INTRODUCTION TO R

Nihan Acar-Denizli, Pau Fonseca

Installing R

- Visit one of the links below to download the latest version of R for your operating system:
  - MicrosoftWindows: http://cran.r-project.org/bin/windows/base/
  - MacOS: http://cran.r-project.org/bin/macosx/
  - Linux: http://cran.r-project.org/bin/linux/
- For windows also exist a portable solution:
  - http://sourceforge.net/projects/rportable/

Working with R

- **One line at a time,** this is the most basic method and is the first one that beginners will use.

- **Multiple lines at a time,** for longer programs (called scripts) there is too much code to write all at once at the command prompt.

Working with R

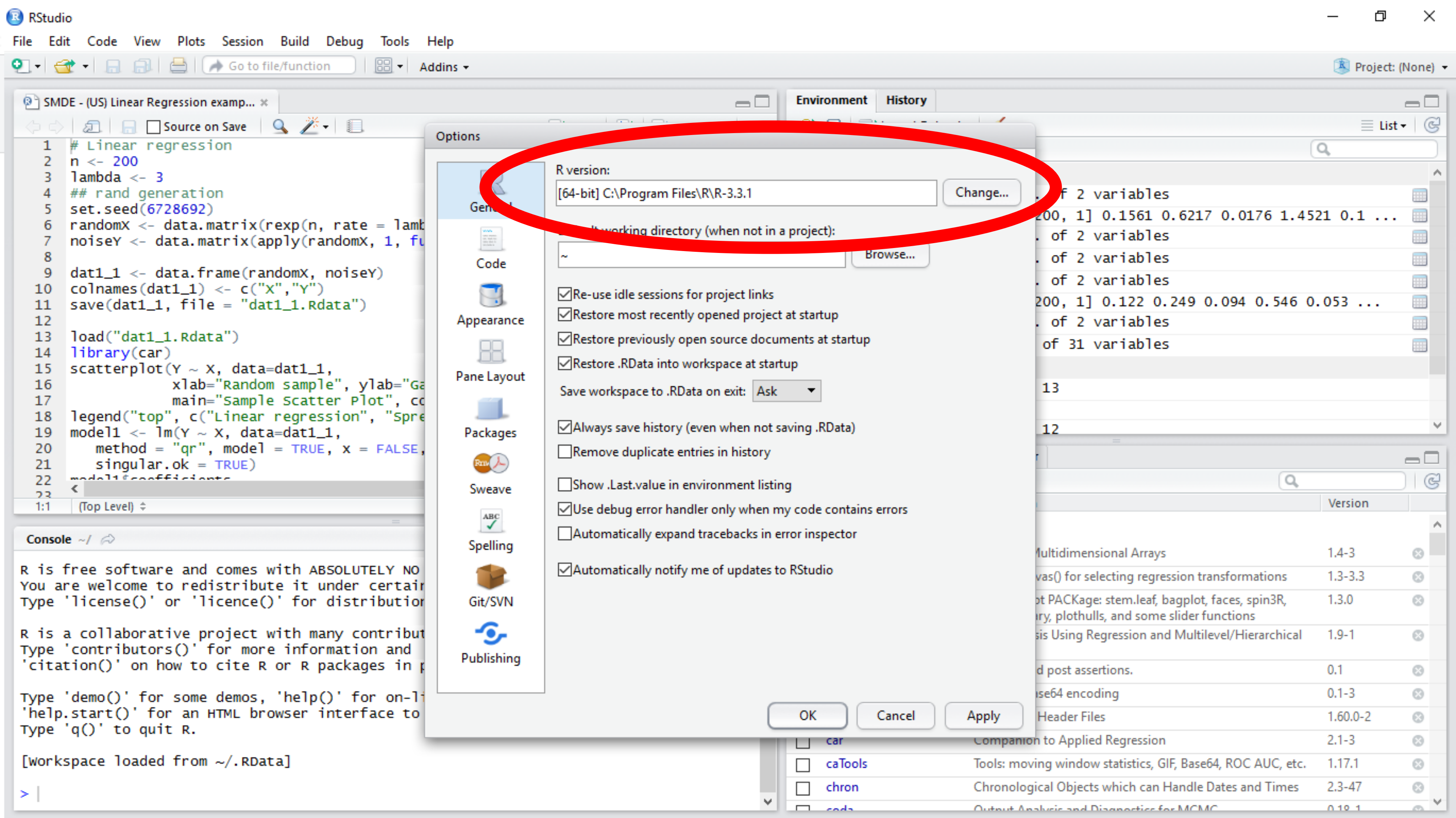
- **R Editor** (Windows): In Microsoft® Windows, RGui has its own built-in script editor, called R Editor.
- From the console window, select File → New Script.
- When satisfied with the code, the user highlights all of the commands and presses Ctrl+R. The commands are automatically run at once in R and the output is shown.

Working with R

- **R Commander** provides a point-and-click interface to many statistical tasks.
- Rcmdr also allows for user-contributed “Plugins” which are separate packages on CRAN that add extra functionality to the Rcmdr package. The plugins are typically named with the prefix RcmdrPlugin to make them easy to identify in the CRAN package list.

Working with RStudio

Working with different R versions



```
# Linear regression
n <- 200
lambda <- 3
## rand generation
set.seed(6728692)
randomX <- data.matrix(rexp(n, rate = lamb
noiseY <- data.matrix(apply(randomX, 1, fu

dat1_1 <- data.frame(randomX, noiseY)
colnames(dat1_1) <- c("X","Y")
save(dat1_1, file = "dat1_1.Rdata")

load("dat1_1.Rdata")
library(car)
scatterplot(Y ~ X, data=dat1_1,
            xlab="Random sample", ylab="Ga
            main="Sample Scatter Plot", co
legend("top", c("Linear regression", "Spre
model1 <- lm(Y ~ X, data=dat1_1,
   method = "qr", model = TRUE, x = FALSE,
   singular.ok = TRUE)
```

Options

R version:
[64-bit] C:\Program Files\R\R-3.3.1 — Change...

Default working directory (when not in a project):
~ — Browse...

- [x] Re-use idle sessions for project links
- [x] Restore most recently opened project at startup
- [x] Restore previously open source documents at startup
- [x] Restore .RData into workspace at startup

Save workspace to .RData on exit: Ask

- [x] Always save history (even when not saving .RData)
- [ ] Remove duplicate entries in history
- [ ] Show .Last.value in environment listing
- [x] Use debug error handler only when my code contains errors
- [ ] Automatically expand tracebacks in error inspector
- [x] Automatically notify me of updates to RStudio

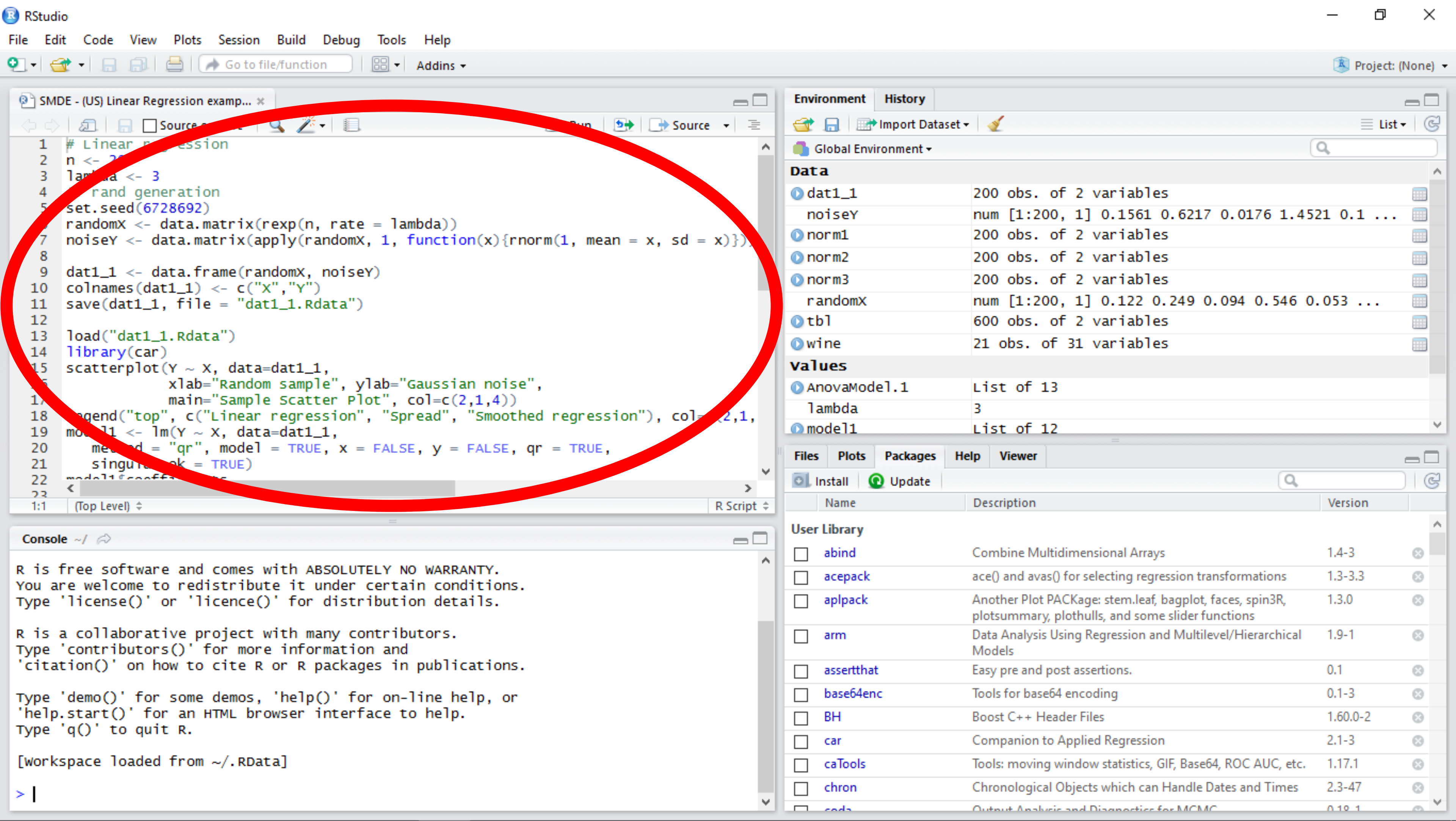
OK Cancel Apply

```
R is free software and comes with ABSOLUTELY NO
You are welcome to redistribute it under certai
Type 'license()' or 'licence()' for distributio

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

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

[Workspace loaded from ~/.RData]

>
```

```
. of 2 variables
200, 1] 0.1561 0.6217 0.0176 1.4521 0.1 ...
. of 2 variables
. of 2 variables
. of 2 variables
200, 1] 0.122 0.249 0.094 0.546 0.053 ...
. of 2 variables
of 31 variables
13
12
```

