

Data Analysis with **augmentedRCBD**

Aravind, J.¹, Mukesh Sankar, S.², Wankhede, D. P.³, and Kaur, V.⁴

2022-07-03

1. Division of Germplasm Conservation, ICAR-National Bureau of Plant Genetic Resources, New Delhi.
 2. Division of Genetics, ICAR-Indian Agricultural Research Institute, New Delhi.
 3. Division of Genomic Resources, ICAR-National Bureau of Plant Genetic Resources, New Delhi.
 4. Division of Germplasm Evaluation, ICAR-National Bureau of Plant Genetic Resources, New Delhi.
-

Contents

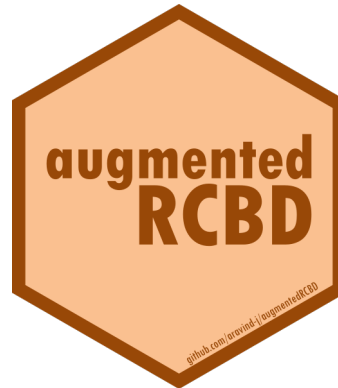
1 Overview	2
2 R software	2
3 Getting Started	3
3.1 Installing R	3
3.2 Installing RStudio	3
3.3 The RStudio Interface	4
3.3.1 Console	5
3.3.2 Source	5
3.3.3 Environment History Connections	5
3.3.4 Files Plots Packages Help Viewer	6
4 Some Basics	6
4.1 Working Directory	6
4.2 Expression and Assignment	6
4.3 Objects and Functions	7
4.3.1 Vector	7
4.3.2 Factor	8
4.3.3 Matrix	10
4.3.4 List	10
4.3.5 Data Frame	11
4.3.6 Functions	11
4.4 Special Elements	12
4.5 Indexing	13
4.6 Help Documentation	15
4.7 Packages	15
4.8 Importing and Exporting Tabular Data	16
4.9 Additional Resources	17
5 Installation of augmentedRCBD	17

6 Data Format	18
7 Data Analysis for a Single Trait	20
7.1 <code>augmentedRCBD()</code>	21
7.2 <code>print.augmentedRCBD()</code>	32
7.3 <code>describe.augmentedRCBD()</code>	35
7.4 <code>freqdist.augmentedRCBD()</code>	37
7.5 <code>gva.augmentedRCBD()</code>	42
7.5 <code>report.augmentedRCBD()</code>	44
8 Data Analysis for a Multiple Traits	45
8.1 <code>augmentedRCBD.bulk()</code>	46
8.2 <code>print.augmentedRCBD.bulk()</code>	49
8.3 <code>report.augmentedRCBD.bulk()</code>	51
9 Citing augmentedRCBD	51
10 Session Info	52
References	53

1 Overview

The software **augmentedRCBD** is built on the **R statistical programming language** as an add-on (or ‘package’ in the *R lingua franca*). It performs the analysis of data generated from experiments in augmented randomised complete block design according to Federer, W.T. (1956, 1956, 1961; 1976). It also computes analysis of variance, adjusted means, descriptive statistics, genetic variability statistics etc. and includes options for data visualization and report generation.

This tutorial aims to educate the users in utilising this package for performing such analysis. Utilising **augmentedRCBD** for data analysis requires a basic knowledge of **R** programming language. However, as many of the intended end-users may not be familiar with **R**, [sections 2 to 4](#) give a ‘gentle’ introduction to **R**, especially those aspects which are necessary to get **augmentedRCBD** up and running for performing data analysis in a Windows environment. Users already familiar with **R** can feel free to skip to [section 5](#).



2 R software

It is a free software environment for statistical computing and graphics. It is free and open source, platform independent (works on Linux, Windows or MacOS), very flexible, comprehensive with robust interfaces for all the popular programming languages as well as databases. It is strengthened by its diverse library of add-on packages extending its ability as well as the incredible community support. It is one of the most popular tools being used in academia today ([Tippmann, 2015](#)).



3 Getting Started

This section details the steps required to set up the R programming environment under a third-party interface called RStudio in Windows.

3.1 Installing R

Download and install R for Windows from <http://cran.r-project.org/bin/windows/base/>.

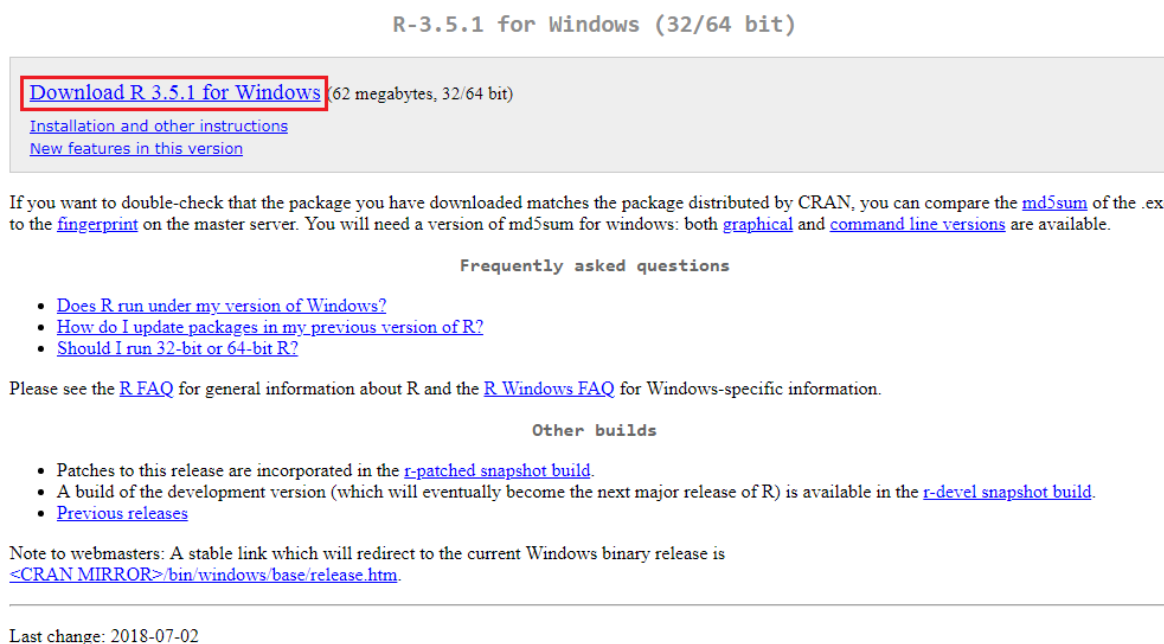


Fig. 1: The R download location.

3.2 Installing RStudio

The basic `command line interface` in native R is rather limiting. There are several interfaces which enhance it's functionality and ease of use, `RStudio` being one of the most popular among R programmers.

Download and install RStudio for Windows from <https://www.rstudio.com/products/rstudio/download/#download>

Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.1.456 - Windows Vista/7/8/10	85.8 MB	2018-07-19	24ca3fe0dad8187aabd4bfb9dc2b5ad
RStudio 1.1.456 - Mac OS X 10.6+ (64-bit)	74.5 MB	2018-07-19	4fc4f4f70845b142bf96dc1a5b1dc556
RStudio 1.1.456 - Ubuntu 12.04-15.10/Debian 8 (32-bit)	89.3 MB	2018-07-19	3493f9d5839e3a3d697f40b7bb1ce961
RStudio 1.1.456 - Ubuntu 12.04-15.10/Debian 8 (64-bit)	97.4 MB	2018-07-19	863ae806120358fa0146e4d14cd75be4
RStudio 1.1.456 - Ubuntu 16.04+/Debian 9+ (64-bit)	64.9 MB	2018-07-19	d96e63548c2add890bac633bdb883f32
RStudio 1.1.456 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	88.1 MB	2018-07-19	1df56c7cd80e2634f8a9fdd11ca1fb2d
RStudio 1.1.456 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	90.6 MB	2018-07-19	5e77094a88fdbdddb0d35708752462

Zip/Tarballs

Zip/tar archives	Size	Date	MD5
RStudio 1.1.456 - Windows Vista/7/8/10	122.9 MB	2018-07-19	659d6bfe716d8c97acbe501270d89fa3
RStudio 1.1.456 - Ubuntu 12.04-15.10/Debian 8 (32-bit)	90 MB	2018-07-19	63117c159deca4d01221a8069bd45373
RStudio 1.1.456 - Ubuntu 12.04-15.10/Debian 8 (64-bit)	98.3 MB	2018-07-19	c53c32a71a400c6571e36c573f83dfde
RStudio 1.1.456 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	88.8 MB	2018-07-19	f4ba2509fb00e30c91414c6821f1c85f
RStudio 1.1.456 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	91.4 MB	2018-07-19	c60db6467421aa86c772227da0945a13

Source Code

A tarball containing source code for RStudio v1.1.456 can be downloaded from [here](#)

Fig. 2: The RStudio download location.

3.3 The RStudio Interface

On opening RStudio, the default interface with four panes/windows is visible as follows. Few panes have different tabs.



Fig. 3: The default RStudio interface with the four panes.

3.3.1 Console

This is where the action happens. Here any authentic R code typed after the `>` prompt will be executed after pressing ‘Enter’ to generate the output.

For example, type `1+1` in the console and press ‘Enter’.

```
1+1
```

```
[1] 2
```

3.3.2 Source

This is where R Scripts (collection of code) can be created and edited. R scripts are text files with a `.R` extension. R Code for analysis can be typed and saved in such R scripts. New scripts can be opened by clicking ‘File|New File’ and selecting ‘R Script’. Code can be selected from R Scripts and sent to console for evaluation by clicking ‘Run’ on the ‘Source’ pane or by pressing ‘Ctrl + Enter’.

3.3.3 Environment|History|Connections

The ‘Environment’ tab shows the list of all the ‘objects’ (see [section 4.3](#)) defined in the current R session. It has also some buttons up top to open, save and clear the environment as well as few options for import of data under `Import Dataset`.

The ‘History’ tab shows a history of all the code that was previously evaluated. This is useful, if you want to go back to some code.

The ‘Connections’ tab helps to establish and manage connections with different databases and data sources.

3.3.4 Files|Plots|Packages|Help|Viewer

The ‘Files’ tab shows a sleek file browser to access the file directory in the computer with options to manage the working directory (see [section 4.1](#)) under the More button.

The ‘Plots’ tab shows all the plots generated in R with buttons to delete unnecessary ones and export useful ones as a pdf file or as an image file.

The ‘Packages’ tab shows a list of all the R add-on packages installed. The check box on the left shows whether they are loaded or not. There are also buttons to install and update R packages.

The ‘Viewer’ tab shows any web content output generated by an R code.

4 Some Basics

This section describes some basics to enable the users to have a working knowledge in R in order to use **augmentedRCBD**.

4.1 Working Directory

It is a file path to a folder on the computer which is recognised by R as the default location to read files from or write files to. The code `getwd()` shows the current working directory, while `setwd()` can be used to change the existing working directory.

```
# Print current working directory
getwd()
```

```
[1] "C:/Users/Computer/Documents"
```

```
# Set new working directory
setwd("C:/Data Analysis/")
getwd()
```

```
[1] "C:/Data Analysis/"
```

One key detail is that file paths in R uses forward slashes (/) as in MacOS or Linux, unlike backward slashes (\) in Windows. This needs to be considered while copying paths from default Windows file explorer.

4.2 Expression and Assignment

Expressions are instructions in the form of code to be entered after the > prompt in the console. Expressions can be a constant, an arithmetic or a condition. A more advanced and most useful expression is a function call (see [section 4.3](#)).

```
# Constant
123
```

```
[1] 123
```

```
# Arithmetic (add two numbers)
1 + 2
```

```
[1] 3
```

```
# Condition
```

```
34 > 25
```

```
[1] TRUE
```

```
1 == 2
```

```
[1] FALSE
```

```
# Function call (mean of a series of numbers)
```

```
mean(c(25,56,89,35))
```

```
[1] 51.25
```

Information from an expression can be stored as an ‘object’ (see [section 4.3](#)) by assigning a name using the operator ‘<-’.

```
# Assign the result of the expression 1 + 2 to an object 'a'
```

```
a <- 1 + 2
```

```
a
```

```
[1] 3
```

It is recommended to add comments to explain the code by using the ‘#’ sign. Any code after the ‘#’ sign will be ignored by R.

4.3 Objects and Functions

R is an object-oriented programming language (OOP). Any kind or construct created in R is an ‘object’. Each object has a ‘class’ (shown using the `class()` function) and different ‘attributes’ which defines what operations can be done on that object. There are different types of data structure objects in R such as vectors, matrices, factors, data frames, and lists. A ‘function’ is also an object, which defines a procedure or a sequence of expressions.

4.3.1 Vector

A vector is a collection of elements of a single type (or ‘mode’). The common vector modes are ‘numeric’, ‘integer’, ‘character’ and ‘logical’. The `c()` function is used to create vectors. The functions `class()`, `str()` and `length()` show the attributes of vectors.

