Analyzing Values Below the Method Detection Limit Using R

Carolyn Huston
Simon Fraser University

Hosted by the Bulkley Valley Research Centre

February 2008

SFU

About this Course

The following lectures provide an introduction to using the statistical software package R. They do not demonstrate all the features of R but concentrate on those that are most useful for this workshop.

Conventions

Throughout this workshop, I will use the following conventions. These are the same conventions as are used in the water quality guidance document.

- The names of R command in the text will appear in a different font, with parentheses; for example, `summary()`.
- The names of R objects in the text will appear in the same font, but will have no parentheses; for example, `ChickWeights`.
- Commands that you can use will appear indented, preceded by the symbol >; you do not need to type > when entering commands. For example,

```
>min(weight)
```

- R output will also appear indented in a different font, for example,

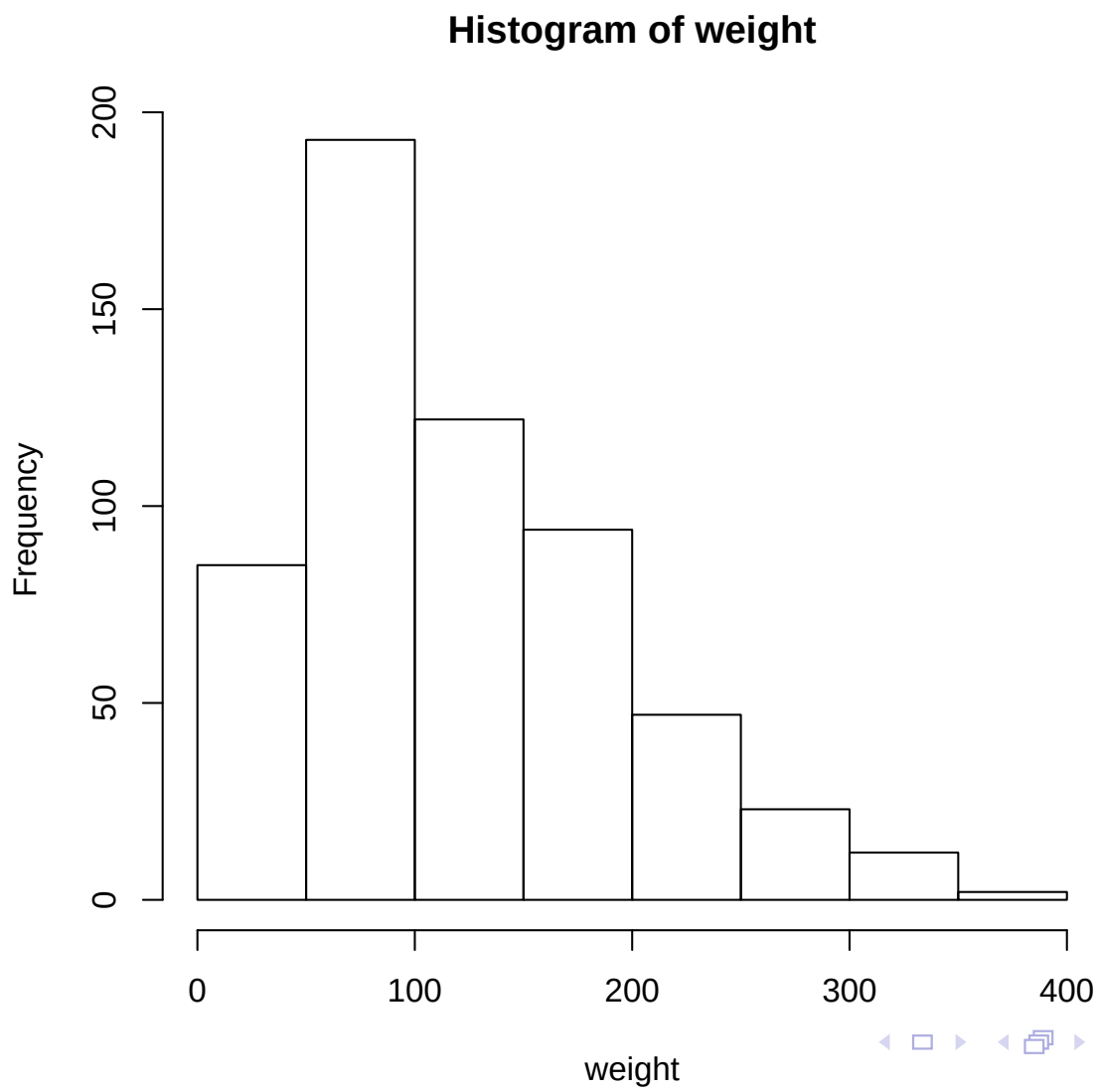
```
[1] 38
```

What is R?