			Version
		ultidimensional Arrays	1.4-3
		vas() for selecting regression transformations	1.3-3.3
		ot PACKage: stem.leaf, bagplot, faces, spin3R, ry, plothulls, and some slider functions	1.3.0
		sis Using Regression and Multilevel/Hierarchical	1.9-1
		d post assertions.	0.1
		se64 encoding	0.1-3
		Header Files	1.60.0-2
☐	car	Companion to Applied Regression	2.1-3
☐	caTools	Tools: moving window statistics, GIF, Base64, ROC AUC, etc.	1.17.1
☐	chron	Chronological Objects which can Handle Dates and Times	2.3-47

Scripts



```
# Linear [illegible]ession
n <- 2[illegible]
la[illegible]da <- 3
[illegible] rand generation
set.seed(6728692)
randomX <- data.matrix(rexp(n, rate = lambda))
noiseY <- data.matrix(apply(randomX, 1, function(x){rnorm(1, mean = x, sd = x)})

dat1_1 <- data.frame(randomX, noiseY)
colnames(dat1_1) <- c("X","Y")
save(dat1_1, file = "dat1_1.Rdata")

load("dat1_1.Rdata")
library(car)
scatterplot(Y ~ X, data=dat1_1,
            xlab="Random sample", ylab="Gaussian noise",
            main="Sample Scatter Plot", col=c(2,1,4))
[illegible]gend("top", c("Linear regression", "Spread", "Smoothed regression"), col=[illegible]2,1,
mo[illegible]l1 <- lm(Y ~ X, data=dat1_1,
   me[illegible]d = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
   singu[illegible]ok = TRUE)
```

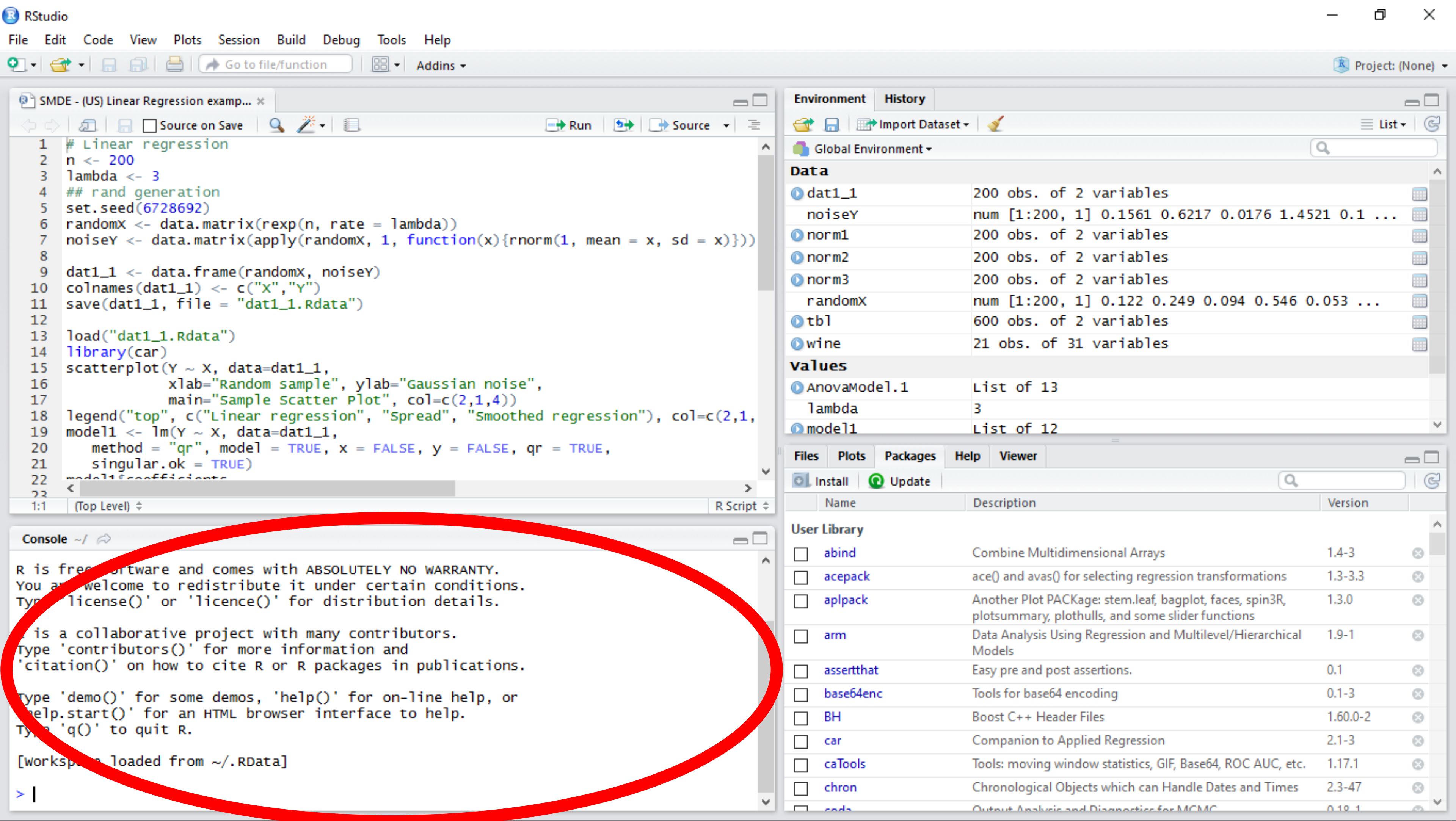
```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

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.

[Workspace loaded from ~/.RData]

> 
```

```
Data
dat1_1        200 obs. of 2 variables
noiseY        num [1:200, 1] 0.1561 0.6217 0.0176 1.4521 0.1 ...
norm1         200 obs. of 2 variables
norm2         200 obs. of 2 variables
norm3         200 obs. of 2 variables
randomX       num [1:200, 1] 0.122 0.249 0.094 0.546 0.053 ...
tbl           600 obs. of 2 variables
wine          21 obs. of 31 variables
Values
AnovaModel.1  List of 13
lambda        3
model1        List of 12
```

User Library

	Name	Description	Version
☐	abind	Combine Multidimensional Arrays	1.4-3
☐	acepack	ace() and avas() for selecting regression transformations	1.3-3.3
☐	aplpack	Another Plot PACKage: stem.leaf, bagplot, faces, spin3R, plotsummary, plothulls, and some slider functions	1.3.0
☐	arm	Data Analysis Using Regression and Multilevel/Hierarchical Models	1.9-1
☐	assertthat	Easy pre and post assertions.	0.1
☐	base64enc	Tools for base64 encoding	0.1-3
☐	BH	Boost C++ Header Files	1.60.0-2
☐	car	Companion to Applied Regression	2.1-3
☐	caTools	Tools: moving window statistics, GIF, Base64, ROC AUC, etc.	1.17.1
☐	chron	Chronological Objects which can Handle Dates and Times	2.3-47
☐	coda	Output Analysis and Diagnostics for MCMC	0.18-1

Console



```r
# Linear regression
n <- 200
lambda <- 3
## rand generation
set.seed(6728692)
randomX <- data.matrix(rexp(n, rate = lambda))
noiseY <- data.matrix(apply(randomX, 1, function(x){rnorm(1, mean = x, sd = x)}))

dat1_1 <- data.frame(randomX, noiseY)
colnames(dat1_1) <- c("X","Y")
save(dat1_1, file = "dat1_1.Rdata")

load("dat1_1.Rdata")
library(car)
scatterplot(Y ~ X, data=dat1_1,
            xlab="Random sample", ylab="Gaussian noise",
            main="Sample Scatter Plot", col=c(2,1,4))
legend("top", c("Linear regression", "Spread", "Smoothed regression"), col=c(2,1,
model1 <- lm(Y ~ X, data=dat1_1,
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
   singular.ok = TRUE)
model1$coefficients
```