Vector modes ‘numeric’ stores real numbers, while ‘integer’ stores integers, which can be enforced by suffixing elements with ‘L’.

```
# A numeric vector
```

```
a <- c(1, 2, 3.3)
```

```
class(a)
```

```
[1] "numeric"
```

```
str(a)
```

```
num [1:3] 1 2 3.3
```

```
length(a)
```

```
[1] 3
```

```
# An integer vector
```

```
b <- c(1L, 2L, 3L)
```

```
class(b)
```

```
[1] "integer"
```

```
str(b)
```

```
int [1:3] 1 2 3
```

```
length(b)
```

```
[1] 3
```

The vector mode 'character' store text.

```
# A character vector
```

```
c <- c("one", "two", "three")
```

```
class(c)
```

```
[1] "character"
```

```
str(c)
```

```
chr [1:3] "one" "two" "three"
```

```
length(c)
```

```
[1] 3
```

The vector mode 'logical' stores 'TRUE' OR 'FALSE' logical data.

```
#logical vector
```

```
d <- c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE)
```

```
class(d)
```

```
[1] "logical"
```

```
str(d)
```

```
logi [1:6] TRUE TRUE TRUE FALSE TRUE FALSE
```

```
length(d)
```

```
[1] 6
```

4.3.2 Factor

A 'factor' in R stores data from categorical data in variables as different levels.

```
catg <- c("male", "female", "female", "male", "male")
```

```
catg
```

```
[1] "male" "female" "female" "male" "male"
```

```
is.factor(catg)
```

```
[1] FALSE
```

```
# Apply the factor function
```

```
factor_catg <- factor(catg)
```

```
factor_catg
```



```
[1] male   female female male   male
Levels: female male
```

```
is.factor(factor_catg)
```

```
[1] TRUE
```

```
class(factor_catg)
```

```
[1] "factor"
```

```
str(factor_catg)
```

```
Factor w/ 2 levels "female","male": 2 1 1 2 2
```

A character, numeric or integer vector can be transformed to a factor by using the `as.factor()` function.

```
# Conversion of numeric to factor
```

```
a <- c(1, 2, 3.3)
```

```
class(a)
```

```
[1] "numeric"
```

```
str(a)
```

```
num [1:3] 1 2 3.3
```

```
fac_a <- as.factor(a)
```

```
class(fac_a)
```

```
[1] "factor"
```

```
str(fac_a)
```

```
Factor w/ 3 levels "1","2","3.3": 1 2 3
```

```
# Conversion of integer to factor
```

```
b <- c(1L, 2L, 3L)
```

```
class(b)
```

```
[1] "integer"
```

```
str(b)
```

```
int [1:3] 1 2 3
```

```
fac_b <- as.factor(b)
```

```
class(fac_b)
```

```
[1] "factor"
```

```
str(fac_b)
```

```
Factor w/ 3 levels "1","2","3": 1 2 3
```

```
# Conversion of character to factor
```

```
c <- c("one", "two", "three")
```

```
class(c)
```

```
[1] "character"
```

```
str(c)

chr [1:3] "one" "two" "three"
fac_c <- as.factor(c)
class(fac_c)

[1] "factor"
str(fac_c)

Factor w/ 3 levels "one","three",...: 1 3 2
```

4.3.3 Matrix

A ‘matrix’ in R is a vector with the attributes ‘nrow’ and ‘ncol’.

```
# Generate 5 * 4 numeric matrix
m <- matrix(1:20, nrow = 5, ncol = 4)
m
```

```
      [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
```

```
class(m)

[1] "matrix" "array"
typeof(m)
```

```
[1] "integer"
# Dimensions of m
dim(m)
```

```
[1] 5 4
```

4.3.4 List

A ‘list’ is a container containing different objects. The contents of list need not be of the same type or mode. A list can encompass a mixture of data types such as vectors, matrices, data frames, other lists or any other data structure.

```
w <- list(a, m, d, list(b, c))
class(w)

[1] "list"
str(w)

List of 4
 $ : num [1:3] 1 2 3.3
 $ : int [1:5, 1:4] 1 2 3 4 5 6 7 8 9 10 ...
 $ : logi [1:6] TRUE TRUE TRUE FALSE TRUE FALSE
 $ :List of 2
```

```
..$ : int [1:3] 1 2 3
..$ : chr [1:3] "one" "two" "three"
```

4.3.5 Data Frame

A ‘data frame’ in R is a special kind of list with every element having equal length. It is very important for handling tabular data in R. It is a array like structure with rows and columns. Each column needs to be of a single data type, however data type can vary between columns.

```
L <- LETTERS[1:4]
y <- 1:4
z <- c("This", "is", "a", "data frame")
df <- data.frame(L, x = 1, y, z)
df
```

```
  L x y      z
1 A 1 1    This
2 B 1 2      is
3 C 1 3       a
4 D 1 4 data frame
```

```
str(df)
```

```
'data.frame':  4 obs. of  4 variables:
 $ L: chr  "A" "B" "C" "D"
 $ x: num  1 1 1 1
 $ y: int  1 2 3 4
 $ z: chr  "This" "is" "a" "data frame"
```

```
attributes(df)
```

```
$names
```

```
[1] "L" "x" "y" "z"
```

```
$class
```

```
[1] "data.frame"
```

```
$row.names
```

```
[1] 1 2 3 4
```

```
rownames(df)
```

```
[1] "1" "2" "3" "4"
```

```
colnames(df)
```

```
[1] "L" "x" "y" "z"
```

4.3.6 Functions

All of the work in R is done by functions. It is an object defining a procedure which takes one or more objects as input (or ‘arguments’), performs some action on them and finally gives a new object as output (or ‘return’). `class()`, `mean()`, `getwd()`, `+`, etc. are all functions.

For example the function `mean()` takes a numeric vector as argument and returns the mean as a numeric vector.

```
a <- c(1, 2, 3.3)
mean(a)
```

```
[1] 2.1
```

The user can also create custom functions. For example the function `foo` adds two numbers and gives the result.

```
foo <- function(n1, n2) {
  out <- n1 + n2
  return(out)
}
foo(2,3)
```

```
[1] 5
```

4.4 Special Elements

In addition to numbers and text, there are some special elements which can be included in different data objects.

NA (not available) indicates missing data.

```
x <- c(2.5, NA, 8.6)
y <- c(TRUE, FALSE, NA)
z <- c("k", NA, "m", "n", "o")
is.na(x)
```

```
[1] FALSE TRUE FALSE
```

```
is.na(z)
```

```
[1] FALSE TRUE FALSE FALSE FALSE
```

```
anyNA(x)
```

```
[1] TRUE
```

```
a
```

```
[1] 1.0 2.0 3.3
```

```
is.na(a)
```

```
[1] FALSE FALSE FALSE
```

Inf indicates infinity.

```
1/0
```

```
[1] Inf
```

NaN (Not a Number) indicates any undefined value.

```
0/0
```

```
[1] NaN
```

4.5 Indexing

The `[]` function is used to extract elements of an object by indexing (numeric or logical). Named elements in lists and data frames can be extracted by using the `$` operator.

Consider a vector `a`.

```
a <- c(1, 2, 3.3, 2.8, 6.7)
# Numeric indexing
# Extract first element
a[1]
```

```
[1] 1
# Extract elements 2:3
a[2:3]
```

```
[1] 2.0 3.3
# Logical indexing
a[a > 3]
```

```
[1] 3.3 6.7
```

Consider a matrix `m`.

```
m <- matrix(1:9, nrow = 3, ncol = 3, byrow = TRUE)
colnames(m) <- c('a', 'b', 'c')
m
```

```
      a b c
[1,] 1 2 3
[2,] 4 5 6
[3,] 7 8 9
```

```
# Extract elements
m[,2] # 2nd column of matrix
```

```
[1] 2 5 8
m[3,] # 3rd row of matrix
```

```
a b c
7 8 9
m[2:3, 1:3] # rows 2,3 of columns 1,2,3
```

```
      a b c
[1,] 4 5 6
[2,] 7 8 9
m[2,2] # Element in 2nd column of 2nd row
```

```
b
5
m[, 'b'] # Column 'b'

[1] 2 5 8
```

```
m[, c('a', 'c')] # Column 'a' and 'c'
```

```
      a c
[1,] 1 3
[2,] 4 6
[3,] 7 9
```

Consider a list w.

```
w <- list(vec = a, mat = m, data = df, alist = list(b, c))
```

```
# Indexing by number
w[2] # As list structure
```

```
$mat
      a b c
[1,] 1 2 3
[2,] 4 5 6
[3,] 7 8 9
```

```
w[[2]] # Without list structure
```

```
      a b c
[1,] 1 2 3
[2,] 4 5 6
[3,] 7 8 9
```

```
# Indexing by name
w$vec
```

```
[1] 1.0 2.0 3.3 2.8 6.7
```

```
w$data
```

```
  L x y      z
1 A 1 1    This
2 B 1 2      is
3 C 1 3       a
4 D 1 4 data frame
```

Consider a data frame df.

```
df
```

```
  L x y      z
1 A 1 1    This
2 B 1 2      is
3 C 1 3       a
4 D 1 4 data frame
```

```
# Indexing by number
df[,2] # 2nd column of data frame
```

```
[1] 1 1 1 1
```

```
df[2] # 2nd column of data frame
```

```
  x
1 1
```

```
2 1
3 1
4 1
```

```
df[3,] # 3rd row of data frame
```

```
  L x y z
3 C 1 3 a
```

```
df[2:3, 1:3] # rows 2,3 of columns 1,2,3
```

```
  L x y
2 B 1 2
3 C 1 3
```

```
df[2,2] # Element in 2nd column of 2nd row
```

```
[1] 1
```

```
# Indexing by name
```

```
df$L
```

```
[1] "A" "B" "C" "D"
```

```
df$z
```

```
[1] "This"      "is"        "a"         "data frame"
```

4.6 Help Documentation

The help documentation regarding any function can be viewed using the `?` or `help()` function. The help documentation shows the default usage of the function including, the arguments that are taken by the function and the type of output object returned ('Value').

```
?ls
help(ls)

?mean

?setwd
```

4.7 Packages

Packages in R are collections of R functions, data, and compiled code in a well-defined format. They are add-ons which extend the functionality of R and at present, there are **18310** packages available for deployment and use at the official repository, the Comprehensive R Archive Network (CRAN).

Valid packages from CRAN can be installed by using the `install.packages()` command.

```
# Install the package 'readxl' for importing data from excel
install.packages(readxl)
```

Installed packages can be loaded using the function `library()`.

```
# Install the package 'readxl' for importing data from excel
library(readxl)
```

4.8 Importing and Exporting Tabular Data

Tabular data from a spreadsheet can be imported into R in different ways. Consider some data such as in Table 1. Copy this data in to a spreadsheet editor such as MS Excel and save it as `augdata.csv`, a comma-separated-value file and `augdata.xlsx`, an Excel file in the working directory (`getwd()`).

Table 1: Example data from an experiment in augmented RCBD design.

blk	trt	y1	y2
I	1	92	258
I	2	79	224
I	3	87	238
I	4	81	278
I	7	96	347
I	11	89	300
I	12	82	289
II	1	79	260
II	2	81	220
II	3	81	237
II	4	91	227
II	5	79	281
II	9	78	311
III	1	83	250
III	2	77	240
III	3	78	268
III	4	78	287
III	8	70	226
III	6	75	395
III	10	74	450

The `augdata.csv` file can be imported into R using the `read.csv()` function or the `read_csv()` function in the `readr` package.

```
data <- read.csv(file = "augdata.csv")
str(data)
```

```
'data.frame': 20 obs. of 4 variables:
 $ blk: Factor w/ 3 levels "I","II","III": 1 1 1 1 1 1 1 2 2 2 ...
 $ trt: num 1 2 3 4 7 11 12 1 2 3 ...
 $ y1 : num 92 79 87 81 96 89 82 79 81 81 ...
 $ y2 : num 258 224 238 278 347 300 289 260 220 237 ...
```

The argument `stringsAsFactors = FALSE` reads the text columns as of type `character` instead of the default `factor`.

```
data <- read.csv(file = "augdata.csv", stringsAsFactors = FALSE)
str(data)
```

```
'data.frame': 20 obs. of 4 variables:
 $ blk: chr "I" "I" "I" "I" ...
 $ trt: num 1 2 3 4 7 11 12 1 2 3 ...
 $ y1 : num 92 79 87 81 96 89 82 79 81 81 ...
 $ y2 : num 258 224 238 278 347 300 289 260 220 237 ...
```


The `augdata.xlsx` file can be imported into R using the `read_excel()` function in the `readxl` package.

```
library(readxl)
data <- read_excel(path = "augdata.xlsx")

'data.frame':  20 obs. of  4 variables:
 $ blk: chr  "I" "I" "I" "I" ...
 $ trt: num  1 2 3 4 7 11 12 1 2 3 ...
 $ y1 : num  92 79 87 81 96 89 82 79 81 81 ...
 $ y2 : num  258 224 238 278 347 300 289 260 220 237 ...
```

The tabular data can be exported from R to a `.csv` (comma-separated-value) file by the `write.csv()` function.

```
write.csv(x = data, file = "augdata.csv")
```

4.9 Additional Resources

To learn more about R, there are umpteen number of online tutorials as well as free courses available. Queries about various aspects can be put to the active and vibrant ‘R community online.

- Online tutorials
 - <http://www.cran.r-project.org/other-docs.html>
 - <https://bookdown.org/ndphillips/YaRrr/>
- Free online courses
 - <http://tryr.codeschool.com/>
 - <https://www.datacamp.com/courses/free-introduction-to-r>
- R community support
 - <http://stackoverflow.com/>
 - R help mailing lists : <http://www.r-project.org/mail.html>

5 Installation of `augmentedRCBD`

The package `augmentedRCBD` can be installed using the following functions.

```
# Install from CRAN
install.packages('augmentedRCBD', dependencies=TRUE)

# Install development version from Github
if (!require('devtools')) install.packages('devtools')
library(devtools)
install_github("aravind-j/augmentedRCBD")
```

The stable release is hosted in [CRAN](#) (see [section 4.7](#)), while the under-development version is hosted as a [Github](#) repository. To install from github, you need to use the `install_github()` function from ‘`devtools`’ package.

Then the package can be loaded using the function

```
library(augmentedRCBD)
```

The current version of the package is 0.1.5. The previous versions are as follows.