R is a freely available language and environment for statistical computing and graphics providing a wide variety of statistical and graphical techniques. It is very similar to a commercial statistics package called S-Plus, which is widely used by statisticians.
R is a command-line driven package. This means that for most commands you have to type the command from the keyboard. The advantage of this is that it is very flexible to add different options to a command; for example,...

What is R

`>hist(x=weight)` will draw a histogram of the data `weight` using default options

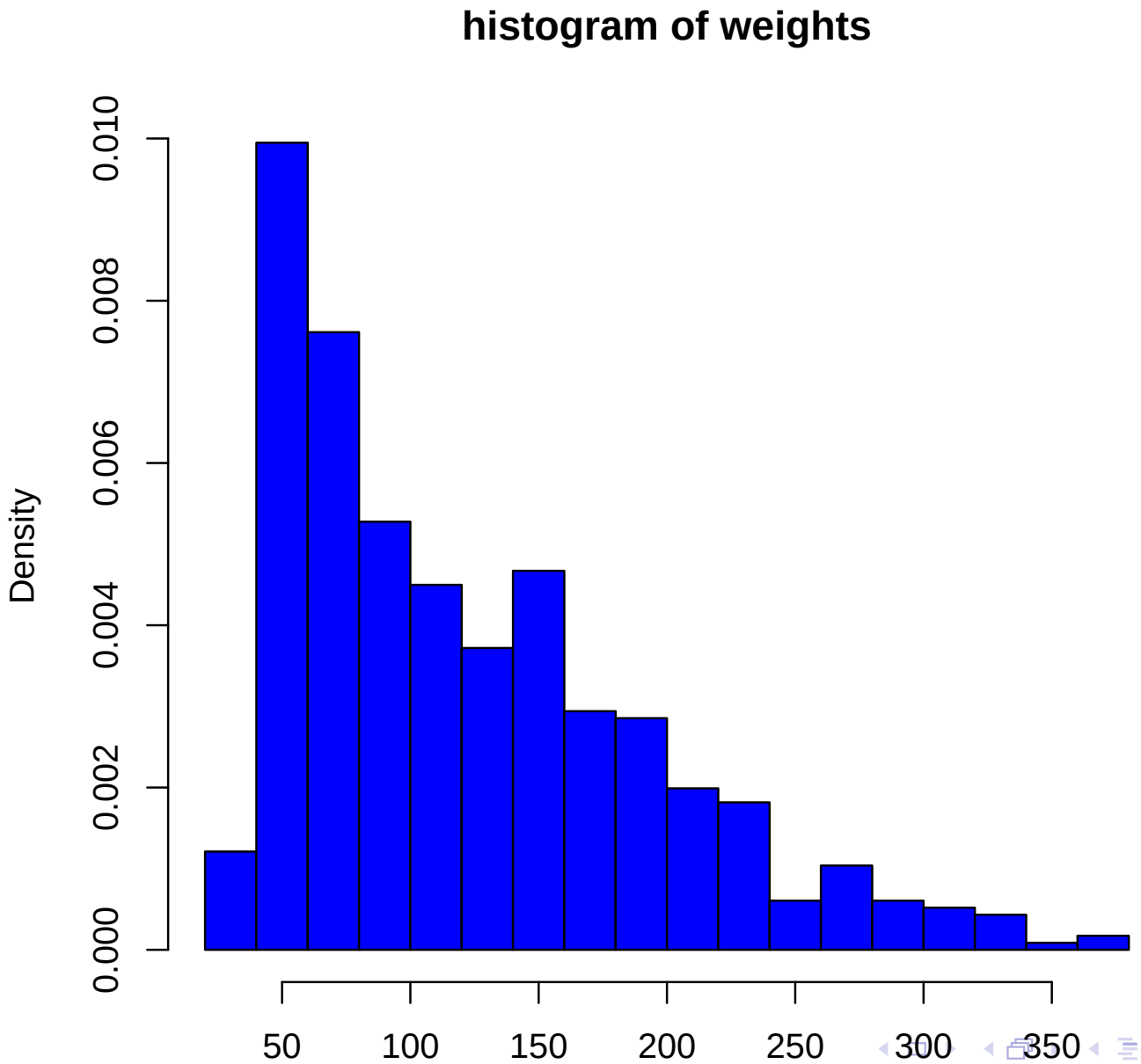


What is R?

