Statistical Analysis of Agricultural Experiments using R

Andrew Kniss & Jens Streibig

Last Updated on 2020-05-28

Contents

Kniss AR, Streibig JC (2018) Statistical Analysis of Agricultural Experiments using R. https://Rstats4ag.org.

Preface

What is this text?

“Statistics, like all other disciplines, is dynamic; it is not a fixed set of unchanging rules passed down from Fisher and company.” -W. Stroup

There are many excellent books and websites available to students and practitioners who would like to learn the R statistical language. Even so, there are few resources that specifically address the type of designed experiments common in the plant and weed science disciplines. Statistics, and the R language, evolves over time. It is not uncommon for a textbook about R or statistics to become out of date within only a few years. So we decided to develop this material for the web rather than a printed textbook so that it could be more easily kept up to date.

This text is not meant to be a complete reference for all the capabilities of the R language, nor should it be used as as a substitute for consultation with a well-trained statistician. This text will not cover many of the underlying statistical concepts for the examples provided, and as such, this is certainly not a standalone statistics resource. The purpose of this text is simply to provide information and examples on how to use the R language to analyze statistical designs that are commonly used in agricultural experiments.

A majority of agricultural research uses a small subset of experimental designs, and thus, there is a high probability that the examples presented here will provide a framework for analysis of many agricultural experiments. It is important, though, that the researcher understands their own data and the experimental designs that were employed in the research so that these examples are not used inappropriately. As of this writing, the examples presented here are heavily focused on agronomic and weed science experiments, as that is the primary expertise of the authors. We welcome additional contributions from related disciplines to broaden the scope and usefulness of this text.

Apart from the differences in the structure of SAS and R languages, there is another important difference: SAS is a commercial product whereas R is an open source language. Although the underlying code for SAS is not in the public domain, history tells us that the SAS Institute does a good job and was in fact developed for analysis of agricultural experiments in the first place. Their contribution to agricultural statistics cannot be overstated.

R, on the other hand is an open source language where all codes are in the public domain and can be checked by anyone with the inclination to do so. There are many capable statisticians that develop the language and add-on packages. But anyone with the ability to code can contribute functions to R. To rephrase: R is written by statisticians and practitioners, and is meant to be used by statisticians and practitioners. An analogy to every day life could be that with SAS or other commercial programs you have a king choose the menu, and hope the chef is a good cook. With R you are given all the ingredients to make a good menu with the bits and pieces. In both cases, it is quite possible to have either a delightful meal, or an unpleasant evening.

While there are many texts already available for learning R, they are typically aimed broadly at statisticians, or targeted at a specific discipline ranging from ecology to the social sciences. The chapters in this text are primarily focused on providing data and code examples for analyzing the most common experimental designs used by agronomists and weed scientists, and thus it will hopefully be useful to students and practitioners as they attempt to learn how to use a new statistical analysis environment. The philosophy of learning by example is that we do not go into much detail about the R functions, or the statistical theory behind each function. In fact we think that the way you learned your mother tongue was to listen to adults and repeat without prior knowledge of the grammar and syntax. The same applies to R. See, do, repeat, and gradually understand the grammar and syntax.

Last but not least, a great many documents and books on R are freely available on the R website and elsewhere on the web. Because R is open source the changes in the language make it difficult to find up-to date commercial books. Consequently, we will not recommend any, but suggest this listing on the R homepage as a starting point for more information.

Cite this text

If you have found this resource useful and would like to cite this as a reference, we suggest using the following citation:

> Kniss AR, Streibig JC (2018) Statistical Analysis of Agricultural Experiments using R. https://Rstats4ag.org.

Chapter 1

Introduction

1.1 How to get help?

Learning a new language can be frustrating - especially when you're not even sure how to begin troubleshooting the problem you're having. It is important to realize there are a variety of ways to solve problems in R.

1.1.1 Help files

If you know the name of a function, but do not know or remember how to use it you can bring up the documentation for that function by typing a question mark in front of the function: `?lm()`. This will work for any function in a currently loaded package. You can do a broader search by typeing two question marks. For example, if you know the name of a function, but are not sure which package you need to load to see it: `??drm()`.

The help pages for most `R` functions start out with usage, saying `lm(...)` which details the arguments available in that function. These are mainly written for people who know the function (or at the very least are familiar with `R` documentation) to begin with. If you are new to `R` we recommend you take a look at this section, but also scroll down to the examples at the bottom of the help page to see if you can use some of the examples that fit your purpose. Some help pages are better than others, and for the developer one of the most tedious jobs is to write help pages in an understandable way for others than already seasoned R users.

1.1.2 Online help

Nearly every question related to using `R` has been already asked, and in many cases, has an answer online. So if the installed help files aren't sufficient, there are several locations to search on the web. A Google search is sometimes all you'll need. For example, searching `How to do regression with R` will bring tens of thousands of hits (varying widely in quality). There are also numeous YouTube clips in various languages for many common analyses.

One difficulty, however, is that the language name `R` sometimes makes it difficult to find the most relevant information. This is less of a problem now that `R` is a widely used lanugage, but if Google fails for this reason, the site Rseek, searches only sites with relevant R information, and can be helpful for both the novice and the seasoned user.

There are many hard-copy texts that you can purchase, and many are of very high quality. However, R is so dynamic that books on specific topics can often contain code that is not functioning even just a couple of years after publication. We recommend caution when purchasing books, and also hesitate to recommend specific titles, especially if the primary goal is finding examples to work through.

1.2 Downloading and Installing R

Instructions for downloading and installing R vary depending on the operating system, and can be found at the homepage for the R project. There is a wealth of information on installing R on the web, so we suggest searching Google or YouTube if you are having trouble or would like a step-by-step guide.

1.2.1 R packages - CRAN

One of the most notable benefits of the R language is the fact that many add-on packages have been written by other statisticians. If there is a common need for some type of analysis or calculation of a common statistic, chances are good that someone has already written code to automate the process. In many cases, this code is contributed to the R user community as an add-on package that can be freely downloaded and used by others. There are currently over 5,000 contributed packages available for free download from the Comprehensive R Archive Network (CRAN). The procedure for installing contributed packages also differs between operating systems. Instructions for installing contributed packages in R can be found in the online R documentation.

The following packages are used for some of the examples in this text, and some will need to be installed to run all of the code provided. Some of the packages below are installed by default with the base R installation, but others will need to be installed afterwards. When a package needs to be loaded, we have tried to include the code to load the library within the relevant chapter.

- `dplyr`
- `lattice`
- `ggplot2`
- `agricolae`
- `Hmisc`
- `nlme`
- `lme4`
- `emmeans`
- `multcomp`
- `drc`

1.2.2 RStudio

We suggest installing RStudio as a useful and consistent interface for R. The default appearance of R differs greatly between Windows, Mac, and Linux operating systems. RStudio is available for all 3 platforms and provides several useful features in addition to a consistent interface. One of the great benefits ot RStudio is the ability to easily use R Markdown - a version of the markdown markup language that can help greatly in conducting reproducible research and creating print and web reports. This website, for example, was created entirely in RStudio using the R Markdown language (as well as the `bookdown` package).

1.3 Conventions

Several typographical conventions will be used throughout this text. References to `R` packages (such as the `drc` package) will be highlighted. Functions will be highlighted similarly, but followed by open parentheses, for example, `aov()` or `lm()`. Example code that is meant to be typed or copied directly into the R console will be enclosed in a shaded box in a monospace font. In most cases, code will be followed by the resulting output, preceded by "##".

```
Input.statement
```

```
## [1] "This is the output"
```

1.4 Basics of using R

At its most basic, R can be used as a calculator. You can enter a mathematical operation, and R will solve it for you.

```
1+1
2*10
2.3 * 10^4
```

As with any good scientific calculator, R can store the results of those operations as an object to be called upon later. This functionality will be used extensively as you learn to use R efficiently. To build on the previous example, the results of the three mathematical operations above will be stored as objects named "a", "b", and "c":

```
1 + 1 -> a
2 * 10 -> b
2.3 * 10^4 -> c
```

The less than (or greater than) and minus symbols ("<" and "-") are used together to create an arrow for storage of a result. The direction of the arrow indicates the direction of the storage operation. The statements `a <- 1 + 1` and `1 + 1 -> a` will produce identical results. These objects can then be called upon later just by typing the name of the object; that is, typing the letter `c` into the R console will print the information stored as the object named `c`. It is important to keep in mind that R is case sensitive, so `a` and `A` are recognized as separate objects in R.

```
c
```

```
## [1] 23000
```

```
a + b + c
```

```
## [1] 23022
```

This same method can (and often will) be used to store far more complex forms of information than the result of a mathematical expression. Two objects that are commonly used to store data are vectors and data frames. Vectors can be thought of as a list of information, whereas a data frame more closely resembles a spreadsheet or table. Vectors can be created using the `concatenate` function (abbreviated with just the first letter `c`). The `data.frame()` function will produce a data frame. In the following example two vectors (`x` and `y`) are created, and then assembled into a single data frame, which is stored under the name `fake.data`.

```r
x <- c(1:10)
y <- c(10,12,13,15,16,18,19,21,23,24)
fake.data <- data.frame(x, y)
```

The vector `x` was created using 1:10; the colon in this context is shorthand to generate all consecutive integers between 1 and 10. Notice that we can name a stored object as a single letter or as a string of letters. The period is recognized by R as just another textual character in this case (as long as it is not the first character in the string), therefore naming the object `fake.data` would be acceptable, as would `fake_data` or `fakeData`. To see the full data frame, simply type the object name into the R console. Additionally, we can get various information about the vectors in the data frame by using other functions such as `summary()`, and `colMeans()`.

```r
fake.data
```

```
##     x  y
## 1   1 10
## 2   2 12
## 3   3 13
## 4   4 15
## 5   5 16
## 6   6 18
## 7   7 19
## 8   8 21
## 9   9 23
## 10 10 24
```

```r
summary(fake.data)
```

```
##        x               y       
##  Min.   : 1.00   Min.   :10.0  
##  1st Qu.: 3.25   1st Qu.:13.5  
##  Median : 5.50   Median :17.0  
##  Mean   : 5.50   Mean   :17.1  
##  3rd Qu.: 7.75   3rd Qu.:20.5  
##  Max.   :10.00   Max.   :24.0
```

```r
colMeans(fake.data)
```

```
##    x    y 
##  5.5 17.1
```

The `summary()` function provides min, max, mean, median, and 1st and 3rd quartiles of each vector in the data.frame. The `colMeans()` function returns the mean of each column in the data frame. It is also possible to apply functions to only one column within the data frame. This is accomplished by specifying the data frame, then the column, separated by a `$`. The examples below will provide the summary, mean, and standard deviation of only the `y` vector (or second column) of the data frame.

```r
summary(fake.data$y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    10.0    13.5    17.0    17.1    20.5    24.0
```

```
mean(fake.data$y)
```

```
## [1] 17.1
```

```
sd(fake.data$y)
```

```
## [1] 4.72464
```

1.5 Loading Data from External Sources

`R` can read data from many sources and file types. However, the goal here is not to provide a comprehensive list of available methods for reading data into R. For all data sets used in this book, we will use the `read.csv()` or `read.table()` functions. The `read.csv()` function is designed to read comma separated value (.csv) files. Comma separated value files can be read by almost any spreadsheet program (including MS Excel[1], Lotus 1-2-3, OpenOffice.org) or text editor, and thus it is a common file type that almost any researcher should be able to read and save.

Most research software and database software will export data to a .csv format, including the commonly used ARM from Gylling Data Management, Inc. The `read.csv()` syntax is similar to other functions used to get data from external files into `R`. If the csv file is in your current working directory, you can simply type `read.csv("filname.csv")`. It is also possible to specify the full path to the file (but this will differ depending on the operating system, Windows, Mac, or Linux). All of the data sets used in this text can be downloaded as a zip file: http://rstats4ag.org/data/Rstats4ag_data.zip. But for simplicity, we have provided the code in each chapter to read the csv files directly from the web by specifying the URL in the `read.csv()` function. For example:

```
corn.dat <- read.csv("http://rstats4ag.org/data/irrigcorn.csv")
```

The data sets are small (the largest file is 19 KB), so even a moderate internet connection should be sufficient. The default behavior of the `read.csv()` function considers periods (`.`) as a decimal point, and commas (`,`) as the symbol used to separate values within a row. Depending on geography, though, this may not be the standard format. In many areas of Europe and Latin America, the decimal seperator is a comma (`,`) and the semicolon (`;`) is the variable separator. The `read.csv()` can still be used with alternate formats by specifying the `sep` and `dec` arguments for separator and decimal symbols, respectively.

```
newdata <- read.csv("filename.csv", sep=";", dec=",")
```

Alternatively, the `read.csv2()` function can be used, where the semicolon and comma are the default seperator and decimal symbols, respectively.

```
corn2.dat <- read.csv2("http://rstats4ag.org/data/irrigcorn.csv")
```

Experience tells us that this is, unfortunately, not a trivial matter, because in some instances the two csv systems are mixed. For other file formats, the `read.table()` function provides additional flexibility. You can learn more about the options available by typing `?read.table`.

For our example above, if no error or warning messages appear then `R` has presumably read the irrigcorn.csv file, and stored the data in an object called `corn.dat`. To ensure the data was read successfully, use the`str()` function, `head()` function, or `summary()` function can be used, all producing different information.

[1] Researchers managing data in MS Excel are encouraged to look into the `read_excel()` function from the `readxl` package.

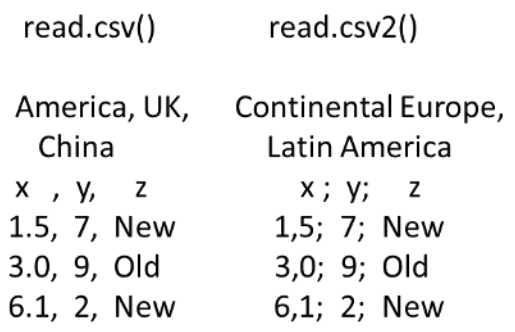


Figure 1.1: *Illustration of the difference between two common forms of comma separated value (csv) files that can be read using the `read.csv()` and `read.csv2()` functions.*

```
str(corn.dat)
```

```
## 'data.frame':    96 obs. of  8 variables:
##  $ Variety      : int  1 1 1 1 2 2 2 2 3 3 ...
##  $ Maturity     : int  92 92 92 92 86 86 86 86 84 84 ...
##  $ Irrig        : chr  "Full" "Full" "Full" "Full" ...
##  $ Population.A : int  23000 23000 23000 23000 23000 23000 23000 23000 23000 23000 ...
##  $ Population.ha: int  56810 56810 56810 56810 56810 56810 56810 56810 56810 56810 ...
##  $ Block        : int  1 2 3 4 1 2 3 4 1 2 ...
##  $ Yield.BuA    : int  202 163 186 178 177 176 179 168 172 150 ...
##  $ Yield.tonha  : num  12.7 10.2 11.7 11.2 11.1 ...
```

The `str()` function provides information on the structure of the `R` object (and can be used with *any* object, not just data frames). For data frames, the `str()` function will tell us the format for each column in the data frame. In the `corn.dat` data frame, we can see there are 96 rows and 8 columns, most of the columns contain interger (`int`) data, the 'Irrig' variable is a character variable, and the last column is a numeric (`num`) variable.

```
head(corn.dat)
```

```
##   Variety Maturity Irrig Population.A Population.ha Block Yield.BuA Yield.tonha
## 1       1       92  Full        23000         56810     1       202       12.68
## 2       1       92  Full        23000         56810     2       163       10.23
## 3       1       92  Full        23000         56810     3       186       11.68
## 4       1       92  Full        23000         56810     4       178       11.17
## 5       2       86  Full        23000         56810     1       177       11.11
## 6       2       86  Full        23000         56810     2       176       11.05
```

The `head()` function shows the first few lines of the data set, and can be useful to ensure the data are structured correctly. However, the `head()` function does not allow you to see if a column of numbers is recognized as numbers or as characters. This is a common problem when reading in data from external files; if the csv or other data file is not properly formatted or contains strange characters (e.g. `"` or `;` or even letters), then the entire column will be recognized as a character or factor variable instead of a numeric variable. This is where the `str()` or `summary()` functions are helpful to recognize this problem.