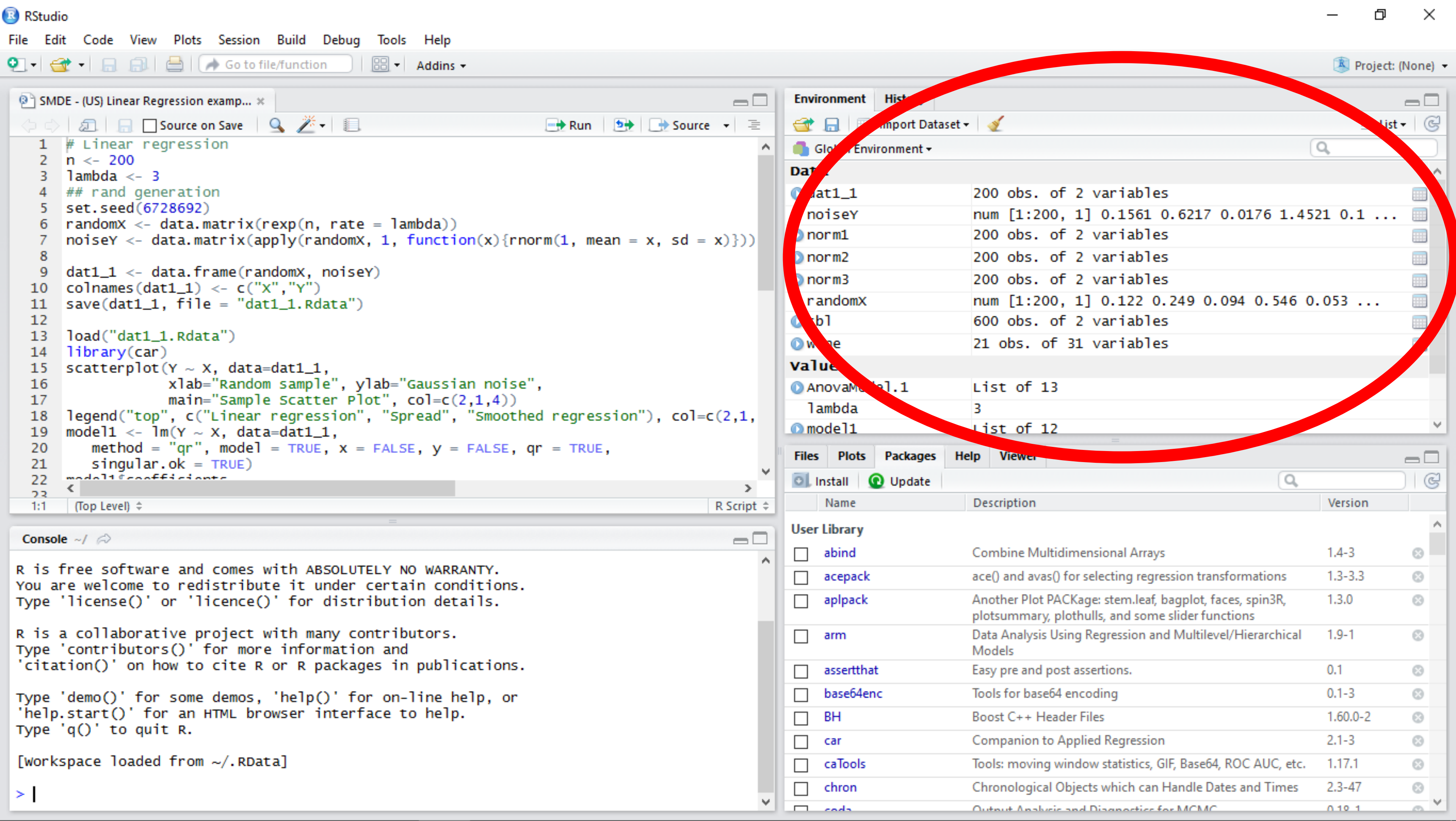
```
Data
dat1_1          200 obs. of 2 variables
noiseY          num [1:200, 1] 0.1561 0.6217 0.0176 1.4521 0.1 ...
norm1           200 obs. of 2 variables
norm2           200 obs. of 2 variables
norm3           200 obs. of 2 variables
randomX         num [1:200, 1] 0.122 0.249 0.094 0.546 0.053 ...
tbl             600 obs. of 2 variables
wine            21 obs. of 31 variables
Values
AnovaModel.1    List of 13
lambda          3
model1          List of 12
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

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.

[Workspace loaded from ~/.RData]

> |
```

User Library

Name	Description	Version
abind	Combine Multidimensional Arrays	1.4-3
acepack	ace() and avas() for selecting regression transformations	1.3-3.3
aplpack	Another Plot PACKage: stem.leaf, bagplot, faces, spin3R, plotsummary, plothulls, and some slider functions	1.3.0
arm	Data Analysis Using Regression and Multilevel/Hierarchical Models	1.9-1
assertthat	Easy pre and post assertions.	0.1
base64enc	Tools for base64 encoding	0.1-3
BH	Boost C++ Header Files	1.60.0-2
car	Companion to Applied Regression	2.1-3
caTools	Tools: moving window statistics, GIF, Base64, ROC AUC, etc.	1.17.1
chron	Chronological Objects which can Handle Dates and Times	2.3-47
coda	Output Analysis and Diagnostics for MCMC	0.19-1

Data



```r
# Linear regression
n <- 200
lambda <- 3
## rand generation
set.seed(6728692)
randomX <- data.matrix(rexp(n, rate = lambda))
noiseY <- data.matrix(apply(randomX, 1, function(x){rnorm(1, mean = x, sd = x)}))

dat1_1 <- data.frame(randomX, noiseY)
colnames(dat1_1) <- c("X","Y")
save(dat1_1, file = "dat1_1.Rdata")

load("dat1_1.Rdata")
library(car)
scatterplot(Y ~ X, data=dat1_1,
            xlab="Random sample", ylab="Gaussian noise",
            main="Sample Scatter Plot", col=c(2,1,4))
legend("top", c("Linear regression", "Spread", "Smoothed regression"), col=c(2,1,
model1 <- lm(Y ~ X, data=dat1_1,
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
   singular.ok = TRUE)
```

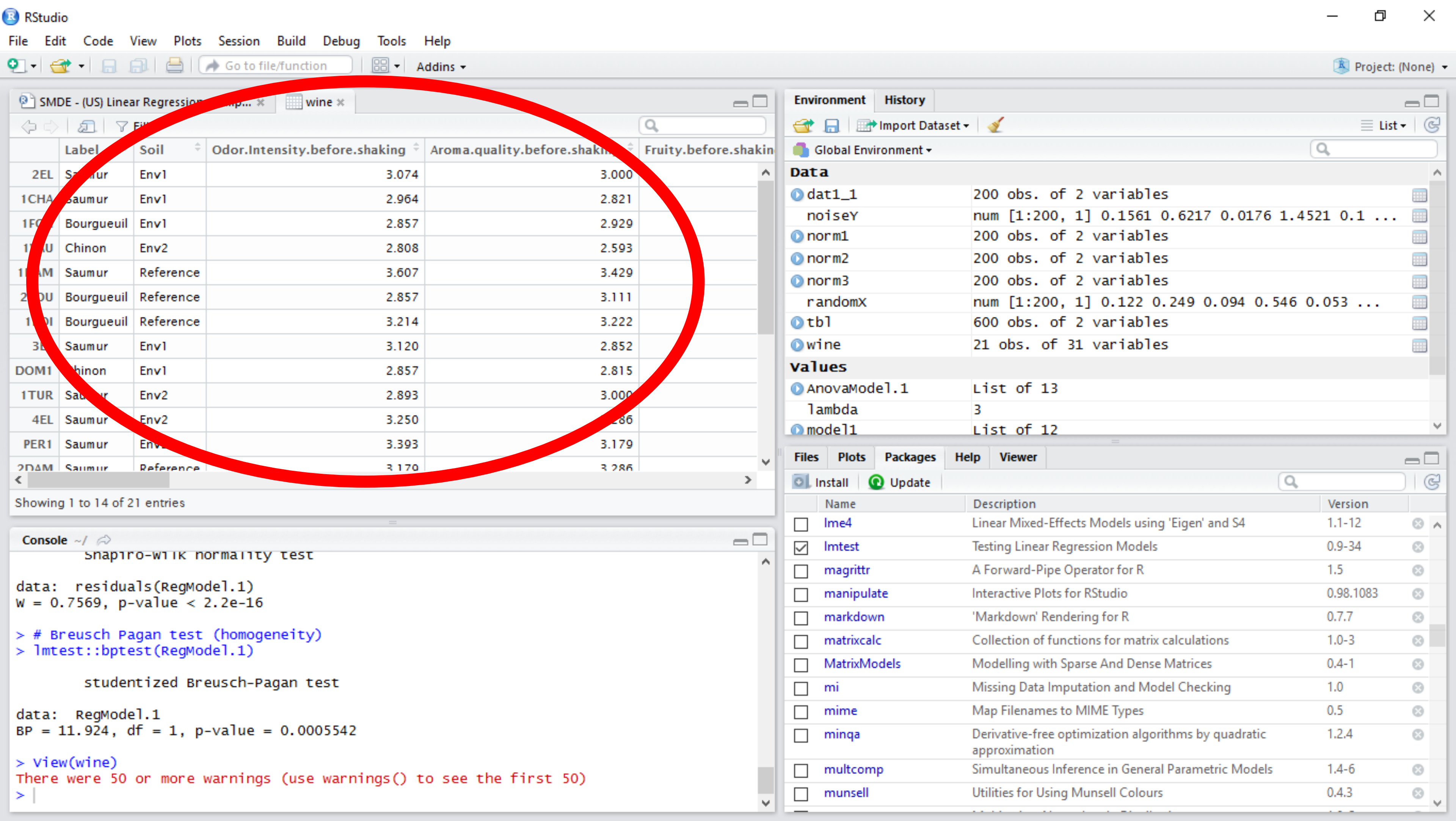
```
Dat
 at1_1        200 obs. of 2 variables
 noiseY       num [1:200, 1] 0.1561 0.6217 0.0176 1.4521 0.1 ...
 norm1        200 obs. of 2 variables
 norm2        200 obs. of 2 variables
 norm3        200 obs. of 2 variables
 randomX      num [1:200, 1] 0.122 0.249 0.094 0.546 0.053 ...
 bl           600 obs. of 2 variables
 w  e         21 obs. of 31 variables
Value
 AnovaM  el.1 List of 13
 lambda       3
 model1       List of 12
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

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.

[Workspace loaded from ~/.RData]

>
```

User Library

Name	Description	Version
abind	Combine Multidimensional Arrays	1.4-3
acepack	ace() and avas() for selecting regression transformations	1.3-3.3
aplpack	Another Plot PACKage: stem.leaf, bagplot, faces, spin3R, plotsummary, plothulls, and some slider functions	1.3.0
arm	Data Analysis Using Regression and Multilevel/Hierarchical Models	1.9-1
assertthat	Easy pre and post assertions.	0.1
base64enc	Tools for base64 encoding	0.1-3
BH	Boost C++ Header Files	1.60.0-2
car	Companion to Applied Regression	2.1-3
caTools	Tools: moving window statistics, GIF, Base64, ROC AUC, etc.	1.17.1
chron	Chronological Objects which can Handle Dates and Times	2.3-47

