



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 TÍTULO DE ESPECIALISTA EN  
BIOINFORMÁTICA

---

DIRECTOR: Dr. Gerardo Cervigni  
CO-DIRECTOR: Mgs. Marcos Prunello

AÑO: 2021

# 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 2021**, 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 inglés *Site Regression model*

DVS: descomposición de valores singulares

GNU: *General Public Licence*

CRAN: *Comprehensive R Archive Network*

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

---

# Resumen

Las variedades mejoradas de cultivos vegetales son el resultado del trabajo de desarrollo genético llevado a cabo en los programas de fitomejoramiento, 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 económicas por malas campañas y el suministro al mercado sea constante. Consecuentemente, un análisis adecuado de la información de los EMA es indispensable para asegurar el éxito del programa de mejoramiento de cultivos. Actualmente, R es uno de los lenguajes de programación más utilizados para el análisis de datos debido a su distribución como software libre y a la gran variedad de herramientas que ofrece. Sin embargo, los mejoradores que no están familiarizados con la programación tienden a utilizar otros tipos de programas que responden a instrucciones por menú en lugar de escribir líneas de código, a pesar de los costos económicos derivados del pago de sus licencias. Mientras que, aquellos que sí tienen afinidad con el uso de código para el análisis de datos se enfrentan con dificultades a la hora de identificar las herramientas apropiadas entre el gran número de instrumentos disponibles. Por lo tanto, en este trabajo se presenta el desarrollo de dos herramientas informáticas para asistir en el análisis de datos provenientes de EMA. Por un lado, se creó un nuevo paquete de R que incluye metodología recientemente publicada que no se encuentra disponible en el software y al mismo tiempo reúne todas aquellas de mayor utilidad, de modo que aquellos usuarios que posean un manejo del lenguaje puedan simplificar su tarea. Por otro lado, se confeccionó una interfaz gráfica de usuario mediante una aplicación web Shiny que permite realizar los principales análisis implementados en el paquete sin necesidad de programar y se encuentra publicada en internet en el servidor de CONICET para su libre acceso.

**Palabras Clave:** análisis estadístico, ensayos multiambientales, interfaz gráfica, lenguaje R, programación

---

# Abstract

Crop improvement is the result of genetic development that requires several years and large investments. In advanced stages of breeding programs, multi-environment trials (MET), which consist of evaluating different cultivars in multiple environments, are fundamental tools to increase crop productivity. Since varieties remain on market for decades, their choice is essential to avoid economic losses due to bad seasons and to ensure a constant supply. Consequently, an adequate analysis of MET data is essential to ensure the success of a breeding program. Currently, R is one of the most widely used programming language for data analysis due to its distribution as free software and the wide variety of tools it offers. However, breeders who are unfamiliar with programming tend to use other types of programs that respond to menu prompts instead of writing lines of code, despite the financial costs of their licenses. Whereas, those who have an affinity with the use of code for data analysis face difficulties in identifying the right tools from the large number of instruments available. Therefore, in this work two tools are develop to analysis of MET data. On one hand, a new R package that includes new methodology not available in the software and at the same time brings together all those most useful was created to facilitate the users task. On the other hand, a graphical user interface was created using a Shiny web application that allows the main analyzes implemented in the package to be carried out without the need for programming and is published on CONICET server for free access.

**Keywords:** multi-environment trials, programming, statistical analysis, user interfaz, R lenguaje

---

# Índice general

Capítulos	Página
<b>1. Introducción</b>	<b>1</b>
<b>2. Objetivos</b>	<b>5</b>
2.1. Objetivo general . . . . .	5
2.2. Objetivos específicos . . . . .	5
<b>3. Métodos</b>	<b>6</b>
3.1. Métodos estadísticos . . . . .	6
3.1.1. Modelo AMMI y SREG . . . . .	6
3.1.2. Modelo AMMI robusto . . . . .	8
3.1.3. Métodos de imputación . . . . .	8
3.2. Paquete de R . . . . .	9
3.2.1. Creación del paquete . . . . .	10
3.2.2. Archivos de código . . . . .	11
3.2.3. Documentación . . . . .	12
3.2.4. Editar el archivo DESCRIPTION . . . . .	12
3.2.5. Testeos . . . . .	14
3.2.6. Compilación e instalación . . . . .	16
3.2.7. Algunos elementos complementarios . . . . .	16
3.2.7.1. Viñetas . . . . .	17
3.2.7.2. Agregar datasets . . . . .	17
3.2.7.3. Archivo README . . . . .	17
3.2.7.4. Archivo NEWS . . . . .	19

---

3.2.7.5.	Crear una página web . . . . .	19
3.2.7.6.	Publicación . . . . .	19
3.3.	Shiny APP . . . . .	20
3.3.1.	Estructura de Shiny APP . . . . .	20
3.3.2.	Desarrollo de Shiny APP . . . . .	21
3.3.3.	Compartiendo una Shiny Web App . . . . .	22
<b>4.</b>	<b>Resultados</b>	<b>23</b>
4.1.	Paquete de R <i>geneticae</i> . . . . .	23
4.1.1.	Conjuntos de datos en <i>geneticae</i> . . . . .	23
4.1.2.	Aplicación de las funciones incluidas en el paquete . . . . .	24
4.2.	Geneticae Shiny Web App . . . . .	35
4.2.1.	Análisis de un caso . . . . .	37
<b>5.</b>	<b>Conclusiones</b>	<b>46</b>
	<b>Bibliografía</b>	<b>48</b>



---

# Índice de figuras

1.1. Representación gráfica de tipos de IGA: (A)IGA crossover, (B) IGA no crossover y (C) no IGA . . . . .	2
3.1. Chequeo de disponibilidad del nombre elegido . . . . .	10
3.2. Creación del paquete <i>geneticae</i> . . . . .	11
3.3. Archivo DESCRIPTION de <i>geneticae</i> . . . . .	14
3.4. Resultado de correr los tests creados para <i>geneticae</i> . . . . .	15
3.5. Análisis de cobertura de los tests de <i>geneticae</i> . . . . .	16
3.6. Fragmento de README de <i>geneticae</i> . . . . .	18
3.7. Archivo NEWS de <i>geneticae</i> . . . . .	19
4.1. Biplot básico obtenido de la función <code>GGEPlot()</code> . . . . .	26
4.2. A: Ranking de cultivares en el ambiente OA93. B: Ranking de ambientes para cultivar Kat . . . . .	27
4.3. Comparación entre dos genotipos obtenido de la función <code>GGEPlot()</code> . . . .	28
4.4. Identificación del mejor cultivar en cada ambiente a partir de la función <code>GGEPlot()</code> . . . . .	29
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 . .	31
4.6. A:Relación entre ambientes. B:Clasificación de ambientes con respecto al ambiente ideal . . . . .	33
4.7. Biplot GE obtenido del modelo clasico AMMI . . . . .	34
4.8. <i>yanwinterwheat</i> dataset disponible en Shiny Web App . . . . .	36
4.9. <i>plrv</i> dataset disponible en Shiny Web App . . . . .	36
4.10. Importar conjunto de datos . . . . .	37

---

4.11. Boxplot de ambientes a través de los genotipos para el conjunto de datos	
Plrv . . . . .	38
4.12. Boxplot de genotipos a través de los ambientes para el conjunto de datos	
Plrv . . . . .	39
4.13. Boxplot de genotipos a través de los ambientes para el conjunto de datos	
Plrv . . . . .	39
4.14. Boxplot de genotipos a través de los ambientes para el conjunto de datos	
Plrv . . . . .	40
4.15. Boxplot de genotipos a través de los ambientes para el conjunto de datos	
Plrv . . . . .	41
4.16. Boxplot de genotipos a través de los ambientes para el conjunto de datos	
Plrv . . . . .	42
4.17. Boxplot de genotipos a través de los ambientes para el conjunto de datos	
Plrv . . . . .	43
4.18. Boxplot de genotipos a través de los ambientes para el conjunto de datos	
Plrv . . . . .	44
4.19. Boxplot de genotipos a través de los ambientes para el conjunto de datos	
Plrv . . . . .	45
4.20. AMMI . . . . .	45

---

# 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. 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. En etapas avanzadas de estos programas, comúnmente se llevan a cabo ensayos multiambiales (EMA) de comparación de rendimientos, donde un conjunto de variedades se evalúan en múltiples ambientes. Estos son esenciales debido a la presencia de interacción genotipo - ambiente (IGA) la cual es inevitable debido a las variaciones en las condiciones climáticas y de suelo de los distintos ambientes analizados. La IGA es considerada casi unánimemente por los fitomejoradores como el principal factor que limita la selección de cultivares superiores y, en general, afecta la eficiencia de los programas de mejoramiento (Crossa et al., 1990; Cruz Medina, 1992); **Kang y Magari, 1996**). Cuando los ambientes son muy diferentes, la IGA usualmente gana importancia porque cambia el rango de las líneas de mejoramiento. Gauch y Zobel (1997) explicaron que si no hubiera interacción, una sola variedad o híbrido 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, conocidas también como sin cambio de rango o *no crossover* (NCOI), de las cualitativas, denominadas a su vez como con cambio de rango o *crossover* (COI) (Cornelius et al., 1996). Cuando dos genotipos X e Y tienen una respuesta diferencial en dos ambientes, se dice que la IGA es del tipo COI si hay cambios en el orden de los genotipos según su rendimiento (Figura 1.1(A)) y del tipo NCOI si su ordenamiento permanece sin cambios (Figura 1.1(B)). Por otro lado, se dice que la IGA es inexistente cuando los genotipos responden de manera similar en ambos ambientes (Figura 1.1(C)).