In some cases, we may have a variable coded as an integer that we would like `R` to recognize as a factor variable. In the corn data set, the 'Variety' variable is numbered, but there is no numeric order to the varieties. To convert this column to a factor variable, we can use the `as.factor()` function, and store the result as the same name as the original variable.

```
corn.dat$Variety <- as.factor(corn.dat$Variety)
str(corn.dat)
```

```
## 'data.frame':    96 obs. of  8 variables:
##  $ Variety      : Factor w/ 6 levels "1","2","3","4",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ Maturity     : int  92 92 92 92 86 86 86 86 84 84 ...
##  $ Irrig        : chr  "Full" "Full" "Full" "Full" ...
##  $ Population.A : int  23000 23000 23000 23000 23000 23000 23000 23000 23000 23000 ...
##  $ Population.ha: int  56810 56810 56810 56810 56810 56810 56810 56810 56810 56810 ...
##  $ Block        : int  1 2 3 4 1 2 3 4 1 2 ...
##  $ Yield.BuA    : int  202 163 186 178 177 176 179 168 172 150 ...
##  $ Yield.tonha  : num  12.7 10.2 11.7 11.2 11.1 ...
```

Sometimes, despite all efforts to avoid the problem, numeric data is recognized by R as text, especially when exporting large data files from Excel or other spreadsheet software. This is sometimes due to spaces or other characters out of place in the data file; but sometimes the origin of the problem is difficult to find (especially with large data files). In these cases, one trick that can be tried is to convert the data to character, then back to numeric with the following code: `data$var <- as.numeric(as.character(data$var))`.

Chapter 2

Welcome to the Tidyverse

The `tidyverse` is a collection of packages that were developed for data science. This group of packages provide some powerful and efficient functions that share a common syntax, making them seamless to use together. It is also possible (and common) to combine functions from the `tidyverse` with base `R` functions. The 'tidy' in the name comes from the concept of tidy data, which is basically a standardized way of managing data.

The `tidyverse` is a meta-package; the package itself does not contain any functions, but when you install or load `tidyverse` into your workspace, it ensures that the full group of packages are installed or loaded. The `tidyverse` includes three packages that we will use in this text (`dplyr`, `tidyr`, and `ggplot2`), plus several others.

```
library(tidyverse)
```

2.1 The `dplyr` package

One of the most useful packages in the `tidyverse` is `dplyr`, and one of the most useful functions within the `tidyverse` is the pipe. Here, we will show how to use the pipe operator `%>%` to more efficiently do some of the same operations we showed in the previous chapter.

2.1.1 The `dplyr` pipe (`%>%`)

The concept of a pipe may be familiar to many programmers, but is often a new concept to scientists without a coding background. The simplest way to think about pipes is to say to yourself "and then…" any time you see the pipe. One of the most common usages of pipes is illustrated below. In the first line, we specify which data we want to work with, *and then* we perform operation 1, *and then* we perform operation 2.

```
##  A common use of pipes (not run):
data.object %>%
  operation 1 %>%
  operation 2
```

To illustrate, we will create some pretend data to use with `dplyr`. The data frame will be named 'df', and will contain 20 rows and 4 columns. The `glimpse()` function from `dplyr` serves the same general purpose as `str()` when applied to data frames.

```r
set.seed(20102)
df1 <- data.frame(fac = rep(c("A","B","C","D"), 5),
                  x = c(1:20),
                  y = round(rnorm(20, 10, 5)*(-9:10)^2),
                  z = round(rnorm(20, 100, 20)))
glimpse(df1) # similar to the `str()` function in base R
```

```
## Rows: 20
## Columns: 4
## $ fac <chr> "A", "B", "C", "D", "A", "B", "C", "D", "A", "B", "C", "D", "A", "B", "C", "D", "A"...
## $ x   <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
## $ y   <dbl> 964, 625, 692, 309, 146, 174, 98, 63, 6, 0, 16, 49, 86, 208, 399, 258, 600, 546, 11...
## $ z   <dbl> 115, 110, 111, 92, 94, 107, 93, 95, 71, 100, 99, 75, 104, 92, 95, 75, 76, 88, 118, 104
```

2.1.2 group_by() and summarize() functions

One of the most useful things we can use dplyr for is to calculate descriptive statistics. For example, if we wanted to calculate the mean of y and z for the whole data set, we could use the `summarize()` function. This example takes the data frame 'df', *and then* uses the `summarize()` function to calculate the means of the columns named *y* and *z*.

```r
### Using a pipe to calculate means.
df1 %>%
  summarize(y.mean = mean(y),
            z.mean = mean(z))
```

```
##   y.mean z.mean
## 1  416.9   95.7
```

In this particular example, the pipe operator (`%>%`) isn't all that helpful. We could have gotten the same result more efficiently using several different methods. One method, still using the `summarize()` function, removes the pipe and adds the data frame as the first argument to the function:

```r
### Doing the same thing without a pipe:
summarize(df1,
          y.mean = mean(y),
          z.mean = mean(z))
```

```
##   y.mean z.mean
## 1  416.9   95.7
```

But pipes become much more useful once we want to use multiple functions. To illustrate, we will combine two functions, the `group_by()` and `summarize()` functions, in order to calculate means for different groups. If we assume the 'fac' variable in the data is a treatment, and the y and z variables are responses to those treatments, then we will probably want to know the means for each treatment group. Using pipes, we can do that this way:

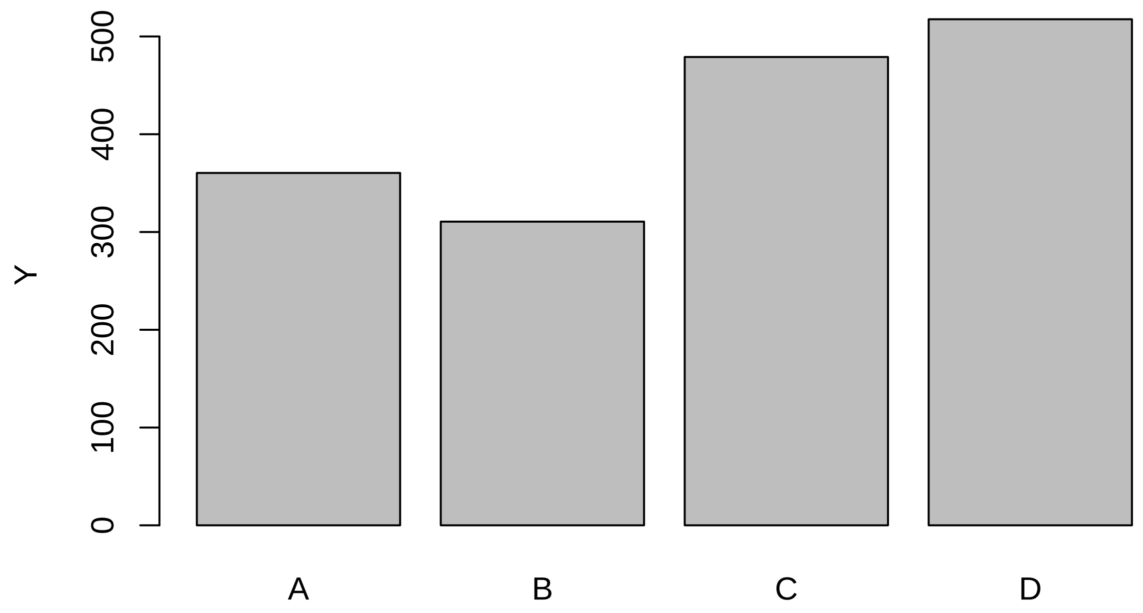
```r
df1 %>%
  group_by(fac) %>%
  summarize(y.mean = mean(y),
            z.mean = mean(z))
```

```
## # A tibble: 4 x 3
##   fac   y.mean z.mean
##   <chr>  <dbl>  <dbl>
## 1 A       360.   92
## 2 B       311.   99.4
## 3 C       479   103.
## 4 D       518.   88.2
```

This example starts with the data frame, *and then* groups the data by levels in the 'fac' column, *and then* summarizes the data by calculating the means. It will be common to use summarized data in tables and figures, so we can also store this summarized data for later use (as in, perhaps, a bar plot):

```
df1.means <- df1 %>%
  group_by(fac) %>%
  summarize(y.mean = mean(y),
            z.mean = mean(z))

### Create a barplot of the means:
barplot(df1.means$y.mean, names.arg=df1.means$fac, ylab="Y")
```



2.1.3 The `mutate()` function

The `mutate()` function is used primarily when you want to create a new column. Common scenarios would include data transformations (such as log or square root), converting units (US customary units to SI units), or calculating a new variable for each observation from the measured data (calculating area from measured length and width). The syntax of `mutate` is similar to the `summarize()` function, but instead of reducing the data, it will keep the number of table rows the same while adding (or replacing) a column. For example, if we wanted to create a new variable that is the sum of *y* and *z* (named *yz*):

```
df1.new <- df1 %>%
  mutate(yz = y+z)
head(df1.new)
```

```
##   fac x   y   z   yz
## 1   A 1 964 115 1079
## 2   B 2 625 110  735
## 3   C 3 692 111  803
## 4   D 4 309  92  401
## 5   A 5 146  94  240
## 6   B 6 174 107  281
```

Another powerful feature of `dplyr` is the ability to reference variables created earlier in the same function. For example, we can create the *yz* variableas before, and then perform a transformation on the new variable, like taking the square root all within the same `mutate()` function:

```
df1.new <- df1 %>%
  mutate(yz = y + z,
         yz.sqrt = sqrt(yz))
head(df1.new)
```

```
##   fac x   y   z   yz  yz.sqrt
## 1   A 1 964 115 1079 32.84814
## 2   B 2 625 110  735 27.11088
## 3   C 3 692 111  803 28.33725
## 4   D 4 309  92  401 20.02498
## 5   A 5 146  94  240 15.49193
## 6   B 6 174 107  281 16.76305
```

2.1.4 `filter()` and `select()` functions

<<<<<<< HEAD The `filter()` and `select()` functions are useful for subsetting or removing rows or columns from your data. The `filter()` function is used to keep or remove *rows* based on a specified criteria. For example, if we wanted to remove all rows with treatment 'C' from the data, we could do so by filtering to *keep* only rows where 'fac' is not equal to 'C'. The operator for 'not equal' in `R` is `!=` (see table of useful operators at the end of this chapter). ======= The `filter()` and `select()` functions are useful for subsetting or removing rows or columns from your data. The `filter()` function is used to keep or remove *rows* based on a specified criteria. For example, if we wanted to remove all rows with treatment 'C' from the data, we could do so by filtering to *keep* only rows where 'fac' is not equal to 'C'. The operator for 'not equal' in `R` is `!=`. >>>>>>> 8e2378fe4699900a62ed1544c8f58ec178943c0c

```
df1.noC <- df1 %>%
  filter(fac != "C")
df1.noC
```

```
##    fac  x    y   z
## 1     A  1  964 115
## 2     B  2  625 110
## 3     D  4  309  92
## 4     A  5  146  94
## 5     B  6  174 107
## 6     D  8   63  95
## 7     A  9    6  71
## 8     B 10    0 100
## 9     D 12   49  75
## 10    A 13   86 104
## 11    B 14  208  92
## 12    D 16  258  75
## 13    A 17  600  76
## 14    B 18  546  88
## 15    D 20 1909 104
```

The behavior of `filter()` is to keep all rows that match whatever criteria we set in the function, so in the previous call, we keep all rows where *fac* does **not** equal "C". We could also do the opposite, and keep only rows where the *fac* variable is equal to *C*. The operator for 'equal to' in `R` is `==`.

```
df1.onlyC <- df1 %>%
  filter(fac == "C")
df1.onlyC
```

```
##   fac  x    y   z
## 1   C  3  692 111
## 2   C  7   98  93
## 3   C 11   16  99
## 4   C 15  399  95
## 5   C 19 1190 118
```

Numeric values can be filtered similarly; for example if we want to keep only rows where 'x' is *less than or equal* to 8:

```
df1.u8 <- df1 %>%
  filter(x <= 8)
df1.u8
```

```
##   fac x   y   z
## 1   A 1 964 115
## 2   B 2 625 110
## 3   C 3 692 111
## 4   D 4 309  92
## 5   A 5 146  94
## 6   B 6 174 107
## 7   C 7  98  93
## 8   D 8  63  95
```

And, like all things in dplyr, we can filter on multiple variables simultaneously. Perhaps for data quality checks, we want to look for all observations of treatment 'C' where 'y' is greater than 1,000. Note in this example, we use two equals signs '==' as the logical operator for 'equal to', and we use the ampersand '&' as the logical AND.

```r
df1 %>%
  filter(y > 1000 & fac == "C")
```

```
##   fac  x    y   z
## 1   C 19 1190 118
```

The 'OR' operator in R is a vertical bar '|'. So if we wanted to filter rows to keep all observations that are greater than 1,000 OR less than 100, we could use the following filter arguments:

```r
df1 %>%
  filter(y > 1000 | y < 100)
```

```
##   fac  x    y   z
## 1   C  7   98  93
## 2   D  8   63  95
## 3   A  9    6  71
## 4   B 10    0 100
## 5   C 11   16  99
## 6   D 12   49  75
## 7   A 13   86 104
## 8   C 19 1190 118
## 9   D 20 1909 104
```

The `select()` function provides similar ability but for columns instead of rows. We can keep variables by including the column names in the `select()` function, and we can remove them by using a '-' sign in front of the name (but typically using one or the other, not both). The following examples will result in the same three columns being retained, just *fac*, *x*, and *y*. In the first, we will explicitly include these three columns by naming them. In the second example, we will exclude the *z* column by using the minus sign.

```r
### Example 1 - keeping columns
df1 %>%
  select(fac, x, y) %>%
  head()
```

```
##   fac x   y
## 1   A 1 964
## 2   B 2 625
## 3   C 3 692
## 4   D 4 309
## 5   A 5 146
## 6   B 6 174
```

```r
### Example 2 - removing a column
df1 %>%
  select(-z) %>%
  head()
```

```
##   fac x   y
## 1   A 1 964
## 2   B 2 625
## 3   C 3 692
## 4   D 4 309
## 5   A 5 146
## 6   B 6 174
```

2.2 The tidyr package

Another package contained in the `tidyverse` is `tidyr`. This package, as you might suspect form the name, relates to the tidy data format Wickham 2014. The two functions that we will use are called `pivot_longer()` which takes a 'wide' data set and reformats it to the long format, and `pivot_wider()` which takes a long data format and reformats it to a wider version. To begin, we'll create some pretend data in the wide format:

```
set.seed(4070)
df.wide <- data.frame(
  replicate = 1:6,
  trt1 = round(rnorm(6, 10, 1),1),
  trt2 = round(rnorm(6, 13, 1),1),
  trt3 = round(rnorm(6, 4, 1),1))
```

To see what the data looks like in wide format, we can print the `df.wide` data frame we just created:

```
df.wide
```

```
##   replicate trt1 trt2 trt3
## 1         1 11.2 13.9  4.8
## 2         2  8.5 13.9  3.8
## 3         3  8.4 13.1  3.7
## 4         4  8.9 12.9  5.1
## 5         5 10.7 13.5  3.0
## 6         6 11.1 14.3  4.3
```

To convert this wide data into the long format that is typically preferred for statistical analysis, we can use the `pivot_longer()` function. For a simple data frame like this, there are four arguments we need to provide:

1. the `data` argument, which tells R which data frame we'll be starting with;
2. the `cols` argument, which a list of columns we don't want to gather from wide to long format;
3. the `names_to` argument, which is what we want the column that contains the treatment information to be named; and
4. the `values_to` argument, which is what we want to name the column that contains the data.

