciencias de la Administración

Escuela de Ingeniería de Sistemas y Telemática

ANÁLISIS Y DESARROLLO DE APLICACIÓN WEB PARA EL ANÁLISIS ESTADÍSTICO DE DATOS DE ENCUESTAS

Trabajo de titulación previo a la obtención del grado en:

Ingeniero en Sistemas y Telemática

Autor:
Juan Miguel Corral Espinoza

Directora:
Daniela Elisabet Ballari

Cuenca – Ecuador

2021

Dedicatoria

Dedico este
trabajo de titulación a mi familia: mi papá
Miguel, mi mamá Chela, mi hermana Paz,
y mi cuñado Juan Diego Sotomayor, por
toda la comprensión, ayuda, soporte, y
buenos momentos que hemos pasado y que
espero seguir teniendo. Sin ellos, no podría
ser la persona plena que soy hoy en día.

A mi primo y
una persona que considero mi hermano,
David Corral, por ser también una de las
personas más valiosas en mi vida y con
quien hemos compartido muchos
momentos agradables juntos.

Y a mis mejores
amigos de la universidad: David Cordero,
Juan Diego Auquilla, Luis Miguel Alvear,
Sol Contreras y Santiago Cedillo, por hacer
mi estadía en la universidad una
experiencia única y muy transformativa,
quienes me mostraron la belleza de las
amistades.

Agradecimiento

Agradezco a mi directora de este proyecto de tesis, Daniela Ballari, por su increíble paciencia y ayuda con el desarrollo de este trabajo, así como su comprensión en todos los asuntos de mi vida personal que requirieron de su comprensión para poder seguir adelante.

Nuevamente, a mi familia y amigos mencionados en mi dedicatoria, quienes con lo valiosos que son me ayudaron a superar este reto con todos los buenos momentos, las palabras de aliento y su amor incondicional, y estoy segura de que me ayudarán a superar muchos otros.

Gracias.

Índice

Resumen

Muchos proyectos de investigación se fundamentan en el uso de encuestas para conocer y describir ciertos hechos y realidades. Además de los análisis estadísticos descriptivos e inferenciales comúnmente usados para este tipo de proyectos, resulta útil a los investigadores contar con una aplicación que permita, de manera intuitiva, realizar exploración de datos y cruces de variables de interés. Este proyecto describe el procedimiento seguido para la definición, estructuración e implementación de una aplicación web con las características descritas, utilizando el paquete Shiny que permite generar páginas web dinámicas para el análisis de datos inicial (de forma exploratoria) con R. Como caso de estudio, se utilizaron los datos de la encuesta de movilidad de escolares llevadas a cabo en el proyecto CEPRA-RES "Evaluación de entornos urbanos peatonales para la identificación de rutas escolares seguras en ciudades intermedias del Ecuador, 2020." El proyecto cuenta también con herramientas que facilitan el descargar y usar la aplicación. Los resultados y conclusiones alcanzados demuestran que la aplicación desarrollada es de gran utilidad para los usuarios que requieren un análisis exploratorio de datos de encuestas, y para los usuarios que desean estudiar o extender el código.

Palabras clave: encuestas, estadística, lenguaje R, shiny, docker.

Abstract

Many research projects are based on the use of polls or surveys to identify and describe certain facts. On top of both descriptive and inferential statistical analysis, which are commonly used for these kinds of projects, researchers can make use of an application that allows them to intuitively explore the collected data and perform analysis by cross-referencing variables of interest. This project describes the steps followed for the definition, structuring, and implementation of a web application with the described features, using the Shiny package which is used to generate web pages with dynamic content for an initial, exploratory analysis of the data using R in the background. As a case study, data describing transportation means of school students collected by CEPRA-RES, titled "Evaluation of urban pedestrian environments for the identification of safe school routes in medium-sized cities in Ecuador, 2020," was used. The project also provides tools that facilitate downloading and using the developed application. The results and conclusions reached at the end of this project confirm that the developed application is of great usefulness for users who need to perform exploratory analysis of survey data, as well as for users who wish to study or extend its code.

Keywords: surveys, polls, statistics, R language, shiny, docker.

Translated by

Juan Miguel Corral

Juan Corral Espinoza

Introducción

Las aplicaciones con un modelo de distribución sin cargo y de código abierto representan un aporte significativo para los desarrolladores de software en cualquier ámbito. El poder no solo reproducir y estudiar el código, sino también modificarlo para cambiar funcionalidades o agregar nuevas, y poder distribuir la versión modificada al público nuevamente, permite que las herramientas de software de este tipo puedan manejarse en un proceso de desarrollo y mejora continua. Además, permite que personas que se estén iniciando con un lenguaje de programación, pero desean estudiarlo o mejorar sus habilidades, puedan estudiar el código de la aplicación y comprender los aspectos del lenguaje de un modo más práctico.

La encuesta CEPRA-RES, realizada en conjunto entre la Universidad del Azuay, la Pontificia Universidad Católica del Ecuador Sede Ibarra y la Universidad de Cuenca, con el apoyo de CEDIA, es una encuesta sobre los medios de transporte y movilidad que utilizan los estudiantes de ciertas escuelas primarias en las ciudades de Cuenca e Ibarra. Esta encuesta fue aplicada a los padres de familia para recolectar información sobre cómo se movilizan los niños y niñas a la escuela, si éstos son acompañados, entre muchas otras preguntas. Los datos recolectados por la encuesta, posteriormente, fueron resumidos en gráficos descriptivos generados por medio del lenguaje de programación R para conseguir información sobre los datos. Sin embargo, los gráficos que se realizaron tenían un alto grado de repetición de código (es decir, falta de uso de funciones o métodos genéricos para la generación de gráficos), así como falta de flexibilidad para cambios que se pudieran realizar *a posteriori* en el conjunto de datos, ya que requeriría escribir código que se adapte a los cambios.

Por estos motivos, se consideró que sería de gran utilidad para el proyecto CEPRA-RES el generar una aplicación cuyo funcionamiento sea genérico a un conjunto de datos y pueda generar los gráficos que el proyecto necesita para generar información. De esta forma, la aplicación podría generar información estadística por medio de los gráficos que se implementen para cualquier conjunto de datos de otras encuestas. Adicional a esto, la aplicación también se decidió desarrollarla bajo un modelo de código abierto para que pueda ser libremente distribuida, estudiada y modificada por otras personas. Finalmente, la aplicación se desarrolló bajo un

framework para aplicaciones web, de tal forma que la aplicación sea accesible por cualquier usuario con tan solo tener acceso a un navegador web. Es importante notar que, la aplicación no se desarrolló con el propósito de poder generar gráficos que puedan ser utilizados como productos finales en investigaciones; sino que su motivación es la de generar gráficos que permitan realizar una exploración de la información inicial. El extender la aplicación para poder generar gráficos mucho más elaborados y de calidad final para publicación se ha identificado como prioridad para futuras investigaciones.

Aunque ya existen aplicaciones que pueden generar gráficos a partir de información estadística, un estudio del estado del arte en este ámbito no encontró un proyecto que se adapte a las características populares del caso de estudio de la encuesta CEPRA-RES que se utilizó como modelo—las aplicaciones normalmente no son de código abierto, o no utilizan R y por tanto los gráficos generados no son los mismos que los que se generan a través de R, o las aplicaciones no son lo suficientemente flexibles para el uso de cualquier tipo de conjunto de datos, sino que están desarrolladas específicamente para un conjunto de datos en particular.

Con todos estos criterios en consideración, lo que este proyecto provee es, a grandes rasgos, un alto grado de flexibilidad en términos de la posibilidad de analizar distintos tipos de datos, cruzar distintos tipos de variables, así como flexibilidad en permitir la extensión de la aplicación al proveer el código fuente por medio de un repositorio disponible públicamente.

Este documento describe el proceso seguido para el desarrollo del proyecto, aplicado al caso de estudio los datos de encuesta de CEPRA-RES. El proceso del proyecto consistió en cinco pasos que se describen a continuación. Primero, se realizó un estudio del estado del arte. Segundo, se eligió un método de desarrollo de software que sirva de guía general, permitiendo mantener un cierto orden y estructura durante la escritura de código y las pruebas realizadas en el proceso[1]. La aplicación utilizó un método de control de versionamiento para agilizar el proceso y permitir la publicación del código en un repositorio público de código. Tercero, se realizó una limpieza de los datos recolectados por la encuesta CEPRA-RES para mejorar su estructura y facilidad de

[1] Se prescindió de utilizar métodos formales de desarrollo de software, ya que el proyecto y la extensión del código a escribir no justifica la inversión de tiempo que los métodos más formales requieren, en especial ya que el desarrollo de la aplicación solo consta de un desarrollador, y los métodos formales suelen estar orientados a mejorar el desarrollo en equipo.

análisis de la información. Cuarto, la aplicación se publicó sobre un servidor Linux, y creó un contenedor Docker también como aporte adicional para personas que no tengan la facilidad de poder configurar los elementos necesarios preliminares a poder publicar la aplicación sobre un servidor Linux. Se eligió utilizar Docker para que la aplicación pueda distribuirse y publicarse de forma más fácil. Quinto y último, se realizaron pruebas sobre la aplicación publicada en el servidor con investigadores, considerando tanto investigadores que conozcan del lenguaje R como investigadores que no, para recibir retroalimentación e identificar la utilidad que tiene la aplicación para investigadores que requieran análisis estadísticos de datos de encuestas.

Marco Teórico

1. Estadística

La estadística es un área de estudio de las matemáticas aplicadas que concierne la colección, el análisis, la interpretación y presentación de datos para obtener información. La estadística cuenta con dos grandes ramas que describen el tipo de información que se obtiene como resultado considerando los datos analizados: la estadística descriptiva y la estadística inferencial. (Witte & Witte, 2017)

Estos dos tipos de estadística, descriptiva e inferencia, son esenciales para el análisis de datos de encuestas. La estadística descriptiva permite describir los resultados de la muestra a través de estadísticos como la frecuencia, media, mediana o moda, desviación estándar o rango intercuartil, y la representación gráfica con gráficos de histogramas, de caja y bigote, entre otros. Por otro lado, la estadística inferencial, a través de pruebas de hipótesis, permiten testear hipótesis de diferencias significativas entre grupos o muestras. (Witte & Witte, 2017)

El análisis estadístico es particularmente útil cuando se utilizan medios gráficos para la representación de los datos. Las herramientas gráficas de la estadística son herramientas como el gráfico de frecuencias o el gráfico de caja y bigote, y permite que la información obtenida de los datos pueda mostrarse a terceros de manera que se minimice el tiempo que puede tomar el entender los resultados de los análisis estadísticos realizados.

2. Estadística Descriptiva

La estadística descriptiva es la rama de la estadística que describe las condiciones y características de una población en base a los datos obtenidos de dicha población, normalmente en base a una muestra (University of Alabama at Birmingham, s/f). Usando dichos datos, se describe la situación actual de la población, normalmente utilizando técnicas como categorización, frecuencias o medias para tal propósito. Por ejemplo, describir la situación económica de una ciudad se realiza

mediante la obtención de una muestra (un número reducido y de selección aleatoria de la población total) que sea representativa de la población total.

Un ejemplo del tipo de análisis estadísticos que se realizan dentro de la estadística descriptiva es el gráfico de frecuencias con las medidas de media, mediana y moda. Este tipo de gráfico, junto con las medidas mencionadas, permite interpretar la información presentada de forma gráfica y resumida, por lo que son las técnicas de análisis más básicas dentro de esta área de estudio.

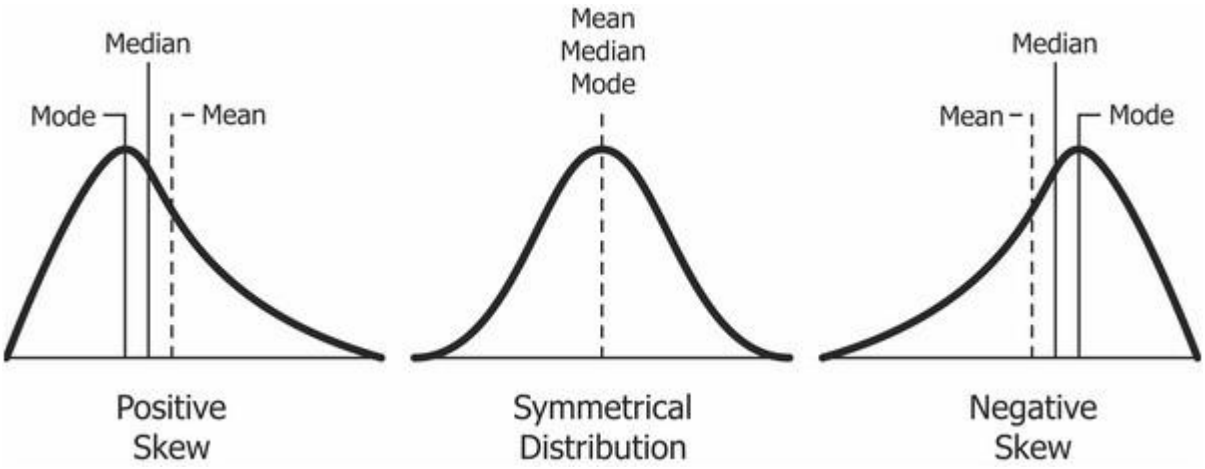


Figura 1. Gráfico de distribución de frecuencias suavizado con las medidas de centralidad de media, mediana y moda mostradas. (Narkhede, 2018)

3. Estadística Inferencial

Por otro lado, la estadística inferencial es la que realiza predicciones sobre una situación de la vida real en base a los datos (University of Alabama at Birmingham, s/f), y realiza conclusiones sobre los parámetros de la población a partir de los resultados obtenidos de una muestra. Es decir que, si la estadística descriptiva podría describirse como la rama que se enfoca en el pasado y presente, la estadística inferencial es aquella que se enfoca en el futuro en términos de los resultados que genera. El mejor ejemplo de estadística inferencial se encuentra en las predicciones meteorológicas, donde en base a datos del pasado y presente, se pueden hacer predicciones sobre el estado futuro del clima por medio del análisis de comportamientos (Narkhede, 2018).

4. Lenguaje de Programación R

El lenguaje de programación R es un lenguaje que está orientado al análisis y presentación de información estadística (The R Project, s/f). Por ende, incluye ya entre sus librerías las formas de realizar distintos análisis estadísticos, así como varias formas de representar la información obtenida de manera gráfica. De igual forma, su sintaxis está orientada para trabajar con datos estadísticos (es decir, datos que normalmente se representan a través de una o varias tablas). La visualización de gráficos en R ha mejorado notablemente en los últimos años gracias al desarrollo de librerías como ggplot (Valero-Mora, 2010) y plotly (Sievert, 2020). En este proyecto, se escribió el código de R por medio de RStudio.

5. RStudio

RStudio es un entorno de desarrollo integrado (comúnmente llamado IDE por sus siglas en inglés de Integrated Development Environment) para el lenguaje R. RStudio simplifica la creación y trabajo de proyectos de R de distintos tipos, así como proveer facilidades y servicios que complementan el desarrollo como la instalación y manejo de paquetes en R, uso de control de versionamiento, ayuda y autocompletado de código, etc (RStudio Team, 2021).

6. Paquetes de R

Los paquetes de R son recopilaciones de código que permiten implementar o agregar funcionalidad a una aplicación de R. La instalación de paquetes en una aplicación R se logra con facilidad a través de RStudio, el cual provee herramientas que simplifican el proceso de instalar, manejar o quitar paquetes. A continuación, se listan los paquetes más importantes que se utilizaron para el proyecto. Nótese que los paquetes listados no representan una lista exhaustiva de todos los paquetes que el proyecto utiliza, sino solo aquellos paquetes que se consideró requieren una explicación por separado.

6.1. Paquete shiny

El paquete Shiny es un paquete de R con el cual se pueden diseñar interfaces gráficas de forma rápida para el análisis y visualización de datos ingresados por R (Shiny Team, s/f). Mediante este paquete, se pueden realizar funciones como el ingreso de datos o interacción dinámica por parte del usuario para la visualización de los datos.

El funcionamiento de Shiny consiste, en esencia, de dos grandes categorías de componentes: la categoría de componentes reactivos, los cuales son elementos que originan una reacción en el resto de la aplicación (por ejemplo, presionar un botón); y la categoría de componentes reactantes, los cuales cambian su presentación o funcionalidad luego de recibir un evento de un componente reactivo (RStudio Team, 2017). Estos dos tipos de componentes son de utilidad para la generación de páginas web dinámicas, donde el cambio del estado de ciertos elementos dentro de una página web no requiere que la página web se refresque o que se navegue a una distinta página dentro de un sitio—basta con generar un evento reactivo para que el contenido se actualice automáticamente.

6.2. Paquete shinydashboard

El paquete shinydashboard es un complemento al paquete Shiny, que cambia el aspecto visual de los componentes de Shiny para hacerlos más atractivos y modernos, así como dar la opción de elegir entre distintos temas (paletas de colores) que una aplicación Shiny utiliza para sus componentes, para lograr una apariencia visual uniforme y consistente (RStudio Team, 2014).

6.3. Paquete ggplot2

Los gráficos que se pueden generar a través del lenguaje R tienen poca capacidad de configuración, y no son muy atractivos visualmente, sino que solo están presentes para una visualización rápida. Para la generación de gráficos con apariencia moderna y altamente configurable en la aplicación, se utilizó el paquete ggplot2. Éste permite generar un gran número de distintos tipos de gráficos estadísticos, así como configurar muchos aspectos de su apariencia, incluyendo la paleta de colores que se usa, el rellenado de los gráficos, dónde se colocan las etiquetas que representan los datos, entre muchas otras opciones (Wickham et al., 2021).

6.4. Paquete vcd

El paquete vcd, que representa las siglas de Visualizing Categorical Data (Visualizando Datos Categóricos) es un paquete similar a ggplot2, en que provee funciones y librerías para la graficación de datos de tipo categórico (Meyer et al., 2020). Vcd se utilizó para la generación de los gráficos de mosaico que el proyecto necesita, los cuales se generan con más configurabilidad que los gráficos de mosaico de ggplot2.

6.5. Paquete viridis

El paquete viridis complementa a la generación de gráficos con ggplot2 al proveer una paleta de colores que está específicamente diseñada para tener una distribución de colores lo más distinguibles para el ojo humano, tanto para personas con visión común como aquellos que tengan alguna forma de daltonismo (Rudis et al., 2021), de forma que la visualización de datos requiera el menor esfuerzo posible por parte de las personas que realizan la interpretación de los gráficos generados.

7. Docker

El servicio Docker es un servicio que ha ganado alta popularidad en el área de desarrollo de software y administración de servidores por ofrecer los llamados contenedores de Docker, contenedores lógicos que almacenan todo un entorno con el propósito de ofrecer algún tipo de servicio o funcionalidad específica a través de ellos (Docker Inc., s/f). Por ejemplo, si una aplicación requiere de la instalación de ciertos servicios o programas por parte del sistema operativo para su funcionamiento, se puede crear una imagen Docker que describa cómo realizar la configuración del contenedor para que todos los requerimientos de la aplicación se instalen de forma aislada. Luego, el poder ejecutar dicha aplicación (o el servicio que el contenedor Docker provea) solo consiste en ejecutar el contenedor. De esta forma, se pueden distribuir servicios a través de contenedores Docker que son fáciles de configurar y ejecutar.

8. Proceso de Desarrollo Incremental e Iterativo

El proceso de desarrollo incremental e iterativo es un proceso en el área de desarrollo de software cuya metodología se enfoca en que el desarrollo se dé por fases que se repiten durante todo el ciclo de dicho desarrollo de software, de manera que el producto esté en constante evolución para adaptarse a nuevas necesidades y requisitos que pueden encontrarse durante el proceso, permitiendo así que la funcionalidad y arquitectura del producto final entregado no esté atado a un diseño realizado antes de la fase de desarrollo del software como tal (Paasivaara, 2006). Lo descrito anteriormente se refiere a la parte de “iterativo” del proceso. Las fases que involucran este proceso iterativo no se encuentran formalmente definidas, pero normalmente incluyen una fase de planeación durante la cual se encuentran los requerimientos; una fase de análisis de los

requerimientos y diseño de la funcionalidad; una fase de implementación de la funcionalidad; y una fase de testeo y evaluación donde se prueba la funcionalidad implementada y se evalúa si se requieren cambios.