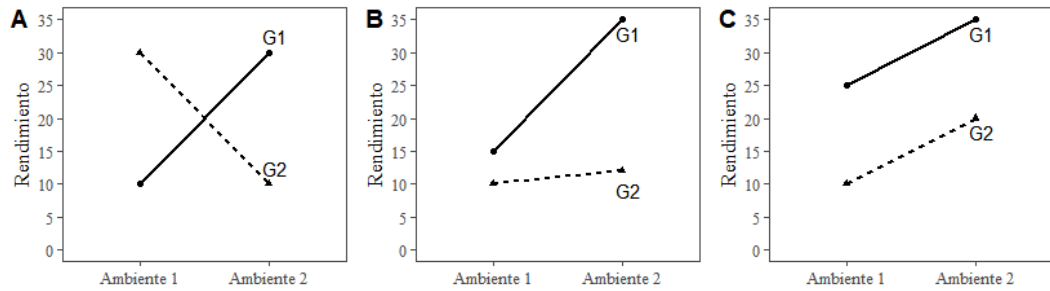


Figura 1.1: Representación gráfica de tipos de IGA: (A) IGA crossover, (B) IGA no crossover y (C) no IGA

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 carácter, 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**). Conceptos importantes tales como regiones ecológicas, ecotipos, mega-ambientes, adaptación específica y estabilidad se pueden analizar a partir de la IGA (Yan y Hunt, 2001).

Un análisis adecuado de la información de los EMA es indispensable para el éxito del programa de mejoramiento genético de los cultivos. 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 apropiada. Las más difundidas para analizar los datos provenientes de EMA se basan 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 como herramientas 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). En su forma regular, AMMI explora la matriz de IGA realizando una descomposición en valores singulares (DVS) sobre la matriz residual de un ANOVA que considera el efecto aditivo de los genotipos (G) y ambientes (A). Mientras que, en SREG el ANOVA sólo incluye el efecto de A, lo que permite analizar G e IGA en forma conjuntamente. El resultado de los dos primeros términos multiplicativos de la DVS a menudo se presentan en un biplot llamado GE para el modelo AMMI y GGE para

---

SREG (**Yan y Hunt (2002)**). Sin embargo, estos modelos no siempre son lo suficientemente eficiente para analizar la estructura de datos MET de programas de mejoramiento vegetal. Por un lado, tiene serias limitaciones frente a información faltante y, apesar de que los MET están diseñados para que todos los genotipos se evalúen en todos los ambientes, la presencia de valores perdidos es muy común. Esto ocurre, por ejemplo, debido a errores de medición o destrucción 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 mal desempeño (Hill y Rosemberg, 1985). Numerosas metodologías de imputación se han estado desarrollando en los ultimos años para solventar esta limitación (CITAS.....). Por otro lado, ambos modelos son sensibles a la presencia de observaciones atípicas, lo cual es una regla más que una excepción cuando se consideran datos reales. Para superar esta fragilidad, recientemente distintas metodologias robustas se han desarrollado para el modelo AMMI (Rodrigues et al. 2015).

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 cumplir 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.

Ante estas dificultades, una alternativa para el análisis es el empleo de algún lenguaje de programación de distribución libre y gratuita, que le confiera al analista la flexibilidad necesaria para cumplir con su objetivo. En este contexto, R es uno de los lenguajes de programación desarrollados para el análisis de datos de mayor uso en la actualidad. R es un software de uso libre y distribuido bajo los términos de la *General Public Licence* (GNU). Este programa se descarga de un repositorio mantenido por *The R Foundation for Statistical Computing* conocido como CRAN (*Comprehensive R Archive Network*), en el cual también se encuentran disponibles miles de paquetes adicionales que consisten en conjuntos de funciones desarrolladas con fines específicos que se distribuyen con un protocolo determinado, garantizando su correcto funcionamiento. Cualquier desarrollador puede producir su propio paquete y publicarlo en CRAN, siempre que cumpla con los requisitos establecidos y pase correctamente por los procedimientos de control. Además, hay paquetes que pueden obtenerse de otros repositorios como Github, Bioconductor, rOpenSci, entre otros. R es propicio para el análisis de datos de EMA puesto que se ha desarrollado metodología específica para este entorno computacional.

---

A pesar de estas ventajas, el análisis de datos de EMA en R presenta algunos desafíos. Por un lado, existen numerosos paquetes con funcionalidad afín que hay que identificar cómo combinar adecuadamente. Por otro lado, el software puede resultar dificultoso para aquellos analistas no familiarizados con la programación. Atendiendo a estas dos necesidades, se crea un paquete que incluya metodología recientemente publicada y reuna las funciones más útiles fin de solventar la primera de ellas. Para la segunda, se crea una aplicación web Shiny de libre acceso mediante conexión a internet que permita realizar los principales análisis implementados en el paquete sin necesidad de escribir líneas de código.

---

# 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

Este capítulo tiene como objetivo focalizar al lector en los aspectos fundamentales sobre los que se apoya este trabajo. Se compone de tres secciones, en la primera se presenta la metodología estadística que se incluirá en el paquete de R. En la segunda y tercera sección, se presenta un flujo de trabajo reproducible para el desarrollo del paquete y la aplicación web Shiny, respectivamente.

#### 3.1.1. Modelo AMMI y SREG

El modelo AMMI (Zobel et al. 1988) y SREG (Cornelius et al. 1996; Crossa y Cornelius 1997 y 2002) son modelos multiplicativos que además de tener una componente aditiva, poseen una multiplicativa. Ellos difieren en las componentes aditivas que se eliminan de la ecuación y que se incorporan en forma multiplicativa. Mientras que AMMI considera los ambientes (A) y el genotipo (G) como aditivos, SREG solo A. Por lo tanto, en el primero la IGA representa la parte multiplicativa y en el segundo G e IGA en forma conjunta.

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 para cada combinación de genotipo y ambiente luego de ajustar por mínimos cuadrados el modelo de efectos principales.

Las ecuaciones de los distintos modelos son:

$$\text{AMMI: } y_{ij} = \mu + G_i + A_j + \sum_{k=1}^K \lambda_k \alpha_{ik} \gamma_{jk}$$

$$\text{SREG: } y_{ij} = \mu + A_j + \sum_{k=1}^K \lambda_k \alpha_{ik} \gamma_{jk}$$

donde



- 
- $y_{ij}$  es el caracter fenotípico evaluado (rendimiento o cualquier otro caracter de interés) del  $i$ -ésimo genotipo en el  $j$ -ésimo ambiente,
  - $\mu$  es la media general,
  - $G_i$  es el efecto del  $i$ -ésimo genotipo con  $i = 1, \dots, g$ ,
  - $A_j$  es el efecto del  $j$ -ésimo ambiente con  $j = 1, \dots, a$ ,
  - $\sum_{k=1}^K \lambda_k \alpha_{ik} \gamma_{jk}$  es la sumatoria de términos multiplicativas utilizadas para modelar la IGA en AMMI o G e IGA en SREG. Siendo,  $K$  el número de términos multiplicativos retenidos en el modelo con  $K \leq \min(g-1, a-1)$  en AMMI y  $K \leq \min(g, a-1)$  en SREG;  $\lambda_k$  el  $k$ -ésimo valor singular y  $\alpha_{ik}$  y  $\gamma_{jk}$  el elemento del autovector asociado con el  $i$ -ésimo genotipo y el  $j$ -ésimo ambiente para el  $k$ -ésimo término multiplicativo. Generalmente los dos primeros términos multiplicativos ( $K = 2$ ) son suficientes para explicar los patrones de la IGA en AMMI y de G e IGA en SREG; la variabilidad remanente se interpreta como ruido aleatorio.

El resultado de los dos primeros términos multiplicativos de la SVD se presenta a menudo en un biplot llamado GE (*Genotipe-Environment*) (**CITA**) en AMMI y GGE en SREG (Yan et al. 2000) y representan una aproximación de dos rangos de los efectos multiplicativos. 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. Un biplot GGE que explica suficiente variabilidad debida a G e IGA de un conjunto de datos MET permite, entre otras cosas, la visualización de tres aspectos importantes:

- (i) las relaciones entre los genotipos y ambientes representadas por el patrón *which-won-where*, que facilitan la investigación del megaambiente (Gauch y Zobel, 1997);
- (ii) las interrelaciones entre los ambientes de prueba que permiten la identificación de mejores para la evaluación de cultivares (Cooper et al., 1997) y de aquellos redundantes que pueden ser descartados (Yan y Rajcan, 2002);
- (iii) las interrelaciones entre genotipos que posibilita la comparación entre ellos y la clasificación de los mismos considerando tanto en el rendimiento medio como la estabilidad (Yan et al., 2001)

En un biplot, el  $i$ -ésimo genotipo se muestra como un punto definido por todos los valores  $g_{ik} = \lambda_k^s \gamma_{ik}$ , y el  $j$ -ésimo ambiente por  $e_{kj} = \lambda_k^{1-s} \delta_{jk}$  ( $k = 1, 2$  para un biplot bidimensional), donde  $s$  es el factor de partición. Cuando  $s = 1$  se denomina escalamiento centrado en los genotipos, centrado en los ambientes si  $s = 0$  y simétrica cuando  $s = 0,5$ . El último factor es el utilizado en el biplot GE y el más utilizado en GGE, aunque dependiendo de los intereses de la investigación, se pueden construir numerosos biplots derivados de SREG. Independientemente del factor de partición utilizado, los biplots GGE revelan el mismo patrón *which-won-where*. Sin embargo, difieren en diversos aspectos. El

---

primero muestra la interrelación entre genotipos con mayor precisión que cualquier otro método, el centrado en el ambiente es el más informativo de las interrelaciones entre los ambientes, mientras que el simétrico permite visualizar la magnitud relativa tanto de la variación de los genotipos como de los ambientes.

### 3.1.2. Modelo AMMI robusto

El modelo AMMI, en su forma estándar, asume que no hay valores atípicos en el conjunto de datos. Sin embargo, la presencia de *outliers* es más una regla que una excepción cuando se consideran datos agronómicos debido características inherentes a los genotipos que se evalúan, errores de medición o el efecto inesperado de plagas o enfermedades que pueden afecta el rendimiento de algunos genotipos.

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 / ACP 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, Yan et al. 2000). 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 (2009) 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 mediante un ACP.
  - Gabriel Eigen: Arciniegas-Alarcón et al. (2010) presentan un método de imputación que combina regresión y aproximación de rango inferior usando DVS.
  - WGabriel Eigen: Arciniegas-Alarcón et al. (2014) plantean una extensión ponderada del método Gabriel Eigen.

