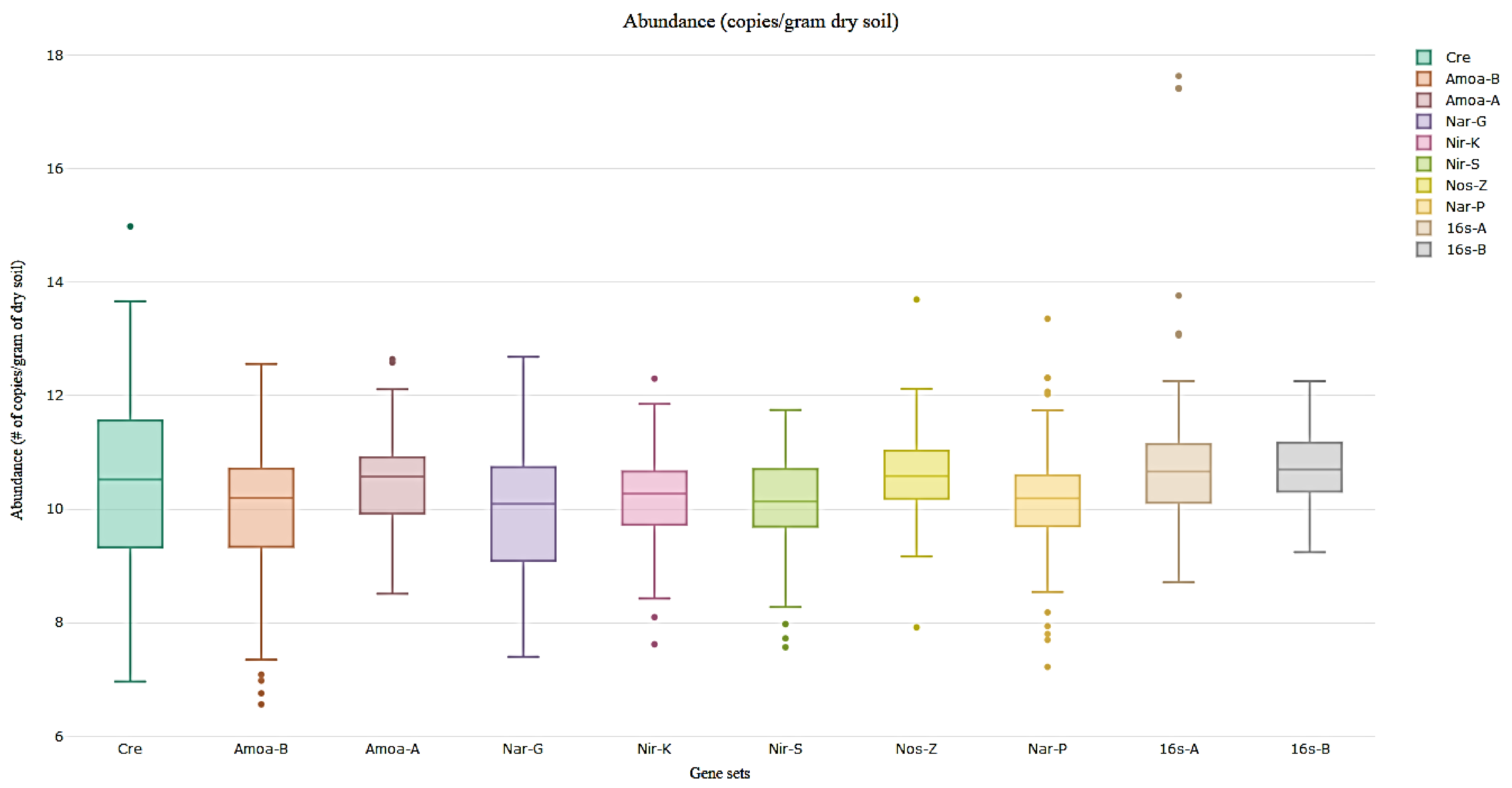
# Introduction to R and Executing General Linear Models

Module 3 & 4

Victor Valdez

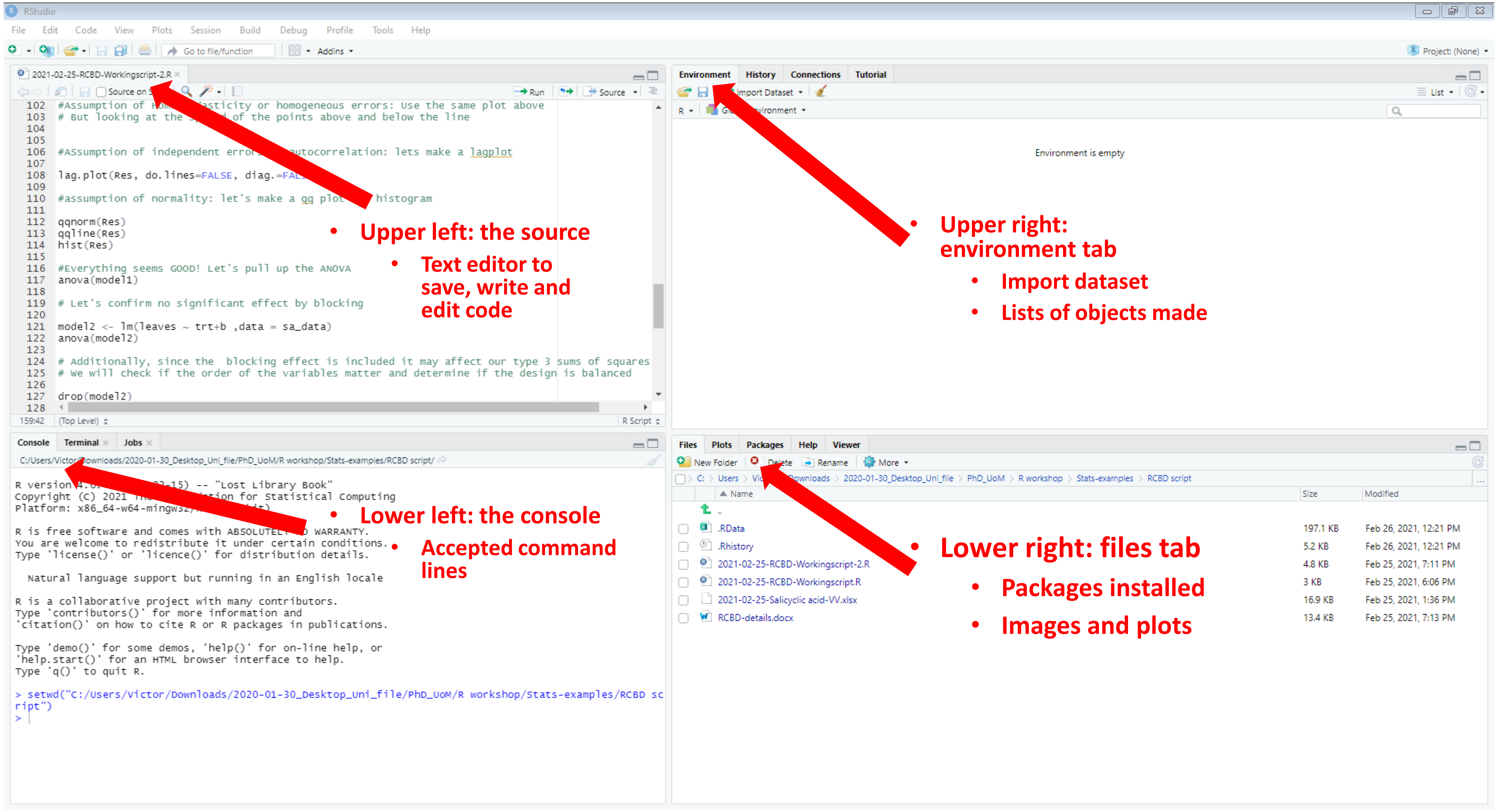
## Why use R?

- FREE
    - Powerful and flexible (kind of) statistical and graphical package
- Most frequently used statistical package
    - An increasing number of users with R
    - Open-source software
    - Versatility with coding allows for robust analysis of data



Abundance (copies/gram dry soil)
18
16
14
12
10
8
6
Abundance (# of copies/gram of dry soil)
Cre
Amoa-B
Amoa-A
Nar-G
Nir-K
Nir-S
Nos-Z
Nar-P
16s-A
16s-B
Gene sets

# What am I looking at?


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
2021-02-25-RCBD-Workingscript-2.R
Source on S
Run
Source
159:42
(Top Level)
R Script
Environment History Connections Tutorial
Import Dataset
List
R
Gl[illegible]vironment
Environment is empty
Console Terminal Jobs
Files Plots Packages Help Viewer
New Folder
Delete
Rename
More
C: > Users > Vi[illegible] > Downloads > 2020-01-30_Desktop_Uni_file > PhD_UoM > R workshop > Stats-examples > RCBD script

```
#Assumption of Hom[illegible]asticity or homogeneous errors: Use the same plot above
# But looking at the s[illegible]d of the points above and below the line


#ASsumption of independent erro[illegible]autocorrelation: lets make a lagplot

lag.plot(Res, do.lines=FALSE, diag.=FAL[illegible]

#assumption of normality: let's make a qq plot [illegible] histogram

qqnorm(Res)
qqline(Res)
hist(Res)

#Everything seems GOOD! Let's pull up the ANOVA
anova(model1)

# Let's confirm no significant effect by blocking

model2 <- lm(leaves ~ trt+b ,data = sa_data)
anova(model2)

# Additionally, since the  blocking effect is included it may affect our type 3 sums of squares
# We will check if the order of the variables matter and determine if the design is balanced

drop(model2)
```

```
C:/Users/Victor/Downloads/2020-01-30_Desktop_Uni_file/PhD_UoM/R workshop/Stats-examples/RCBD script/

R version 4.[illegible]-15) -- "Lost Library Book"
Copyright (C) 2021 [illegible]ation for Statistical Computing
Platform: x86_64-w64-mingw32/[illegible]it)

R is free software and comes with ABSOLUTEL[illegible]O WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> setwd("C:/Users/Victor/Downloads/2020-01-30_Desktop_Uni_file/PhD_UoM/R workshop/Stats-examples/RCBD sc
ript")
> 
```

| Name | Size | Modified |
|---|---|---|
| .. | | |
| .RData | 197.1 KB | Feb 26, 2021, 12:21 PM |
| .Rhistory | 5.2 KB | Feb 26, 2021, 12:21 PM |
| 2021-02-25-RCBD-Workingscript-2.R | 4.8 KB | Feb 25, 2021, 7:11 PM |
| 2021-02-25-RCBD-Workingscript.R | 3 KB | Feb 25, 2021, 6:06 PM |
| 2021-02-25-Salicyclic acid-VV.xlsx | 16.9 KB | Feb 25, 2021, 1:36 PM |
| RCBD-details.docx | 13.4 KB | Feb 25, 2021, 7:13 PM |

- **Upper left: the source**
  - **Text editor to save, write and edit code**
- **Upper right: environment tab**
  - **Import dataset**
  - **Lists of objects made**
- **Lower left: the console**
  - **Accepted command lines**
- **Lower right: files tab**
  - **Packages installed**
  - **Images and plots**

# Getting started

- RStudio
  - Makes R easier to use
  - Start by keeping your work organized by:

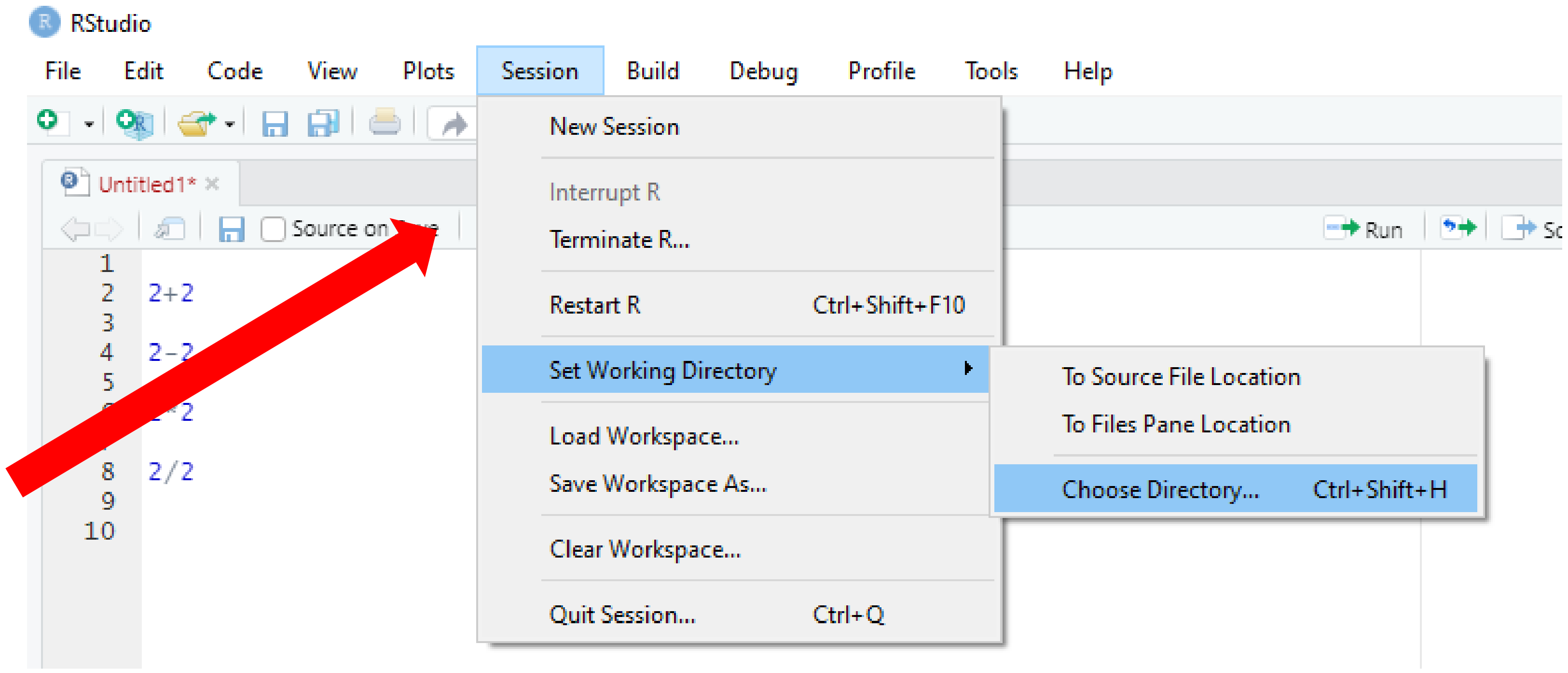
    **Set your *<u>"Working Directory"</u>***

    **OR**

    Ctrl+shift+H

    **OR**

    setwd("C:/Users/Home/Downloads/Workshop_data")



```

2+2

2-2

2*2

2/2
```

# Getting started

- Write some simple stuff in the text editor

  2 + 2

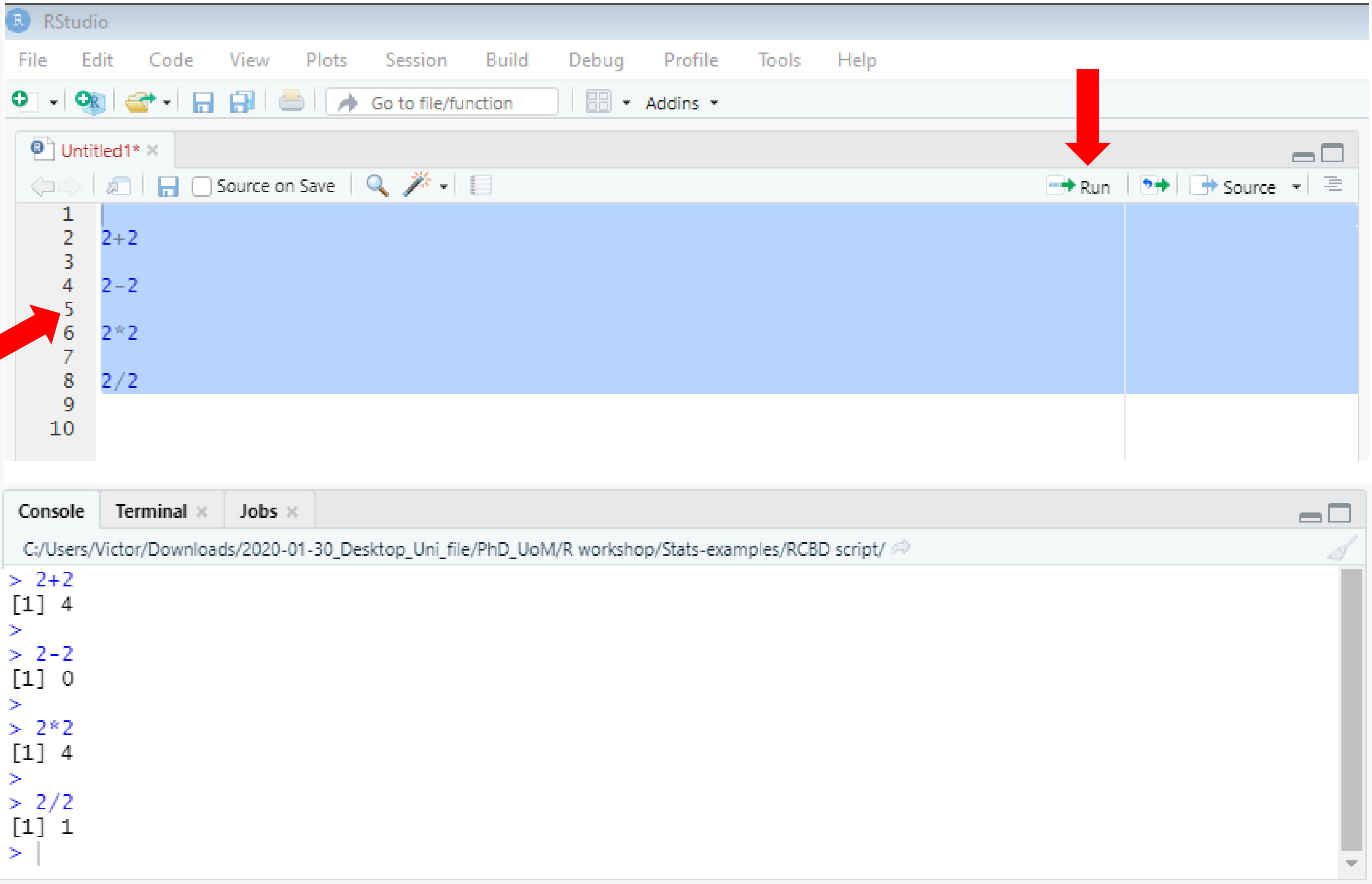
  2 - 2

  2 * 2

  2 / 2

- **Highlight** code and press "ctrl + enter" or "Run"

