FACULTAD DE CIENCIAS AGRARIAS
UNIVERSIDAD NACIONAL DE ROSARIO

Paquete de R y aplicación Web para el análisis de datos provenientes de ensayos multiambientales

JULIA ANGELINI

TRABAJO FINAL PARA OPTAR AL TITULO DE ESPECIALISTA EN BIOINFORMÁTICA

DIRECTOR: Gerardo Cervigni
CO-DIRECTOR: Marcos Prunello

AÑO: 2020

Paquete de R y aplicación Web para el análisis de datos provenientes de ensayos multiambientales

Julia Angelini

Licenciada en Estadística – Universidad Nacional de Rosario

Este Trabajo Final es presentado como parte de los requisitos para optar al grado académico de Especialista en **Bioinformática**, de la Universidad Nacional de Rosario y no ha sido previamente presentada para la obtención de otro título en ésta u otra Universidad. El mismo contiene los resultados obtenidos en investigaciones llevadas a cabo en **el Centro de Estudios Fotosintéticos y Bioquímicos (CEFOBI)**, durante el período comprendido entre **los años 2017 y 2020**, bajo la dirección del **Dr. Gerardo Cervigni y Mgs. Marcos Prunello**.

Nombre y firma del autor

Nombre y firma del Director

Nombre y firma del Co - Director

Defendida: _____ de 20____.

Agradecimientos

En este trabajo final, directa o indirectamente, participaron muchas personas a las que les quiero agradecer.

En primer lugar al Dr. Gerardo Cervigni por confiar en mí y permitirme explorar el mundo de la Bioinformática durante mi tesis doctoral, para que hoy sea parte de mis conocimientos. Al Mgs. Marcos Prunello por acompañarme en el desarrollo del trabajo final, por su dedicación y sus consejos.

Todo esto nunca hubiera sido posible sin el apoyo y el cariño de mis padres, de mi hermano, de Otto, de Segundo y Kalita. Siempre estuvieron a mi lado, las palabras nunca serán suficientes para agradecerles.

A mis compañeras Jor y Lu, por su ayuda y por compartir excelentes momentos.

A Gaby y Euge mis compañeras de CEFOBI, gracias a ustedes este camino ha sido más fácil!

A mis amigos, por estar siempre presentes.

Muchas gracias a todos!

Abreviaturas y Símbolos

EMA: ensayos multiambientales.

IGA: interacción genotipo ambiente.

NCOI: interacción sin cambio de rango, del inglés *no crossover interaction*

COI: interacción con cambio de rango, del inglés *crossover interaction*

ANOVA: análisis de la variancia, del inglés *analysis of variance*

AMMI: modelo de los efectos principales aditivos y interacción multiplicativa, del inglés *Additive Main effects and Multiplicative Interaction*

ACP: análisis de componentes principales

SREG: modelo de regresión por sitio, del inlés *Site Regression model*

DVS: descomposición de valores singulares

GNU: *General Public Licence*

CRAN: *Comprehensive R Archve Network*

EM: maximización de la esperanza, del inglés *Expectation-Maximization*

Resumen

Las variedades mejoradas son el resultado del trabajo de desarrollo genético llevado a cabo en los programas de fi

tomejoramiento, los cuales se extienden a lo largo de varios años y requieren cuantiosas inversiones. En etapas avanzadas, los ensayos multiambientales (EMA), que comprenden experimentos en múltiples ambientes, son herramientas fundamentales para incrementar la productividad y rentabilidad de los cultivos. La vigencia comercial de las variedades puede extenderse durante varias décadas, por lo que su elección es crítica para que el productor evite pérdidas ecoómicas por malas campañas y el suministro al mercado sea constante. Consecuentemente, un análisis adecuado de la información de los EMA es indispensable para que el programa de mejoramiento de los cultivos sea efie

caz. Los programas informáticos se han convertido, hoy en día, en una herramienta esencial par el análisis de datos. Actualmente, R es uno de los programas más utilizados debido a su potencia y a su distribución como software libre. Actualmente existen numerosos paquetes de R lo cual provoca que no sea sencillo encontrar un paquete que sea útil para un determinado

propósito sino que se debe recurrir a varios de ellos. Frecuentemente, los mejoradores no tienen un manejo fluido de paquetes estadísticos que permitan entender la dinámica del problema. En este sentido el paquete Shiny que permite crear una interfaz gráfica entre R y el usuario, permitiendo acercar la potencia de R a todo tipo de usuarios. En el presente trabajo se desarrolló un paquete de R que permita analizar los datos provenientes de EMA para aquellos usuarios que tengan manejo del lenguaje de programación y una interfaz en Shiny que permita realizar los principales análisis del paquete sin necesidad de programar.

Palabras Clave:

Abstract

Keywords:

Índice general

Capítulos **Página**

Índice de figuras

Capítulo 1

Introducción

A lo largo de la historia de la agricultura, el hombre ha desarrollado el mejoramiento vegetal en forma sistemática y lo ha convertido en un instrumento esencial para incrementar la producción agrícola en términos de cantidad, calidad y diversidad.

El fitomejoramiento, en un sentido amplio, busca alterar la frecuencia alélica de los genes para obtener cultivares genéticamente superiores, adaptados a condiciones específicas, con mayor rendimiento y mejor calidad que las variedades nativas o criollas (Allard, 1967). En otras palabras, su objetivo es desarrollar genotipos cuya superioridad genética esté de acuerdo con las condiciones agroclimáticas donde se producen, necesidades y recursos de todos aquellos que elaboran, transforman y consumen productos vegetales.

Las variedades mejoradas son el resultado del trabajo llevado a cabo en los programas de fitomejoramiento, los cuales se extienden a lo largo de varios años y requieren cuantiosas inversiones. Generalmente, en etapas tempranas de estos programas existe un gran número de genotipos experimentales con pocos antecedentes de evaluación; mientras que en etapas posteriores se evalúa un número menor con más repeticiones y en más ambientes/años. Estos ensayos multiambientales (EMA) son herramientas fundamentales para evaluar la productividad para así asegurar la rentabilidad de los cultivos.

Como consecuencia de que los EMA se llevan a cabo en múltiples ambientes/años, la aparición de la interacción genotipo × ambiente (IGA) es inevitable debido a las variaciones en las condiciones climáticas y de suelo. La IGA es considerada por los fitomejoradores como el principal factor factor limitante de respuesta a la selección y, en general, de la eficiencia de los programas de mejoramiento, por provocar respuestas altamente variables en los diferentes ambientes (Crossa et al., 1990; Cruz Medina, 1992); **Kang y Magari, 1996)**. Gauch y Zobel (1997) explicaron que si no hubiera interacción, una sola variedad / híbridos rendirían al máximo en todo el mundo, además los materiales podrían evaluarse en un solo lugar y proporcionarían resultados universales.

Peto (1982) ha distinguido las interacciones cuantitativas, conocida también como sin cambio de rango o *no crossover* (NCOI), de las cualitativas, denominada también con cambio de rango o *crossover* (COI) (Cornelius et al., 1996). Cuando dos genotipos X e Y tienen una respuesta diferencial en dos ambientes, y hay cambios en el orden de los genotipos se dice que la IGA es del tipo COI (Figura 1.1(A)), es NCOI cuando su ordenación permanece sin cambios (Figura 1.1(B)), y. finalmente, es inexistente cuando los genotipos responden de manera similar en ambos ambientes (Figura 1.1(C)).

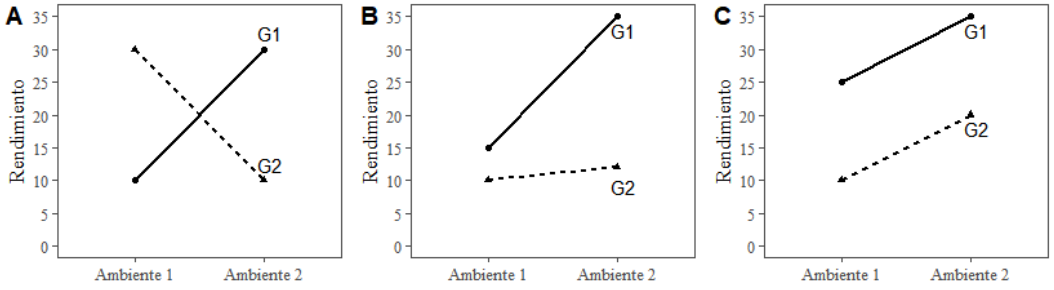


Figura 1.1: Representación gráfica de tipos de IGA: (A)IGA crossover, (B) IGA no crossover y (C) no IGA

Entre las implicancias negativas de la IGA en los programas de mejoramiento se encuentra el impacto negativo sobre la heredabilidad, cuanto menor sea la heredabilidad de un caracter, mayor será la dificultad para mejorarlo. Como consecuencia, la información sobre la estructura y la naturaleza de la IGA es particularmente útil para determinar si se deben desarrollar cultivares con adaptación amplia o específica (**Bridges, 1989**). La decisión sobre qué tipo de estrategia seguir, involucra considerar y analizar conceptos como regiones ecológicas, ecotipos y mega-ambientes (**Kang et al., 2004**).

La vigencia comercial de las variedades puede extenderse durante varias décadas, por lo que su elección es crítica para que el productor evite pérdidas económicas por malas campañas y el suministro al mercado sea constante. Consecuentemente, un análisis adecuado de la información de los EMA es indispensable para que el programa de mejoramiento genético de los cultivos sea eficaz. El rendimiento medio en los ambientes es un indicador suficiente del rendimiento genotípico sólo en ausencia de IGA (Yan y Kang, 2003). Sin embargo, la aparición de IGA es inevitable y no basta con la comparación de las medias de los genotipos, sino que se debe recurrir a una metodología estadística más aporopiada. La metodología estadística más difundida para analizar los datos provenientes de EMA se basa en modificaciones de los modelos de regresión, análisis de variancia (ANOVA) y técnicas de análisis multivariado.

Particularmente, para el estudio de la IGA y los análisis que de ella se derivan, dos

modelos multiplicativos han aumentado su popularidad entre los fitomejoradores, especialmente como una herramienta de análisis gráfico: el modelo de los efectos principales aditivos e interacción multiplicativa (*Additive Main effects and Multiplicative Interaction*, AMMI) (Kempton, 1984; Gauch, 1988) , y el de regresión por sitio (*Site Regression model*, SREG) (**Cornelius et al., 1996**; (Gauch y Zobel, 1997). Estos modelos combinan un ANOVA con la descomposición de valores singulares (DVS) o un análisis de componentes principales (ACP) sobre la matriz residual de ANOVA. En SREG, el ANOVA se realiza sobre el efecto ambiental (A), mientras que en AMMI el ANOVA se realiza sobre los efectos principales de genotipos (G) y A. A partir de estos modelos se pueden visualizar los patrones puramente atribuibles a los efectos la IGA, mediante el biplot derivado del modelo AMMI y de aquel obtenido del SREG explorar simultáneamente patrones de variación conjunta de G e IGA (**Yan y Hunt (2002)**).

Una limitación importante de la mayoría de las propuestas del análisis de EMA es que requieren que el conjunto de datos esté completo. Aunque los EMA están diseñados para que la totalidad de los genotipos se evalúen en todos los ambientes, las tablas de datos genotipo $\times$ ambiente completas son poco frecuentes (no todos los genotipos se encuentran en todos los ambientes). Esto ocurre, por ejemplo, debido a errores de medición o pérdidas de plantas por presencia de animales, inundaciones o problemas durante la cosecha, además de la dinámica propia de la evaluaciones en las que se incorporan y se descartan genotipos debido a su pobre desempeño (Hill y Rosemberg, 1985). En estos casos, antes de llevar a cabo el análisis de interés resulta necesario imputar los datos faltantes con metodología apropiada(**CITA**).

En este contexto, el análisis de datos provenientes de EMA requiere metodología estadística cuyas rutinas informáticas no se encuentran disponibles en programas comerciales debido a su reciente desarrollo o bien, se deben utilizar varios de ellos para lograr un único objetivo. Esto último genera el inconveniente de tener que disponer de todos los programas necesarios para los distintos análisis, atender los requerimientos de formatos de datos usados por cada uno de ellos, y comprender los diversos tipos de salidas en las que se presentan los resultados obtenidos. Además, los costos de las licencias algunos programas pueden resultar muy elevados.

El software R, desarrollado por *The R Foundation for Statistical Computing*, se trata de un proyecto colaborativo de uso libre y distribuido bajo los términos de la *General Public Licence* (GNU). Este software surge como resultado de la implementación del lenguaje S, uno de los más utilizados en investigación por la comunidad estadística. A diferencia de los programas utilizados frecuentemente para el análisis de datos, al ser R un lenguaje de programación, no dispone de una interfaz gráfica que facilite su uso mediante menús, generando cierta dificultad para aquéllos que no se encuentran familiarizados con

la programación. Sin embargo, por ser un software libre, permite a los usuarios definir sus propias funciones dando lugar a mayores posibilidades respecto del manejo y análisis de los datos disponibles. Si bien la versión básica del programa dista mucho de ser amigable, RStudio, un entorno de desarrollo integrado (IDE) gratuito y de código abierto para R, permite una interacción más fluida con el programa, actuando como una interfaz con el usuario. Al formar parte de un proyecto colaborativo, R promueve el hecho de que los usuarios compartan con la comunidad las funciones creadas por ellos, por lo que está en continuo desarrollo y actualización. A menudo, no resulta sencillo reutilizar una función creada por algún usuario, por ello, se ha introducido la posibilidad de crear paquetes (*package*). Éstos son una colección de objetos desarrollados y organizados siguiendo un protocolo fijo, lo cual garantiza un soporte mínimo para el usuario así como la ausencia de errores (de sintaxis) en la programación.

Actualmente, R cuenta con 14 paquetes básicos y 29 recomendados para su funcionamiento que se instalan automaticamente en él, tal como *base* o *stats*. Entre los paquetes que extienden las funciones básicas de R se encuentran, *plyr*, *lubridate*, *reshape2* y *stringr* para la manipulación de los datos; *ggplot2* y *rgl* para la visualización; *knitr* y *xtable* para la presentación de resultados; entre otros. La lista completa de los paquetes oficiales puede consultarse en CRAN[1], que contaba con más de 14.000 paquetes disponibles hasta junio de 2019. Esta gran variedad de paquetes es una de las razones de la gran difusión de R, ya que cada usuario puede tratar su problematica utilizando un paquete desarrollado por otro usuario. Además de los paquetes oficiales, existen otros que pueden instalarse desde repositorios como Github, Bioconductor, rOpenSci, entre otros. Sin embargo, no es sencillo encontrar un paquete que incluya todas las funciones necesarias para un cierto análisis, sino que debe recurrirse a varios de ellos.

Con frecuencia, los mejoradores usan programas de análisis de datos que cuentan con interfaz gráfica de usuario, evitando el manejo de un lenguaje de programación. Hoy en día las aplicaciones web son muy populares debido a la facilidad de su uso, ya que no requiere de una instalación en el ordenador del usuario, simplemente se accede a través de un navegador siendo posible utilizarlas desde cualquier dispositivo con conexión a internet.

Crear aplicaciones web puede resultar difícil para la mayoría de los usuarios de R debido a que se necesita un conocimiento profundo de otros lenguajes de programación como HTML, CSS y JavaScript; y además el desarrollo de aplicaciones interactivas requiere un análisis cuidadoso de los flujos de trabajo para asegurarse de que cuando una entrada cambie, solo se actualicen las salidas relacionadas. Sin embargo, Winston C., et

[1]CRAN (Comprehensive R Archve Network) es el repositorio oficial de paquetes de R, el lugar donde se publican las nuevas versiones del programa, etc. Contiene la lista completa de paquetes oficiales. `https://cran.r-project.org/web/packages/available_packages_by_name.html`

al. (2012) crearon el paquete *shiny* de R que facilita el desarrollo de aplicaciones web, acercando la potencia del software a todo tipo de usuarios sin tener la necesidad de conocer el lenguaje de programación. Este paquete le facilita al programador de R el proceso de creación de aplicaciones web al proporcionar un conjunto de funciones de interfaz de usuario que generan el HTML, CSS y JavaScript sin necesidad de conocer los detalles de dichos lenguajes.