For this example, the data we are using is called 'df.wide'. The columns we don't want to include are any 'housekeeping' variables; in this case it is only one column, named 'replicate'. We will store the treatment in a new column named 'treatment', and we will put the data into a new column named 'weight'. So the `pivot_longer()` function would look like this:

```
df.long <- pivot_longer(data = df.wide,
                        cols = -replicate,
                        names_to = "treatment",
                        values_to = "weight")
df.long
```

```
## # A tibble: 18 x 3
##    replicate treatment weight
##        <int> <chr>      <dbl>
##  1         1 trt1        11.2
##  2          1 trt2       13.9
##  3          1 trt3        4.8
##  4          2 trt1        8.5
##  5          2 trt2       13.9
##  6          2 trt3        3.8
##  7          3 trt1        8.4
##  8          3 trt2       13.1
##  9          3 trt3        3.7
## 10          4 trt1        8.9
## 11          4 trt2       12.9
## 12          4 trt3        5.1
## 13          5 trt1       10.7
## 14          5 trt2       13.5
## 15          5 trt3        3
## 16          6 trt1       11.1
## 17          6 trt2       14.3
## 18          6 trt3        4.3
```

We could also use the 'pipes' from `dplyr` as in the previous example to achieve the same result:

```
df.long <- df.wide %>%
  pivot_longer(cols = -replicate,
               names_to = "treatment",
               values_to = "weight")
```

In this example, there is no real benefit for using the pipes. However, if we wanted to do more operations on the same data, like filter or summarize the data, then the efficiency of pipes becomes substantial. For example, if we wanted to convert the weight from pounds to kilograms, and then summarize the data by treatment, we could use the following code:

```
df.wide %>%
  pivot_longer(cols = -replicate,
               names_to = "treatment",
               values_to = "weight") %>%
  mutate(weight.kg = weight * 0.4536) %>%
  group_by(treatment) %>%
  summarize(meanWeight.kg = mean(weight.kg))
```

```
## # A tibble: 3 x 2
##   treatment meanWeight.kg
##   <chr>             <dbl>
## 1 trt1               4.45
## 2 trt2               6.17
## 3 trt3               1.87
```

It is also possible to use pipes with base `R` functions. One useful implementation of this is to mutate (or group or summarize) data as part of the input process. For example, we can read the data file in with `read.csv`, then pipe various other functions so that the data is in the correct format for analysis right from the start. For example, the following code will get the data, use the `mutate()` function to convert a blocking variable to a factor and transform a response variable with a log transformation, then pass that to the `filter()` function to remove one year of data (2009):

```
beanDat <- read.csv("http://rstats4ag.org/data/FlumiBeans.csv") %>%
  mutate(block = factor(block),
         logDensity = log(population.4wk)) %>%
  filter(year != 2009)
glimpse(beanDat)
```

```
## Rows: 56
## Columns: 5
## $ year           <int> 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, ..
## $ treatment      <chr> "Nontreated", "Nontreated", "Nontreated", "Nontreated", "flumioxazin + t..
## $ block          <fct> 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, ..
## $ population.4wk <int> 45302, 52272, 55757, 48787, 10454, 41818, 10454, 20909, 17424, 17424, 20..
## $ logDensity     <dbl> 10.721106, 10.864216, 10.928758, 10.795219, 9.254740, 10.641082, 9.25474..
```

Table of useful operators in the `R` language.

operator	meaning
==	equal to
!=	not equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
&	and
|	or

Chapter 3

Descriptive Statistics

3.1 Calculating group means

One of the most basic exploratory tasks with any data set involves computing the mean, variance, and other descriptive statistics. A previous section has already demonstrated how to obtain many of these statistics from a data set, using the `summary()`, `mean()`, and `sd()` functions. However these functions were used in the context of an entire data set or column from a data set; in most cases it will be more informative to calculate these statistics for groups of data, such as experimental treatments. The `tapply()` function can be used for this purpose. Summary statistics for the corn irrigation data set were calculated previously using the `summary()` function. The `tapply()` function allows us to calculate similar descriptive statistics for groups of data, most commonly for treatments, but also for other logical groups, such as by experimental sites, or years. The `tapply()` function requires three arguments:

- The data column you want to summarize (response variable)
- The data column you wish to group the data by (treatment or grouping variable)
- The function you want to calculate (mean, standard deviation, maximum, etc.)

The example below will calculate mean yield for each irrigation block (Full irrigation or Limited irrigation).

```
corn.dat <- read.csv("http://rstats4ag.org/data/irrigcorn.csv")
tapply(corn.dat$Yield.BuA, corn.dat$Irrig, mean)
```

```
##     Full  Limited
## 181.1667 163.2708
```

To specify multiple grouping variables, we use the `list()` function within the `tapply()` function:

```
tapply(corn.dat$Yield.BuA, list(corn.dat$Population.A, corn.dat$Irrig), mean)
```

```
##           Full  Limited
## 23000 171.1667 147.3333
## 33000 191.1667 179.2083
```

3.2 Calculating other group statistics with tapply

We can use the same syntax to calculate nearly any statistic for the groups by replacing the `mean` argument with another function, such as `max`, `min`, `sd` etc.

```
tapply(corn.dat$Yield.BuA, list(corn.dat$Population.A, corn.dat$Irrig), median)
```

```
##       Full Limited
## 23000  171   152.5
## 33000  191   182.5
```

You can also specify other functions or operations. For example, there is no default function in `R` to calculate the CV, or coefficient of variation. This statistic is commonly used in crop sciences. The CV is simply the standard deviation divided by the mean, and so we can calculate the CV using by storing the output from `tapply()` into an object that we can call upon later.

```
popirr.sd   <- tapply(corn.dat$Yield.BuA,
                      list(corn.dat$Population.A, corn.dat$Irrig), sd)
popirr.mean <- tapply(corn.dat$Yield.BuA,
                      list(corn.dat$Population.A, corn.dat$Irrig), mean)
popirr.cv<-popirr.sd/popirr.mean
round(popirr.cv, 3)
```

```
##        Full Limited
## 23000 0.075   0.102
## 33000 0.052   0.098
```

The `round()` function simply rounds the result to 3 decimal places to make the output more readable. Alternatively, if the CV is a statistic we will use regularly, it may be worthwhile to write our own function to save keystrokes later on. We can do this using `function()`.

```
cv <- function(x) {
  xm  <- mean(x)
  xsd <- sd(x)
  xcv <- xsd/xm
  round(xcv, 4)
}
```

Or, more efficiently:

```
cv <- function(x) {
  xcv <- round(sd(x) / mean(x), 4)
}
tapply(corn.dat$Yield.BuA, list(corn.dat$Population.A, corn.dat$Irrig), cv)
```

```
##         Full Limited
## 23000 0.0750  0.1021
## 33000 0.0523  0.0976
```

This approach is desirable if the CV will be used repeatedly. Once the function `cv()` has been defined, you can call upon it repeatedly with very few keystrokes.

3.3 Using dplyr to summarize data

In addition to using `tapply()` from base stats R package, there is an excellent package for summarizing data called `dplyr`. We can get similar information from the corn data set using the `dplyr` package using the following syntax (explained in more detail below).

```
library(dplyr)
corn.summary <- corn.dat %>%
  group_by(Population.A, Irrig) %>%
  summarize(N        = length(Yield.BuA),
            AvgYield = round(mean(Yield.BuA),1),
            CV       = cv(Yield.BuA))
corn.summary
```

```
## # A tibble: 4 x 5
## # Groups:   Population.A [2]
##   Population.A Irrig       N AvgYield     CV
##          <int> <chr>   <int>    <dbl>  <dbl>
## 1        23000 Full       24     171. 0.075
## 2        23000 Limited    24     147. 0.102
## 3        33000 Full       24     191. 0.0523
## 4        33000 Limited    24     179. 0.0976
```

We will explain and demonstrate some of the basic uses of `dplyr` using herbicide resistant weed data from Canada. The following data set was assembled using data from weedscience.org.

```
resistance<-read.csv("http://rstats4ag.org/data/CanadaResistance2.csv")
resistance<-na.omit(resistance)
head(resistance)
```

```
##   Province ID          SciName Year                                  MOA
## 1  Alberta  1  Stellaria media 1988                  ALS inhibitors (B/2)
## 2  Alberta  2  Kochia scoparia 1989                  ALS inhibitors (B/2)
## 3  Alberta  3      Avena fatua 1989 Multiple Resistance: 2 Sites of Action
## 6  Alberta  4  Setaria viridis 1989          Microtubule inhibitors (K1/3)
## 7  Alberta  5      Avena fatua 1991                ACCase inhibitors (A/1)
## 8  Alberta  6 Sinapis arvensis 1993                  ALS inhibitors (B/2)
```

In the data file, there are instances of multiple resistance (resistance to more than one mode of action). The second line of the code above (`na.omit()`) removes all lines in the data set that contain `NA` (which is the default value in R for missing data). This is necessary for this particular example so that instances of multiple resistance are not counted multiple times. We may want to use the information from those rows later on, however, and therefore this method is preferable to deleting the rows from the raw data file.

One of the first things we might like to do is simply count the number of herbicide resistant weed cases by mode of action. We could do this rather quickly using the `tapply()` function discussed in the previous section: `tapply(resistance$ID, resistance$MOA, length)`. In the code below, we will use the `dplyr` package to get some interesting information from this data set. First, we can simply count the number of herbicide resistant weeds for each mode of action in the data set. Initially it seems like much more typing compared to using `tapply()`, but later on it will become clear why this is a potentially more powerful method.

```
library(dplyr)
bymoa<-group_by(resistance, MOA)
moatotal<-summarize(bymoa, total=length(ID))
print(moatotal)
```

In the code above, we are loading the dplyr package, then using two functions from that package. The `group_by()` function will provide the information on how the data are grouped. In this case, we want to group the data by MOA. We can then use the the `summarize()` function to calculate means, counts, sums, or any other operation on the data. Above, we used the `length()` function to count the unique instances of the ID variable for each mode of action. Storing each step with the `<-` operator can be a little cumbersome, especially if we will not need to use the intermediary steps again. The `dplyr` package uses the `%>%` operator to string together several steps. The code below will result in the same summary as above, with fewer keystrokes, and without storing the intermediary grouping as a separate object. The first line provides the name of the data set (`resistance`) followed by the `%>%` operator. This tells `R` to expect a function on the next line from the dplyr package. The second line groups the resistance data set by MOA, and the third line summarizes the data by calculating the length of the ID variable for each level of the grouping factor. Effectively, this counts each instance of resistance. The last line prints out the resulting data frame.

```
moatotal <- resistance %>%
  group_by(MOA) %>%
  summarize(total=length(ID))
moatotal
```

```
## # A tibble: 13 x 2
##    MOA                                       total
##    <chr>                                     <int>
##  1 "ACCase inhibitors (A/1)"                     9
##  2 "ALS inhibitors (B/2)"                       42
##  3 "EPSP synthase inhibitors (G/9)"              2
##  4 "Lipid Inhibitors (thiocarbamates) (N/8)"     2
##  5 "Microtubule inhibitors (K1/3)"               3
##  6 "Multiple Resistance: 2 Sites of Action "    12
##  7 "Multiple Resistance: 3 Sites of Action "     3
##  8 "Multiple Resistance: 4 Sites of Action "     1
##  9 "Photosystem II inhibitors (C1/5)"           12
## 10 "PSI Electron Diverter (D/22)"                3
## 11 "PSII inhibitor (Ureas and amides) (C2/7)"    3
## 12 "PSII inhibitors (Nitriles) (C3/6)"           2
## 13 "Synthetic Auxins (O/4)"                      3
```

The real power and elegance of the `dplyr` package becomes apparent once we begin using multiple groupings, and creating new variables. Below, we generate totals for each mode of action again, but also by year of listing by using the `group_by()` and `sumarize()` functions.

```
totals <- resistance %>%
  group_by(Year, MOA) %>%
  summarize(total = length(ID))
head(totals)
```

```
## # A tibble: 6 x 3
## # Groups:   Year [6]
##    Year MOA                              total
##   <int> <chr>                           <int>
## 1  1957 Synthetic Auxins (O/4)              1
## 2  1973 Photosystem II inhibitors (C1/5)    1
## 3  1976 Photosystem II inhibitors (C1/5)    2
## 4  1977 Photosystem II inhibitors (C1/5)    3
## 5  1980 Photosystem II inhibitors (C1/5)    1
## 6  1981 Photosystem II inhibitors (C1/5)    3
```

```
tail(totals)
```

```
## # A tibble: 6 x 3
## # Groups:   Year [3]
##    Year MOA                                      total
##   <int> <chr>                                    <int>
## 1  2010 "EPSP synthase inhibitors (G/9)"             1
## 2  2011 "ACCase inhibitors (A/1)"                    1
## 3  2011 "ALS inhibitors (B/2)"                       2
## 4  2011 "Multiple Resistance: 2 Sites of Action "    2
## 5  2012 "ALS inhibitors (B/2)"                       1
## 6  2012 "Multiple Resistance: 2 Sites of Action "    3
```

We can then create a new variable using the `mutate()` function; we will add a cumulative total of herbicide resistance cases over time, while retaining information about mode of action by first ungrouping the data, then using mutate to calculate the cumulative total using the `cumsum()` from the default stats package.

```
totals <- resistance %>%
  group_by(Year, MOA) %>%
  summarize(total = length(ID)) %>%
  ungroup() %>%
  mutate(cumulative = cumsum(total))
head(totals)
```

```
## # A tibble: 6 x 4
##    Year MOA                              total cumulative
##   <int> <chr>                            <int>      <int>
## 1  1957 Synthetic Auxins (O/4)               1          1
## 2  1973 Photosystem II inhibitors (C1/5)     1          2
## 3  1976 Photosystem II inhibitors (C1/5)     2          4
## 4  1977 Photosystem II inhibitors (C1/5)     3          7
## 5  1980 Photosystem II inhibitors (C1/5)     1          8
## 6  1981 Photosystem II inhibitors (C1/5)     3         11
```

```
tail(totals)
```

```
## # A tibble: 6 x 4
##    Year MOA                                      total cumulative
##   <int> <chr>                                    <int>      <int>
## 1  2010 "EPSP synthase inhibitors (G/9)"             1         88
## 2  2011 "ACCase inhibitors (A/1)"                    1         89
## 3  2011 "ALS inhibitors (B/2)"                       2         91
## 4  2011 "Multiple Resistance: 2 Sites of Action "    2         93
## 5  2012 "ALS inhibitors (B/2)"                       1         94
## 6  2012 "Multiple Resistance: 2 Sites of Action "    3         97
```

We can also use `dplyr` to look at other grouping, including by province or weed species, and then use the `subset()` function to print out only one group. In the below example, we will calculate totals for each province by weed species then keep only the cases from Saskatchewan.

```
resistance.SK <- resistance %>%
  group_by(Province, SciName) %>%
  summarize(total=length(ID)) %>%
  filter(Province == "Saskatchewan") %>%
  arrange(-total)
resistance.SK
```

```
## # A tibble: 12 x 3
## # Groups:   Province [1]
##    Province     SciName                  total
##    <chr>        <chr>                    <int>
##  1 Saskatchewan Avena fatua                  4
##  2 Saskatchewan Setaria viridis              3
##  3 Saskatchewan Kochia scoparia              2
##  4 Saskatchewan Amaranthus retroflexus       1
##  5 Saskatchewan Capsella bursa-pastoris      1
##  6 Saskatchewan Chenopodium album            1
##  7 Saskatchewan Galium spurium               1
##  8 Saskatchewan Lolium persicum              1
##  9 Saskatchewan Salsola tragus               1
## 10 Saskatchewan Sinapis arvensis             1
## 11 Saskatchewan Stellaria media              1
## 12 Saskatchewan Thlaspi arvense              1
```

Chapter 4

Simple Correlation

To quantify relationships between quantitative treatment structures (such as increasing rates) and a response variable, regression analysis is most appropriate. But in many cases it is of interest to determine the strength of a relationship between two measured variables (where no cause-effect or dose-response relationship is obvious). This can most easily be done with correlation analysis. Examples where correlation analysis would be appropriate include observational studies where soil texture (e.g. percent clay content) is thought to be related to density of a particular weed species; or the amount of sunlight reaching the crop canopy is related to crop biomass. In these cases, experimental treatments may not be explicitly designed, but rather many areas of a field could be sampled for both variables, and the correlation analysis conducted to quantify the strength of the relationship.