Table 2. Version history of `augmentedRCBD` R package.

Version	Date
0.1.0	2018-07-10
0.1.1	2019-07-21
0.1.2	2020-03-19
0.1.3	2020-07-27
0.1.4	2021-02-17

To know detailed history of changes use `news(package='augmentedRCBD')`.

6 Data Format

Certain details need to be considered for arranging experimental data for analysis using the **augmentedRCBD** package.

The data should be in long/vertical form, where each row has the data from one genotype per block. For example, consider the following data (Table 3) recorded for a trait from an experiment laid out in an augmented block design with 3 blocks and 12 genotypes(or treatment) with 6 to 7 genotypes/block. 8 genotypes (Test, G 5 to G 12) are not replicated, while 4 genotypes (Check, G 1 to G 4) are replicated.

Table 3: Data from an experiment in augmented RCBD design.

Block I	G12	G4	G11	G2	G1	G7	G3
	82	81	89	79	92	96	87
Block II	G5	G9	–	G3	G1	G2	G4
	79	78	–	81	79	81	91
Block III	G4	G2	G1	G6	G10	G3	G8
	78	77	83	75	74	78	70

This data needs to be arranged with columns showing block, genotype (or treatment) and the data of the trait for each genotype per block (Table 4).

Table 4: Data from an experiment in augmented RCBD design arranged in long-form.

Block	Treatment	Trait
Block I	G 1	92
Block I	G 2	79
Block I	G 3	87
Block I	G 4	81
Block I	G 7	96
Block I	G 11	89
Block I	G 12	82
Block II	G 1	79
Block II	G 2	81
Block II	G 3	81
Block II	G 4	91
Block II	G 5	79
Block II	G 9	78
Block III	G 1	83
Block III	G 2	77

Block	Treatment	Trait
Block III	G 3	78
Block III	G 4	78
Block III	G 8	70
Block III	G 6	75
Block III	G 10	74

The data for block and genotype (or treatment) can also be depicted as numbers (Table 5).

Table 5: Data from an experiment in augmented RCBD design arranged in long-form (Block and Treatment as numbers).

Block	Treatment	Trait
1	1	92
1	2	79
1	3	87
1	4	81
1	7	96
1	11	89
1	12	82
2	1	79
2	2	81
2	3	81
2	4	91
2	5	79
2	9	78
3	1	83
3	2	77
3	3	78
3	4	78
3	8	70
3	6	75
3	10	74

Multiple traits can be added as additional columns (Table 6).

Table 6: Data from an experiment in augmented RCBD design arranged in long-form (Multiple traits).

Block	Treatment	Trait1	Trait2
Block I	G 1	92	258
Block I	G 2	79	224
Block I	G 3	87	238
Block I	G 4	81	278
Block I	G 7	96	347
Block I	G 11	89	300
Block I	G 12	82	289
Block II	G 1	79	260
Block II	G 2	81	220
Block II	G 3	81	237

Block	Treatment	Trait1	Trait2
Block II	G 4	91	227
Block II	G 5	79	281
Block II	G 9	78	311
Block III	G 1	83	250
Block III	G 2	77	240
Block III	G 3	78	268
Block III	G 4	78	287
Block III	G 8	70	226
Block III	G 6	75	395
Block III	G 10	74	450

Data should preferably be balanced i.e. all the check genotypes should be present in all the blocks. If not, a warning is issued. The number of test genotypes can vary within a block. There should not be any missing values. Rows of genotypes with missing values for one or more traits should be removed.

Such a tabular data should be imported ([see section 7.8](#)) into R as a data frame object ([see section 4.3.5](#)). The columns with the block and treatment categorical data should be of the type factor ([see section 4.3.2](#)), while the column(s) with the trait data should be of the type integer or numeric ([see section 4.3.1](#)).

7 Data Analysis for a Single Trait

Analysis of data for a single trait can be performed by using **augmentedRCBD** function. It generates an object of class **augmentedRCBD**. Such an object can then be taken as input by the several functions to print the results to console (**print.augmentedRCBD**), generate descriptive statistics from adjusted means (**describe.augmentedRCBD**), plot frequency distribution (**freqdist.augmentedRCBD**) and computed genetic variability statistics (**gva.augmentedRCBD**). All these outputs can also be exported as a MS Word report using the **report.augmentedRCBD** function.

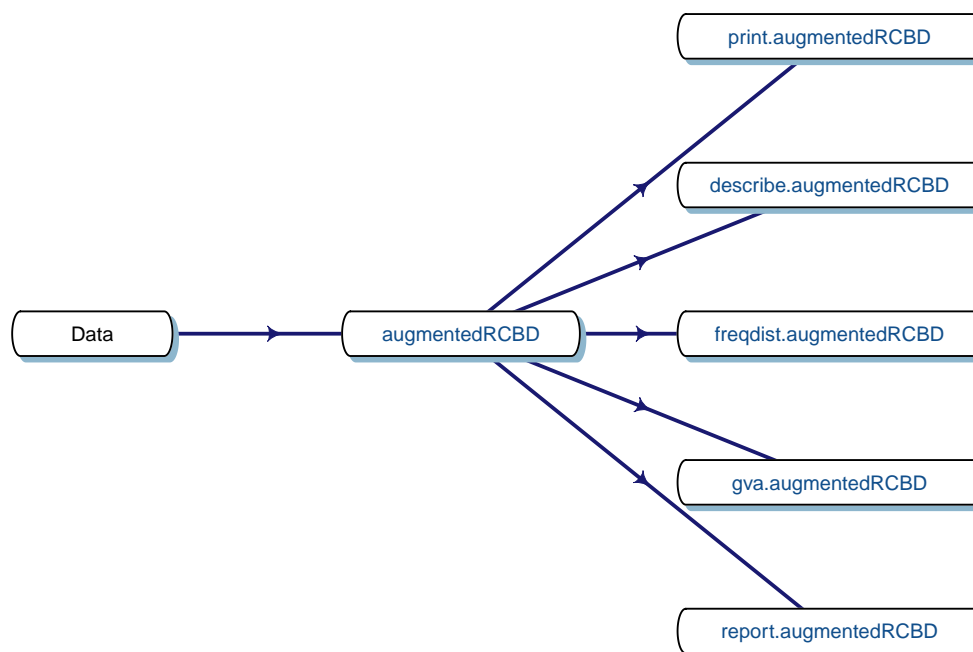


Fig. 4. Workflow for analysis of single traits with `augmentedRCBD`.

7.1 `augmentedRCBD()`

Consider the data in [Table 1](#). The data can be imported into R as [vectors](#) as follows.

```

blk <- c(1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3)
trt <- c(1, 2, 3, 4, 7, 11, 12, 1, 2, 3, 4, 5, 9, 1, 2, 3, 4, 8, 6, 10)
y1 <- c(92, 79, 87, 81, 96, 89, 82, 79, 81, 81, 91, 79, 78, 83, 77, 78, 78,
        70, 75, 74)
y2 <- c(258, 224, 238, 278, 347, 300, 289, 260, 220, 237, 227, 281, 311, 250,
        240, 268, 287, 226, 395, 450)
  
```

The `blk` and `trt` vectors with the block and treatment data need to be converted into factors as follows before analysis.

```

# Convert block and treatment to factors
blk <- as.factor(blk)
trt <- as.factor(trt)
  
```

With the data in appropriate format, the analysis can be performed as follows for the trait `y1` as follows.

```

out1 <- augmentedRCBD(blk, trt, y1, method.comp = "lsd",
                      alpha = 0.05, group = TRUE, console = TRUE)
  
```

Augmented Design Details

=====

```

Number of blocks      "3"
Number of treatments  "12"
Number of check treatments "4"
Number of test treatments "8"
Check treatments      "1, 2, 3, 4"

```

ANOVA, Treatment Adjusted

=====

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Block (ignoring Treatments)	2	360.1	180.04	6.675	0.0298 *
Treatment (eliminating Blocks)	11	285.1	25.92	0.961	0.5499
Treatment: Check	3	52.9	17.64	0.654	0.6092
Treatment: Test and Test vs. Check	8	232.2	29.02	1.076	0.4779
Residuals	6	161.8	26.97		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

ANOVA, Block Adjusted

=====

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Treatment (ignoring Blocks)	11	575.7	52.33	1.940	0.215
Treatment: Check	3	52.9	17.64	0.654	0.609
Treatment: Test	7	505.9	72.27	2.679	0.125
Treatment: Test vs. Check	1	16.9	16.87	0.626	0.459
Block (eliminating Treatments)	2	69.5	34.75	1.288	0.342
Residuals	6	161.8	26.97		

Treatment Means

=====

	Treatment	Block	Means	SE	r	Min	Max	Adjusted Means
1	1		84.66667	3.844188	3	79	92	84.66667
2	10	3	74.00000	NA	1	74	74	77.25000
3	11	1	89.00000	NA	1	89	89	86.50000
4	12	1	82.00000	NA	1	82	82	79.50000
5	2		79.00000	1.154701	3	77	81	79.00000
6	3		82.00000	2.645751	3	78	87	82.00000
7	4		83.33333	3.929942	3	78	91	83.33333
8	5	2	79.00000	NA	1	79	79	78.25000
9	6	3	75.00000	NA	1	75	75	78.25000
10	7	1	96.00000	NA	1	96	96	93.50000
11	8	3	70.00000	NA	1	70	70	73.25000
12	9	2	78.00000	NA	1	78	78	77.25000

Coefficient of Variation

=====

6.372367

Overall Adjusted Mean

=====

81.0625

Standard Errors

```
=====
                                Std. Error of Diff.  CD (5%)
Control Treatment Means                4.240458 10.37603
Two Test Treatments (Same Block)       7.344688 17.97180
Two Test Treatments (Different Blocks)  8.211611 20.09309
A Test Treatment and a Control Treatment 6.704752 16.40594
```

```
Treatment Groups
=====
```

```
Method : lsd
```

	Treatment	Adjusted Means	SE	df	lower.CL	upper.CL	Group
8	8	73.25000	5.609598	6	59.52381	86.97619	1
9	9	77.25000	5.609598	6	63.52381	90.97619	12
10	10	77.25000	5.609598	6	63.52381	90.97619	12
5	5	78.25000	5.609598	6	64.52381	91.97619	12
6	6	78.25000	5.609598	6	64.52381	91.97619	12
2	2	79.00000	2.998456	6	71.66304	86.33696	12
12	12	79.50000	5.609598	6	65.77381	93.22619	12
3	3	82.00000	2.998456	6	74.66304	89.33696	12
4	4	83.33333	2.998456	6	75.99637	90.67029	12
1	1	84.66667	2.998456	6	77.32971	92.00363	12
11	11	86.50000	5.609598	6	72.77381	100.22619	12
7	7	93.50000	5.609598	6	79.77381	107.22619	2

```
class(out1)
```

```
[1] "augmentedRCBD"
```

Similarly the analysis for the trait y2 can be computed as follows.

```
out2 <- augmentedRCBD(blk, trt, y2, method.comp = "lsd",
                      alpha = 0.05, group = TRUE, console = TRUE)
```

```
Augmented Design Details
```

```
=====
```

```
Number of blocks      "3"
Number of treatments  "12"
Number of check treatments "4"
Number of test treatments "8"
Check treatments       "1, 2, 3, 4"
```

```
ANOVA, Treatment Adjusted
```

```
=====
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Block (ignoring Treatments)	2	7019	3510	12.261	0.007597 **
Treatment (eliminating Blocks)	11	58965	5360	18.727	0.000920 ***
Treatment: Check	3	2150	717	2.504	0.156116
Treatment: Test and Test vs. Check	8	56815	7102	24.810	0.000473 ***
Residuals	6	1718	286		

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ANOVA, Block Adjusted

=====

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Treatment (ignoring Blocks)	11	64708	5883	20.550	0.000707 ***
Treatment: Check	3	2150	717	2.504	0.156116
Treatment: Test	7	34863	4980	17.399	0.001366 **
Treatment: Test vs. Check	1	27694	27694	96.749	6.36e-05 ***
Block (eliminating Treatments)	2	1277	639	2.231	0.188645
Residuals	6	1717	286		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Treatment Means

=====

	Treatment	Block	Means	SE	r	Min	Max	Adjusted Means
1		1	256.0000	3.055050	3	250	260	256.0000
2	10	3	450.0000	NA	1	450	450	437.6667
3	11	1	300.0000	NA	1	300	300	299.4167
4	12	1	289.0000	NA	1	289	289	288.4167
5	2		228.0000	6.110101	3	220	240	228.0000
6	3		247.6667	10.170764	3	237	268	247.6667
7	4		264.0000	18.681542	3	227	287	264.0000
8	5	2	281.0000	NA	1	281	281	293.9167
9	6	3	395.0000	NA	1	395	395	382.6667
10	7	1	347.0000	NA	1	347	347	346.4167
11	8	3	226.0000	NA	1	226	226	213.6667
12	9	2	311.0000	NA	1	311	311	323.9167

Coefficient of Variation

=====

6.057617

Overall Adjusted Mean

=====

298.4792

Standard Errors

=====

	Std. Error	of Diff.	CD (5%)
Control Treatment Means	13.81424	33.80224	
Two Test Treatments (Same Block)	23.92697	58.54719	
Two Test Treatments (Different Blocks)	26.75117	65.45775	
A Test Treatment and a Control Treatment	21.84224	53.44603	

Treatment Groups

=====

Method : lsd

	Treatment	Adjusted Means	SE	df	lower.CL	upper.CL	Group
8	8	213.6667	18.274527	6	168.9505	258.3828	12


```

2          2          228.0000  9.768146  6 204.0982 251.9018  1
3          3          247.6667  9.768146  6 223.7649 271.5685 123
1          1          256.0000  9.768146  6 232.0982 279.9018 1234
4          4          264.0000  9.768146  6 240.0982 287.9018  234
12         12          288.4167 18.274527  6 243.7005 333.1328  345
5          5          293.9167 18.274527  6 249.2005 338.6328  345
11         11          299.4167 18.274527  6 254.7005 344.1328   45
9          9          323.9167 18.274527  6 279.2005 368.6328   56
7          7          346.4167 18.274527  6 301.7005 391.1328   56
6          6          382.6667 18.274527  6 337.9505 427.3828   67
10         10          437.6667 18.274527  6 392.9505 482.3828    7

```

```
class(out2)
```

```
[1] "augmentedRCBD"
```

The data can also be imported as a [data frame](#) and then used for analysis. Consider the data frame [data](#) imported from [Table 1](#) according to the instructions in [section 4.8](#).