- Output is in the console!



```
2+2

2-2

2*2

2/2
```

C:/Users/Victor/Downloads/2020-01-30_Desktop_Uni_file/PhD_UoM/R workshop/Stats-examples/RCBD script/

```
> 2+2
[1] 4
>
> 2-2
[1] 0
>
> 2*2
[1] 4
>
> 2/2
[1] 1
>
```

# Manual data entry

## Vectors and Matrices

- Create a **vector object** called "a" and "b" of 3 lengths each

  a <-c(0,1,2)

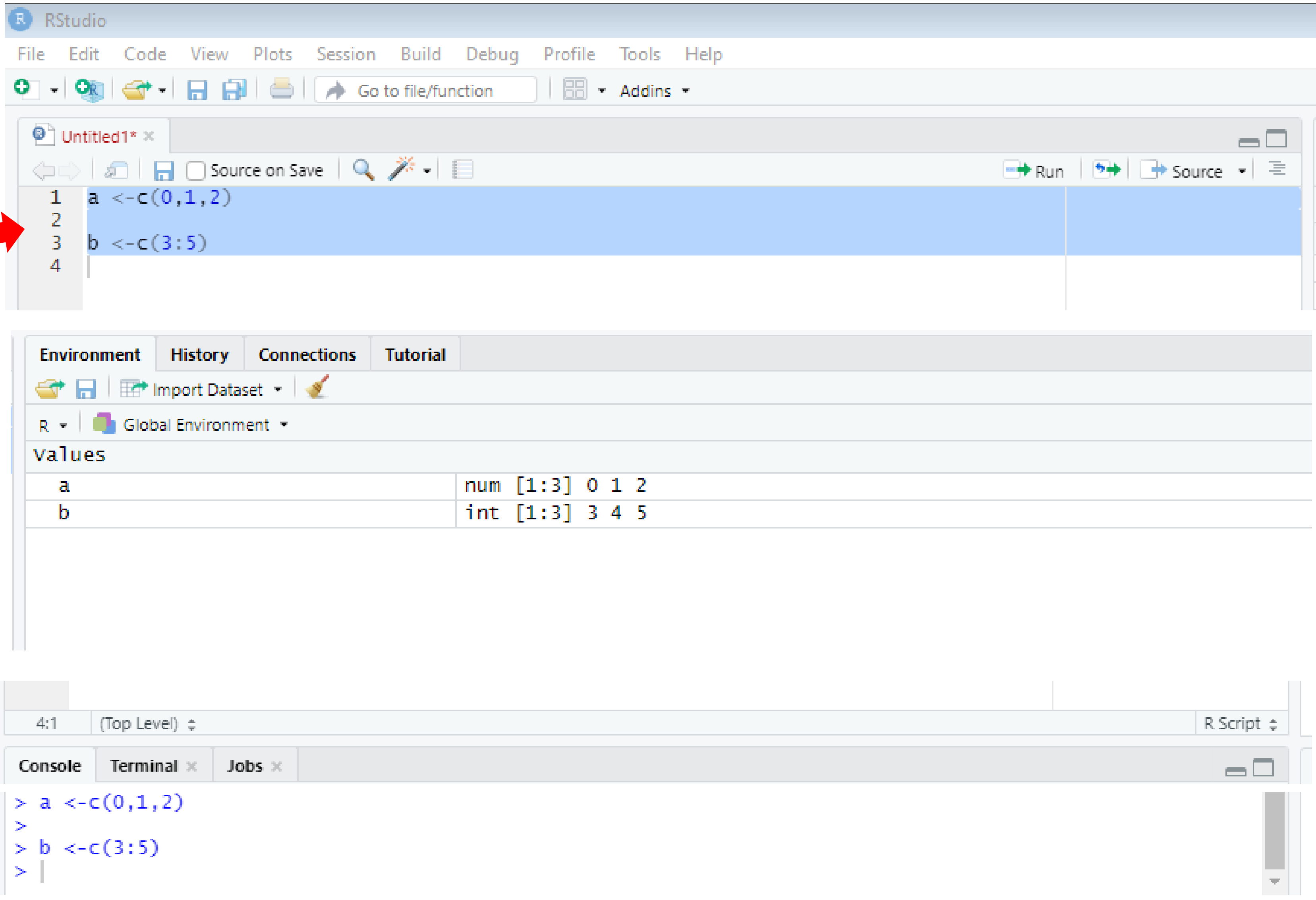
  b <-c(3:5)

- **Type name** and **run code** ***OR*** **click** on object **in global env.**

- Output in console!

- What it means:

  **a** and **b** is the vector object created
  **<-** is the direction items are going
  **c()** is the concatenate function
  **0,1,2** are the items being added ***OR***
  **3:5** is an inclusive function



```
a <-c(0,1,2)

b <-c(3:5)
```

```
Values
  a        num [1:3] 0 1 2
  b        int [1:3] 3 4 5
```

```
> a <-c(0,1,2)
>
> b <-c(3:5)
>
```

# Manual data entry

## Vectors and Matrices

- Create an object matrix “c” with object vectors “a” and “b”

  c <-c(a, b)

- **Type name** and **run code** ***OR*** **click** on object **in global env.**

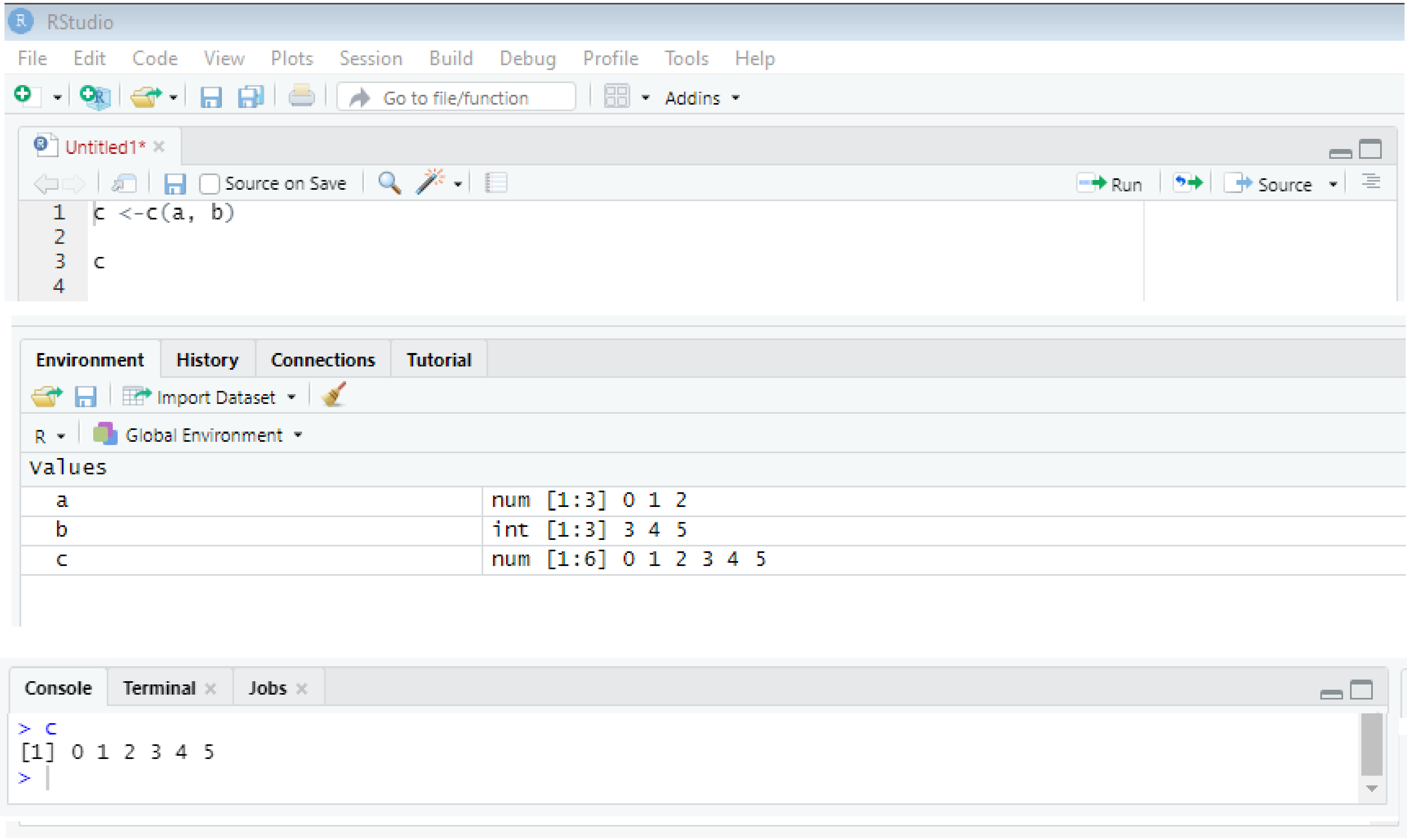
- What it means:

  **c** is the new object with **a** and **b**

  **<-** is the direction items are going

  **c()** is the concatenate function

  **a** and **b** are the items being added



```
c <-c(a, b)

c
```

```
Values
  a          num [1:3] 0 1 2
  b          int [1:3] 3 4 5
  c          num [1:6] 0 1 2 3 4 5
```

```
> c
[1] 0 1 2 3 4 5
>
```

# Manual data entry

## Vectors and Matrices

- Create a character vector with letters

  d<-letters [1:10]

- Then bind to numeric vector by column

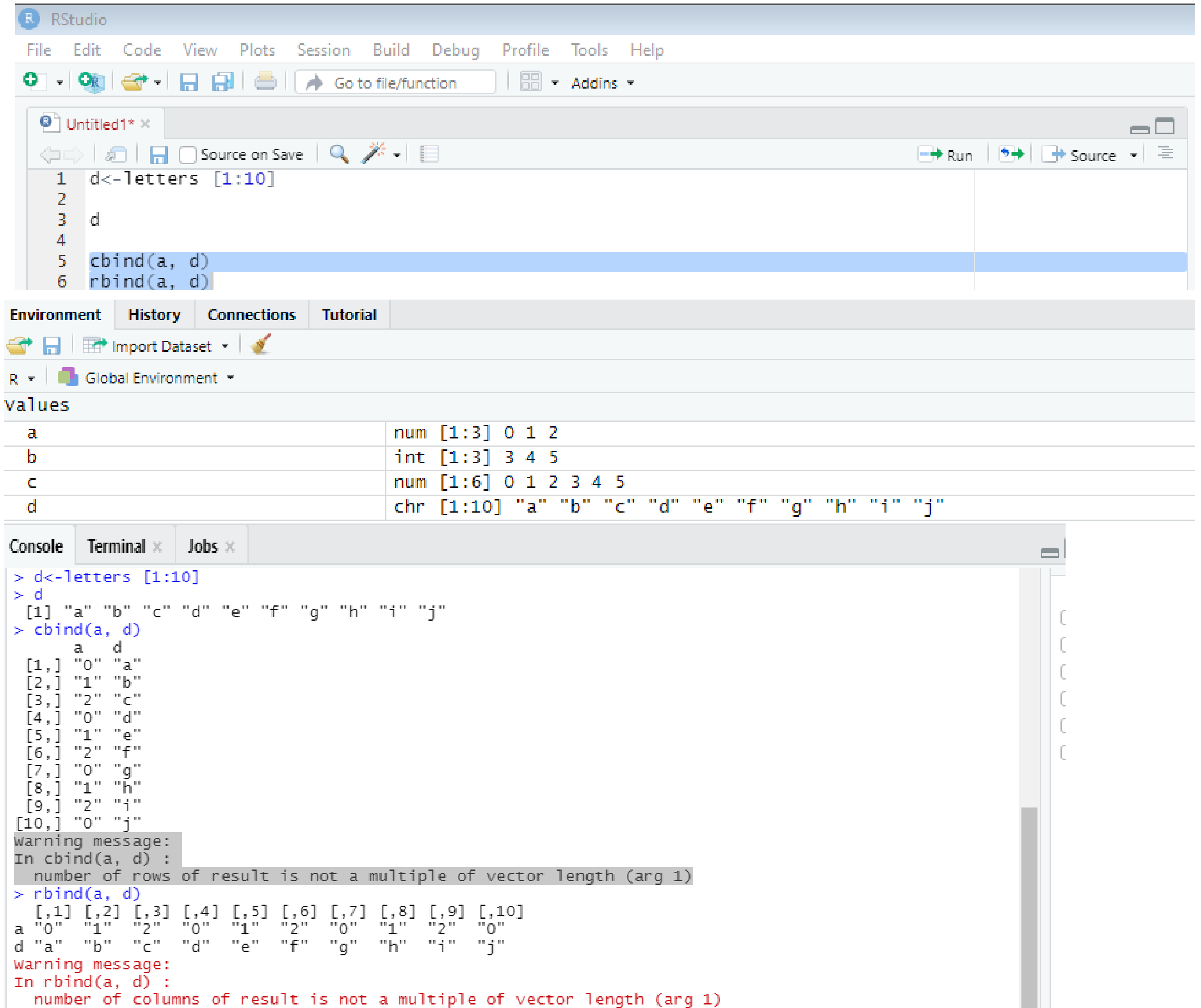
  cbind(a, d)

- Now bind by row

  rbind(a, d)

- What it means:

  **Letter []** is the alphabet function
  **[]** specify parts of the object
  **cbind()** bind by column
  **rbind()** bind by row
  **Warning message** R tells us when something is wrong. We can ignore this



```
d<-letters [1:10]

d

cbind(a, d)
rbind(a, d)
```

```
Values
a        num [1:3] 0 1 2
b        int [1:3] 3 4 5
c        num [1:6] 0 1 2 3 4 5
d        chr [1:10] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
```

```
> d<-letters [1:10]
> d
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
> cbind(a, d)
      a   d
 [1,] "0" "a"
 [2,] "1" "b"
 [3,] "2" "c"
 [4,] "0" "d"
 [5,] "1" "e"
 [6,] "2" "f"
 [7,] "0" "g"
 [8,] "1" "h"
 [9,] "2" "i"
[10,] "0" "j"
Warning message:
In cbind(a, d) :
  number of rows of result is not a multiple of vector length (arg 1)
> rbind(a, d)
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
a "0"  "1"  "2"  "0"  "1"  "2"  "0"  "1"  "2"  "0"
d "a"  "b"  "c"  "d"  "e"  "f"  "g"  "h"  "i"  "j"
Warning message:
In rbind(a, d) :
  number of columns of result is not a multiple of vector length (arg 1)
```

# Manual data entry

- Create matrix object

  newmat <- matrix(1:9, nrow=3, ncol=3)