## 3.2. Paquete de R

Un paquete de R consisten en conjuntos de funciones desarrolladas con fines específicos que se distribuyen con un protocolo determinado, garantizando su correcto funcionamiento. 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 utilizan *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

```

1 Package: geneticae
2 Type: Package
3 Title: Statistical analysis tool for agricultural research
4 Version: 0.0.9000
5 Date: 13-04-2019
6 Author: Julia Angelini - Marcos Prunello - Gerardo Cervigni
7 Maintainer: Julia Angelini <jangelini_93@hotmail.com>
8 Description: Original idea was presented in the thesis to obtain the degree of
9 Bioinformatics, National University Rosario (UNR), Rosario Argentina. Some
10 experimental data for the examples come from INTA San Pedro and others research.
11 Geneticae offers extensive statistical analysis tool for agricultural and plant
12 breeding experiments, which can also be useful for other purposes.
13 License: GPL
14 Encoding: UTF-8
15 LazyData: true
16 NeedsCompilation: no
17 Repository: CRAN
18 RoxygenNote: 7.1.1
19 Imports: stats,
20         GGEbiplots,
21         ggforce,
22         ggplot2,
23         scales,
24         MASS,
25         pcaMethods,
26         rrcov,
27         dplyr,
28         bcv,
29         missMDA,
30         calibrate,
31         graphics,
32         agrdat,
33         reshape2,
34         matrixStats,
35         tidyr,
36         prettydoc
37 Suggests:
38         knitr,
39         rmarkdown,
40         testthat (>= 2.1.0)
41 VignetteBuilder: knitr
42 Depends: R (>= 2.12.0)
43 URL: https://github.com/jangelini/geneticae
44 BugReports: https://github.com/jangelini/geneticae/issues
45

```

Figura 3.3: Archivo DESCRIPTION de geneticae

### 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 `use_testthat()` 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 `auto-test()`. 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 `test-nombre_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 unidades de testeo pasen la prueba. En la Figura 3.4 se muestra el resultado de correr los test creados para el paquete `geneticae`.

```
> devtools::test()
Loading geneticeae

Attaching package: 'testthat'

The following object is masked from 'package:devtools':

  test_file

Testing geneticeae
✓ | OK F W S | Context
✓ | 6         | GGE_Model [0.1 s]
✓ | 1         | GGE_Plot
✓ | 4         | impüte [0.3 s]
✓ | 3         | r_AMMI [0.2 s]

— Results —
Duration: 0.7 s

OK:      14
Failed:   0
Warnings: 0
Skipped:  0
> |
```

Figura 3.4: Resultado de correr los tests creados para `geneticae`

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. Por un lado puede evaluarse en cada archivo .R mediante `devtools::test.coverage.file()`, o bien, la cobertura total usando `devtools::test.coverage()`. El paquete `geneticae` tiene un porcentaje total de cobertura de los test igual a 16.83 % (Figura 3.5).

## geneticae coverage - 16.83%

Files		Source					
File	Lines	Relevant	Covered	Missed	Hits / Line	Coverage	
<a href="#">R/GGE_Plot.R</a>	446	233	0	233	0	0.00%	
<a href="#">R/W_GabrielEigen.R</a>	264	131	0	131	0	0.00%	
<a href="#">R/GabrielEigen.R</a>	145	66	0	66	0	0.00%	
<a href="#">R/rAMMI.R</a>	229	121	2	119	0	1.65%	
<a href="#">R/impute.R</a>	188	49	37	12	1	75.51%	
<a href="#">R/GGE_Model.R</a>	107	27	21	6	4	77.78%	
<a href="#">R/EM_AMMI.R</a>	151	59	49	10	6	83.05%	

Figura 3.5: Análisis de cobertura de los tests de genetiae

## ampliar los test para tener mas cobertura

### 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. La diferencia con los testeos realizados mediante `use.testthat()`, es que la última evalúa si las funciones desarrolladas realizan lo deseado, resultando propios de cada paquete. En cambio, lo realizado por R CMD check es común para todos los paquetes.

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 a la hora de desarrollar un paquete, pero que ayudan a la comprensión y difusión del mismo.

---

### 3.2.7.1. Viñetas

Una viñeta es un tipo especial de documentación que puede agregarse al paquete para dar más detalles y ejemplos sobre el uso del mismo. En ella se brinda una descripción del problema que el paquete está diseñado para resolver y muestra al lector cómo resolverlo. Se diferencian de las páginas de ayuda en que su adición es opcional y no sigue una estructura fija, dándole la libertad al autor de enseñar de la forma que más le guste cómo usar su paquete.

Muchos de los paquetes existentes tienen viñetas las 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 `usethis::use_vignette("tutorial_del_paquete")`. La misma crea un directorio `vignettes/`, agrega las dependencias necesarias a `DESCRIPTION` y crea el archivo para redactar la viñeta.

### 3.2.7.2. Agregar datasets

A menudo es útil incluir datos en un paquete a fin de proporcionar ejemplos de las funciones incluidas en él. Esto es posible realizarlo con la función `usethis::use_data()` que crea un archivo `.RData` y lo almacena en el directorio `data/`. Notar que el archivo `DESCRIPTION` 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`.

### 3.2.7.3. Archivo README

Un archivo `README` es una forma de documentación de software que contiene información acerca de otros archivos en una carpeta. 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. Se diferencian de la viñeta ya que sólo presenta una descripción breve del paquete.

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 mismo y será además agregado a `.Rbuildignore`. Luego, al compilarlo con knitr se obtendrá un archivo `README.md`, que será la cara visible del paquete si, por ejemplo, en GitHub.

### badges y logo

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 en el 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)`.

Muchos de los paquetes disponibles disponen de un logo con forma hexagonal, que generalmente termina en forma de sticker: los `hexStickers`. Estos permiten terminar de darle identidad a tu paquete y hacerlo más vistoso. Para crearlos existe 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`. La Figura 3.6 presenta un fragmento GitHub del paquete `geneticae`, donde se muestra parte del contenido del archivo `README`, badges y logo.

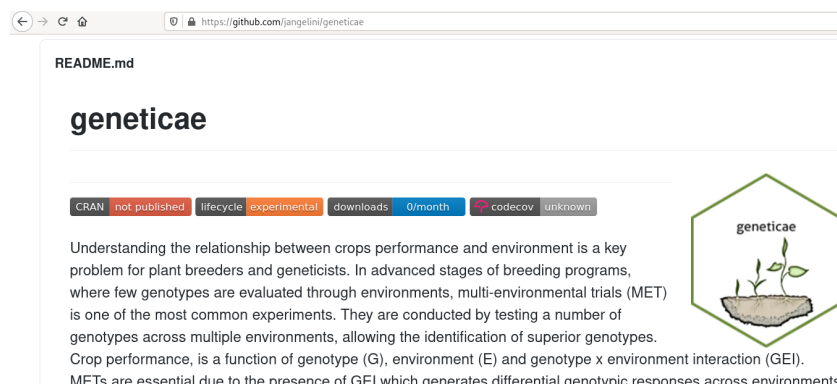


Figura 3.6: Fragmento de README de `geneticae`

---

#### 3.2.7.4. Archivo NEWS

Mientras que el README apunta a ser leído por nuevos usuarios, el archivo NEWS es para aquellos que ya usan el paquete. Este archivo se encarga de contar los cambios presentes en cada versión nueva del paquete que publicamos. 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()`. Hasta el momento, del paquete *geneticae* solamente se cuenta con la versión de desarrollo (Figura 3.7)

```
1 # geneticae 0.0.9000
2
3 * This is the development version of the package.
4
5 .
```

Figura 3.7: Archivo NEWS de *geneticae*

#### 3.2.7.5. Crear una página web

Para mayor visualización del paquete es posible crear una página web <sup>1</sup>. El paquete `pkgdown` está diseñado para la creación de un sitio web de manera rápida y sencilla. Utiliza todo lo creado hasta el momento y lo convierte automáticamente en una página web mediante la función `pkgdown::build_site()`.

## No aparecen los badges en la pag web

#### 3.2.7.6. Publicación

Por último, para que otros usuarios puedan utilizar el paquete es necesario compartirlo en alguno de los repositorios más populares de paquetes de R, entre los cuales se encuentran: CRAN, Bioconductor, GitHub, rOpenSci, R-Forge y RForge. El paquete *geneticae* se encuentra en GitHub, por lo tanto, para instalar el mismo se debe utilizar `devtools::install_github("jangelini/geneticae")`.

---

<sup>1</sup>Para visitar la página web de *geneticae* debe dirigirse a `https://.....`

---

### 3.3. Shiny APP

Shiny es un paquete R que permite construir aplicaciones web directamente desde RStudio 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.

Una característica importante de las aplicaciones web creadas mediante Shiny es que son dinámicas e interactivas. Para que shiny funcione correctamente, es necesario tener instalado R 3.0.2 o cualquier versión posterior.

#### 3.3.1. Estructura de Shiny APP

Las aplicaciones están compuestas por la interfaz de usuario, ui (*user interfaz*), sección server y la función shinyApp().

##### Interfaz del usuario

La interfaz del usuario (user interface o ui, por sus siglas en inglés) controla el diseño de la aplicación, recibe los inputs y muestra los outputs en el navegador. En general, definir las características de la interfaz puede no resultar tan sencillo ya que muchas de sus herramientas están vinculadas a otros lenguajes de programación, por ejemplo HTML, CSS o JavaScript. Sin embargo, las funciones del paquete shiny facilitan la tarea sin necesidad de conocer en profundidad estos lenguajes.

##### Server

En la sección server se escribe el código de R que le indica a la app qué debe hacer y cómo debe funcionar, incluyendo la lectura y manipulación de datos, el armado de gráficos, el ajuste de modelos, etc. Para esto, se define una función que debe tener dos argumentos: input y output. Los mismos son listas que almacenan elementos de entrada (datos u opciones elegidas por el usuario a través de la ui) y elementos de salida para mostrar en la app (resultados, tablas, gráficos, mapas, etc.), respectivamente.