Data view



	Label	Soil	Odor.Intensity.before.shaking	Aroma.quality.before.shaki	Fruity.before.shakin
2EL	S...ur	Env1	3.074	3.000	
1CHA	...aumur	Env1	2.964	2.821	
1F...	Bourgueuil	Env1	2.857	2.929	
1...U	Chinon	Env2	2.808	2.593	
1...M	Saumur	Reference	3.607	3.429	
2...DU	Bourgueuil	Reference	2.857	3.111	
1...I	Bourgueuil	Reference	3.214	3.222	
3...	Saumur	Env1	3.120	2.852	
DOM1	...hinon	Env1	2.857	2.815	
1TUR	Sa...r	Env2	2.893	3.000	
4EL	Saumur	Env2	3.250	...86	
PER1	Saumur	En...	3.393	3.179	
2DAM	Saumur	Reference	3.179	3.286	

Showing 1 to 14 of 21 entries

Data

dat1_1	200 obs. of 2 variables
noiseY	num [1:200, 1] 0.1561 0.6217 0.0176 1.4521 0.1 ...
norm1	200 obs. of 2 variables
norm2	200 obs. of 2 variables
norm3	200 obs. of 2 variables
randomX	num [1:200, 1] 0.122 0.249 0.094 0.546 0.053 ...
tbl	600 obs. of 2 variables
wine	21 obs. of 31 variables

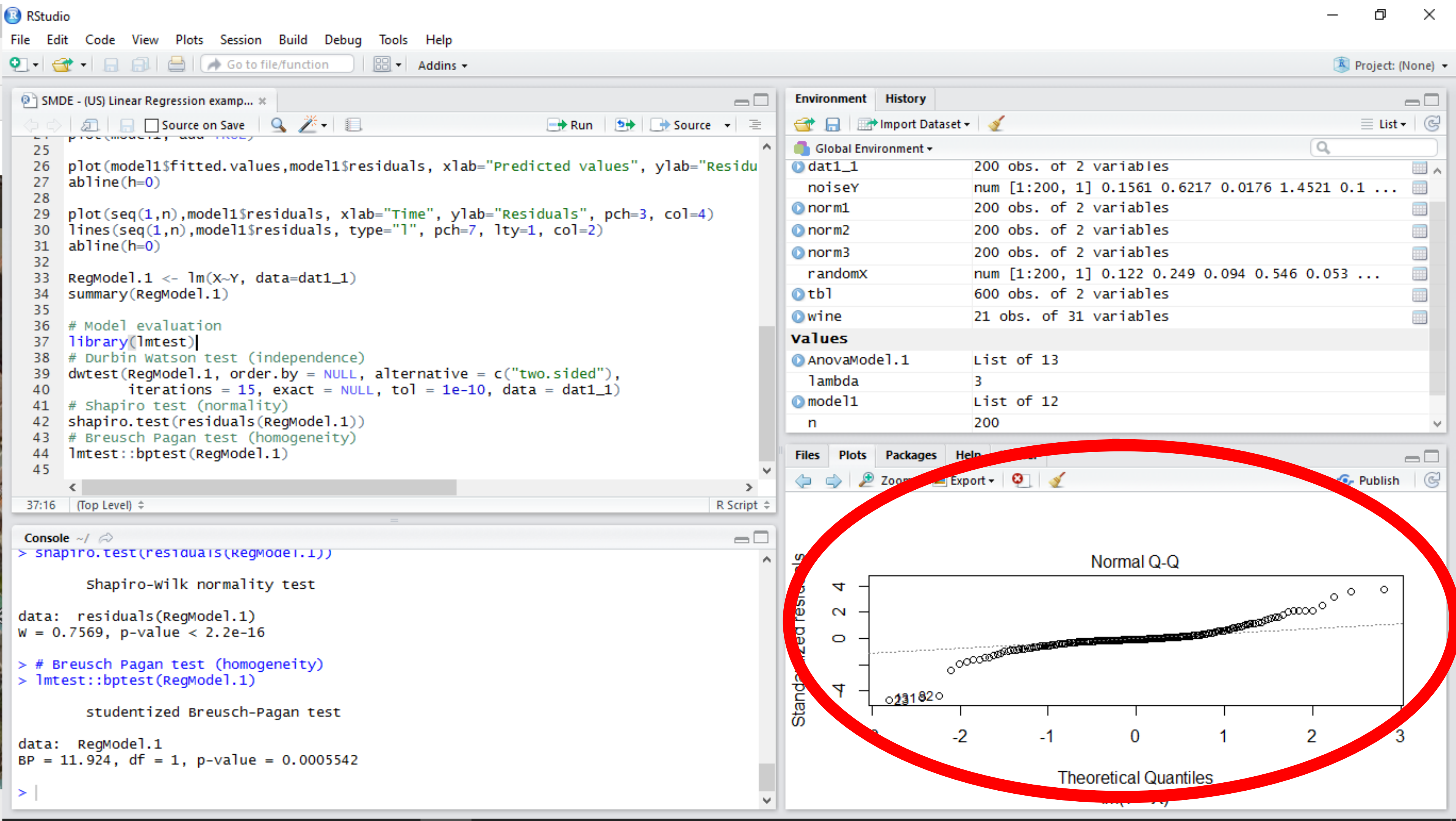
Values

AnovaModel.1	List of 13
lambda	3
model1	List of 12

```
        Shapiro-Wilk normality test

data:  residuals(RegModel.1)
W = 0.7569, p-value < 2.2e-16

> # Breusch Pagan test (homogeneity)
> lmtest::bptest(RegModel.1)

        studentized Breusch-Pagan test

data:  RegModel.1
BP = 11.924, df = 1, p-value = 0.0005542

> View(wine)
There were 50 or more warnings (use warnings() to see the first 50)
> 
```

	Name	Description	Version
☐	lme4	Linear Mixed-Effects Models using 'Eigen' and S4	1.1-12
☑	lmtest	Testing Linear Regression Models	0.9-34
☐	magrittr	A Forward-Pipe Operator for R	1.5
☐	manipulate	Interactive Plots for RStudio	0.98.1083
☐	markdown	'Markdown' Rendering for R	0.7.7
☐	matrixcalc	Collection of functions for matrix calculations	1.0-3
☐	MatrixModels	Modelling with Sparse And Dense Matrices	0.4-1
☐	mi	Missing Data Imputation and Model Checking	1.0
☐	mime	Map Filenames to MIME Types	0.5
☐	minqa	Derivative-free optimization algorithms by quadratic approximation	1.2.4
☐	multcomp	Simultaneous Inference in General Parametric Models	1.4-6
☐	munsell	Utilities for Using Munsell Colours	0.4.3

Plots

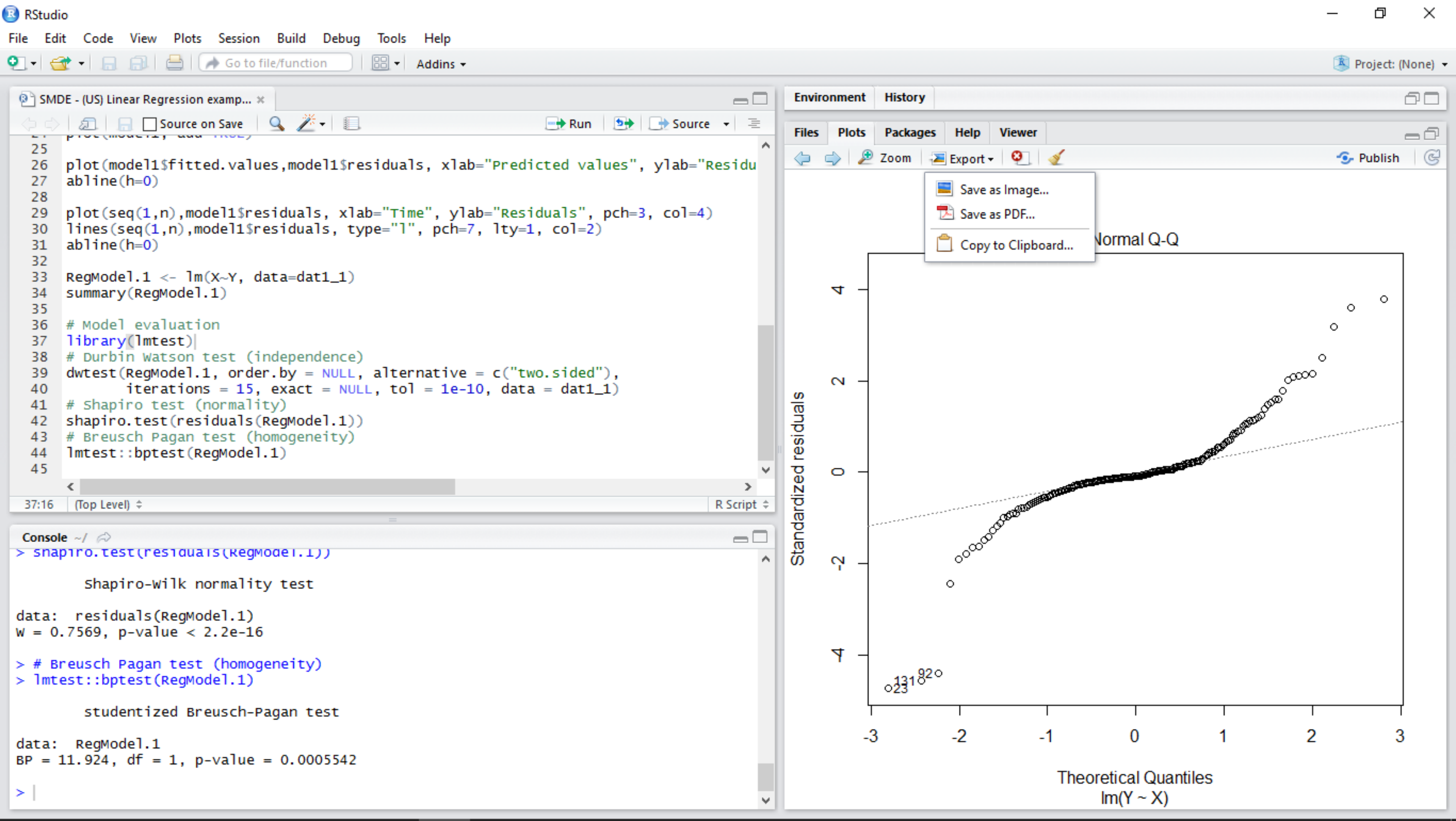


```r
plot(model1$fitted.values,model1$residuals, xlab="Predicted values", ylab="Residu
abline(h=0)

plot(seq(1,n),model1$residuals, xlab="Time", ylab="Residuals", pch=3, col=4)
lines(seq(1,n),model1$residuals, type="l", pch=7, lty=1, col=2)
abline(h=0)

RegModel.1 <- lm(X~Y, data=dat1_1)
summary(RegModel.1)

# Model evaluation
library(lmtest)
# Durbin Watson test (independence)
dwtest(RegModel.1, order.by = NULL, alternative = c("two.sided"),
       iterations = 15, exact = NULL, tol = 1e-10, data = dat1_1)
# Shapiro test (normality)
shapiro.test(residuals(RegModel.1))
# Breusch Pagan test (homogeneity)
lmtest::bptest(RegModel.1)
```

```
dat1_1        200 obs. of 2 variables
noiseY        num [1:200, 1] 0.1561 0.6217 0.0176 1.4521 0.1 ...
norm1         200 obs. of 2 variables
norm2         200 obs. of 2 variables
norm3         200 obs. of 2 variables
randomX       num [1:200, 1] 0.122 0.249 0.094 0.546 0.053 ...
tbl           600 obs. of 2 variables
wine          21 obs. of 31 variables
Values
AnovaModel.1  List of 13
lambda        3
model1        List of 12
n             200
```

```
> shapiro.test(residuals(RegModel.1))

        Shapiro-Wilk normality test

data:  residuals(RegModel.1)
W = 0.7569, p-value < 2.2e-16

> # Breusch Pagan test (homogeneity)
> lmtest::bptest(RegModel.1)

        studentized Breusch-Pagan test

data:  RegModel.1
BP = 11.924, df = 1, p-value = 0.0005542

>
```