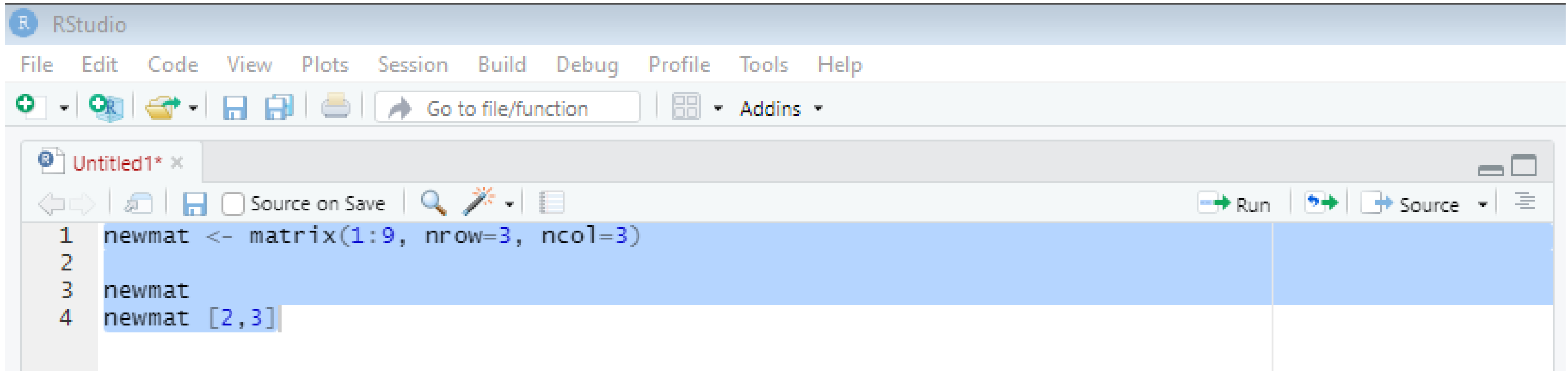
  newmat

  newmat[2,3]

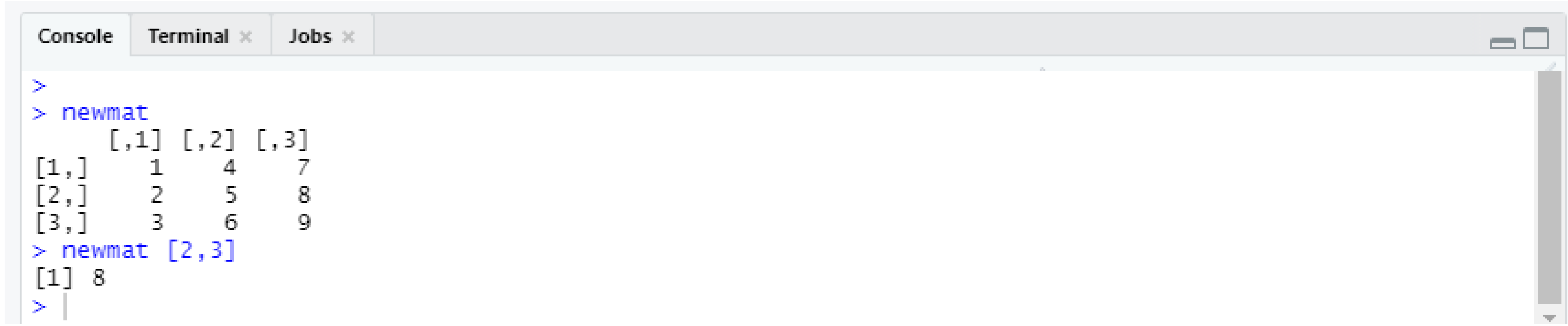
- Notice how rows come first then columns

- What it all means:

  **newmat** is a matrix object
  **matrix()** is the matrix object function
  **nrow** is the number of row function
  **ncol** is the number of columns function
  **=** denotes the value to use



```
newmat <- matrix(1:9, nrow=3, ncol=3)

newmat
newmat [2,3]
```

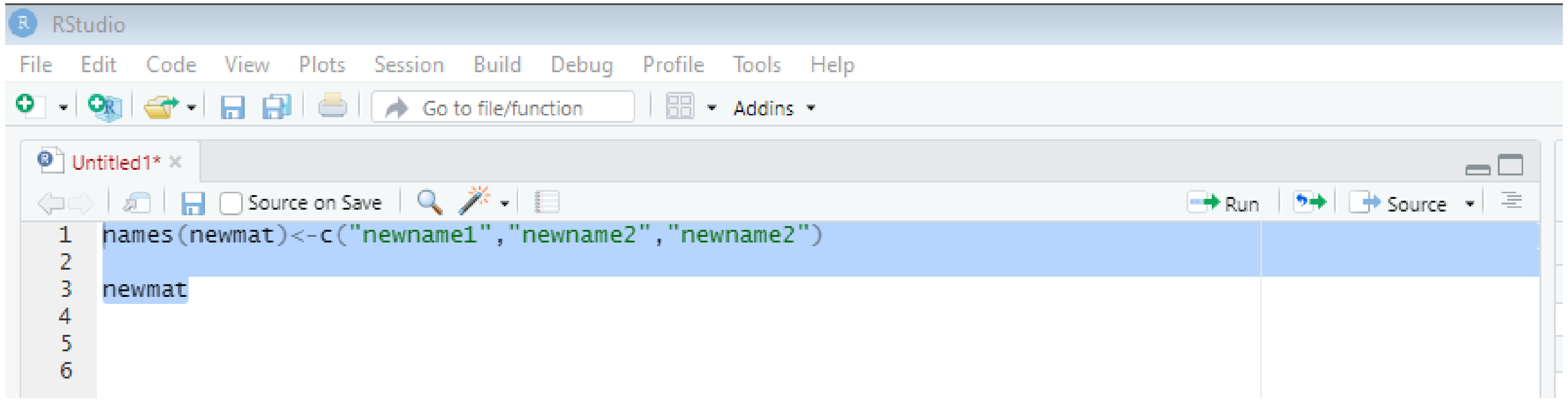


```
>
> newmat
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> newmat [2,3]
[1] 8
>
```

# Manual data entry

- To rename any column in your object

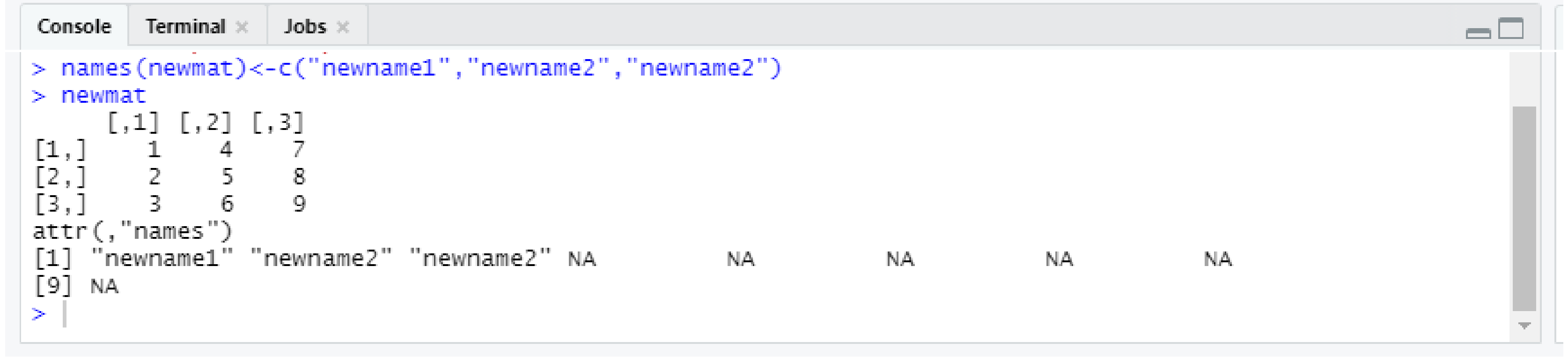
  names(newmat)<-c("newname1", "newname2","newname2")



```
names(newmat)<-c("newname1","newname2","newname2")

newmat
```

- What it all means:

  **names()** is the renaming function
  **<-** where to put the items
  **newmat** is the matrix object
  **c()** is concatenate function
  **newname1** is the new name of the column
  **“”** denotes what to rename the columns as
  **,** is used to separate the different names



```
> names(newmat)<-c("newname1","newname2","newname2")
> newmat
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
attr(,"names")
[1] "newname1" "newname2" "newname2" NA         NA         NA         NA         NA        
[9] NA
> 
```

# Manual data entry:

- Make a data frame

  Data <-data.frame(a,b)

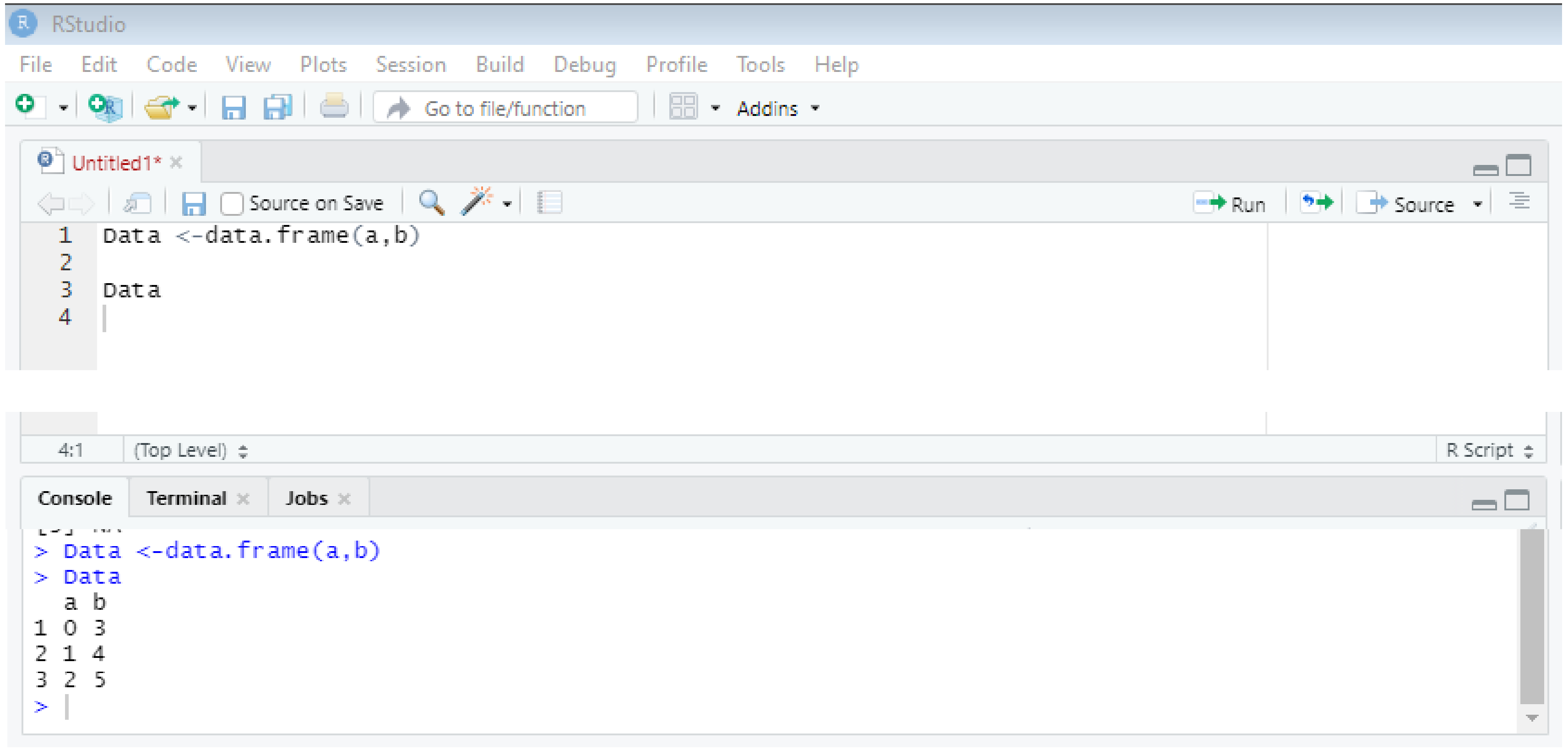
- A data frame is similar to an excel sheet; it is a list of vector of equal lengths

- But stores data as a table, can contain multiple data types in multiple columns called fields

- What it means:

  **data.frame ()** is the function that turns objects into a data frame

  **a,b** are your objects



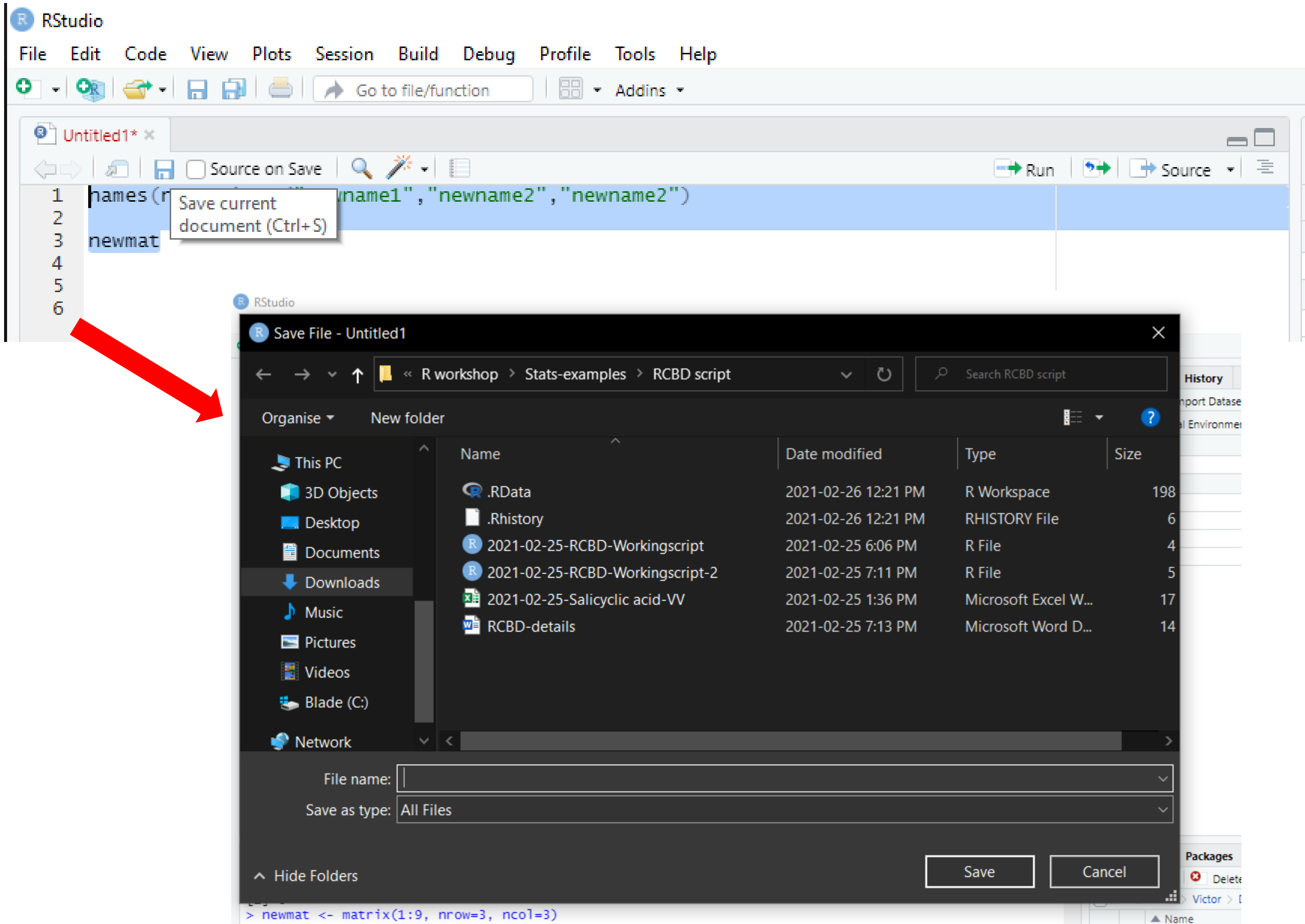
```
Data <-data.frame(a,b)

Data
```

```
> Data <-data.frame(a,b)
> Data
  a b
1 0 3
2 1 4
3 2 5
>
```

# Manual data entry

- To keep any working code you generated, and to edit it at a later time

- Choose the location of your files



```
hames(r...name1","newname2","newname2")

newmat
```

| Name | Date modified | Type | Size |
|---|---|---|---|
| .RData | 2021-02-26 12:21 PM | R Workspace | 198 |
| .Rhistory | 2021-02-26 12:21 PM | RHISTORY File | 6 |
| 2021-02-25-RCBD-Workingscript | 2021-02-25 6:06 PM | R File | 4 |
| 2021-02-25-RCBD-Workingscript-2 | 2021-02-25 7:11 PM | R File | 5 |
| 2021-02-25-Salicyclic acid-VV | 2021-02-25 1:36 PM | Microsoft Excel W... | 17 |
| RCBD-details | 2021-02-25 7:13 PM | Microsoft Word D... | 14 |

```
> newmat <- matrix(1:9, nrow=3, ncol=3)
```