```
str(data)
```

```

'data.frame':  20 obs. of  4 variables:
 $ blk: Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 2 2 2 ...
 $ trt: Factor w/ 12 levels "1","2","3","4",...: 1 2 3 4 7 11 12 1 2 3 ...
 $ y1 : num  92 79 87 81 96 89 82 79 81 81 ...
 $ y2 : num  258 224 238 278 347 300 289 260 220 237 ...

```

```
# Convert block and treatment to factors
```

```
data$blk <- as.factor(data$blk)
```

```
data$trt <- as.factor(data$trt)
```

```
# Results for variable y1
```

```
out1 <- augmentedRCBD(data$blk, data$trt, data$y1, method.comp = "lsd",
                      alpha = 0.05, group = TRUE, console = TRUE)
```

```
Augmented Design Details
```

```
=====
```

```

Number of blocks          "3"
Number of treatments      "12"
Number of check treatments "4"
Number of test treatments "8"
Check treatments          "1, 2, 3, 4"

```

```
ANOVA, Treatment Adjusted
```

```
=====
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Block (ignoring Treatments)	2	360.1	180.04	6.675	0.0298 *
Treatment (eliminating Blocks)	11	285.1	25.92	0.961	0.5499
Treatment: Check	3	52.9	17.64	0.654	0.6092
Treatment: Test and Test vs. Check	8	232.2	29.02	1.076	0.4779
Residuals	6	161.8	26.97		

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ANOVA, Block Adjusted

=====

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Treatment (ignoring Blocks)	11	575.7	52.33	1.940	0.215
Treatment: Check	3	52.9	17.64	0.654	0.609
Treatment: Test	7	505.9	72.27	2.679	0.125
Treatment: Test vs. Check	1	16.9	16.87	0.626	0.459
Block (eliminating Treatments)	2	69.5	34.75	1.288	0.342
Residuals	6	161.8	26.97		

Treatment Means

=====

	Treatment	Block	Means	SE	r	Min	Max	Adjusted Means
1	1		84.66667	3.844188	3	79	92	84.66667
2	10	3	74.00000	NA	1	74	74	77.25000
3	11	1	89.00000	NA	1	89	89	86.50000
4	12	1	82.00000	NA	1	82	82	79.50000
5	2		79.00000	1.154701	3	77	81	79.00000
6	3		82.00000	2.645751	3	78	87	82.00000
7	4		83.33333	3.929942	3	78	91	83.33333
8	5	2	79.00000	NA	1	79	79	78.25000
9	6	3	75.00000	NA	1	75	75	78.25000
10	7	1	96.00000	NA	1	96	96	93.50000
11	8	3	70.00000	NA	1	70	70	73.25000
12	9	2	78.00000	NA	1	78	78	77.25000

Coefficient of Variation

=====

6.372367

Overall Adjusted Mean

=====

81.0625

Standard Errors

=====

	Std. Error	of Diff.	CD (5%)
Control Treatment Means	4.240458		10.37603
Two Test Treatments (Same Block)	7.344688		17.97180
Two Test Treatments (Different Blocks)	8.211611		20.09309
A Test Treatment and a Control Treatment	6.704752		16.40594

Treatment Groups

=====

Method : lsd

	Treatment	Adjusted Means	SE	df	lower.CL	upper.CL	Group
8	8	73.25000	5.609598	6	59.52381	86.97619	1
9	9	77.25000	5.609598	6	63.52381	90.97619	12
10	10	77.25000	5.609598	6	63.52381	90.97619	12
5	5	78.25000	5.609598	6	64.52381	91.97619	12

```

6          6          78.25000 5.609598 6 64.52381 91.97619 12
2          2          79.00000 2.998456 6 71.66304 86.33696 12
12         12          79.50000 5.609598 6 65.77381 93.22619 12
3          3          82.00000 2.998456 6 74.66304 89.33696 12
4          4          83.33333 2.998456 6 75.99637 90.67029 12
1          1          84.66667 2.998456 6 77.32971 92.00363 12
11         11          86.50000 5.609598 6 72.77381 100.22619 12
7          7          93.50000 5.609598 6 79.77381 107.22619 2

```

```
class(out1)
```

```
[1] "augmentedRCBD"
```

```
# Results for variable y2
```

```
out2 <- augmentedRCBD(data$blk, data$trt, data$y2, method.comp = "lsd",
                      alpha = 0.05, group = TRUE, console = TRUE)
```

```
Augmented Design Details
```

```
=====
```

```

Number of blocks          "3"
Number of treatments      "12"
Number of check treatments "4"
Number of test treatments "8"
Check treatments          "1, 2, 3, 4"

```

```
ANOVA, Treatment Adjusted
```

```
=====
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Block (ignoring Treatments)	2	7019	3510	12.261	0.007597 **
Treatment (eliminating Blocks)	11	58965	5360	18.727	0.000920 ***
Treatment: Check	3	2150	717	2.504	0.156116
Treatment: Test and Test vs. Check	8	56815	7102	24.810	0.000473 ***
Residuals	6	1718	286		

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ANOVA, Block Adjusted
```

```
=====
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Treatment (ignoring Blocks)	11	64708	5883	20.550	0.000707 ***
Treatment: Check	3	2150	717	2.504	0.156116
Treatment: Test	7	34863	4980	17.399	0.001366 **
Treatment: Test vs. Check	1	27694	27694	96.749	6.36e-05 ***
Block (eliminating Treatments)	2	1277	639	2.231	0.188645
Residuals	6	1717	286		

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Treatment Means
```

```
=====
```

	Treatment	Block	Means	SE	r	Min	Max	Adjusted Means
1	1	1	256.0000	3.055050	3	250	260	256.0000

2	10	3	450.0000	NA	1	450	450	437.6667
3	11	1	300.0000	NA	1	300	300	299.4167
4	12	1	289.0000	NA	1	289	289	288.4167
5	2		228.0000	6.110101	3	220	240	228.0000
6	3		247.6667	10.170764	3	237	268	247.6667
7	4		264.0000	18.681542	3	227	287	264.0000
8	5	2	281.0000	NA	1	281	281	293.9167
9	6	3	395.0000	NA	1	395	395	382.6667
10	7	1	347.0000	NA	1	347	347	346.4167
11	8	3	226.0000	NA	1	226	226	213.6667
12	9	2	311.0000	NA	1	311	311	323.9167

Coefficient of Variation

=====

6.057617

Overall Adjusted Mean

=====

298.4792

Standard Errors

=====

Std. Error of Diff. CD (5%)

Control Treatment Means	13.81424	33.80224
Two Test Treatments (Same Block)	23.92697	58.54719
Two Test Treatments (Different Blocks)	26.75117	65.45775
A Test Treatment and a Control Treatment	21.84224	53.44603

Treatment Groups

=====

Method : lsd

	Treatment	Adjusted Means	SE	df	lower.CL	upper.CL	Group
8	8	213.6667	18.274527	6	168.9505	258.3828	12
2	2	228.0000	9.768146	6	204.0982	251.9018	1
3	3	247.6667	9.768146	6	223.7649	271.5685	123
1	1	256.0000	9.768146	6	232.0982	279.9018	1234
4	4	264.0000	9.768146	6	240.0982	287.9018	234
12	12	288.4167	18.274527	6	243.7005	333.1328	345
5	5	293.9167	18.274527	6	249.2005	338.6328	345
11	11	299.4167	18.274527	6	254.7005	344.1328	45
9	9	323.9167	18.274527	6	279.2005	368.6328	56
7	7	346.4167	18.274527	6	301.7005	391.1328	56
6	6	382.6667	18.274527	6	337.9505	427.3828	67
10	10	437.6667	18.274527	6	392.9505	482.3828	7

class(out2)

[1] "augmentedRCBD"

Check genotypes are inferred by default on the basis of number of replications. However, if some test genotypes are also replicated, they may also be falsely detected as checks. To avoid this, the checks can

be specified by the `checks` argument.

```
# Results for variable y1 (checks specified)
out1 <- augmentedRCBD(data$blk, data$trt, data$y1, method.comp = "lsd",
  alpha = 0.05, group = TRUE, console = TRUE,
  checks = c("1", "2", "3", "4"))
```

Augmented Design Details

=====

```
Number of blocks          "3"
Number of treatments      "12"
Number of check treatments "4"
Number of test treatments "8"
Check treatments          "1, 2, 3, 4"
```

ANOVA, Treatment Adjusted

=====

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Block (ignoring Treatments)	2	360.1	180.04	6.675	0.0298 *
Treatment (eliminating Blocks)	11	285.1	25.92	0.961	0.5499
Treatment: Check	3	52.9	17.64	0.654	0.6092
Treatment: Test and Test vs. Check	8	232.2	29.02	1.076	0.4779
Residuals	6	161.8	26.97		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

ANOVA, Block Adjusted

=====

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Treatment (ignoring Blocks)	11	575.7	52.33	1.940	0.215
Treatment: Check	3	52.9	17.64	0.654	0.609
Treatment: Test	7	505.9	72.27	2.679	0.125
Treatment: Test vs. Check	1	16.9	16.87	0.626	0.459
Block (eliminating Treatments)	2	69.5	34.75	1.288	0.342
Residuals	6	161.8	26.97		

Treatment Means

=====

	Treatment	Block	Means	SE	r	Min	Max	Adjusted Means
1	1		84.66667	3.844188	3	79	92	84.66667
2	10	3	74.00000	NA	1	74	74	77.25000
3	11	1	89.00000	NA	1	89	89	86.50000
4	12	1	82.00000	NA	1	82	82	79.50000
5	2		79.00000	1.154701	3	77	81	79.00000
6	3		82.00000	2.645751	3	78	87	82.00000
7	4		83.33333	3.929942	3	78	91	83.33333
8	5	2	79.00000	NA	1	79	79	78.25000
9	6	3	75.00000	NA	1	75	75	78.25000
10	7	1	96.00000	NA	1	96	96	93.50000
11	8	3	70.00000	NA	1	70	70	73.25000
12	9	2	78.00000	NA	1	78	78	77.25000

Coefficient of Variation

=====

6.372367

Overall Adjusted Mean

=====

81.0625

Standard Errors

=====

Std. Error of Diff. CD (5%)

Control Treatment Means	4.240458	10.37603
Two Test Treatments (Same Block)	7.344688	17.97180
Two Test Treatments (Different Blocks)	8.211611	20.09309
A Test Treatment and a Control Treatment	6.704752	16.40594

Treatment Groups

=====

Method : lsd

	Treatment	Adjusted Means	SE	df	lower.CL	upper.CL	Group
8	8	73.25000	5.609598	6	59.52381	86.97619	1
9	9	77.25000	5.609598	6	63.52381	90.97619	12
10	10	77.25000	5.609598	6	63.52381	90.97619	12
5	5	78.25000	5.609598	6	64.52381	91.97619	12
6	6	78.25000	5.609598	6	64.52381	91.97619	12
2	2	79.00000	2.998456	6	71.66304	86.33696	12
12	12	79.50000	5.609598	6	65.77381	93.22619	12
3	3	82.00000	2.998456	6	74.66304	89.33696	12
4	4	83.33333	2.998456	6	75.99637	90.67029	12
1	1	84.66667	2.998456	6	77.32971	92.00363	12
11	11	86.50000	5.609598	6	72.77381	100.22619	12
7	7	93.50000	5.609598	6	79.77381	107.22619	2

Results for variable y2 (checks specified)

```
out2 <- augmentedRCBD(data$blk, data$trt, data$y2, method.comp = "lsd",
  alpha = 0.05, group = TRUE, console = TRUE,
  checks = c("1", "2", "3", "4"))
```

Augmented Design Details

=====

Number of blocks	"3"
Number of treatments	"12"
Number of check treatments	"4"
Number of test treatments	"8"
Check treatments	"1, 2, 3, 4"

ANOVA, Treatment Adjusted

=====

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
Block (ignoring Treatments)	2	7019	3510	12.261	0.007597	**
Treatment (eliminating Blocks)	11	58965	5360	18.727	0.000920	***
Treatment: Check	3	2150	717	2.504	0.156116	
Treatment: Test and Test vs. Check	8	56815	7102	24.810	0.000473	***
Residuals	6	1718	286			

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

ANOVA, Block Adjusted

=====

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
Treatment (ignoring Blocks)	11	64708	5883	20.550	0.000707	***
Treatment: Check	3	2150	717	2.504	0.156116	
Treatment: Test	7	34863	4980	17.399	0.001366	**
Treatment: Test vs. Check	1	27694	27694	96.749	6.36e-05	***
Block (eliminating Treatments)	2	1277	639	2.231	0.188645	
Residuals	6	1717	286			