El objetivo del presente trabajo es: (i) desarrollar un paquete de R para el análisis de datos provenientes de EMA que incluya metodologías existentes, modificadas para favorecer su uso, y otras recientemente publicadas y no disponibles en R y (ii) desarrollar una aplicación web Shiny que sirva como interfaz gráfica de usuario para que el paquete pueda ser utilizados por aquellos que no tienen conocimientos del lenguaje de programación.

Capítulo 2

Objetivos

2.1. Objetivo general

Desarrollar un paquete de R para el análisis de datos provenientes de EMA y una interfaz gráfica de usuario para el mismo a través de la aplicación web Shiny.

2.2. Objetivos específicos

- Mostrar un flujo de trabajo reproducible para la construcción de paquetes de R.
- Programar e incluir en el paquete metodología para el análisis de datos provenientes de EMA recientemente publicada y no disponible en R.
- Añadir en el paquete de R funciones ya existentes con modificaciones o agregados para favorecer su uso.
- Desarrollar una aplicación web shiny que sirva como interfaz gráfica de usuario para el paquete.
- Publicar el paquete y la aplicación web para su libre uso.

Capítulo 3

Métodos

3.1. Métodos estadísticos

3.1.1. Modelo AMMI y SREG

El modelo AMMI propuesto por Zobel et al. (1988) es un modelo multiplicativo en el cual se expresa el fenotipo de un genotipo en un ambiente de la siguiente forma:

$$y_{ij} = \mu + G_i + A_j + \sum_{k=1}^{q} \lambda_k \alpha_{ik} \gamma_{jk} \qquad i = 1, ..., g; \;\; j = 1, ..., a; \;\; q = min(g-1, a-1)$$

donde

- y_{ij} es el caracter fenotípico evaluado (rendimiento o cualquier otro caracter de interes) del i-ésimo genotipo en el j-ésimo ambiente,
- μ es la media general,
- G_i es el efecto del i-ésimo genotipo,
- A_j es el efecto del j-ésimo ambiente
- $\sum_{k=1}^{q} \lambda_k \alpha_{ik} \gamma_{jk}$ es la sumatoria de componentes multiplicativas utilizadas para modelar la IGA. Siendo, λ_k el valor singular para la k-ésima componente principal (PC) α_{ik} y γ_{jk} son los scores de las PC para el i-ésimo genotipo y el j-ésimo ambiente para la k-ésima componente, respectivamente.

En cambio, el modelo SREG (Cornelius et al., 1996; Crossa y Cornelius, 1997 y 2002) expresa el fenotipo observado en función del efecto ambiental en forma aditiva y del genotipo e interacción agrupados y en forma multiplicativa:

$$y_{ij} = \mu + A_j + \sum_{k=1}^{q} \lambda_k \alpha_{ik} \gamma_{jk} \qquad i = 1, ..., g; \;\; j = 1, ..., a; \quad q = min(g-1, a)$$

Los parámetros multiplicativos, tanto en el modelo AMMI como en el SREG, se estiman por medio de la Descomposición en Valores Singulares (DVS) de la matriz que

contiene los residuos del modelo aditivo luego de ajustar por mínimos cuadrados el modelo de efectos principales. Generalmente los dos primeros términos multiplicativos son suficientes para explicar los patrones de la IGA, así como de G e IGA en forma conjunta; la variabilidad remanente se interpreta como ruido.

Gabriel (1971) presentó el concepto del biplot que consiste en la representación de las filas (individuos) y las columnas (variables) de una matriz de datos en un mismo gráfico. Éstos biplots, son herramientas poderosas para el análisis e interpretación de la estructura de datos provenientes de ensayos multiambientales utilizados en los programas de mejoramiento (Ebdon y Gauch, 2002; Samonte et al., 2004; Yan et al., 2000; Zobel et al., 1988). Del modelo AMMI se obtiene el biplot GE (*Genotipe-Environment*) (**CITA**) que permite interpretar la variación producida por los efectos de la IGA; mientras que, en el biplot GGE (*Genotipe* plus *Genotipe-Environment*) (Yan et al., 2000), derivado del modelo SREG, se analizan conjuntamente el efecto de G e IGA.

Dado que para seleccionar cultivares, el efecto de G e IGA debe considerarse simultáneamente, el modelo SREG resulta superior a AMMI para visualizar patrones en datos EMA. El biplot GGE permite investigar la existencia de megaambientes (grupo de ambientes en donde los cultivares de mejor desempeño son los mismos) entre los ambientes en estudio, seleccionar cultivares superiores en un megaambiente dado y seleccionar los mejores ambientes de evaluación para analizar las causas de la IGA.

Hablar un poco de los metodos de SVD del modelo SREG que da lugar a distintos graficos. Un oracion tipo dependiendo del escalado utilizado se pueden obtener distintas interpretaciones.... o no... no se... pero despues en resultados presentamos distintos escalados ... aunq pongo que vayan a leeer a tal autor nose...

3.1.2. Modelo AMMI robusto

El modelo AMMI, en su forma estándar, asume que no hay valores atípicos en el conjunto de datos. La presencia de *outliers* es más una regla que una excepción cuando se consideran datos agronómicos debido a errores de medición, algunas plagas / enfermedad que puede influir en algunos genotipos resultando por ejemplo en un rendimiento inferior al esperado en un ambiente, o incluso debido a alguna característica inherente de los genotipos que se evalúan.

Rodrigues et al. (2015) proponen una generalización robusta del modelo AMMI, que resulta de ajustar la regresión robusta basada en el estimador M-Huber (Huber, 1981) y luego utilizar un procedimiento DVS / PCA robusto. Consideraron varios métodos de DVS

/ PCA dando lugar a un total de cinco modelos robustos llamados: R-AMMI, H-AMMI, G-AMMI, L-AMMI, PP-AMMI.

El empleo de la versión robusta del modelo AMMI puede ser extremadamente útil debido a que una mala representación de genotipos y ambientes puede resultar en un mala decisión con respecto a qué genotipos seleccionar para un conjunto dado de ambientes (Gauch1997,Yanetal2000). A su vez, la elección de los genotipos incorrectos pueden provocar grandes pérdidas en términos de rendimiento. Los biplots obtenidos de los modelos robustos mantienen las características e interpretación estándar del modelo AMMI clásico (Rodrigues et al., 2015).

3.1.3. Métodos de imputación

Una limitación importante que presentan los modelos multiplicativos descriptos previamente es que requieren que el fenotipo de todas las combinaciones de genotipos y ambientes se encuentre registrado, es decir no admiten valores perdidos. Aunque los EMA están diseñados para que todos los genotipos se evalúen en todos los ambientes, la presencia de valores faltantes es muy común debido a errores de medición o pérdidas de plantas por animales, inundaciones o problemas durante la cosecha, además de la dinámica propia de la evaluaciones en las que se incorporan y se descartan genotipos debido a su pobre desempeño (Hill y Rosemberg, 1985)

Se han propuesto numerosas metodologías para superar el problema de valores ausentes en el conjunto de datos, entre las cuales se encuentran:

- EM-AMMI: Gauch y Zobel (1990) desarrollaron un procedimiento iterativo que utiliza el algoritmo de maximización de la esperanza (EM, del inglés *Expectation-Maximization*) incorporando el modelo AMMI.

- EM-SVD: Perry (2009a) propone un método de imputación que combina el algoritmo EM con DVS.

- EM-PCA: Josse y Husson (2013) proponen imputar los valores faltantes de un conjunto de datos con el modelo de Análisis de componentes principales.

- Gabriel Eigen: Arciniegas-Alarcón et al. (2010) propuso un método de imputación que combina regresión y aproximación de rango inferior usando DVS.

- WGabriel Eigen:

3.2. Paquete de R

Un paquete de R es la unidad básica para la distribución de código de R, que reúne de forma estructurada funciones, datos, archivos con documentación y testeos. Para la creación del mismo se deben seguir ciertas convenciones, existiendo elementos obligatorios y otros opcionales, entre los primeros se encuentran:

- Archivo DESCRIPTION: describe el contenido del paquete y establece cómo se va a relacionar con otros.
- Carpeta R: contiene el o los archivos de código de R con las funciones del paquete.
- Carpeta man: incluye archivos con la documentación del paquete, funciones y datasets.
- Archivo NAMESPACE: declara las funciones del paquete que se ponen a disposición de los usuarios y de qué funciones de otros paquetes hace uso.

Los elementos opcionales que se pueden agregar son por ejemplo:

- Carpeta data: contiene objetos de R que contienen datos.
- Carpeta vignettes: contiene los tutoriales que muestran ejemplos de uso del paquete, generalmente escritos en Rmarkdown.
- Carpeta tests: incluye código que permiten someter al paquete a diversos controles.

Para la creación del paquete, se deben instalar y cargar en la sesión de trabajo los paquetes: *devtools*, *usethis*, *roxygen2*, *ustestthat*, *knitr*, *available*. Además, en caso de utilizar el Windows se debe descargar e instalar Rtools .

3.2.1. Creación del paquete

En primer lugar se debe elegir el nombre del paquete, el cual debe cumplir con ciertas reglas: solo puede contener letras, números o puntos; tener al menos dos caracteres y empezar con una letra y no terminar con un punto. Una vez elegido el nombre, se debe chequear si el mismo está disponible en los repositorios *GitHub*, *CRAN* y *Bioconductor*, donde se alojan los paquetes. Para ello, se utiliza el paquete *available*, que además indicará si el nombre elegido tiene algún significado especial que podemos desconocer (revisa las webs de Wikipedia, Wiktionary y Urban Dictionary) (Figura 3.1).

```
# Cargar la libreria devtools
library(available)
# Crear el paquete geneticae
available("geneticae")
```

```
> library(available)
> available("geneticae")
Urban Dictionary can contain potentially offensive results,
  should they be included? [Y]es / [N]o:
1: Y
── geneticae ──────────────────────────────────────────────
Name valid: ✔
Available on CRAN: ✔
Available on Bioconductor: ✔
Available on GitHub:  ✔
Abbreviations: http://www.abbreviations.com/geneticae
Wikipedia: https://en.wikipedia.org/wiki/geneticae
Wiktionary: https://en.wiktionary.org/wiki/geneticae
Urban Dictionary:
  Not found.
Sentiment:???
>
```

Figura 3.1: Chequeo de disponibilidad del nombre elegido

Para la creación del paquete se utilizán *devtools* y *usethis* que incluyen funciones que simplifican la tarea. La función create_package("nombre_paquete") generará una carpeta con el nombre provisto (Figura 3.2). Si nose especifica una ubicación entonces se creará en el directorio actual.

```
# Cargar la libreria devtools
library(devtools)
# Crear el paquete geneticae
create_package("/home/julia-fedora/Escritorio/geneticae")
```

```
> # Cargar la libreria devtools
> library(devtools)
Loading required package: usethis
> # Crear el paquete geneticae
> create_package("/home/julia-fedora/Escritorio/geneticae")
✓ Creating '/home/julia-fedora/Escritorio/geneticae/'
✓ Setting active project to '/home/julia-fedora/Escritorio/geneticae'
✓ Creating 'R/'
✓ Writing 'DESCRIPTION'
Package: geneticae
Title: What the Package Does (One Line, Title Case)
Version: 0.0.0.9000
Authors@R (parsed):
    * First Last <first.last@example.com> [aut, cre] (YOUR-ORCID-ID)
Description: What the package does (one paragraph).
License: `use_mit_license()`, `use_gpl3_license()` or friends to
    pick a license
Encoding: UTF-8
LazyData: true
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.1.1
✓ Writing 'NAMESPACE'
✓ Writing 'geneticae.Rproj'
✓ Adding '.Rproj.user' to '.gitignore'
✓ Adding '^geneticae\\.Rproj$', '^\\.Rproj\\.user$' to '.Rbuildignore'
✓ Opening '/home/julia-fedora/Escritorio/geneticae/' in new RStudio session
✓ Setting active project to '<no active project>'
>
```

Figura 3.2: Creación del paquete geneticae

3.2.2. Archivos de código

Una vez creada la estructura del paquete se deben programar las funciones que el mismo contendrá. Cada una de ellas debe ser guardada en un archivo de extensión .R, en el subdirectorio R/. Para ello, se utiliza la función use_r() la cual crea un script ubicado en la carpeta R/, donde el código de interés será agregado.

A medida que se va desarrollando el paquete, con funciones internas y otras que se exportan, con algunas que se relacionan entre si y que a su vez dependen de otros paquetes, se deben ir realizando pruebas para asegurarse que los creados códigos realizan lo que realmente se desea. Para ello, la función load_all() simula el proceso de construcción, instalación y carga del paquete, permitiendo probar la función de manera interactiva.

Muy frecuentemente se utilizan funciones que se encuentran disponibles en otros paquetes. La función use_package() agrega el paquete de interés a la sección Imports del archivo DESCRIPTION, y luego para llamar a las mismas dentro de una función se debe utilizar @importFrom paquete función. Alternativamente, si se utilizan repetidamente muchas funciones de otro paquete, es posible importarlas todas utilizando @import paquete. Sin embargo, esta es la solución menos recomendada porque hace que el código sea más difícil de leer, y si tiene muchos paquetes, aumenta la posibilidad de que entren en conflicto nombres de funciones.

3.2.3. Documentación

Uno de los aspectos más importantes del paquete es la documentación donde se describe cómo se usa cada función, para qué sirven los argumentos, aclarar qué tipo de resultado devuelve, proveer ejemplos para el uso, etc. El paquete *roxygen2*, provee pautas para escribir comentarios con un formato especial que incluyan toda la información requerida justo antes de la definición de la función. El código y la documentación son adyacentes, de modo que cuando el código se modifique le exigirá que actualice la documentación.

El flujo de trabajo para crear la documentación con el paquete *roxygen2* es el siguiente:

- Agregar comentarios a los archivos .R. Estos deben comenzar con #', para distinguirlo de los comentarios regulares, y preceden a una función. La primera oración se convierte en el título y el segundo párrafo es una descripción de la función. Para el resto de los campos de la documentación, se utilizan de etiquetas que comienzan con @, siendo las más importantes a incluir:
  - @param: se detalla para qué sirve cada parámetro de la función.
  - @return: para explicar qué objeto devuelve la función.

- @details: agregar cualquier aclaración que se considere necesaria.
- @examples: incluir ejemplos de uso de la función.
- @export: indicar que esta función tiene que estar disponible cuando alguien cargue el paquete con library(). No es necesario exportar funciones auxiliares de utilidad interna.

- Ejecutar devtools::document() para convertir los comentarios de roxygen en archivos .Rd que compondrán el manual y que deben ir guardados en la carpeta man.

Roxygen2 permite utilizar la descripción de los parámetros de otras funciones usando @inheritParams. Esta documentará los parámetros que no están documentados en la función actual, pero que si lo están en la función fuente. La fuente puede ser una función en el paquete actual, vía @inheritParams function, u otro paquete, vía @inheritParams package::function. Además *Roxygen2* permite incluir referencias utilizando @references. En caso de importar paquetes, como se indicó en la sección anterior, se deben declarar usando @importFrom o @import, previo a la definición de la función.