# Descriptive statistics

- Other mathematical functions we can use w/ our data frames

  min (dataset$variable_name)
  max (dataset$ variable_name)
  IQR (dataset$ variable_name)
  mean (dataset$ variable_name)
  median (dataset$ variable_name)
  var (dataset$ variable_name)
  sd (dataset$ variable_name)

- What it means:

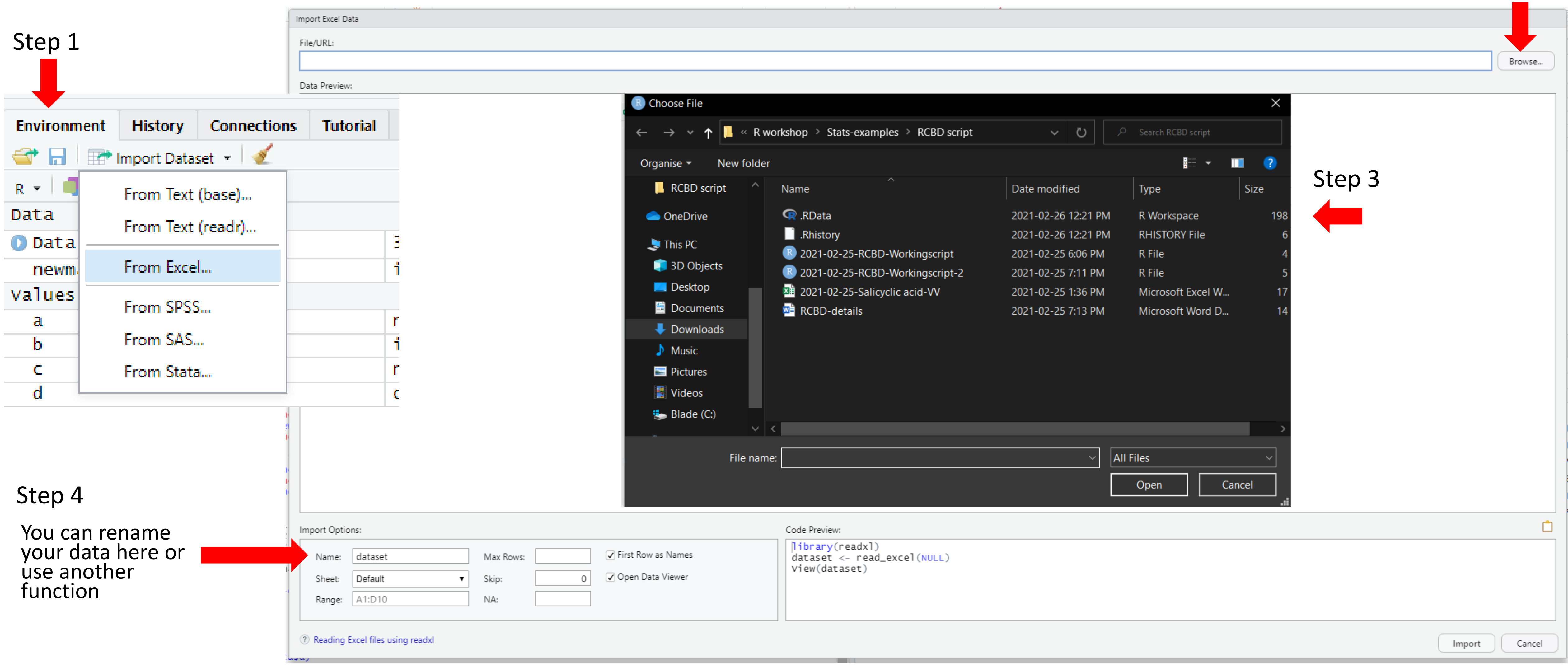
  **dataset** is the data frame you are referring to
  **$** is used to call up the list of columns in the data frame
  **variable_name** is the column

# Import data into R

Step 1

Step 2

Step 3



| Name | Date modified | Type | Size |
|---|---|---|---|
| .RData | 2021-02-26 12:21 PM | R Workspace | 198 |
| .Rhistory | 2021-02-26 12:21 PM | RHISTORY File | 6 |
| 2021-02-25-RCBD-Workingscript | 2021-02-25 6:06 PM | R File | 4 |
| 2021-02-25-RCBD-Workingscript-2 | 2021-02-25 7:11 PM | R File | 5 |
| 2021-02-25-Salicyclic acid-VV | 2021-02-25 1:36 PM | Microsoft Excel W... | 17 |
| RCBD-details | 2021-02-25 7:13 PM | Microsoft Word D... | 14 |

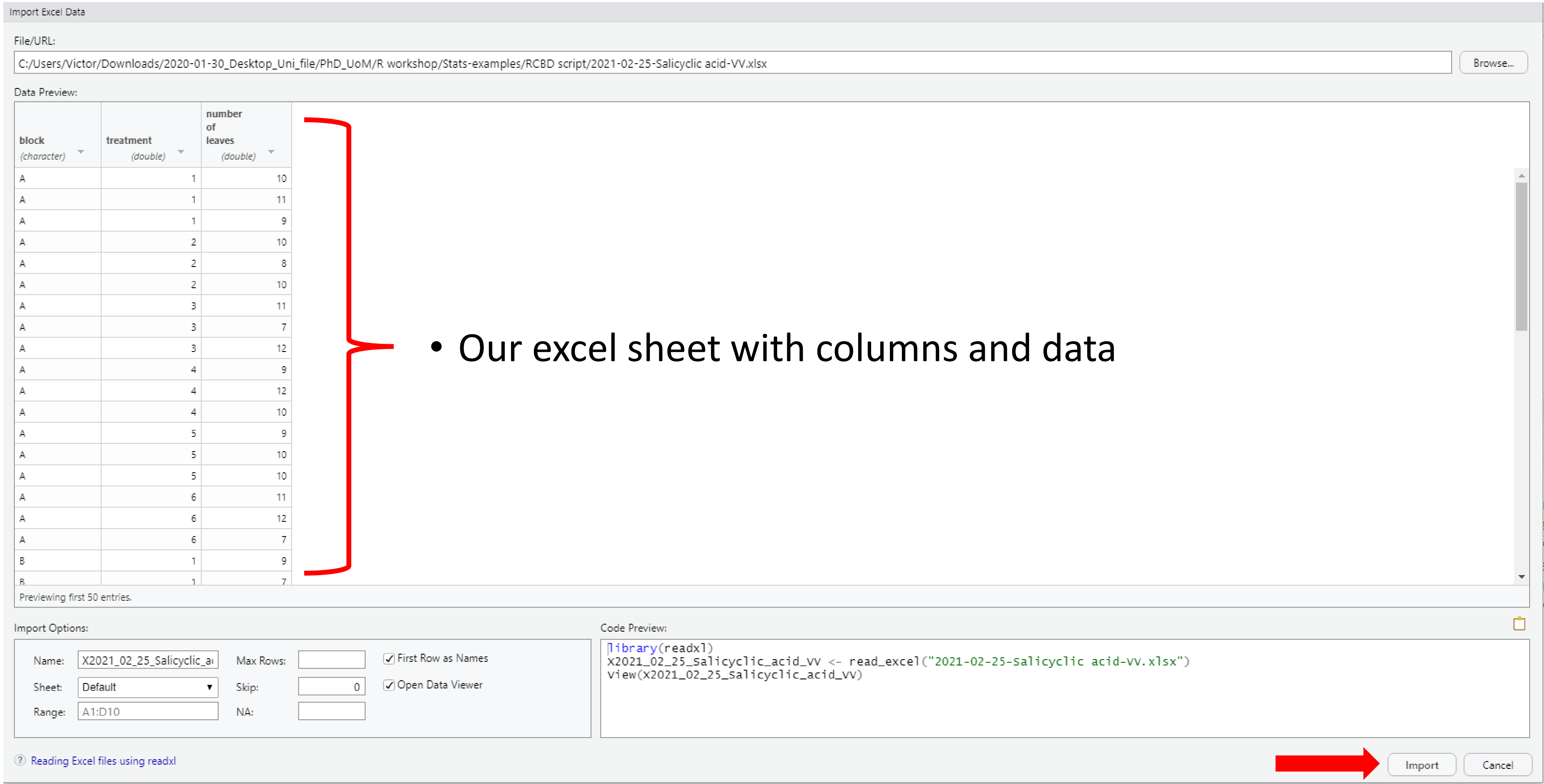
| | | | |
|---|---|---|---|
| Name: | dataset | Max Rows: | |
| Sheet: | Default | Skip: | 0 |
| Range: | A1:D10 | NA: | |

```
library(readxl)
dataset <- read_excel(NULL)
View(dataset)
```

Step 4

- You can rename your data here or use another function

- You should now see the following; ***click*** import



Import Excel Data

File/URL:

C:/Users/Victor/Downloads/2020-01-30_Desktop_Uni_file/PhD_UoM/R workshop/Stats-examples/RCBD script/2021-02-25-Salicyclic acid-VV.xlsx

Data Preview:

| block (character) | treatment (double) | number of leaves (double) |
|---|---|---|
| A | 1 | 10 |
| A | 1 | 11 |
| A | 1 | 9 |
| A | 2 | 10 |
| A | 2 | 8 |
| A | 2 | 10 |
| A | 3 | 11 |
| A | 3 | 7 |
| A | 3 | 12 |
| A | 4 | 9 |
| A | 4 | 12 |
| A | 4 | 10 |
| A | 5 | 9 |
| A | 5 | 10 |
| A | 5 | 10 |
| A | 6 | 11 |
| A | 6 | 12 |
| A | 6 | 7 |
| B | 1 | 9 |
| B | 1 | 7 |

- Our excel sheet with columns and data

Previewing first 50 entries.

Import Options:

Name: X2021_02_25_Salicyclic_ac

Sheet: Default

Range: A1:D10

Max Rows:

Skip: 0

NA:

✓ First Row as Names

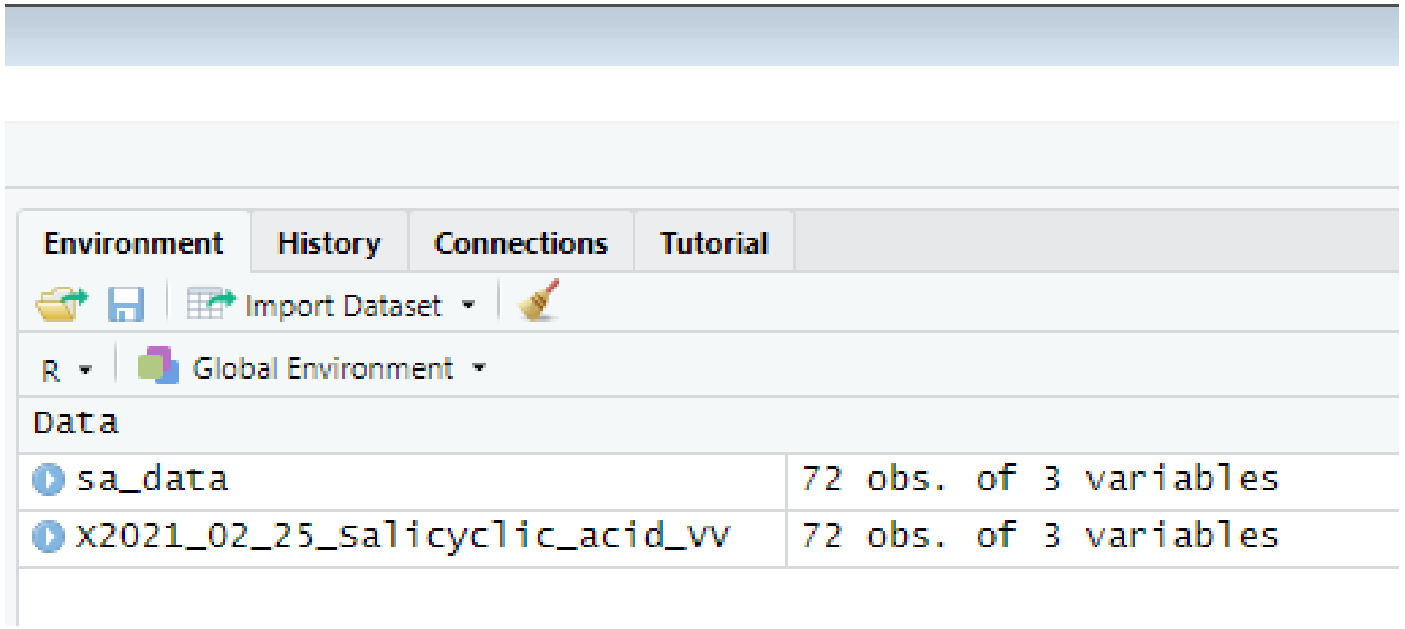
✓ Open Data Viewer

Code Preview:

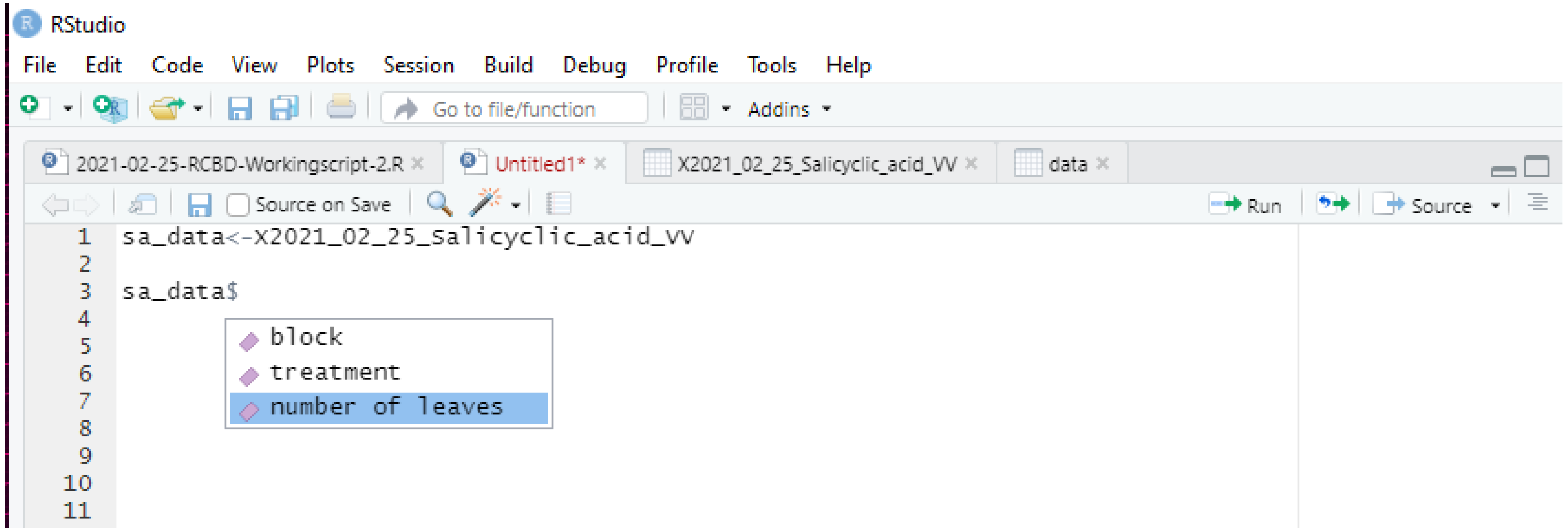
```
library(readxl)
X2021_02_25_Salicyclic_acid_VV <- read_excel("2021-02-25-Salicyclic acid-VV.xlsx")
View(X2021_02_25_Salicyclic_acid_VV)
```

- We now see the contents of the file name
- The file name may be too long; not ideal for coding
- Let's rename:

sa_data<-X2021_02_25_Salicyclic_acid_VV



```
sa_data                              72 obs. of 3 variables
X2021_02_25_Salicyclic_acid_VV       72 obs. of 3 variables
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
2021-02-25-RCBD-Workingscript-2.R
Untitled1*
X2021_02_25_Salicyclic_acid_VV
data
Source on Save
Run
Source