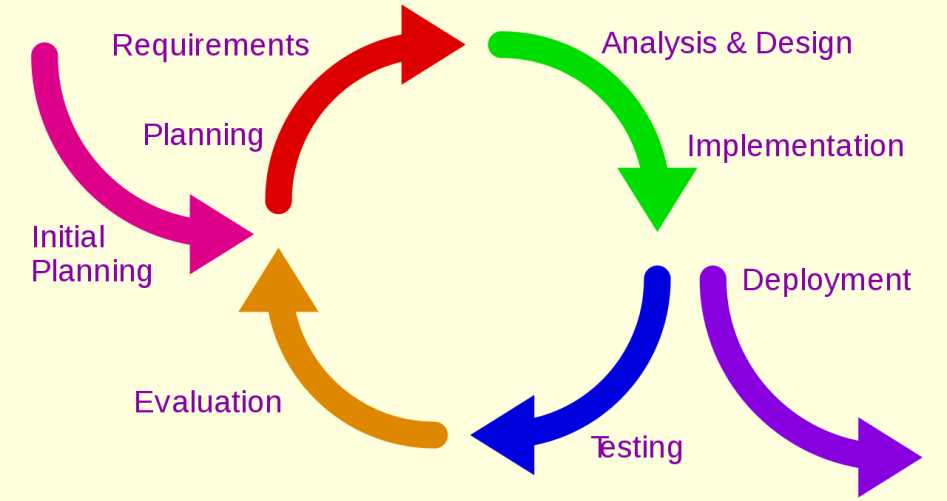


Figura 2. Diagrama de una forma particular de implementar desarrollo iterativo. (Wikipedia, 2020)

En cuanto al aspecto incremental del proceso, se refiere a definir y producir componentes del producto final que sean pequeños, y que, a medida que el proceso iterativo continúa, estos componentes se vayan agregando entre sí para "incrementar" la funcionalidad del producto, hasta que se considere que todos los componentes necesarios se hayan producido y la combinación de estos contengan toda la funcionalidad requerida por las partes interesadas del proyecto de software (Larman & Basili, 2003).

Cabe recalcar que este proceso de desarrollo no está definido de forma estrictamente formal, y por tanto es altamente adaptable a las necesidades del proyecto en el que se aplique. Como se describió anteriormente, las fases que se apliquen en el aspecto iterativo del proceso pueden variar, así como puede variar cómo se definan el tamaño de un componente y el tiempo necesario a invertir en el desarrollo de dicho componente.

En este trabajo se consideró que un proceso de desarrollo como el iterativo e incremental permitía tener un cierto nivel de definición de las fases en las que se desarrollará el software, pero con la flexibilidad requerida por este tipo de proyecto. Otros procesos como las metodologías ágiles implementan también en gran parte una filosofía incremental e iterativa, pero este tipo de

metodologías normalmente requieren de la definición de pasos y partes interesadas de forma más estricta, y están diseñadas a aplicarse con equipos de trabajo de mayor tamaño al del trabajo presente (Todaro, 2019).

Estado del Arte

Para la revisión del estado del arte en el ámbito del proyecto, se realizaron búsquedas de artículos científicos en fuentes académicas, de manera que se pueda describir los distintos proyectos que se encuentren realizados sobre el paquete Shiny de R, y relacionar así la validez el problema de investigación planteado y su contribución, en el contexto académico-científico.

La búsqueda de artículos relevantes se realizó en tres bibliotecas digitales:

- ACM
- IEEE Xplore
- SpringerLink

El criterio de búsqueda para estas tres librerías se dio por medio de la cadena de búsqueda: **"R language AND shiny"**, con la palabra clave *AND* siendo un conector lógico para la búsqueda. Dicha cadena de búsqueda se adaptó para la sintaxis que los motores de búsqueda de cada librería requieren. Adicionalmente, se realizó una búsqueda de artículos relevantes por medio de Google Scholar.

De los artículos encontrados se seleccionaron un subconjunto por medio de la técnica de *screening*. Para ello se realizó la lectura del título y *abstract* para determinar su relevancia respecto al estudio del estado del arte en este proyecto. El resultado final mostró que no existe un proyecto igual o muy parecido al propuesto en este trabajo de tesis.

Por ejemplo, el artículo "*Interactive pharmacometric applications using R and the Shiny package*" (Wojciechowski et al., 2015) es una aplicación realizada con R y el paquete Shiny que se desarrolló para la visualización y control de la información sobre simulaciones en el área farmacéutica. Esta aplicación está orientada a dicho ámbito, y por tanto no puede ser utilizado para análisis estadísticos de encuestas.

Otro ejemplo es el artículo "*Tools for Interactive Visualization of Global Demographic Concepts in R*" (Walker, 2016), el cual también implementa métodos de visualización de datos con Shiny,

pero los utiliza para la visualización de datos demográficos por medio del paquete de graficación *plotly*, utilizando como ejemplo datos obtenidos de la base de datos de censos en los Estados Unidos.

Artículos como "*CheckMyIndex: A web-based R/Shiny interface for choosing compatible sequencing indexes*" (Varet & Coppée, 2019), "*CIPR: a web-based R/shiny app and R package to annotate cell clusters in single cell RNA sequencing experiments*" (Ekiz et al., 2020), "*Ioncopy: An R Shiny app to call copy number alterations in targeted NGS data*" (Budczies et al., 2018) y "*ShinyCircos: An R/Shiny application for interactive creation of Circos plot*" (Yu et al., 2018) demuestran el uso de R y el paquete Shiny para la selección de datos o creación de ciertos tipos de gráficos, como el gráfico Circos, utilizado para la visualización de información genética.

De forma similar, otra de las publicaciones halladas, "MetaInsight: An interactive web-based tool for analyzing, interrogating, and visualizing network meta-analyses using R-shiny and netmeta" (Owen et al., 2019) demuestra un uso de las herramientas de software para el análisis de datos de redes que, en su aplicación, puedan utilizarse en redes de información médica.

Otros artículos a mencionar son "*An implementation of cloud-based platform with R packages for spatiotemporal analysis of air pollution*" (Yang et al., 2020), un artículo orientado al análisis de datos sobre contaminantes atmosféricos; y "*minotaur: A platform for the analysis and visualization of multivariate results from genome scans with R Shiny*" (Verity et al., 2017), un artículo que se enfoca en el análisis y visualización de datos multivariables. Los análisis de los datos realizados en estos artículos están relacionados con el análisis estadístico—no obstante, su área de estudio es específica y no permite un análisis de datos más generalizado.

Un artículo donde se menciona haberse utilizado un caso de estudio hipotético es en "*R and Shiny for Cost-Effectiveness Analyses: Why and When? A Hypothetical Case Study*" (Hart et al., 2020). Su principal enfoque es en analizar la eficiencia entre utilizar R y Excel y por tanto no tiene mucha relevancia en términos de la realización de una aplicación en R y Shiny; sin embargo, su uso de un caso de estudio (a pesar de ser hipotético y no obtenido de datos reales) provee una buena línea base para el desarrollo del proyecto de tesis actual y su aplicación en el caso de estudio de los datos de movilidad de CEPRA.

El artículo más relevante al tema propuesto en este proyecto de tesis es "*Implementing novel, flexible, and powerful survey designs in R Shiny*" (Kaufman, 2020). Este artículo está orientado hacia el uso dentro de la aplicación de encuestas, permitiendo que un investigador pueda utilizar la solución dada para la creación de encuestas que permitan tener flexibilidad al diseñar y distribuir las preguntas que se aplicarán en la encuesta. Éste es uno de los artículos más novedosos hallados en el estudio del estado del arte, ya que demuestra la capacidad de generar encuestas dando flexibilidad al usuario al momento de hacerlo. No obstante, su aplicación no está relacionada con el análisis de los datos de encuestas como el de este proyecto de tesis.

Todos los artículos mencionados demuestran que la aplicación del paquete Shiny con R para el análisis de datos de encuestas con un cierto grado de generalización no se encuentra disponible en una publicación científica. Adicionalmente, varios de los artículos revisados no proveían acceso al código fuente de las aplicaciones realizadas en R y Shiny, de manera que se tenga la capacidad de revisar, mejorar o extender la funcionalidad de las aplicaciones.

Considerando todo lo anterior, se puede concluir que el proyecto actual que se desarrollará provee una contribución e innovación en el ámbito científico, en especial al dar acceso al código fuente de manera que la funcionalidad de la solución pueda utilizarse o extenderse por otros desarrolladores para cubrir brechas que se puedan encontrar en futuras investigaciones.

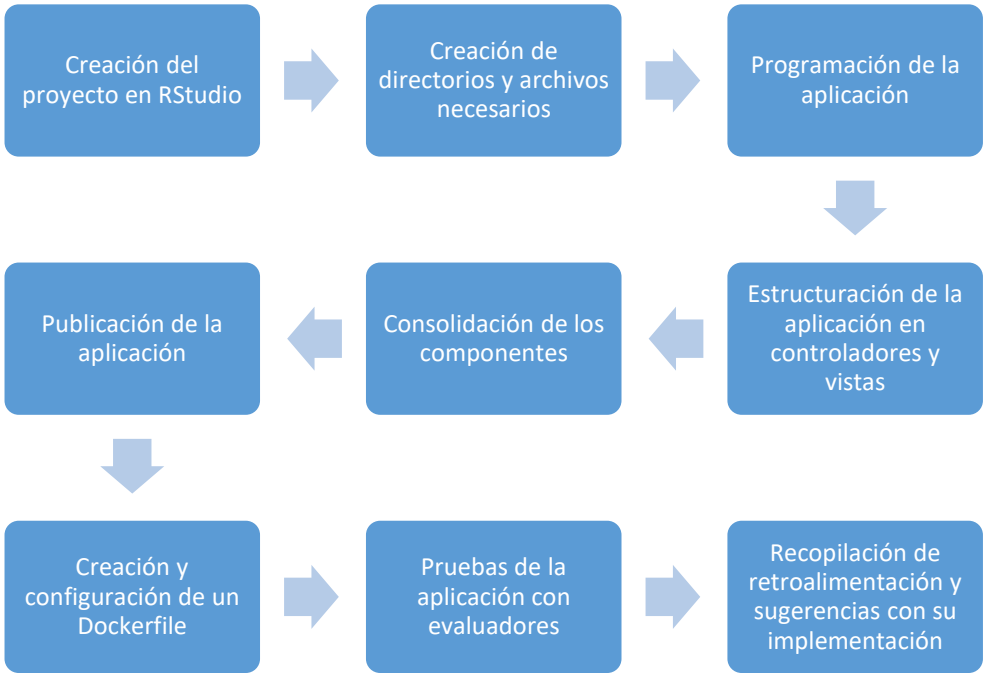
Métodos

9. Estructuración de los pasos en el proyecto

Siguiendo la metodología de desarrollo incremental e iterativa propuesta para el proyecto, se realizó una descripción inicial de los pasos y etapas en los que se debía trabajar.

A grandes rasgos, el proyecto está orientado no solo hacia el uso por parte del usuario de la aplicación final que se genere en este proyecto, sino también hacia publicar el código en un repositorio de libre acceso de manera que otros desarrolladores e investigadores puedan estudiar el código. Asimismo, la disponibilidad del código permite a otros desarrolladores descargar el código y extenderlo por su propia cuenta.

Considerando estas características, los pasos propuestos inicialmente fueron:



10. Limpieza y unión de los datos de CEPRA-RES de Cuenca e Ibarra

Los datos que se obtuvieron de la encuesta CEPRA-RES se encuentran divididos entre distintos archivos para los datos que son de Cuenca y los datos que son de Ibarra. Esto significa que, para determinar la información de ambas ciudades, deben examinarse dos archivos por separado. Se consideró entonces que unir los datos entre los dos archivos sería importante para poder visualizar los datos en conjunto de ambas ciudades si se desea, lo cual a su vez ayudaría a que la aplicación pueda realizar un análisis por la ciudad donde se recolectaron los datos con solo utilizar un archivo unificado.

Primero, se determinó que los datos que contenía el archivo de Ibarra tenían los nombres de columnas más apropiados, ya que los datos de la ciudad de Cuenca contaban con nombres de columnas que describía si una columna trataba información correspondiente a una pregunta dentro de una pregunta anterior. Por ejemplo, el campo con nombre "**Datos encuestado/3.3. Dirección donde vive su hijo/a o representado**" en los datos de Cuenca tiene por nombre solo "**3.3. Dirección donde vive su hijo/a o representado**", que se considera es mejor para el listado dentro de las opciones de variables en la aplicación, haciéndolo más fácil de leer para el usuario.

Segundo, los datos de Ibarra contenían columnas extra repetidas de anteriores columnas que no poseían datos. Por tanto, estas columnas se eliminaron de los datos de Ibarra.

Tercero, los datos de Ibarra ya incluían dos columnas que los datos de Cuenca no contenían: una columna para el nombre de la ciudad (donde todos los datos tienen como valor en esta columna el valor de "Ibarra"), y otra para el nombre de la escuela donde se aplicó la encuesta. Los datos de Cuenca no poseen información sobre la escuela donde se recolectaron en el archivo, pero se sabe que la escuela donde se recolectaron los datos se indica de forma implícita en la columna de código de la encuesta, la cual contiene las iniciales de la escuela donde se obtuvo la información. Se sabe que las escuelas de la ciudad de Cuenca que participaron en la encuesta de CEPRA-RES son las escuelas Luis Cordero, Nicolás Sojos e Isabel Moscoso. Por tanto, para un código de encuesta que comienza por "LC_", se puede llenar el campo de escuela para esa fila con el valor "Luis Cordero".

Con todos estos cambios realizados en un archivo de Excel, se guardaron los datos concatenados con la nueva información como "**Datos Concatenados.xlsx**", el cual será posteriormente utilizado por la aplicación para el análisis de la información recolectada.

11. Creación del proyecto base en RStudio

RStudio es un entorno de desarrollo integrado (*integrated development environment* en inglés, y comúnmente abreviado como IDE, cuyas siglas serán utilizadas de aquí en adelante para referirse a este tipo de herramientas) para el lenguaje R, el cual provee de varias funcionalidades que ayudan a un desarrollador del lenguaje el agilizar y facilitar el desarrollo de scripts o proyectos (RStudio Team, 2021). Para este proyecto, las funcionalidades que se consideraron de utilidad para el desarrollo incluyeron:

- **Asistencia con escritura de código:** abarca varias funcionalidades hijas que ayudan a que la escritura del código se vea simplificada y se puedan detectar errores de sintaxis. Incluye funcionalidades como coloreo de sintaxis, con la cual las palabras clave del lenguaje R se resaltan de distintas formas dependiendo de su funcionalidad; asistencia para completar código, con la cual se puede presionar un conjunto de teclas cuando se está escribiendo código de manera que el IDE genere sugerencias sobre nombres de variables, o nombres y argumentos de funciones; y el subrayado de errores de sintaxis, con la cual el código que sea sintácticamente incorrecto se resalta con una línea ondulada de color rojo para indicar que el código escrito no se ejecutará correctamente y debe ser corregido.

- **Uso de renv: renv** es un administrador del entorno de un proyecto del lenguaje R, que se encuentra incluido para su uso en RStudio. Aunque se pueden crear proyectos que no hagan uso de **renv**, usarlo en un proyecto contribuye a la portabilidad del proyecto. **renv** maneja un listado de todas las librerías que se hayan instalado para el proyecto, y se encarga de que estas librerías se encuentren disponibles para el uso del proyecto. Este listado significa que no es necesario exportar el proyecto junto a todas las librerías que el proyecto utilice para que pueda ser publicado en un repositorio en línea (que fácilmente puede representar un incremento excesivo del tamaño de los archivos del proyecto)—en cambio, solo se debe exportar el archivo que mantiene el listado, que normalmente solo es de un tamaño de pocos *kilobytes*. Cualquier persona que desee ejecutar el proyecto en un entorno diferente para el que fue creado solo debe asegurarse de tener la herramienta **renv** disponible y ejecutar un comando en la línea de comandos de R, para que **renv** restaure

todas las librerías que el proyecto necesita, replicando así el entorno original del proyecto (Ushey, 2021).

- **Conexión con control de versionamiento y repositorios con Git:** RStudio permite realizar control de versionamiento por Git, así como conectarse a un repositorio remoto alojado en un servicio como GitHub, para mantener un historial de los cambios realizados al proyecto y alojarlo en un repositorio en internet. Así, el proyecto puede recuperarse en caso de pérdida de los archivos locales. El factor más importante, sin embargo, es el libre acceso a otras personas al publicar el proyecto en GitHub. Debido a que uno de los objetivos de este proyecto es que el código pueda ser estudiado y extendido por otros desarrolladores si lo desean, publicarlo en GitHub es fundamental para alcanzar este objetivo.

- **Soporte directo para proyectos Shiny:** RStudio permite crear los archivos base para un proyecto Shiny funcional, así como proveer funcionalidades como iniciar o detener el servidor para el proyecto, cuyo propósito es ejecutar el proyecto Shiny dentro de la red local como si ya se encontrase publicado en un servidor en internet, y probar el funcionamiento de la aplicación en un navegador web.

La creación de un proyecto Shiny en el IDE RStudio con todas estas funcionalidades se realiza de la siguiente manera: luego de escoger la opción de “Nuevo Proyecto” en el IDE, se escogieron las opciones de crear un proyecto Shiny en un nuevo directorio, se le dio el nombre de “**encuestas**” al proyecto, y se marcaron las casillas para crear un repositorio Git (para versionamiento) y para utilizar **renv** en el proyecto, como se observa en la figura 3.

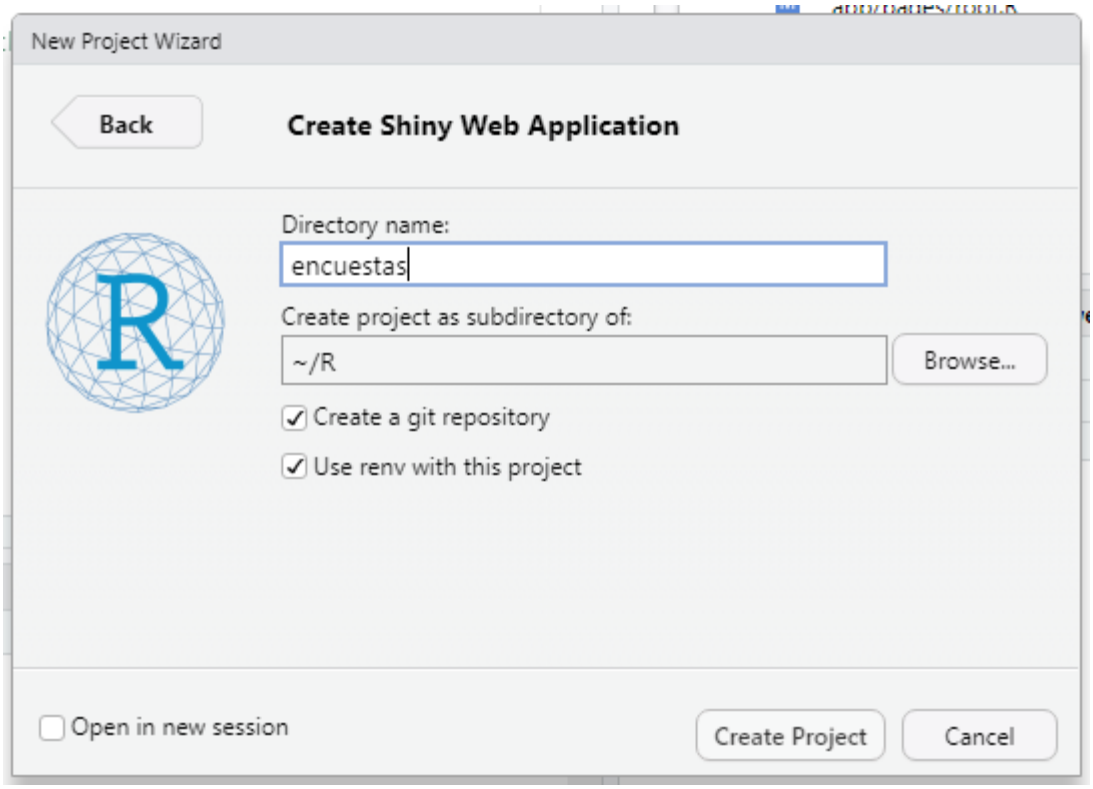


Figura 3. El diálogo para la creación de un proyecto Shiny en RStudio

Con ello, se crean todos los archivos y carpetas del proyecto para tener una aplicación Shiny básica.

11.1. Creación de directorios y archivos adicionales para organización del código

Separación de los componentes de interfaz de usuario y componentes de lógica de negocio

Los archivos que se crean para una aplicación Shiny por parte de RStudio no se encuentran organizados en carpetas dentro del directorio del proyecto. Adicionalmente, RStudio crea los archivos que permiten el funcionamiento de la aplicación Shiny de la forma más simplificada posible — esto es, crea un solo archivo llamado *app.R* que contiene todo el código para la ejecución de la aplicación en un servidor web.

Las aplicaciones Shiny son dinámicas por naturaleza, y tienen dos grandes componentes: el código para la interfaz de usuario, que se encarga de mostrar los elementos visuales al usuario, como son los campos de texto o los espacios para los gráficos; y el código de servidor, para la lógica interna,

que se encarga de realizar cálculos y procesamiento, y de actualizar los elementos visuales de la interfaz con el procesamiento realizado.

Por medio del código de servidor es que con una aplicación Shiny se puede lograr funcionalidad como que un usuario seleccione una opción en una caja de selección en una página web, y un gráfico estadístico en la misma página se actualice automáticamente luego de la selección, sin necesidad de refrescar la página. Esta funcionalidad dinámica es la deseada como objetivo de este proyecto.

Para aplicaciones más simples, tener estos dos grandes componentes de código dentro de un mismo archivo puede ser suficiente, pero el código organizado de esta forma puede volverse difícil de leer y mantener cuando la aplicación es más compleja. Shiny, afortunadamente, también considera una aplicación válida y funcional el tener el código del archivo *app.R* separado en dos archivos, *server.R* y *ui.R*, con el código correspondiente a la lógica de negocio en el archivo *server.R*, y el código para la interfaz en *ui.R*. Aunque el código de la aplicación no se vaya a escribir solo en estos dos archivos, y por ende este cambio con respecto a utilizar el archivo *app.R* no tenga un impacto significativo para este proyecto en particular, se consideró importante que esta separación lógica de los diferentes componentes se aplique de igual forma al resto del proyecto, de tal manera que todo el proyecto sea consistente, y por tanto defina un cierto estándar a seguirse para personas que deseen extender el código.