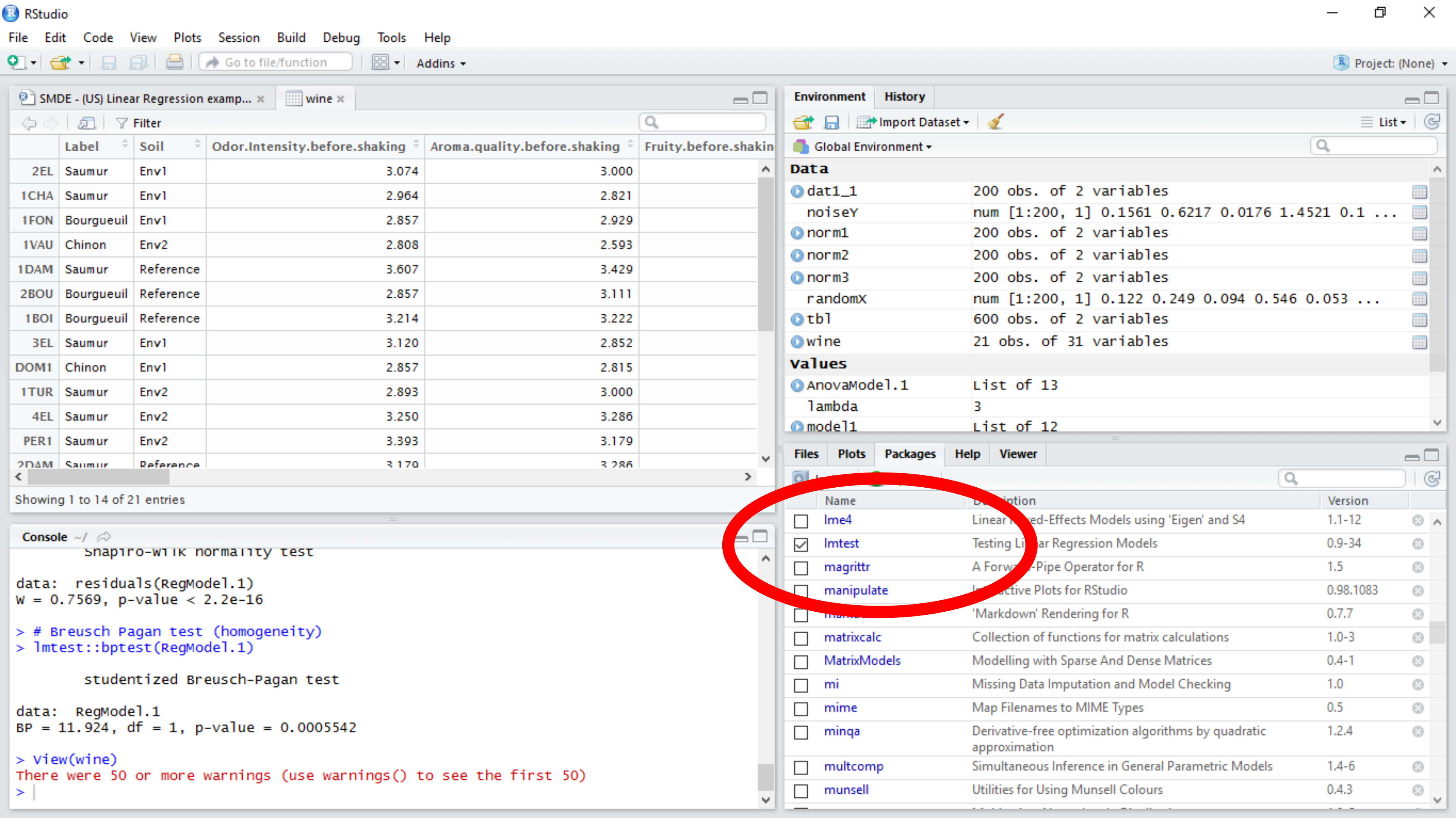
Plots



```
plot(model1$fitted.values,model1$residuals, xlab="Predicted values", ylab="Residu
abline(h=0)

plot(seq(1,n),model1$residuals, xlab="Time", ylab="Residuals", pch=3, col=4)
lines(seq(1,n),model1$residuals, type="l", pch=7, lty=1, col=2)
abline(h=0)

RegModel.1 <- lm(X~Y, data=dat1_1)
summary(RegModel.1)

# Model evaluation
library(lmtest)
# Durbin Watson test (independence)
dwtest(RegModel.1, order.by = NULL, alternative = c("two.sided"),
       iterations = 15, exact = NULL, tol = 1e-10, data = dat1_1)
# Shapiro test (normality)
shapiro.test(residuals(RegModel.1))
# Breusch Pagan test (homogeneity)
lmtest::bptest(RegModel.1)
```

```
> shapiro.test(residuals(RegModel.1))

	Shapiro-Wilk normality test

data:  residuals(RegModel.1)
W = 0.7569, p-value < 2.2e-16

> # Breusch Pagan test (homogeneity)
> lmtest::bptest(RegModel.1)

	studentized Breusch-Pagan test

data:  RegModel.1
BP = 11.924, df = 1, p-value = 0.0005542

> |
```

Packages



RStudio

File Edit Code View Plots Session Build Debug Tools Help

Go to file/function · Addins · Project: (None)

SMDE - (US) Linear Regression examp... · wine

Filter

	Label	Soil	Odor.Intensity.before.shaking	Aroma.quality.before.shaking	Fruity.before.shakin
2EL	Saumur	Env1	3.074	3.000	
1CHA	Saumur	Env1	2.964	2.821	
1FON	Bourgueuil	Env1	2.857	2.929	
1VAU	Chinon	Env2	2.808	2.593	
1DAM	Saumur	Reference	3.607	3.429	
2BOU	Bourgueuil	Reference	2.857	3.111	
1BOI	Bourgueuil	Reference	3.214	3.222	
3EL	Saumur	Env1	3.120	2.852	
DOM1	Chinon	Env1	2.857	2.815	
1TUR	Saumur	Env2	2.893	3.000	
4EL	Saumur	Env2	3.250	3.286	
PER1	Saumur	Env2	3.393	3.179	
2DAM	Saumur	Reference	3.179	3.286	

Showing 1 to 14 of 21 entries

Environment · History

Import Dataset · List

Global Environment

```
Data
dat1_1        200 obs. of 2 variables
noiseY        num [1:200, 1] 0.1561 0.6217 0.0176 1.4521 0.1 ...
norm1         200 obs. of 2 variables
norm2         200 obs. of 2 variables
norm3         200 obs. of 2 variables
randomX       num [1:200, 1] 0.122 0.249 0.094 0.546 0.053 ...
tbl           600 obs. of 2 variables
wine          21 obs. of 31 variables
Values
AnovaModel.1  List of 13
lambda        3
model1        List of 12
```

Console ~/

```
        Shapiro-Wilk normality test

data:  residuals(RegModel.1)
W = 0.7569, p-value < 2.2e-16

> # Breusch Pagan test (homogeneity)
> lmtest::bptest(RegModel.1)

        studentized Breusch-Pagan test

data:  RegModel.1
BP = 11.924, df = 1, p-value = 0.0005542

> View(wine)
There were 50 or more warnings (use warnings() to see the first 50)
>
```