The relationship between visual assessment of crop injury and dry weight (or yield) production of a crop may also be of interest to a researcher. This would not be a cause-effect relationship (injury symptoms do not cause dry weight reduction), and therefore correlation analysis is more appropriate than regression. In a study by Lyon and Kniss (2010), proso millet was tested for its tolerance to preemergence applications of the herbicide saflufenacil. Injury to the crop was evaluated visually at 10 and 20 days after treatment, and dry weight was measured 28 days after treatment. It is expected that visual injury symptoms would have a strong (negative) relationship with dry weight production; however, exceptions to this relationship may exist. Correlation analysis can be conducted in R using the cor.test function.

```
millet.dat <- read.csv("http://rstats4ag.org/data/millet.csv")
cor.test(millet.dat$injury.10dat,millet.dat$drywt)
```

```
##
##  Pearson's product-moment correlation
##
## data:  millet.dat$injury.10dat and millet.dat$drywt
## t = -12.529, df = 407, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.5941612 -0.4538439
## sample estimates:
##        cor
## -0.5275917
```

```
cor.test(millet.dat$injury.20dat,millet.dat$drywt)
```

```
##
```

```
##  Pearson's product-moment correlation
##
## data:  millet.dat$injury.20dat and millet.dat$drywt
## t = -14.724, df = 411, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.6474176 -0.5206627
## sample estimates:
##        cor
## -0.5876339
```

Both injury evaluations were highly correlated (P<0.0001) to dry weight. The 20 day evaluation had a stronger relationship with dry weight (r=-0.59) compared with the 10 day evaluation (r=-0.53). This would be expected as the 20 day evaluation was taken only 8 days before the plants were harvested for dry weight.

The output from cor.test also indicates that Pearson's correlation was used, which is the default correlation method, and in this case is appropriate. The cor.test function can also conduct Kendall and Spearman correlation, simply by adding the "method" as an argument within the function:

```
var1<-1:5
var2<-2:6
cor.test(var1, var2, method="kendall")
```

```
##
##  Kendall's rank correlation tau
##
## data:  var1 and var2
## T = 10, p-value = 0.01667
## alternative hypothesis: true tau is not equal to 0
## sample estimates:
## tau
##   1
```

```
cor.test(var1, var2, method="spearman")
```

```
##
##  Spearman's rank correlation rho
##
## data:  var1 and var2
## S = 4.4409e-15, p-value = 0.01667
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
## rho
##   1
```

The pairs function allows a quick method for visualizing relationships between variables. In the example below, the 4th through 6th data columns (both injury evaluations and dry weight) are included in pairwise scatterplots.

```
par(cex=2, mgp=c(2,.7,0))
pairs(millet.dat[4:6], pch=19, col=rgb(red=0.1, green=0.1, blue=0.1, alpha=0.2))
```

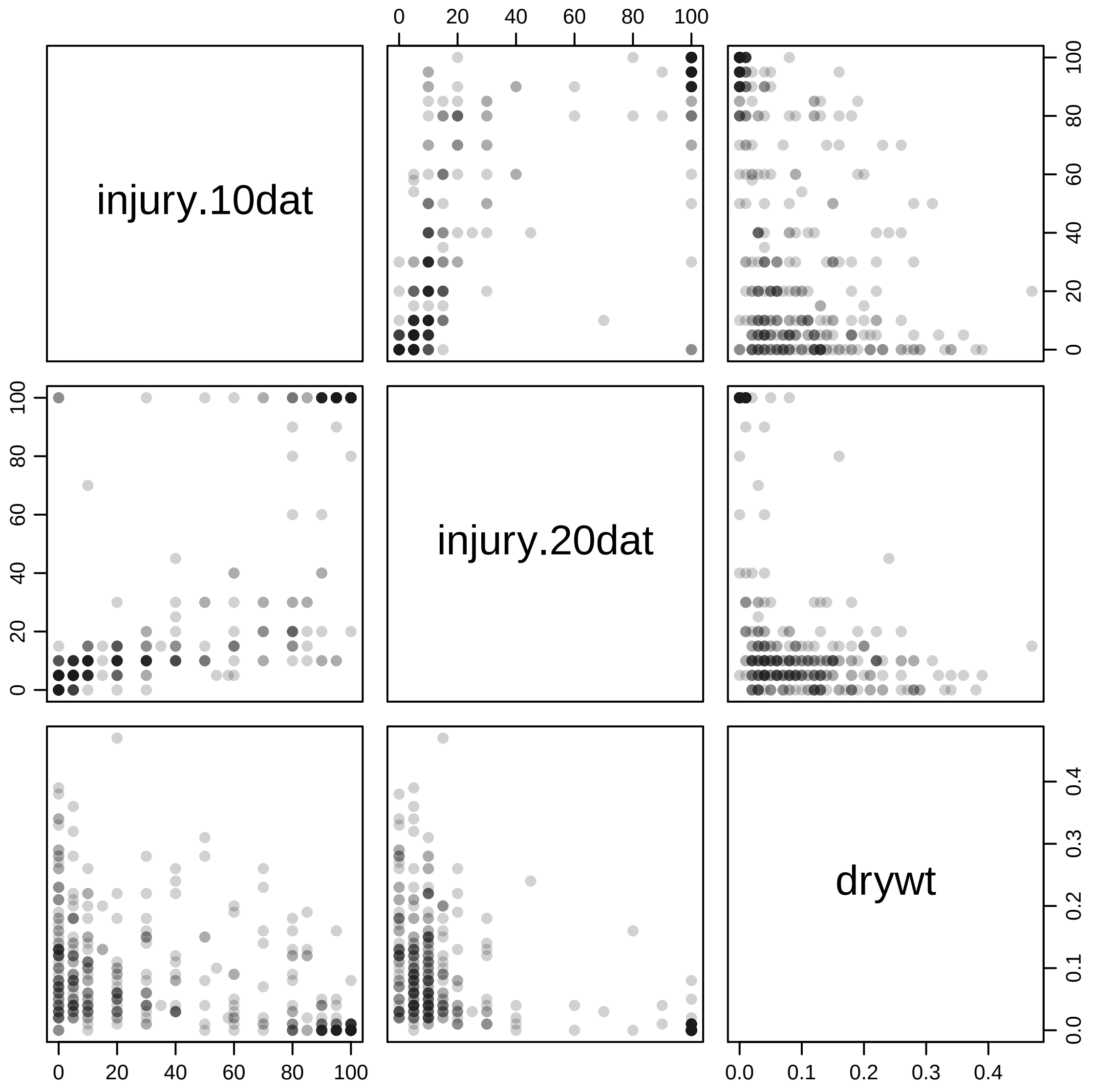


Figure 4.1: *Relationship between injury and dry weight using the* `pairs()` *function.*

Chapter 5

Student's t-test

5.1 One-sample t-test

One of the most basic statistical analyses involves testing whether a group of data is statistically similar to a known value. For example, a researcher may want to know whether the yield data from a wheat experiment was similar to the state-wide average wheat yield for that year. In the example below, yield data (in kg/ha) from 10 different plots is entered manually, and stored in an object named "yield". The state-wide dryland wheat average for the same year was 1812 kg/ha. Summary information of the wheat yield, and the differences between the experiment yields and the state average ("yld.diff") are calculated. The `t.test()` function is used to calculate the one-sample Student's t-test.

```r
library(tidyverse)
yld.dat <- data.frame(yield = c(2280, 2690, 2080,
                                 2820, 1340, 2080,
                                 2480, 2420, 2150, 1880)) %>%
  mutate(yld.diff = yield - 1810)
yld.dat
```

```
##    yield yld.diff
## 1   2280      470
## 2   2690      880
## 3   2080      270
## 4   2820     1010
## 5   1340     -470
## 6   2080      270
## 7   2480      670
## 8   2420      610
## 9   2150      340
## 10  1880       70
```

```r
colMeans(yld.dat)
```

```
##    yield yld.diff
##     2222      412
```

The mean wheat yield in the experiment was 2222 kg/ha, 412 kg/ha greater than the state-wide average. There are several possible methods to conduct the one-sample t-test. The first is to subtract the state-wide mean from each observation from the experiment, then test whether the mean difference in yield is different from zero. This is done simply by running the `t.test()` function on the "yld.diff" object calculated above.

```r
t.test(yld.dat$yld.diff)
```

```
##
##  One Sample t-test
##
## data:  yld.dat$yld.diff
## t = 3.065, df = 9, p-value = 0.01346
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  107.9233 716.0767
## sample estimates:
## mean of x
##       412
```

The `t.test()` output specifies that the alternative hypothesis is that the true mean is not equal to zero, and gives a P-value of 0.013; there is fairly strong evidence that the wheat yield in the experiment is different from the state-wide average. The `t.test()` function also allows the user to specify a value other than zero to test. This eliminates the need to first calculate the difference of each mean from the state-wide average, and instead, specify the state-wide average within the `t.test()` function using the `mu` argument. `mu` is a common statistical symbol for the population mean.

```r
t.test(yld.dat$yield, mu=1810)
```

```
##
##  One Sample t-test
##
## data:  yld.dat$yield
## t = 3.065, df = 9, p-value = 0.01346
## alternative hypothesis: true mean is not equal to 1810
## 95 percent confidence interval:
##  1917.923 2526.077
## sample estimates:
## mean of x
##      2222
```

Results are identical to testing whether the difference between experiment yields and the state-wide average yield is different from zero. In both of the above cases, a two-sided alternative hypothesis is tested; that is, the alternate hypothesis being tested is that the experimental average is different from the state-wide average yield. In some cases, though, a one-sided alternative hypothesis would make more practical sense. Perhaps it is assumed *a priori* that the treatments in this experiment will improve wheat yields. Therefore it would be more logical to test whether the wheat yield in the experiment was greater than (not simply different from) the state-wide average. This can be done using the `alternative` argument in the `t.test()` function. The `alternative` argument can be set to either 'greater' or 'less'.

```r
t.test(yld.dat$yield, alternative="greater", mu=1810)
```

```
##
##  One Sample t-test
```

```
##
## data:  yld.dat$yield
## t = 3.065, df = 9, p-value = 0.006731
## alternative hypothesis: true mean is greater than 1810
## 95 percent confidence interval:
##  1975.595      Inf
## sample estimates:
## mean of x
##      2222
```

As expected, the one-sided alternative produces a p-value of exactly one-half of the two-sided alternative. It appears that there is strong evidence that the experimental mean wheat yield is greater than the state-wide average wheat yield.

5.2 Two-sample t-Test

The two-sample t-test is more common than the one-sample test in designed agricultural experiments. In a two-sample t-test, there are two groups (often experimental treatments) on which data is collected. Some examples where the two-sample t-test would be used might include testing isogenic crop varieties to test for yield drag, or comparing two herbicide formulations for absorption or efficacy. Another example from the published literature can be found in Kniss et al. (2011), where glyphosate-resistant sugarbeet was compared with conventional sugarbeet varieties. For this study, 11 fields in Wyoming were split, with one side planted to glyphosate-resistant varieties, and the other side planted to conventional varieties. The two sides of the field were then managed as the grower thought best for the rest of the year.

```r
sbeet.dat <- read.csv("http://rstats4ag.org/data/sugarbeet.csv") %>%
  mutate(Yield = round(Yield * 2.24, 1)) # Convert data to SI units
head(sbeet.dat)
```

```
##   Field Type Yield
## 1     1    C  72.8
## 2     1    R  74.4
## 3     2    C  64.1
## 4     2    R  65.4
## 5     3    C  56.7
## 6     3    R  68.8
```

In the “sugarbeet.csv” file, each field is represented in two data rows: one row for the glyphosate-resistant (‘R’) side, and a second row for the conventional side (‘C’). Therefore, prior to running the t.test function, the data is divided into two separate data frames using the subset function. The subset function takes the `sbeet.dat` data frame, and copies only the observations (rows) where the column “Type” matches an argument; ”R” for the first, and ”C” for the second. This results in one data frame that contains only the glyphosate-resistant observations (“RR”) and one data frame that contains only the conventional observations (“CON”).

```r
RR <- subset(sbeet.dat, sbeet.dat$Type == "R")
CON <- subset(sbeet.dat, sbeet.dat$Type == "C")
t.test(RR$Yield, CON$Yield)
```

```
##
##  Welch Two Sample t-test
```

```
##
## data:  RR$Yield and CON$Yield
## t = 1.3745, df = 19.079, p-value = 0.1852
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4.064981 19.628617
## sample estimates:
## mean of x mean of y
##  58.60909  50.82727
```

The t.test function then compares the "Yield" column from the RR data frame to yield data in the the CON data frame. The t-test output provides the mean yield for each group (58.6 tons/ha for glyphosate-resistant, and 50.8 tons/ha for the conventional). In this case it appears that due to the variability in the data, there is not strong evidence for a difference between the two systems (P=0.185). Since there are only two groups in the "Type" column, we can simplify the code above by not subsetting the data ahead of time. We can simply use the `t.test()` function and specify that Type contains the relevant treatment information by using the formula `response ~ groups`, specifically in this case: `Yield ~ Type`.

```
t.test(Yield ~ Type, data=sbeet.dat)
```

5.3 Paired t-Test

A special case of the two-sample t-test is the paired t-test. In this sugarbeet example, the two-sample t-test assumes the 11 observations on the glyphosate-resistant fields are independent from the 11 observations on the conventional fields. In reality, this was not the case. The way the study was carried out, conventional sugarbeet varieties were planted in the same (or immediately adjacent) field as the glyphosate-resistant varieties. Therefore, the glyphosate-resistant observation was not actually independent from the observation on the conventional side of the field. For each field, weather, irrigation, fertility, etc. were all the same for both sides of the field. We can therefore assume that most of the variability between the conventional and glyphosate-resistant portions of the first field is due to the varieties, or the way each side of the field was managed (herbicides, tillage, etc.), and not due to unrelated external factors. Because the purpose of the study was to compare the glyphosate-resistant and conventional systems, it is more desirable to consider the two sides of the same field as "paired" samples, to which different treatments were applied. This can be achieved by adding the argument `paired=T` to the `t.test()` function.

```
t.test(Yield ~ Type, data=sbeet.dat, paired=T)
```

```
##
##  Paired t-test
##
## data:  Yield by Type
## t = -3.3302, df = 10, p-value = 0.007615
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -12.988388  -2.575248
## sample estimates:
## mean of the differences
##               -7.781818
```

In this case, we reach a very different conclusion; that within a field pair, there is very strong evidence that the difference between group means is different from zero (P=0.008); or that within each field, yield in the glyphosate-resistant system was different from conventional sugarbeet yields. On average, the glyphosate-resistant system resulted in 7.8 tons/ha greater yield than the conventional system. Notice that the paired t-test provides the exact same result as if we subtract the conventional yields from the glyphosate-resistant yields for each pair, then conduct a one-sample t-test on the difference.

```
RR$Yield - CON$Yield -> diff.Yield
t.test(diff.Yield)
```

In this example, there was no *a priori* expectation that the glyphosate-resistant system would yield more or less than the conventional system, and therefore the two-sided alternative hypothesis is appropriate.