3.2.4. Editar el archivo DESCRIPTION

El archivo DESCRIPTION provee toda la metadata sobre el paquete que se esta creando. En este archivo hay algunos campos que tienen que estar presentes de forma obligatoria y otros que son opcionales. Los elementos obligatorios son:

- Package: nombre del paquete
- Title: título del paquete (hasta 65 caracteres, Escrito De Esta Forma).
- Version: número de la versión actual del paquete (por ejemplo, 0.2.1)
- Author, Maintainer o Authors@R: quiénes han participado en el paquete.
- Description: un párrafo que describa el paquete.
- License: nombre de la licencia bajo la cual se distribuye el paquete. Si se pretende que cualquiera lo puede usar, entonces se debe recurrir a los tipos mas comunes de licencia para código abierto: CC0, MIT o GPL. Como se muestra en la Figura 3.3, el paquete geneticae se encuentra bajo la licencia GPL-3. Para esto, se utilizó la función use_gpl3_license() del paquete usethis, la cual agrega la información al archivo DESCRIPTION y además crea un archivo LICENSE al directorio del paquete.

En cambio, los elementos no obligatorios:

- Date: fecha de publicación de esta versión del paquete.
- Imports, Depends, Suggests: es muy común que las funciones desarrolladas necesiten hacer uso de algunas que pertenecen a otros paquetes. Estos serán indicados en los campos Imports, Depends, Suggests del archivo DESCRIPTION. Como se muestra

en la Figura 3.3, en el campo Imports se indica que geneticae necesita los paquetes: stats, GGEBiplots, ggforce, ggplot2, etc. Mientras que los listados en Suggest indica que se podría hacer uso de los mismos, aunque no son indispensables. Por último, el paquete geneticae se puede utilizar en versiones de R iguales o superiores a la 2.12, como se establece en Depends.

- URL: dirección de la página web del paquete.
- BugReports: dirección donde los usuarios pueden enviar avisos con los problemas que encuentren al utilizar el paquete.

El archivo DESCRIPTION del paquete geneticae se muestra en la Figura 3.3

```
Package: geneticae
Type: Package
Title: Statistical analysis tool for agricultural research
Version: 0.0.9000
Date: 13-04-2019
Author: Julia Angelini - Marcos Prunello - Gerardo Cervigni
Maintainer: Julia Angelini <jangelini_93@hotmail.com>
Description: Original idea was presented in the thesis to obtain the degree of
Bioinformatics, National University Rosario (UNR), Rosario Argentina. Some
experimental data for the examples come from INTA San Pedro and others research.
Geneticae offers extensive statistical analysis tool for agricultural and plant
breeding experiments, which can also be useful for other purposes.
License: GPL
Encoding: UTF-8
LazyData: true
NeedsCompilation: no
Repository: CRAN
RoxygenNote: 7.1.1
Imports: stats,
         GGEBiplots,
         ggforce,
         ggplot2,
         scales,
         MASS,
         pcaMethods,
         rrcov,
         dplyr,
         bcv,
         missMDA,
         calibrate,
         graphics,
         agridat,
         reshape2,
         matrixStats,
         tidyr,
         prettydoc
Suggests:
    knitr,
    rmarkdown,
    testthat (>= 2.1.0)
VignetteBuilder: knitr
Depends: R (>= 2.12.0)
URL: https://github.com/jangelini/geneticae
BugReports: https://github.com/jangelini/geneticae/issues

```

Figura 3.3: Archivo DESCRIPTION de geneticae

- HASTA ACA -

3.2.5. Testeos

Las pruebas resultan fundamentales en el desarrollo de paquetes, asegura que el código haga lo que realmente se desea. Existen pruebas informales como aquellas realizadas con la función load_all() que permite que las funciones creadas estén disponible rápidamente para uso interactivo. Sin embargo, las pruebas interactivas pueden convertirse en scripts reproducibles, los cuales resultan superiores debido a que se indica explícitamente cómo debería comportarse el código, provocando que los errores solucionados no vuelvan a ocurrir. Para ello, se utiliza la función usetestthat() del paquete *testthat* (Wickham,2011). Esta agrega testthat al campo Suggests en el archivo DESCRIPTION, crea un directorio tests/ para alojar cualquier tipo de unidad de testeo, una subcarpeta testthat donde se ubicaran los testeos escritos bajo este sistema y además, crea un archivo testthat.R, que se encarga de la ejecución de todos los testeos.

Los testeos se organizan en tres niveles:

- Archivo de tests: uno por cada archivo .R en la carpeta R/.
- Ejecutar pruebas automáticamente cada vez que algo cambie con la función autotest(). Estas son útiles cuando las pruebas se ejecutan con frecuencia. Si se modifica un archivo de prueba, probará ese archivo; si se modifica un archivo de código, volverá a cargar ese archivo y volverá a ejecutar todas las pruebas.
- Expectation: es el nivel más desagregado, corre cierto código y se compara el resultado obtenido con el esperado.

La función use_test(), creará los archivos de prueba cuyo nombre tienen que ser testnombre_archivo_de_codigo.R y los ubicará en la carpeta test/testthat. Una vez escritos estos archivos, podemos evaluar los resultados de los testeos con devtools::test(). Ante cada error encontrado, nos detenemos para corregirlo y repetimos este proceso hasta que todas las unit tests pasen la prueba.

Una medida de la calidad de un paquete está dada por el porcentaje de su código que es evaluado durante los testeos. El paquete covr permite hacer ese cálculo, además de mostrar interactivamente qué partes del código fueron evaluadas y cuáles no.

```
# Evaluar cobertura del archivo abierto actualmente
devtools::test_coverage_file()
# Evaluar cobertura de todo el paquete
devtools::test_coverage()
```

PONER IMAGEN

3.2.6. Compilación e instalación

La función check() o R CMD check ejecutado en el shell, es utilizado para verificar que un paquete R esta en pleno funcionamiento. La misma verificará que no haya errores de sintaxis o no se generen warnings. Está compuesto por más de 50 chequeos individuales entre los cuales se encuentran: la estructura del paquete, el archivo descripción, namespace, el código de R, los datos, la documentación, entre otros. Se aconseja realizar verificaciones completas de que todo funciona a medida que se van incorporando funciones ya que si se incorporan muchas y luego se verifican será difícil identificar y resolver los problemas. Una vez que se desarrollaron todos los elementos necesarios para el paquete y no se detectan errores, advertencias o notas, se ejecuta la función install(), con el objetivo de instalar el paquete en la biblioteca.

3.2.7. Algunos elementos complementarios

Existen algunas componentes que no son obligatorias pero que .....

3.2.7.1. Viñetas

A diferencia de la documentación, en la cual se detalla como se utiliza cada una de las funciones del paquete, una viñeta es una descripción el problema que el paquete está diseñado para resolver y muestra al lector cómo resolverlo.

Muchos de los paquetes existentes tienen viñetas la cuales se pueden encontrar utilizando la función browseVignettes("packagename") si el mismo se encuentra instalado, sino deben consultarse en su página de CRAN, por ejemplo para el paquete *dplyr*: `http://cran.r-project.org/web/packages/dplyr`. Cada viñeta proporciona el archivo fuente original, una página HTML o PDF y un archivo de código R.

Las Viñetas se pueden construir de diversas formas, en este trabajo se utiliza se utiliza usethis::use_vignette("my-vignette"). La misma crea un directorio vignettes/, agrega las dependencias necesarias a DESCRIPTION y redacta la viñeta. Las tres componentes fundamentales de la misma son las siguientes:

- El bloque inicial de metadatos, que contiene la siguiente información:

```
---
title: "Vignette Title"
output: rmarkdown::html_vignette
vignette: >
```

```
%\VignetteIndexEntry{Vignette Title}
%\VignetteEngine{knitr::rmarkdown}
\usepackage[utf8]{inputenc}
---
```

- Markdown para formatear texto.
- Knitr para interpretar texto, código y resultados.

3.2.7.2. Archivo README

Un archivo README es nua forma de documentacion de software. Usualmente es un archivo de texto plano que permite describir brevemente por qué y para qué alguien tendría que usar el paquete, a la vez que indicar cómo conseguirlo o instalarlo.

Para generar el readme con R Markdown se utiliza la función use_readme_rmd() la cual crea un archivo de Rmarkdown con una plantilla donde se escribirá el README y la agrega a .Rbuildignore. Luego, al compilarlo con knitr se obtendrá un archivo README.md, que será la cara visible de nuestro paquete si, por ejemplo, lo subimos a GitHub donde los README vienen a tener el rol de portada en cierta forma.

3.2.7.3. Archivo NEWS

Mientras que el README apunta a ser leído por nuevos usuarios, el archivo NEWS es para la gente que ya usa el paquete. Este archivo se encarga de contar qué tenemos en cada versión nueva del paquete que publicamos: lo nuevo, lo que cambió y lo que se eliminó. Se sugiere usar markdown para escribir este archivo y colocar un título principal para cada versión, seguido por títulos secundarios que describen lo realizado (cambios principales, bugs arreglados, etc.). Si se trata de cambios impulsados por otras personas, por ejemplo, a través de sugerencias hechas en GitHub, se los menciona. Una buena práctica es ir escribiendo este archivo cada vez que se realiza algo nuevo en el paquete. La función que nos permite crear este archivo automáticamente es usethis::use_news_md().

3.2.7.4. Agregar datasets

A menudo es útil incluir datos en un paquete a fin de proporcionar ejemplos de aplicaciones de las funciones incluidas en él. Ellos se almacenan en el directorio data/, siendo cada archivo un .RData que sólo contiene un objeto. Para esto, se utiliza la función usethis::use_data(). Notar que el archivo DESCRIPTION creado con la función create_package(), mencionada anteriormente, contiene el campo LazyData: true, lo cual genera que los conjuntos de datos no ocupen memoria hasta que sean usados.

Los objetos en la carpeta data siempre se exportan, por lo cual hay que agregar documentación para los mismos. A diferencia de las funciones que son documentadas directamente, para los objetos en data/, se debe crear un archivo y guardarlo en el directorio R/. Esto se puede hacer con roxygen en cualquier Rscript de la carpeta R, aunque se acostumbra juntar toda la documentación para todos los datasets en un único archivo llamado data.R.

3.2.7.5. Añadir badges

Las insignias o badges son unos íconos que señalan distintas características del paquete, como su nivel de maduración, el nivel de cobertura en el testeo, cantidad de descargas, número de versión, resultado de los controles de CRAN, etc. Son visualmente muy atractivas y se colocan el en archivo README. El paquete usethis trae un conjunto de funciones que generan automáticamente el código a incluir en el README.Rmd para agregar los badges:

use_badge(badge_name, href, src)

use_cran_badge()

use_bioc_badge()

use_lifecycle_badge(stage)

use_binder_badge(sturlpath = NULL)

3.2.7.6. Diseñar un logo

Seguramente has notado la costumbre de que cada paquete de R tenga su logo con forma hexagonal, que generalmente termina en forma de sticker: los hexStickers. Bueno, para terminar de darle identidad a tu paquete y hacerlo más vistoso, podés diseñar tu logo. Por suerte, también hay un paquete que permite hacerlo sin demasiadas complicaciones: el paquete HexSticker. Una vez creado el logo, le podemos pasar su ubicación a la función use_logo(), que se encargará de darle el tamaño adecuado, guardarlo en la carpeta man del paquete y producir el código de markdown para incluirlo en el README.

3.2.7.7. Crear una página web

Si llegaste hasta acá siguiendo todos los pasos anteriores, tenés en tu haber un montón de material muy bueno sobre tu paquete: páginas de ayuda, ejemplos, tutoriales, novedades sobre los cambios, logo, insignias, etc. Suficiente como para crear una página web y que

tu paquete tenga presencia real en el más allá. No sé cuántas veces dije "por suerte" en este material, pero lo voy a decir una vez más. Por suerte existe un paquete que se encarga de tomar todo el material hecho anteriormente y convertirlo en una página web AUTOMÁTICAMENTE. Sumado a que GitHub nos da lugar para hospedar nuestras páginas de manera libre y gratuita, no hay excusas para no hacerlo. Claro que cuanto más quieras personalizar tu web, más vas a tener que explorar algunas opciones e incluso toquetear algo de código de html, pero esto no es necesario, ya que el aspecto logrado por default es muy satisfactorio. El paquete responsable de esto es pkgdown y lo único que hay que hacer es correr pkgdown::build_site() desde el directorio de nuestro paquete cada vez que publiquemos una nueva versión. Más información y detalles en https://pkgdown.r-lib.org/index.html.

3.2.7.8. Publicación

Un repositorio es el lugar dónde se encuentran alojados los paquetes y desde el cuál los usuarios pueden descargarlos. Entre los repositorios más populares de paquetes R se encuentran:

- **CRAN**: es el principal repositorio de paquetes de R, está coordinado por la fundación R. Previa a la publicación en este repositorio el paquete debe pasar por diferentes pruebas para asegurar que cumple con las políticas de CRAN.
- **Bioconductor**: se trata de un repositorio específico para bioinformática. Del mismo modo que CRAN, tiene sus propias políticas de publicaciones y procesos de revisión.
- **GitHub**: a pesar que no es específico para R, github es con toda seguridad el repositorio más popular para la publicación de proyectos *open source* (del inglés, código abierto). Su popularidad procede del espacio ilimitado que proporciona para el alojamiento de proyectos *open source*, la integración con git (un software de control de versiones) y, la facilidad de compartir y colaborar con otras personas. Una de sus desventajas es que no proporciona procesos de control.
- **R-Forge** y **RForge**: son entornos de desarrollo de paquetes y repositorios. Eso significa que incluyen control de fuente, seguimiento de errores y otras características. Puede obtener versiones de desarrollo de paquetes de estos.

El paquete *geneticae* se encuentra en GitHub, para instalar el mismo se deben seguir las siguientes instrucciones:

```
library(devtools)
install_github("jangelini/geneticae")
```

Por otro lado, se crea también una página web[1] para el paquete utilizando *pkgdown*,

[1]Para visitar la página web del paquete debe dirigirse a `https://.....`

mediante la función pkgdown::build_site(). En ella se podrá encontrar una breve descripción del paquete, las funciones que incluyen los mismos, la vignette, las distintas versiones del paquete, entre otras cosas.

3.3. Shiny APP

Shiny es un paquete R para crear aplicaciones web interactivas sin necesidad de conocer en profundidad los lenguajes HTML / CSS / JavaScript . Estas aplicaciones constituyen una interfaz gráfica entre el usuario y R, que permiten realizar un análisis a través de un navegador web sin necesidad de programar.

El esquema interno de una Shiny APP puede observarse en la Figura 3.4. Las mismas están compuestas por la interfaz de usuario, ui (*user interfaz*), que controla el diseño de la aplicación, recibe los inputs y muestra los outputs en el navegador; el server que contiene las funciones de R con las instrucciones neesarias para obtener los resultados de los análisis incluidos en la aplicación; y shinyApp es la función que crea objetos de aplicación Shiny a partir de ui / servidor.

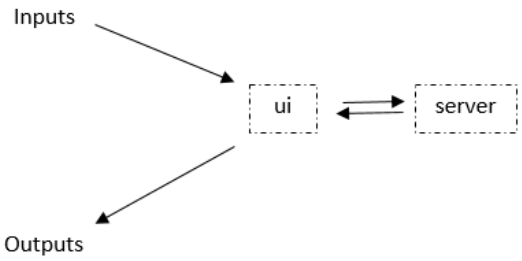


Figura 3.4: Esquema interno de la aplicación.