Creación de directorios para organización de los archivos de la aplicación

La aplicación consta con archivos de distintas índoles, sean código, datos u otra categoría. Por ejemplo, la herramienta **renv**, en el momento de creación del proyecto, ya separa sus archivos dentro de un directorio llamado de la misma forma, *renv*. Como el proyecto cuenta con varios archivos que pueden categorizarse de forma distinta por su funcionalidad, se decidió crear 3 directorios:

- Directorio **app**: Donde se almacenarán todos los archivos de código que correspondan directamente con el funcionamiento de la aplicación. Puede considerarse el directorio por defecto para archivos de código fuente que no puedan categorizarse bajo otros directorios.

- Directorio **app/controllers**: Es decir, un subdirectorio llamado **controllers** dentro del directorio **app**. Aquí se alojarán los archivos que contengan el código de servidor (es decir, el código encargado de la lógica de la app y que sus componentes sean reactivos) para cada página web disponible en la aplicación. Los archivos dentro de este directorio que quieran ser usados por la aplicación deben terminar con el nombre **_controller.R** para que sean importados apropiadamente.

- Directorio **app/views**: Aquí se alojarán los archivos que contengan el código para la interfaz de usuario. Los archivos dentro de este directorio que se deseen utilizar por la aplicación deben terminar con el nombre **_ui.R**.

- Directorio **data**: Directorio para almacenar los archivos de datos, correspondientes a las encuestas de CEPRA-RES, que la aplicación utilizará por defecto para los análisis estadísticos. Aquí se encuentran, por ejemplo, los archivos en formato **.xlsx** de Microsoft Excel de los datos de la encuesta. Se debe recalcar que este directorio contiene información que se considera sensible, ya que son los datos de la encuesta CEPRA-RES que, una vez el proyecto se publique en un repositorio público, cualquier persona podría acceder a la información de la encuesta, incluso considerando que los datos de la encuesta no contienen información identificable de cada persona que participó. Por ello, este directorio no se publicará y los archivos con los datos de la encuesta solo se podrán conseguir de forma privada.

- Directorio **services**: Archivos de código fuente que sean "transversales" al funcionamiento de la aplicación. Es decir, son archivos que podrían ser o que son usados por varios otros componentes o archivos de la aplicación; así, el código aquí provee un "servicio", y de ahí el nombre escogido para este directorio. Por ejemplo, aquí se podría alojar un archivo que se encargue de realizar cálculos matemáticos, independientemente de qué uso se les dé a los resultados de dichos cálculos en la aplicación. Por esta misma naturaleza, el código en este directorio debe ser "agnóstico" al resto de la aplicación, y su funcionalidad siempre deberá consistir en recibir información, procesarla, y entregar un resultado.

- Directorio **www**, subdirectorios **www/assets** y **www/css**: Este directorio almacenará recursos web que se vayan a utilizar con la aplicación. El subdirectorio **assets** almacenará

imágenes que se muestren en la app; el subdirectorio **css**, por otra parte, almacenará archivos **.css** que se utilicen para definir estilos en la app y cambiar la apariencia. El nombre de este directorio se eligió ya que Shiny por defecto considera los archivos dentro de la carpeta **www** como disponibles para uso del código de interfaz. Así, por ejemplo, para referenciar un archivo **.css** que esté dentro del directorio **www/css** en el código de interfaz, solo hace falta referenciarlo como "css/nombre_del_archivo.css", lo cual es intuitivo de entender.

Con la creación de estos directorios en el proyecto, se tiene un esqueleto de la arquitectura de la aplicación. La aplicación en este estado se considera que alcanzó un hito a ser marcado como tal mediante el uso de Git.

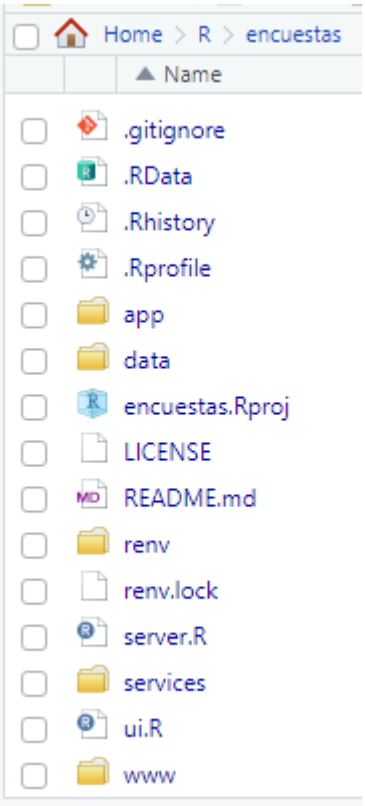


Figura 4. La estructura de carpetas del proyecto

Uso de Git y sincronización con el repositorio remoto en GitHub

Git es la herramienta de control de versionamiento que se escogió para la aplicación debido a que RStudio se integra directamente con ella y con GitHub. Para usar Git con RStudio, solo hace falta tenerlo instalado en la misma máquina donde se encuentra RStudio.

Con el esqueleto de la aplicación realizado del paso anterior, se realizó un *commit* con Git. Un *commit* es el nombre de la acción que Git usa para referirse a realizar un "guardado" o "confirmación" de los cambios realizados en un directorio de archivos (Git Team, s/f-a). Los *commits*, por convención y buenas prácticas, se realizan cada vez que se han introducido cambios en los archivos de una aplicación que representen, en conjunto, un cambio "lógico" en el funcionamiento o estructura de la aplicación. En este caso, por ejemplo, el cambio lógico en un primer *commit* es que la aplicación pase de no existir a existir con una arquitectura definida.

Una vez se ha realizado el *commit*, es recomendable sincronizar los cambios locales con los cambios en un repositorio remoto (en este caso, en GitHub). Antes de poder sincronizar los cambios, se debe crear un repositorio remoto en la página de GitHub que sea el que almacenará los archivos de la aplicación. Para este proyecto, se creó un repositorio llamado **encuestas**, con URL https://github.com/HollowVin/encuestas. El repositorio se creó sin un archivo **README.md** y sin un archivo **LICENSE**, que GitHub normalmente permite crear por defecto, los cuales se utilizan, respectivamente, para proveer a otros usuarios una descripción y otras instrucciones sobre el proyecto, y la licencia sobre la cual el proyecto de software se hace disponible a otros usuarios para que pueda ser utilizada y extendida. Estos archivos no se crearon inicialmente porque, si son creados en el repositorio remoto, intentar sincronizar el repositorio local con el repositorio en GitHub causa problemas, ya que Git piensa que ambos repositorios son iguales, pero tienen archivos que difieren significativamente, por lo que la creación de estos archivos se realiza después de que la sincronización haya sido ejecutada.

Una vez creado el repositorio remoto, éste provee URLs para configurar el proyecto local y lograr la sincronización. Entre las URLs que provee GitHub se incluye una URL por el protocolo HTTPS, la cual es la más sencilla de configurar y provee seguridad en transmisión de los datos de archivos desde el repositorio local al remoto, por lo que se consideró la mejor opción, considerando también que la guía de GitHub también recomienda el uso de HTTPS.

Con esta URL, se configura Git en RStudio para el proyecto para que utilice esta URL como identificador del repositorio remoto. Un paso más que se consideró conveniente realizar antes de proceder a la sincronización fue la creación de los archivos **README.md** y **LICENSE** mencionados anteriormente. Por el momento, el archivo **README.md** solo contará con una descripción breve del proyecto. Por otro lado, el archivo **LICENSE** contará con el texto específico que se utiliza para definir a un proyecto por medio de la licencia MIT. Esta licencia se escogió al dar un alto grado de libertad a otros desarrolladores de utilizar y extender la aplicación, con la condición de que el usuario acepta que dicha aplicación se provee sin ninguna garantía legal sobre su funcionamiento correcto, así como liberar de responsabilidad al autor del proyecto por cualquier tipo de daño o injuria legal (Open Source Initiative, s/f). Es, en resumen, una licencia que da tanto libertad de tomar y usar o extender la aplicación como otros deseen sin ningún compromiso legal con el autor. El texto completo de este archivo se encuentra en el Anexo. El haber agregado estos archivos luego de realizar el último *commit*, correspondiente a la arquitectura de la aplicación, causa cambios nuevos con Git, con lo que estos nuevos archivos tuvieron que volver a guardarse por medio de otra acción *commit*.

Después de haber configurado la URL, se puede realizar la acción de **push** para sincronizar el proyecto. **push** significa que los cambios locales se envían, o "empujan" hacia el repositorio remoto (Git Team, s/f-c). Su acción contraria es **pull**, la que se utiliza para recibir, o "halar", cambios desde el repositorio remoto (Git Team, s/f-b). Para este proyecto, **pull** nunca se utilizó ya que solo tiene sentido usarlo si existen cambios en el repositorio remoto que no se hubiesen realizado localmente—es decir, si otro desarrollador trabaja con su propia copia del repositorio. Como el proyecto no cuenta con otros desarrolladores, solamente se utiliza **push** para respaldar los cambios locales de forma remota. Luego de realizar la acción *push*, ambos *commits* son enviados al repositorio remoto para actuar como respaldo en caso de pérdida local de los archivos del proyecto.

El repositorio remoto resultante se observa de la siguiente forma:

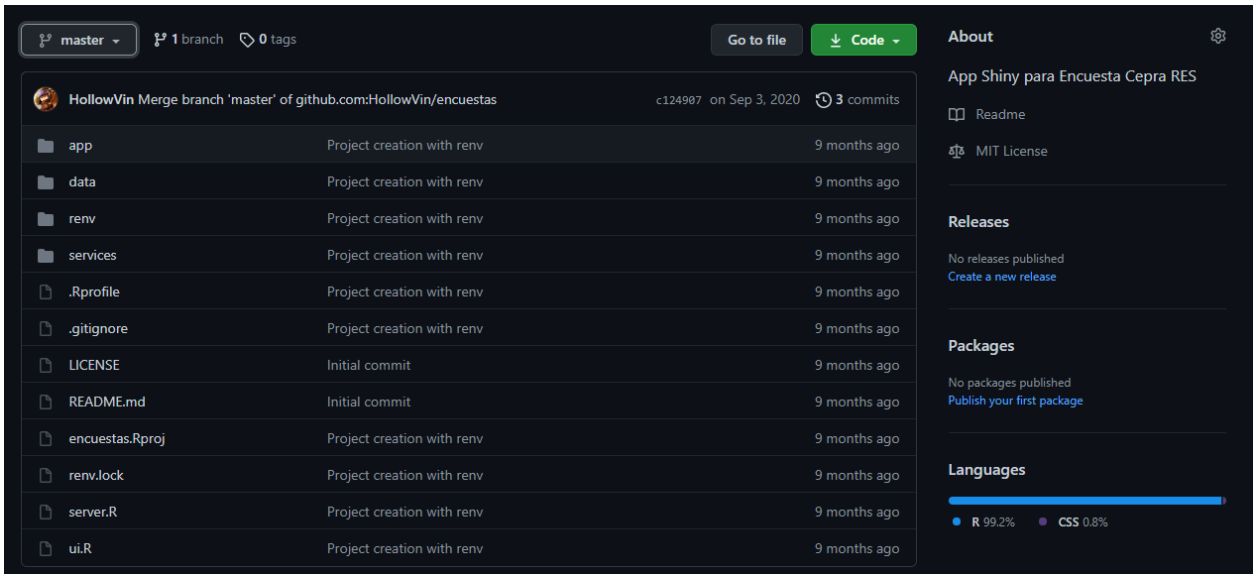


Figura 5. La vista de la estructura del proyecto en el repositorio de GitHub, https://github.com/HollowVin/encuestas

Con estos pasos realizados, se consideró que la base de la aplicación y todas las herramientas necesarias estaban disponibles y correctamente configuradas, con lo cual se continuó el desarrollo del proyecto con la escritura de código para empezar a dar funcionamiento a la aplicación en Shiny.

12. Programación de la aplicación Shiny

La parte de programación de la aplicación involucró el realizar un proceso de presentación a las partes interesadas y recibir retroalimentación a medida que el desarrollo avanzaba. El beneficio de realizar un desarrollo que incluya una retroalimentación de las partes interesadas es acercar lo más rápido posible la funcionalidad de la aplicación a las necesidades que las partes interesadas tienen y qué desean poder realizar y obtener como resultado de la aplicación. Con esto, se cumple el objetivo planteado inicialmente de utilizar un proceso de desarrollo iterativo e incremental.

12.1. Funcionamiento de Shiny

Para explicar algunos de los pasos realizados durante el desarrollo de la app, es importante primero describir cómo funcionan los componentes internos de Shiny.

Las páginas web dinámicas son aquellas que cambian su estado e información que se observa en pantalla ante la acción de un usuario, y son por ende el tipo de aplicaciones web más populares hoy en día. Cualquier funcionalidad que requiera de interacción por parte de un usuario sin necesidad de redireccionar o cambiar toda la página web se beneficia del uso de usar características dinámicas.

Shiny, por su naturaleza de estar basado en el lenguaje R para propósitos estadísticos, también ofrece páginas web dinámicas. La manera en que Shiny provee la capacidad de crear páginas web dinámicas está basada en dos tipos de componentes que existen dentro del framework de Shiny: los componentes reactantes y los componentes reactivos (RStudio Team, 2017).

Para ejemplificar lo que es un componente reactante y uno reactivo, considérese una página web simple donde se muestra un campo de texto y un gráfico de barras que representa una serie de datos precargados. El campo de texto se utiliza para que el usuario pueda ingresar un número, y el gráfico de barras cambie, en consecuencia, el número de barras que utiliza para representar los datos. En este caso, el componente reactante es el campo de texto, ya que es el que causa que se dé una reacción en la página web—es decir, cambiar el campo de texto hace que componentes reactivos en la aplicación cambien para reflejar la información ingresada en dicho campo. Por su parte, entonces, el gráfico de barras es un componente reactivo, ya que responde a cambios en un reactante y altera su presentación o funcionamiento dependiendo de cómo haya sido configurado

a responder a cambios en distintos componentes reactantes. Un reactante podría, en teoría, afectar a varios componentes reactivos a la vez; así mismo, un componente reactivo podría ser afectado por cambios por parte de varios componentes reactivos. La comprensión y el correcto de este tipo de componentes fue de suma importancia para desarrollar la aplicación de este proyecto.

12.2. Creación de los archivos controlador y vista

Para la visualización de ciertos componentes web en la aplicación, se tiene el directorio **app/views** y los archivos y código respectivo que éste contenga. Para que estos componentes logren ser reactivos a cambios por parte del usuario, ya sea al presionar un botón, ingresar o cambiar un valor en un campo de texto, o incluso subir un archivo, se tiene el directorio **app/controllers**. Estos dos nombres se eligieron porque la estructura de la aplicación intenta atenerse a los estándares definidos por la arquitectura Modelo-Vista-Controlador, abreviado como MVC. MVC define, a grandes rasgos, tres grandes capas claramente delimitadas de componentes de software: el Modelo, que representa colecciones o estructuras de datos; la Vista, que define los componentes de interfaz de usuario, y el Controlador, que define los componentes que ejecutan lógica de servidor y preparan los modelos y otros componentes lógicos para su visualización en la Vista (Microsoft, s/f).

Para el funcionamiento de esta aplicación, se consideró que no harían falta el uso de componentes de modelo. Los modelos, comúnmente, se utilizan para representar estructuras que se almacenen en una base de datos. Por ejemplo, en una aplicación web donde se tenga un usuario que puede autenticarse, el modelo Usuario se usa para representar cómo se almacena un usuario en una base de datos, y cómo se accede a los atributos de un usuario a través de código. La aplicación no utilizará una base de datos ni tiene necesidad de representar estructuras de datos lógicas a través de código, por lo que se prescindió de crear una carpeta dentro del directorio **app** para estos componentes.

Los componentes de vistas y controladores, sin embargo, sí tienen relación directa con cómo debe funcionar la aplicación: debe tenerse código que ejecute lógica de servidor (es decir, controladores), y este código debe encargarse de preparar o actualizar las vistas correspondientes. De ahí los directorios **app/controllers** y **app/views**.

Debido a que Shiny no permite, nativamente, tener una aplicación con diferentes rutas (por ejemplo, las rutas http://example.com/index y http://example.com/user representan separaciones lógicas del código de una aplicación), se decidió realizar una separación de los distintos aspectos de la aplicación, dentro de una sola página web, por medio de pestañas que muestren su contenido dependiendo de en cuál se haya hecho click. De esta forma, el usuario no visualiza todos los componentes de la aplicación en una sola página, sino que puede escoger cuál aspecto de la aplicación utilizar en cualquier momento. Las pestañas que se definieron para la aplicación fueron:

- La *pestaña de inicio*, en la cual se da una pequeña descripción sobre la aplicación y se proveen botones o enlaces al resto de pestañas de la aplicación para fácil acceso.

- La *pestaña de manejo de datos*, donde se pueden visualizar los datos que se encuentren cargados actualmente (por defecto, la aplicación cargará los datos de la encuesta CEPRA-RES sobre movilización escolar) o subir nuevos a través de archivos **.csv** o **.xlsx** (Excel).

- La *pestaña de gráficos de barras o histogramas*, donde se podrá controlar las variables, apariencias y otros componentes para realizar gráficos de barras o histogramas que representen los datos que hayan sido cargados en la anterior pestaña.

- La *pestaña de gráficos de mosaico*, donde se pueden seleccionar las dos variables que representen la asociación entre las mismas a través de la visualización del gráfico de mosaico

- La *pestaña acerca del proyecto*, donde se detalla que el proyecto es un trabajo de titulación para la Universidad del Azuay y la carrera de Ingeniería de Sistemas y Telemática

- La *pestaña acerca de CEPRA-RES*, donde se detalla cómo se dio la recolección de los datos de movilización de la encuesta CEPRA-RES, incluyendo los contribuyentes a tal proyecto.

La separación del código entre distintos archivos para las vistas es entonces sencilla: se separó el código de interfaz de cada pestaña en su propio archivo. De esta forma, realizar modificaciones a una pestaña es sencillo porque involucra realizar modificaciones a archivos con menos código que si todo estuviese junto.

La separación de código para los controladores es un poco más compleja. Esto es debido a que, por ejemplo, la reactividad entre las pestañas de datos, gráficos de barras y gráficos de mosaico son interdependientes. Es decir, si se cargan datos a través de un archivo en la pestaña de datos, el componente que muestre el gráfico en la pestaña de gráficos de barras debe también actualizarse para reflejar los nuevos datos cargados. Por ello, no se puede realizar una separación de código en un archivo para cada pestaña, porque entonces el código que cargue los datos de un archivo no realizará cambios en las pestañas de gráficos.

Para que se consiga la interdependencia entre estas pestañas, el código de la pestaña de datos, del gráfico de barras y de mosaico, deberá encontrarse todo junto dentro de un solo archivo. Otro archivo se creó para que almacene el código de la pestaña de inicio que provee botones o enlaces para cambiar a las otras pestañas de forma conveniente al iniciar la aplicación. Ya que el código de la pestaña de inicio sí es independiente del código de datos y gráficos, puede estar en un archivo separado.

12.3. Consolidación de los distintos componentes

Para lograr consolidar todo el código definido en la sección anterior, se decidió por un patrón que, adicionalmente, contribuye a la facilidad de que la app pueda ser extendida en un futuro. El patrón consiste en que cada archivo de vista (es decir, aquél dentro del directorio **app/views**) defina una sola variable que contenga todo el código que represente la vista de una pestaña. Todos los archivos que provean código de interfaz deberán terminar con el nombre **_ui.R**. El archivo que define la interfaz de toda la aplicación Shiny, llamado **ui.R** que se encuentra en el directorio raíz de la aplicación, contiene código para importar todos los archivos que terminen por **_ui.R** dentro del directorio **app/views**:

```
VIEWS_PATH <- "app/views/"
views <- list.files(path = VIEWS_PATH, pattern = "_ui.R$", recursive = TRUE)
for (view in views) {
  view_path <- paste(VIEWS_PATH, view, sep = "")
  source(view_path)
}
```

Este extracto de código importa todo el código que haya sido definido en los archivos con el patrón mencionado, incluyendo las variables definidas en ellos. Si cada archivo define una variable de interfaz para una pestaña, agregar una nueva pestaña a la aplicación para proveer nueva funcionalidad es sencillo, y solo requiere agregar la variable que represente la nueva interfaz al código de las pestañas.

En el código de interfaz definido de forma global en este mismo archivo **ui.R**, la forma de definir una pestaña visualmente (no definiendo todavía la interfaz propia que la pestaña debe mostrar) se encuentra en el siguiente extracto de código, el cual se basa en el uso de la librería **shinydashboard**:

```
sidebarMenu(
  id = "tabs",
  menuItem("Inicio", tabName = "home", icon = icon("home")),
  menuItem("Datos", tabName = "data", icon = icon("database")),
  menuItem("Gráfico Barras", tabName = "bars", icon = icon("chart-bar")),
  menuItem("Gráfico Mosaico", tabName = "mosaic", icon = icon("table")),
  menuItem("Acerca del proyecto", tabName = "project", icon = icon("book")),
  menuItem("Acerca De CEPRA-RES", tabName = "cepra", icon = icon("id-card"))
)
```

Cada pestaña visual se define a través de una llamada al método **menuItem()**, la cual recibe como argumento el nombre de la pestaña. También recibe, como argumentos nombrados, el nombre único interno de la pestaña que se utiliza para luego atarla al código de interfaz por medio del nombre **tabName**, y, opcionalmente, el argumento **icon** que define el icono que acompaña al nombre de la pestaña. Shiny ya posee un método que se utiliza con este argumento, llamado **icon()**, al cual se le envía el nombre del icono como se provee por las librerías de iconos de uso libre llamadas Font Awesome Free y Glyphicons.