##### Ejecución

Por último, se llama a la función shinyApp(), cuyos dos argumentos principales son ui y server, es decir, cada uno de los elementos definidos anteriormente. Ejecutar esta función da como resultado el lanzamiento de la aplicación, la cual podremos utilizar dentro de RStudio o usando nuestro navegador (Google Chrome, Mozilla Firefox, Microsoft Edge, etc.). Es importante destacar que, al seguir estos pasos, la aplicación sólo funcionará mientras la sesión de RStudio desde la cual se lanzó siga vigente.

---

### 3.3.2. 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, 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.

### **3.3.3. 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](http://www.cefobi.com). Además el proyecto se encuentra en GitHub [https://github.com/jangelini/shinyAPP\\_geneticae](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 sesión 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 del 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
```

- *plr* 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")
```

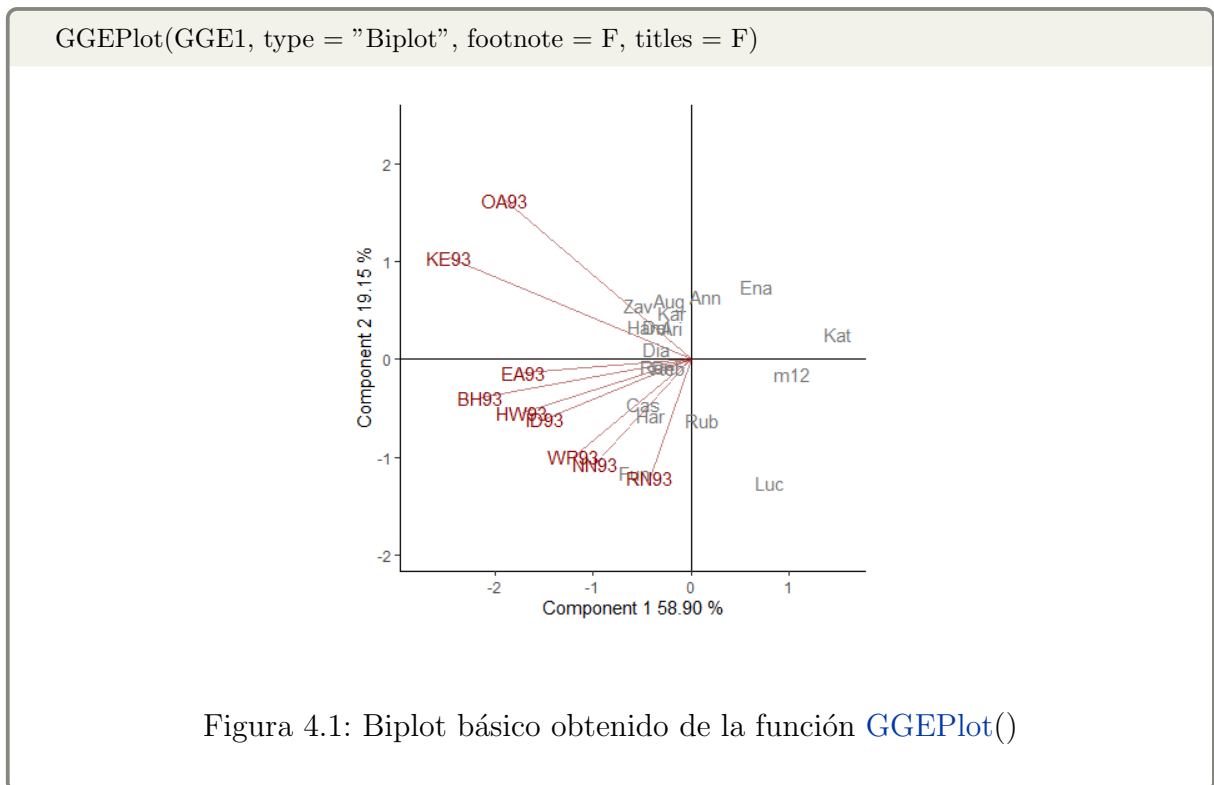
### Biplot GGE

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  $90^\circ$  con todos los ambientes. Por otro lado, fun presenta un rendimiento por encima de la media en todas 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.



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)
```

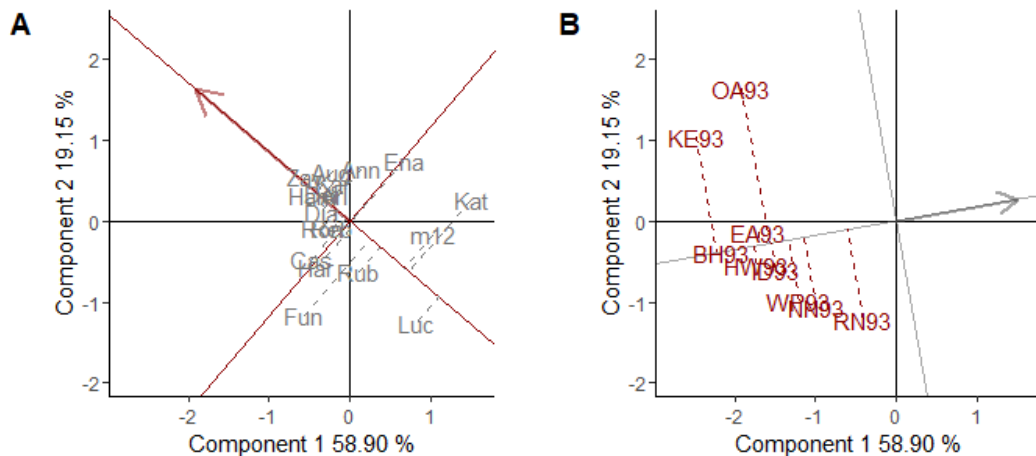
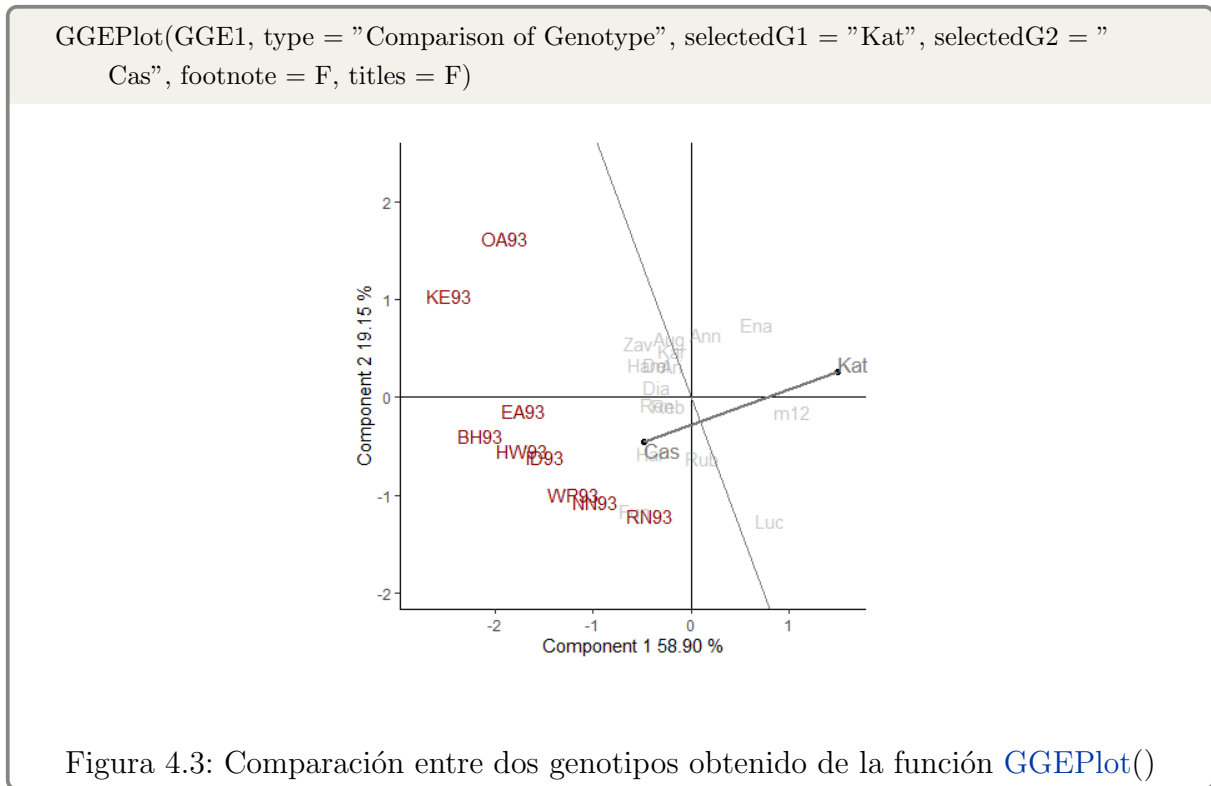


Figura 4.2: A: Ranking de cultivares en el ambiente OA93. B: Ranking de ambientes para cultivar Kat

Para comparar dos cultivares, por ejemplo Kat y Cas, una línea recta que una a los genotipos a comparar se debe trazar y luego una perpendicular a la anterior (figura 4.3). Este biplot se 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.



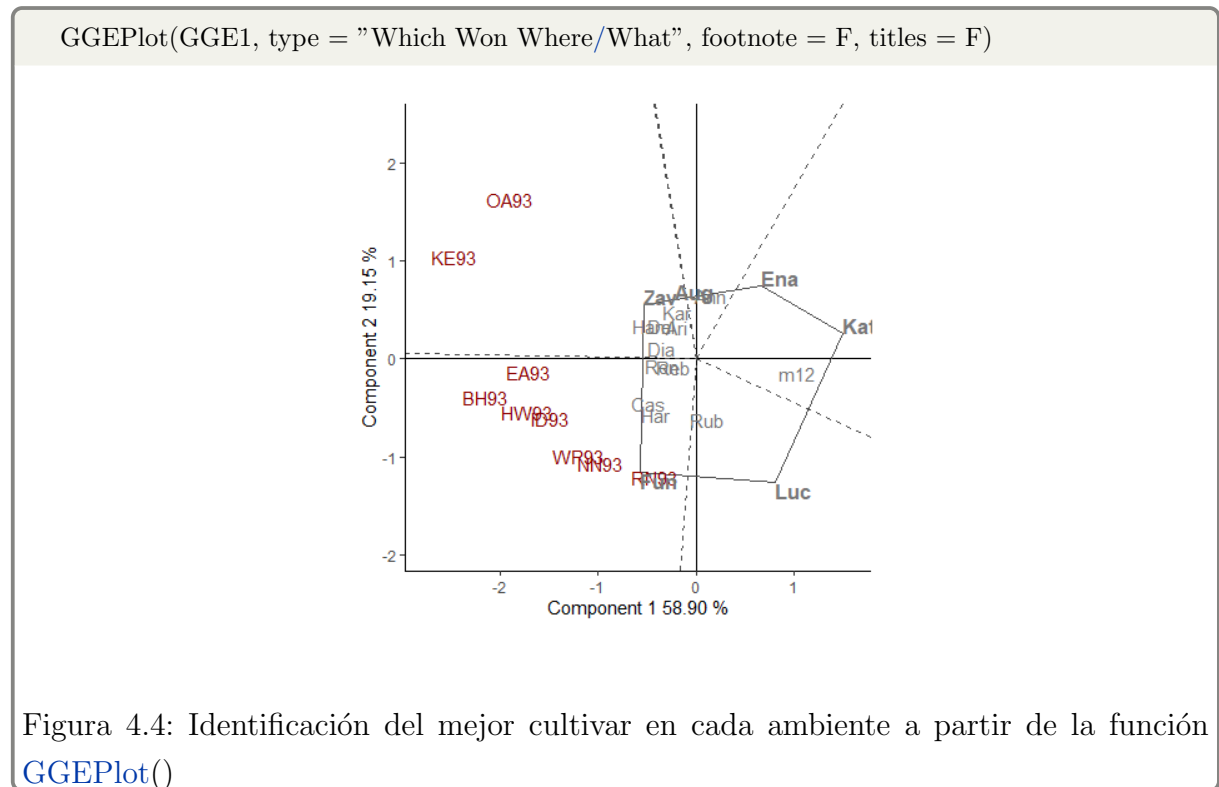
### Identificación de mega-ambientes con GGE biplot

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 “quién 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.



### 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)
```