3.3.1. Desarrollo de Shiny APP

Una forma de desarrollar una aplicación es a partir de un nuevo directorio con un sólo archivo llamado app.R, como se muestra a continuación.

```
library(shiny)
ui<- ...
server<- ...
shinyApp(ui = ui, server = server)
```

En este archivo se carga el paquete shiny, se define la interfaz de usuario y la función server y por último, se ejecuta función que permite construir e iniciar una aplicación. Al ejecutar la aplicación la misma aparecerá, de manera predeterminada, en una ventana

emergente. Sin embargo, otras dos opciones se pueden configurar desde el menú desplegable de *Run App*. Una de ellas es la ejecución en el panel del visor que permite verla al mismo tiempo que ejecuta el código. La segunda opción es ejecutar en un navegador externo mostrando la aplicación como la mayoría de los usuarios la verán. Dado que la sesión de R estará monitoreando la aplicación y ejecutando las ordenes dadas por el usuario, no se podrá ejecutar ningún comando.

En cualquier lenguaje de programación tener el código duplicado genera un desperdicio computacional y, lo que es más importante, aumenta la dificultad de mantener o depurar el código. Cuando se programa en R, se utilizan dos técnicas para lidiar con el código duplicado: guardar un valor usando una variable o utilizar una función para almacenar un cálculo. Ninguno de estos enfoques son apropiados en una Shiny APP, sino que se utilizan expresiones reactivas. Una expresión reactiva tiene una diferencia importante con una variable: sólo se ejecuta la primera vez que se llama y luego almacena en caché el resultado de la misma hasta que necesite actualizarse. La programación reactiva es un estilo de programación que enfatiza valores que cambian con el tiempo, y cálculos y acciones que dependen de esos valores. Esto es importante para las aplicaciones Shiny porque son interactivas: los usuarios cambian los inputs, lo que hace que la lógica se ejecute en el servidor que finalmente resultan en actualización de los outputs/resultados.

Entre los problemas que pueden surgir al crear una Shiny app se encuentran los errores inesperados, no se obtiene ningún error pero el valor obtenido es incorrecto, o bien todos los resultados son correctos, pero no se actualizan cuando se deben. Una vez localizada la fuente del error, la herramienta más poderosa es el depurador interactivo, éste detiene la ejecución y brinda una consola interactiva donde puede se ejecutar cualquier código para descubrir el error. Para iniciar el mismo, se puede agregar la función browser() en el código fuente, o bien agregar un punto de interrupción RStudio haciendo clic a la izquierda del número de línea.

Al modificar la aplicación, se la ejecuta para poder ver los cambios realizados, por lo tanto resulta esencial reducir la velocidad de iteración. La primera forma acelerar el proceso consiste en escribir el código, utilizar el atajo del teclado Cmd/Ctrl+ Shift+ Enter en lugar del botón "Ejecutar aplicación", experimentar interactivamente con la aplicación y cerrar la aplicación, repitiendo este proceso al realizar cualquier cambio. Otra forma de reducir aún más la velocidad de iteración es activar la recarga automática (options(shiny.autoreload = TRUE)) y luego ejecutar la aplicación en un trabajo en segundo plano. Con este flujo de trabajo cuando se guarde un archivo, su aplicación se reiniciará: no es necesario cerrarla y reiniciarla, lo cual conduce a un flujo de trabajo aún más rápido. La principal desventaja de esta técnica es que debido a que la aplicación se ejecuta en un proceso separado, es considerablemente más difícil de depurar.

3.3.2. Compartiendo una Shiny Web App

Una vez creada la aplicación se la publica para su libre uso. En este caso la Shiny Web App encuentra disponible en el servidor de CONICET `www.cefobi.com`. Además el proyecto se encuentra en GitHub `https://github.com/jangelini/shinyAPP_geneticae`.

Capítulo 4

Resultados

4.1. Paquete de R *geneticae*

Una vez instalado el paquete, se debe cargar en la sesion de R mediante el comando: library(geneticae). Información detallada sobre las funciones del paquete geneticae se puede obtener mediante help(package = "geneticae"). La ayuda para una función, por ejemplo imputation(), en una sesión R se puede obtener usando *?imputation* o help(imputation). La función browseVignettes("geneticae") permite obtener la viñeta del paquete, es decir una descripción el problema que está diseñado para resolver así como ejemplos de aplicación del mismo.

Se encuentra disponible una página web del paquete (http://....) en la que se cuenta con una breve descripción del paquete, las funciones que se incluyen en él, la viñeta, un enlace de acceso a la shiny app, entre otra información.

4.1.1. Conjuntos de datos en geneticae

El paquete geneticae contiene dos conjuntos de datos que son usados a modo de ejemplo en las funciones incluidas para analizar los datos provenientes de EMA.

- *yan.winterwheat dataset* (Wright, 2018): rendimiento de 18 variedades de trigo de invierno cultivadas en nueve ambientes en Ontario en 1993. A pesar de que el experimento contaba con cuatro bloques o réplicas en cada ambiente en el conjunto de datos, sólo el rendimiento medio para cada combinación de variedad y ambiente se encuentra disponible.

```
data(yan.winterwheat)
head(yanwinterwheat)
```

```
##   gen  env yield
## 1 Ann BH93 4.460
## 2 Ari BH93 4.417
## 3 Aug BH93 4.669
## 4 Cas BH93 4.732
## 5 Del BH93 4.390
## 6 Dia BH93 5.178
```

- *plrv dataset* (CITA AGRICOLAE): rendimiento, peso de planta y parcela de 28 genotipos en 6 ubicaciones en Perú, con el fin de estudiar la resistencia a PLRV (*Patato Leaf Roll Virus*) causante del enrollamiento de la hoja. Cada clon fue evaluado tres veces en cada ambiente.

```
data(plrv)
head(plrv)
```

```
##   Genotype Locality Rep WeightPlant WeightPlot    Yield
## 1   102.18     Ayac   1   0.5100000       5.10 18.88889
## 2   104.22     Ayac   1   0.3450000       2.76 12.77778
## 3   121.31     Ayac   1   0.5425000       4.34 20.09259
## 4   141.28     Ayac   1   0.9888889       8.90 36.62551
## 5   157.26     Ayac   1   0.6250000       5.00 23.14815
## 6    163.9     Ayac   1   0.5120000       2.56 18.96296
```

4.1.2. Aplicación de las funciones incluidas en el paquete

A fin de ilustrar la metodología incluida en el paquete se utiliza el conjunto de datos *yan.winterwheat*.

Modelo SREG

Para visualizar conjuntamente el efecto de G e IGA, Yan et al. (2000) propuso el biplot GGE con el cual se pueden abordar visualmente diversos aspectos relacionados con la evaluación de genotipos y ambientes. Para obtener dicho biplot en primer lugar se debe ajustar el modelo SREG mediante la función GGEmodel().

La función GGEmodel() propuesta en este paquete, es menos restrictiva que las disponibles actualmente en R en cuanto al conjunto de datos de entrada. En particular, es una modificación de la función GGEModel() de GGEBiplots **(CITA)** en la que el con-

junto de datos de entrada los genotipos se encuentren en las filas y los ambientes en las columnas y por lo tanto, no admite replicas. A diferencia de esto, la función GGEmodel() incluida en *geneticae* requiere que los datos se encuentren en formato largo, es decir las observaciones en las filas y las variables en las columnas, lo cual es más ordenado a la hora de registrar la información **(CITA)**. Además, la base de datos puede contener otras variables que no serán utilizadas en el análisis ya que al ajustar el modelo se deben indicar en qué columnas se encuentra la información requerida por la técnica que se aplicará. Por otro lado, dado que el modelo requiere de una única observación para cada combinación de genotipo y ambiente, en caso de existir repeticiones el valor fenotípico promedio es calculado automáticamente por la función antes de ajustar el modelo. Los valores faltantes no están permitidos, en caso de haber, un mensaje de error saldrá en la consola de R.

La sentencia utilizada para ajustar el modelo GGE en el conjunto de datos *yan.winterwheat* se muestra a continuación. El primer argumento de la misma consiste en el nombre del *dataframe* y los restantes indican los nombres que reciben las columnas que contienen la información de los genotipos, ambientes y del rasgo fenotípico de interés. Por defecto, la función considera que no hay replicas en el conjunto de datos, sin embargo, si existieran la opción *rep* en la cual se indica el nombre de la columna con dicha información debe ser adicionada. Otros argumentos de dicha función son el método de centrado, de SVD y de escalado. Por defecto los datos se centran por G y GE, dando lugar al modelo SREG, otra opción en dicho argumento dará lugar a un modelo diferente. La elección del método de SVD no altera las relaciones o interacciones relativas entre los genotipos y los ambientes, aunque la apariencia del biplot será diferente, se recomienda la lectura de Yan (2002). Por último, diferentes biplots se pueden generar dependiendo del método de escalado, mayor información se encuentra disponible en Yan and Kang (2003) y Yan and Tinker (2006). En el modelo ajustado a continuación tanto el centrado, como el método SVD y de escalado son los seteados por defecto en la función.

```
GGE1 <- GGEmodel(yanwinterwheat, genotype = "gen", environment = "env", response
    = "yield")
```

Biplot GGE

La salida de la función GGEmodel() es una lista que contiene las coordenadas para los genotipos y ambientes de todas las componentes, el vector de valores propios de cada componente, la variancia total, el porcentaje de variancia explicada por cada componente, entre otros. Utilizando dicha información como entrada, la función GGEPlot() construye el biplot GGE. En dichos gráficos los cultivares se muestran en minúscula, para diferenciarlos de los ambientes, que están en mayúsculas. La primer componente principal se usa como la

abscisa y la segunda como ordenada. Además, es posible adicionar el método de centrado, escalado y de SVD utilizado, así como también el porcentaje G + GE explicado por los dos ejes como una nota al pie del biplot con la opción *footnote=T*, así como también el título del gráfico con *titles = T*.

Comparaciones simples utilizando GGE biplot

Dependiendo del objetivo del fitomejorador mejorador, diversos tipos de biplots GGE se pueden construir utilizando la función GGEPlot(). Utilizando la opción *Biplot* en el argumento *type* se obtiene un biplot básico como se muestra en la figura 4.1. Se observa que explica el 78 % de la variabilidad total de G + GE. Para interpretar este gráfico se consideran los ángulos entre los vectores de los genotipos y los ambientes. Así se ve, por ejemplo, que el genotipo kat presenta un rendimiento por debajo del promedio en todos los ambientes, al tener un ángulo mayor a 90° con todos los ambientes. Por otro lado, fun presenta un rendimiento por encima de la media en todos las localidades salvo en OA93 y en KE93, ya que los ángulos son agudos. La longitud de los vectores ambientales dan una medida de la capacidad del medio ambiente para discriminar entre los cultivos.

```
GGEPlot(GGE1, type = "Biplot", footnote = F, titles = F)
```

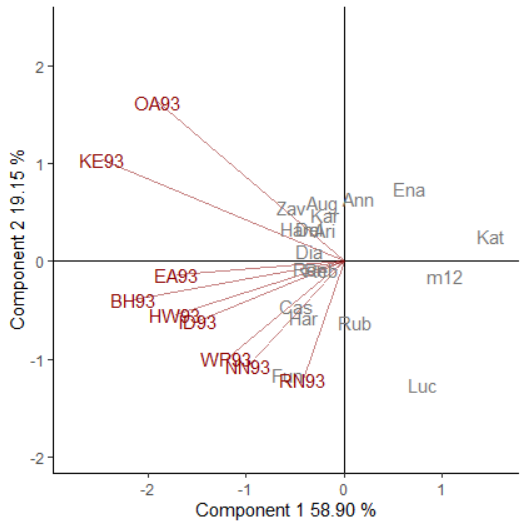


Figura 4.1: Biplot básico obtenido de la función GGEPlot()

Los mejoradores en general están interesados en identificar los cultivares más adaptados a su área, lo cual es posible a través del biplot GGE. Para esto, Yan y Hunt (2002) sugieren constituir un eje del ambiente de interés, por ejemplo OA93, trazando una recta que una el identificador del ambiente y el origen de coordenadas. Los genotipos se clasifican en función del rendimiento en dicho ambiente de acuerdo con sus proyecciones, en

la dirección indicada por el eje OA93 (Figura 4.2). Para ello, se indica la opción *Selected Environment* en el argumento *type* de la función y además el ambiente a evaluar en el argumento *selectedE*. Se observa en este caso que el cultivar de mayor rendimiento fue es Zav seguido por Aug, Ham, y así sucesivamente hasta llegar al genotipo Luc, que es el de menor rendimiento en ese ambiente. El eje perpendicular al del ambiente de interés, separa los genotipos con rendimiento mayor al promedio, de Zav a Cas, de aquellos con valores inferior a la media, de Ema a Luc, en OA93.

En forma similar, el ambiente más adecuado para un cultivar es posible determinarlo graficando una línea que una el origen de coordenadas y el marcador del genotipo de interés, por ejemplo Kat (Figura 4.2) (Yan y Hunt (2002)). Los ambientes se clasifican a lo largo del eje del genotipo en la dirección indicada por la flecha. Para obtener este gráfico la opción *Selected Genotype* debe indicarse en el argumento *type*, y el genotipo de interés en *selectedG*. El eje perpendicular al del genotipo separa los ambientes en los que Kat presentó un rendimiento por debajo de su promedio, en todos los ambientes estudiados.

```
# Ranking de cultivares en el ambiente OA93
GGEPlot(GGE1, type = "Selected Environment", selectedE = "OA93", footnote = F, titles
    = F)

# Ranking de ambientes para cultivar Kat
GGEPlot(GGE1, type = "Selected Genotype", selectedG = "Kat", footnote = F, titles = F)
```

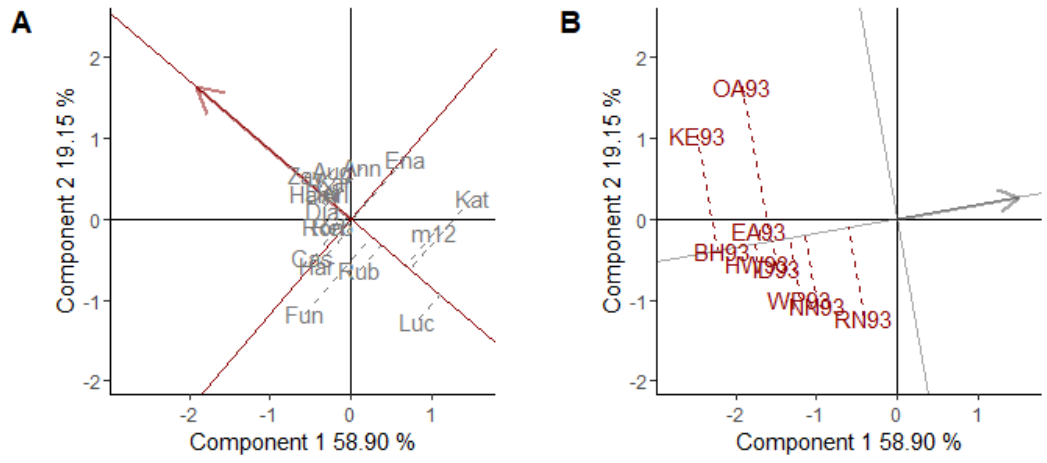


Figura 4.2: A: Ranking de cultivares en el ambiente OA93. B: Ranking de ambientes para cultivar Kat

Para comparar dos cultivares, por ejemplo Kat y Cas, una linea recta que una a los genotipos a comparar se debe trazar y luego una perpendicular a la anterior (figura 4.3). Este biplot te obtiene colocando *Comparison of Genotype* en el argumento *type* y los

genotipos de interés en *selectedG1* y *selectedG2*. Se observa que todos las localidades se encuentran del mismo lado de la línea perpendicular que Cas, indicando que fue más rendidor que Kat en todos los ambientes.

```
GGEPlot(GGE1, type = "Comparison of Genotype", selectedG1 = "Kat", selectedG2 = "
    Cas", footnote = F, titles = F)
```

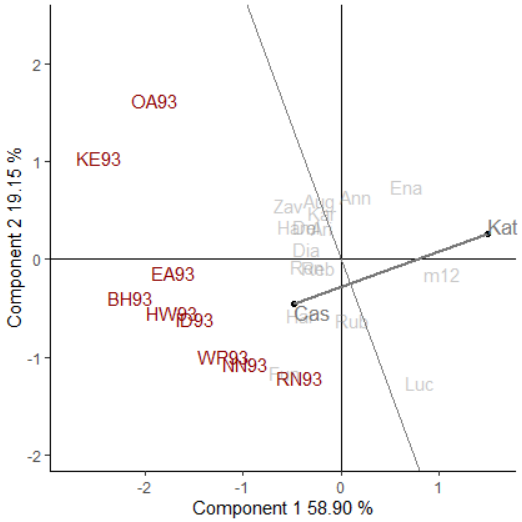


Figura 4.3: Comparación entre dos genotipos obtenido de la función GGEPlot()

Identificación de mega-ambientes con GGE biplot

La vista poligonal del biplot GGE, obtenida al indicar *Which Won Where/What* en el argumento *type*, proporciona un medio eficaz de visualización del patrón "quíen ganó dónde"de un conjunto de datos EMA (Figura 4.4). El polígono se obtiene uniendo los cultivares (fun, zav, ena, kat y luc) que se encuentran más alejados del origen de coordenadas, de modo que todos los restantes se encuentren contenidos en el polígono. La distancia de los cultivares respecto del origen de coordenadas, en sus respectivas direcciones, es una medida de la capacidad de respuesta a los ambientes. Los ubicados en los vértices son los más alejados, por lo tanto son los cultivares que más responden, mientras que los que se encuentran en el origen de coordenadas no responden en absoluto a los ambientes estudiados.