Para atar, entonces, esta pestaña que está definida solo de forma visual, al código que representa la interfaz que la pestaña debe mostrar cuando sea seleccionada, se realiza mediante el siguiente extracto de código, también presente en el archivo **ui.R**:

```
tabItems(
```

```
  tabItem(tabName = "home", .home_ui),
  tabItem(tabName = "data", .data_ui),
  tabItem(tabName = "bars", .barplot_ui),
  tabItem(tabName = "mosaic", .mosaic_plot_ui),
  tabItem(tabName = "project", .about_project_ui),
  tabItem(tabName = "cepra", .about_cepra_ui)
)
```

La relación entre el código visual de las pestañas y el que define la interfaz a mostrar es evidente: para el caso de la primera pestaña, si ésta se definió visualmente a través del método **menuItem()** con el argumento **tabName** como **"home"**, entonces la llamada al método para atarla, **tabItem()**, también recibe un argumento **tabName** con **"home"**. El segundo argumento que se envía al método es la variable que contiene los componentes de interfaz. Para la primera pestaña entonces, en algún archivo dentro del directorio **app/views**, se espera que se haya definido una variable llamada **.home_ui** (para seguir el patrón y por conveniencia, esta variable se definió en la aplicación dentro del archivo **home_ui.R**).

Si se desea crear una nueva pestaña en la aplicación con sus propios componentes visuales, solo hace falta seguir el mismo patrón: definir una variable en un archivo dentro de **app/views** que contenga los componentes visuales de la nueva pestaña, definir la nueva pestaña en **ui.R** a través de **menuItem()**, y finalmente atarla a la variable de interfaz mediante la llamada a **tabItem()**.

La definición del código de servidor para los controladores sigue un patrón similar. Esta vez, la distinción es que el código de servidor se realiza a través de la llamada a un método que recibe argumentos llamados **input**, **output** y **session**, los cuales son usados internamente por Shiny para manejar la lógica de la aplicación. El siguiente código es cómo se importa el código en los archivos que terminen por **_controller.R** dentro de **app/controllers**, y luego llama a las funciones que hayan sido definidas dentro de dichos archivos:

```
CONTROLLERS_PATH <- "app/controllers/"
controllers <- list.files(path = CONTROLLERS_PATH, pattern = "_controller.R$",
recursive = TRUE)
for (controller in controllers) {
  controller_path <- paste(CONTROLLERS_PATH, controller, sep = "")
```

```
  source(controller_path)
}


shinyServer(function(input, output, session) {
  home_controller(input, output, session)
  graph_controller(input, output, session)
})
```

El método **home_controller()**, por ejemplo, se definió en su archivo correspondiente **home_controller.R** de la siguiente forma (nótese que el código que maneja la lógica se abrevió):

```
home_controller <- function(input, output, session) {
  # codigo de logica
}
```

Al haber definido este método en su archivo correspondiente, solo hace falta llamarlo en el archivo **server.R** como parte del cuerpo del método propio de Shiny **shinyServer()**.

13. Publicación de la aplicación en un servidor web

Luego del desarrollo de la parte visual y lógica de la aplicación web, así como de preparar el conjunto de datos que se carga por defecto para que la información esté limpia, se consideró que la aplicación estaba lista para ser puesta a prueba con el grupo de investigadores que van a darle uso a su funcionalidad para el análisis de datos de la encuesta CEPRA-RES. La mejor forma para llevar a cabo las pruebas con los investigadores sería a través de la publicación de la aplicación en algún servidor web que permita el acceso fácil a los investigadores desde cualquier parte.

Con el conocimiento de que la Universidad del Azuay posee un servidor web que se puede utilizar para publicar la aplicación y que ésta sea accesible desde internet, se decidió preparar la aplicación para que pueda ser publicada en el entorno del servidor. Las características del servidor son las siguientes:

- **SO:** Ubuntu Linux 14.04.5

- **Kernel and CPU:** Linux 3.19.0-80-generic on x86_64
- **Processor:** Intel(R) Xeon(R) Silver 4116 CPU @ 2.10GHz, 4 cores
- **RAM:** 12 gb

El servidor, además, cuenta ya con el servidor de Shiny instalado como servicio, de forma que la publicación de la aplicación consistió en compartir el repositorio público de GitHub con el administrador del servidor para que descargue el código de la aplicación al servidor y aloje el código en el directorio que el servidor de Shiny utiliza para la publicación de aplicaciones. Este proceso se dio de forma sencilla, al solo requerir que el administrador del servidor realizara una operación de clonado del repositorio del proyecto de encuestas en GitHub, y por separado agregara el archivo Excel con los datos concatenados, generados durante el paso de la limpieza de datos (este paso hizo falta realizarlo por separado porque, como se describió en la etapa de publicación del código en el repositorio público, la carpeta **data** y sus contenidos no fueron enviados al repositorio).

Sin embargo, se consideró que puede haber personas que deseen probar el código de la aplicación de encuestas pero que no tengan ya instalado un servidor Shiny, por lo que no podrían probar la aplicación fácilmente sin antes haber realizado la instalación y configuración del servicio de Shiny. El proceso de publicación fue rápido y directo gracias a que el administrador del servidor tenía ya instalado el servicio de Shiny y tenía conocimiento sobre cómo instalar una aplicación Shiny en el servidor. No toda persona con interés en el proyecto puede tener este mismo nivel de conocimiento o facilidad de acceso a las mismas herramientas.

Como aporte adicional a este proyecto de tesis, se decidió crear una imagen con Docker para el sistema operativo Ubuntu o Debian (ya que Ubuntu está basado en Debian), con el propósito de dar la posibilidad de que una persona pueda descargar dicha imagen y, con la ejecución de un par de comandos en la línea de comandos de Linux, tenga la aplicación funcionando. Docker es un servicio que permite crear contenedores que pueden ser configurados y recreados para crear un entorno aislado, donde todos los componentes que un servicio requiere se encuentren disponibles para que dicho servicio funcione de forma confiable. Una imagen creada en Docker replica todo el entorno que se configure en dicha imagen, y esta imagen puede luego compartirse con otras personas, quienes, para poder hacer uso de la imagen, solo necesitan instalar el servicio Docker y

luego cargar la imagen. Una imagen en Docker representa las acciones que debe realizar Docker para replicar un entorno, mientras que un contenedor es la imagen ya en ejecución (Docker Inc., s/f). Tener una imagen Docker también permite aislar el contenido de forma que, si ya no se desea tener a la aplicación de encuestas o el servicio de Shiny en la máquina con Ubuntu/Debian, solo se debe eliminar el contenedor de Docker, y todo el espacio que ocupaba será liberado.

Adicionalmente, el usar Ubuntu facilita su uso sobre una máquina que tenga el sistema operativo Windows 10. Windows 10 posee un servicio llamado Windows Subsystem for Linux, o WSL, que permite tener una instalación de una distribución de Linux (la más popular siendo Ubuntu) para ser utilizada dentro de Windows 10 mismo, como una máquina aislada pero más directamente integrada sobre Windows que utilizar una máquina virtual (Microsoft, 2020).

El objetivo fue, entonces, crear una imagen en Docker sobre Ubuntu, de tal forma que acciones como la instalación de paquetes, la configuración de aspectos del sistema operativo, o la instalación del mismo servicio que permite publicar una aplicación Shiny, se encuentren todos almacenados bajo una imagen en Docker, y asegurarse que el entorno es replicable de tal forma que la aplicación se encuentre disponible en cualquier momento que la imagen de Docker se ejecuta sobre cualquier otra máquina con Ubuntu. El administrador del servidor de la universidad, entonces, solo debe ejecutar la imagen de Docker para que todo el entorno necesario para la aplicación Shiny se replique en un contendor aislado, y cada vez que dicho contenedor se ejecute, la aplicación estaría disponible en el servidor.

13.1. Configuración del Dockerfile

El concepto sobre cómo crear un contenedor replicable en Docker se centra sobre configurar de manera apropiada un archivo llamado “Dockerfile”. En él, se configuran, en orden, las acciones que el sistema operativo debe realizar para tener todo disponible para el entorno. Si, por ejemplo, debe instalarse un paquete de Ubuntu, se configura el Dockerfile para que ejecute el comando de Ubuntu en la línea de comandos que instala el paquete. De igual forma, si el entorno requiere que un archivo se copie de la máquina que ejecuta la imagen al contenedor, se agrega al Dockerfile la línea que realiza el copiado del archivo (Docker, s/f).

Por motivos de conveniencia y disponibilidad, el Dockerfile no se configuró desde cero para el proyecto. Más bien, se decidió tomar como punto de partida una imagen de Docker ya disponible

públicamente a través de un repositorio en GitHub. Esta imagen ya provee un Dockerfile y todos los recursos adicionales necesarios para crear y configurar el entorno que permita tener un servidor de Shiny listo para la publicación de aplicaciones Shiny, en tanto el entorno base sea Ubuntu (o Debian, ya que Ubuntu está basado en Debian) (The Rocker Project, 2020). El nombre del Dockerfile publicado se llama "shiny" y está publicado por The Rocker Project, un grupo de desarrolladores que proveen imágenes de Docker para entornos del lenguaje R. The Rocker Project describe en su sitio web que "Los Dockerfiles de Rocker están licenciados bajo la GPL 2 o mayor" (Boettiger et al., s/f).

GPL 2 se refiere a la Licencia de Público General GNU versión 2 (GNU General Public License), la cual fue creada por la Free Software Foundation, y la cual, a grandes rasgos, permite la redistribución y modificación de los programas originales en tanto esta redistribución no se haga bajo una política de código cerrado (Free Software Foundation, 2017). En otras palabras, en tanto un programa publicado bajo la GPL 2 sea modificado, y la versión modificada esté publicada también bajo la licencia GPL 2, y además la versión modificada no se distribuya a otras personas (ya sea que se les distribuye de forma pagada o gratis) sin que esas personas tengan acceso al código fuente, no se considera que la licencia haya sido violada.

Considerando entonces que esta imagen de Docker se modificará para que soporte la aplicación realizada en este proyecto, y dicha imagen se encontrará también disponible en GitHub bajo el mismo tipo de licencia, no existe ninguna violación de la licencia del proyecto original, y se pueden realizar modificaciones apropiadas para los propósitos de este proyecto de forma libre.

13.2. Modificaciones al Dockerfile

Al probar la imagen "shiny" the The Rocker Project, se comprueba bajo una máquina con Ubuntu Linux 20.04 que el servicio de Shiny y todo su entorno se inicializa y configura correctamente para permitir la publicación de aplicaciones Shiny simples.

Sin embargo, la publicación de la aplicación desarrollada en este proyecto no funcionó con este contenedor de Docker, debido en menor parte a un paquete de Ubuntu que hace falta instalar para la aplicación, pero en gran parte al uso de **renv**, el manejador de las librerías de un proyecto de Shiny. **renv** requiere que los paquetes necesarios para el funcionamiento de la aplicación se encuentren instalados. El equipo desarrollador de **renv** provee una guía sobre cómo publicar

aplicaciones soportadas por **renv** con Docker, y sugiere que la mejor forma de trabajar con las librerías de un proyecto con **renv** es exponer un directorio de la máquina física a un directorio del contenedor. El propósito de esto es hacer que las librerías de **renv** no se encuentren dentro del contenedor donde están sujetas a poder perderse al ejecutar o detener la ejecución de la imagen de Docker (Ushey, s/f).

Por tanto, los cambios que se realizaron a la imagen Docker original fueron los siguientes:

- Agregar el paquete de Linux que falta, llamado **libssl-dev**, a la lista de paquetes que Docker debe instalar para inicializar el contenedor

- Instalar la versión de R que se utilizó para desarrollar la aplicación. La imagen original instala R 3.6.3, pero la aplicación fue desarrollada con R 4.0.2. Para evitar cualquier problema de compatibilidad, se consideró ideal instalar la misma versión de R que la del entorno en la que se desarrolló la aplicación.

- Realizar un copiado de un archivo de configuración del servidor Shiny desde el proyecto de Docker al sistema de archivos del contenedor. Por defecto, este archivo de configuración, cuando se instala Shiny, está configurado para que se puedan publicar varias aplicaciones en un directorio, y se tenga acceso a cada una a través de un índice. Debido a que el propósito de la nueva imagen de Docker es que publique la aplicación de encuestas, se cambia este atributo para que solo publique una aplicación en el directorio raíz del servidor.

- Por motivos del anterior punto, también se agregó un directorio donde se encuentren los archivos de la aplicación de encuestas, y que Dockerfile, durante la inicialización, copie estos archivos al directorio donde debe estar la aplicación que debe publicar.

- Realizar la instalación de la librería **renv**, de forma que la aplicación pueda volver a instalar los paquetes necesarios para su funcionamiento, o utilizarlos si ya se encuentran disponibles.

14. Pruebas de la aplicación con los investigadores

Con la aplicación disponible en el servidor en internet, se definió una fecha para realizar una reunión con motivos de realizar una evaluación y retroalimentación del funcionamiento de la aplicación. La recolección de retroalimentación y nivel de satisfacción de los investigadores se realizó a través de una encuesta aplicada a los investigadores en una sesión en vivo. La medición de satisfacción con la aplicación, por motivos de tiempo y extensión del proyecto, no siguió un proceso riguroso de calidad de software, sino que se realizó para tener una idea general sobre su utilidad y aporte a las personas del proyecto CEPRA-RES.

La evaluación estuvo conformada por el autor del proyecto, la directora de tesis Agrim. Daniela Ballari, y un grupo de investigadores, en sí conformado por 6 personas en total, con diversos conocimientos de R y sobre el proyecto CEPRA-RES. Es decir, algunos investigadores tenían experiencia con escribir código en R y estaban involucrados con el proyecto CEPRA-RES; otros tenían solo experiencia con R, o solo experiencia con el proyecto CEPRA-RES; y el resto eran investigadores que no tenían conocimiento de R ni del proyecto. El propósito de tener esta diversidad entre las personas que realizan la evaluación fue determinar que la aplicación fuera útil tanto para gente que ya sabe de R, así como aquellos que no saben de él, además que para gente que conozca los datos recolectados para el proyecto CEPRA-RES o quienes no conocieran los datos y los utilizaran por primera vez.

La fecha para la evaluación fue establecida el día 3 de junio de 2021, a las 15:00. La reunión se realizó a través de la plataforma Zoom por motivos de la pandemia por COVID-19, así como por conveniencia de las personas involucradas. Dos personas no pudieron estar presentes en la reunión, pero se les envió el enlace web a la aplicación y el enlace al formulario en Google Forms para que realizaran una evaluación y respondieran al formulario cuando tuviesen la disponibilidad.

La reunión comenzó con una introducción sobre el proyecto, y una presentación sobre cómo funciona la aplicación, de forma que los evaluadores tuviesen conocimiento preliminar sobre qué esperar sobre la aplicación. Luego de esto, se dio un tiempo para que los evaluadores utilizaran la aplicación en vivo, ya sea usando los datos precargados de CEPRA-RES o sus propios datos, y generaran gráficos a su gusto, con motivo de que al final, la retroalimentación se basara en conocer qué tan fácil o difícil se hizo el manejo y generación de gráficos en la aplicación. El tiempo que tardó el probar la aplicación por parte de los evaluadores fue de aproximadamente 1 hora. La reunión en total duró hasta aproximadamente las 16:20 del mismo día.

14.1. Preguntas del formulario

Las preguntas que los evaluadores contestaron al proveer retroalimentación luego de su evaluación fueron las siguientes:

- ¿Cuál es su nivel de conocimiento del lenguaje de programación R?, con las posibles respuestas de:
    - Lo conozco bien y lo utilizo con cierta frecuencia
    - Lo he utilizado pero no tengo mucha experiencia con el lenguaje
    - No tengo ninguna experiencia con el lenguaje R

Si el usuario respondía a la primera pregunta con la primera o segunda respuesta, se pasa a la segunda sección, donde se realiza la pregunta de:

- ¿Considera que la generación de los gráficos en la aplicación fue más rápida e intuitiva que escribir código en R para generarlos?, con las posibles respuestas de:
    - No, la generación de gráficos habría sido más o igual de rápida e intuitiva al escribir código en R
    - La generación fue más rápida, pero no considero que fuera más intuitiva
    - La generación fue más intuitiva, pero habría tardado lo mismo al escribir el código
    - Sí, la generación fue más rápida y más intuitiva que escribir código en R

Cuando el usuario respondía a la pregunta anterior, se pasaba a la tercera sección del formulario. Se pasaba también a la tercera sección del formulario al responder con “No tengo ninguna experiencia con el lenguaje R” a la primera pregunta. La tercera sección del formulario constaba de las siguientes preguntas:

- **¿Qué tan fácil le resultó navegar la aplicación para encontrar las distintas funcionalidades que ofrece?**, calificada en una escala del 1 al 5, donde 1 representaba que fue muy difícil y poco intuitivo, y 5 representaba que fue muy fácil e intuitivo.

- **¿Qué tan fácil le resultó previsualizar los datos cargados en la aplicación?**, calificada en una escala del 1 al 5, usando el mismo criterio que la primera pregunta de la sección

- ¿Qué tan fácil le resultó generar gráficos de barras o histogramas, incluyendo la generación de facetas para análisis de múltiples variables?, calificada en una escala del 1 al 5, usando el mismo criterio que la primera pregunta de la sección

- **¿Qué tan fácil le resultó generar e interpretar los gráficos de mosaico en la aplicación?**, calificada en una escala del 1 al 5, usando el mismo criterio que la primera pregunta de la sección

- Una última pregunta abierta con un campo de texto de libre escritura para el usuario, que preguntaba: **¿Tiene observaciones o recomendaciones sobre cómo mejorar la aplicación?**

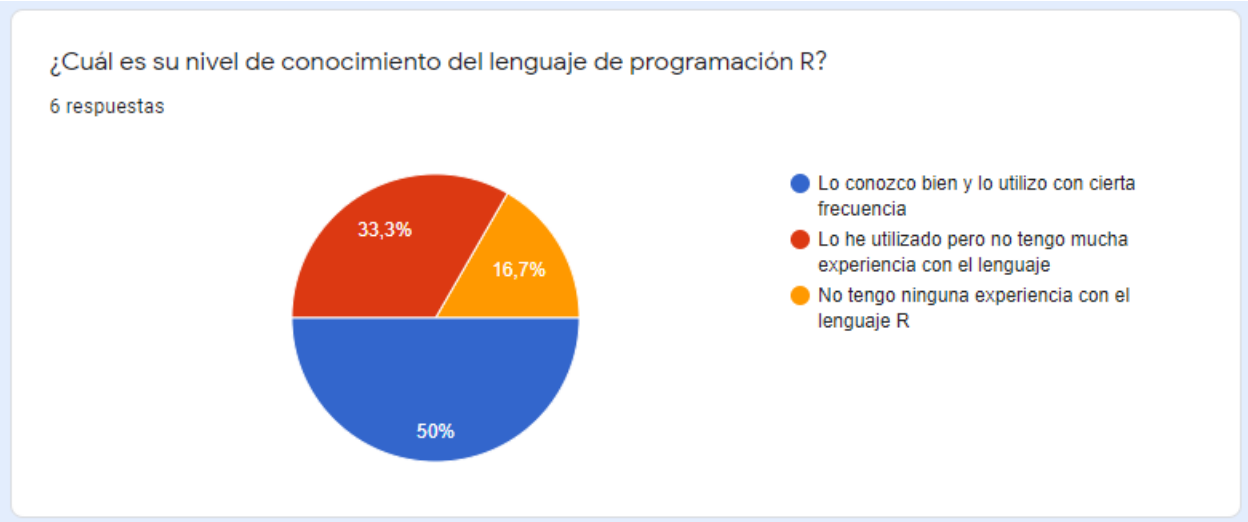
Estas preguntas fueron contestadas por los investigadores, y sus resultados se detallan en la siguiente sección, Resultados de la evaluación.