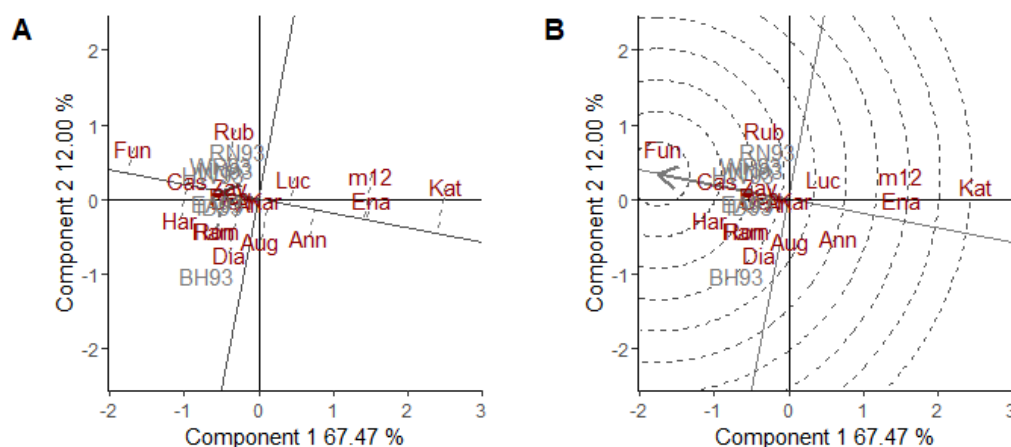


Figura 4.5: A: Evaluación de los cultivares con base en el rendimiento promedio y la estabilidad. B: Clasificación de genotipos con respecto al genotipo ideal

### 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 así 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  $10^\circ$  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  $90^\circ$ . 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 megambiente.

```
# 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)
```

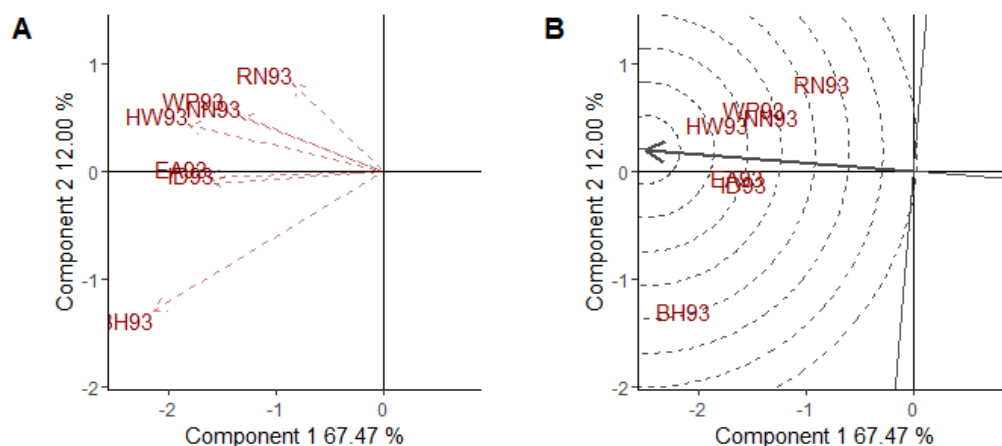


Figura 4.6: A:Relación entre ambientes. B:Clasificación de ambientes con respecto al ambiente ideal