Las perpendiculares a los lados del polígono dividen al biplot en mega-ambientes, siendo el cultivar de mayor rendimiento en todos los ambientes que se encuentran en él aquel que se encuentra en el vértice de dicho sector. Por un lado, se observa que OA93 y KE93 conforman un mega-ambiente y que Zav es el mejor cultivar. Otro está formado por el resto de los ambientes, al cual llamaremos ME1 en futuros análisis, siendo Fun el que se encuentra en el vértice. En el sector con ena, kat y luc en los vértices del polígono no se

observó ningún ambiente, lo cual indica que estos cultivares fueron los menos rendidores en algunos o todos los ambientes considerados.

Se requieren dos criterios para sugerir la existencia de diferentes mega-ambientes (Gauch and Zobel, 1997). Primero, diferentes variedades superiores en los diferentes ambientes estudiados; segundo, la variación entre grupos debería ser significativamente mayor que la variación dentro del grupo. Ambos criterios se cumplen en el presente caso (Figura 4.4). La sugerencia de dos mega-ambientes coincide con la distribución geográfica de los ambientes. La ubicación de OA (Ottawa) y KE (Kemptville) se extiende hacia el este de Ontario; BH (Bath) a pesar de pertenecer a la misma zona, tiene un clima mucho más cálido que las otras dos localidades. Las otras seis localidades consideradas se ubican al oeste o sur de la provincia.

GGEPlot(GGE1, type = "Which Won Where/What", footnote = F, titles = F)

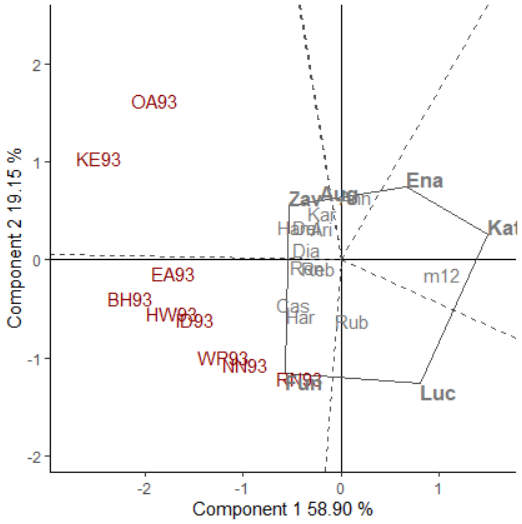


Figura 4.4: Identificación del mejor cultivar en cada ambiente a partir de la función GGEPlot()

Evaluación de los cultivos dentro de un mega-ambientes con GGE biplot

Una vez identificado los mega-ambientes, el siguiente paso es seleccionar cultivares dentro de cada uno de ellos. De acuerdo con la figura 4.4, zav es el mejor cultivar para los ambientes en uno de los mega-ambiente y fun para el otro. Sin embargo, los fitomejoradores no seleccionarán un único cultivar en cada mega-ambiente, sino que es necesario evaluar todos los cultivares con el fin de conocer su desempeño (rendimiento y estabilidad).

El biplot GGE, particularmente enfocando la SVD en los genotipos, es decir utilizando el argumento *SVP=row* en la función GGEmodel(), proporciona un medio superior para visualizar tanto el rendimiento medio como la estabilidad de los genotipos (Figura 4.5).

La visualización del rendimiento medio y la estabilidad de los genotipos a partir de un eje promedio de cada uno de los mega-ambientes formados (AEC, average environment coordinate), por ejemplo ME1, se logra indicando *Mean vs. Stability* en el argumento *type* (Figura 4.5). Mientras que la abscisa representa el efecto de G la ordenada el de la IGA, que es una medida de la variabilidad o inestabilidad, asociada con cada genotipo. Una mayor proyección sobre la ordenada AEC, independientemente de la dirección, significa mayor inestabilidad. Por lo tanto, rub y dia son más variables y menos estables que otros cultivares (Figura 4.5). El ambiente BH93 está ubicado en el este de Ontario caracterizado por inviernos más fríos, mientras que RN93 en el sur de Ontario caracterizado por un clima cálido y húmedo en la mayoría de los años. Esto justifica la inestabilidad de rub y dia, ya que el primero es temprano y tiene una poca resistencia al invierno por lo que se desempeñó bien en RN93 y mal en BH93, y el segundo, con un comportamiento contrario al anterior cultivar debido a que es tardío y alto. Los cultivares próximos al eje de las abscisas, cas, zav, reb, del, ari y kar, son más estables que el resto.

Además, usando el mismo enfoque en SVD, para cada uno de los mega-ambientes identificados, es posible visualizar el rendimiento medio y la estabilidad de los genotipos en una única medida. Un cultivar ideal es definido y a pesar de que rara vez exista, se utiliza como referencia para la evaluación de los cultivares. La distancia entre los cultivares con respecto al ideal, ubicado en el centro del conjunto de círculos concéntricos, se puede utilizar como una medida de su conveniencia. Es posible realizarlo con la función GGEPlot() indicando *Ranking Genotypes* en el argumento *type* (Figura 4.5). Para el ME1, el cultivar fun es el más próximo al ideal, y por lo tanto, el más deseable de todos los estudiados, seguido por cas y heno, que a su vez son seguidos por ron, ham, rub, zav, del y reb, etc. (figura 4.5).

```
# Modelo SREG enfocando SVD en los genotipos
GGE_Gpartition <- GGEmodel(ME1, genotype="gen", environment="env", response="
    yield", SVP="row")

# Visualizacion del rendimiento medio y la estabilidad
GGEPlot(GGE_Gpartition, type = "Mean vs. Stability", footnote = F, titles = F)

# Ranking de los genotipos respecto a uno ideal
GGEPlot(GGE_Gpartition, type = "Ranking Genotypes", footnote = F, titles = F)
```

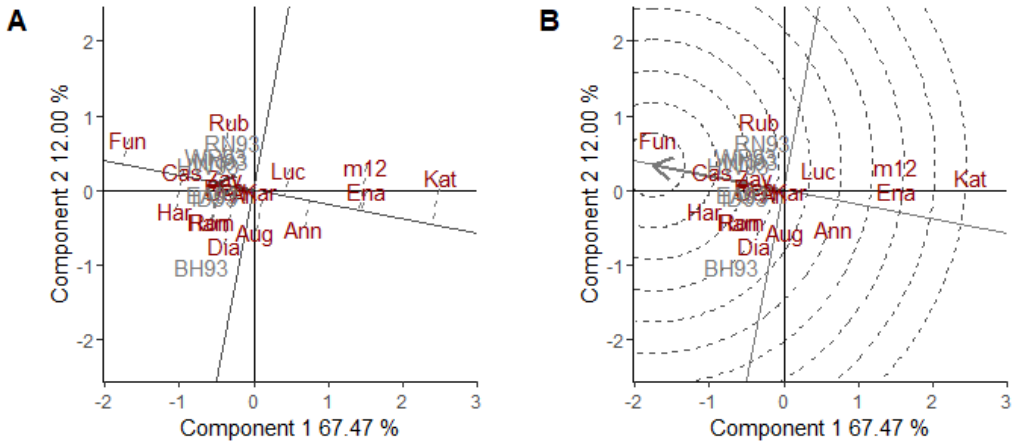


Figura 4.5: A: Evaluación de los cultivares con base en el rendimiento promedio y la estabilidad. B: Clasificación de genotipos con respecto al genotipo ideal

Evaluación de los ambientes con GGE biplot

A pesar de que el objetivo principal de los EMA es seleccionar cultivares también es posible evaluar los ambientes, enfocando la SVD en los ambientes. Puede resultar de interés determinar si la región objetivo pertenece a uno o varios mega-ambientes; identificar mejores ambientes de prueba; establecer ambientes redundantes que no brindan información adicional sobre los cultivares asi como determinar ambientes que pueden usarse para la selección indirecta.

En la figura 4.6 se observa que los ambientes están conectados con el origen de coordenadas a través de vectores, permitiendo comprender las interrelaciones entre los distintos ambientes del oeste y sur de Ontario que forman el mega-ambiente. Esta visualización del biplot GGE se obtiene enfocando la SVD en los ambientes con la opción *column* del argumento *SVP* en GGEmodel() y luego, indicando en GGEPlot() *Relationship Among Environments* (Figura 4.6). El coseno del ángulo entre los vectores de dos ambientes se aproxima al coeficiente de correlación entre ellos. Por ejemplo, NN93 y WP93 tienen un ángulo de aproximadamente 10° entre sus vectores; por lo tanto, se encuentran estre-

chamente relacionados; por le contrario RN93 y OA93 presentan una correlación leve y negativa ya que el ángulo que forman sus vectores supera los 90°. Por lo tanto, la Figura 4.6 ayuda a identificar ambientes redundantes. Si algunos de los ambientes tienen ángulos pequeños y, por lo tanto, están altamente correlacionados, la información sobre los genotipos obtenidos de estos ambientes debe ser similar. Si esta similitud es repetible a través de los años, estos ambientes son redundantes y por lo tanto, uno solo debería ser suficiente. Obtener la misma o mejor información utilizando menos ambientes reducirá el costo y aumentará la eficiencia de producción.

La capacidad de discriminación así como la representatividad respecto del ambiente objetivo, son medidas fundamentales para un ambiente. Si no tiene capacidad de discriminación, no proporciona información sobre los cultivares y, por lo tanto, carece de utilidad. A su vez, si no es representativo no sólo que carece de utilidad sino que también puede proporcionar información sesgada sobre los cultivares evaluados. Para visualizar estas medidas, se define un ambiente promedio (AEC mencionado anteriormente) y el ambiente ideal como el centro de un conjunto de círculos concéntricos (Figura 4.6). Para obtener este biplot, se debe enfocar la SVD en los ambientes, como se indicó con anterioridad, y luego, *Ranking Environments* en (GGEPlot() (Figura 4.6). El ángulo entre el vector de un ambiente y el eje proporciona una medida de la representatividad. Por lo tanto, EA93 e ID93 son los más representativos, mientras que RN93 y BH93 son los menos representativos del ambiente promedio, cuando se analiza el mega-ambiente que no contiene a OA93 y a KE93 4.6. Por otro lado, para ser discriminativo debe estar cercano al ambiente ideal. HW93 es el ambiente más cercano al ideal y, por lo tanto, es el más deseable del ME1, seguido por EA93 e ID93, que a su vez son seguidos por WP93 y NN93. RN93 y BH93 fueron los ambientes de prueba menos deseable de dicho megamabiente.

```
# Modelo SREG enfocando SVD en los ambientes
GGE_Epartition <- GGEmodel(ME, genotype="gen", environment="env", response="
    yield", SVP="column")

# Relacion entre ambientes
GGEPlot(GGE_Epartition, type = "Relationship Among Environments", footnote = F, titles
    = F)

# Clasificacion de ambientes con respecto al ambiente ideal
GGEPlot(GGE_Epartition, type = "Ranking Environments", footnote = F, titles = F)
```

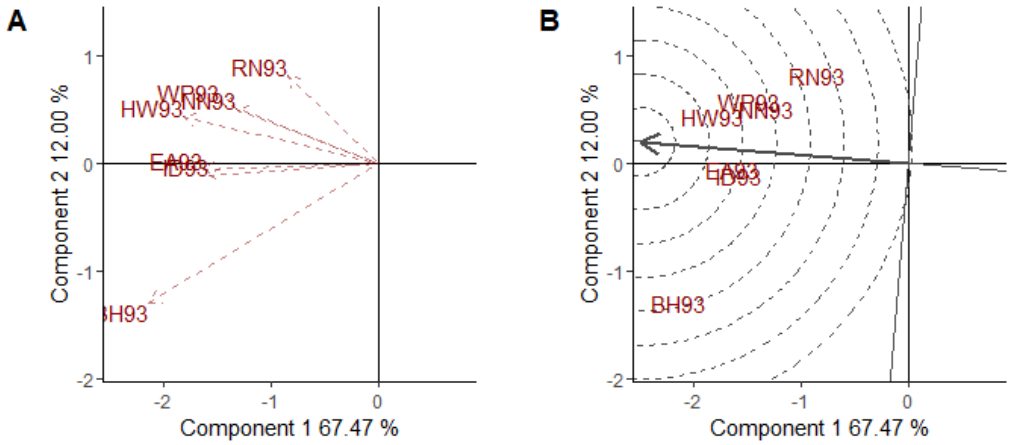


Figura 4.6: A:Relación entre ambientes. B:Clasificación de ambientes con respecto al ambiente ideal

Ver bien estas interpretaciones...Falta un grafico q es discriminacion vs representatividad q no se como se interpreta....

Biplot GE

Para visualizar solamente el efecto de IGA se utiliza el biplot GE obtenido del modelo AMMI. Este gráfico es posible obtenerlo utilizando la función rAMMI(), la cual no se encuentra disponible actualmente en R, y es una modificación de la publicación de Rodrigues et al. (2015). El formato del conjunto de datos de entrada es análogo al descripto en la función GGEmodel().