You may also specify some of these graphing options more exactly:

```
>hist(x=weight,breaks=15,freq=FALSE, col="blue",
main="Histogram of Weights")
```

What is R?


histogram of weights
Density
0.000
0.002
0.004
0.006
0.008
0.010
50
100
150
200
250
300
350

Disadvantages of R

The main disadvantage of R, or any command-line driven program, is that it may take a little time to learn the commands. Many of the commands in R are quite intuitive, making the learning process easier.

Advantages of R (again)

- R is free and available for all major platforms(PC, linux, Mac).
- It has excellent built-in help.
- It has excellent graphical capacities.
- It is powerful with many built-in statistical functions.
- It is easy to extend with user-defined functions.

For More Information On R

For more information on R including downloads, packages introducing extra functions, manuals, and links to related websites see the R website:

http://www.r-project.org

Starting R

When R starts, a window (console) should open showing the following text (the text may vary slightly depending on the version you are using):

```
R version 2.5.0 (2007-04-23)
Copyright (C) 2007 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

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.

  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.
>
```

Starting R

The symbol, $>$, appears in red, and is called the *prompt*. This indicates that R is waiting for you to type in a command. If a command is too long to fit on a line, a $+$ is used to indicate that the command is continued from a previous line. All commands that you type appear in red, and R output appears in blue.
When you are inputting data into R, you do not need to type the prompt, $>$.

Entering Commands

The most obvious way of using R to enter commands is to type directly into the console window. For example, try typing 2+2 at the command prompt, and pressing enter. You should see the output:

```
[1] 4
```

the [1] indicates that the answer is a vector of length one; we will return to this later

Entering Commands

You can use the cursor key to edit things in the command line, similar to how you would in a program like Microsoft Word. Additionally, using the up and down will allow you to scroll backwards and forwards through previous commands; this can save a lot of typing!

The main window that you type in is the command/console window. Other windows that we will talk about are the command history window, and the graphics window.

Command History

Rather than scrolling through a long list of your previous commands using the cursor, R provides another way of retrieving previous commands. Type `history()`, and a new window will appear listing the last 25 commands that you have used. This is a text file, so then you can copy and paste the command(s) you want to use. To retrieve more than 25 previous commands, for example to retrieve the last 100, type `history(max.show=100)`.

Script Files

A third alternative allowing you to retrieve previous commands is to type your commands in a *script* file. A script file is essentially a text file, and means that you can type and edit R commands similar to how you would type and edit sentences in Microsoft Word! You can then save these commands for your referral and use, or so a project analysis can be easily shared between co-workers.
It is also much less frustrating to fix typos when writing commands if you are using a script file!!!

Script Files

To open a script file in R, simply select File→New Script to open a window for editing R scripts. You can then simply type the commands that you wish to use into the script window.
In order to save a script, you can simply select File→Save As, and save as you ordinarily would.

Script Files

When writing commands in a script file, lines are used to separate commands. If you wish to put comments in your commands that R should ignore, you simply begin the line with a hash '#'. R will ignore everything that is typed after the hash on a line. This allows you to include explanatory comments between your commands.

```
#draws a histogram
hist(weight)
#draws a slightly different histogram
hist(weight,breaks=15,freq=FALSE,
col="blue",main="Histogram of Weights")
```

Script Files

To run commands that you have typed into a script file you can do one of the following:

- If you wish to run all of the commands in your script file (excluding comments) in order, you can highlight all of the text and hold down the ctrl-r keys. Alternatively,
- You can also source a script file (without having to open the script) by typing `source("h:/analysis.txt")`. Finally,
- if you just want to run selected parts of your file, you can copy and paste selected commands, or highlight the desired commands and hold down the ctrl-R keys

We will discuss and practice more about using script files in upcoming sections.

Getting Help

One of the easiest ways to access help in R is through the Help menu. To do this, simply select Help→Html help. This displays the help in Internet Explorer, similar to a webpage. The main page has several topic areas. 'An Introduction to R' provides basic instructions and help, similar to what we are discussing this morning.

You can use the section 'Search Engine and Keywords' to search for different commands by keyword, or topic. You can also access this help using the `help.search("topic of interest")` command in the R console.