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Treatment Means

=====

	Treatment	Block	Means	SE	r	Min	Max	Adjusted Means
1	1		256.0000	3.055050	3	250	260	256.0000
2	10	3	450.0000	NA	1	450	450	437.6667
3	11	1	300.0000	NA	1	300	300	299.4167
4	12	1	289.0000	NA	1	289	289	288.4167
5	2		228.0000	6.110101	3	220	240	228.0000
6	3		247.6667	10.170764	3	237	268	247.6667
7	4		264.0000	18.681542	3	227	287	264.0000
8	5	2	281.0000	NA	1	281	281	293.9167
9	6	3	395.0000	NA	1	395	395	382.6667
10	7	1	347.0000	NA	1	347	347	346.4167
11	8	3	226.0000	NA	1	226	226	213.6667
12	9	2	311.0000	NA	1	311	311	323.9167

Coefficient of Variation

=====

6.057617

Overall Adjusted Mean

=====

298.4792

Standard Errors

=====

	Std. Error of Diff.	CD (5%)
Control Treatment Means	13.81424	33.80224
Two Test Treatments (Same Block)	23.92697	58.54719
Two Test Treatments (Different Blocks)	26.75117	65.45775
A Test Treatment and a Control Treatment	21.84224	53.44603

Treatment Groups

=====

Method : lsd

	Treatment	Adjusted Means	SE	df	lower.CL	upper.CL	Group
8	8	213.6667	18.274527	6	168.9505	258.3828	12
2	2	228.0000	9.768146	6	204.0982	251.9018	1
3	3	247.6667	9.768146	6	223.7649	271.5685	123
1	1	256.0000	9.768146	6	232.0982	279.9018	1234
4	4	264.0000	9.768146	6	240.0982	287.9018	234
12	12	288.4167	18.274527	6	243.7005	333.1328	345
5	5	293.9167	18.274527	6	249.2005	338.6328	345
11	11	299.4167	18.274527	6	254.7005	344.1328	45
9	9	323.9167	18.274527	6	279.2005	368.6328	56
7	7	346.4167	18.274527	6	301.7005	391.1328	56
6	6	382.6667	18.274527	6	337.9505	427.3828	67
10	10	437.6667	18.274527	6	392.9505	482.3828	7

In case the large number of treatments or genotypes, it is advisable to avoid treatment comparisons with the `group = FALSE` argument as it will be memory and processor intensive. Further it is advised to simplify output with `simplify = TRUE` in order to reduce output object size.

If `truncate.means = TRUE`, then any negative adjusted means will be truncated to zero with a warning.

7.2 `print.augmentedRCBD()`

The results of analysis in an object of class `augmentedRCBD` can be printed to the console as follows.

```
# Print results for variable y1
print(out1)
```

Augmented Design Details

=====

```
Number of blocks      "3"
Number of treatments  "12"
Number of check treatments "4"
Number of test treatments "8"
Check treatments      "1, 2, 3, 4"
```

ANOVA, Treatment Adjusted

=====

	Df	Sum Sq	Mean Sq	F	value	Pr(>F)
Block (ignoring Treatments)	2	360.1	180.04	6.675	0.0298	*
Treatment (eliminating Blocks)	11	285.1	25.92	0.961	0.5499	
Treatment: Check	3	52.9	17.64	0.654	0.6092	
Treatment: Test and Test vs. Check	8	232.2	29.02	1.076	0.4779	
Residuals	6	161.8	26.97			

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

ANOVA, Block Adjusted

=====

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Treatment (ignoring Blocks)	11	575.7	52.33	1.940	0.215
Treatment: Check	3	52.9	17.64	0.654	0.609
Treatment: Test	7	505.9	72.27	2.679	0.125
Treatment: Test vs. Check	1	16.9	16.87	0.626	0.459
Block (eliminating Treatments)	2	69.5	34.75	1.288	0.342
Residuals	6	161.8	26.97		

Treatment Means

=====

	Treatment	Block	Means	SE	r	Min	Max	Adjusted Means
1	1		84.66667	3.844188	3	79	92	84.66667
2	10	3	74.00000	NA	1	74	74	77.25000
3	11	1	89.00000	NA	1	89	89	86.50000
4	12	1	82.00000	NA	1	82	82	79.50000
5	2		79.00000	1.154701	3	77	81	79.00000
6	3		82.00000	2.645751	3	78	87	82.00000
7	4		83.33333	3.929942	3	78	91	83.33333
8	5	2	79.00000	NA	1	79	79	78.25000
9	6	3	75.00000	NA	1	75	75	78.25000
10	7	1	96.00000	NA	1	96	96	93.50000
11	8	3	70.00000	NA	1	70	70	73.25000
12	9	2	78.00000	NA	1	78	78	77.25000

Coefficient of Variation

=====

6.372367

Overall Adjusted Mean

=====

81.0625

Standard Errors

=====

	Std. Error of Diff.	CD (5%)
Control Treatment Means	4.240458	10.37603
Two Test Treatments (Same Block)	7.344688	17.97180
Two Test Treatments (Different Blocks)	8.211611	20.09309
A Test Treatment and a Control Treatment	6.704752	16.40594

Treatment Groups

=====

Method : lsd

	Treatment	Adjusted Means	SE	df	lower.CL	upper.CL	Group
8	8	73.25000	5.609598	6	59.52381	86.97619	1
9	9	77.25000	5.609598	6	63.52381	90.97619	12
10	10	77.25000	5.609598	6	63.52381	90.97619	12
5	5	78.25000	5.609598	6	64.52381	91.97619	12
6	6	78.25000	5.609598	6	64.52381	91.97619	12

```

2          2          79.00000 2.998456 6 71.66304 86.33696 12
12         12         79.50000 5.609598 6 65.77381 93.22619 12
3          3          82.00000 2.998456 6 74.66304 89.33696 12
4          4          83.33333 2.998456 6 75.99637 90.67029 12
1          1          84.66667 2.998456 6 77.32971 92.00363 12
11         11         86.50000 5.609598 6 72.77381 100.22619 12
7          7          93.50000 5.609598 6 79.77381 107.22619 2

```

```

# Print results for variable y2
print(out2)

```

Augmented Design Details

```
=====
```

```

Number of blocks          "3"
Number of treatments      "12"
Number of check treatments "4"
Number of test treatments "8"
Check treatments          "1, 2, 3, 4"

```

ANOVA, Treatment Adjusted

```
=====
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
Block (ignoring Treatments)	2	7019	3510	12.261	0.007597	**
Treatment (eliminating Blocks)	11	58965	5360	18.727	0.000920	***
Treatment: Check	3	2150	717	2.504	0.156116	
Treatment: Test and Test vs. Check	8	56815	7102	24.810	0.000473	***
Residuals	6	1718	286			

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ANOVA, Block Adjusted

```
=====
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
Treatment (ignoring Blocks)	11	64708	5883	20.550	0.000707	***
Treatment: Check	3	2150	717	2.504	0.156116	
Treatment: Test	7	34863	4980	17.399	0.001366	**
Treatment: Test vs. Check	1	27694	27694	96.749	6.36e-05	***
Block (eliminating Treatments)	2	1277	639	2.231	0.188645	
Residuals	6	1717	286			

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Treatment Means

```
=====
```

	Treatment	Block	Means	SE	r	Min	Max	Adjusted Means
1	1		256.0000	3.055050	3	250	260	256.0000
2	10	3	450.0000	NA	1	450	450	437.6667
3	11	1	300.0000	NA	1	300	300	299.4167
4	12	1	289.0000	NA	1	289	289	288.4167
5	2		228.0000	6.110101	3	220	240	228.0000
6	3		247.6667	10.170764	3	237	268	247.6667
7	4		264.0000	18.681542	3	227	287	264.0000

8	5	2	281.0000	NA	1	281	281	293.9167
9	6	3	395.0000	NA	1	395	395	382.6667
10	7	1	347.0000	NA	1	347	347	346.4167
11	8	3	226.0000	NA	1	226	226	213.6667
12	9	2	311.0000	NA	1	311	311	323.9167

Coefficient of Variation

=====

6.057617

Overall Adjusted Mean

=====

298.4792

Standard Errors

=====

	Std. Error of Diff.	CD (5%)
Control Treatment Means	13.81424	33.80224
Two Test Treatments (Same Block)	23.92697	58.54719
Two Test Treatments (Different Blocks)	26.75117	65.45775
A Test Treatment and a Control Treatment	21.84224	53.44603

Treatment Groups

=====

Method : lsd

	Treatment	Adjusted Means	SE	df	lower.CL	upper.CL	Group
8	8	213.6667	18.274527	6	168.9505	258.3828	12
2	2	228.0000	9.768146	6	204.0982	251.9018	1
3	3	247.6667	9.768146	6	223.7649	271.5685	123
1	1	256.0000	9.768146	6	232.0982	279.9018	1234
4	4	264.0000	9.768146	6	240.0982	287.9018	234
12	12	288.4167	18.274527	6	243.7005	333.1328	345
5	5	293.9167	18.274527	6	249.2005	338.6328	345
11	11	299.4167	18.274527	6	254.7005	344.1328	45
9	9	323.9167	18.274527	6	279.2005	368.6328	56
7	7	346.4167	18.274527	6	301.7005	391.1328	56
6	6	382.6667	18.274527	6	337.9505	427.3828	67
10	10	437.6667	18.274527	6	392.9505	482.3828	7

7.3 `describe.augmentedRCBD()`

The descriptive statistics such as count, mean, standard error, minimum, maximum, skewness (with p-value from D'Agostino test of skewness (D'Agostino (1970))) and kurtosis (with p-value from Anscombe-Glynn test of kurtosis (Anscombe and Glynn (1983))) for the adjusted means from the results in an object of class `augmentedRCBD` can be computed as follows.

```
# Descriptive statistics for variable y1
describe.augmentedRCBD(out1)
```

```
$Count
[1] 12
```

```

$Mean
[1] 81.0625

$Std.Error
[1] 1.547002

$Std.Deviation
[1] 5.358973

$Min
[1] 73.25

$Max
[1] 93.5

$`Skewness(statistic)`
      skew      z
0.9250344 1.6745760

$`Skewness(p.value)`
[1] 0.09401746

$`Kurtosis(statistic)`
      kurt      z
3.522807 1.282305

$`Kurtosis(p.value)`
[1] 0.1997357

# Descriptive statistics for variable y2
describe.augmentedRCBD(out2)

$Count
[1] 12

$Mean
[1] 298.4792

$Std.Error
[1] 18.92257

$Std.Deviation
[1] 65.5497

$Min
[1] 213.6667

$Max
[1] 437.6667

$`Skewness(statistic)`
      skew      z

```

```
0.7449405 1.3680211
```

```
$`Skewness(p.value)`  
[1] 0.1713055
```

```
$`Kurtosis(statistic)`  
      kurt      z  
2.787997 0.536812
```

```
$`Kurtosis(p.value)`  
[1] 0.5913975
```

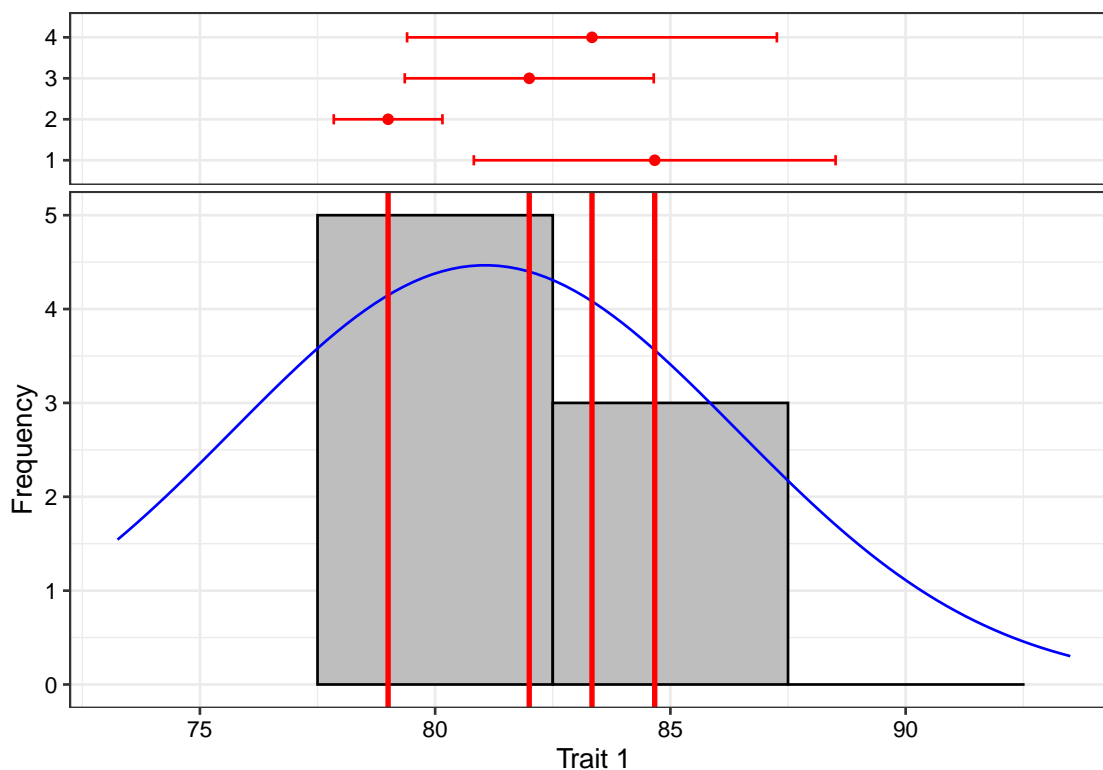
7.4 `freqdist.augmentedRCBD()`

The frequency distribution of the adjusted means from the results in an object of class `augmentedRCBD` can be plotted as follows.

```
# Frequency distribution for variable y1  
freq1 <- freqdist.augmentedRCBD(out1, xlab = "Trait 1")
```

Warning: Removed 2 rows containing missing values (geom_bar).