Resultado final

15. Resultados de la evaluación

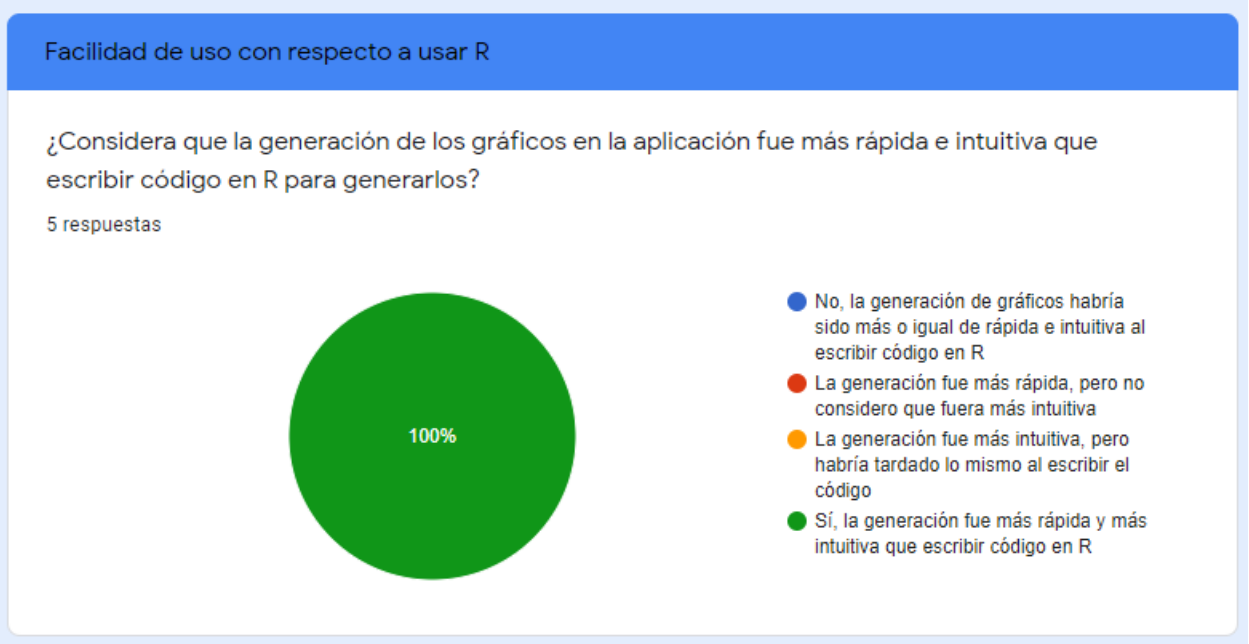
Los resultados de la evaluación realizada a través de Google Forms se detallan antes que los resultados de la aplicación misma, ya que la evaluación dio como resultado algunas sugerencias y retroalimentación por parte de los investigadores que concluyó en que se realicen algunos cambios menores a la aplicación, por lo que es consecuente primero detallar estas sugerencias, y luego detallar cómo esos cambios se ven y funcionan en la aplicación final.

Para la primera pregunta, **¿Cuál es su nivel de conocimiento del lenguaje de programación R?**, se obtuvieron las siguientes respuestas:



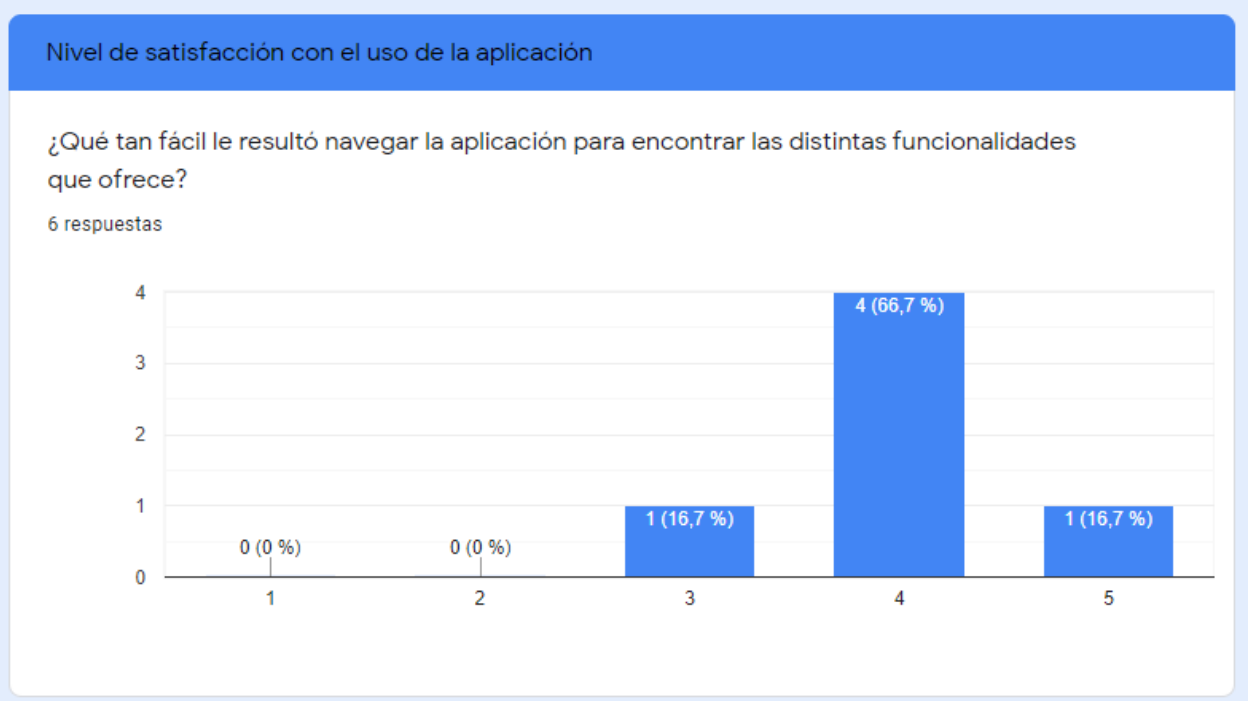
Esto significa que 3 de los 6 evaluadores tienen un uso frecuente de R. En general, 5 de los 6 evaluadores lo han utilizado alguna vez.

Para la segunda pregunta, realizada solo a los 5 evaluadores que respondieron que han utilizado el lenguaje R alguna vez, las respuestas fueron las siguientes:



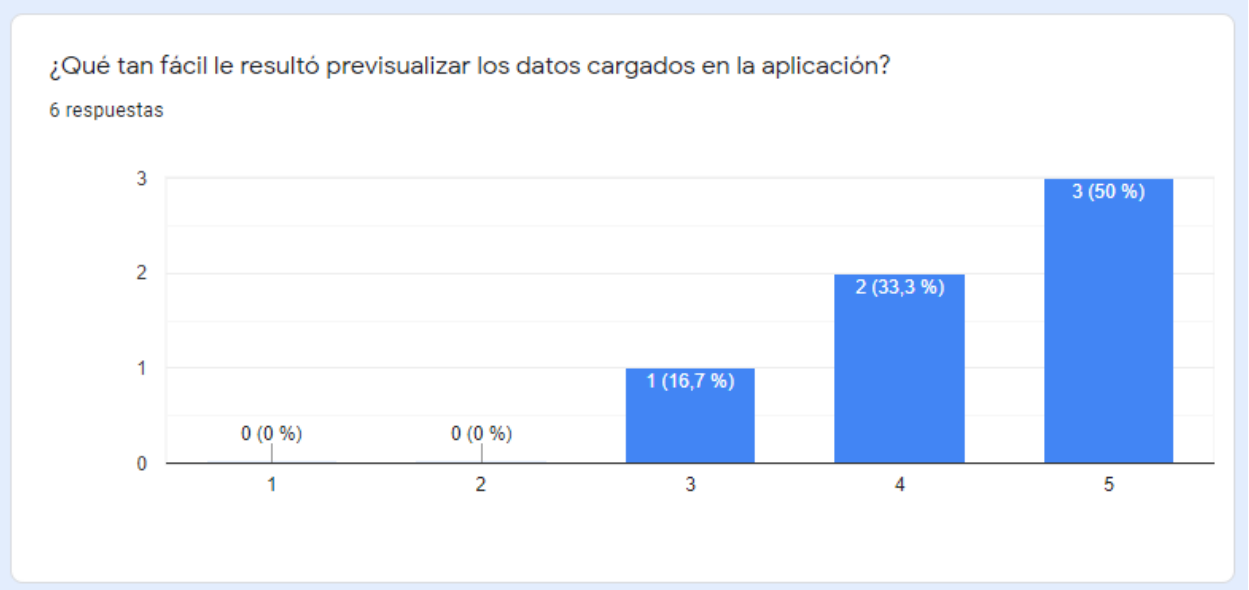
Es decir, todos los investigadores que utilizaron R consideran que la generación de gráficos fue más rápida e intuitiva que utilizar R directamente, por lo que se concluye que la aplicación como forma de realizar una exploración de datos es más ágil y sencilla de utilizar que hacerlo a través de código en R.

La tercera pregunta, donde ésta y todas las siguientes preguntas fueron aplicadas nuevamente a todos los 6 evaluadores (es decir, sin importar su experiencia con R), mide la facilidad de navegación y de encontrar las funcionalidades de la aplicación, y tuvo las siguientes respuestas:



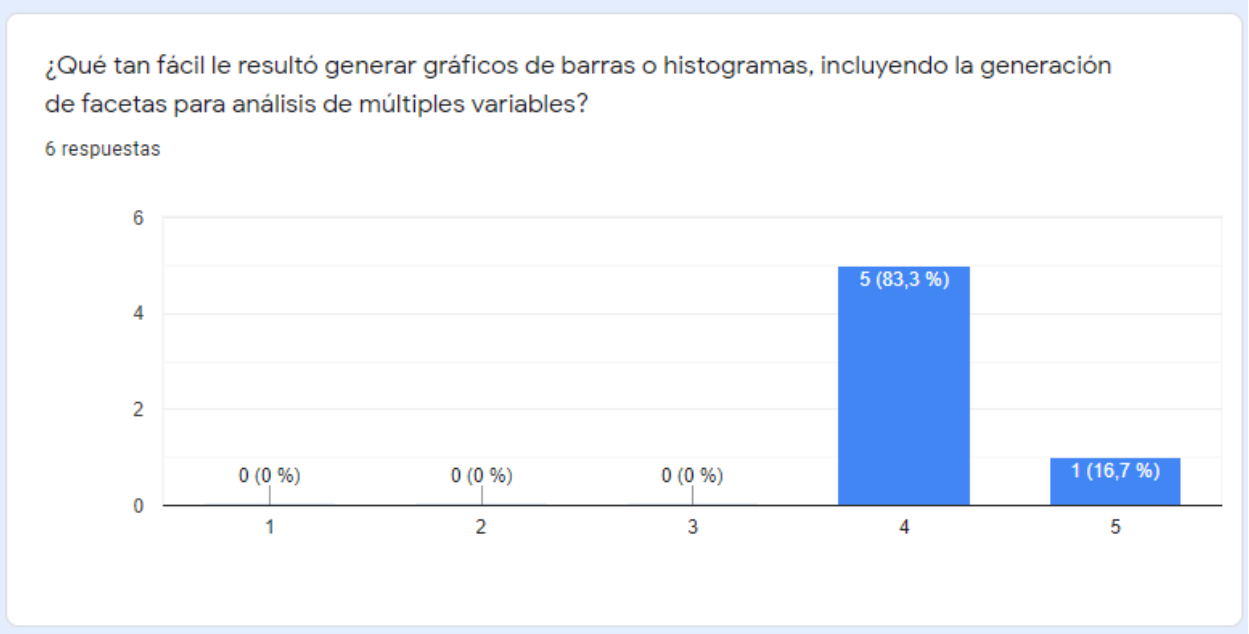
Se encuentra así un nivel de satisfacción de 4/5 para cuatro de los seis evaluadores. Uno de los evaluadores restantes respondió con un nivel de facilidad medio (3/5), y otro respondió con un nivel de facilidad completo (5/5). En general, se puede decir que la facilidad para navegar la aplicación es suficiente dada la extensión y alcance del proyecto.

La siguiente pregunta, que mide la facilidad en la previsualización de los datos subidos en la aplicación, tuvo las siguientes respuestas:



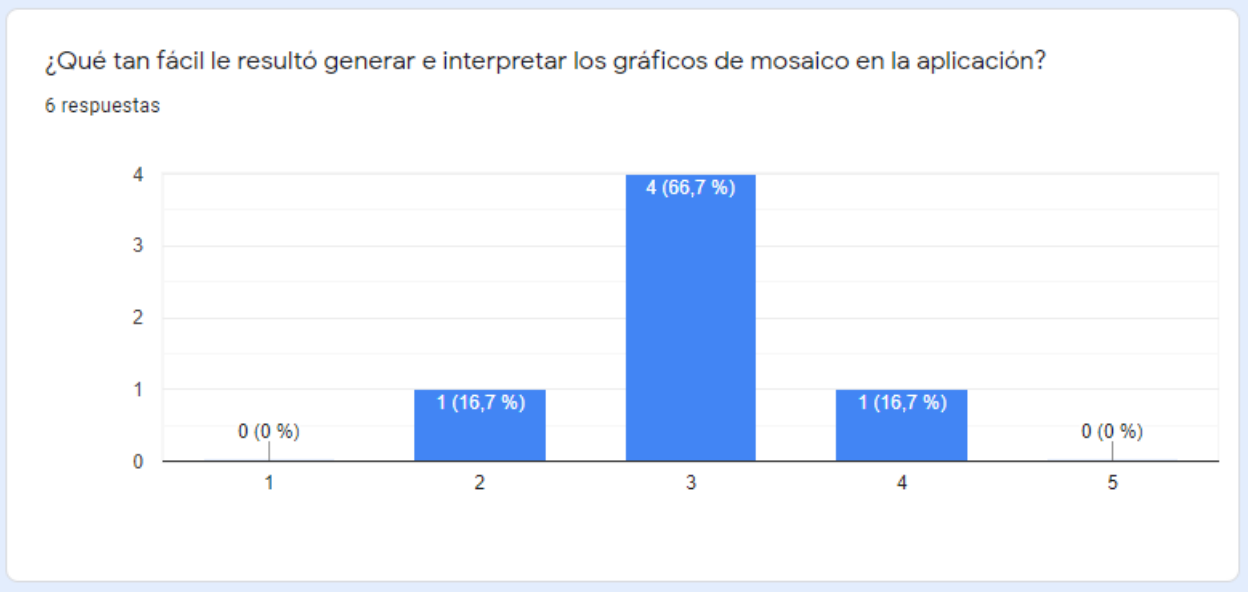
Se encuentra un nivel general alto de facilidad para previsualizar dichos datos. Esto puede atribuirse a que los datos se muestran en forma de tabla el momento de ser subidos a la aplicación, por lo que realizar una previsualización de cómo se ven los datos es intuitiva ya que se asemeja mucho a visualizar los datos en un programa de hoja de cálculos. La mitad de los evaluadores reportó un nivel de facilidad máximo, con otros dos reportando una facilidad medio-alta y uno reportando una facilidad media.

La siguiente pregunta, realizada para medir el nivel de facilidad en generar gráficos de barras, tuvo las siguientes respuestas:



Durante la evaluación, surgieron preguntas con respecto a la interpretación de los gráficos de barras, especialmente cuando los evaluadores realizaron pruebas con múltiples variables. Por esta razón, es comprensible que, a esta pregunta, la mayoría de los evaluadores no reportara una facilidad muy alta en generar y comprender los gráficos, sino de un nivel menor.

Finalmente, la última pregunta para evaluación de la aplicación fue la siguiente:



Los resultados se explican ya que, el momento de la reunión, muchos de los evaluadores mostraron no estar familiarizados con los gráficos de mosaico, lo cual es comprensible, ya que son gráficos que no son muy utilizados. En consecuencia, se dieron varias preguntas el momento de la reunión sobre cómo interpretar el gráfico debido a esta falta de familiarización, incluso dado que el método para la realización de gráficos de mosaico consiste en elegir dos columnas de los datos cargados, un proceso más simple que el utilizado para los gráficos de barras o histogramas, además de que estos gráficos están acompañados de una explicación detallada y un ejemplo práctico utilizando los datos de CEPRA-RES. Por ende, el nivel de facilidad medio que fue reportado en promedio para este tipo de gráficos puede atribuirse más a la falta de familiarización con los gráficos de mosaico y cómo interpretarlos, mas no con el funcionamiento de la aplicación sobre cómo generarlos.

15.1. Observaciones realizadas por los investigadores

Una última pregunta abierta se encontraba disponible en el formulario de Google Forms, donde los evaluadores podían hacer comentarios y sugerencias propios sobre cómo podría mejorar la aplicación. Además de eso, durante la evaluación, se recogieron otras sugerencias que se mencionaban por parte de los evaluadores pero que quizá lo olviden el momento de llenar la encuesta. La recopilación de las sugerencias fueron las siguientes, donde un asterisco luego de la

sugerencia indica que ésta no fue implementada en el resultado final de la aplicación, ya sea por motivos de tiempo o de que la aplicación no tiene tal alcance para este proyecto:

- Agregar una indicación sobre cómo se puede guardar un gráfico realizado como una imagen al computador.
- Agregar una caja de selección para el filtrado de variables, donde la caja contenga todos los valores que la columna contiene y no tenga que escribirse en un campo de texto el valor deseado.
- Cambiar la indicación sobre el orden que debe seguirse para realizar gráficos de barras o histogramas con múltiples variables a la parte superior de la pestaña para que dicha indicación sea más fácil de notar y leer
- Especificar que la variable en la pestaña de Básico en gráficos de barras e histogramas es la variable principal
- Agregar una pestaña de preguntas frecuentes
- Agregar una explicación por medio de un enlace externo sobre qué son y cómo se interpretan los gráficos de mosaico
- Cambiar la visualización de datos entre valores absolutos y valores porcentuales*
- Utilizar una rampa de colores para las paletas de colores*
- Dar más espacio en el gráfico de mosaico al mover la escala de Pearson en dicho gráfico a la parte inferior* (no se conoce una forma de hacer un cambio de este tipo, ya que el gráfico de mosaico es generado por una librería que no permite control sobre todas las formas en la que el gráfico se genera)
- Realizar los gráficos de barras e histogramas con el mismo ancho de barras para todas las condiciones, porque existen situaciones donde los anchos de barras cambian entre gráficos sin facetas y gráficos con facetas*

- Cambiar el tamaño de las etiquetas para que se ajusten dinámicamente dependiendo del contenido*

- Dar la posibilidad de que el usuario elija el orden en que las variables se muestran en los gráficos*

- Realizar un control cuando se suben datos que contienen una o más columnas con valores de tipo fecha y que no se encuentran con un formato estandarizado*

Las sugerencias que no se hayan implementado se discuten en la sección de "Discusión sobre limitantes del proyecto actual y trabajo futuro", ya que se pueden considerar sugerencias a seguir para hacer que la aplicación sea más completa.

16. Resultado de la aplicación Shiny

Las siguientes figuras muestran el aspecto visual de cada pestaña de la aplicación, funcionando en un servidor local ya que RStudio permite ejecutar una aplicación Shiny de forma local para pruebas (Shiny Team, s/f). Después de cada captura de pantalla, se proveen descripciones sobre librerías utilizadas para conseguir el aspecto y funcionalidad que se ve en las imágenes.

16.1. Pestaña de Inicio

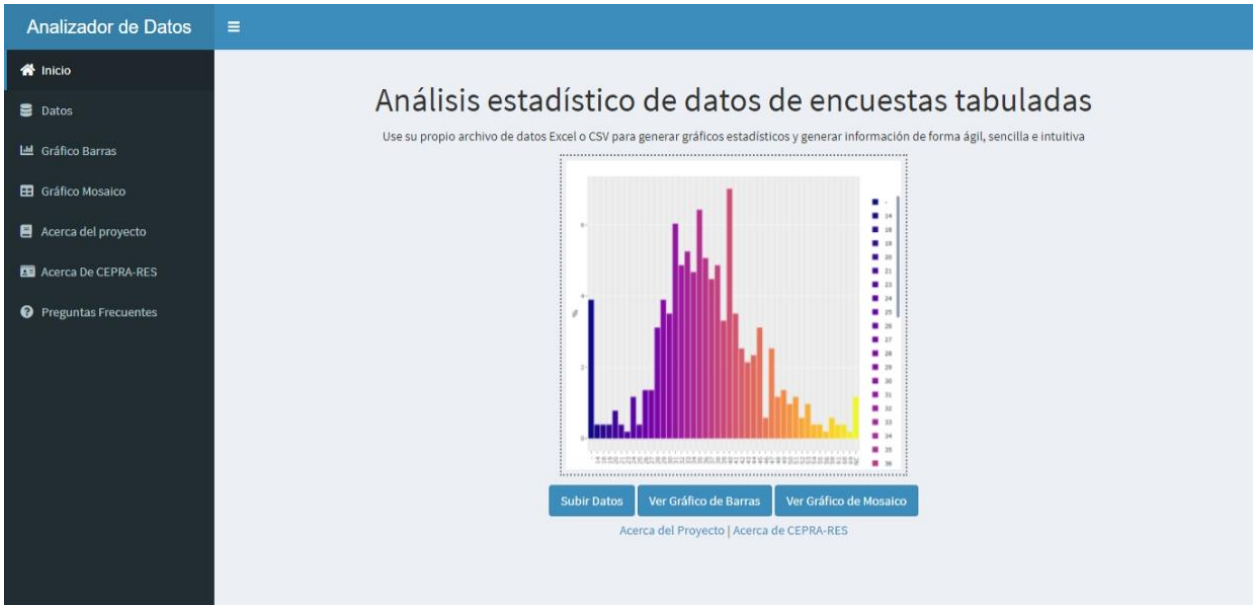


Figura 6. La vista de la aplicación web en su pestaña de inicio

La página de inicio tiene una breve descripción de la aplicación, junto a una imagen de ejemplo sobre cómo se ve un gráfico que se genera con la aplicación, que usa R por detrás.

Los tres botones y dos enlaces debajo del gráfico de ejemplo, al hacer click, dirigen al usuario a cada una de las otras pestañas en la aplicación.