```
sa_data<-X2021_02_25_Salicyclic_acid_VV

sa_data$
        block
        treatment
        number of leaves
```

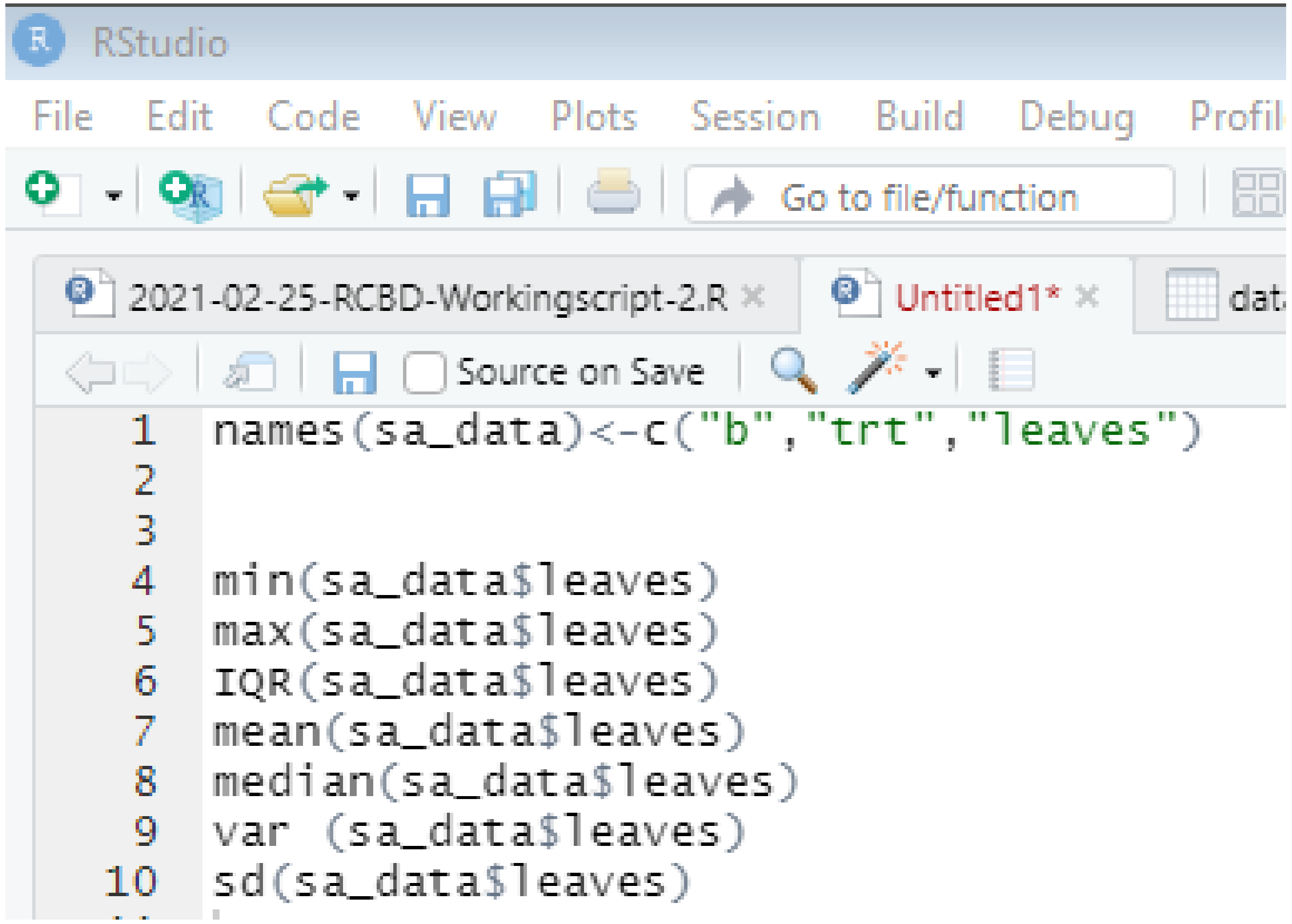
- We can now use the autofill feature by using **$ before the dataset**

  **e.g., data$....**

# Rename Columns and Run Descriptive Statistics

```
names(sa_data)<-c("b","trt","leaves")
```

```
min(sa_data$leaves)
max(sa_data$leaves)
IQR(sa_data$leaves)
mean(sa_data$leaves)
median(sa_data$leaves)
var(sa_data$leaves)
sd(sa_data$leaves)
```

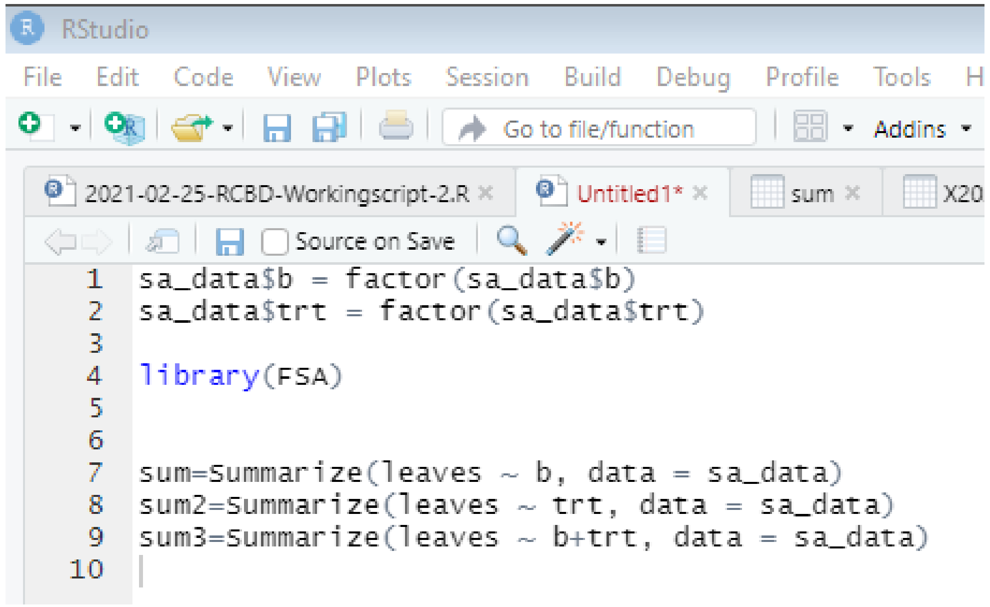


```
names(sa_data)<-c("b","trt","leaves")


min(sa_data$leaves)
max(sa_data$leaves)
IQR(sa_data$leaves)
mean(sa_data$leaves)
median(sa_data$leaves)
var (sa_data$leaves)
sd(sa_data$leaves)
```

# Run Descriptive Statistics Function

Change blocks and treatments to factorial:

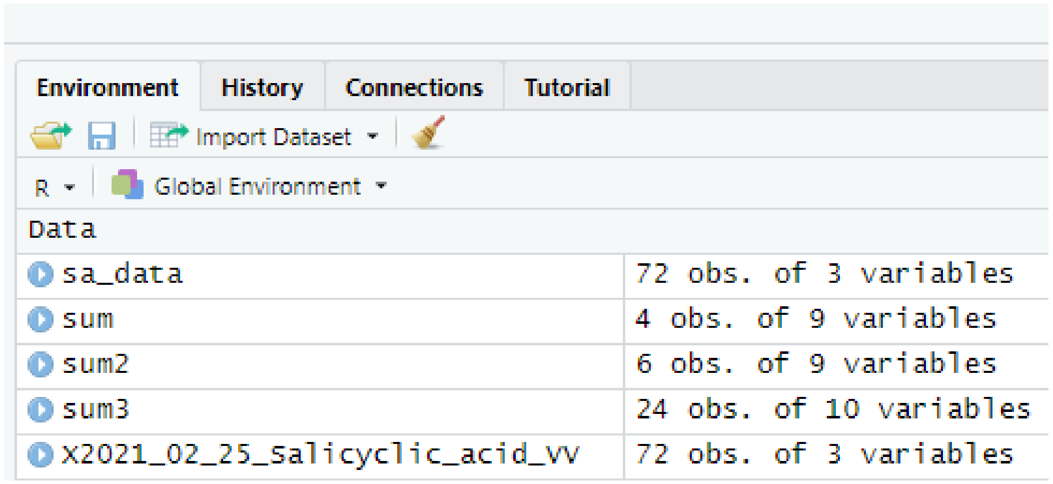
sa_data$b = factor(sa_data$b)
sa_data$trt = factor(sa_data$trt)


RStudio
File Edit Code View Plots Session Build Debug Profile Tools H
Go to file/function
Addins
2021-02-25-RCBD-Workingscript-2.R
Untitled1*
sum
X20
Source on Save

```
sa_data$b = factor(sa_data$b)
sa_data$trt = factor(sa_data$trt)

library(FSA)


sum=Summarize(leaves ~ b, data = sa_data)
sum2=Summarize(leaves ~ trt, data = sa_data)
sum3=Summarize(leaves ~ b+trt, data = sa_data)

```

Now use the summary function in FSA package:

sum=Summarize(leaves ~ b, data = sa_data)
sum2=Summarize(leaves ~ trt, data = sa_data)
sum3=Summarize(leaves ~ b+trt, data = sa_data)



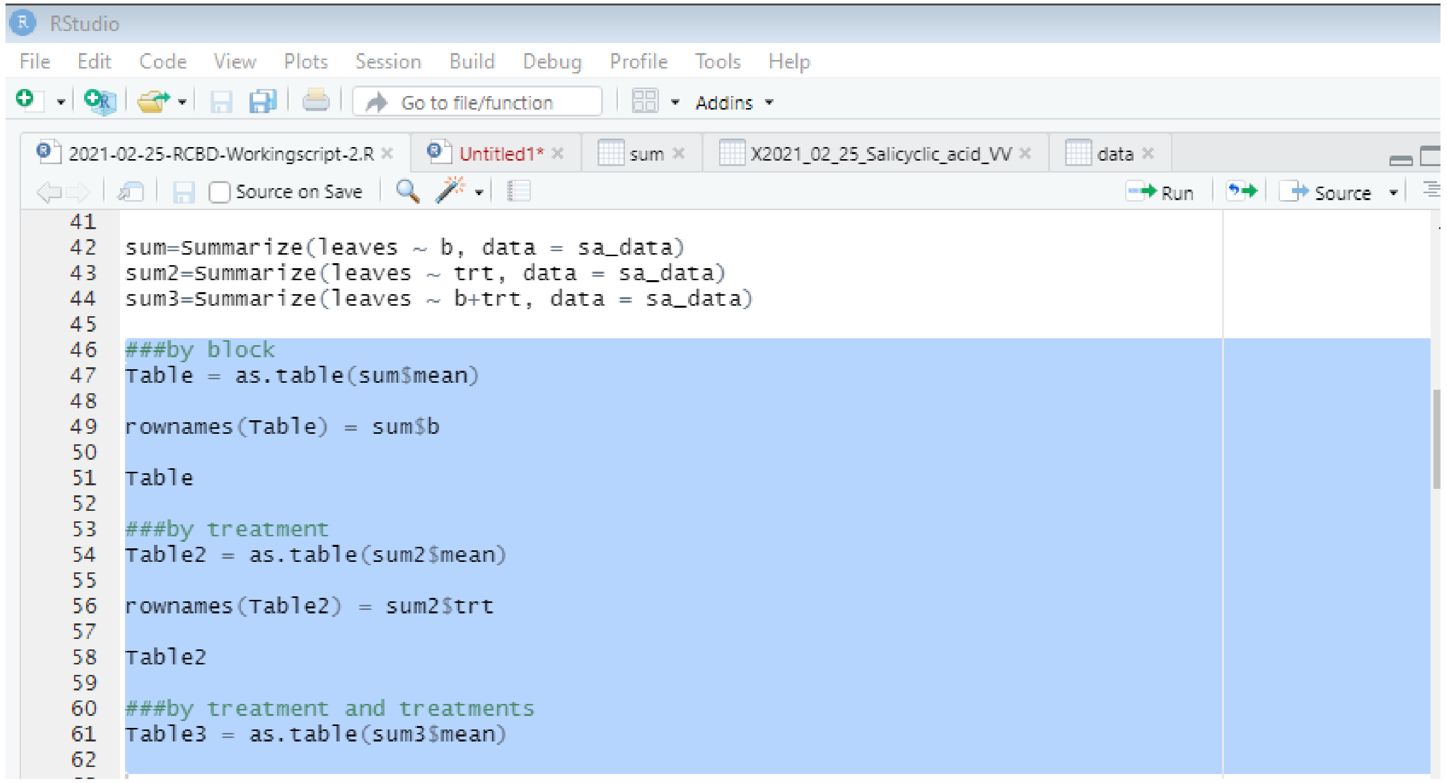
| Data | |
|---|---|
| sa_data | 72 obs. of 3 variables |
| sum | 4 obs. of 9 variables |
| sum2 | 6 obs. of 9 variables |
| sum3 | 24 obs. of 10 variables |
| X2021_02_25_Salicyclic_acid_VV | 72 obs. of 3 variables |

Let's look at the tables generated

# Plotting data: extract data

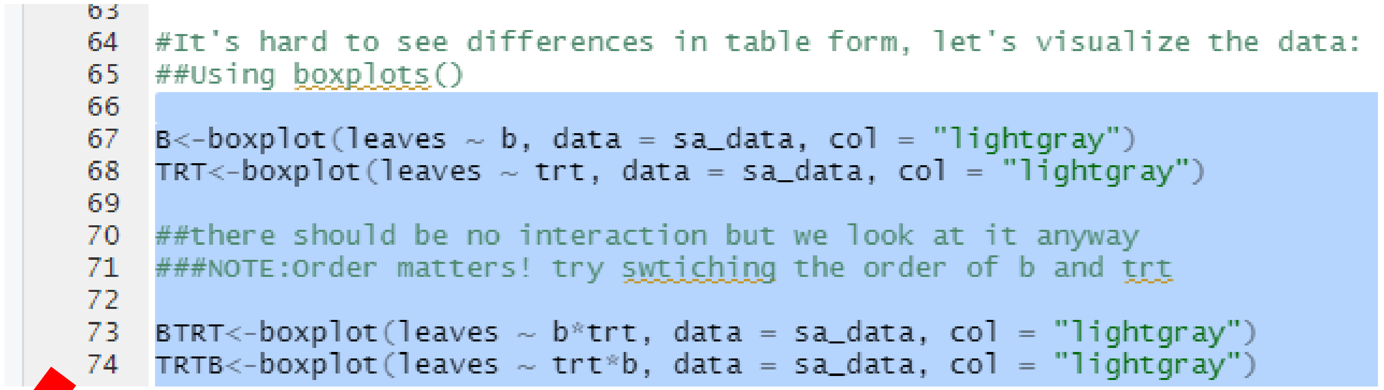
- Data is best plotted, let's make some boxplots
- First extract the mean from the table:

```
Table = as.table(sum$mean)

rownames(Table) = sum$b

Table2 = as.table(sum2$mean)

rownames(Table2) = sum2$trt

Table3 = as.table(sum3$mean)
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
2021-02-25-RCBD-Workingscript-2.R
Untitled1*
sum
X2021_02_25_Salicyclic_acid_VV
data
Source on Save
Run
Source

```

sum=Summarize(leaves ~ b, data = sa_data)
sum2=Summarize(leaves ~ trt, data = sa_data)
sum3=Summarize(leaves ~ b+trt, data = sa_data)

###by block
Table = as.table(sum$mean)

rownames(Table) = sum$b

Table

###by treatment
Table2 = as.table(sum2$mean)

rownames(Table2) = sum2$trt

Table2

###by treatment and treatments
Table3 = as.table(sum3$mean)

```

# Plotting data



```
#It's hard to see differences in table form, let's visualize the data:
##Using boxplots()

B<-boxplot(leaves ~ b, data = sa_data, col = "lightgray")
TRT<-boxplot(leaves ~ trt, data = sa_data, col = "lightgray")

##there should be no interaction but we look at it anyway
###NOTE:Order matters! try swtiching the order of b and trt

BTRT<-boxplot(leaves ~ b*trt, data = sa_data, col = "lightgray")
TRTB<-boxplot(leaves ~ trt*b, data = sa_data, col = "lightgray")
```

- Plot the boxplot :