El biplot clásico para el conjunto de datos *yan.winterwheat* se muestra en la figura 4.7 junto con la sentencia utlizada para obtener el mismo. El primer argumento es el conjunto de datos de entrada, luego se indican los nombres de las columnas en las cuales se encuentra la información necesaria para aplicar la técnica y además el biplot que se desea obtener. Se observa que la magnitud de los vectores de los ambientes BH93, KE93 y OA93 es mayor a la de los demás ambientes, es decir que son los que más contribuyen a

la interacción. La cercanía de los marcadores de los genotipos m12 y Kat indica que esos genotipos tienen patrones de interacción similares, y a la vez, muy distintos a los de los genotipos Ann y Aug. Del biplot también se destacan las cercanías entre el genotipo dia y el ambiente BH93 lo que indica, debido a la gran distancia al origen, una fuerte asociación positiva entre el genotipos y el ambientes, es decir, es un ambiente muy favorable para ese genotipo. Entre las altas asociaciones negativas se puede mencionar a la del ambiente OA93 con el genotipo Luc (marcadores opuestos en el biplot) y se interpreta que ese ambiente es considerablemente desfavorable para ese genotipo. También se observa que los genotipos Cas y Reb están próximos al origen, lo que quiere decir que se adaptan en igual medida a todos los ambientes.

```
rAMMI(yanwinterwheat, genotype = "gen", environment = "env", response = "yield", type
    = "AMMI")
```

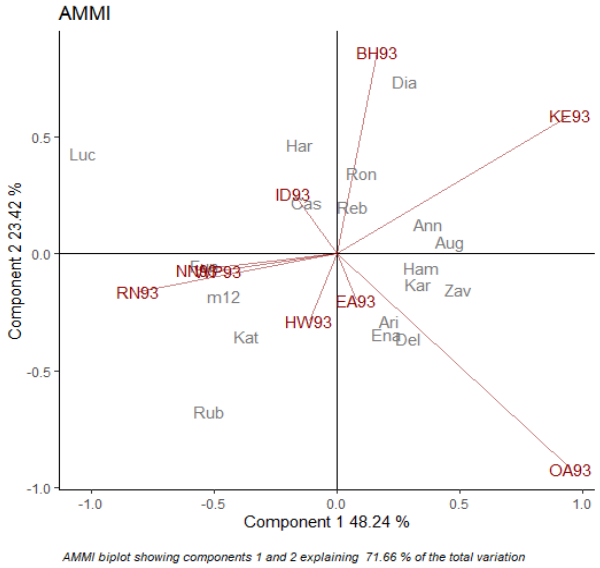


Figura 4.7: Biplot GE obtenido del modelo clasico AMMI

En caso de contar con observaciones atípicas se debe recurrir al biplot que resulta de algunos de los cinco modelos AMMI robustos propuestos por Rodrigues et al. (2015). Para ello se debe indicar en el argumento *type* cuál de ellos se desea ajustar: "rAMMI", "hAMMI", "gAMMI", "lAMMI", "ppAMMI".

Dado que el conjunto de datos *yan.winterwheat* no presentan atípicas, las conclusiones obtenidas con los biplots robustos no serán muy diferentes de las realizadas con el biplot clásico (Rodrigues et al., 2015) y por lo tanto carece de sentido interpretar dichos biplots en este ejemplo.

Métodos de imputación

Una limitación importante de los modelos presentados anteriormente es que requieren una que el conjunto de datos este completo, es decir que todos los genotipos sean evaluados en todos los ambientes. Por lo tanto, en el paquete se incluyen una serie de metodologías propuestas, algunas de las cuales no se encuentran disponible en R, para superar el problema de las observaciones perdidas. Entre los métodos incluidos se encuentran: "EM-AMMI", "EM-SVD", "Gabriel","WGabrielz "EM-PCA", los cuales se indican en la opción *type* de la función imputation(). El formato requerido para el conjunto de datos de entrada es análogo al indicado en las otras funciones incluidas en el paquete.

```
imputation(yanwinterwheat, PC.nb = 2, genotype = "gen", environment = "env", response
    = "yield", type = "EM−AMMI")
```

4.2. Geneticae Shiny Web App

Generalmente los mejoradores utilizan programas estadísticos que funcionan mediante una interfaz gráfica lo cual permite realizar el análisis de interés sin necesidad del manejo de un lenguaje de programación. *Geneticae Shiny Web App*, es una aplicación web que permite a los usuarios realizar muchos de los análisis incluidos en el paquete geneticae, sin necesidad de conocer el lenguaje de programación R.

Al iniciar *Geneticae Shiny Web App*, se muestra una pantalla en la cual se carga el conjunto de datos a analizar. Se admiten archivos con extensión .csv, delimitados por coma o punto y coma. Dado que esta aplicación se conecta con R y utiliza las funciones del paquete *geneticae* se debe respetar el formato del conjunto de datos de entrada requerido por el mismo. Como se indicó anteriormente, la base de datos debe estar en formato largo, es decir que las observaciones se encuentran en las filas y las variables, genotipos, ambientes, repeticiones (en caso de que haya) y el fenotipo observado, en las columnas. Además puede incluir otras variables que no serán utilizadas en el análisis ya que al cargar el conjunto de datos se debe indicar que columna corresponde al genotipo, al ambiente, a las repeticiones y al valor fenotípico a analizar. La primera fila del conjunto de datos puede contener los nombres de las variables, y en ese caso se indicará al cargarlo en la aplicación. El número de repeticiones puede diferir con los genotipos y los ambientes.

Dos conjuntos de datos de ejemplo, *plrv* y *yanwinterwheat*, incluidos en el paquete geneticae, se encuentran disponibles en la aplicación para ser descargados y probar la misma (Figura 4.8,4.9). La ruta para realizar esto es siguente: *The data* → *Example datasets*.

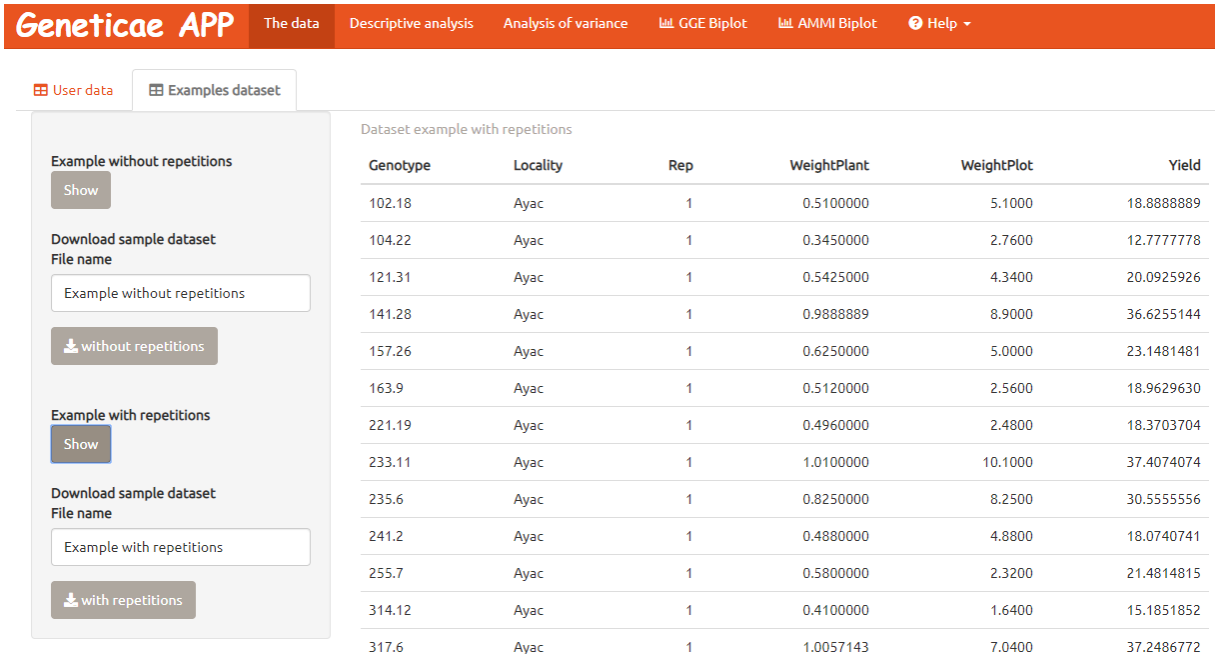


Figura 4.8: yanwinterwheat dataset disponible en Shiny Web App

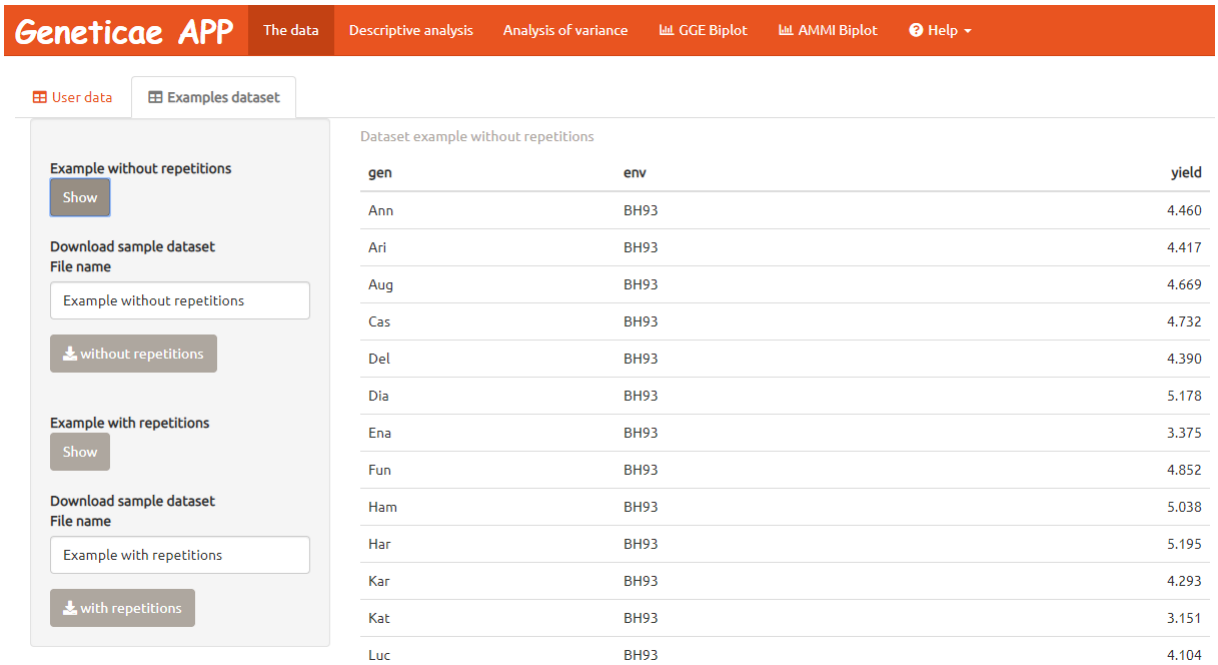


Figura 4.9: plrv dataset disponible en Shiny Web App

A fin de ilustrar los diferentes análisis que se pueden realizar con esta aplicación, se utiliza el conjunto de datos *yanwinterwheat* (Figura 4.8). Como se dijo anteriormente,

cuenta con información sobre el rendimiento medio de 18 variedades de trigo de invierno cultivadas en nueve ambientes en Ontario en 1993.

4.2.1. Análisis de un caso

En primer lugar se debe importar el conjunto de datos, indicando que el archivo .csv se encuentra delimitado por punto y coma, que la primera fila contiene los nombres de cada variable y además, el nombre de la columna que contiene la información de los genotipos, ambientes y valores fenotípicos de interés (Figura 4.10). Como en este caso no se cuenta con repeticiones dicho casillero quedará sin ninguna marca.

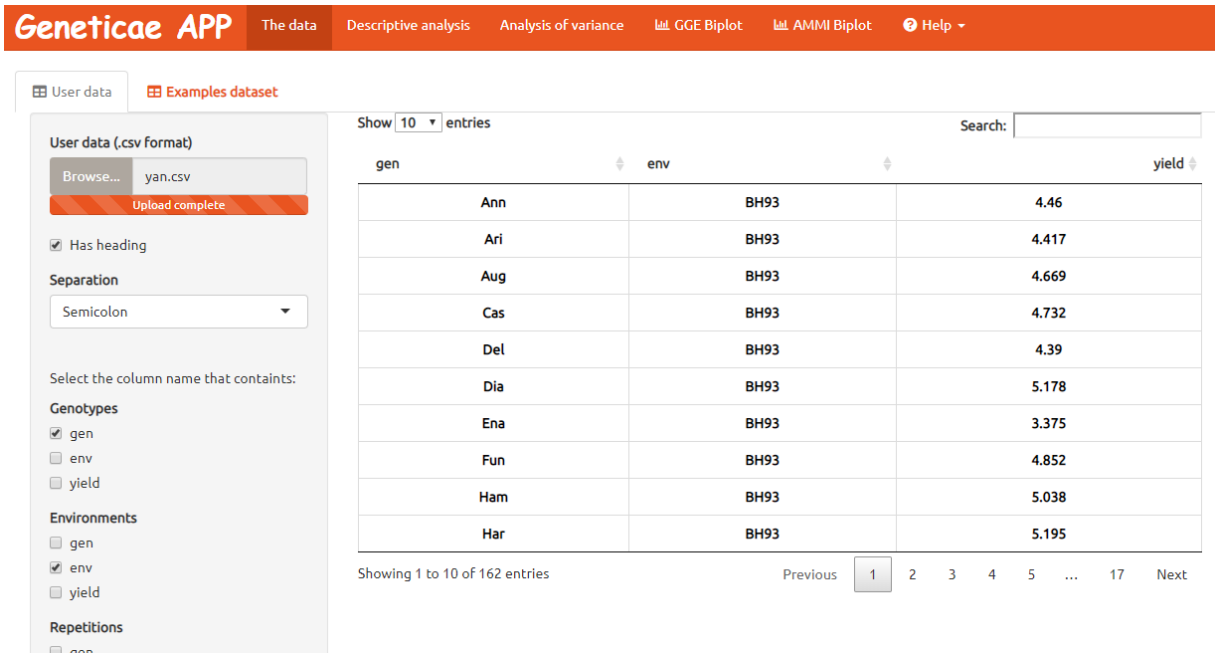


Figura 4.10: Importar conjunto de datos

Análisis descriptivo

El primer paso de cualquier estudio debe ser un análisis descriptivo del conjunto de datos. La pestaña *Descriptive analysis* brinda diversas herramientas para llevar a cabo dicho análisis. Una de ellas es un boxplot que compara el caracter cuantitativo de interés a través de los ambientes (Figura 4.11) o a través de los genotipos. Este gráfico es interactivo, al posicionarse sobre cada una de las cajas se muestran las medidas resumen utilizadas para la construcción del mismo. Además los mismos se pueden descargar en formato interactivo (.HTML) al hacer click en *Download* así como también en formato .png al hacer click en la cámara que aparece encima del gráfico (Figura 4.11). Algunos aspectos estilísticos, como el color de las cajas y los nombres de los ejes, pueden ser personalizados

por el usuario.