La apariencia visual de la aplicación (la barra superior y su título, el menú en el lado izquierdo con sus pestañas, y los colores y otros estilos) son el resultado del uso de la librería de R **shinydashboard** (RStudio Team, 2014), la cual provee la mayoría de la temática visual que se observa en la imagen. El botón de menú hamburguesa en la parte superior  es parte de esta librería también y oculta o muestra el menú de la izquierda.

La librería también tiene opciones para distintos temas visuales, aunque el que se decidió usar es el que está configurado por defecto en la librería ya que se consideró que tiene el mejor aspecto visual. Los temas visuales disponibles se encuentran publicados en la página del proyecto de **shinydashboard**, y en el proyecto, si se desea cambiar el tema visual, solo debe agregarse la opción *skin* al método *dashboardPage()* en el archivo **ui.R** (RStudio Team, s/f). Por ejemplo:

```
dashboardPage(
  # ...,
  skin = "blue"
)
```

16.2. Pestaña de Datos

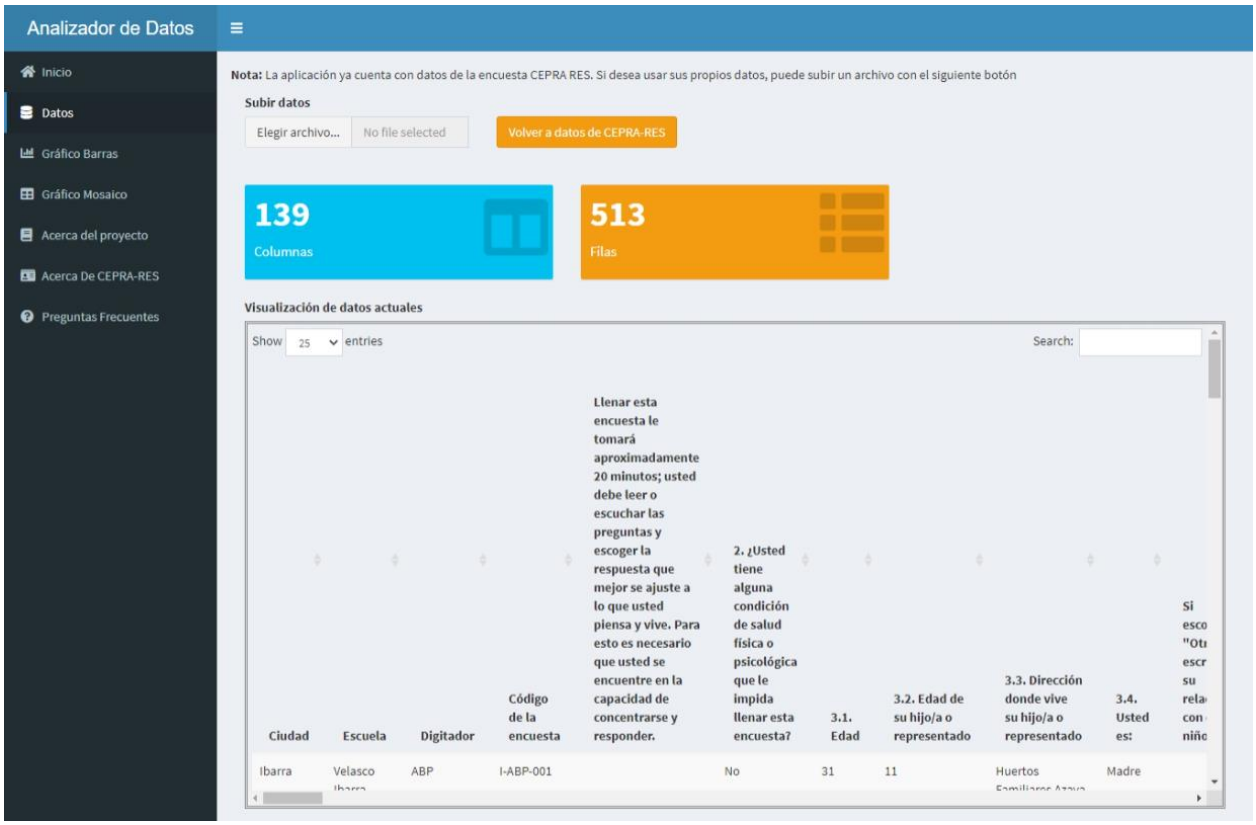


Figura 7. La vista de la aplicación web en su pestaña de manejo de datos

Esta pestaña incluye el componente para subir un archivo desde el computador por parte del usuario que desee utilizar un distinto conjunto de datos. El diálogo para subir un archivo permite archivos de tipo **.csv** (archivo separado por comas) o **.xlsx** (Excel), y muestra un error si se intenta subir un archivo no permitido. La lectura de archivos tipo .csv o .xlsx se logró con el uso de las librerías de R llamadas **readr** (Wickham & Hester, s/f) y **readxl** (Wickham & Bryan, s/f). Junto a este componente está el botón "Volver a datos de CEPRA-RES", que restaura el conjunto de datos usado a los datos por defecto de la encuesta CEPRA-RES, por si el usuario subió otro conjunto de datos, pero desea regresar al uso de los datos de CEPRA-RES sin necesidad de que tenga que refrescar la página.

Debajo de estos componentes, se tienen dos componentes que muestran, respectivamente, las columnas y filas del conjunto de datos que se está utilizando actualmente.

Por último, la tabla muestra una prevista de los datos subidos. La tabla realiza paginación para que todas las filas del conjunto de datos no se previsualicen al mismo tiempo. También posee un campo de texto debajo de cada columna, donde, si se ingresa un valor, la previsualización de los datos se filtra para que solo se muestren las filas que contienen el texto ingresado en la columna respectiva.

16.3. Pestaña de gráfico de barras

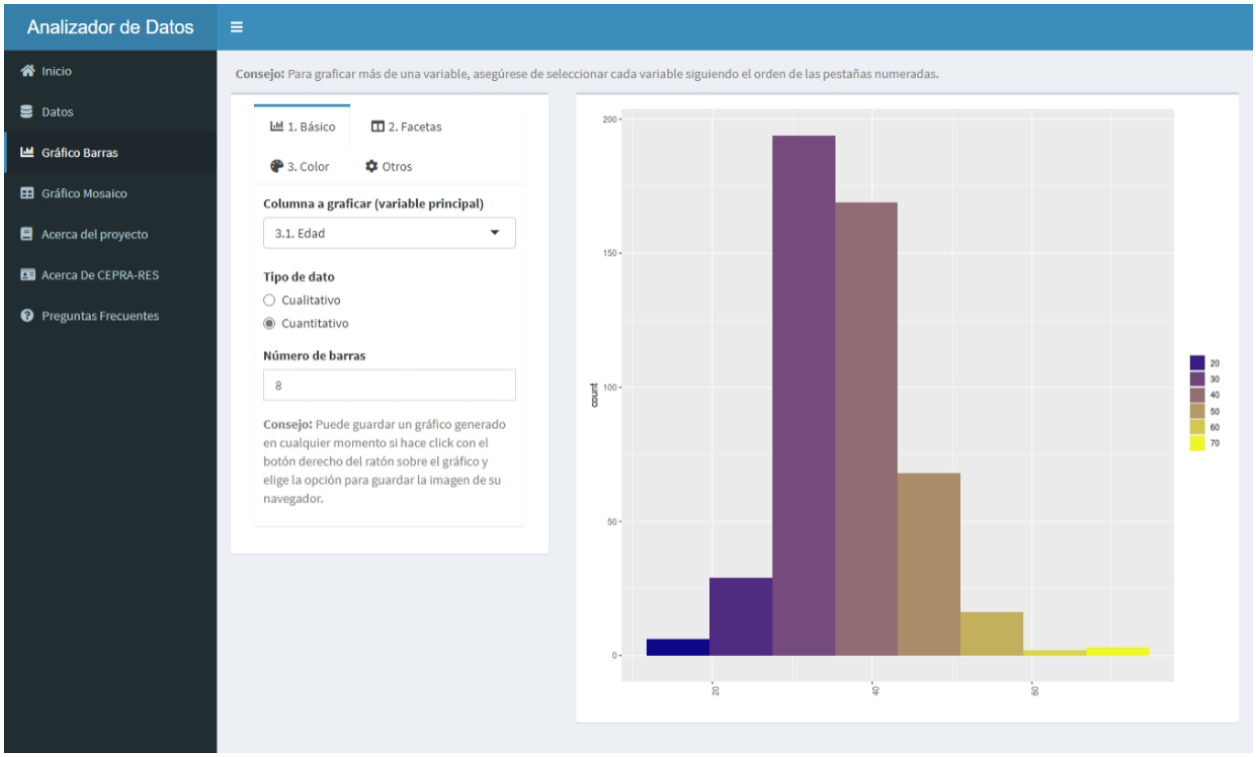


Figura 8. La vista de la aplicación web en su pestaña de gráficos de barras e histogramas

En esta pestaña se pueden generar gráficos de barras o histogramas. La librería que se usa para la generación de gráficos es **ggplot2**, la cual es altamente utilizada por desarrolladores de R por su flexibilidad (Wickham et al., 2021). Los componentes que controlan las variables y otros aspectos visuales de los gráficos se encuentran separados en pestañas por motivos de no sobrecargar la interfaz de usuario. Las pestañas se encuentran numeradas porque si se desea realizar un gráfico que incluya facetas (donde las facetas representan varias variables siendo graficadas y separadas por categorías en un solo gráfico, con el propósito de realizar análisis estadístico de variables

múltiples) se debe seguir el orden de las pestañas numeradas. Cada pestaña y sus componentes se describen a continuación.

Figura 9. El panel de control para el manejo de los gráficos de barras e histogramas. La primera pestaña del panel de control es visible.

En esta pestaña, se escoge la variable a graficar en el gráfico de barras/histograma. Para cambiar entre el uso de gráfico de barras o histograma, se debe escoger entre que la variable seleccionada es de tipo cualitativo (para el gráfico de barras) o cuantitativo (para histograma). El campo de texto de "Número de barras" se habilita y solo puede editarse cuando se escoge el tipo de dato cuantitativo, y cambia el número de barras del histograma. El campo solo acepta números positivos, por lo que, si se ingresa un número no positivo o un valor que no sea numérico, se

muestra un mensaje de error y el gráfico no se actualiza hasta que se haya ingresado un valor correcto. Por ejemplo:

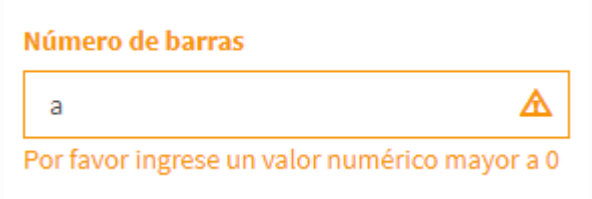


Figura 10. La visualización del error cuando se ingresa un valor inválido en el campo de Número de barras para los histogramas.

Figura 11. El panel de control para el manejo de los gráficos de barras e histogramas. La segunda pestaña del panel de control es visible.

En esta segunda pestaña de facetas, se puede cambiar el gráfico para mostrar variables adicionales en una faceta horizontal y vertical. La variable principal se separa entonces en subcategorías dentro del gráfico, donde cada subcategoría horizontal o vertical representa el rango de valores que se encuentra en el conjunto de datos para la variable respectiva. Por ende, las variables escogidas aquí deberían ser de tipo cualitativas, y de preferencia con un rango pequeño de valores, para que el gráfico y sus valores no se vuelvan excesivamente pequeños al intentar subcategorizar todas las facetas.

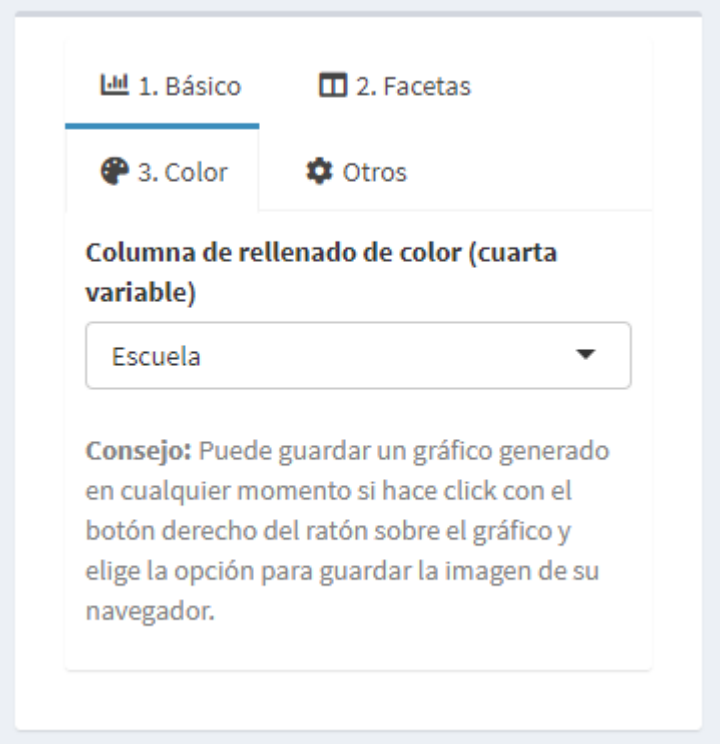


Figura 12. El panel de control para el manejo de los gráficos de barras e histogramas. La tercera pestaña del panel de control es visible.

En esta tercera pestaña, se puede escoger el rellenado que se aplica a las barras del gráfico, las cuales representan una cuarta variable (ya que el rellenado solo se puede usar si también se están usando facetas). Esto significa que, en un solo gráfico, se pueden tener hasta cuatro variables visibles en forma de gráfico: la variable principal que se representa por las frecuencias (las alturas) de las barras en el gráfico; las dos variables de facetas horizontal y vertical; y una última variable donde las barras de los gráficos se rellenan por colores dependiendo de los valores de dicha

variable de rellenado. Igual que con las facetas, la variable de rellenado debería ser de tipo cualitativo.

Figura 13. El panel de control para el manejo de los gráficos de barras e histogramas. La cuarta pestaña del panel de control es visible.

La última pestaña no es numerada, ya que no representa más uso de variables dentro del gráfico. Esta pestaña, más bien, se utiliza para dos opciones: una que representa el filtrado, donde el usuario puede escoger una columna del conjunto de datos, ingresar un valor en el campo "Valor de filtrado", y con esto los datos que se muestran en el gráfico solo representarán los que contengan el valor ingresado para la columna seleccionada. Se puede considerar ser una quinta

variable en el muestreo de datos, aunque como solo se muestra un filtrado de una columna a la vez, no es igual a los anteriores campos de variables donde todos los valores posibles se muestran en el gráfico a la vez.

Por otro lado, la opción de “Paleta de Color” permite escoger entre distintas paletas de colores a utilizarse en el gráfico, dependiendo de las preferencias del usuario. Las paletas de colores disponibles se consiguieron de la librería **viridis**, la cual provee paletas de colores que han sido probadas ser las más efectivas para distinción entre los distintos colores por categoría para el ojo humano, incluso para personas que tienen daltonismo (Rudis et al., 2021).

Si el usuario en cualquier momento generó un gráfico que desea guardar en su computador, solo necesita abrir el menú de contexto del navegador sobre la imagen (para Windows y Linux, esto es posicionar el cursor sobre la imagen y hacer click derecho) y escoger la opción del navegador para guardar la imagen.

Los gráficos de barras o histogramas se mostraban en la página inicialmente por medio de una librería llamada **plotly**, la cual provee en aplicaciones Shiny varias opciones adicionales en un gráfico como cambiar la magnificación (zoom) del gráfico o guardar la imagen con un click en un botón—sin embargo, problemas suscitados con el formato de los gráficos causó que se tuviera que desistir de seguir usando **plotly** junto con todas sus funciones adicionales mencionadas.

A continuación, se mostrará un ejemplo paso a paso para la generación de un gráfico donde se interrelacionan 5 variables diferentes:

Paso 1: selección de la paleta de colores Plasma

Se seleccionó la opción de “Plasma” en la opción de paleta de colores en la pestaña Otros, para que el gráfico se genere con este tipo de colores.

Paso 2: selección de la variable principal

Se seleccionó la variable de edad. Al ser un tipo de dato cuantitativo, se escoge ese tipo de dato de entre las opciones, y se deja el campo de Número de barras con 8 por defecto.

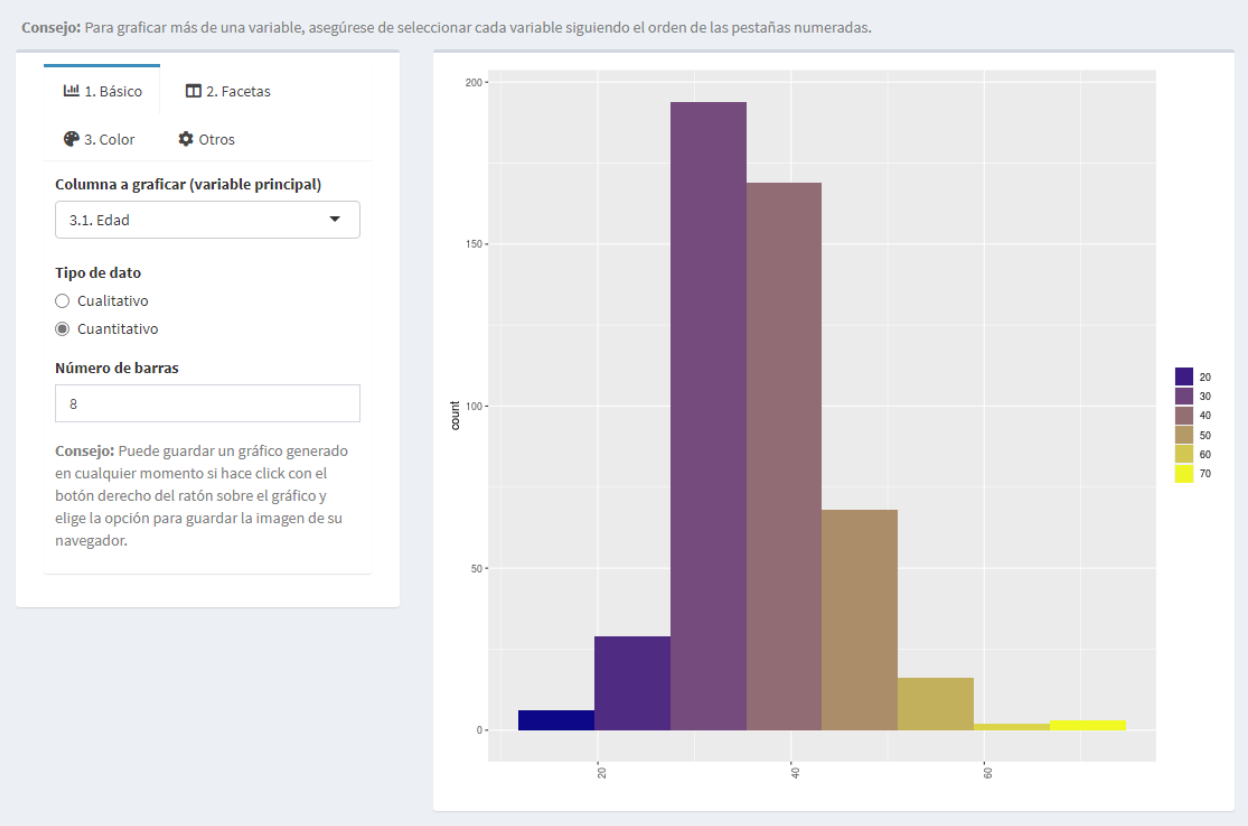


Figura 14. El ejemplo de progresión del histograma al hacer análisis de múltiples variables. En este caso con una sola variable graficada.

Paso 3: selección de la faceta horizontal

Se seleccionó, como faceta horizontal, la columna de Ciudad, para visualizar a las edades divididas entre las dos ciudades de Cuenca e Ibarra.