Chapter 6

Analysis of Variance (ANOVA)

6.1 One-Way Analysis of Variance

Some of the most often used experimental designs in weed science include completely randomized designs (CRD) and randomized complete block designs (RCBD). These experiments are typically analyzed using analysis of variance (ANOVA). The one-way ANOVA is used to determine the effect of a single factor (with at least three levels) on a response variable. Where only two levels of a single factor are of interest, the `t.test()` function will be more appropriate. There are several ways to conduct an ANOVA in the base R package. The `aov()` function requires a response variable and the explanatory variable separated with the `~` symbol. It is important when using the `aov()` function that your data are balanced, with no missing values. For data sets with missing values or unbalanced designs, please see the section on mixed effects models.

The "FlumiBeans" data set is from a herbicide trial conducted in Wyoming during three different years (2009 through 2011). Five different herbicide treatments were applied, plus nontreated and handweeded controls for a total of seven treatments. The study was a randomized complete block design (RCBD) each year. There were 3 replicates in 2009, but 4 replicates in subsequent years. The response variable in this data set is dry bean (*Phaseolus vulgaris*) density 4 weeks after planting expressend in plants per acre.

It is important to note that some of our factor variables have been expressed in the .csv file as numbers (e.g. 'year' and 'block'), and consequently, R will recognized these factors as numeric variables. It is important for our ANOVA that these variables be included as factors, and not as numeric variables. We can use the `dplyr` package from the `tidyverse` to convert these variables to the proper form as we read them with the `mutate()` function.

```
library(tidyverse)

bean.dat <- read.csv("http://rstats4ag.org/data/FlumiBeans.csv") %>%
  mutate(population.4wk = population.4wk * 2.47, # Convert to SI units
         year = factor(year),
         block = factor(block),
         treatment = factor(treatment))
glimpse(bean.dat)
```

```
## Rows: 77
## Columns: 4
## $ year           <fct> 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, ..
## $ treatment      <fct> Nontreated, Nontreated, Nontreated, flumioxazin + trifluralin, flumioxaz..
## $ block          <fct> 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, ..
## $ population.4wk <dbl> 137719.79, 111895.94, 94682.51, 34429.33, 68858.66, 77466.61, 25821.38, ..
```

For this analysis, we want to look at a One-Way analysis of variance, and so we will focus on the data from only one year. We can subset the data to include only this year using the `filter()` function from the `dplyr` package, and call this subsetted data `bean.09`.

```
bean.09 <- bean.dat %>%
  filter(year == 2009)
```

We will then use the `aov()` function to generate the One-way ANOVA to see if herbicide treatment had an effect on dry bean population (expressed in plants per hectare) in 2009.

```
aov(population.4wk ~ block+treatment, data=bean.09)->b09.aov
```

6.1.1 Checking ANOVA assumptions

ANOVA assumes (1) errors are normally distributed, (2) variances are homogeneous, and (3) observations are independent of each other. ANOVA is fairly robust to violations of the normality assumption, and thus it typically requires a rather severe violation before remedial measures are warranted. However, it is still wise to check the normality assumption prior to interpreting the ANOVA results. Three methods are provided to check normality of the response variable of interest: the Shapiro-Wilks test, a stem and leaf plot, and a Normal Q-Q plot. The model residuals are automatically stored within the ANOVA model produced with the `aov()` function, and thus we can call upon the model to view and test the residuals for normality. We can get the residuals by using the `resid()` function, or by appending `$resid` onto the model fit. `resid(b09.aov)` and `b09.aov$resid` will provide the same result.

```
shapiro.test(b09.aov$resid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  b09.aov$resid
## W = 0.9718, p-value = 0.7726
```

```
stem(b09.aov$resid)
```

```
##
##   The decimal point is 4 digit(s) to the right of the |
##
##   -2 | 690
##   -0 | 88607431
##    0 | 222599
##    2 | 2787
```

```
par(mar=c(3.2,3.2,2,0.5), mgp=c(2,.7,0))
qqnorm(b09.aov$resid)
qqline(b09.aov$resid)
```

It appears that the residuals in this case are at least close to normally distributed. The most common next step if residuals are not normally distributed would be to attempt a transformation. However, it is worth noting that there are other, more modern methods to deal with non-normality that should be strongly considered.

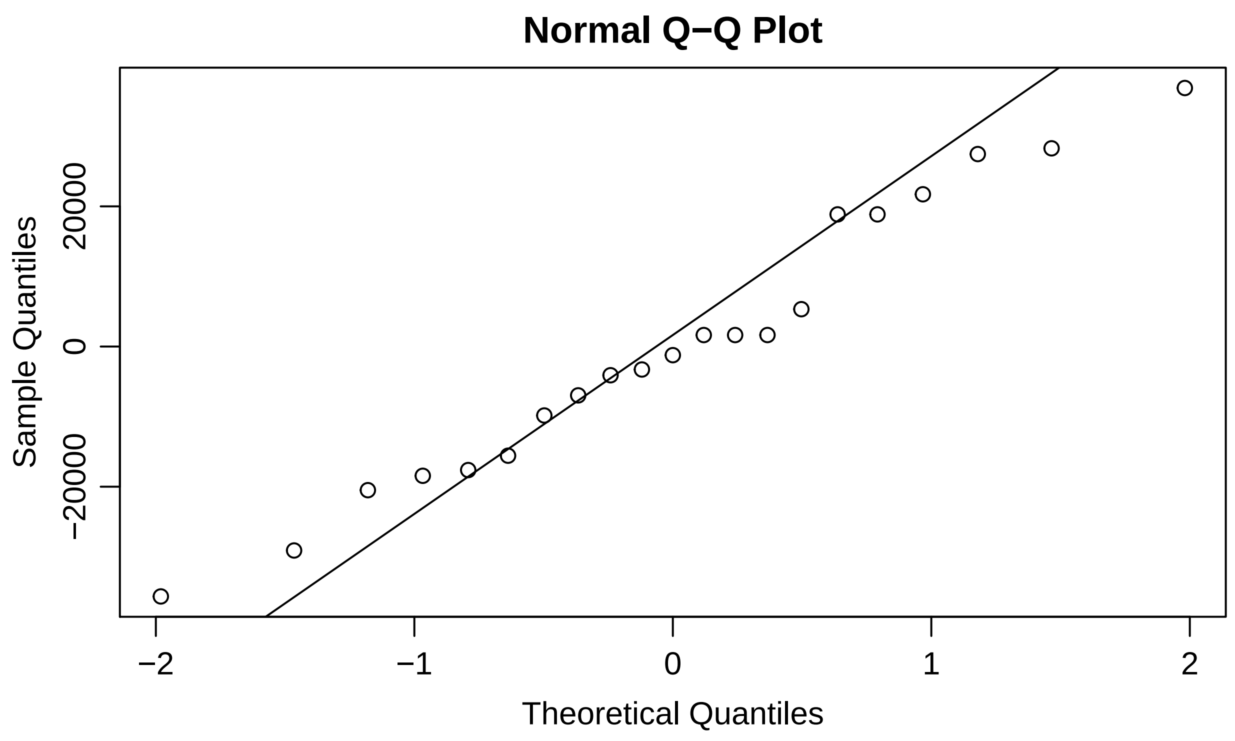


Figure 6.1: *Normal Q-Q plot to check the assumption of normal distribution of residuals.*

The next assumption to test is homogeneity of variance. Formal tests for homogeneity of variance include the Bartlett (parametric) and Fligner-Killeen (non-parametric) tests. These tests can be used in conjunction with box plots to determine the level of heterogeneity of variance between treatments.

```
bartlett.test(bean.09$population.4wk, bean.09$treatment)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  bean.09$population.4wk and bean.09$treatment
## Bartlett's K-squared = 3.7865, df = 6, p-value = 0.7055
```

```
fligner.test(bean.09$population.4wk, bean.09$treatment)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  bean.09$population.4wk and bean.09$treatment
## Fligner-Killeen:med chi-squared = 3.2494, df = 6, p-value = 0.777
```

Formal tests do not indicate a problem (p>0.7), and box plots (Figure 6.2) indicate that things are reasonable similar. Perhaps the handweeded and EPTC + ethalfluralin treatments have less variability than the others, but differences are not too dramatic, especially considering there were only 3 replicates per treatment in 2009.

```
par(mar=c(4.2,12,0.5,1.5), mgp=c(2,.7,0))
boxplot(bean.09$population.4wk/1000 ~ bean.09$treatment, horizontal=T,
        las=1, xlab="Bean plants per hectare (thousands)",
        ylab="")
```

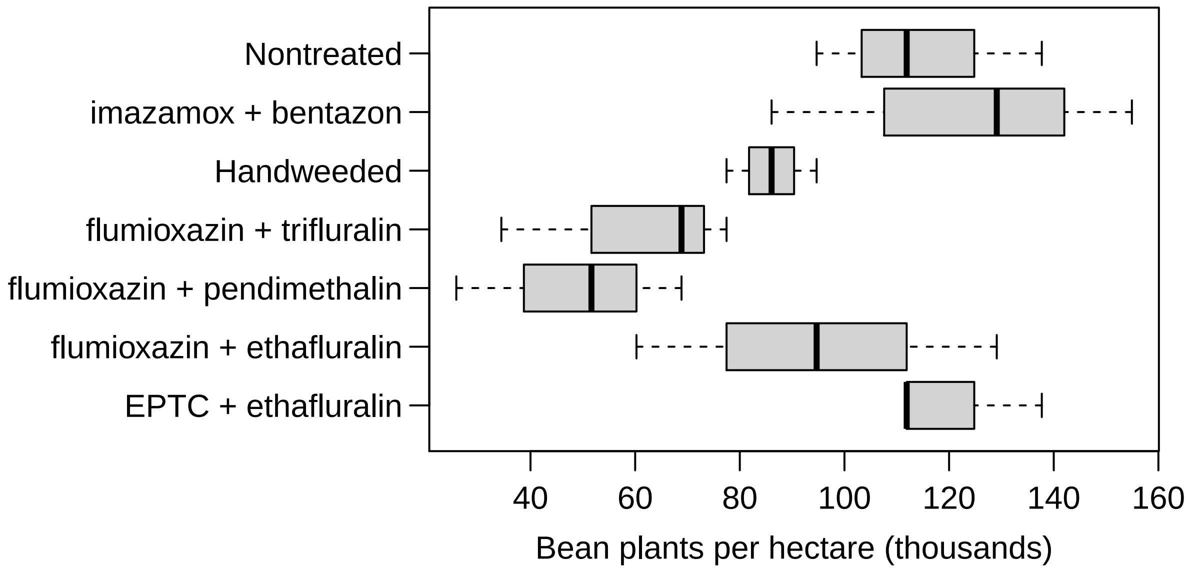


Figure 6.2: *Boxplots showing the effect of weed control treatment on dry edible bean plants per hectare.*

Simply plotting the aov fit in R [e.g. Figure 6.3] will provide a series of diagnostic plots as well. The first two plots are the residuals vs fitted values, and the Normal Q-Q plot, which can be plotted at the same time with the following code:

```
par(mar=c(4.2,4.2,2.2,1), mfrow=c(1,2), mgp=c(2,.7,0))
plot(b09.aov, which=1:2)
```

Since neither the normality nor homogeneity of variance assumptions were dramatically violated, and ANOVA is fairly robust to violations of these assumptions, it seems safe to proceed with interpretation of the ANOVA results. The third assumption (independence of observations) is not something we will test statistically; it is up to the researcher to ensure that they are collecting data in a way that does not violate this assumption.

Finally, after the assumptions have been checked, and remedial measures taken if necessary, we will want to know whether there is a significant effect of herbicide treatment on dry bean population density. Once again, we can use the `summary()` function along with the stored model to obtain this information. The resulting ANOVA table indicates that we have pretty strong evidence that there are differences in dry bean plants per acre due to herbicide treatment (p = 0.019).

```
summary(b09.aov)
```

```
##             Df    Sum Sq   Mean Sq F value Pr(>F)
## block        2 5.574e+08 2.787e+08   0.432 0.6589
## treatment    6 1.569e+10 2.615e+09   4.055 0.0188 *
## Residuals   12 7.740e+09 6.450e+08
```

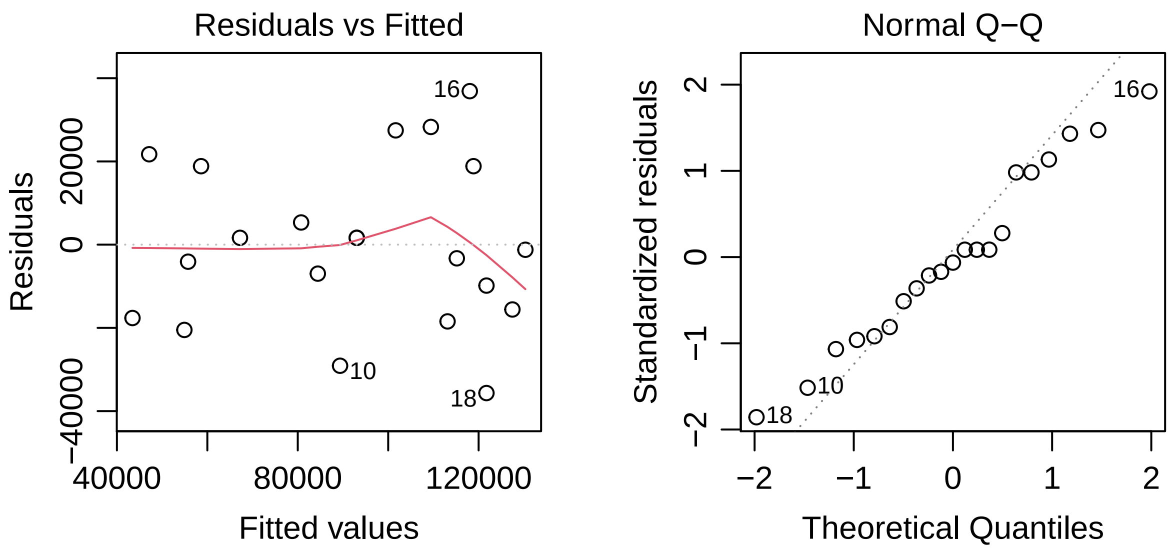


Figure 6.3: *Model diagnostic plots for the dry edible bean ANOVA model.*

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

6.1.2 Mean Separation - agricolae package

The `agricolae` package has several functions for mean separation, including the `LSD.test()` function that will compare means using Fisher's LSD. The LSD.test computes a lot of useful information that can be extracted for later use. You can use the `str(bean.lsd)` function to learn more about what is available. Below, we will look at the `$group` section which gives the treatment means and mean separation, and the `$statistics` section which provides the LSD value (among other things).

```
library(agricolae)
bean.lsd <- LSD.test(b09.aov, trt="treatment")
bean.lsd$group
```

```
##                              population.4wk groups
## imazamox + bentazon               123373.21      a
## EPTC + ethafluralin               120503.89      a
## Nontreated                        114766.08      a
## flumioxazin + ethafluralin         94682.51     ab
## Handweeded                         86074.56    abc
## flumioxazin + trifluralin          60251.53     bc
## flumioxazin + pendimethalin        48775.09      c
```

```
bean.lsd$statistics
```

```
##     MSerror Df     Mean       CV  t.value      LSD
##   645034519 12 92632.41 27.41754 2.178813 45182.03
```

To get standard deviations and confidence limits, we could type `bean.lsd$means`. The `agricolae` package can also calculate Tukey's HSD using the `HSD.test()` function. The syntax above is the same, simply changing `LSD.test` to `HSD.test`.