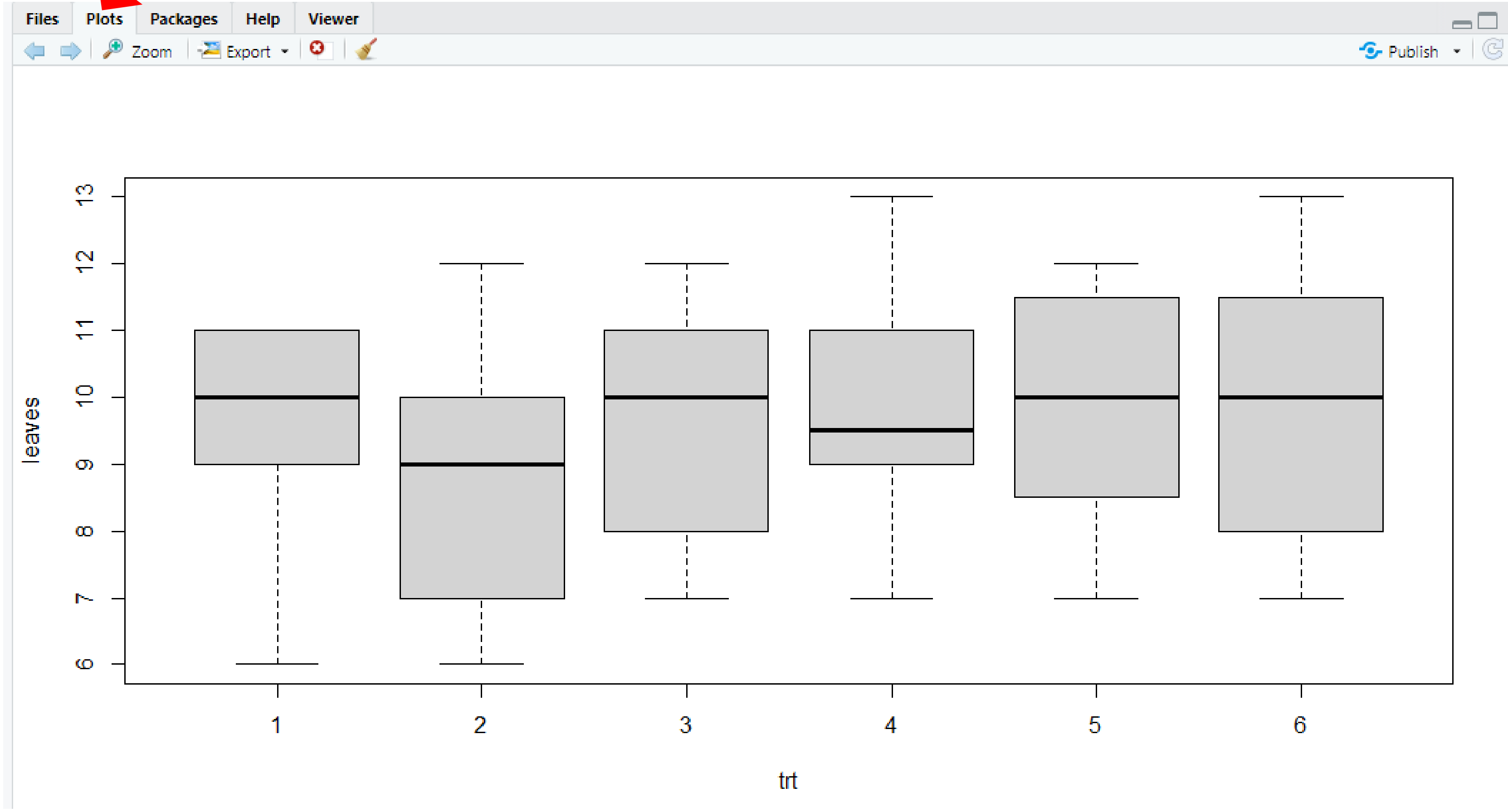
B<-boxplot(leaves ~ b, data = sa_data, col = "lightgray")

TRT<-boxplot(leaves ~ trt, data = sa_data, col = "lightgray")

BTRT<-boxplot(leaves ~ b*trt, data = sa_data, col = "lightgray")

TRTB<-boxplot(leaves ~ trt*b, data = sa_data, col = "lightgray")

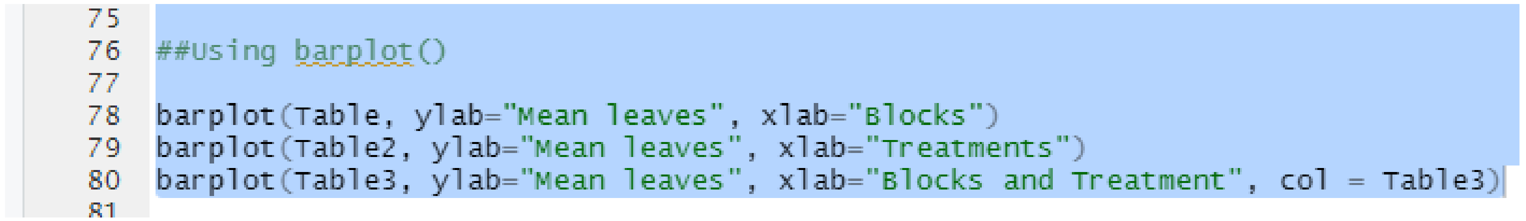
- Let's view it in the plot output:



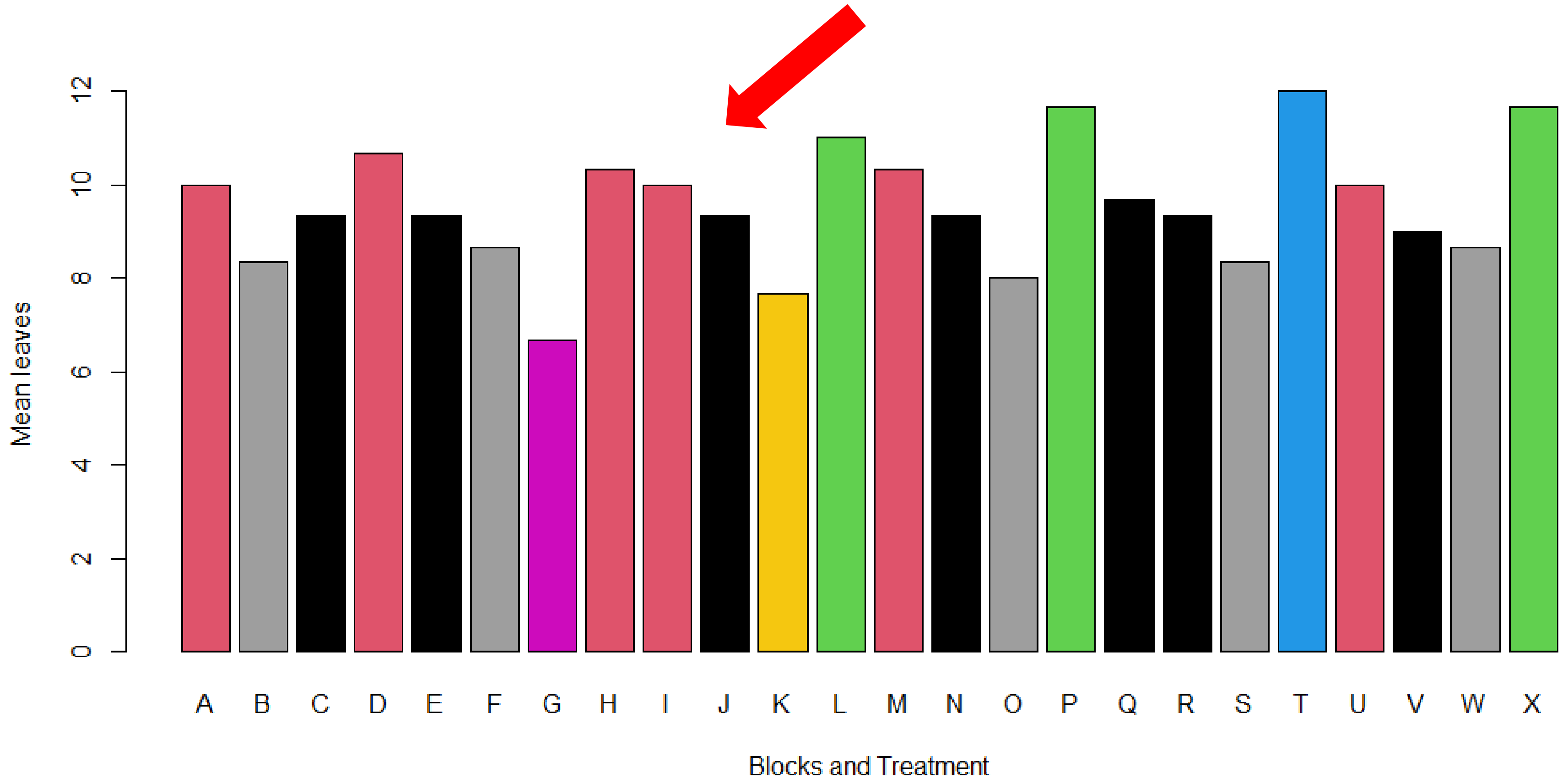
# Plotting data

- Plot barplots:

barplot(Table, ylab="Mean leaves", xlab="Blocks")
barplot(Table2, ylab="Mean leaves", xlab="Treatments")
barplot(Table3, ylab="Mean leaves", xlab="Blocks and Treatment", cc
= Table3)



```
##Using barplot()

barplot(Table, ylab="Mean leaves", xlab="Blocks")
barplot(Table2, ylab="Mean leaves", xlab="Treatments")
barplot(Table3, ylab="Mean leaves", xlab="Blocks and Treatment", col = Table3)
```



# Executing assumptions of normality: LMM

- We can execute our command for linear mixed models in two ways in R

```
model1 <- lm(leaves ~ b+trt,data = sa_data)
```

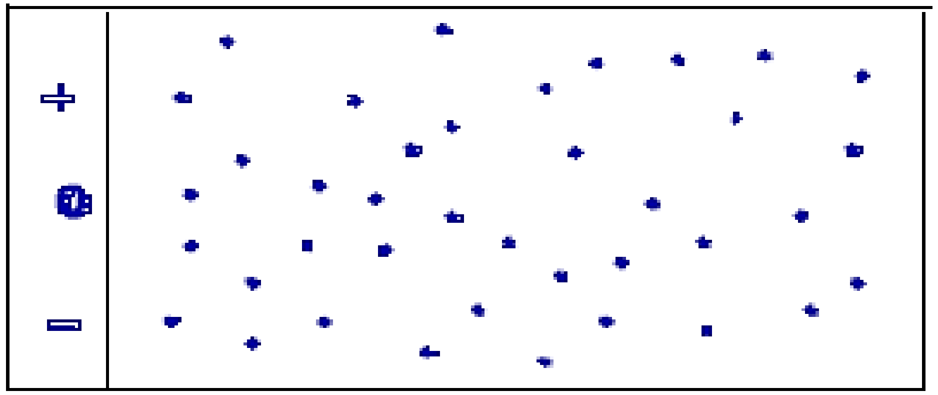
***OR***

```
model1 <- lm(data$Y~data$X)
```

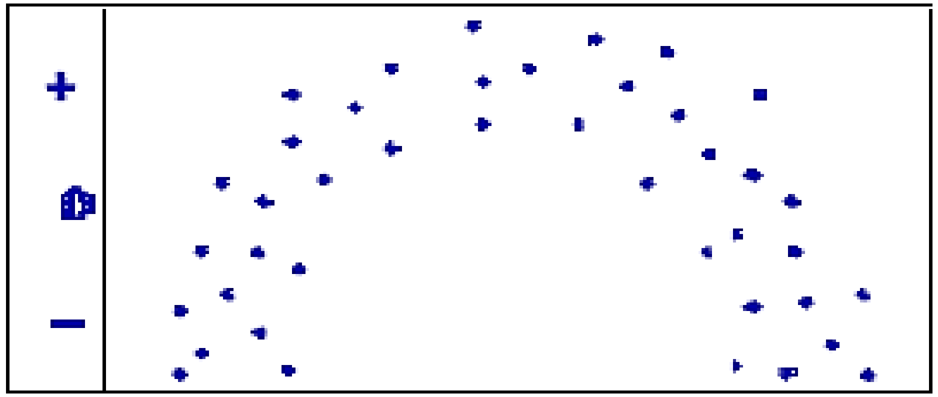
- From here we would look at the error structure using the following:

```
Res<-resid(model1)

Fit<-fitted(model1)

Plot(X,Y, xlab=”Fits”, ylab = “Residuals”, main = “Residual vs. Fits plot for normality”)

# Where X = Fit
# And Y = Res

abline(h=0, col="red")
#helps with visualization
```