For example, using either the help menu, or the R console, read the documentation for the command `history()`

Quitting

So, you have had enough of R, and you want to escape!!

To close R, use the command `q()`. Or select File→Exit. R will ask you if you wish to save your workspace. You answer 'yes' to this if you want to keep data that you have created for the next time you open R. I generally suggest saying 'no' to this though, so your R workspace doesn't become cluttered and slow. An alternative is to save your *script* file with commands on how to create any data you were using.

Commands and Objects

R is an *object-oriented* program. This means that every piece of information that R stores is an object. For example, data, vectors, the results of functions-all are objects. Each object has a name. As an R user, you can perform actions on these objects via functions/commands. Some functions behave differently depending on the type of object they are given.

Although this might sound complicated now, in practice it makes R easy and efficient to use. We will be building more on the idea of functions and objects as we go along.

Listing Objects

Sometimes it can be useful to see a list of the objects that are available to you in your R workspace. To see a list of objects that have been created on your R workspace, you can use the `ls()` function. Typing `ls()` will cause R to list all the objects that you have access to in your current R session.

Removing Objects

It can also be useful to delete some of the objects in your R workspace. For example, data objects that have errors in them, or that you are no longer analyzing. This can also prevent confusing two objects with one another. To remove an object from your workspace, you can use the `rm()` command. For example, to remove a data object called *MyData* you would use the following syntax:

```
rm(MyData)
```

Removing Objects

On rare occasions, you might want to remove all of the objects stored in your R workspace and start again from scratch. This should be done with **extreme caution** though, because once objects are removed they **cannot** be restored. To remove all of the objects in your R workspace, type

```
rm(list=ls())
```

Simple Arithmetic

Any mathematical expression typed at the prompt, >, is evaluated, and the result printed. You can use simple arithmetic:

```
>1 + 2 + 3   #addition
[1] 6

>3*4+2   #multiplication is done first
[1] 14

>1+3/2   #so is division
[1] 2.5
```

Simple Arithmetic

```
>(1+3)/2   #use brackets to change the order
[1] 2

>4**2  #Use ** to exponentiate
[1] 16

>4^2   #or use ^
[1] 16
```

Order of Operations

As with arithmetic in general, R uses the following order of operations in its equations.

B Brackets
E Exponents
D Division and
M Multiplication
A Addition and
S Subtraction

Simple Numeric Functions

More complicated mathematical expressions can be calculated using built in functions in R. Built in functions each have their own name, and require an *argument*. For example, the function to calculate a square root has the name `sqrt()`. The argument is the number (or set of numbers) that we want to find the square root of. To make this function work in R, we place the argument inside the bracket. To find the square root of two, the argument is 2 so we type as follows:

```
> sqrt(2)
[1] 1.414214
```

Simple Numeric Functions

Similarly,

```
> log(3.14159)        #log of pi
[1] 1.144729

> log(pi)       #pi is a built in constant
[1] 1.144729
```

Simple Numeric Functions

A table of commonly used mathematical functions is given below.

`abs()`	Absolute value (without +/-)
`sign()`	1 if argument is positive, -1 if it is negative.
`sqrt()`	square root
`log()`	natural logarithm
`exp()`	exponential, e
`sin(), asin()`	sin and $\sin^{-1}$
`cos(), acos()`	cos and $\cos^{-1}$
`tan(), atan()`	tan and $\tan^{-1}$

Variables

Variables are called objects in R. You can assign a value to an object (in this case, a variable) using '=', or the assignment operator '<- (a 'less than' sign followed by a minus sign). These two methods of assigning names to objects are interchangeable.

```
x=5     #creates a variable called x
```

From now on you can use `x` in place of the number 5

Variables

```
>x+2
[1] 7

>sqrt(x)
[1] 2.236068

>y=sqrt(x)  #make a new variable, y
```

Variables

Typing the name of an object prints the value of it to the screen.

```
>y
[1] 2.236068
```

Logical Values

A logical value is either TRUE or FALSE.

```
>z=10  #set z equal to 10

>z>10  #is z strictly greater than 10?
[1] FALSE

> z<=10  #is z less than or equal to 10?
[1] TRUE
```

Logical Values

```
>z==10  #is z equal to 10?
[1] TRUE
```

To test if an object is equal to something, you must use the == operator, with a double equals sign. Be careful with this. It is easy to make a mistake and use = instead.

End of Section 1