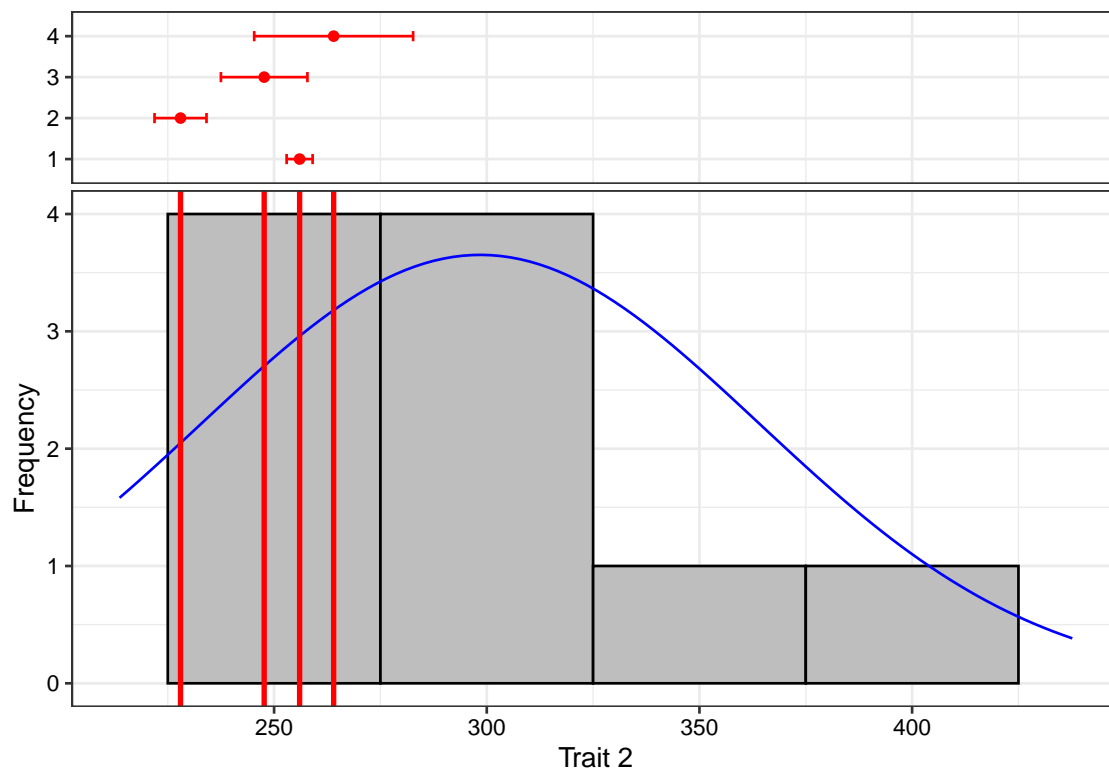
```
plot(freq1)
```



```
# Frequency distribution for variable y2  
freq2 <- freqdist.augmentedRCBD(out2, xlab = "Trait 2")
```

Warning: Removed 2 rows containing missing values (geom_bar).

```
plot(freq2)
```



The colours for the check values may be specified using the argument `check.col`.

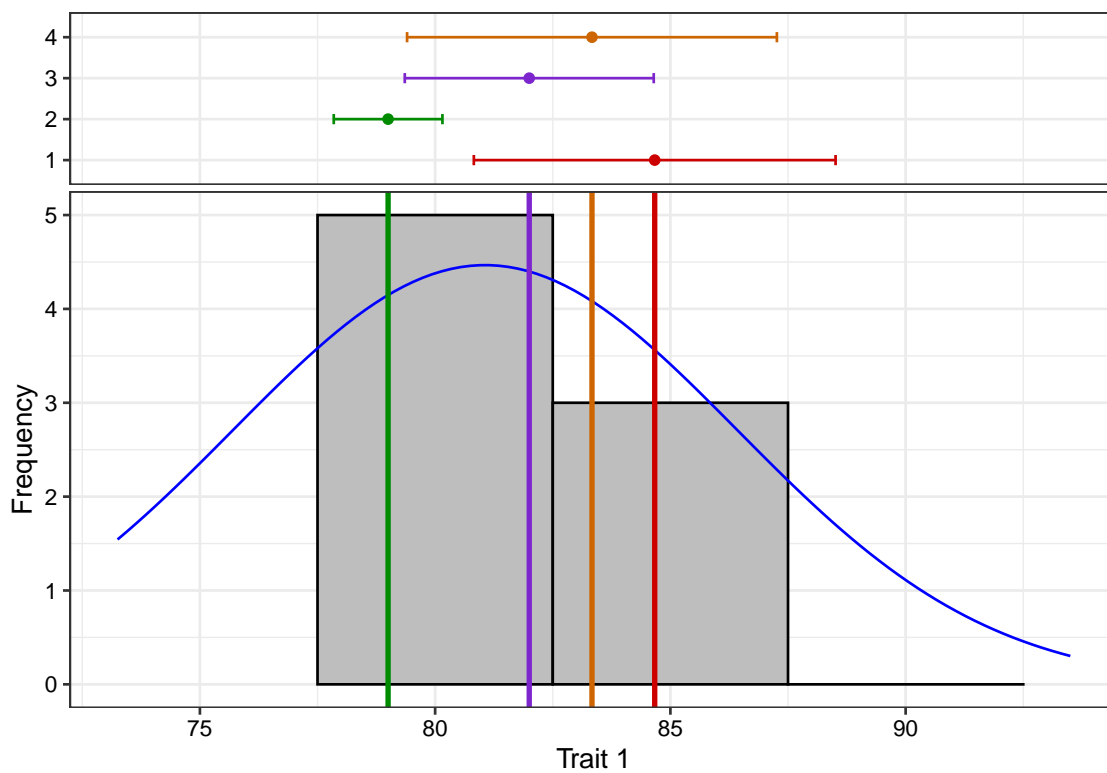
```
colset <- c("red3", "green4", "purple3", "darkorange3")
```

```
# Frequency distribution for variable y1
```

```
freq1 <- freqdist.augmentedRCBD(out1, xlab = "Trait 1", check.col = colset)
```

Warning: Removed 2 rows containing missing values (geom_bar).

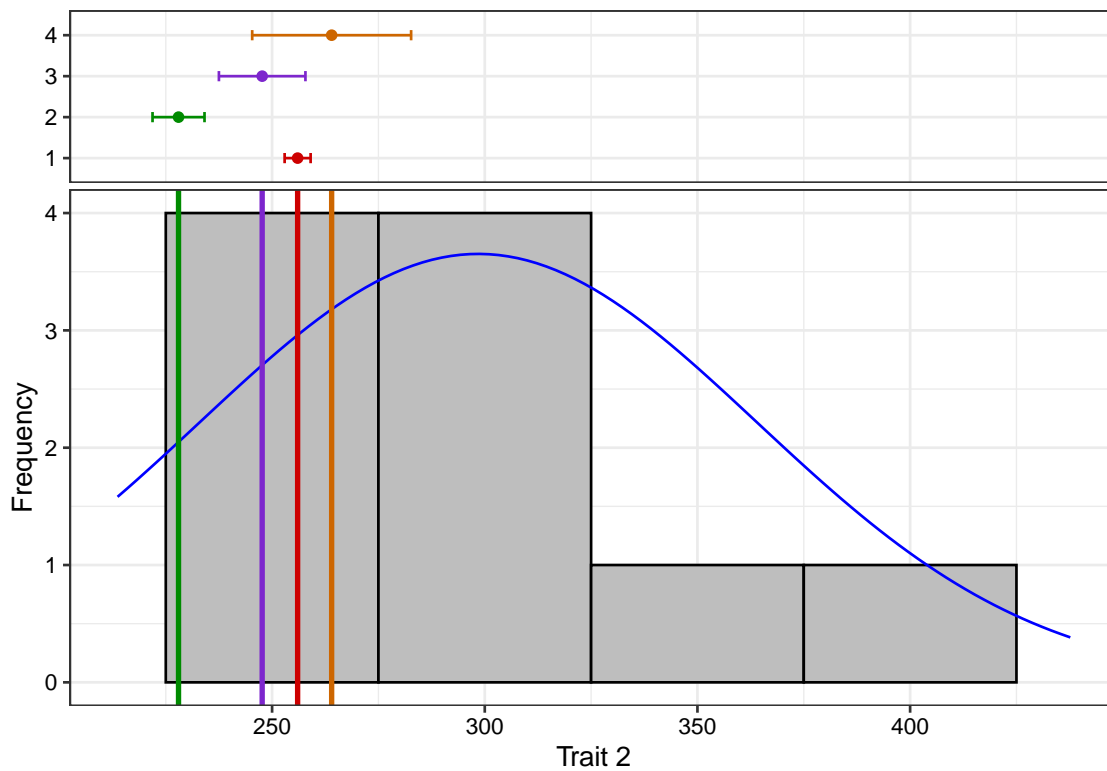
```
plot(freq1)
```



```
# Frequency distribution for variable y2
freq2 <- freqdist.augmentedRCBD(out2, xlab = "Trait 2", check.col = colset)
```

Warning: Removed 2 rows containing missing values (geom_bar).

```
plot(freq2)
```

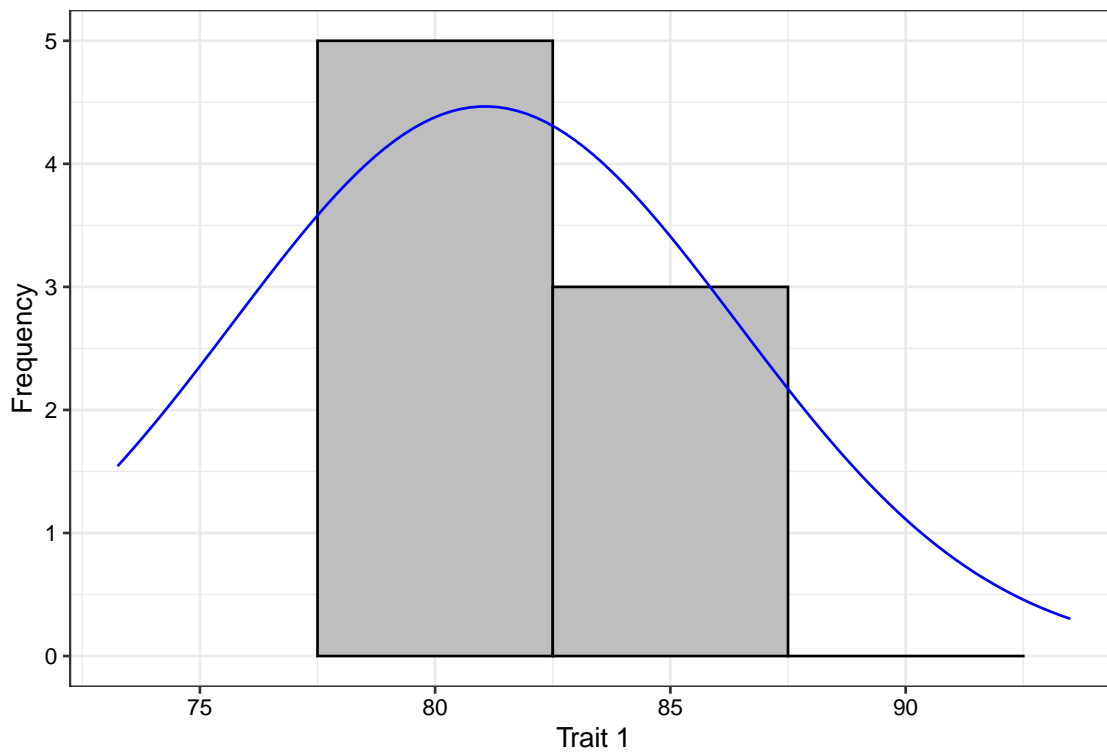


The default the check highlighting can be avoided using the argument `highlight.check = FALSE`.

```
# Frequency distribution for variable y1
freq1 <- freqdist.augmentedRCBD(out1, xlab = "Trait 1",
                                highlight.check = FALSE)
```

Warning: Removed 2 rows containing missing values (geom_bar).