Files · Plots · Packages · Help · Viewer

	Name	Description	Version
☐	lme4	Linear [illegible]ed-Effects Models using 'Eigen' and S4	1.1-12
☑	lmtest	Testing Li[illegible]ar Regression Models	0.9-34
☐	magrittr	A Forw[illegible]-Pipe Operator for R	1.5
☐	manipulate	[illegible]active Plots for RStudio	0.98.1083
☐	[illegible]	'Markdown' Rendering for R	0.7.7
☐	matrixcalc	Collection of functions for matrix calculations	1.0-3
☐	MatrixModels	Modelling with Sparse And Dense Matrices	0.4-1
☐	mi	Missing Data Imputation and Model Checking	1.0
☐	mime	Map Filenames to MIME Types	0.5
☐	minqa	Derivative-free optimization algorithms by quadratic approximation	1.2.4
☐	multcomp	Simultaneous Inference in General Parametric Models	1.4-6
☐	munsell	Utilities for Using Munsell Colours	0.4.3

Arithmetic

```
> 2 + 3 # add
        [1] 5

> 4 * 5 / 6 # multiply and divide
        [1] 3.333333

> 7^8 # 7 to the 8th power
        [1] 5764801
```

Arithmetic

```
> options(digits = 16)
> 10/3 # see more digits
        [1] 3.333333333333333
> sqrt(2) # square root
        [1] 1.414213562373095
> exp(1) # Euler's constant, e
        [1] 2.718281828459045
> pi
        [1] 3.141592653589793
> options(digits = 7) # back to default
```

Assignment, object names and data types

```
> x <- 7*41/pi # don't see the calculated value
> x # take a look
        [1] 91.35494
```

- Data types:
  - **integer**: the values 0, 1, 2, . . ; these are represented exactly by R.
  - **double**: real numbers (rational and irrational); these numbers are not represented exactly (save integers or fractions with a denominator that is a power of 2).
  - **character**: elements that are wrapped with pairs of " or ‘
  - **logical**: includes TRUE, FALSE, and NA (which are reserved words); the NA stands for “not available”, i.e., a missing value.

The type of an object

```
> sqrt(-1) # isn't defined
        [1] NaN
> sqrt(-1+0i) # is defined
        [1] 0+1i
> sqrt(as.complex(-1)) # same thing
        [1] 0+1i
> (0 + 1i)^2 # should be -1
        [1] -1+0i
> typeof((0 + 1i)^2)
        [1] "complex"
```

Vectors

```
> x <- c(74, 31, 95, 61, 76, 34, 23, 54, 96)
> x
      [1] 74 31 95 61 76 34 23 54 96
```

- The elements of a vector are usually coerced by R to the **most general type** of any of the elements, so if you do c(1, "2") then the result will be c("1", "2").
  - newXa <- sapply(xa, as.numeric)

Vectors

- Scan: x <- scan()
  - This method is useful when the data are stored somewhere else. For instance, you may type x <- scan() at the command prompt and R will display 1: to indicate that it is waiting for the first data value. Type a value and press Enter, at which point R will display 2:, and so forth.
  - Entering an empty line stops the scan. This method is especially handy when you have a column of values, say, stored in a text file or spreadsheet.
  - You may copy and paste them all at the 1: prompt, and R will store all of the values instantly in the vector x.

Vectors

- **repeated data:** regular patterns: the **seq** function will generate all sorts of sequences of numbers.
- It has the arguments from, to, by, and **length.out** which can be set in concert with one another. We will do a couple of examples to show you how it works.

```
> seq(from = 1, to = 5)
        [1] 1 2 3 4 5
> seq(from = 2, by = -0.1, length.out = 4)
        [1] 2.0 1.9 1.8 1.7
```

Vectors

- Indexing data vectors Sometimes we do not want the whole vector, but just a piece of it. We can access the intermediate parts with the [] operator.

```
> x <- c(74, 31, 95, 61, 76, 34, 23, 54, 96)
> x[1]
        [1] 74
> x[2:4]
        [1] 31 95 61
> x[c(1,3,4,8)]
        [1] 74 95 61 54
> x[-c(1,3,4,8)]
        [1] 31 76 34 23 96
```

Vectors

- Also, we can define a vectors of letters

```
> LETTERS[1:5]
        [1] "A" "B" "C" "D" "E«

> letters[-(6:24)]
        [1] "a" "b" "c" "d" "e" "y" "z"
```

Data frames

- x <- 5:8
- y <- letters[3:6]
- A <- data.frame(v1 = x, v2 = y)

- Is compatible with Rcommander.
- Notice that x and y are the same length.

Data frames

```
> A[3, ]
      v1 v2
      3  7  e
> A[1, ]
      v1 v2
      1  5  c
> A[, 2]
      [1] c d e f
      Levels: c d e f
```

Functions and expressions

```
> x <- 1:5
> sum(x)
        [1] 15
> length(x)
        [1] 5
> min(x)
        [1] 1
> mean(x) # sample mean
        [1] 3
> sd(x) # sample standard deviation
        [1] 1.581139
```

Functions and expressions

```
> intersect
function (x, y)
{
    y <- as.vector(y)
    unique(y[match(as.vector(x), y, 0L)])
}
<bytecode: 0x05e7432c>
<environment: namespace:base>
```

External resources

- The **R** Project for Statistical Computing: (http://www.r-project.org/)
- The Comprehensive **R** Archive Network: (http://cran.r-project.org/) This is where R is stored along with thousands of contributed packages. There are also loads of contributed information (books, tutorials, etc.). There are mirrors all over the world with duplicate information.
- **R**-Forge: (http://r-forge.r-project.org/) another location where R packages are stored. Here you can find development code which has not yet been released to CRAN.
- **R** Wiki: (http://wiki.r-project.org/rwiki/doku.php) many tips and tricks listed here. If you find a trick of your own, login and share it with the world.
- R Graph Gallery (http://addictedtor.free.fr/graphiques/) and R Graphical Manual (http://bm2.genes.nig.ac.jp/RGM2/index.php) have literally thousands of graphs to peruse.
- RSeek (http://www.rseek.org) is a search engine based on Google specifically tailored for R queries.

Displaying data

Charts and graphs for statistical data

Data

- **Quantitative:** data associated with a measurement of some quantity on an observational unit.
- **Qualitative:** data associated with some quality or property of the observational unit.
- **Logical:** data to represent true or false and which play an important role later.
- **Missing:** data that should be there but are not.

Quantitative data

- **Discrete** data take values in a finite or countable infinite set of numbers, that is, all possible values could (at least in principle) be written down in an ordered list.
  - Examples: counts, number of arrivals, or number of successes. They are often represented by integers, say, 0, 1, 2, etc..
- **Continuous** data take values in an interval of numbers. These are also known as scale data, interval data, or measurement data.
  - Examples: height, weight, length, time, etc. Continuous data are often characterized by fractions or decimals: 3.82, 7.0001, 4 5 8 , etc..

Dot plots (Strip charts)

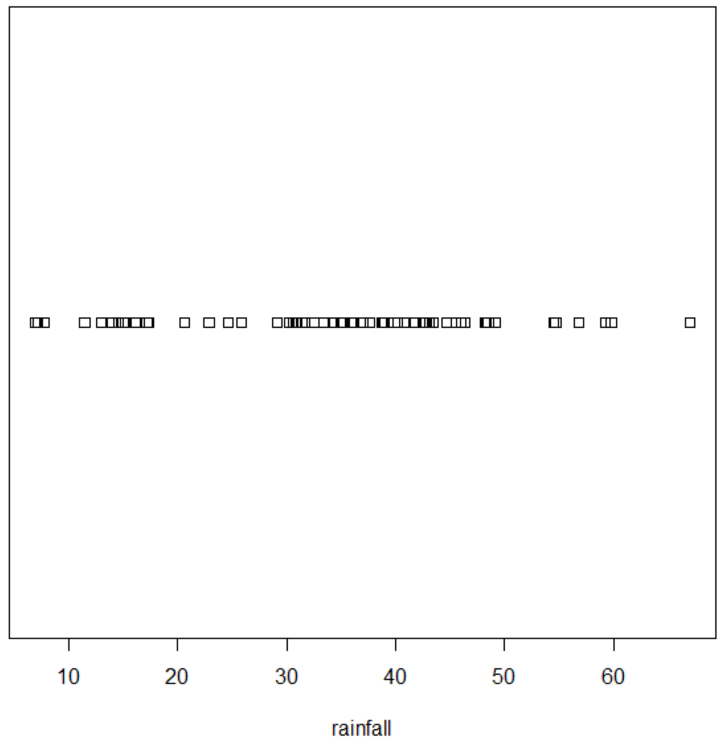
- These can be used for discrete or continuous data, and usually look best when the **data set is not too large.**

- Along the horizontal axis is a numerical scale above which the data values are plotted.

Dot plots (Strip charts)

- **overplot** plots ties covering each other. This method is good to display only the distinct values assumed by the data set.

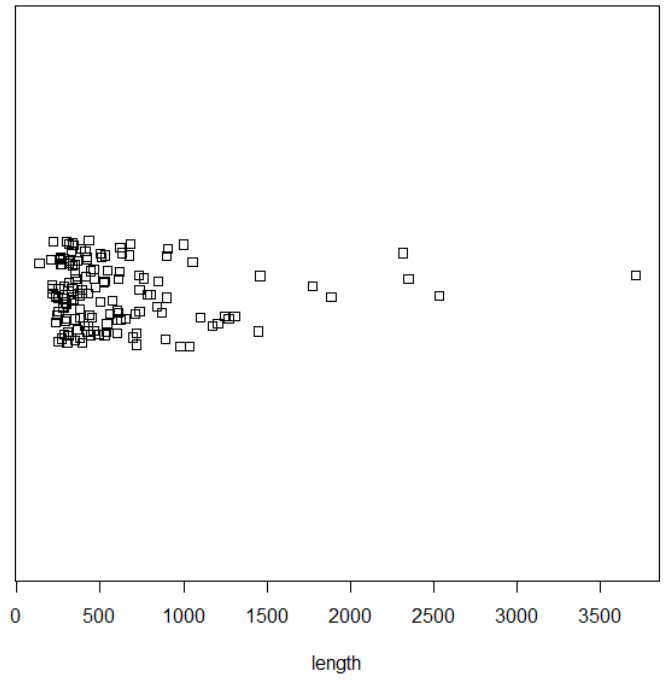
```
> stripchart(precip, xlab = "rainfall")
```



Dot plots (Strip charts)

- **jitter** adds some noise to the data in the y direction in which case the data values are not covered up by ties.