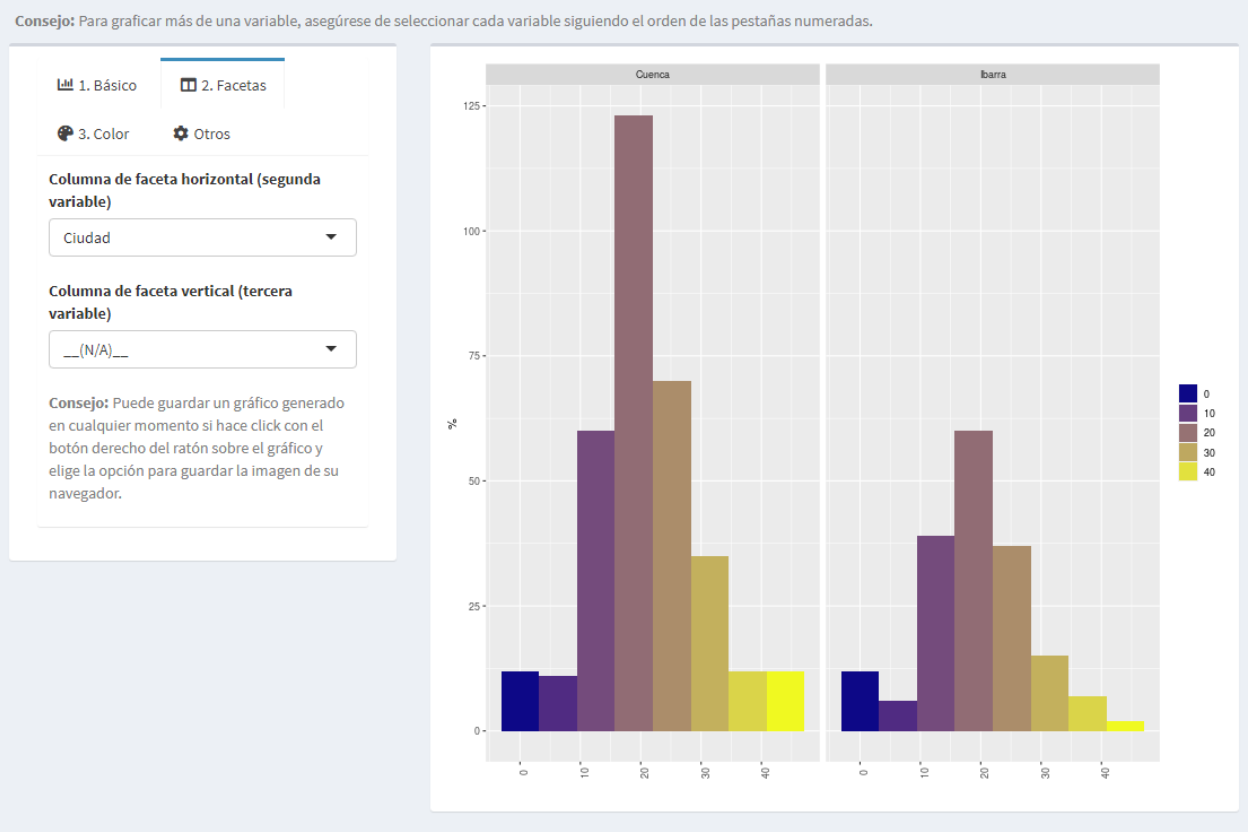


Figura 15. El ejemplo de progresión del histograma al hacer análisis de múltiples variables. En este caso, con la variable principal y la variable de faceta horizontal graficadas.

Paso 4: selección de la faceta vertical

La subdivisión de edades entre las dos ciudades se subdividió nuevamente, ahora de forma vertical para categorizar por qué tipo de pariente es con respecto al niño: padre, madre, otro, o si no hay respuesta.

Figura 16. El ejemplo de progresión del histograma al hacer análisis de múltiples variables. En este caso con la variable principal graficada, y las facetas vertical y horizontal también visualizables por sus columnas respectivas.

Paso 5: selección de la variable de color

En la tercera pestaña de Color, se escogió la columna de Escuela para observar los datos delineados con distintos colores. Los colores se pueden distinguir dependiendo de qué tipo de Escuela representan por la leyenda que se encuentra en la parte derecha del gráfico.

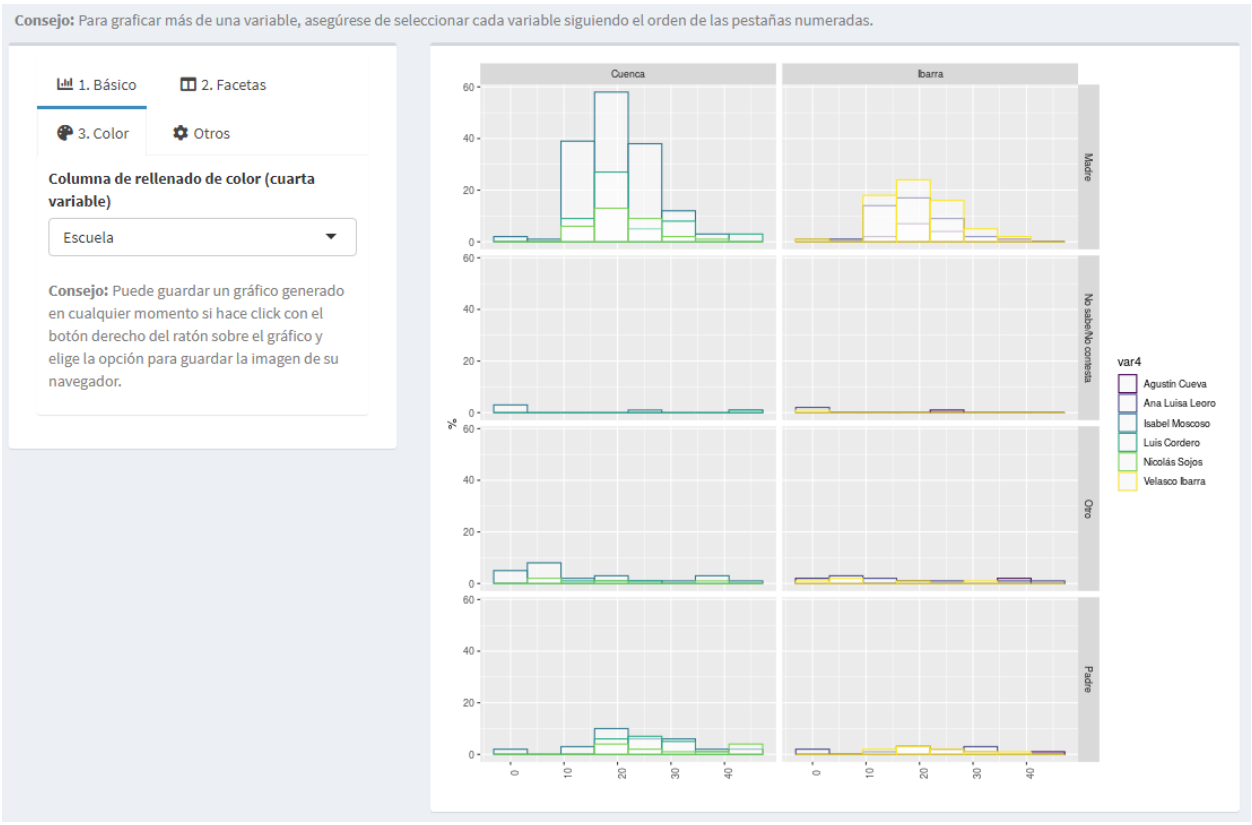


Figura 17. El ejemplo de progresión del histograma al hacer análisis de múltiples variables. En este caso con una variable principal, dos en facetas, y la variable de escuela delineando con colores las distintas frecuencias.

Paso 6: selección de filtrado de una columna

Finalmente, los datos que se visualizan se pueden filtrar para visualizar solo aquellos que contienen un cierto valor para una cierta columna. Aquí, en la pestaña Otros, se escogió la columna que representa si el hijo de la persona que llena la encuesta es niño, niña, o no contesta. Como valor de filtrado, se escogieron los registros que solo corresponden a niñas, con lo que el gráfico cambió para reflejar tal modificación.

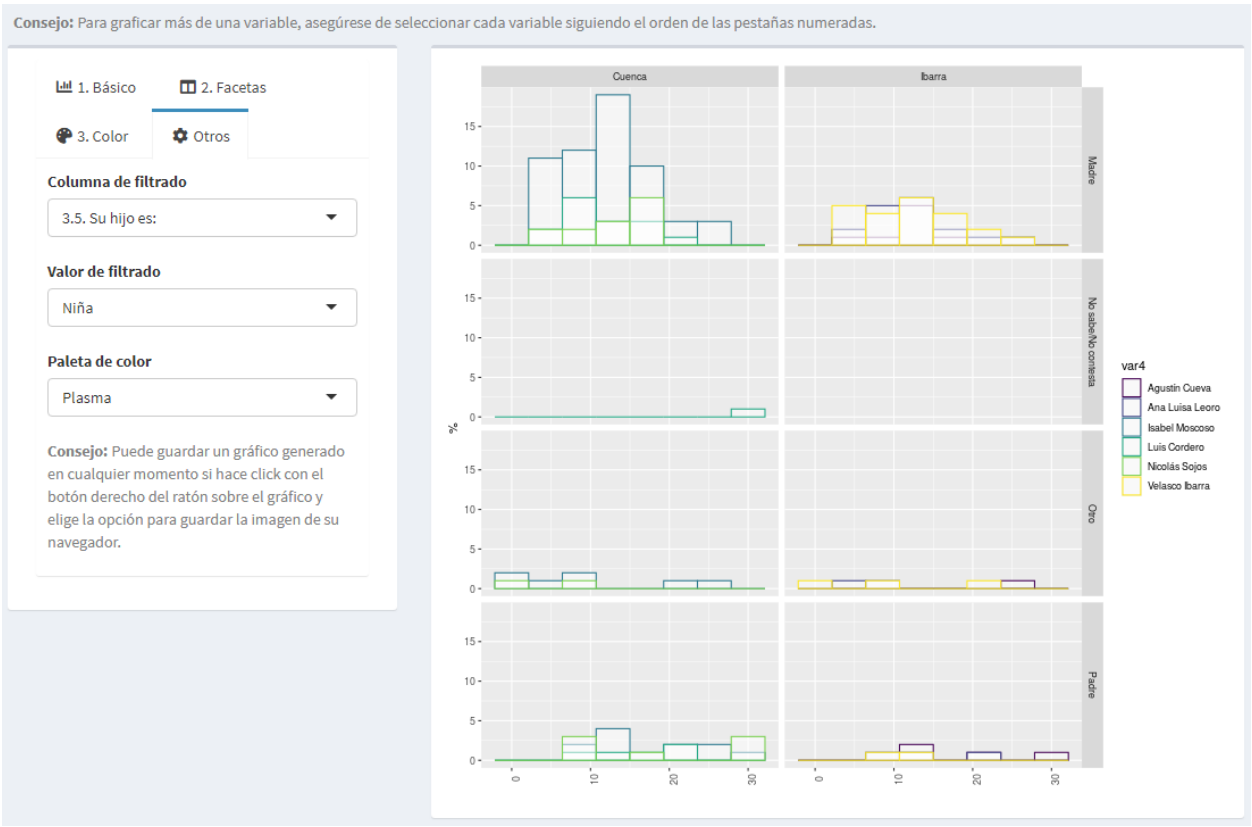


Figura 18. El ejemplo de progresión del histograma al hacer análisis de múltiples variables. En este caso con todas las variables mencionadas anteriormente graficadas. Adicionalmente, las frecuencias de edades solo muestran aquellos registros donde es niña.

16.4. Pestaña de gráfico de mosaico

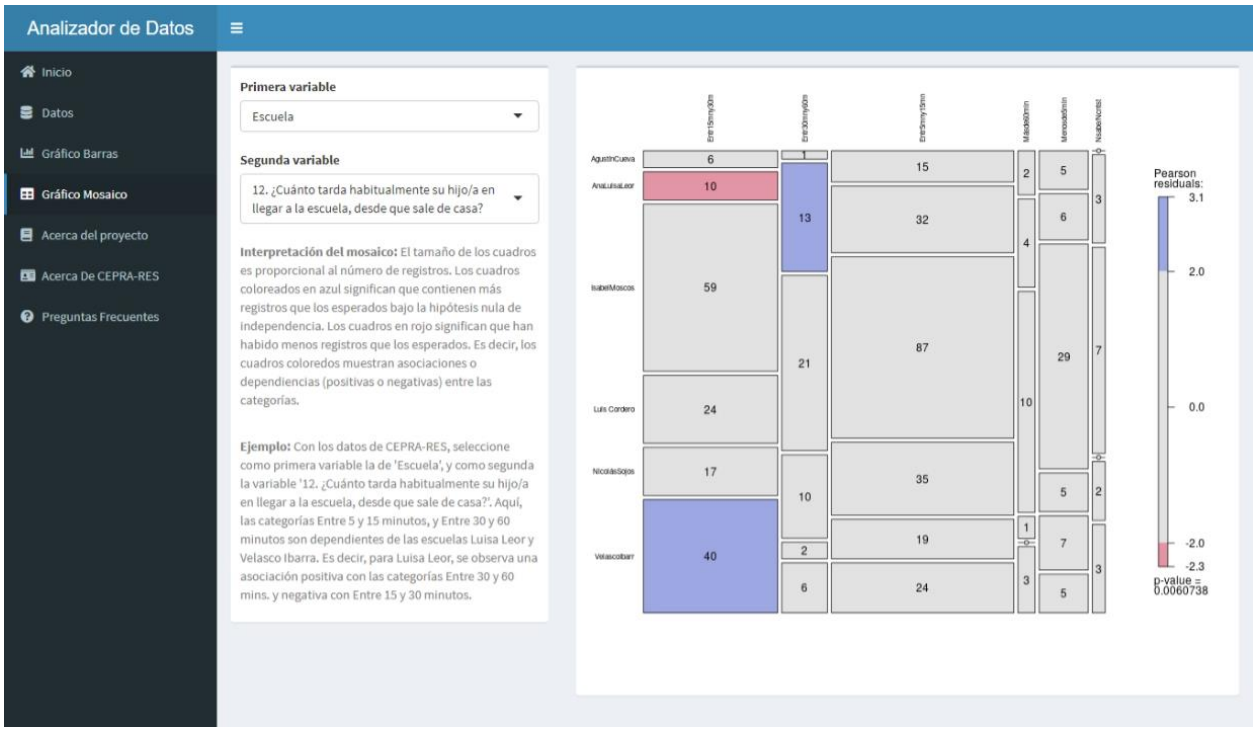


Figura 19. La visualización de la aplicación en su pestaña de gráficos de mosaico

En esta pestaña se pueden generar gráficos de mosaico, útiles para analizar la asociación (positiva o negativa) entre dos variables cualitativas—es decir, si los valores o categorías de una variable son (o no) dependientes, con cierto grado de confianza, a los valores o categorías de otra variable. Por este propósito, se deben elegir dos variables diferentes (la aplicación no falla si se escoge la misma variable para ambos campos, pero el resultado generado no es útil porque la asociación entre una variable y la misma variable siempre es del 100%).

Esta pestaña también posee un texto acompañante donde se describe el propósito del gráfico de mosaico y cómo interpretarlo, así como un pequeño ejemplo que el usuario puede seguir si está usando los datos de CEPRA-RES para ver el aspecto visual de variables con asociación positiva, negativa, o donde las variables sean independientes. Se decidió hacer esto porque los gráficos de mosaico no suelen ser muy conocidos, por lo que acompañarlo con una explicación breve se consideró ser importante.

16.5. Pestaña sobre el proyecto

Figura 20. La visualización de la aplicación en su pestaña que contiene información acerca del proyecto

Esta pestaña describe información básica sobre este proyecto de tesis, con la directora, el tribunal de tesis, y una breve descripción que incluye la escuela y universidad para la que se realiza este proyecto.

16.6. Pestaña sobre CEPRA-RES

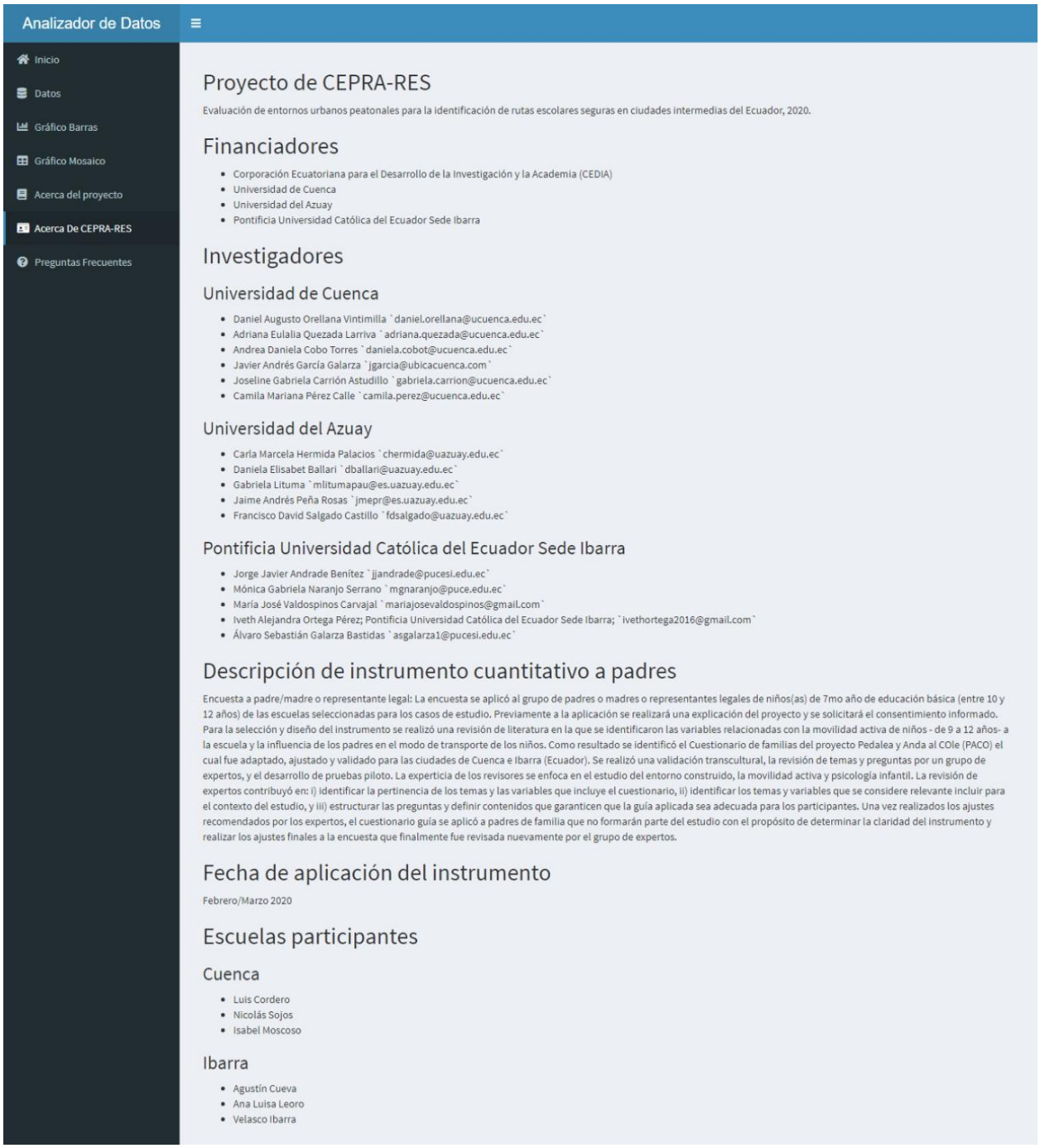


Figura 21. La visualización de la aplicación en su pestaña que contiene información del proyecto CEPRA-RES

La penúltima pestaña del menú describe los autores, contribuyentes e información sobre la encuesta CEPRA-RES, incluyendo las escuelas donde se realizaron las encuestas.

16.7. Pestaña sobre preguntas frecuentes

Esta última pestaña, agregada por parte de recomendación de uno de los evaluadores luego de la evaluación realizada, contiene una serie de preguntas consideradas las más frecuentes que podrían darse el momento de utilizar la aplicación. También cuenta con un enlace a un sitio web externo que realiza una explicación sobre cómo generar e interpretar gráficos de mosaico. El enlace se abre automáticamente en una pestaña nueva para que no reemplace la visualización de la aplicación en su pestaña actual.

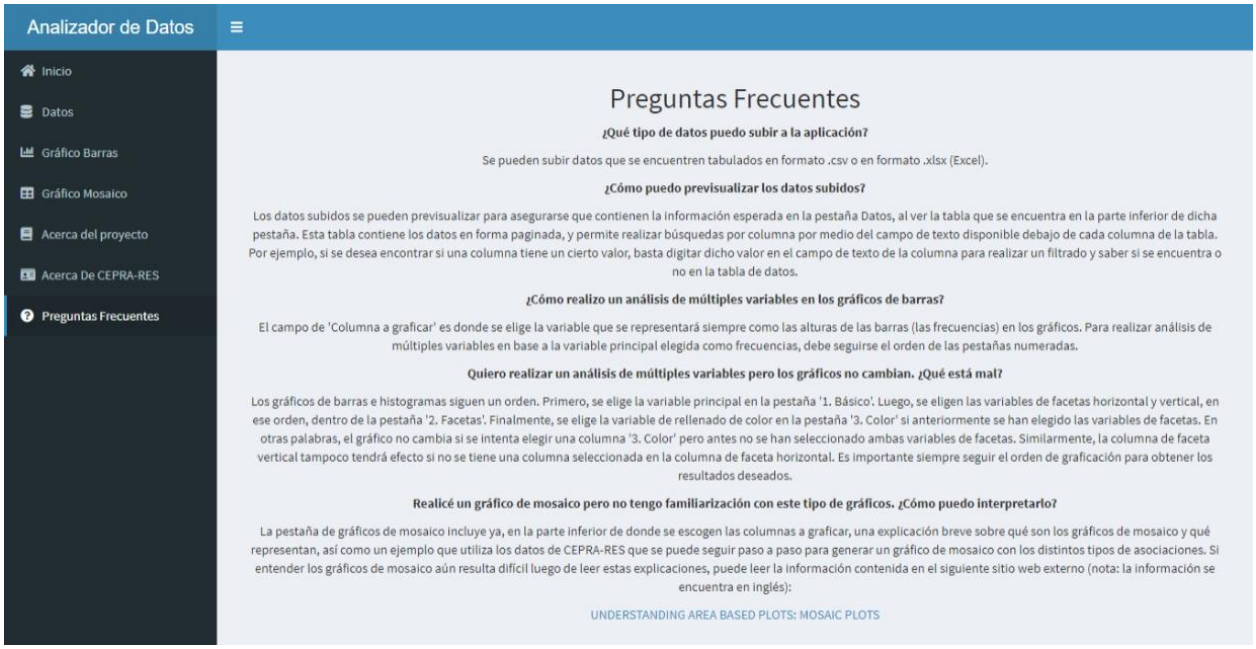


Figura 22. La visualización de la aplicación en su pestaña que contiene una lista de preguntas consideradas importantes de resolver para el usuario.