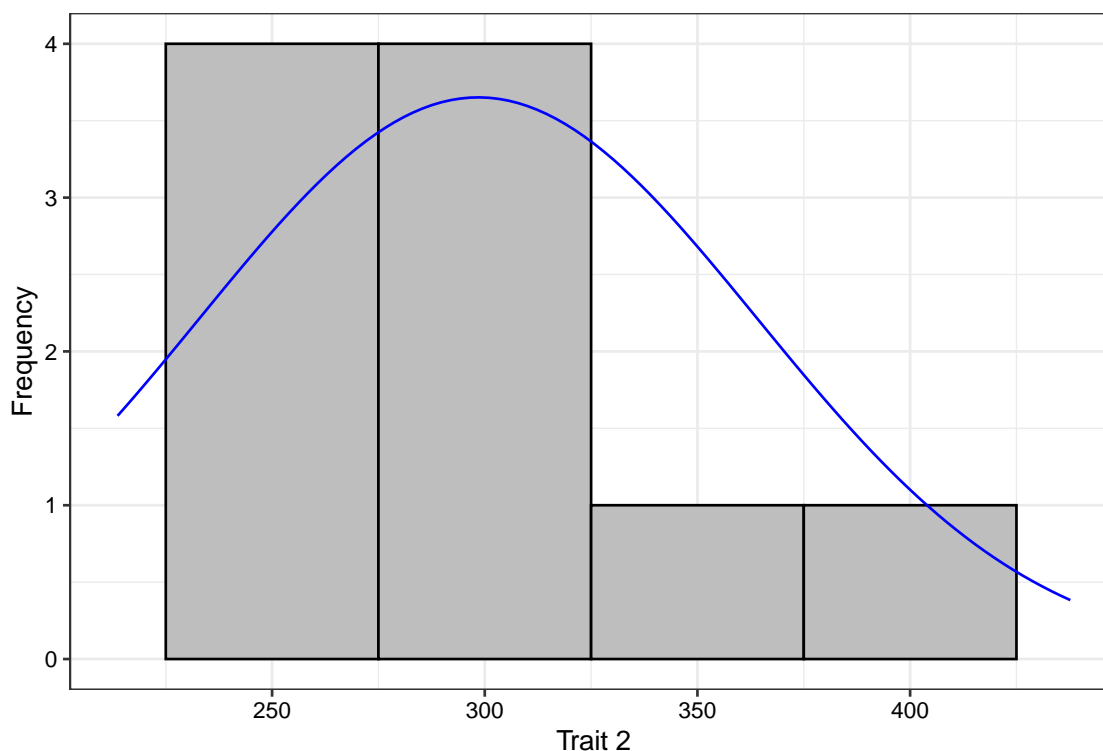
```
plot(freq1)
```

```
# Frequency distribution for variable y2
freq2 <- freqdist.augmentedRCBD(out2, xlab = "Trait 2",
                                highlight.check = FALSE)
```

Warning: Removed 2 rows containing missing values (geom_bar).

```
plot(freq2)
```



7.5 `gva.augmentedRCBD()`

The genetic variability statistics such as mean, phenotypic, genotypic and environmental variation (Federer and Searle (1976)), phenotypic, genotypic and environmental coefficient of variation (Burton (1951), Burton (1952)), category of phenotypic and genotypic coefficient of variation according to Sivasubramaniam and Madhavamenon (1973), broad-sense heritability (H^2) (Lush (1940)), H^2 category according to Robinson (1966), Genetic advance (GA), genetic advance as per cent of mean (GAM) and GAM category according to Johnson et al. (1955) are computed from an object of class `augmentedRCBD` as follows. Genetic variability analysis needs to be performed only if the sum of squares of “Treatment: Test” are significant.

```
# Genetic variability statistics for variable y1
gva.augmentedRCBD(out1)
```

```
Warning in gva.augmentedRCBD(out1): P-value for "Treatment: Test" is > 0.05. Genetic variability anal
trait.
```

```
$Mean
[1] 81.0625
```

```
$PV
[1] 72.26786
```

```
$GV
[1] 45.29563
```

```
$EV
[1] 26.97222

$GCV
[1] 8.302487

$`GCV category`
[1] "Low"

$PCV
[1] 10.48703

$`PCV category`
[1] "Medium"

$ECV
[1] 6.406759

$hBS
[1] 62.67743

$hBS category`
[1] "High"

$GA
[1] 10.99216

$GAM
[1] 13.5601

$`GAM category`
[1] "Medium"

# Genetic variability statistics for variable y2
gva.augmentedRCBD(out2)

$Mean
[1] 298.4792

$PV
[1] 4980.411

$GV
[1] 4694.161

$EV
[1] 286.25

$GCV
[1] 22.95435

$`GCV category`
[1] "High"
```

```
$PCV
[1] 23.64387

$`PCV category`
[1] "High"

$ECV
[1] 5.668377

$hBS
[1] 94.25248

$hBS category`
[1] "High"

$GA
[1] 137.2223

$GAM
[1] 45.97382

$`GAM category`
[1] "High"
```

Negative estimates of variance components if computed are not abnormal. For information on how to deal with these, refer Robinson et al. ([1955](#)) and Dudley and Moll ([1969](#)).

7.5 `report.augmentedRCBD()`

The results generated by the analysis can be exported to a MS Word file as follows.

```
# MS word report for variable y1
report.augmentedRCBD(aug = out1,
                     target = file.path(tempdir(), "augmentedRCBD output - y1.docx"))

# MS word report for variable y2
report.augmentedRCBD(aug = out1,
                     target = file.path(tempdir(), "augmentedRCBD output - y2.docx"))
```

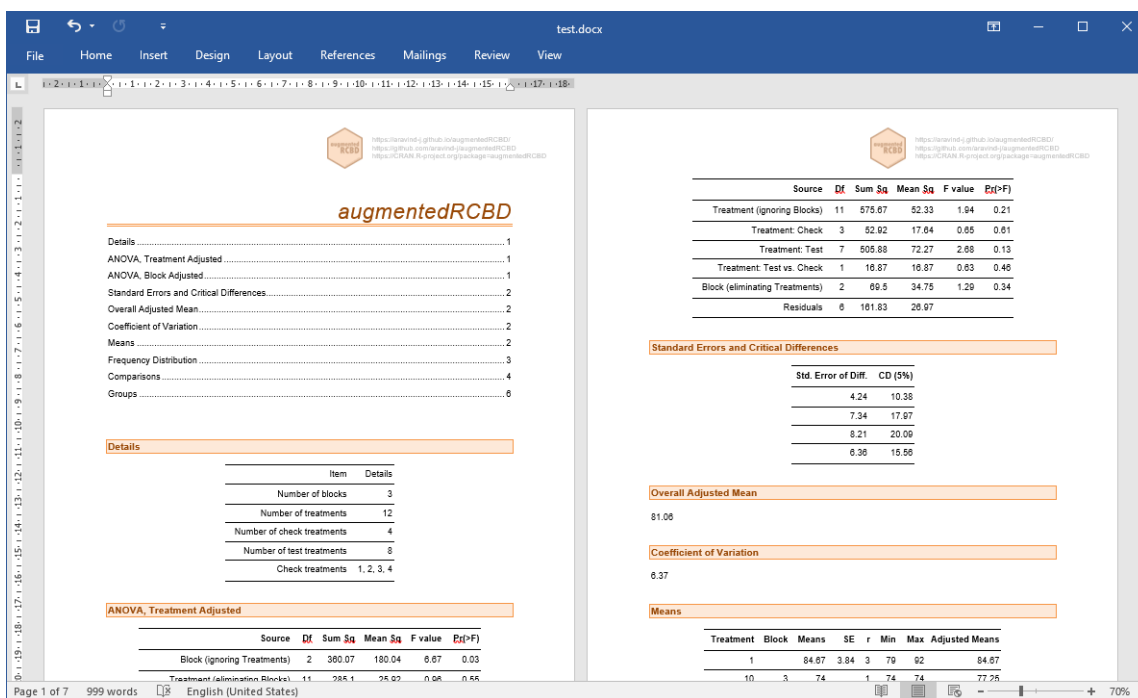


Fig. 6: MS Word report generated with `report.augmentedRCBD` function.

8 Data Analysis for a Multiple Traits

Analysis of data for a multiple traits simultaneously can be performed by using `augmentedRCBD.bulk` function. It generates an object of class `augmentedRCBD.bulk`. Such an object can then be taken as input by `print.augmentedRCBD.bulk` to print the results to console. The results can also be exported as a MS Word report using the `report.augmentedRCBD.bulk` function.

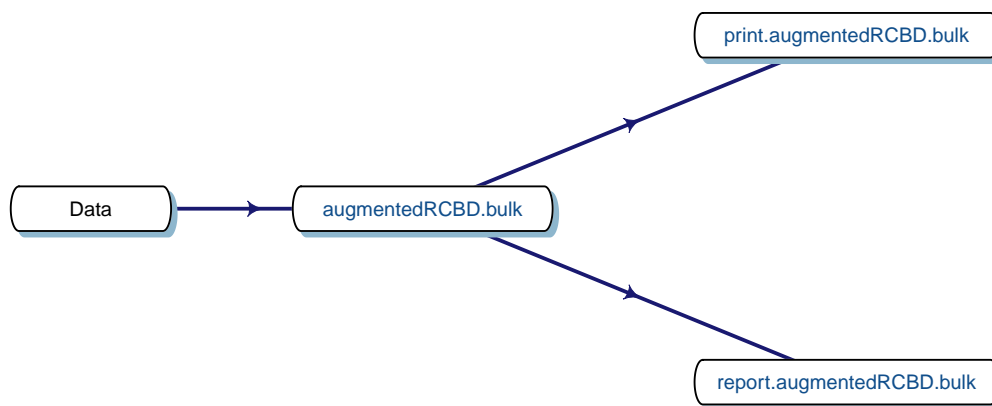


Fig. 7. Workflow for analysis of multiple traits with `augmentedRCBD`.

8.1 `augmentedRCBD.bulk()`

Consider the data frame `data` imported from [Table 1](#) according to the instructions in [section 4.8](#).

```
str(data)

'data.frame':  20 obs. of  4 variables:
 $ blk: Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 2 2 2 ...
 $ trt: Factor w/ 12 levels "1","2","3","4",...: 1 2 3 4 7 11 12 1 2 3 ...
 $ y1 : num  92 79 87 81 96 89 82 79 81 81 ...
 $ y2 : num  258 224 238 278 347 300 289 260 220 237 ...

# Convert block and treatment to factors
data$blk <- as.factor(data$blk)
data$trt <- as.factor(data$trt)
```

Rather than performing the analysis individually for each variable/trait separately using `augmentedRCBD`, the analysis can be performed simultaneously for both the traits using `augmentedRCBD.bulk` function. It is a wrapper around the `augmentedRCBD` core function and its associated helper functions.

However in this case treatment comparisons/grouping by least significant difference or Tukey's honest significant difference method is not computed. Also the output object size is reduced using the `simplify = TRUE` argument in the `augmentedRCBD` function.

The logical arguments `describe`, `freqdist` and `gva` can be used to specify whether to generate the descriptive statistics, frequency distribution plots and genetic variability statistics respectively. If `gva =`

TRUE, then plots to compare phenotypic and genotypic coefficient of variation, broad sense heritability and genetic advance over mean between traits are also generated.

```
bout <- augmentedRCBD.bulk(data = data, block = "blk",
  treatment = "trt", traits = c("y1", "y2"),
  checks = NULL, alpha = 0.05, describe = TRUE,
  freqdist = TRUE, gva = TRUE,
  check.col = c("brown", "darkcyan",
    "forestgreen", "purple"),
  console = TRUE)
```

ANOVA for y1 computed (1/2)

ANOVA for y2 computed (2/2)

Augmented Design Details

=====

Number of blocks	"3"
Number of treatments	"12"
Number of check treatments	"4"
Number of test treatments	"8"
Check treatments	"1, 2, 3, 4"
Number of traits	"2"
Traits	"y1, y2"

ANOVA, Treatment Adjusted

=====

	Source	Df		y1	y2
1	Block (ignoring Treatments)	2	180.04 *	3509.67	**
2	Treatment (eliminating Blocks)	11	25.92 ns	5360.49	**
3	Treatment: Check	3	17.64 ns	716.75	ns
4	Treatment: Test and Test vs. Check	8	29.02 ns	7101.89	**
5	Residuals	6	26.97	286.25	

ANOVA, Block Adjusted

=====

	Source	Df		y1	y2
1	Treatment (ignoring Blocks)	11	52.33 ns	5882.5	**
2	Treatment: Check	3	17.64 ns	716.75	ns
3	Treatment: Test vs. Check	1	16.87 ns	27694.41	**
4	Treatment: Test	7	72.27 ns	4980.41	**
5	Block (eliminating Treatments)	2	34.75 ns	638.58	ns
6	Residuals	6	26.97	286.25	

Coefficient of Variation

=====

	Trait	CV
1	y1	6.37
2	y2	6.06

Overall Adjusted Mean

=====

	Trait	Overall.adjusted.mean
1	y1	81.06
2	y2	298.48

Standard Errors

=====

	Comparison	y1	y2
1	A Test Treatment and a Control Treatment	6.7	21.84
2	Control Treatment Means	4.24	13.81
3	Two Test Treatments (Different Blocks)	8.21	26.75
4	Two Test Treatments (Same Block)	7.34	23.93

Critical Difference

=====

	Comparison	y1	y2
1	A Test Treatment and a Control Treatment	16.41	53.45
2	Control Treatment Means	10.38	33.8
3	Two Test Treatments (Different Blocks)	20.09	65.46
4	Two Test Treatments (Same Block)	17.97	58.55

Descriptive Statistics

=====

	Trait	Count	Mean	Std.Error	Std.Deviation	Min	Max	Skewness	Kurtosis
skew...1	y1	12	81.06	1.55	5.36	73.25	93.5	0.93 ns	3.52 ns
skew...2	y2	12	298.48	18.92	65.55	213.67	437.67	0.74 ns	2.79 ns

Genetic Variability Analysis

=====

	Trait	Mean	PV	GV	EV	GCV	GCV.category	PCV	PCV.category	ECV	hBS	hBS.category
1	y1	81.06	72.27	45.3	26.97	8.3	Low	10.49	Medium	6.41	62.68	High
2	y2	298.48	4980.41	4694.16	286.25	22.95	High	23.64	High	5.67	94.25	High

Warning Messages

=====

y1
Removed 2 rows containing missing values (geom_bar).
y2
Removed 2 rows containing missing values (geom_bar).