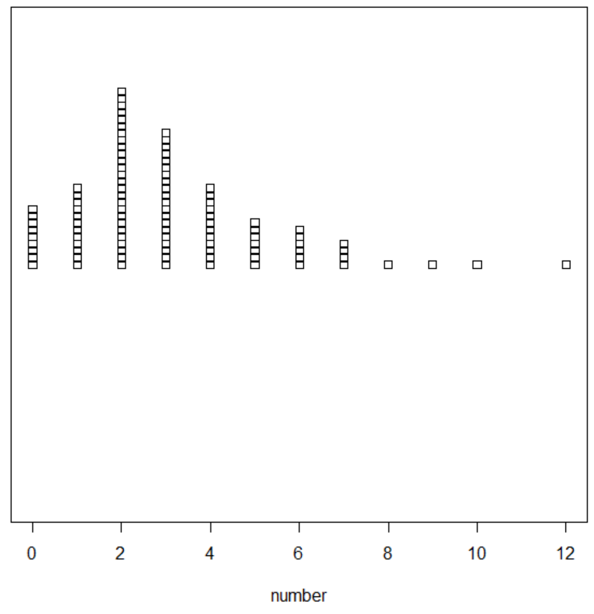
```
> stripchart(rivers, method = "jitter", xlab = "length")
```



Dot plots (Strip charts)

- **stack** plots repeated values stacked on top of one another. This method is best used for discrete data with a lot of ties; if there are no repeats then this method is identical to overplot.

```
> stripchart(discoveries, method = "stack", xlab = "number")
```

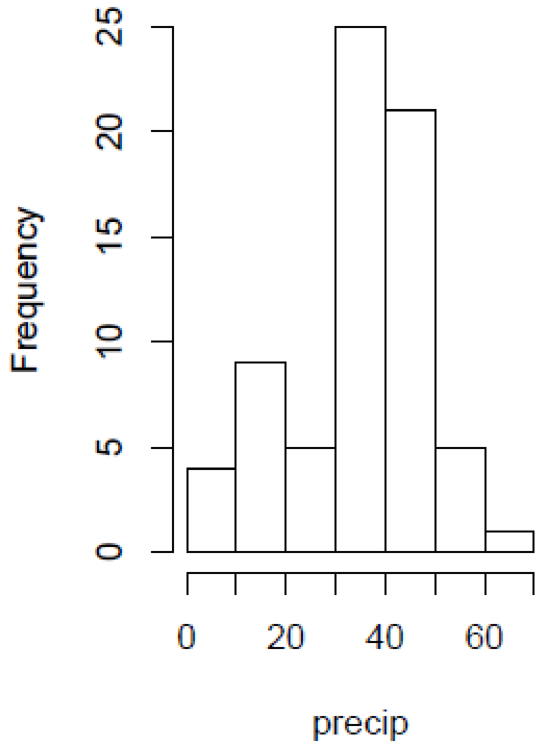


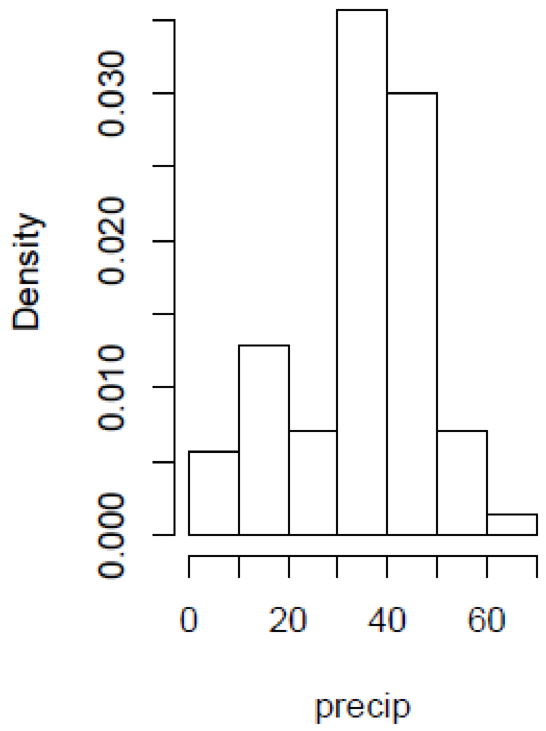
Histogram

- These are typically used for **continuous** data. A histogram is constructed by first deciding on a set of classes, or bins, which partition the real line into a set of boxes into which the data values fall. Then vertical bars are drawn over the bins with height proportional to the number of observations that fell into the bin.
- These are one of the most common summary displays, and they are often misidentified as “Bar Graphs”. The scale on the y axis can be frequency, percentage, or density (relative frequency). The term histogram was coined by Karl Pearson in 1891.

Histograms

```
> hist(precip, main = "")
> hist(precip, freq = FALSE, main = "")
```

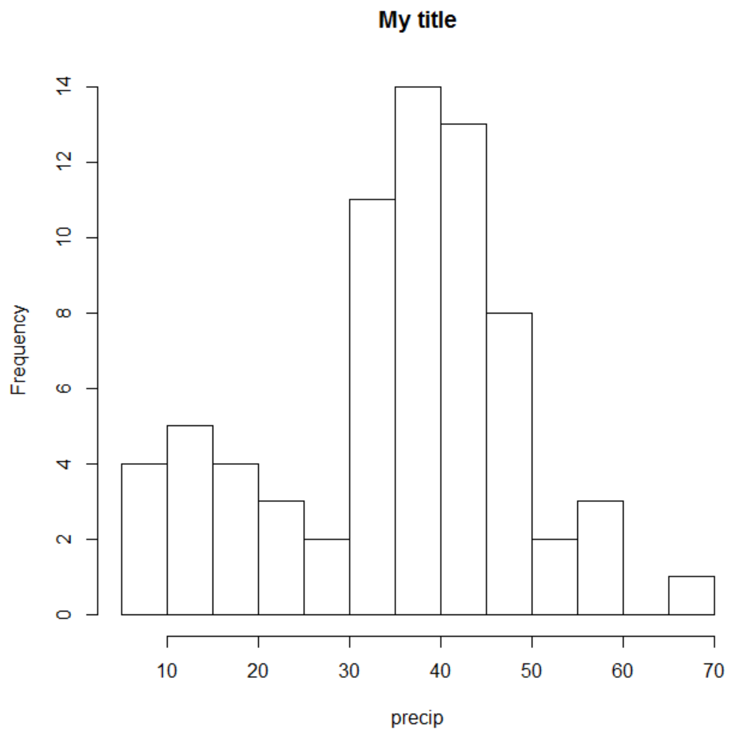


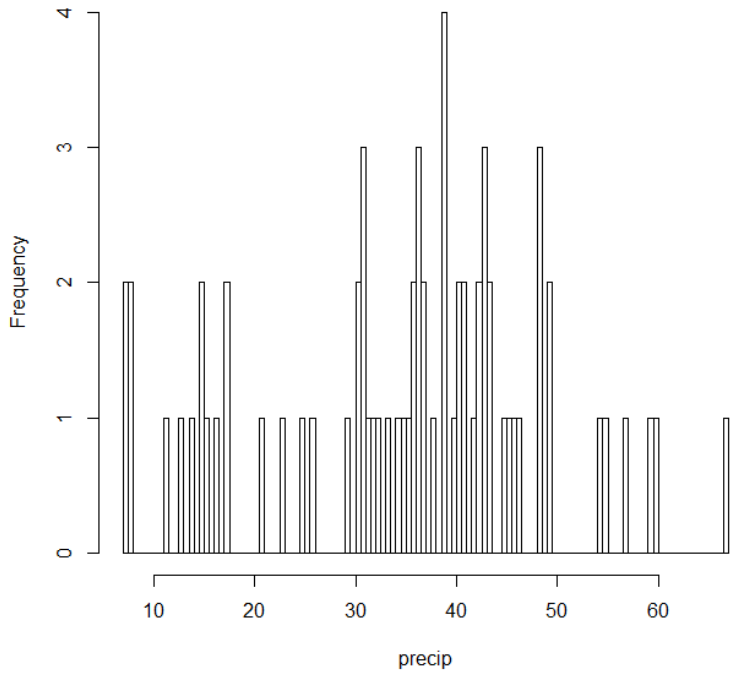


Histograms

- Selection of bins.

```
> hist(precip, breaks = 10, main = “My title")
> hist(precip, breaks = 200, main = "")
```





Stemplots

- Two basic parts: stems and leaves.
- The final digit of the data values is taken to be a leaf, and the leading digit(s) is (are) taken to be stems. We draw a vertical line, and to the left of the line we list the stems. To the right of the line, we list the leaves beside their corresponding stem.
- There will typically be several leaves for each stem, in which case the leaves accumulate to the right. It is sometimes necessary to round the data values, especially for **larger data sets**.