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 descrito 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 utilizada 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 genotipo y el ambiente, 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")
```

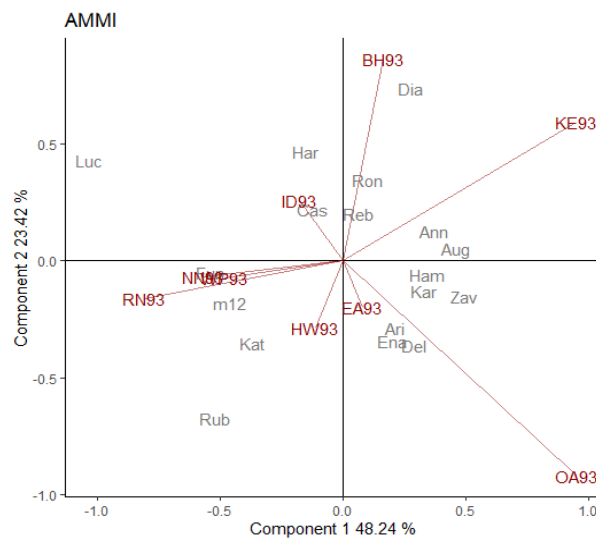


Figura 4.7: Biplot GE obtenido del modelo clasico AMMI

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”, “WGabriel”, “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, *plr* 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 siguiente: *The data* → *Example datasets*.

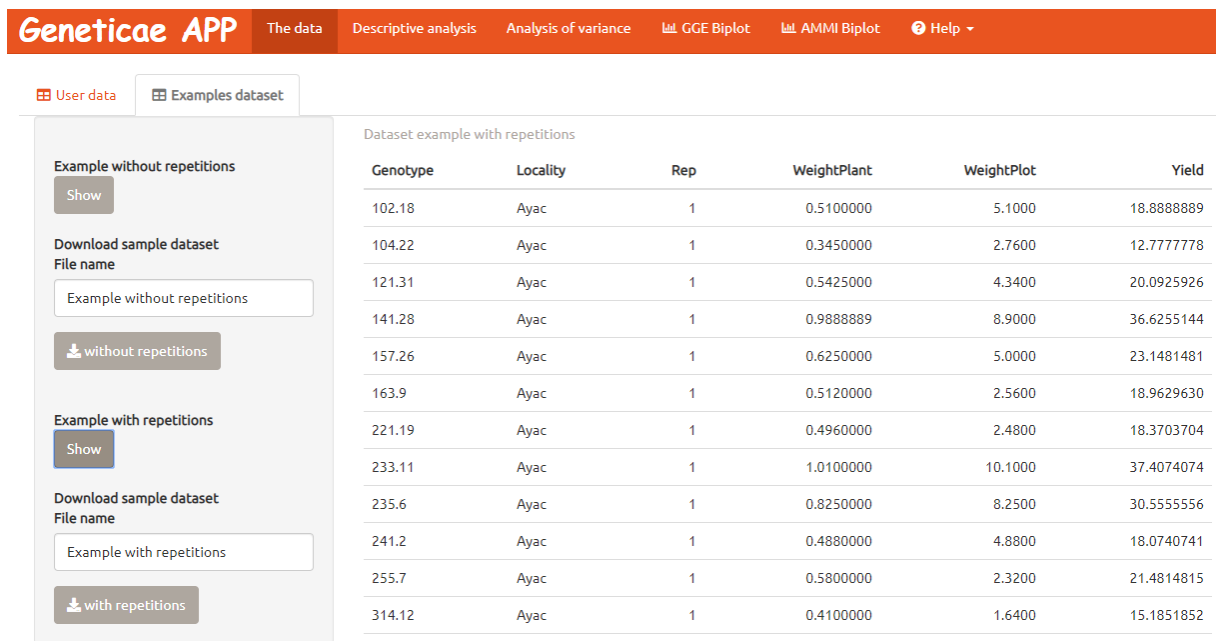


Figura 4.8: yanwinterwheat dataset disponible en Shiny Web App

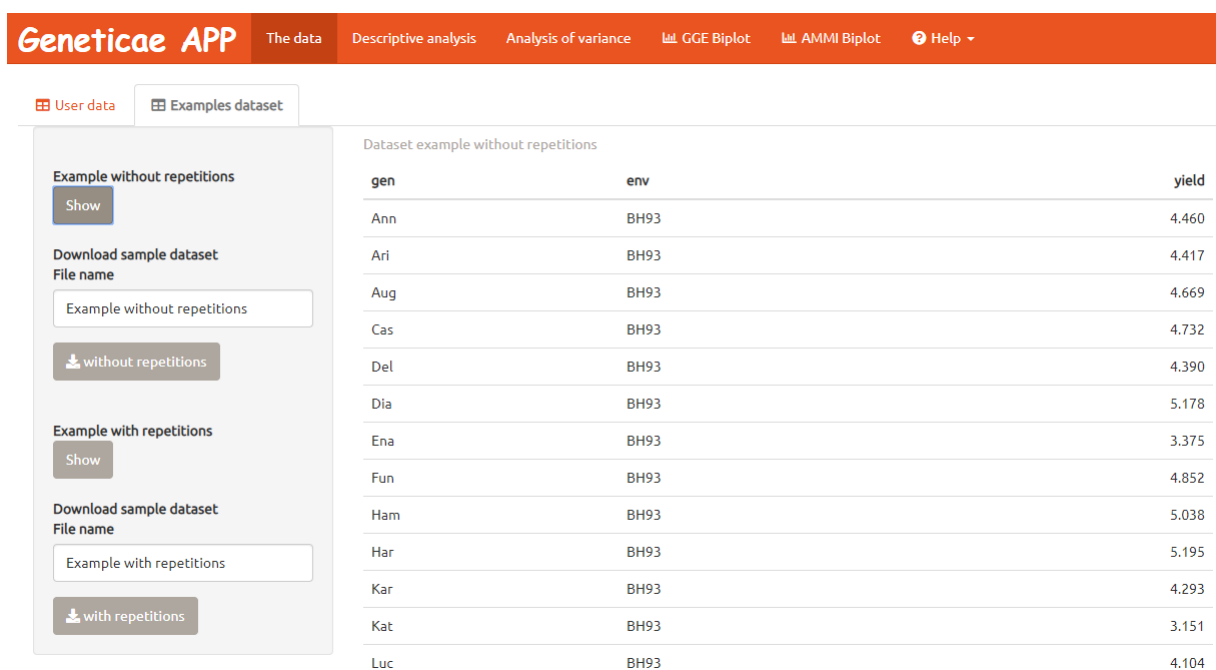


Figura 4.9: plrv dataset disponible en Shiny Web App

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.

The screenshot shows the Geneticae APP interface. The top navigation bar includes 'The data', 'Descriptive analysis', 'Analysis of variance', 'GGE Biplot', 'AMMI Biplot', and 'Help'. The 'User data' tab is active, showing a sidebar for 'User data (.csv format)' with options to browse, upload, and configure the data. The main area displays a table of 10 entries with columns 'gen', 'env', and 'yield'.