Treatment Means

=====

	Treatment	y1	y2
1	1	84.67	256
2	10	77.25	437.67
3	11	86.5	299.42
4	12	79.5	288.42

5	2	79	228
6	3	82	247.67
7	4	83.33	264
8	5	78.25	293.92
9	6	78.25	382.67
10	7	93.5	346.42
11	8	73.25	213.67
12	9	77.25	323.92

8.2 `print.augmentedRCBD.bulk()`

The results of analysis in an object of class `augmentedRCBD.bulk` can be printed to the console as follows.

```
# Print results
print(bout)
```

Augmented Design Details

=====

Number of blocks	"3"
Number of treatments	"12"
Number of check treatments	"4"
Number of test treatments	"8"
Check treatments	"1, 2, 3, 4"
Number of traits	"2"
Traits	"y1, y2"

ANOVA, Treatment Adjusted

=====

	Source	Df		y1	y2
1	Block (ignoring Treatments)	2	180.04 *	3509.67	**
2	Treatment (eliminating Blocks)	11	25.92 ns	5360.49	**
3	Treatment: Check	3	17.64 ns	716.75	ns
4	Treatment: Test and Test vs. Check	8	29.02 ns	7101.89	**
5	Residuals	6	26.97	286.25	

ANOVA, Block Adjusted

=====

	Source	Df		y1	y2
1	Treatment (ignoring Blocks)	11	52.33 ns	5882.5	**
2	Treatment: Check	3	17.64 ns	716.75	ns
3	Treatment: Test vs. Check	1	16.87 ns	27694.41	**
4	Treatment: Test	7	72.27 ns	4980.41	**
5	Block (eliminating Treatments)	2	34.75 ns	638.58	ns
6	Residuals	6	26.97	286.25	

Coefficient of Variation

=====

	Trait	CV
1	y1	6.37
2	y2	6.06

Overall Adjusted Mean

=====

	Trait	Overall.adjusted.mean
1	y1	81.06
2	y2	298.48

Standard Errors

=====

	Comparison	y1	y2
1	A Test Treatment and a Control Treatment	6.7	21.84
2	Control Treatment Means	4.24	13.81
3	Two Test Treatments (Different Blocks)	8.21	26.75
4	Two Test Treatments (Same Block)	7.34	23.93

Critical Difference

=====

	Comparison	y1	y2
1	A Test Treatment and a Control Treatment	16.41	53.45
2	Control Treatment Means	10.38	33.8
3	Two Test Treatments (Different Blocks)	20.09	65.46
4	Two Test Treatments (Same Block)	17.97	58.55

Descriptive Statistics

=====

	Trait	Count	Mean	Std.Error	Std.Deviation	Min	Max	Skewness	Kurtosis
skew...1	y1	12	81.06	1.55	5.36	73.25	93.5	0.93 ns	3.52 ns
skew...2	y2	12	298.48	18.92	65.55	213.67	437.67	0.74 ns	2.79 ns

Genetic Variability Analysis

=====

	Trait	Mean	PV	GV	EV	GCV	GCV.category	PCV	PCV.category	ECV	hBS	hBS.category
1	y1	81.06	72.27	45.3	26.97	8.3	Low	10.49	Medium	6.41	62.68	High
2	y2	298.48	4980.41	4694.16	286.25	22.95	High	23.64	High	5.67	94.25	High

Warning Messages

=====

y1
Removed 2 rows containing missing values (geom_bar).
y2
Removed 2 rows containing missing values (geom_bar).

Treatment Means

=====

	Treatment	y1	y2
1	1	84.67	256
2	10	77.25	437.67

```

3      11  86.5 299.42
4      12  79.5 288.42
5       2   79   228
6       3   82  247.67
7       4  83.33   264
8       5  78.25 293.92
9       6  78.25 382.67
10      7  93.5 346.42
11      8  73.25 213.67
12      9  77.25 323.92

```

8.3 `report.augmentedRCBD.bulk()`

The results generated by the analysis can be exported to a MS Word file as follows.

```

# MS word report
report.augmentedRCBD.bulk(aug.bulk = bout,
                          target = file.path(tempdir(),
                          "augmentedRCBD bulk output.docx"))

```

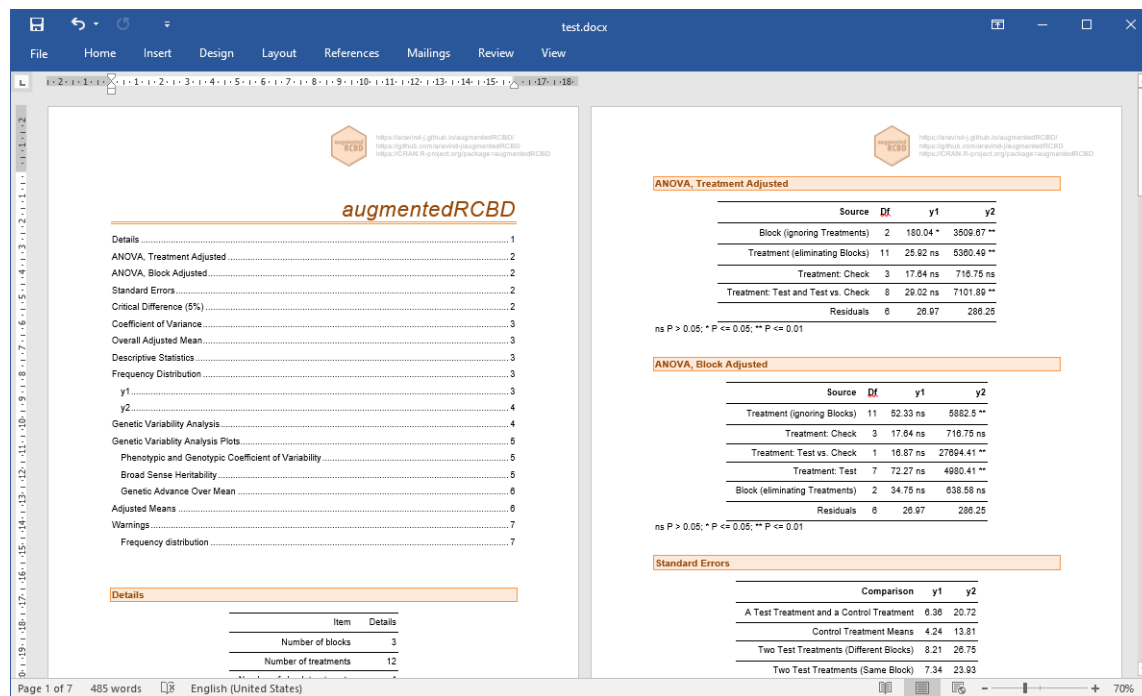


Fig. 8: MS Word report generated with `report.augmentedRCBD.bulk` function.

9 Citing `augmentedRCBD`

To cite the R package '`augmentedRCBD`' in publications use:

Aravind, J., Mukesh Sankar, S., Wankhede, D. P., and Kaur, V. (2021). `augmentedRCBD`: Analysis of A Complete Block Designs. R package version 0.1.5,
<https://aravind-j.github.io/augmentedRCBD/><https://cran.r-project.org/package=augmentedRCBD>.

A BibTeX entry for LaTeX users is

```
@Manual{,
  title = {augmentedRCBD: Analysis of Augmented Randomised Complete Block Designs},
  author = {J. Aravind and S. {Mukesh Sankar} and Dhammaprakash Pandhari Wankhede and Vikender Kaur},
  year = {2021},
  note = {R package version 0.1.5},
  note = {https://aravind-j.github.io/augmentedRCBD/},
  note = {https://cran.r-project.org/package=augmentedRCBD},
}
```

This free and open-source software implements academic research by the authors and co-workers. If you use the project by citing the package.

10 Session Info

```
sessionInfo()
```

```
R Under development (unstable) (2022-06-05 r82452 ucrt)
```

```
Platform: x86_64-w64-mingw32/x64 (64-bit)
```

```
Running under: Windows 10 x64 (build 19044)
```

```
Matrix products: default
```

```
locale:
```

```
[1] LC_COLLATE=English_India.utf8  LC_CTYPE=English_India.utf8    LC_MONETARY=English_India.utf8 LC_N
```

```
[5] LC_TIME=English_India.utf8
```

```
attached base packages:
```

```
[1] stats      graphics  grDevices  utils      datasets  methods    base
```

```
other attached packages:
```

```
[1] diagram_1.6.5      shape_1.4.6        augmentedRCBD_0.1.5
```

```
loaded via a namespace (and not attached):
```

```
[1] Rdpack_2.3.1      DBI_1.1.3          bitops_1.0-7       remotes_2.4.2      testthat_3.1.4
[8] magrittr_2.0.3    multcomp_1.4-19    hunspell_3.0.1     compiler_4.3.0     systemfonts_1.0.4
[15] reshape2_1.4.4    stringr_1.4.0      pkgconfig_2.0.3    crayon_1.5.1       fastmap_1.1.0
[22] pander_0.6.5      utf8_1.2.2         rmarkdown_2.14     sessioninfo_1.2.2  ps_1.7.1
[29] xfun_0.31         cachem_1.0.6       covr_3.5.1         jsonlite_1.8.0     highr_0.9
[36] R6_2.5.1          rhub_1.1.1         stringi_1.7.6      pkgload_1.2.4      brio_1.1.3
[43] assertthat_0.2.1  knitr_1.39         usethis_2.1.6      zoo_1.8-10         clisymbols_1.2.0
[50] Matrix_1.4-1      splines_4.3.0      tidyselect_1.1.2   rstudioapi_0.13    yaml_2.3.5
[57] processx_3.6.1    pkgbuild_1.3.1     lattice_0.20-45    tibble_3.1.7       plyr_1.8.7
[64] coda_0.19-4       evaluate_0.15      moments_0.14.1     desc_1.4.1         survival_3.3-1
[71] xml2_1.3.3        pillar_1.7.0       rex_1.2.1          generics_0.1.2     mathjaxr_1.6-0
[78] RCurl_1.98-1.7    ggplot2_3.3.6      munsell_0.5.0      scales_1.2.0       xtable_1.8-4
[85] emmeans_1.7.5     lazyeval_0.2.2     tools_4.3.0        data.table_1.14.2  goodpractice_1.0.2
[92] XML_3.99-0.10     grid_4.3.0         cyclocomp_1.1.0    rbibutils_2.2.8    colorspace_2.0-3
[99] cli_3.3.0         rappdirs_0.3.3     officer_0.4.3      rcmdcheck_1.4.0    fansi_1.0.3
[106] gtable_0.3.0      praise_1.0.0       digest_0.6.29      TH.data_1.1-1      farver_2.1.0
```

[113] `htmltools_0.5.2` `pkgdown_2.0.5` `lifecycle_1.0.1` `multcompView_0.1-8` `httr_1.4.3`

References

- Anscombe, F. J., and Glynn, W. J. (1983). Distribution of the kurtosis statistic b_2 for normal samples. *Biometrika* 70, 227–234. doi:[10.1093/biomet/70.1.227](https://doi.org/10.1093/biomet/70.1.227).
- Burton, G. W. (1951). Quantitative inheritance in pearl millet (*Pennisetum glaucum*). *Agronomy Journal* 43, 409–417. doi:[10.2134/agronj1951.00021962004300090001x](https://doi.org/10.2134/agronj1951.00021962004300090001x).
- Burton, G. W. (1952). Qualitative inheritance in grasses. Vol. 1. in *Proceedings of the 6th International Grassland Congress, Pennsylvania State College*, 17–23.
- D’Agostino, R. B. (1970). Transformation to normality of the null distribution of g_1 . *Biometrika* 57, 679–681. doi:[10.1093/biomet/57.3.679](https://doi.org/10.1093/biomet/57.3.679).
- Dudley, J. W., and Moll, R. H. (1969). Interpretation and use of estimates of heritability and genetic variances in plant breeding. *Crop Science* 9, 257–262. doi:[10.2135/cropsci1969.0011183X000900030001x](https://doi.org/10.2135/cropsci1969.0011183X000900030001x).
- Federer, W. T. (1956). Augmented (or Hoonuiaku) Designs. New York: Cornell University Available at: <https://ecommons.cornell.edu/handle/1813/32841>.
- Federer, W. T. (1956). Augmented (or Hoonuiaku) designs. *The Hawaiian Planters’ Record* LV(2), 191–208.
- Federer, W. T. (1961). Augmented designs with one-way elimination of heterogeneity. *Biometrics* 17, 447–473. doi:[10.2307/2527837](https://doi.org/10.2307/2527837).
- Federer, W. T., and Searle, S. R. (1976). Model Considerations and Variance Component Estimation in Augmented Completely Randomized and Randomized Complete Blocks Designs-Preliminary Version. New York: Cornell University Available at: <https://hdl.handle.net/1813/32691>.
- Johnson, H. W., Robinson, H. F., and Comstock, R. E. (1955). Estimates of genetic and environmental variability in soybeans. *Agronomy journal* 47, 314–318. doi:[10.2134/agronj1955.00021962004700070009x](https://doi.org/10.2134/agronj1955.00021962004700070009x).
- Lush, J. L. (1940). Intra-sire correlations or regressions of offspring on dam as a method of estimating heritability of characteristics. *Proceedings of the American Society of Animal Nutrition* 1940, 293–301. doi:[10.2527/jas1940.19401293x](https://doi.org/10.2527/jas1940.19401293x).
- Robinson, H. F. (1966). Quantitative genetics in relation to breeding on centennial of Mendelism. *Indian Journal of Genetics and Plant Breeding*, 171.
- Robinson, H. F., Comstock, R. E., and Harvey, P. H. (1955). Genetic variances in open pollinated varieties of corn. *Genetics* 40, 45–60. doi:[10.1093/genetics/40.1.45](https://doi.org/10.1093/genetics/40.1.45).
- Sivasubramaniam, S., and Madhavamenon, P. (1973). Genotypic and phenotypic variability in rice. *The Madras Agricultural Journal* 60, 1093–1096.
- Tippmann, S. (2015). Programming tools: Adventures with R. *Nature News* 517, 109. doi:[10.1038/517109a](https://doi.org/10.1038/517109a).