6.1.3 Mean Separation - emmeans and multcomp packages

The `emmeans` and `multcomp` packages are also useful for separating means, and has the advantage that it can be used for linear, nonlinear, and mixed effects models.

```
library(emmeans)
library(multcomp)
b09.emm <- emmeans(b09.aov, ~ treatment)
pairs(b09.emm) # prints out all Tukey-adjusted pairwise comparisons
multcomp::cld(b09.emm, Letters=LETTERS) # corresponding compact letter display (cld)
pairs(b09.emm, adjust="none") # prints out all unadjusted pairwise comparisons
multcomp::cld(b09.emm, Letters=LETTERS, adjust="none") # corresponding compact letter display (cld)
```

6.2 Multi-Factor ANOVA

We'll now look at the second and third year of the study, beginning by subsetting the data to include only these years.

```
bean2.dat <- bean.dat %>%
  filter(year==2010 | year==2011)
summary(bean2.dat)
```

```
##    year                           treatment block  population.4wk
##  2009: 0   EPTC + ethafluralin        :8   1:14   Min.   : 17216
##  2010:28   flumioxazin + ethafluralin :8   2:14   1st Qu.: 51645
##  2011:28   flumioxazin + pendimethalin:8   3:14   Median : 95761
##            flumioxazin + trifluralin  :8   4:14   Mean   : 86038
##            Handweeded                 :8          3rd Qu.:115665
##            imazamox + bentazon        :8          Max.   :163541
##            Nontreated                 :8
```

We'll now incorporate year as a fixed effect in the analysis. We're going to exclude the 2009 data because it does not have the same number of replicates as the other two years of data. The `aov()` function is not appropriate for unbalanced data. We will look at how to best handle this in a later section using mixed effects. We're also going to incorporate the blocking design into the model using the `Error()` argument. Blocking was used in each year of the field study (in a randomized complete block design). Because the trial was located in a different field each year, there is no reason to think that block 1 would result in a similar impact from one year to the next. Therefore, the effect of block is nested within year. We can specify this structure in the `aov()` function by using the `/` operator.

```
bean2.aov<-aov(population.4wk ~ year*treatment + Error(year/block),
               data=bean2.dat)
summary(bean2.aov)
```

```
##
## Error: year
##      Df   Sum Sq  Mean Sq
## year  1 5.46e+09 5.46e+09
##
## Error: year:block
##           Df    Sum Sq   Mean Sq F value Pr(>F)
## Residuals  6 1.363e+09 227216916
##
## Error: Within
##                  Df    Sum Sq   Mean Sq F value   Pr(>F)
## treatment       6 6.588e+10 1.098e+10  51.925 2.91e-16 ***
## year:treatment  6 5.422e+09 9.036e+08   4.273  0.00238 **
## Residuals      36 7.612e+09 2.114e+08
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Adding the Error term into the model results in 3 separate sections of the ANOVA table, corresponding to the three separate error terms: year, year:block, and within. In this case no F or P-values are calculated for the year effect. We could do that manually by dividing the Mean Square for year by the Mean Square for year:block, but it is clear just by looking at the numbers that the F value will be quite high (greater than 23, in fact). However, if we want `aov()` to calculate the P-value for us, we can specify the year*block error term by creating a new variable that contains the combined information. We will do this below using the `with()` function.

```
bean2.dat$yrblock  <- with(bean2.dat, factor(year):factor(block))
bean2.aov2<-aov(population.4wk ~ year*treatment + Error(yrblock),
                data=bean2.dat)
summary(bean2.aov2)
```

```
##
## Error: yrblock
##           Df    Sum Sq   Mean Sq F value  Pr(>F)
## year       1 5.460e+09 5.460e+09   24.03 0.00271 **
## Residuals  6 1.363e+09 2.272e+08
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: Within
##                  Df    Sum Sq   Mean Sq F value   Pr(>F)
## treatment       6 6.588e+10 1.098e+10  51.925 2.91e-16 ***
## year:treatment  6 5.422e+09 9.036e+08   4.273  0.00238 **
## Residuals      36 7.612e+09 2.114e+08
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The Mean Square for year and the year:block interaction are the same as before, as expected, but now the F and P-values are provided. However, this is not as important in this case, as there is a significant year by treatment interaction effect, making the main effect of year less interesting.

6.2.1 Mean Separation

Due to the different error terms, getting mean separation from the `LSD.test()` function in this case is not as straightforward as with only one factor. It can still be done by specifying the response, treatment, error degrees of freedom, and mean square error manually.

```
bean2.lsd<-LSD.test(bean2.dat$population.4wk,
                   with(bean2.dat, year:treatment),
                   36, 2.11e+08)
bean2.lsd$groups
```

```
##                                  bean2.dat$population.4wk groups
## 2010:imazamox + bentazon                        142023.15      a
## 2010:EPTC + ethafluralin                        139871.16      a
## 2010:Handweeded                                 124807.87     ab
## 2010:Nontreated                                 124807.87     ab
## 2011:Handweeded                                 104369.23     bc
## 2011:Nontreated                                  97375.43      c
## 2011:imazamox + bentazon                         95761.28      c
## 2011:EPTC + ethafluralin                         94147.76      c
## 2011:flumioxazin + trifluralin                   57026.74      d
## 2010:flumioxazin + trifluralin                   51644.61     de
## 2010:flumioxazin + pendimethalin                 49493.24     de
## 2011:flumioxazin + pendimethalin                 48956.64     de
## 2010:flumioxazin + ethafluralin                  38733.92     de
## 2011:flumioxazin + ethafluralin                  35506.87      e
```

Alternatively, we can use the `emmeans` package, to get the confidence intervals and pairwise comparisons.

```
emmeans(bean2.aov2, specs = pairwise ~ year:treatment)
```

It seems there is general trend for treatments containing flumioxazin to have fewer bean plants. But having the different years mixed together makes it difficult to see, so an alternative approach would be to simply calculate the means using `tapply()` and print out the LSD value.

```
round(tapply(bean2.dat$population.4wk,
             list(bean2.dat$treatment, bean2.dat$year), mean), -3)
```

```
##                              2009   2010   2011
## EPTC + ethafluralin            NA 140000  94000
## flumioxazin + ethafluralin     NA  39000  36000
## flumioxazin + pendimethalin    NA  49000  49000
## flumioxazin + trifluralin      NA  52000  57000
## Handweeded                     NA 125000 104000
## imazamox + bentazon            NA 142000  96000
## Nontreated                     NA 125000  97000
```

```
bean2.lsd$statistics
```

```
##    MSerror Df     Mean       CV  t.value     LSD
##   2.11e+08 36 86037.55 16.88314 2.028094 20831.2
```

Looking at the data in this way makes it appear that the year interaction was due to a higher bean population in 2010 compared to 2011 for most treatments. However, the flumioxazin treatments had similar bean populations in both years, much reduced compared to the nontreated or handweeded control treatments. The LSD value of 0 allows us to compare the same treatment between years, or compare between treatments within a year.

Chapter 7

Linear Regression

7.1 Simple Linear Regression

Linear regression in R is very similar to analysis of variance. In fact, the mathematics behind simple linear regression and a one-way analysis of variance are basically the same. The main difference is that we use ANOVA when our treatments are unstructured (say, comparing 5 different pesticides or fertilizers), and we use regression when we want to evaluate structured treatments (say, evaluating 5 different rates of the same pesticide or fertilizer). In linear regression, we will use the `lm()` function instead of the `aov()` function that was used in the ANOVA chapter. However, it should be noted that the `lm()` function can also be used to conduct ANOVA. The primary advantage of `aov()` for ANOVA is the ability to incorporate multi-strata error terms (like for split-plot designs) easily.

To illustrate linear regression in R, we will use data from (Kniss et al. 2012), quantifying the sugar yield of sugarbeet in response to volunteer corn density. The response variable is sucrose production in pounds/acre (LbsSucA), and the independent variable is volunteer corn density in plants per foot of row. The `lm()` function syntax (for the purpose of regression) requires a formula relating the response variable (Y) to independent variable(s), separated by a `~`, for example: `lm(Y ~ X)`.

```
beets.dat <- read.csv("http://rstats4ag.org/data/beetvolcorndensity.csv")
head(beets.dat)
```

```
##    Loc Density  LbsSucA
## 1 WY09    0.00 10106.34
## 2 WY09    0.03  8639.42
## 3 WY09    0.05  7752.20
## 4 WY09    0.08  5718.38
## 5 WY09    0.11  7953.72
## 6 WY09    0.15  6012.16
```

```
density.lm <- lm(LbsSucA ~ Density, data = beets.dat)
```

7.1.1 Model Diagnostics

The assumptions can be checked in the same manner as with the fitted aov object (see ANOVA section for full details). Quick diagnostic plots can be obtained by plotting the fitted lm object.

```
par(mfrow=c(1,2), mgp=c(2,.7,0), mar=c(3.2,3.2,2,.5))
plot(density.lm, which=1:2)
```

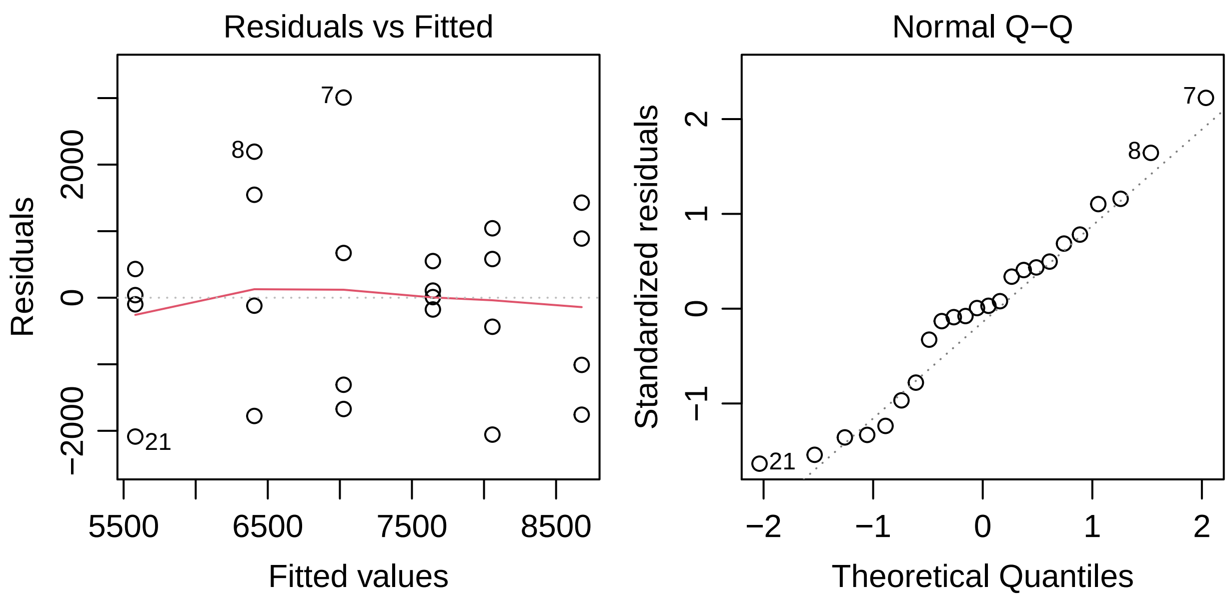


Figure 7.1: *Model diagnostic plots for linear model of sugarbeet yield as a function of volunteer corn density.*

We can also conduct a lack-of-fit test to be sure a linear model is appropriate. The simplest way to do this is to fit a linear model that removes the linear structure of the independent variable (treating it as a factor variable). The models can then be compared using the `anova()` function.

```
density.lm2 <- lm(LbsSucA ~ as.factor(Density), data = beets.dat)
anova(density.lm, density.lm2)
```

```
## Analysis of Variance Table
##
## Model 1: LbsSucA ~ Density
## Model 2: LbsSucA ~ as.factor(Density)
##   Res.Df      RSS Df Sum of Sq      F Pr(>F)
## 1     22 42066424
## 2     18 40058981  4   2007444 0.2255 0.9206
```

When 2 linear models are provided to the `anova()` function, the models will be compared using an F-test. This is only appropriate when the models are nested; that is, one model is a reduced form of the other. The highly non-significant p-value above ($p = 0.92$) indicates that removing the structured nature of the density variable does not increase the model fit. Therefore, we will proceed with the regression analysis.

7.1.2 Model Interpretation

Using the `anova()` function on a a single fitted lm object produces output similar to the `aov()` function. The ANOVA table indicates strong evidence that volunteer corn density has an effect on the sugar production, as expected. To get the regression equation, we use the `summary()` function on the fitted lm model:

```r
anova(density.lm)
```

```
## Analysis of Variance Table
##
## Response: LbsSucA
##           Df   Sum Sq  Mean Sq F value   Pr(>F)
## Density    1 25572276 25572276  13.374 0.001387 **
## Residuals 22 42066424  1912110
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(density.lm)
```

```
##
## Call:
## lm(formula = LbsSucA ~ Density, data = beets.dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2086.02 -1084.01    23.92   726.23  3007.73
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8677.6      485.6  17.869 1.38e-14 ***
## Density     -20644.7     5645.2  -3.657  0.00139 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1383 on 22 degrees of freedom
## Multiple R-squared:  0.3781, Adjusted R-squared:  0.3498
## F-statistic: 13.37 on 1 and 22 DF,  p-value: 0.001387
```

The resulting output provides estimates for the intercept (8678), and slope (-20645) of the regression line under the heading "Coefficients." The `summary()` function also prints standard errors, t-statistics, and P-values, and fit statistics such as the R^2 for these estimates are also provided. The regression line for this example would be:

Y = 8678 - 20645*X

To plot the data along with the fitted line:

```r
par(mfrow=c(1,1), mar=c(3.2,3.2,2,.5), mgp=c(2,.7,0))
plot(beets.dat$LbsSucA ~ beets.dat$Density, bty="l",
     ylab="Sugar yield (lbs/A)", xlab="Volunteer corn density (plants/ft row)",
     main="Lingle, WY 2009", ylim=c(0,10000))
  abline(density.lm) # Add the regression line
# to add the regression equation into the plot:
  int<-round(summary(density.lm)$coefficients[[1]]) # get intercept
  sl<- round(summary(density.lm)$coefficients[[2]]) # get slope
  reg.eq<-paste("Y =", int, sl, "* X") # create text regression equation
  legend("bottomleft",reg.eq, bty="n")
```

The sugar yield was decreased by 20,600 lbs/A for each volunteer corn plant per foot of sugarbeet row; this seems unintuitive since the Y-intercept was only 8,678 lbs/A. The maximum volunteer corn density used in