gen	env	yield
Ann	BH93	4.46
Ari	BH93	4.417
Aug	BH93	4.669
Cas	BH93	4.732
Del	BH93	4.39
Dia	BH93	5.178
Ena	BH93	3.375
Fun	BH93	4.852
Ham	BH93	5.038
Har	BH93	5.195

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 carácter 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.

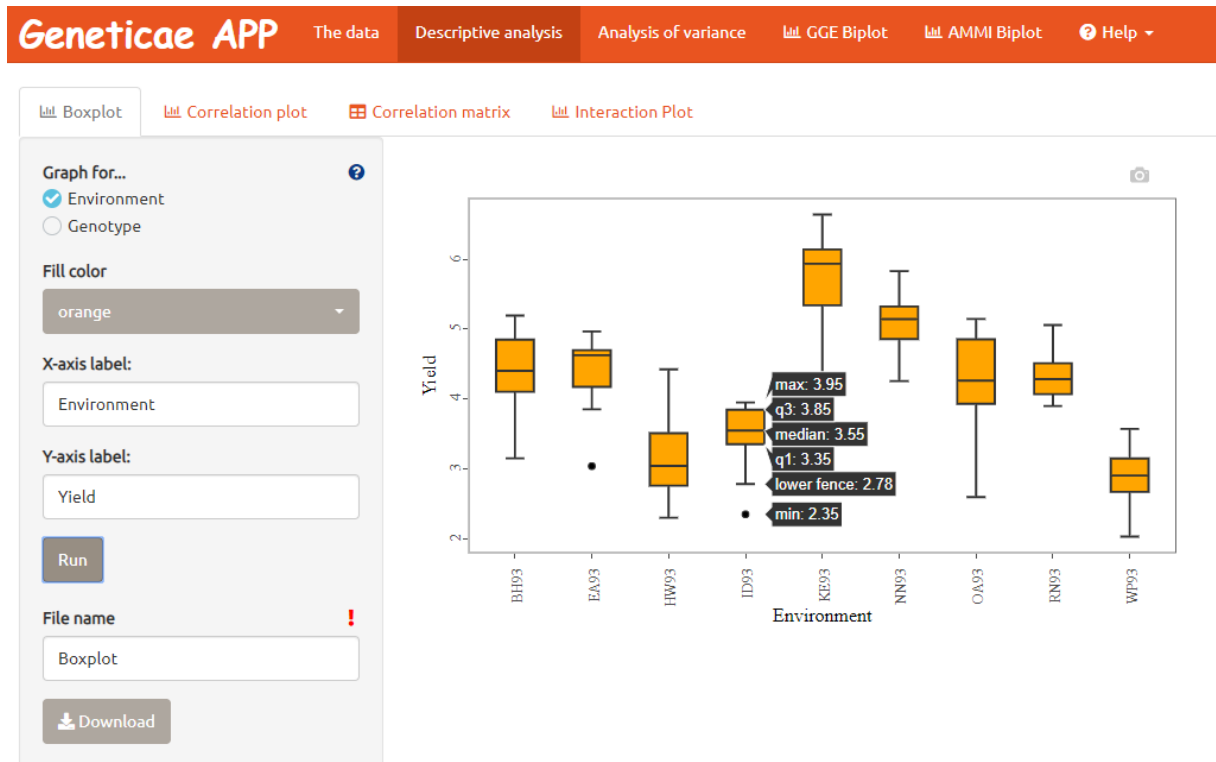


Figura 4.11: Boxplot de ambientes a través de los genotipos para el conjunto de datos Plrv

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.



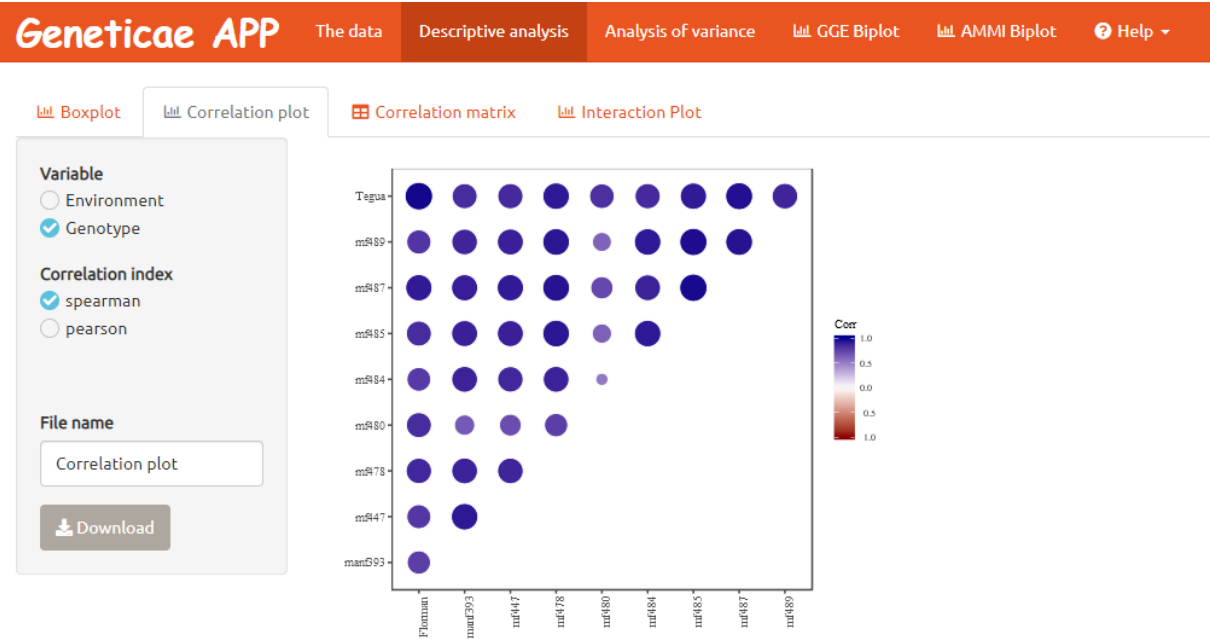


Figura 4.12: Boxplot de genotipos a través de los ambientes para el conjunto de datos Pluv

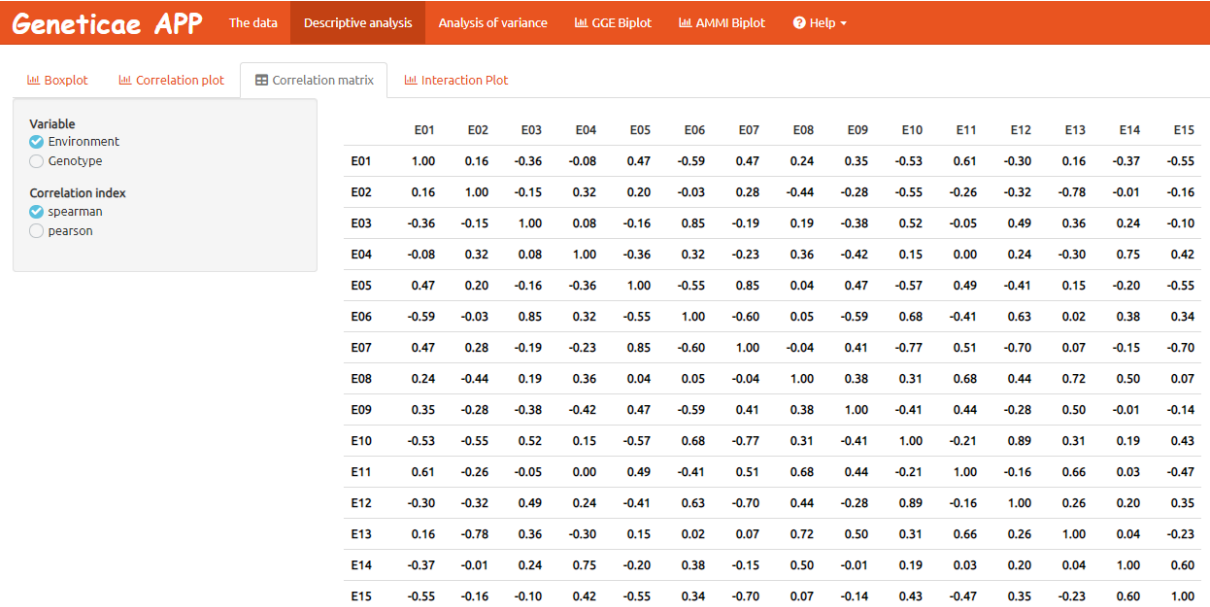


Figura 4.13: Boxplot de genotipos a través de los ambientes para el conjunto de datos Pluv

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.

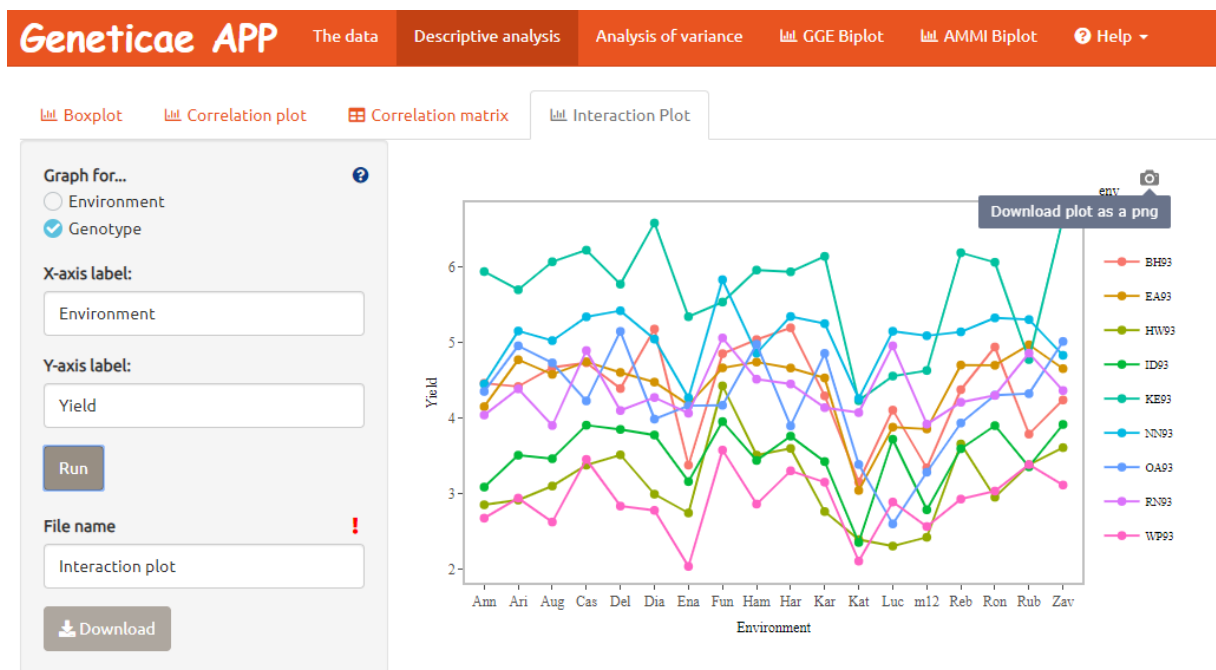


Figura 4.14: Boxplot de genotipos a través de los ambientes para el conjunto de datos Plrv

## 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 probar si existe efecto genotípicos y ambientales (Figura 4.15)

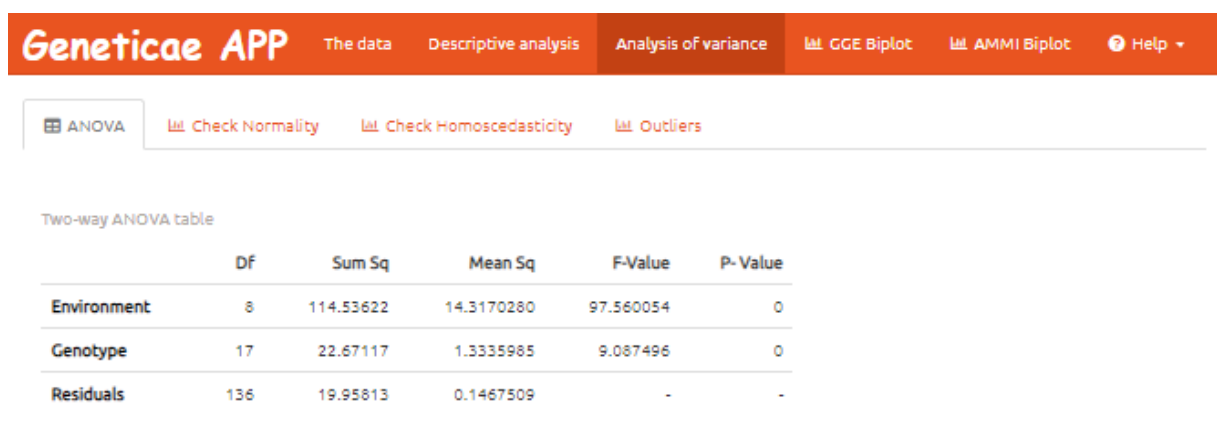