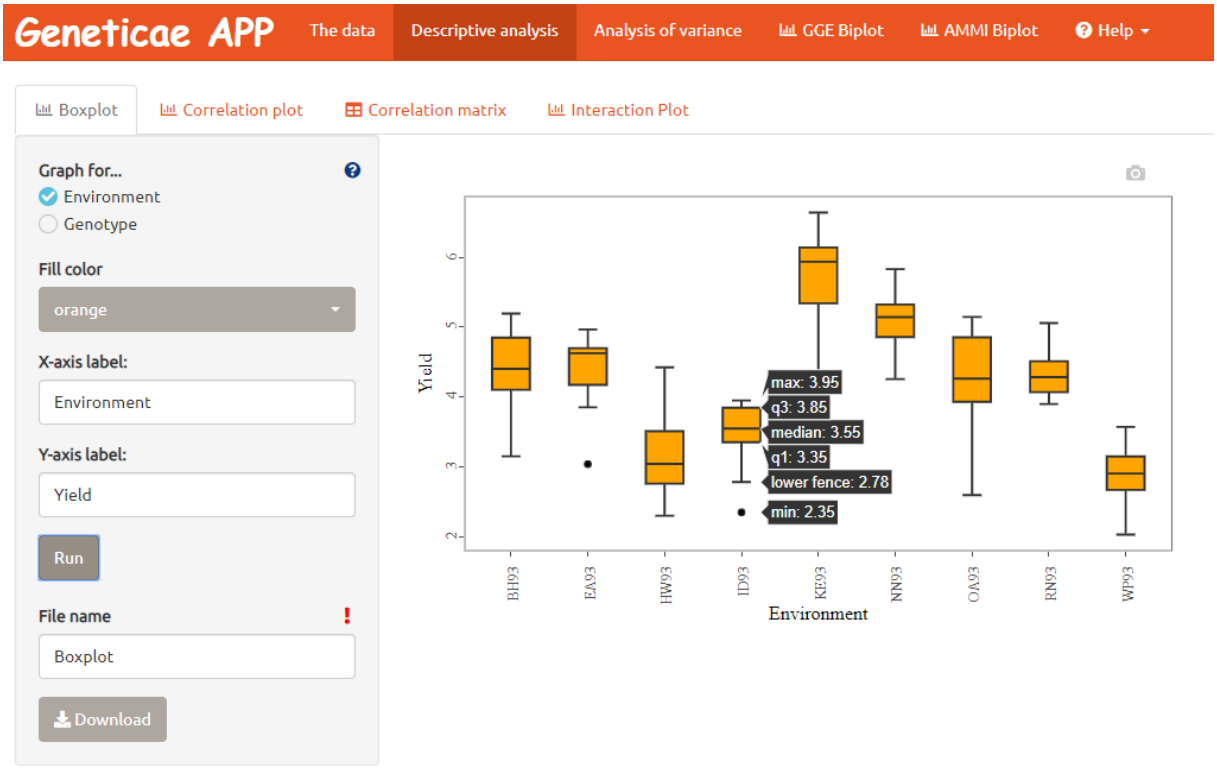


Figura 4.11: Boxplot de ambientes a través de los genotipos para el conjunto de datos Plrv

Otro análisis de interés puede ser estudiar la correlación entre los genotipos o entre los ambientes, para ello tanto el correlograma o gráfico de correlación como la matriz de correlación se pueden realizar (Figura 4.12 y 4.13). En ambos casos, se pueden estimar las correlaciones de Pearson o de Spearman. En el correlograma las correlaciones positivas se muestran en azul y las negativas en rojo y la intensidad del color y el tamaño del círculo son proporcionales a los coeficientes de correlación (Figura 4.12). Dicho gráfico puede ser descargado.

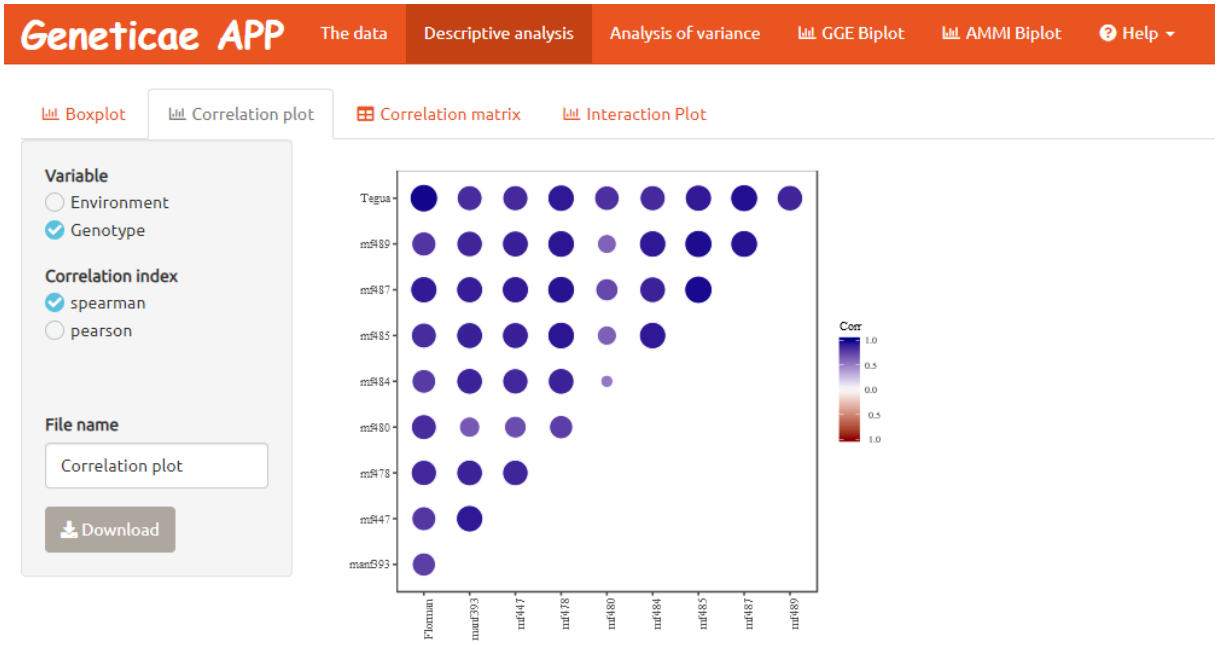
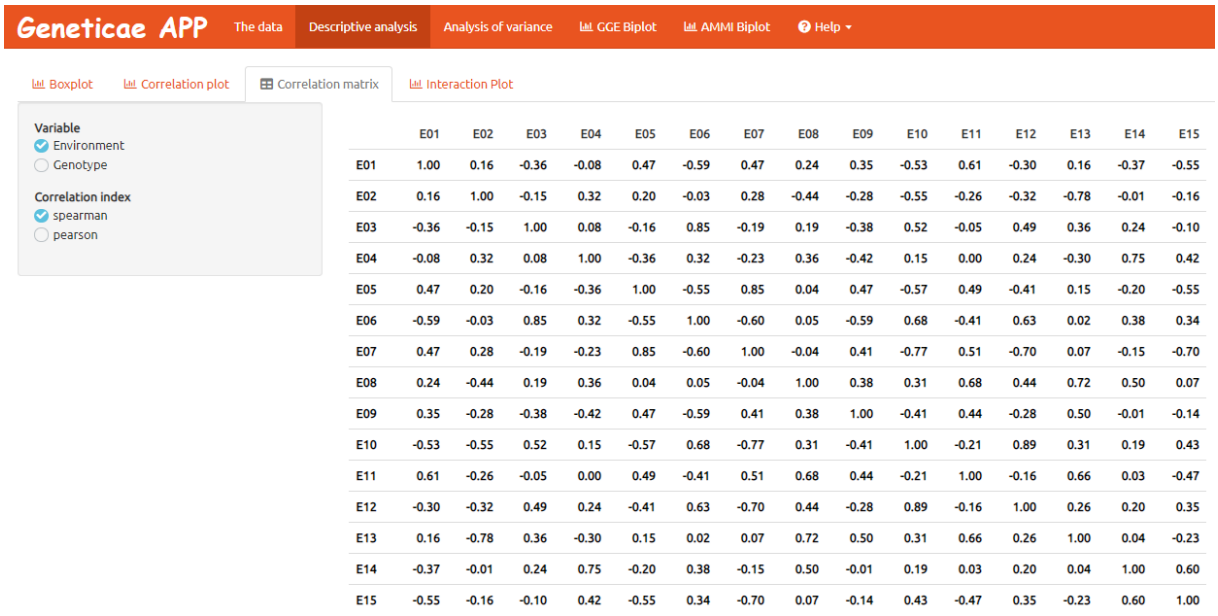


Figura 4.12: Boxplot de genotipos a través de los ambientes para el conjunto de datos Plrv



	E01	E02	E03	E04	E05	E06	E07	E08	E09	E10	E11	E12	E13	E14	E15
E01	1.00	0.16	-0.36	-0.08	0.47	-0.59	0.47	0.24	0.35	-0.53	0.61	-0.30	0.16	-0.37	-0.55
E02	0.16	1.00	-0.15	0.32	0.20	-0.03	0.28	-0.44	-0.28	-0.55	-0.26	-0.32	-0.78	-0.01	-0.16
E03	-0.36	-0.15	1.00	0.08	-0.16	0.85	-0.19	0.19	-0.38	0.52	-0.05	0.49	0.36	0.24	-0.10
E04	-0.08	0.32	0.08	1.00	-0.36	0.32	-0.23	0.36	-0.42	0.15	0.00	0.24	-0.30	0.75	0.42
E05	0.47	0.20	-0.16	-0.36	1.00	-0.55	0.85	0.04	0.47	-0.57	0.49	-0.41	0.15	-0.20	-0.55
E06	-0.59	-0.03	0.85	0.32	-0.55	1.00	-0.60	0.05	-0.59	0.68	-0.41	0.63	0.02	0.38	0.34
E07	0.47	0.28	-0.19	-0.23	0.85	-0.60	1.00	-0.04	0.41	-0.77	0.51	-0.70	0.07	-0.15	-0.70
E08	0.24	-0.44	0.19	0.36	0.04	0.05	-0.04	1.00	0.38	0.31	0.68	0.44	0.72	0.50	0.07
E09	0.35	-0.28	-0.38	-0.42	0.47	-0.59	0.41	0.38	1.00	-0.41	0.44	-0.28	0.50	-0.01	-0.14
E10	-0.53	-0.55	0.52	0.15	-0.57	0.68	-0.77	0.31	-0.41	1.00	-0.21	0.89	0.31	0.19	0.43
E11	0.61	-0.26	-0.05	0.00	0.49	-0.41	0.51	0.68	0.44	-0.21	1.00	-0.16	0.66	0.03	-0.47
E12	-0.30	-0.32	0.49	0.24	-0.41	0.63	-0.70	0.44	-0.28	0.89	-0.16	1.00	0.26	0.20	0.35
E13	0.16	-0.78	0.36	-0.30	0.15	0.02	0.07	0.72	0.50	0.31	0.66	0.26	1.00	0.04	-0.23
E14	-0.37	-0.01	0.24	0.75	-0.20	0.38	-0.15	0.50	-0.01	0.19	0.03	0.20	0.04	1.00	0.60
E15	-0.55	-0.16	-0.10	0.42	-0.55	0.34	-0.70	0.07	-0.14	0.43	-0.47	0.35	-0.23	0.60	1.00

Figura 4.13: Boxplot de genotipos a través de los ambientes para el conjunto de datos Plrv

Por último, dado que inconsistencias en el rendimiento de los genotipo en diferentes ambientes complican la tarea de los fitomejoradores ya que no existe un genotipo superior en todos los ambientes estudiados y que las técnicas de principal interés de la aplicación carecen de sentido cuando dicho efecto no esta presente en el conjunto de datos, es posible realizar un gráfico de interacción. En la figura 4.14 se observa el cambio en el efecto genotípico a través de los ambientes, sin embargo es posible también mostrar el cambio en el efecto ambiental a través de los genotipos. En forma análoga al boxplot, éste es un gráfico interactivo, y por lo tanto, es posible descargarlo en formato interactivo (.HTML) a partir del boton *Download*, así como también en formato .png al hacer click en la cámara (Figura 4.14). A su vez, los nombres de los ejes pueden ser personalizados por el usuario.

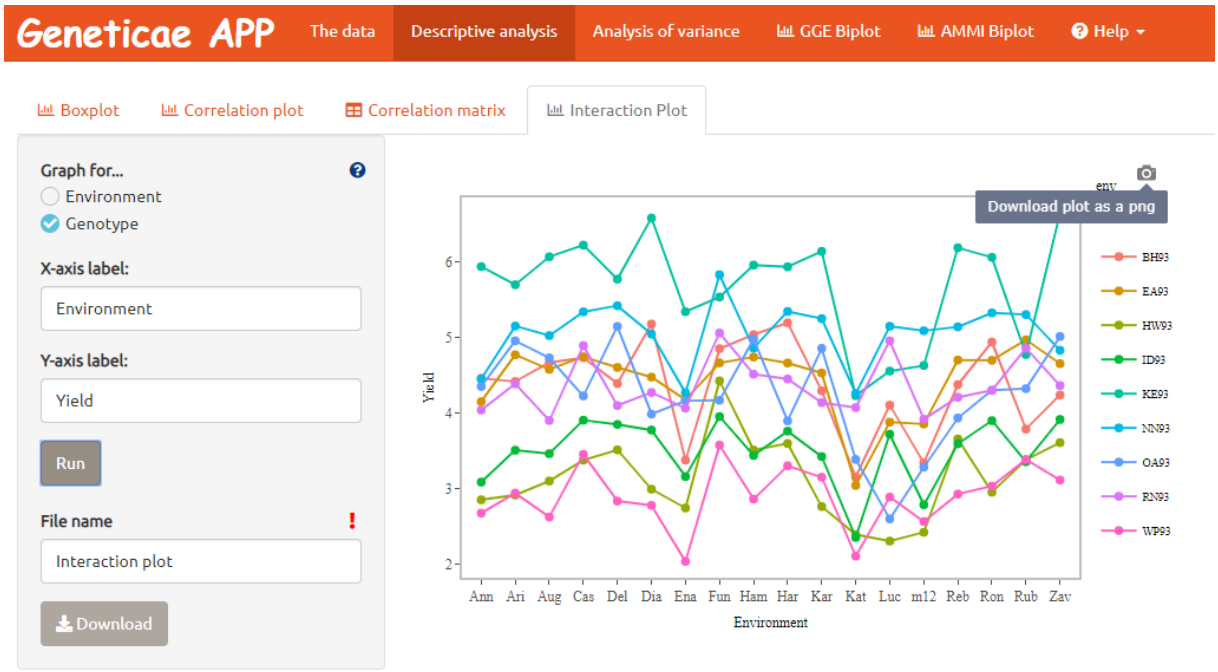


Figura 4.14: Boxplot de genotipos a través de los ambientes para el conjunto de datos Plrv

Análisis de la variancia

La variación fenotípica se puede explicar a partir de los efectos ambientales, genotípicos y de la interacción entre los anteriores. Para probar la significancia de dichos efectos se realiza un ANOVA. Si únicamente los efectos de G y A resultan significativos (es decir, no hay efecto interacción), la interacción debe ser ignorada y los biplots carecen de sentido. En caso de no contar con repeticiones en el conjunto de datos, la interacción no podrá ser testeada y un mensaje aparecerá aclarando lo mencionado ***"The interaction effect***

***can be tested since there are repetitions in the data set"*, VER**. Por lo tanto, en este caso que no se cuenta con repeticiones, únicamente se puede pruebar si existe efecto genotípicos y ambienteales (Figura 4.15)

	Df	Sum Sq	Mean Sq	F-Value	P- Value
Environment	8	114.53622	14.3170280	97.560054	0
Genotype	17	22.67117	1.3335985	9.087496	0
Residuals	136	19.95813	0.1467509	-	-

Figura 4.15: Boxplot de genotipos a través de los ambientes para el conjunto de datos Plrv

La validez de las conclusiones del ANOVA depende del cumplimiento de que los errores tengan distribución normal con media cero y variancia constante. Tres pestañas de la aplicación: *Check normality*, *Check homocedasticity* y *Outliers* permiten verificar los supuestos mencionados.

El supuesto de normalidad se puede verificar gráficamente mediante un histograma y un gráfico de probabilidad normal (Figura 4.16), así como también con el test de shapiro-wilks.