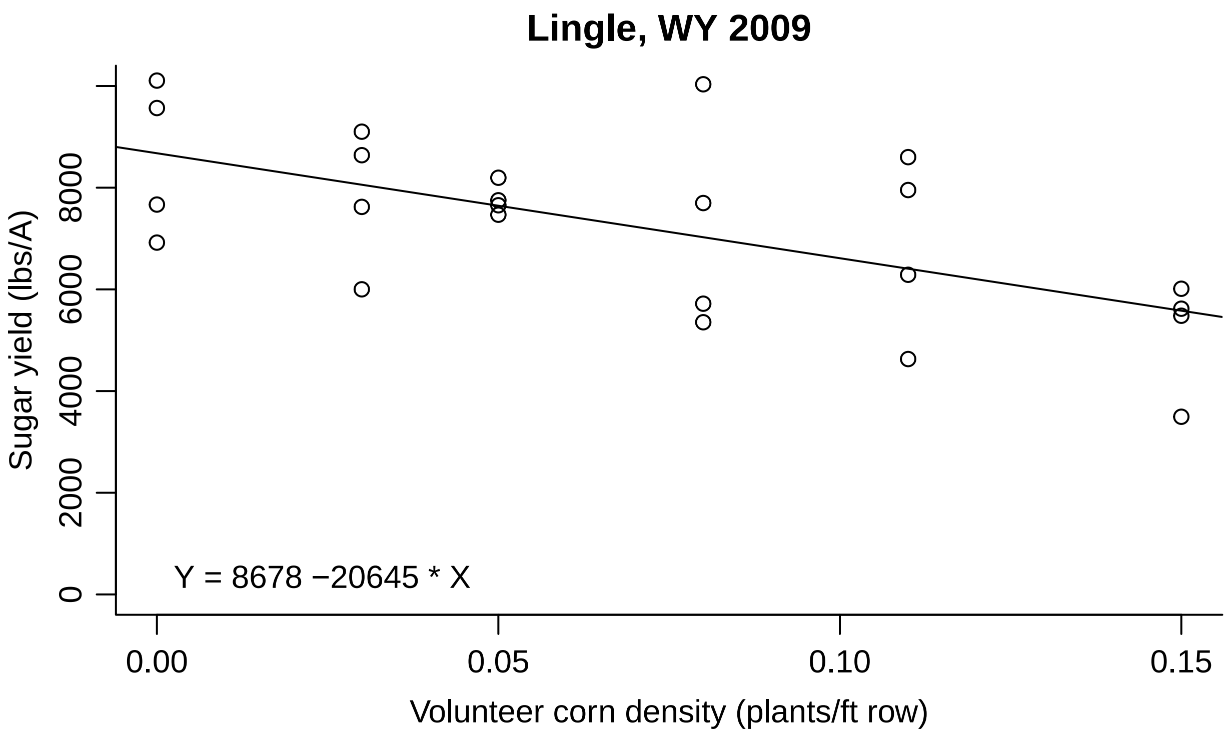


Figure 7.2: *Effect of volunteer corn density on sugarbeet yield.*

the study was only 0.15 plants per foot of sugarbeet row, so the units of the slope are not directly applicable to the data observed in the study. Although the linear regression model is appropriate here, it is extremely important *not* to extrapolate the linear model results to volunteer corn densities outside the observed data range (densities greater than 0.15 plants per foot of row). This can be illustrated by changing the Y and X limits on the figure. If extrapolation is absolutely necessary, a nonlinear model would be more appropriate (although still not ideal). The ideal situation would be to repeat the study with weed densities that caused crop yield to drop close to the minimum yield.

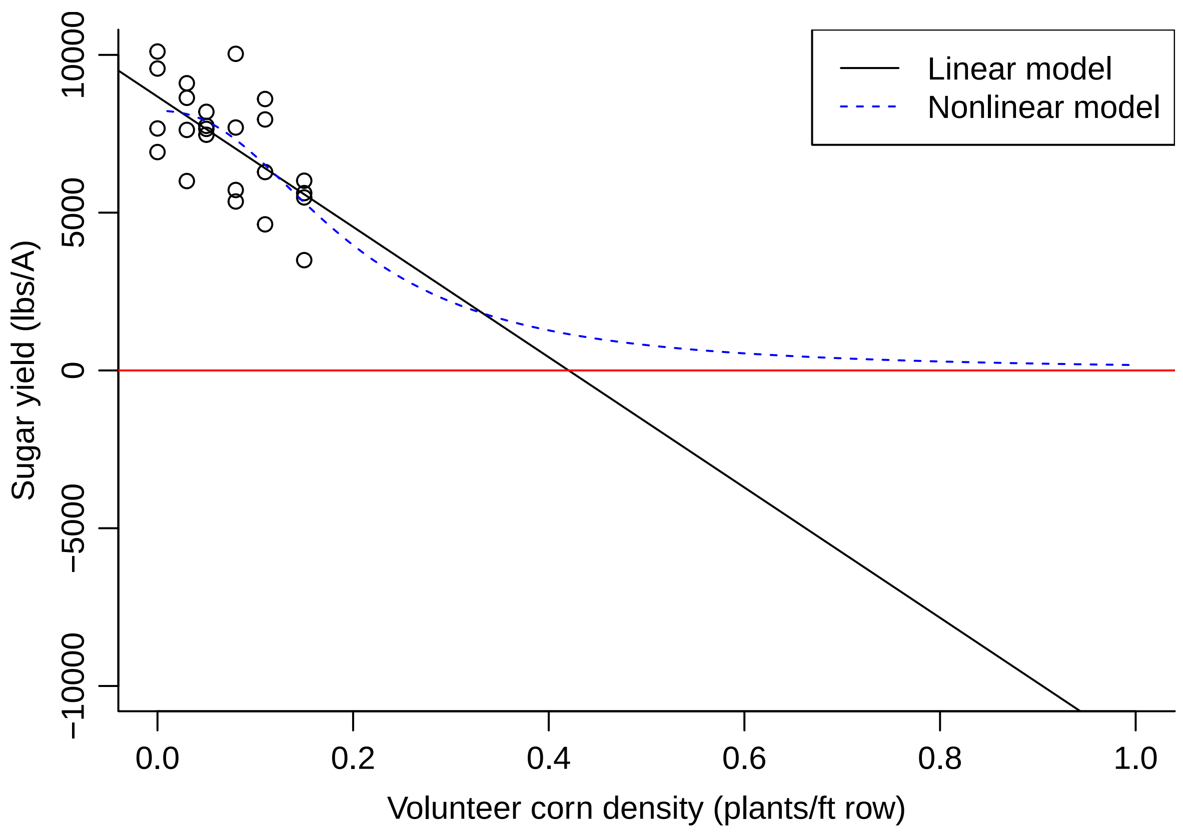


Figure 7.3: *Linear model vs nonlinear model for sugarbeet response to volunteer corn density.*

Chapter 8

Analysis of Covariance (ANCOVA)

In an ANOVA the interest lies in the differences among means. In a linear regression the interest lies in the intercept and slope parameters of regression lines, or perhaps other parameters of biological interest, e.g. asymptotic and effective doses (e.g, $latexLD_{50}$ levels in the nonlinear case. The combination of ANOVA and regression is called Analysis of Covariance (ANCOVA). It is an underutilized method that makes sequential tests for testing various hypotheses. Usually it is confined to linear regressions but can be equally powerful when used in nonlinear relationships. However, in this section we only consider the linear ANCOVA.

8.1 Example 1: Mechanical Weed Control

A mechanical weed control experiment (Rasmussen et al. 2008] tested the effect of intensity of harrowing (1, 2, 3, or 4 times), either along or across the direction of sowing (factors) on the weed cover left after the harrowing.

```
Harrowing<- read.csv2("http://rstats4ag.org/data/Harrowing.csv")
head(Harrowing,3)
```

```
##   plot       leaf    weeds int direction yield
## 1    1 0.18623052 103.33763   0     along  4.21
## 2    2 0.12696533  54.02022   1     along  4.08
## 3    3 0.09006582  23.44035   2     along  3.92
```

Lets look at the relationships in Figure 8.1.

```
library(lattice)
xyplot(weeds ~ int | direction, data = Harrowing,
       ylab="Weed Cover",xlab="Intensity of Harrowing"  )
```

The relationships above does not look very linear, it looks like an exponential decreasing leaf cover. If this assumption is correct we can make the relationship linear by taking the logarithm of weed cover.

```
xyplot(log(weeds) ~ int | direction, data = Harrowing,
       ylab="Log(Weed Cover)",xlab="Intensity of Harrowing",
       panel = function(x,y)
       {panel.xyplot(x,y)
        panel.lmline(x,y)})
```

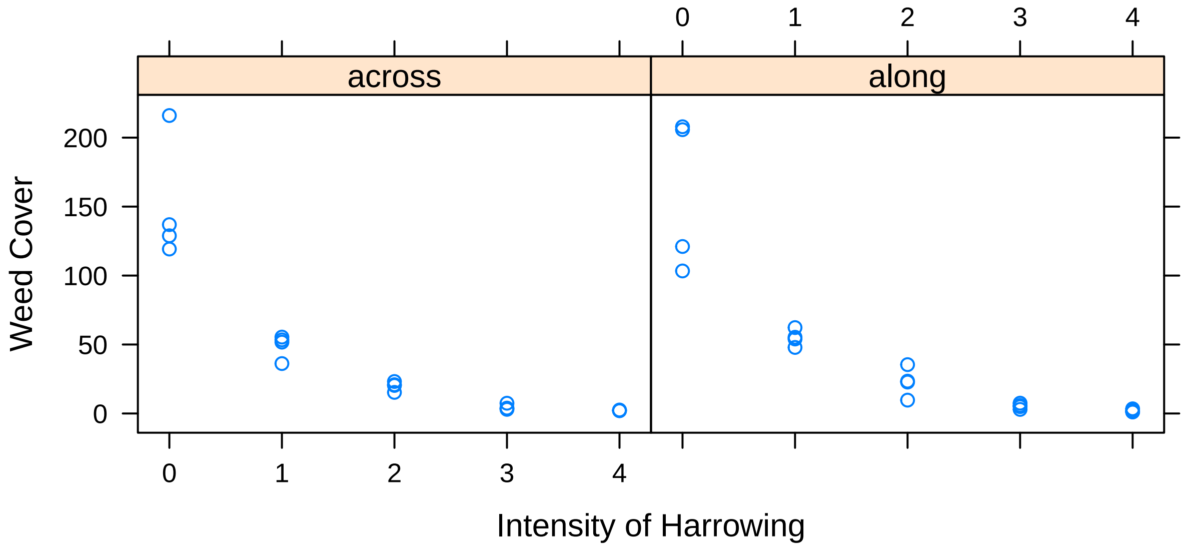


Figure 8.1: *The relationship between weed cover and intensity of harrowing either across or along the direction of sowing.*

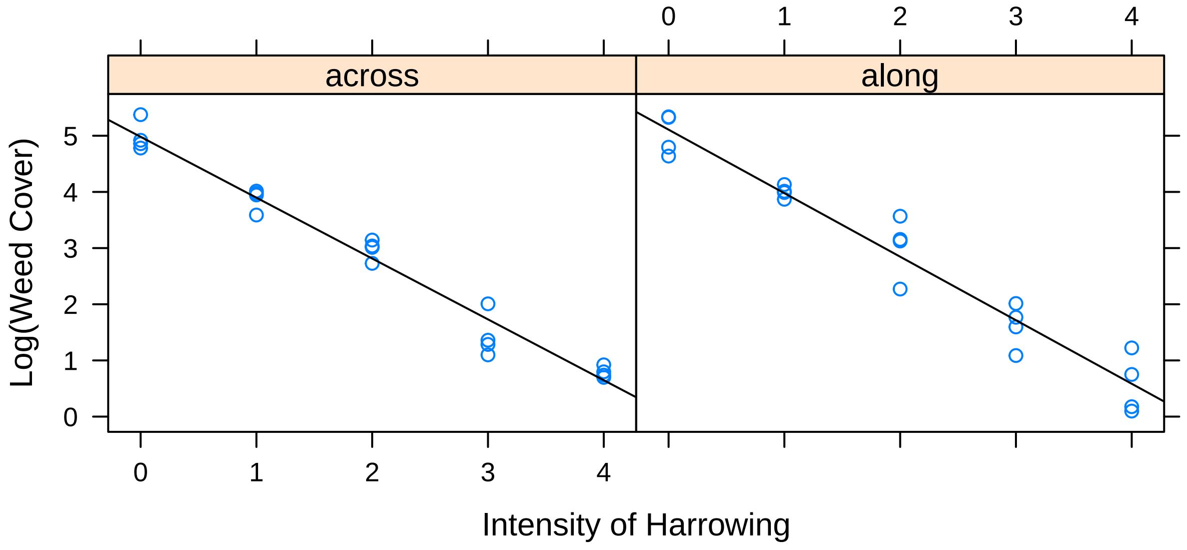


Figure 8.2: *Same data as in 8.1 using the logarithm of weed cover (y-axis) gives a relationship that looks linear on the intensity of harrowing within the directions of harrowing.*

Apparently, the assumption of a straight line relationship looks good, but we have to do statistics to substantiate this hypothesis. It is of interest to know whether the harrowing direction (across or along the rows) had the same effect on weed cover. Statistically, we want to know if the two regression lines are similar in term of regression slopes. This can be done by sequential testing:

- Ordinary ANOVA where `int` is turned into a factor `factor(int)`
- ANCOVA with interaction; meaning that the regression lines can have both different slopes and intercepts

```
Harrowing$Log.weeds <- log(Harrowing$weeds)
m1 <- lm(Log.weeds ~ factor(int) * direction, data = Harrowing)
m2 <- lm(Log.weeds ~ int * direction, data = Harrowing)
anova(m1, m2)
```

```
## Analysis of Variance Table
##
## Model 1: Log.weeds ~ factor(int) * direction
## Model 2: Log.weeds ~ int * direction
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     30 3.5350
## 2     36 4.2754 -6  -0.74048 1.0474 0.4152
```

`m1`, an ordinary ANOVA, is the most general model with no assumption of any particular relationship. `m2`is the ANCOCVA, which assumes there is a linear relationship between harrowing direction and intensity. The `int * direction` in `m2` means that there is an interaction between the slopes depending on the direction of harrowing. Or in other words the slopes of the regression lines are different. The test for lack of fit is not significant (p=0.42) so we can assume a linear relationship applies to the log-transformed response variable.