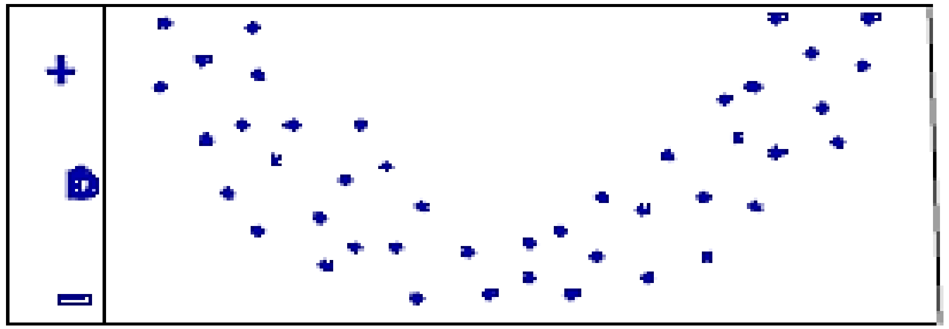
# Normal error model assumptions graphs



✓ACCEPTABLE



× NOT ACCEPTABLE

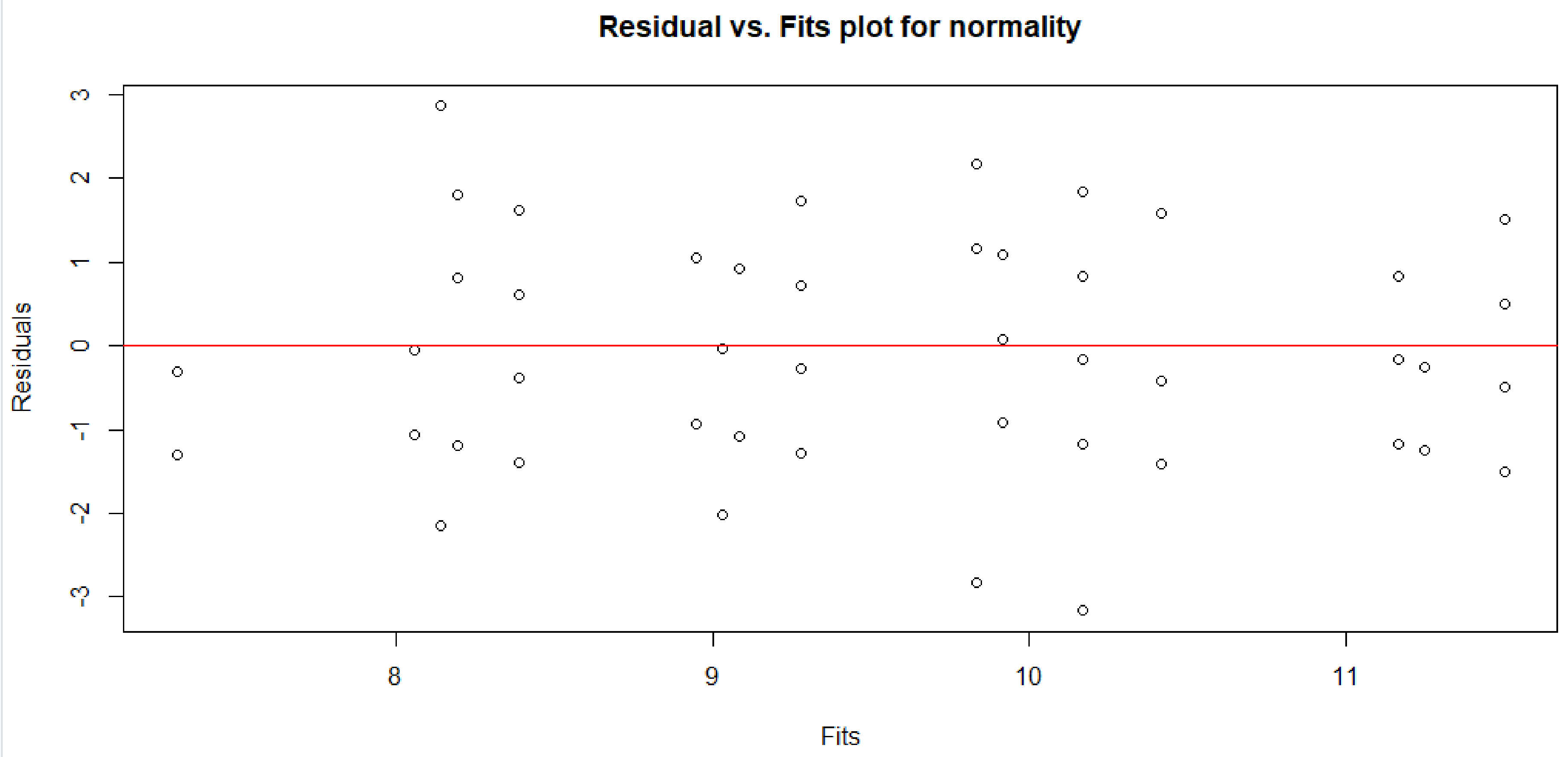


× NOT ACCEPTABLE

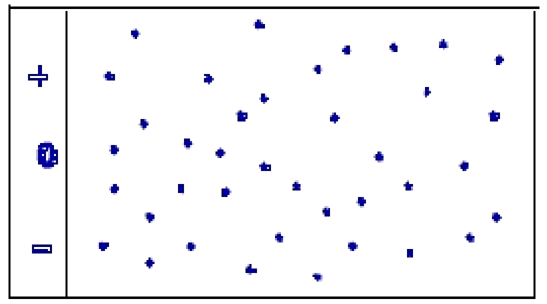
# Normal error model assumptions: LMM

## Homogeneous errors

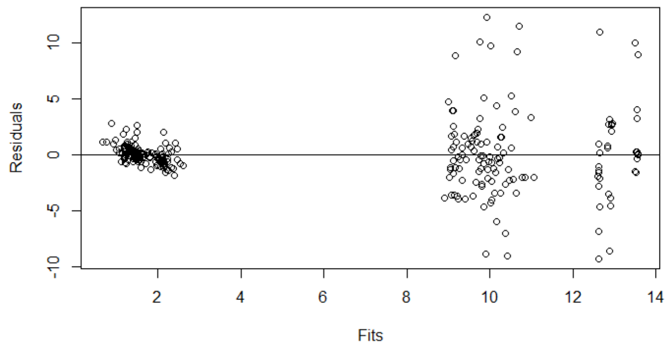
Use the plot we made



✓ACCEPTABLE



× NOT ACCEPTABLE

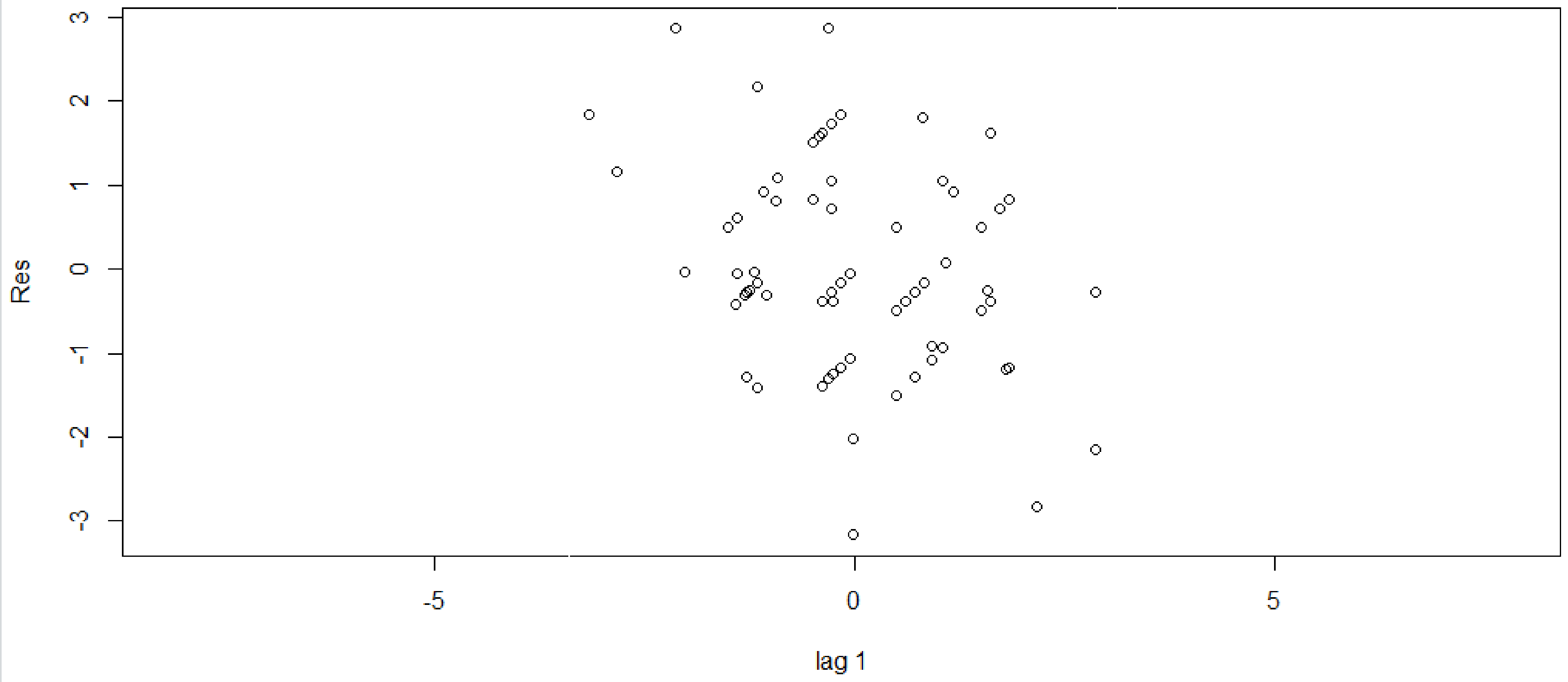


# Normal error assumptions: LMM

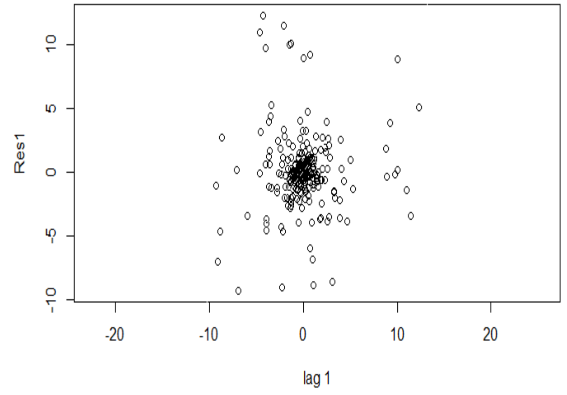
Independent or autocorrelation errors

## Plot a lag plot

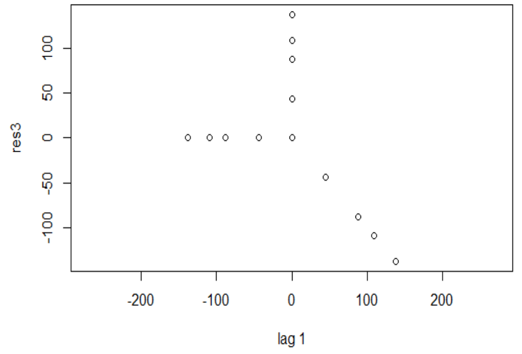
```
lag.plot(Res, do.lines=FALSE, diag.=FALSE)
```



✓ACCEPTABLE



× NOT ACCEPTABLE

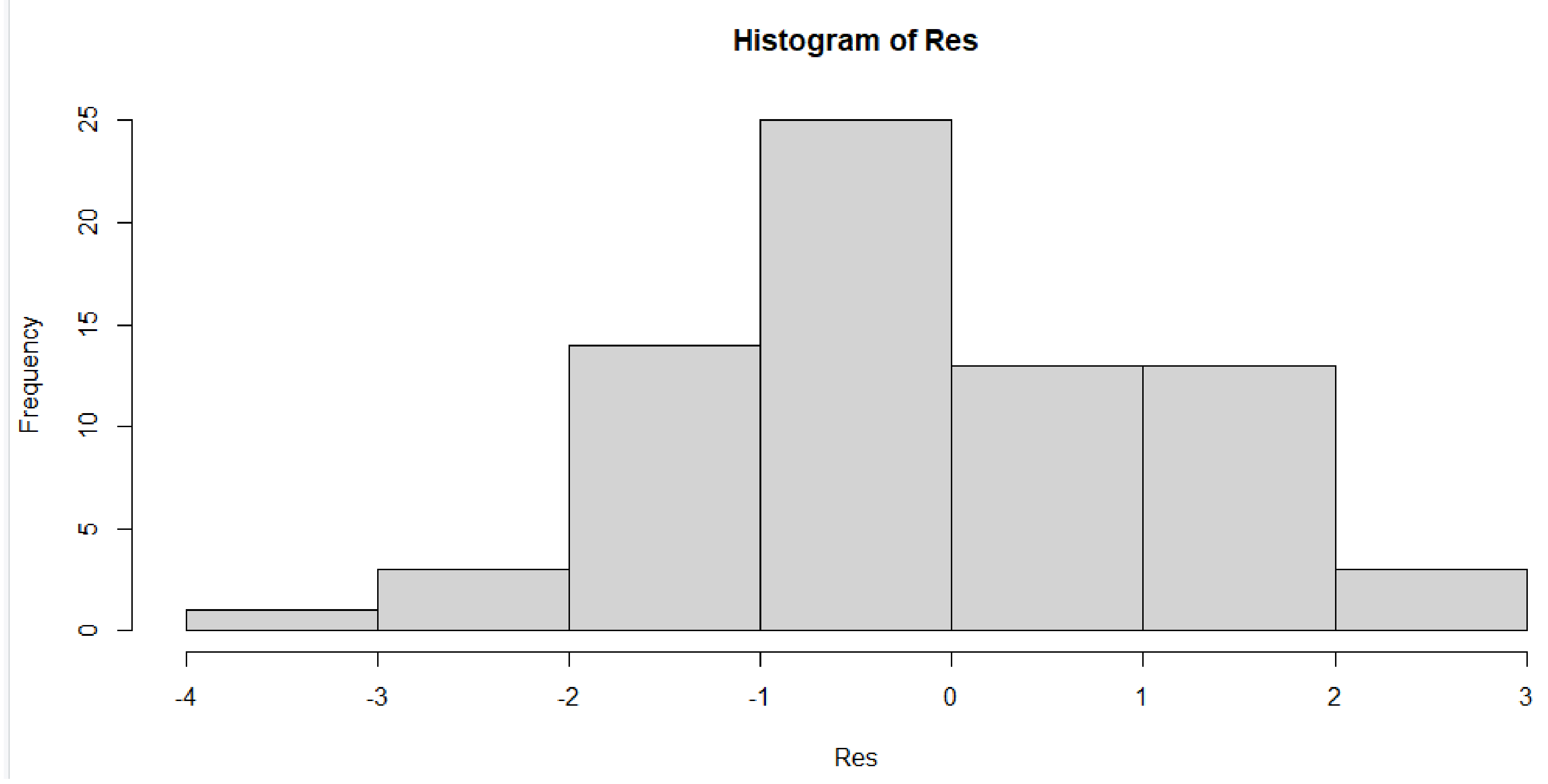


# Normal error model assumptions: LMM

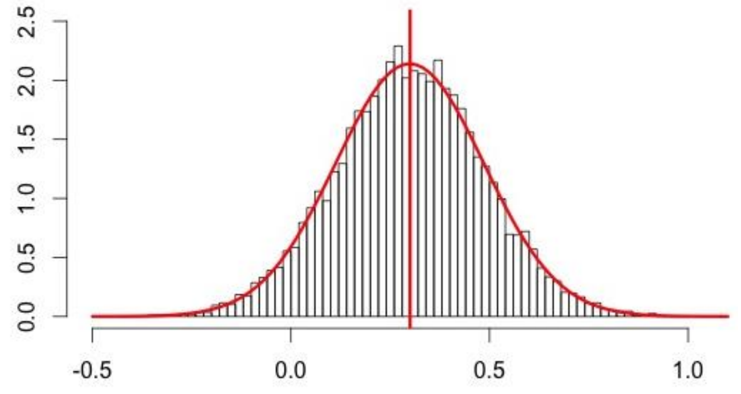
Normal errors

Plot a histogram:

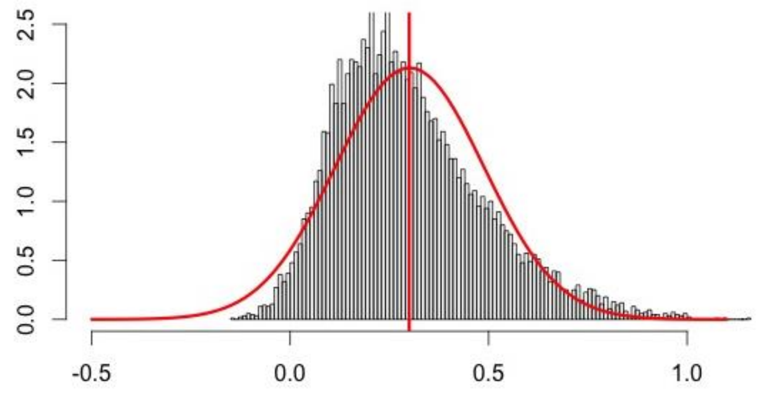
hist(Res)