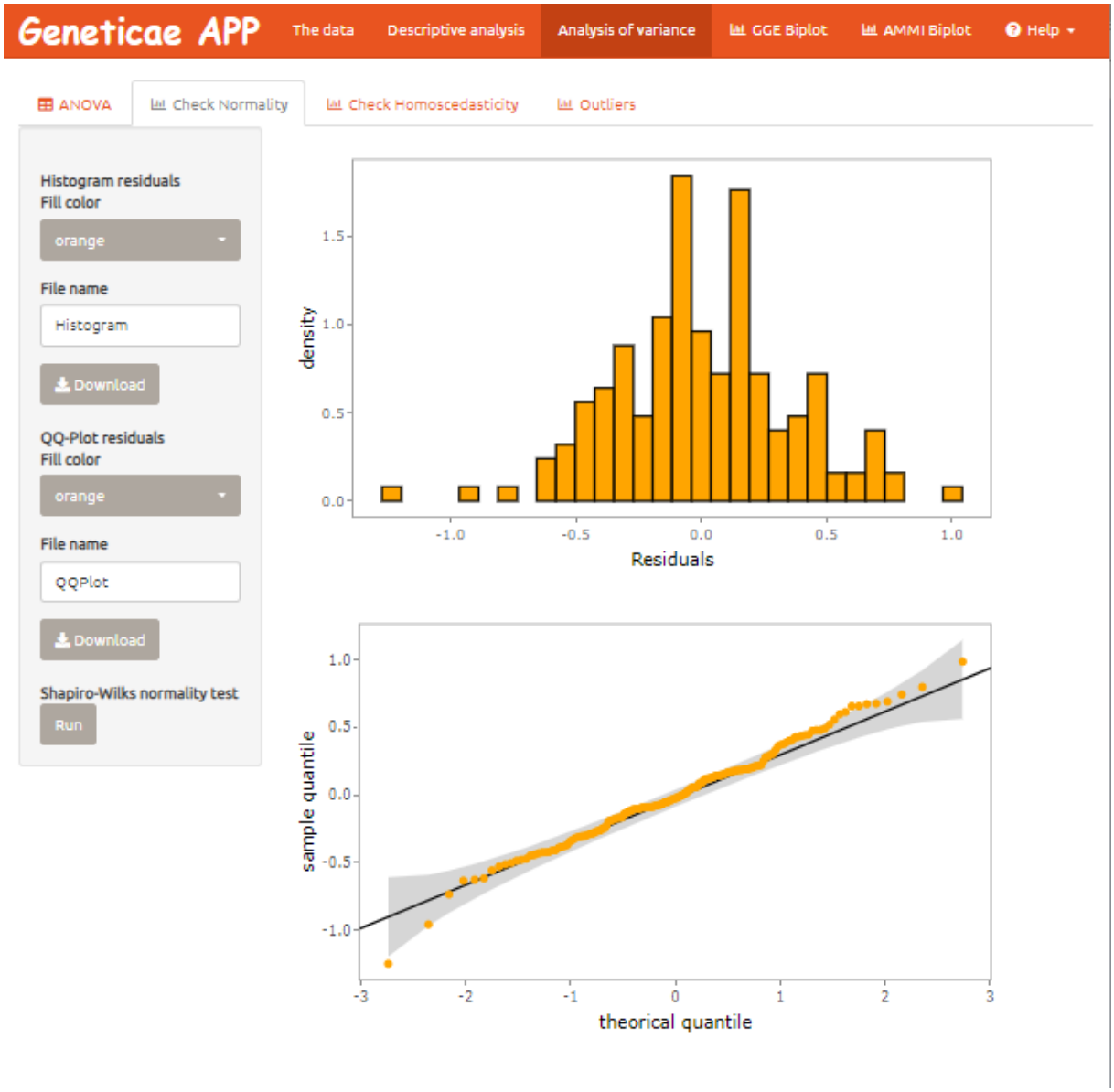


Figura 4.16: Boxplot de genotipos a través de los ambientes para el conjunto de datos Plrv

El supuesto de variancia constante u homocedasticidad se puede probar con un gráfico de residuos vs. valores predichos, así como también con el test de levene (Figura 4.17).

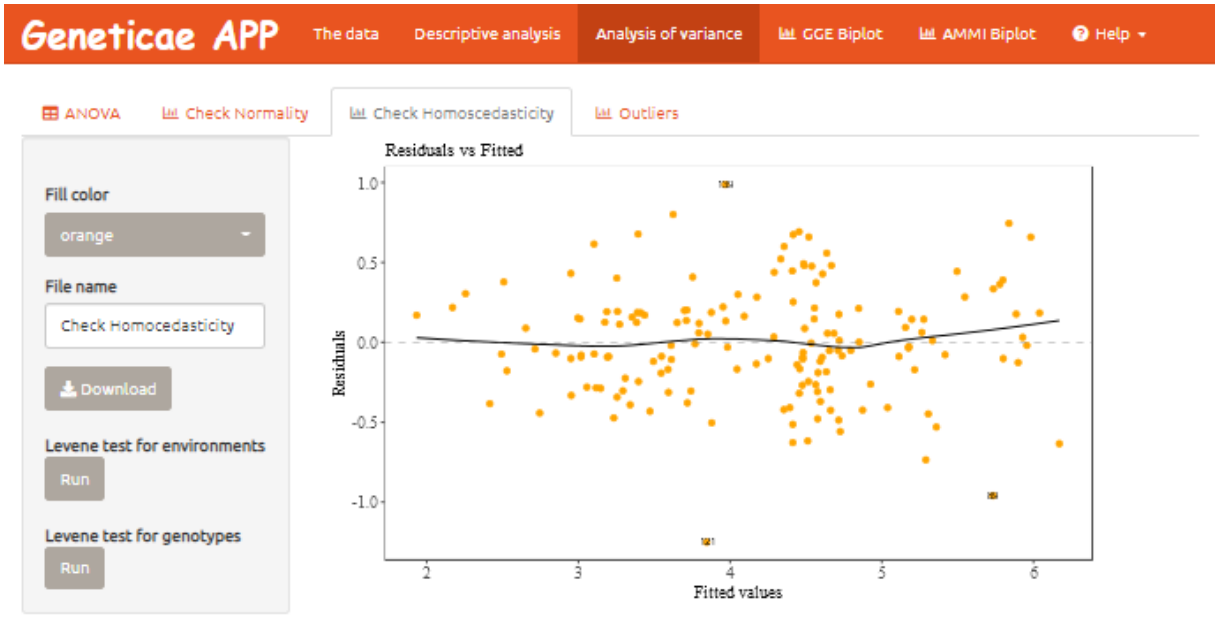


Figura 4.17: Boxplot de genotipos a través de los ambientes para el conjunto de datos Plrv

Por último, el ANOVA no es robusto ante la presencia de observaciones atípicas, por lo tanto se incluyen gráficos para detectar la presencia de las mismas (Figura 4.18).

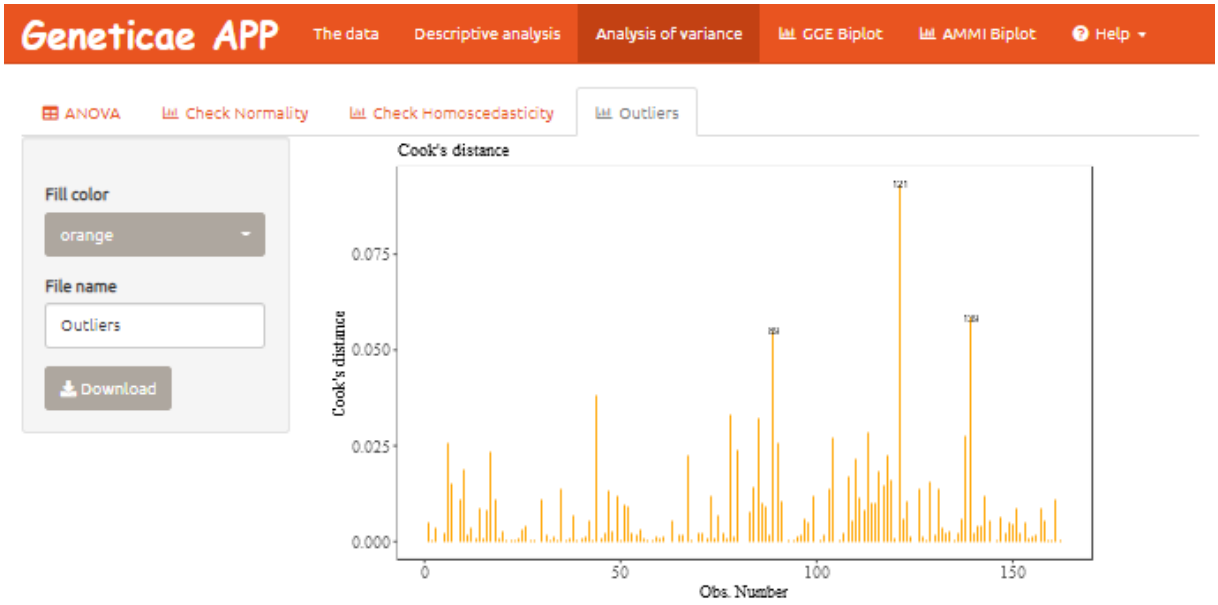


Figura 4.18: Boxplot de genotipos a través de los ambientes para el conjunto de datos Plrv

Todos los gráficos mostrados para verificar los supuestos requeridos por la técnica ANOVA pueden ser descargados mediante el botón *Download* de la pestaña correspondiente.

Biplots

Geneticae Shiny Web App permite obtener tanto el biplot GGE (Figura 4.19) como el GE (Figura 4.20) . Ciertos atributos estilísticos de dichos gráficos se pueden personalizar y además pueden ser descargados.

Dada la importancia del biplot GGE, se incluyen aquellos más utilizados en el análisis de datos provenientes de EMA. Entre ellos, el biplot básico y aquellos que permiten estudiar la relación entre los ambientes (*Relationship Among Environments*), la identificación del mejor cultivar en cada ambiente (*Which Won Where/What*), Discrimination vs. representativeness (**Este es el q me falta interpretar en el paquete**), clasificación de los ambientes con respecto al ambiente ideal (*Ranking Environments*), evaluación de los cultivares con base en el rendimiento promedio y la estabilidad (*Mean vs. Stability*) y clasificación de genotipos con respecto al genotipo ideal (*Ranking Genotypes*). A la hora de indicar el biplot que se desea realizar se debe especificar el método de SVD, centrado

y escalado correspondiente.

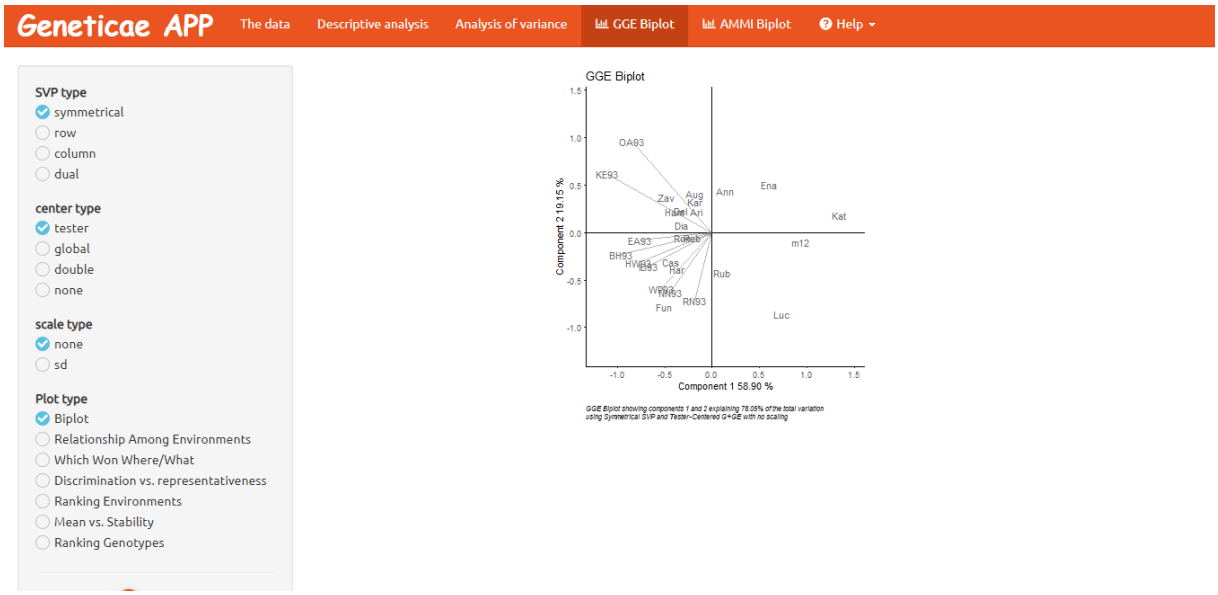


Figura 4.19: Boxplot de genotipos a través de los ambientes para el conjunto de datos Plrv

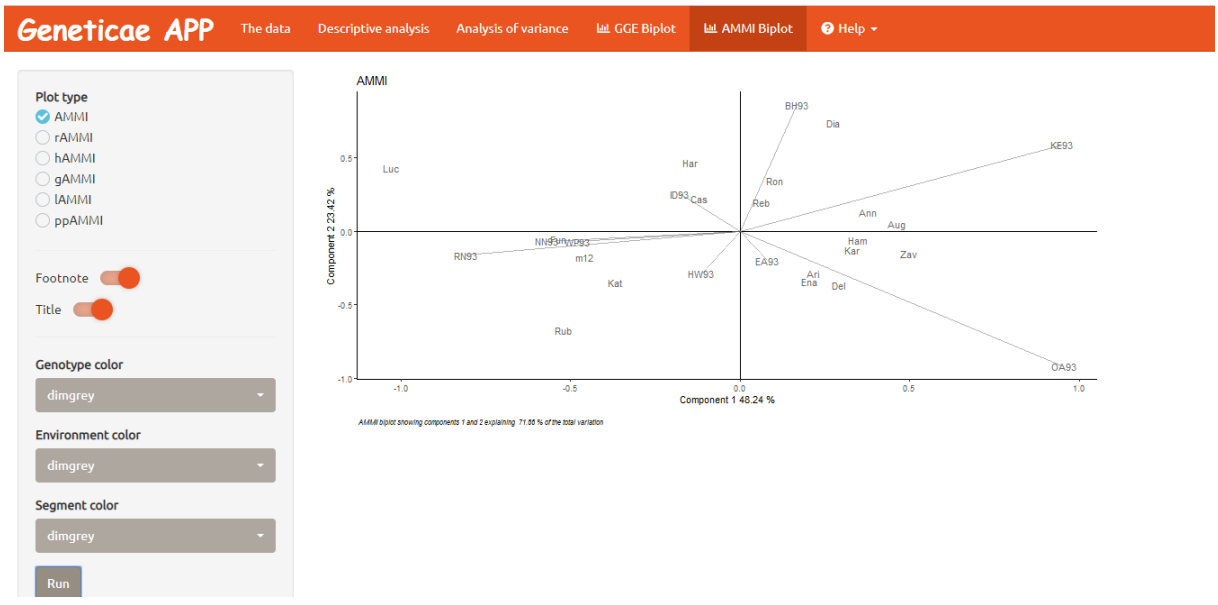


Figura 4.20: AMMI

Ayuda

Información general, ejemplos y un video tutorial de cómo utilizar la aplicación se encuentran disponibled en la última pestaña de la misma.

Capítulo 5

Conclusiones

Perspectivas futuro

permitir imputar en la aplicación, permitir hacer graficas por mega ambiente

Bibliografía

R.W. Allard. *Principios de la mejora genética de las plantas*. Ediciones Omega, 1967.

J. Crossa, H.G. Gauch, y R.W. Zobel. Additive main effects and multiplicative interaction analysis of two international maize cultivar trials. *Crop Science*, 30:493–500, 1990.

R. Cruz Medina. Some exact conditional tests for the multiplicative models to explain genotype-environment interaction. *Heredity*, 69:128—-132, 1992.

H. G. Gauch. Model selection and validation for yield trials. *Theoretical and Applied Genetics*, 80:153–160, 1988.

H.G. Gauch y R.W. Zobel. Identifying mega-environments and targeting genotypes. *Crop Science*, 37:311—-326, 1997.

W. Hadley, J. Hester, y W. Chang. *devtools: Tools to Make Developing R Packages Easier*, 2019. URL `https://CRAN.R-project.org/package=devtools`. R package version 2.1.0.

R. A. Kempton. The use of biplot in interpreting variety by environment interactions. *Journal of Agricultural Science*, 122:335–342, 1984.

W. Yan y M. Kang. *GGE Biplot Analysis: A Graphical Tool for Breeders, Geneticists*. CRC Press, 2003.