Figura 4.15: Boxplot de genotipos a través de los ambientes para el conjunto de datos Plrv

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 homoscedasticity* 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.

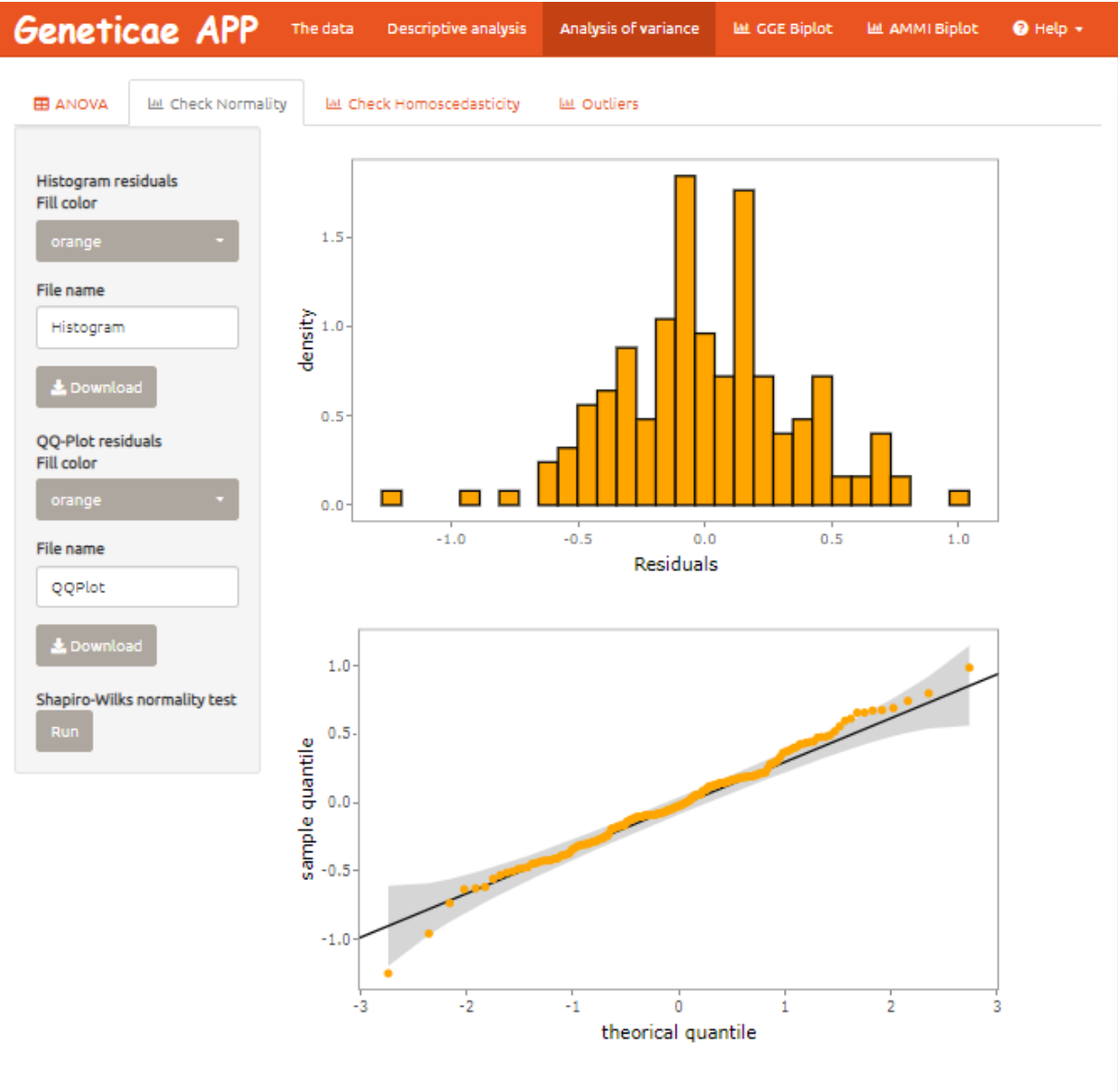


Figura 4.16: Boxplot de genotipos a través de los ambientes para el conjunto de datos Plrv

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).

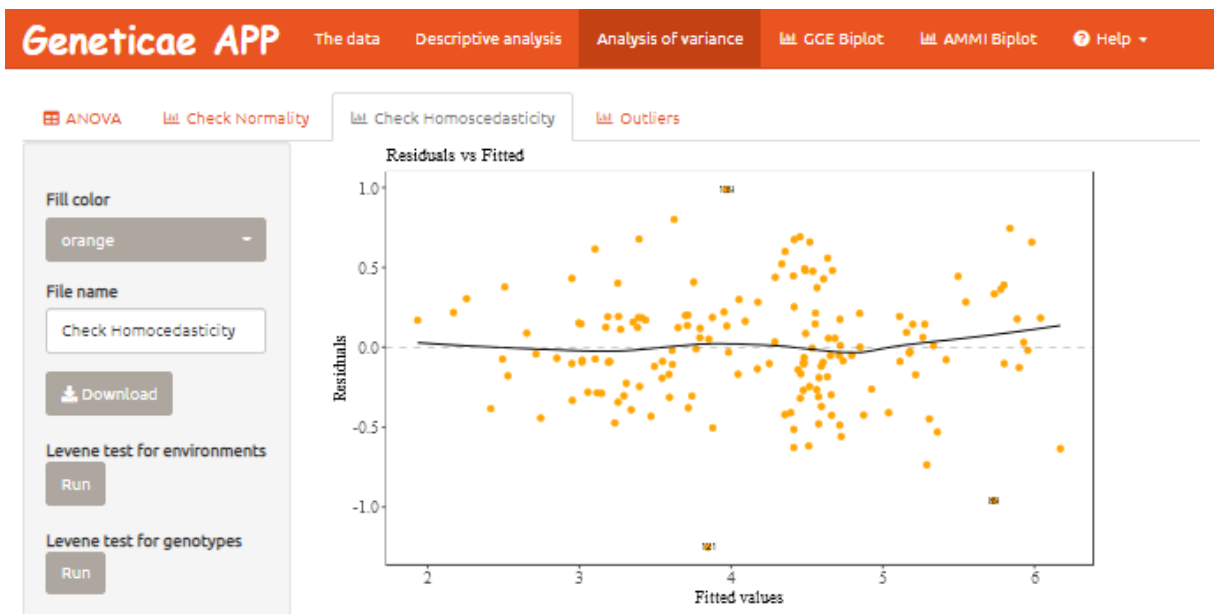


Figura 4.17: Boxplot de genotipos a través de los ambientes para el conjunto de datos Plrv

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).

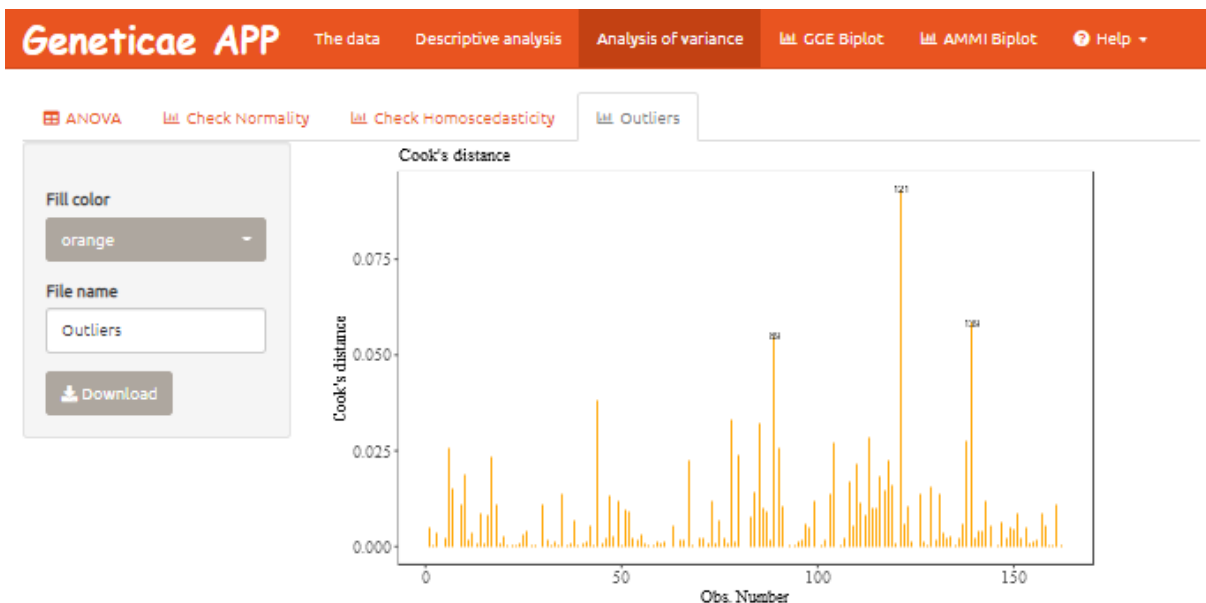


Figura 4.18: Boxplot de genotipos a través de los ambientes para el conjunto de datos Plrv

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.

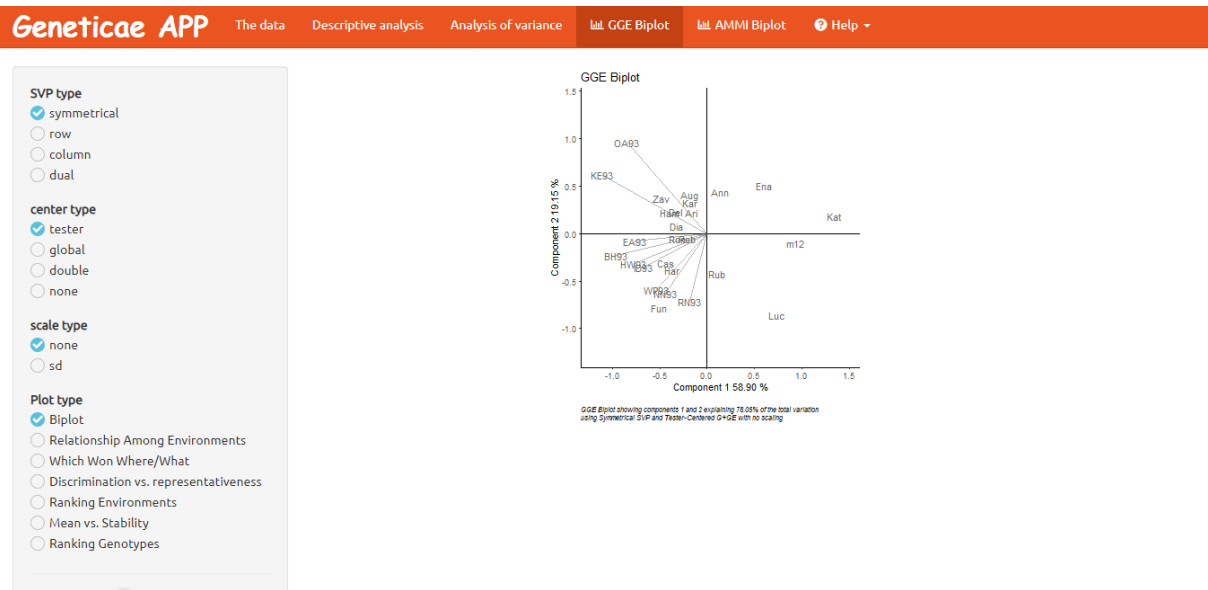


Figura 4.19: Boxplot de genotipos a través de los ambientes para el conjunto de datos Plrv

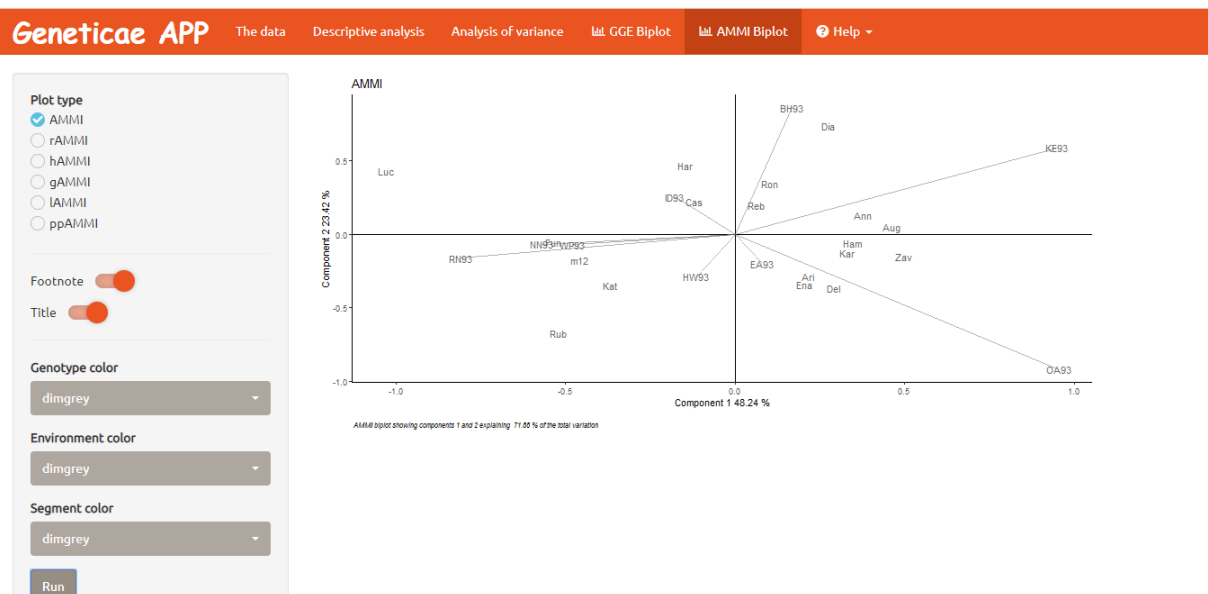


Figura 4.20: AMMI

## Ayuda

Información general, ejemplos y un video tutorial de cómo utilizar la aplicación se encuentran disponibles 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.
- 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.