✓ACCEPTABLE



× NOT ACCEPTABLE

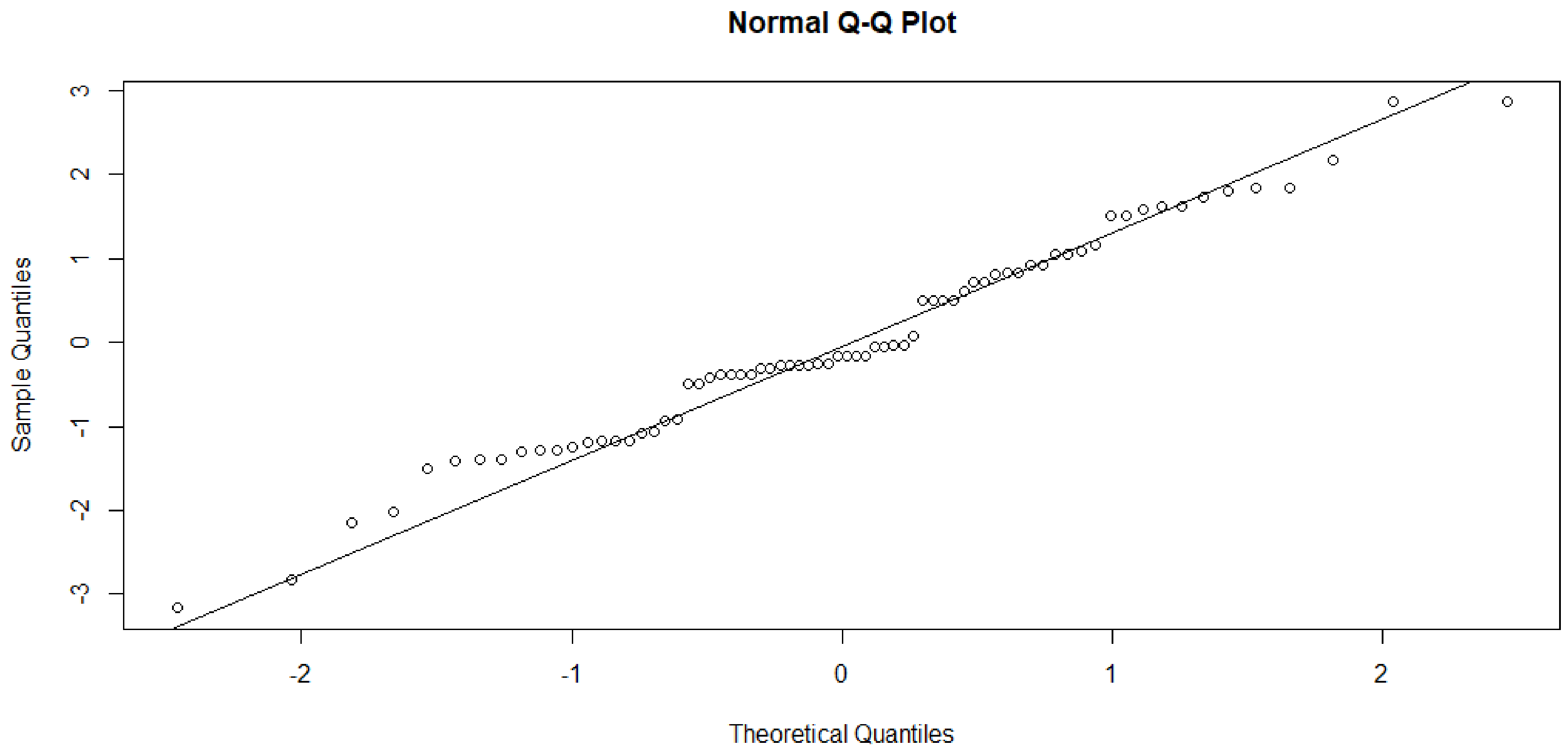


# Normal error model assumptions: LMM

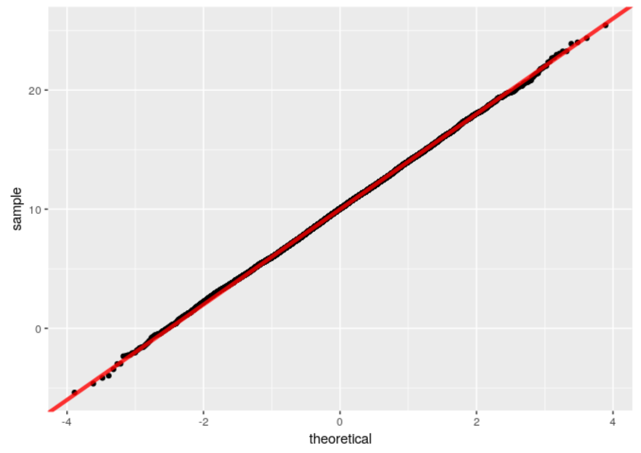
## Normal errors

Make a quantile-quantile normality plot

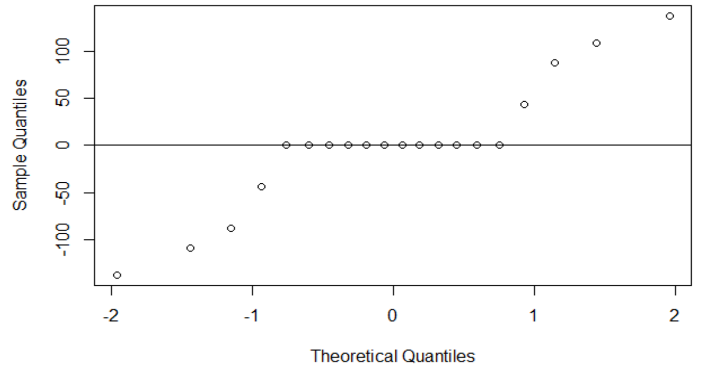
```
qqnorm(Res)
qqline(Res)
```



✓ACCEPTABLE



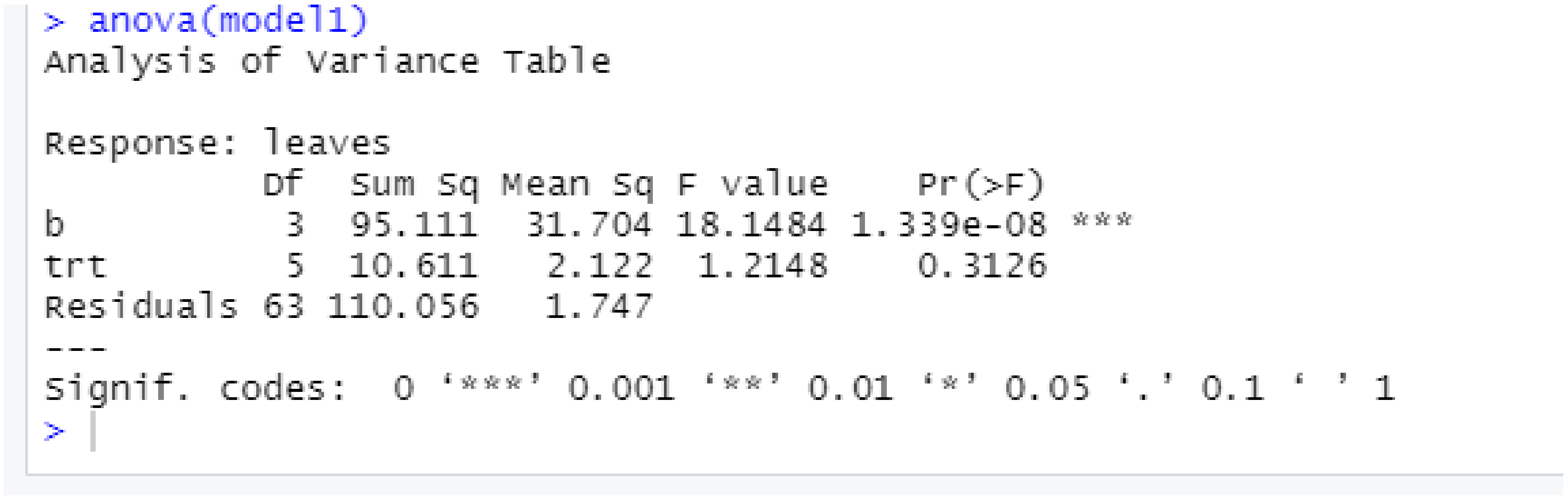
× NOT ACCEPTABLE



# LMM ANOVA output

Code and run the ANOVA function

anova(model4)

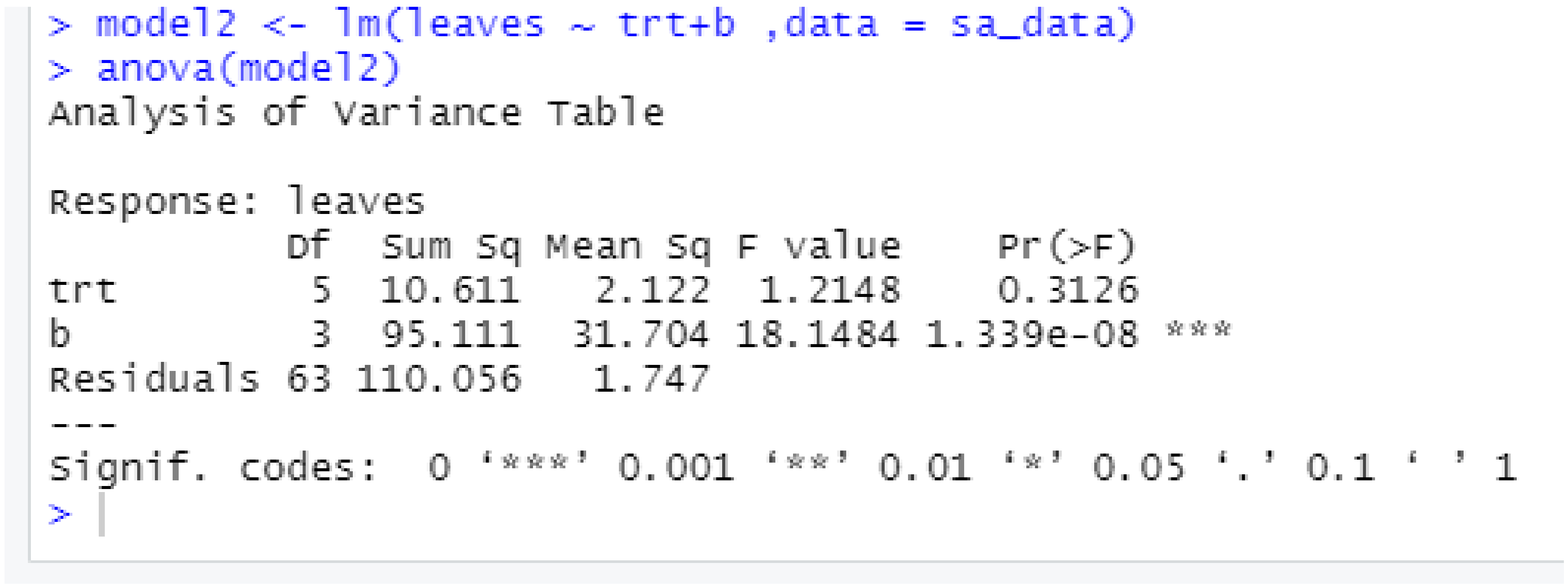


```
> anova(model1)
Analysis of Variance Table

Response: leaves
          Df  Sum Sq Mean Sq F value    Pr(>F)    
b          3  95.111  31.704 18.1484 1.339e-08 ***
trt        5  10.611   2.122  1.2148    0.3126    
Residuals 63 110.056   1.747                      
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> 
```

# LMM ANOVA output

Our blocks can affect the adjusted p-valye let’s check the model again with different ordering

model5 <- lm(leaves ~ trt+b ,data = sa_data)
anova(model5)

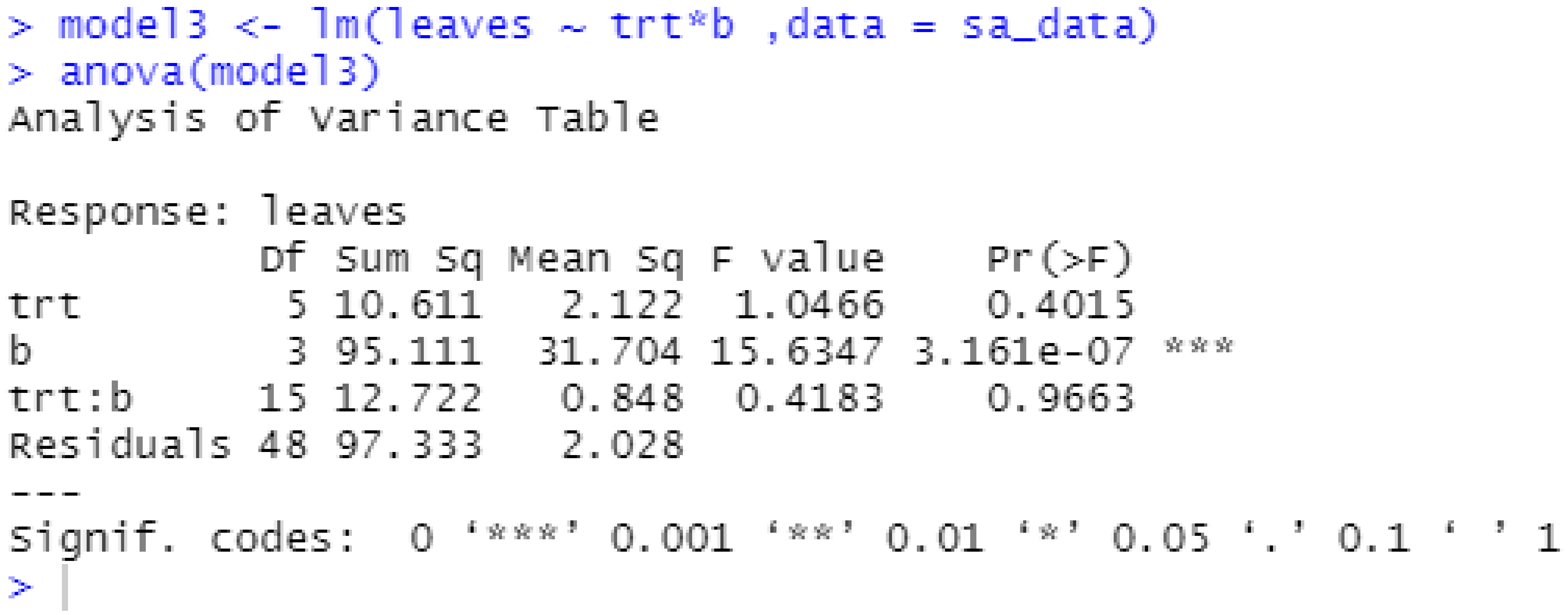


```
> model2 <- lm(leaves ~ trt+b ,data = sa_data)
> anova(model2)
Analysis of Variance Table

Response: leaves
          Df  Sum Sq Mean Sq F value    Pr(>F)    
trt        5  10.611   2.122  1.2148    0.3126    
b          3  95.111  31.704 18.1484 1.339e-08 ***
Residuals 63 110.056   1.747                      
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> 
```

# LMM ANOVA output

- The block and treatment interaction term can affect the ANOVA
- In this case, the interaction term has a very small weight, let’s check

model6 <- lm(leaves ~ trt*b ,data = sa_data)
anova(model6)

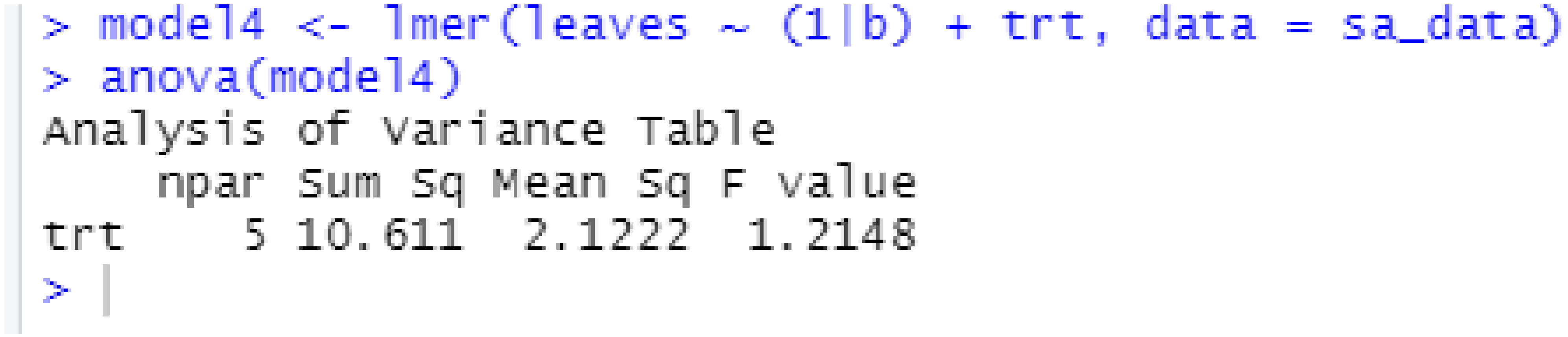


```
> model3 <- lm(leaves ~ trt*b ,data = sa_data)
> anova(model3)
Analysis of Variance Table

Response: leaves
          Df Sum Sq Mean Sq F value    Pr(>F)    
trt        5 10.611   2.122  1.0466    0.4015    
b          3 95.111  31.704 15.6347 3.161e-07 ***
trt:b     15 12.722   0.848  0.4183    0.9663    
Residuals 48 97.333   2.028                      
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> 
```

# LMM ANOVA output

## Let’s try another approach

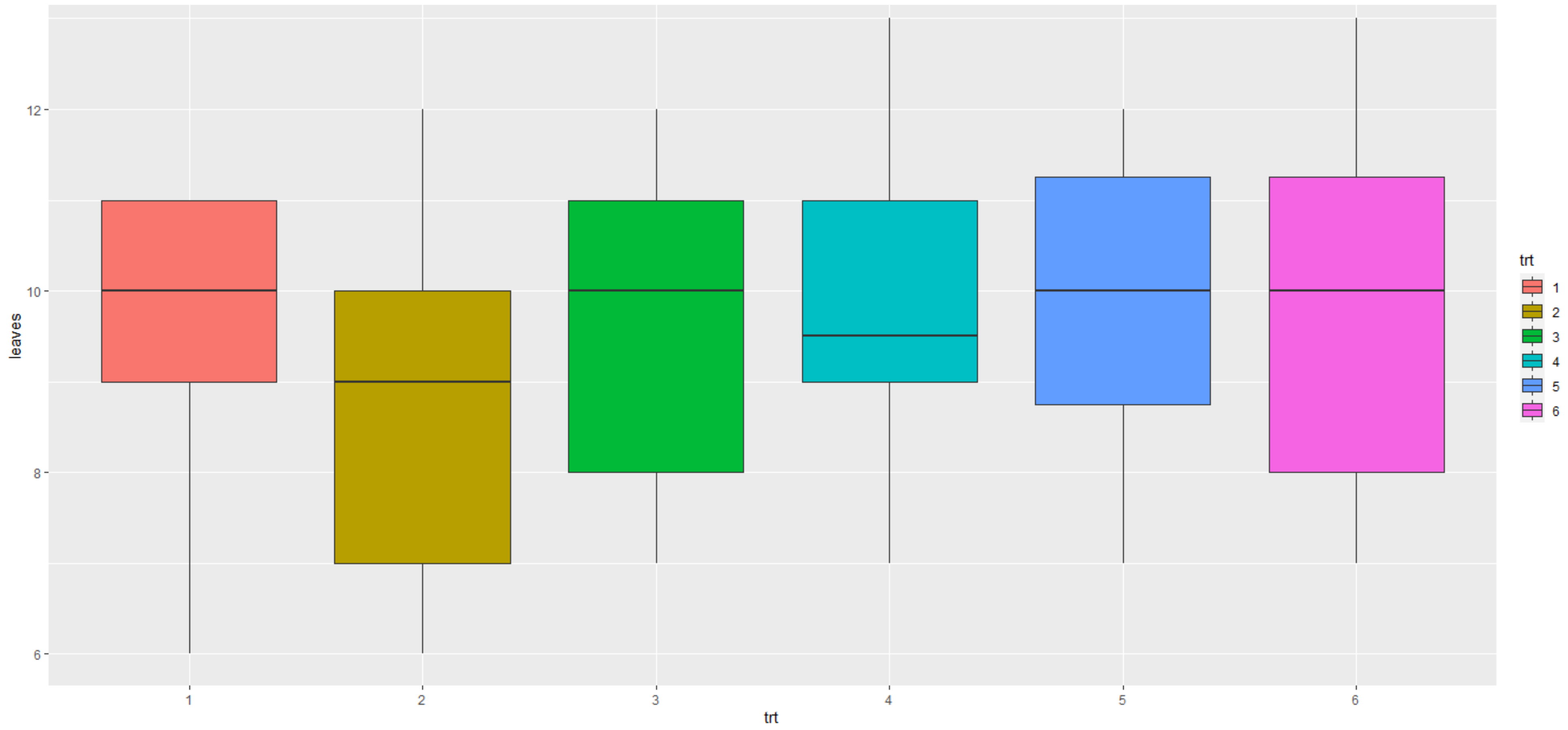
```
library(lme4)
model7 <- lmer(leaves ~ (1|b) + trt, data = sa_data)
anova(model7)
1-  pf(1.2148,5,63)
```



```
> model4 <- lmer(leaves ~ (1|b) + trt, data = sa_data)
> anova(model4)
Analysis of Variance Table
    npar Sum Sq Mean Sq F value
trt    5 10.611  2.1222  1.2148
> 
```

# Publication quality boxplots

```
library(ggplot2)
p1<-ggplot(sa_data, aes(x=trt, y=leaves, fill=trt)) + geom_boxplot()

p2<-ggplot(sa_data, aes(y=leaves, fill=b)) + geom_boxplot() + facet_wrap(~trt,ncol = 6)
+  theme( axis.text.x = element_blank(), axis.ticks.x = element_blank())
```


leaves
trt
trt