Stemplots

```
> library(aplpack)

> stem.leaf(UKDriverDeaths, depth = FALSE)

> UKDriverDeaths

      Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
1969 1687 1508 1507 1385 1632 1511 1559 1630 1579 1653 2152 2148
1970 1752 1765 1717 1558 1575 1520 1805 1800 1719 2008 2242 2478
1971 2030 1655 1693 1623 1805 1746 1795 1926 1619 1992 2233 2192
1972 2080 1768 1835 1569 1976 1853 1965 1689 1778 1976 2397 2654
1973 2097 1963 1677 1941 2003 1813 2012 1912 2084 2080 2118 2150
1974 1608 1503 1548 1382 1731 1798 1779 1887 2004 2077 2092 2051
1975 1577 1356 1652 1382 1519 1421 1442 1543 1656 1561 1905 2199
1976 1473 1655 1407 1395 1530 1309 1526 1327 1627 1748 1958 2274
1977 1648 1401 1411 1403 1394 1520 1528 1643 1515 1685 2000 2215
1978 1956 1462 1563 1459 1446 1622 1657 1638 1643 1683 2050 2262
1979 1813 1445 1762 1461 1556 1431 1427 1554 1645 1653 2016 2207
1980 1665 1361 1506 1360 1453 1522 1460 1552 1548 1827 1737 1941
1981 1474 1458 1542 1404 1522 1385 1641 1510 1681 1938 1868 1726
1982 1456 1445 1456 1365 1487 1558 1488 1684 1594 1850 1998 2079
1983 1494 1057 1218 1168 1236 1076 1174 1139 1427 1487 1483 1513
1984 1357 1165 1282 1110 1297 1185 1222 1284 1444 1575 1737 1763
```

```
1 | 2: represents 120
leaf unit: 10
n: 192

10 | 57
11 | 136678
12 | 123889
13 | 0255666888899
14 | 000012223444445555566677788889
15 | 0000111112222223444455555566677779
16 | 012223334444455555678888889
17 | 11233344566667799
18 | 00011235568
19 | 01234455667799
20 | 000011355778899
21 | 145599
22 | 013467
23 | 9
24 | 7
HI: 2654
```

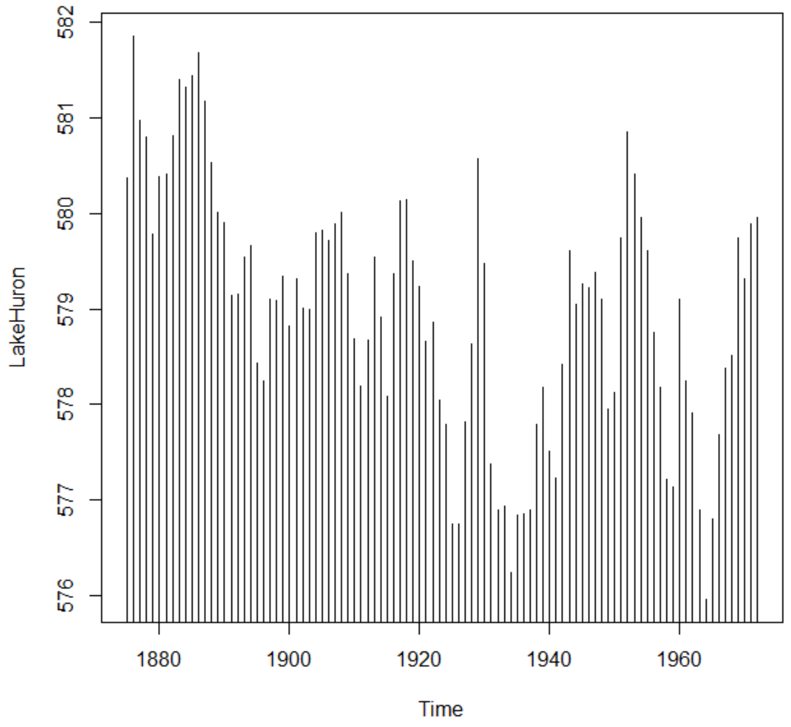
Index plot

- These are good for plotting **data which are ordered**, for example, when the data are measured over time. That is, the first observation was measured at time 1, the second at time 2, etc. It is a two dimensional plot, in which the index (or time) is the x variable and the measured value is the y variable.

Index plot

- spikes: draws a vertical line from the x-axis to the observation height (type = "h").

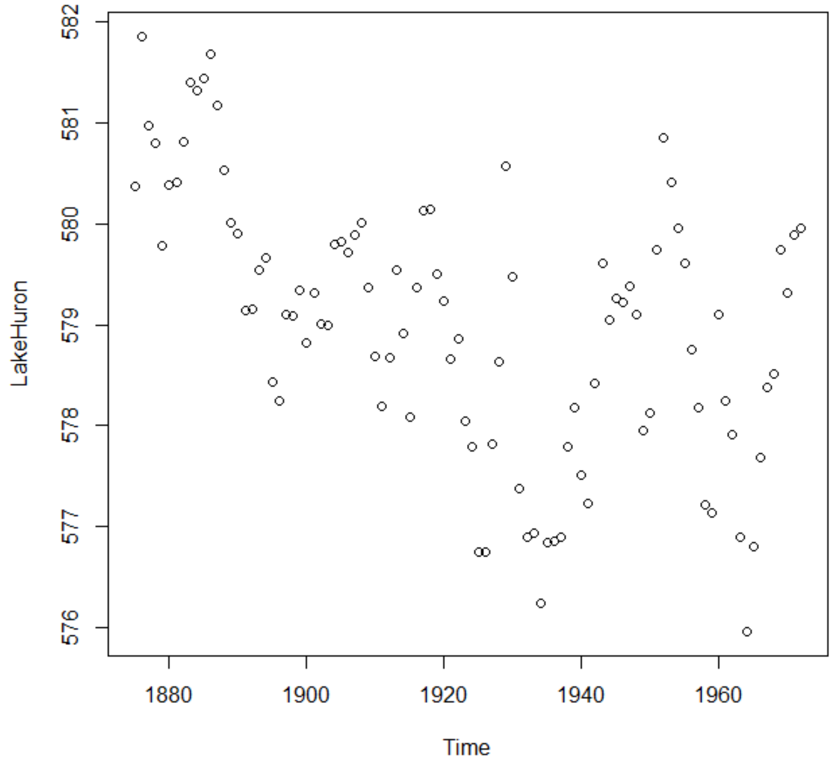
```
> plot(LakeHuron, type = "h")
```



Index plot

- points: plots a simple point at the observation height (type = "p").

```
> plot(LakeHuron, type = "p")
```



Qualitative data

- Qualitative data are simply any type of data that **are not numerical**, or do not represent numerical quantities.
- Examples of qualitative variables include a subject’s name, gender, race/ethnicity, political party, socioeconomic status, class rank, driver’s license number, and social security number (SSN).

Qualitative data

- Some data **look to be quantitative but are not,** because they do not represent numerical quantities and do not obey mathematical rules.
  - For example, a person's shoe size is typically written with numbers: 8, or 9, or 12, or 12 1 2 . Shoe size is not quantitative, however, because if we take a size 8 and combine with a size 9 we do not get a size 17.

Qualitative data

- Some qualitative data serve merely **to identify** the observation (subject's name, driver's license number, or SSN). This type of data does not usually play much of a role in statistics.
- Other qualitative variables serve **to subdivide** the data set into categories; we call these **factors.**
  - Examples are gender, race, political party, or socioeconomic status.
- The possible values of a factor are called its **levels.**
  - Example, the factor gender would have two levels, namely, male and female. Socioeconomic status typically has three levels: high, middle, and low.

Qualitative data

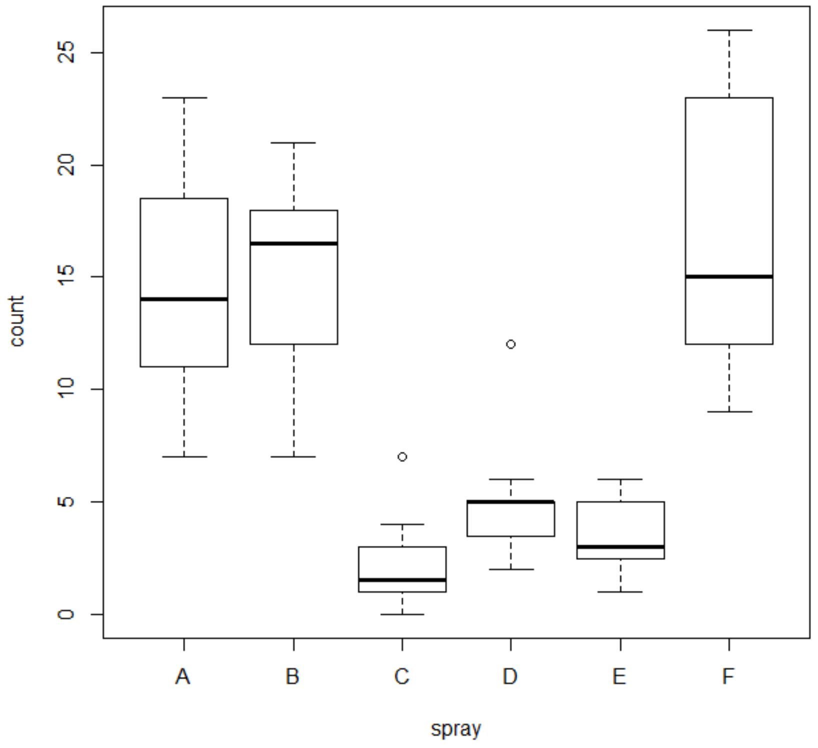
- Factors may be of two types: nominal and ordinal.
  - **Nominal** factors have levels that correspond to names of the categories, with no implied ordering. Examples of nominal factors would be hair color, gender, race, or political party. There is no natural ordering to "Democrat" and "Republican"; the categories are just names associated with different groups of people.
  - **Ordinal** factors have some sort of ordered structure to the underlying factor levels. For instance, socioeconomic status would be an ordinal categorical variable because the levels correspond to ranks associated with income, education, and occupation. Another example of ordinal categorical data would be class rank.

Qualitative data

- Factors have special status in R.
  - They are represented internally by numbers.
  - Their values do not convey any numeric meaning or obey any mathematical rules.
  - “Young” + “Baby” is not equal to old man (this is a no sense).

Box diagram

- To represent the intervals depending on the levels of a factor.
  - > boxplot(count~spray, ylab="count", xlab="spray", data=InsectSprays)



Tables

- To summarize qualitative data → table of the data values.
- We may count frequencies with the table function or list proportions with the **prop.table** function (whose input is a frequency table).
  - With R Commander you can do it with Statistics → Frequency Distribution. Also you can look at tables for all factors in the Active data set you can do Statistics → Summaries → Active Dataset.