17. Resultados del contenedor Docker

Con la aplicación lista y el Dockerfile también preparado para crear el contenedor Docker que provee todas las utilidades y la aplicación de encuestas, se inicializó un nuevo contenedor con el siguiente comando ejecutado sobre la línea de comandos de la máquina con Ubuntu Linux:

```
sudo docker build --rm -t shiny-encuestas:latest .
```

El comando construye el contenedor, usando el nombre “shiny-encuestas” y asignándole un nombre de versión “latest”.

Una vez se creó el contenedor, para ejecutarlo y tener a la aplicación disponible en el puerto 3838 de la máquina, se ejecutó el script que también se incluye en el proyecto Docker, llamado **run.sh**, el cual se compone de lo siguiente:

```
#!/bin/sh



HOST_CACHE=~/renv/cache

CONTAINER_CACHE=/renv/cache

DOCKER_IMAGE_NAME=shiny-encuestas



mkdir -p ${HOST_CACHE}

sudo docker run --rm -e "RENV_PATHS_CACHE=${CONTAINER_CACHE}" -v
"${HOST_CACHE}:${CONTAINER_CACHE}" -p 3838:3838 ${DOCKER_IMAGE_NAME} R -s -e
'renv::restore(); shiny::runApp(host = "0.0.0.0", port = 3838)'
```

Este comando configura variables de entorno para indicar el directorio en la máquina anfitrión (no el contenedor) donde los paquetes de **renv** de la aplicación encuestas deben instalarse. Al final, ejecuta el comando donde este directorio de la máquina anfitrión se expone al contenedor. Adicionalmente, expone el puerto 3838 del contenedor al puerto 3838 del anfitrión, para que la aplicación esté disponible públicamente en el puerto 3838 como si estuviese publicada directamente en la máquina anfitrión y no un contenedor de Docker; si el puerto no se expone, la aplicación Shiny no estaría disponible en internet, y solo podría accederse por parte de la máquina anfitrión.

Ahora, cada vez que se ejecute el contenedor a través del comando **run.sh**, la aplicación está disponible para accederse como cualquier otro tipo de aplicación web, a través de la URL correspondiente al servidor, o a través de **localhost:3838/encuestas** por defecto. Con esto se comprobó que el Dockerfile funciona como se tenía previsto para que otras personas puedan utilizarlo con solo tener un entorno Linux con Ubuntu o Debian.

18. Resultado de la publicación de la aplicación en el servidor web de la universidad

La aplicación, como ya se mencionó en la sección de métodos, fue entregada directamente a través de su repositorio disponible en GitHub al administrador del servidor, ya que el servidor Shiny ya

se encontraba instalado en el servidor y hacer uso de un contenedor Docker no fue necesario en este caso. El administrador se encargó de descargar los archivos al servidor en Linux. Con copiar el archivo **Datos Concatenados.xlsx** a una carpeta **data** en la raíz del proyecto (archivo que se mantuvo fuera del repositorio en GitHub por motivos de privacidad), el administrador comprobó que no se dieron problemas ni hicieron falta pasos adicionales para que la aplicación funcione en el servidor, con lo que se verificó que la aplicación creada es fácil de reproducir. La aplicación, al momento de la publicación, se encuentra alojada en la dirección http://opengeodata.uazuay.edu.ec:3838/encuesta-cepra-res/.

Discusión y conclusiones

1. Resumen de resultados

A partir de la evaluación realizada con los investigadores, se concluye que la aplicación cumple con sus objetivos: provee tanto a personas que tienen experiencia con R, como a los que no, la posibilidad de generar los gráficos ofrecidos para realizar una exploración de cualquier conjunto de datos que sean cargados en la aplicación, incluyendo el proveer esto a los investigadores de CEPRA-RES por defecto debido a que los datos de dicha investigación fueron los utilizados como caso de ejemplo y se encuentran precargados en la aplicación. En general, los resultados de la evaluación demuestran que existe un nivel de intuición y satisfacción con su uso que es alto, y los principales obstáculos encontrados el momento de la evaluación estuvieron más relacionados con un desconocimiento de algunos aspectos como la interpretación de los gráficos de mosaico, mas no de forma notable con la funcionalidad o el uso de la aplicación misma.

Así mismo, la estructuración del código de la aplicación en diversos módulos, separados por directorios que indican su funcionalidad y permiten que programadores interesados en extender la funcionalidad de la aplicación tengan una estructura base sobre la que trabajar para mantener consistencia con la lógica arquitectónica del proyecto. Aunque medir la facilidad de extensión del código de la aplicación es una tarea más compleja de realizar, y por ende escapa al alcance del presente proyecto. La facilidad con la que el administrador del servidor de la Universidad del Azuay pudo publicar la aplicación a través del servicio de Shiny demuestra que la aplicación solo requiere ser copiada localmente para tener una aplicación cerca al 100% de funcionamiento; solo la falta del archivo Excel que contiene los datos de CEPRA-RES (no publicados en el repositorio público de código por motivos de privacidad y mantener la información recolectada de forma privada) previene de llegar a un 100%. No obstante, es fácilmente solucionable con tan solo modificar el código que carga el archivo a que cargue un archivo con otro nombre de la carpeta **data**. Adicional a la facilidad de publicación, es importante recalcar nuevamente que la contribución adicional realizada a través de proveer una imagen Docker que solo requiere una previa instalación de Ubuntu o Debian, con la que se puede crear un entorno que contenga el

servidor Shiny y la aplicación ejecutándose en un servidor local, ayuda también con la facilidad de publicación por terceros.

El proyecto completo, al final de todo el proceso, se consideró ser satisfactorio para las necesidades de todas las personas involucradas, recolectándose posibles mejoras y trabajo futuro que se podría realizar tomando el proyecto como base. La flexibilidad que el proyecto da en hacer al código fácil de extender y modificar es de gran importancia para hacer que las recomendaciones de trabajo futuro sean especialmente útiles.

Naturalmente, el proyecto en sí, así como la retroalimentación recibida durante el proceso de evaluación, muestran que el proyecto requiere ser extendido en su nivel de funcionalidad. Dichas nuevas funcionalidades obtenidas de la retroalimentación se encuentran fuera del ámbito y extensión de este proyecto. Estas limitantes se discuten en la siguiente sección, notando siempre que la facilidad de extensión de la aplicación hace más sencillo el poder solucionar dichas limitantes por parte de otros desarrolladores que así lo deseen y tengan el suficiente conocimiento técnico con R y Shiny.

2. Discusión sobre limitantes del proyecto actual y trabajo futuro

La principal limitante que se encontró con el proyecto, especialmente con la retroalimentación por parte de los evaluadores, estuvo relacionada con el grado de configuración del aspecto visual de los gráficos generados. Aunque los gráficos se adaptan fácilmente a cualquier conjunto de datos, y se puede controlar las variables a mostrar en los gráficos, así como la paleta de colores que se utiliza, el crear una aplicación desde cero que permita un alto grado de adaptabilidad a todo tipo de conjunto de datos significa que no se pueden ofrecer formas de adaptar a los gráficos de formas particulares. Por ejemplo, algunas de las características que los gráficos actuales no permiten configurar incluyen:

- Cambiar el tamaño de las etiquetas que se utilizan para los ejes de los gráficos. Si los datos de una columna graficada son muy largos en extensión de texto, los gráficos probablemente no puedan mostrar toda la extensión del texto. Una parte del texto quedará, inevitablemente, cortada. Así mismo, cambiar el tamaño de las barras dinámicamente para que el espacio se utilice de forma eficiente.

- Cambiar los datos utilizados de forma dinámica. Por ejemplo, durante la evaluación, un investigador preguntó si se podía graficar una columna de los datos de CEPRA-RES como "Sí" y "No" en vez de los valores que la columna poseía, que constaban de los valores "1" y "0", que son los que se muestran en el gráfico, pero pueden resultar no del todo intuitivos al momento de visualizarlos como tal en un gráfico.

- Dar un orden en el que se muestra cada valor de una columna en un gráfico. Al momento, el orden de las columnas se da por orden lexicográfico—es decir, las columnas se ordenan por cada uno de sus caracteres de forma ascendente. Esto puede no ser de preferencia para ciertos casos. Por ejemplo, si una columna consta de los valores "$400", $1000" y "$5000", por orden lexicográfico, las columnas se mostrarán en el orden de "$1000", "$400" y "$5000", lo cual no resulta intuitivo.

- Cambiar entre la visualización de frecuencias de un gráfico entre valores porcentuales o valores absolutos. El poder visualizar los datos de estas dos formas puede resultar útil, pero los cambios necesarios requeridos para que los gráficos se generen dinámicamente de las dos formas dependiendo de lo que el usuario elija se consideró caer fuera del ámbito del proyecto.

- Dar más opciones para las paletas de colores utilizadas, incluyendo la sugerencia recibida durante la evaluación sobre incluir una paleta de colores que sea en rampa: es decir, con colores más distinguibles.

- Dar una opción para escoger el orden en que los datos de tipo cualitativo se muestran en la leyenda y en los gráficos, tanto de barras/histogramas como de mosaico.

- Manejar la aparición de datos de tipo fecha cuando tienen un formato no estandarizado. Actualmente, si R encuentra una columna de tipo fecha, intenta convertir este tipo en datos de tipo cualitativo al clasificarlos por el espacio de tiempo. Por ejemplo, si los valores de una columna de tipo fecha describen un día calendario del año, los datos se clasificarán como tal. La limitante que existe con este tipo de clasificación es que, si la columna tiene un formato que R considera ambiguo (por ejemplo, 01/02/21), no se puede realizar la categorización. La solución a este problema, al momento, es manualmente cambiar los

valores de una columna tipo fecha a un formato que no sea ambiguo. Por ejemplo: 6 October, 2020.

Todas estas características se podrían implementar a través de la extensión del código disponible públicamente, por lo que se considera que es trabajo futuro potencial que otros desarrolladores pueden encontrar útil implementar. Para motivos de su uso en el proyecto CEPRA-RES, y el tiempo que tomaría realizar estos cambios en el proyecto actual, esta funcionalidad se considera estar fuera del alcance del proyecto.

Otros aspectos de la aplicación que también podrían cambiarse en la aplicación son los aspectos estéticos. Aunque su aspecto visual se considera atractivo para su uso, la apariencia podría cambiarse significativamente si se desea para que se adapte mucho mejor para tener un aspecto más adaptado a aquellos encontrados en páginas web muy modernas. Este tipo de extensión requeriría de personas que tengan conocimientos de diseño gráfico y uso de estilos a través de archivos **.css**.

Finalmente, se encuentra la extensibilidad de la aplicación a través del agregar nuevos tipos de gráficos a generar. La aplicación solo permite generar gráficos de barras, histogramas o gráficos de mosaicos. Gráficos de pastel, o gráficos de caja y bigote son algunos ejemplos de gráficos estadísticos que también pueden ser de utilidad para personas que utilicen la aplicación, y que podrían agregarse dependiendo de las necesidades encontradas fuera del ámbito de este proyecto de tesis.

3. Conclusiones

Después de todos los pasos descritos en este documento, se concluye que:

- La aplicación es de utilidad para investigadores que deseen realizar análisis de datos de forma exploratoria, tanto si son personas que tienen conocimiento de R como si no. En la encuesta realizada a los evaluadores, se demostró que la generación de gráficos fue más intuitiva y rápida que al utilizar R. A su vez, aquellas personas que no conozcan R consiguen más beneficios de la aplicación, ya que les permite generar los gráficos que se pueden generar por R y sus librerías sin necesidad de aprender el lenguaje.

- La documentación paso a paso de la creación de la aplicación Shiny a través del código, y de la creación del Dockerfile para crear un contenedor que aloje la aplicación, permiten fácilmente que otras personas puedan leer el documento, descargar los recursos disponibles, replicarlos, estudiarlos, modificarlos y extenderlos para adaptarse a sus necesidades. El proceso de replicación también se considera ser sencillo de realizar, ya que sigue prácticas estandarizadas en el área de desarrollo de software, utilizando repositorios de código públicos, control de versionamiento y estructuración de la aplicación por arquitecturas conocidas como es la Modelo-Vista-Controlador utilizada por la aplicación.

- La forma en que se estructuró la aplicación permite también la fácil extensibilidad de la aplicación. Un desarrollador que desee extender la aplicación solo debe entender la estructura a grandes rasgos de la aplicación para poder agregar nueva funcionalidad sin necesidad de realizar una reestructuración de la aplicación.

- Finalmente, que el proyecto dispone de los recursos necesarios para dar a cualquier tipo de usuario que tenga interés en este proyecto las formas de utilizar la aplicación, replicarla, o modificarla, a través de los siguientes recursos:

    - URL del sitio web público: http://opengeodata.uazuay.edu.ec:3838/encuesta-cepra-res/

    - URL del repositorio de código de la aplicación: https://github.com/HollowVin/encuestas

    - URL del repositorio que contiene el Dockerfile y los recursos necesarios para generar un contenedor con la aplicación: https://github.com/HollowVin/encuestas-docker

Bibliografía

Boettiger, C., Eddelbuettel, D., & Ross, N. (s/f). *The Rocker Project*. Recuperado el 6 de abril de 2021, de https://www.rocker-project.org/

Budczies, J., Pfarr, N., Romanovsky, E., Endris, V., Stenzinger, A., & Denkert, C. (2018). Ioncopy: An R Shiny app to call copy number alterations in targeted NGS data. *BMC Bioinformatics*, *19*(1), 1–4. https://doi.org/10.1186/s12859-018-2159-5

Docker. (s/f). *Dockerfile reference*. Docker docs. Recuperado el 2 de abril de 2021, de https://docs.docker.com/engine/reference/builder/

Docker Inc. (s/f). *Docker*. Recuperado el 4 de mayo de 2021, de https://www.docker.com/

Ekiz, H. A., Conley, C. J., Stephens, W. Z., & O'Connell, R. M. (2020). CIPR: a web-based R/shiny app and R package to annotate cell clusters in single cell RNA sequencing experiments. *BMC bioinformatics*, *21*(1), 191. https://doi.org/10.1186/s12859-020-3538-2

Free Software Foundation. (2017). *GNU General Public License, version 2*. GNU Operating System. https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html

Git Team. (s/f-a). *Git - commit*. Recuperado el 1 de mayo de 2021, de https://git-scm.com/docs/git-commit

Git Team. (s/f-b). *Git - pull*. Recuperado el 21 de septiembre de 2020, de https://git-scm.com/docs/git-pull

Git Team. (s/f-c). *Git - push*. Recuperado el 21 de septiembre de 2020, de https://git-scm.com/docs/git-push

Hart, R., Burns, D., Ramaekers, B., Ren, S., Gladwell, D., Sullivan, W., Davison, N., Saunders, O., Sly, I., Cain, T., & Lee, D. (2020). R and Shiny for Cost-Effectiveness Analyses: Why and When? A Hypothetical Case Study. *PharmacoEconomics*, *38*(7), 765–776. https://doi.org/10.1007/s40273-020-00903-9

Kaufman, A. R. (2020). Implementing novel, flexible, and powerful survey designs in R Shiny. *PLoS ONE*, *15*(4), 1–15. https://doi.org/10.1371/journal.pone.0232424

Larman, C., & Basili, V. R. (2003). IIDHistory. *Computer*, *36*(6), 47–56. https://doi.org/10.1109/MC.2003.1204375

Meyer, D., Zeileis, A., Hornik, K., Gerber, F., & Friendly, M. (2020). *vcd: Visualizing Categorical Data*. CRAN. https://cran.r-project.org/web/packages/vcd/index.html

Microsoft. (s/f). *ASP.NET MVC Pattern*. Recuperado el 28 de septiembre de 2020, de https://dotnet.microsoft.com/apps/aspnet/mvc

Microsoft. (2020). *What is the Windows Subsystem for Linux?* https://docs.microsoft.com/en-us/windows/wsl/about

Narkhede, S. (2018). *Understanding descriptive statistics*. https://towardsdatascience.com/understanding-descriptive-statistics-c9c2b0641291

Open Source Initiative. (s/f). *The MIT License*. Recuperado el 2 de octubre de 2020, de https://opensource.org/licenses/MIT

Owen, R. K., Bradbury, N., Xin, Y., Cooper, N., & Sutton, A. (2019). MetaInsight: An interactive web-based tool for analyzing, interrogating, and visualizing network meta-analyses using R-shiny and netmeta. *Research Synthesis Methods*, *10*(4), 569–581. https://doi.org/10.1002/jrsm.1373

Paasivaara, M. (2006). *Using iterative and incremental processes in global software development*. 42–47. https://doi.org/10.1049/ic:20040312

RStudio Team. (s/f). *shinydashboard - Appearance*. Recuperado el 20 de abril de 2021, de https://rstudio.github.io/shinydashboard/appearance.html

RStudio Team. (2014). *shinydashboard*. https://rstudio.github.io/shinydashboard/

RStudio Team. (2017). *Reactivity - An overview*. https://shiny.rstudio.com/articles/reactivity-overview.html

RStudio Team. (2021). *RStudio*. https://www.rstudio.com/

Rudis, B., Ross, N., & Garnier, S. (2021). *Introduction to the viridis color maps*. CRAN. https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html

Shiny Team. (s/f). *Shiny*. Recuperado el 9 de julio de 2020, de https://shiny.rstudio.com/

Sievert, C. (2020). *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman and Hall/CRC. https://plotly-r.com

The R Project. (s/f). *What is R?* Recuperado el 14 de julio de 2020, de https://www.r-project.org/about.html

The Rocker Project. (2020). *rocker-org - shiny*. GitHub. https://github.com/rocker-org/shiny

Todaro, D. (2019). *The Epic Guide to Agile: More Business Value on a Predictable Schedule with Scrum*. R9 Publishing LLC.

University of Alabama at Birmingham. (s/f). *Descriptive statistics vs. inferential statistics*. Recuperado el 8 de julio de 2020, de https://businessdegrees.uab.edu/blog/descriptive-statistics-vs-inferential-statistics/

Ushey, K. (s/f). *Using renv with Docker*. Recuperado el 26 de abril de 2021, de https://rstudio.github.io/renv/articles/docker.html

Ushey, K. (2021). *Introduction to renv*. https://rstudio.github.io/renv/articles/renv.html

Valero-Mora, P. M. (2010). ggplot2: Elegant Graphics for Data Analysis . En *Journal of Statistical Software* (Vol. 35, Número Book Review 1). Springer-Verlag. https://doi.org/10.18637/jss.v035.b01

Varet, H., & Coppée, J. Y. (2019). CheckMyIndex: A web-based R/Shiny interface for choosing compatible sequencing indexes. *Bioinformatics*, *35*(5), 901–902. https://doi.org/10.1093/bioinformatics/bty706

Verity, R., Collins, C., Card, D. C., Schaal, S. M., Wang, L., & Lotterhos, K. E. (2017). minotaur: A platform for the analysis and visualization of multivariate results from genome scans with R Shiny. *Molecular Ecology Resources*, *17*(1), 33–43. https://doi.org/10.1111/1755-0998.12579

Walker, K. E. (2016). Tools for Interactive Visualization of Global Demographic Concepts in R. *Spatial Demography*, *4*(3), 207–220. https://doi.org/10.1007/s40980-016-0029-1

Wickham, H., & Bryan, J. (s/f). *readxl*. Tidyverse. Recuperado el 3 de octubre de 2020, de https://readxl.tidyverse.org/

Wickham, H., & Hester, J. (s/f). *readr*. Tidyverse. Recuperado el 3 de octubre de 2020, de https://readr.tidyverse.org/

Wickham, H., Winston, C., Henry, L., Lin, T., Takahashi, K., Wilke, C., Woo, K., Yutani, H., Dunnington, D., & RStudio Team. (2021). *ggplot2 - Overview*. Tidyverse. https://ggplot2.tidyverse.org/

Wikipedia. (2020). *https://en.wikipedia.org/wiki/Iterative_and_incremental_development*. https://en.wikipedia.org/wiki/Iterative_and_incremental_development

Witte, R. S., & Witte, J. S. (2017). *Statistics* (11a ed.). Wiley.

Wojciechowski, J., Hopkins, A. M., & Upton, R. N. (2015). Interactive pharmacometric applications using R and the Shiny package. *CPT: Pharmacometrics and Systems Pharmacology*, *4*(3), 146–159. https://doi.org/10.1002/psp4.21

Yang, C. T., Chan, Y. W., Liu, J. C., & Lou, B. S. (2020). An implementation of cloud-based platform with R packages for spatiotemporal analysis of air pollution. *Journal of Supercomputing*, *76*(3), 1416–1437. https://doi.org/10.1007/s11227-017-2189-1

Yu, Y., Ouyang, Y., & Yao, W. (2018). ShinyCircos: An R/Shiny application for interactive creation of Circos plot. *Bioinformatics*, *34*(7), 1229–1231.

https://doi.org/10.1093/bioinformatics/btx763