We can further refine the analysis through sequential testing: * Assuming similar slope but different intercept depending of direction of harrowing * Assuming that the directions of harrowing does not influence regression lines, e.i, similar line

To do this, we first test the regression model with an interaction term, `m2`, with a regression model where we assume no interaction,`m3`, which mean the slopes of the regression lines are similar:

```
m3 <- lm(Log.weeds ~ int + direction, data = Harrowing)
anova(m2,m3)
```

```
## Analysis of Variance Table
##
## Model 1: Log.weeds ~ int * direction
## Model 2: Log.weeds ~ int + direction
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     36 4.2754
## 2     37 4.3207 -1 -0.045295 0.3814 0.5407
```

The lack of fit test is again not significant so we can assume that there is no interaction, and conclude that the regression slopes are similar. The final test will determine whether the direction of harrowing influences the relationship between log(Weed Cover) and intensity of harrowing?

```
m4 <- lm(Log.weeds ~ int, data = Harrowing)
anova(m4, m3)
```

```
## Analysis of Variance Table
##
## Model 1: Log.weeds ~ int
## Model 2: Log.weeds ~ int + direction
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     38 4.3311
## 2     37 4.3207  1  0.010344 0.0886 0.7677
```

Again, no significance here either (p=0.77); so we end up from the most general model (ANOVA) to the most simple ANCOVA: direction of harrowing does not matter and the relationship between `log(weeds)`and intensity of harrowing is linear.

```
summary(m4)
```

```
##
## Call:
## lm(formula = Log.weeds ~ int, data = Harrowing)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.63622 -0.25226  0.06307  0.28256  0.73686
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.04478    0.09246   54.56   <2e-16 ***
## int         -1.10710    0.03775  -29.33   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3376 on 38 degrees of freedom
## Multiple R-squared:  0.9577, Adjusted R-squared:  0.9566
## F-statistic: 860.3 on 1 and 38 DF,  p-value: < 2.2e-16
```

In order to illustrate the result we use the code below to create Figure 8.3. For illustration purposes we have separated the means into direction across and along even though it is not necessary.

```
library(dplyr)
Plot.means <- Harrowing %>%
  group_by(int, direction) %>%
  summarise(count = n(), Log.Weeds = mean(Log.weeds))

par(mgp=c(2,.7,0), mar=c(3.2,3.2,.5,.5))
plot(Log.Weeds~int, data = Plot.means, bty="l",
     pch = as.numeric(direction),
     xlab="Harrowing intensity",
     ylab="Log(weed cover)")
```

```
## Warning in FUN(X[[i]], ...): NAs introduced by coercion
```

```
abline(m4)
legend("bottomleft", c("Across", "Along"), pch = c(2,1))
```

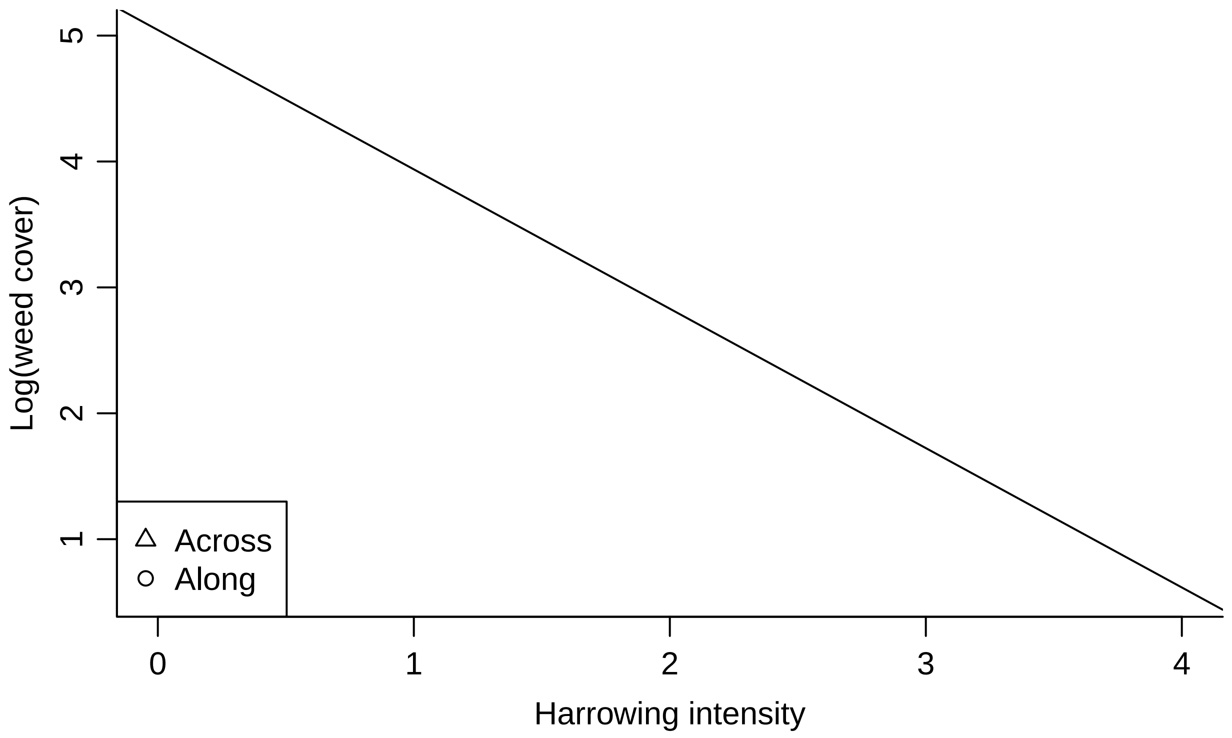


Figure 8.3: *Summary plot of the harrowing experiments. There is no difference between the direction of harrowing and, therefore, only one regression line is necessary.*

The slope is -1.11 (0.04) which means that with an increase in harrowing intensity the logarithm of weed cover decreases by -1.11. In fact the relationship is an exponential decay curve where the relative rate of change is exp(-1.11)= 0.3296 wherever we are on the curve.

8.2 Example 2: Duration of competition

In 2006 and 2007, field studies were conducted near Powell, Wyoming to determine the effects of Salvia reflexa (lanceleaf sage) interference with sugarbeet. This data was published in Odero et al. 2010. Salvia reflexa was seeded alongside the sugarbeet crop at a constant density, and allowed to compete for time periods ranging from 36 to 133 days after sugarbeet emergence. Yields (`Yield_Mg.ha`) were determined at the end of the season.

```
lanceleaf.sage<-read.csv("http://rstats4ag.org/data/lanceleafsage.csv")
head(lanceleaf.sage,n=3)
```

```
##   Replicate Duration Year Yield_Mg.ha Yield.loss.pct
## 1         1       36 2006       67.26          98.05
## 2         1       50 2006       60.34          87.96
## 3         1       64 2006       53.69          78.27
```

```
library(lattice)
xyplot(Yield_Mg.ha~Duration|factor(Year),data=lanceleaf.sage,
        ylab= "Sugarbeet root yield (t/ha)",
        xlab="Duration of competition from _Salvia reflexa_",
        panel = function(x,y)
       {panel.xyplot(x,y)
        panel.lmline(x,y)})
```

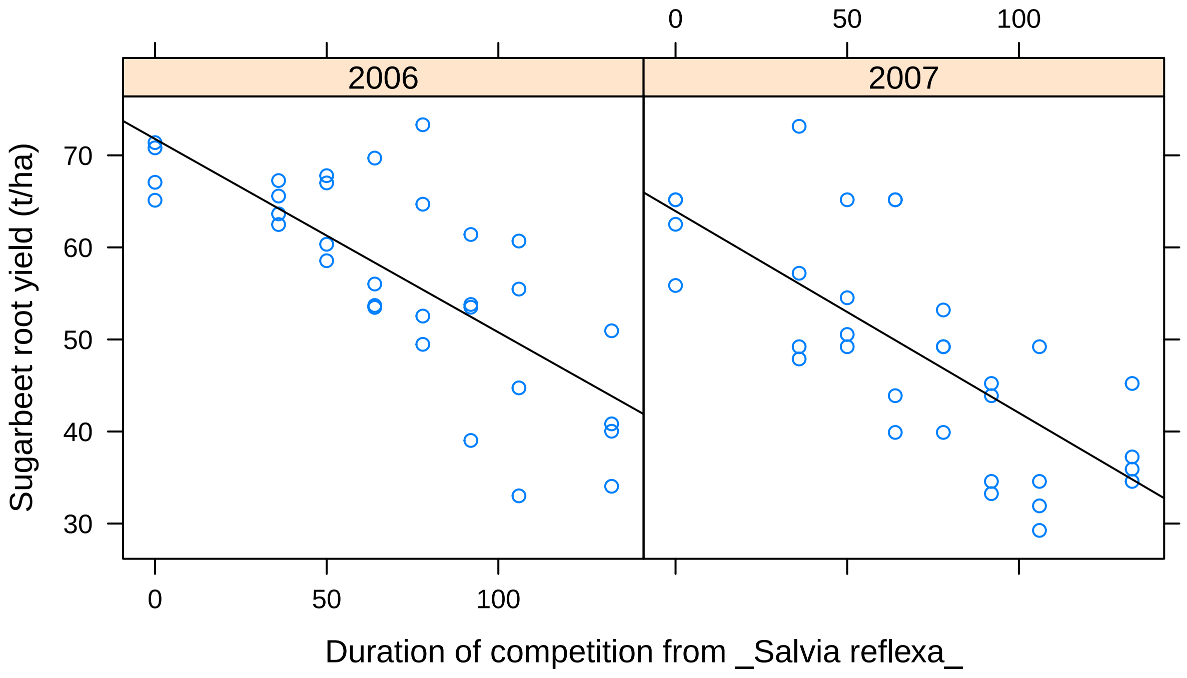


Figure 8.4: *Duration of* Salvia reflexa *competition effect on yield of sugar beet in two years.*

As was the case with the harrowing experiment we do sequential testing. First, we compare the ANCOVA with the most general model, an AVOVA:

```
lm1 <- lm(Yield_Mg.ha ~ factor(Duration) * factor(Year), data = lanceleaf.sage) # ANOVA
lm2 <- lm(Yield_Mg.ha ~ Duration * factor(Year), data = lanceleaf.sage) # Regression
anova(lm1, lm2) # Lack of fit test
```

```
## Analysis of Variance Table
##
## Model 1: Yield_Mg.ha ~ factor(Duration) * factor(Year)
## Model 2: Yield_Mg.ha ~ Duration * factor(Year)
##   Res.Df    RSS  Df Sum of Sq      F Pr(>F)
## 1     48 3198.3
## 2     60 3667.1 -12   -468.78 0.5863 0.8423
```

The test for lack of fit shows that we can safely assume a linear regression (p=0.84). The next step is to test if we can assume similar slopes for the regression in the two years.

```
lm3 <- lm(Yield_Mg.ha ~ Duration + factor(Year), data = lanceleaf.sage)
anova(lm2, lm3)
```

```
## Analysis of Variance Table
##
## Model 1: Yield_Mg.ha ~ Duration * factor(Year)
## Model 2: Yield_Mg.ha ~ Duration + factor(Year)
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     60 3667.1
## 2     61 3669.1 -1   -2.0366 0.0333 0.8558
```

Again we can confidently assume that the slopes are the same irrespective of year (p=0.86). The next question: are the regression lines identical?

```
lm4 <- lm(Yield_Mg.ha ~ Duration, data = lanceleaf.sage)
anova(lm4, lm3)
```

```
## Analysis of Variance Table
##
## Model 1: Yield_Mg.ha ~ Duration
## Model 2: Yield_Mg.ha ~ Duration + factor(Year)
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     62 4819.3
## 2     61 3669.1  1    1150.2 19.123 4.862e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We get a extremely low p value (p<0.0001). Therefore, the reduced model significantly reduces the fit. In other words, the `lm3` model is the model that should be used to summarize the experiment.

```
summary(lm3)
```

```
##
## Call:
## lm(formula = Yield_Mg.ha ~ Duration + factor(Year), data = lanceleaf.sage)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.3477  -5.2428  -0.7246   3.5724  17.9560
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      72.09580    2.20523  32.693  < 2e-16 ***
## Duration         -0.21451    0.02472  -8.678 3.02e-12 ***
## factor(Year)2007 -8.47875    1.93890  -4.373 4.86e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.756 on 61 degrees of freedom
## Multiple R-squared:  0.6075, Adjusted R-squared:  0.5947
## F-statistic: 47.21 on 2 and 61 DF,  p-value: 4.081e-13
```

“‘

The slopes are similar but not the intercepts and the summary of the experiment can be neatly presented graphically in Figure 8.5.

```
library(dplyr)
averages<-lanceleaf.sage %>%
  group_by(Duration, Year) %>%
  summarise(count = n(), YIELD = mean(Yield_Mg.ha))

par(mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(YIELD ~ Duration, data = averages, bty="l",
     pch = as.numeric(factor(Year)), ylim=c(10, 80),
     ylab="Sugarbeet root yield (t/ha)",
     xlab="Duration of competition from Salvia reflexa")
x <- c(0, 300)#dummy intensities
lines(x, predict(lm3, newdata = data.frame(Year = "2006",
                                           Duration = x)), lty = 2)
lines(x, predict(lm3, newdata = data.frame(Year = "2007",
                                             Duration = x)), lty = 1)
legend("topright", c("2006", "2007"), lty = c(2, 1),
       pch = c(1,2),   merge = TRUE)
```

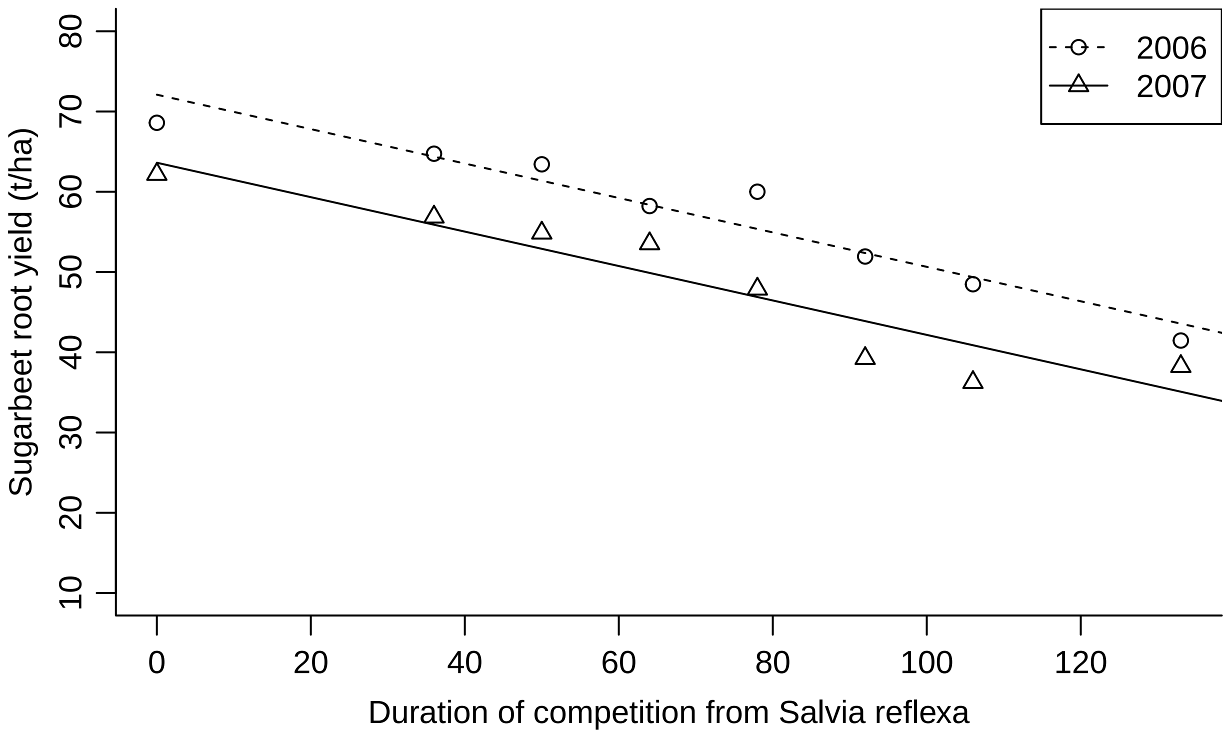


Figure 8.5: *Sugarbeet yield in response to duration of* Salvia reflexa *competition.*

Note the codes to obtain the regression lines for each year are more involved that in the harrowing example in Figure 8.3. You must separate the regression lines on the basis of `Year` and define an independent variable `x`. Since it is a strait line you only need two ‘x’s within year to draw the lines.

Even though the maximum yield (y-intercept) was different between years (72.1 in 2006 compared with 63.6 in 2007), yield reduction was similar in response to duration of weed competition. Each day of competition reduced the yield by 0.21 tons/ha (slope= -0.215 (0.025)).

Chapter 9

Mixed Models - ANOVA

There are many examples in agronomy and weed science where mixed effects models are appropriate. In ANOVA, everything except the intentional (fixed) treatment(s), reflect random variation. This includes soil variability, experimental locations, benches in the greenhouse, weather patterns between years; many things can affect experimental results that simply cannot be controlled. In many cases, we can try to account for these sources of variation by blocking. We can consider the selection of the blocks to take place at random in the sense that we usually cannot tell in advance whether or not the next block will exhibit a low, high, or more moderate response level. In this context, we can describe random effects as effects that cannot be controlled and therefore cannot be explained by the treatment structure of the experiment. In many designed experiments, we are not inherently interested in these random effects, but in an adequate analysis we need to acknowledge the variation that they contribute. We can do so using a mixed effects model that contains both fixed and random effects.

To illustrate mixed effects ANOVA, we will use the same dry bean herbicide data set that was used in the ANOVA section to allow for comparison; please read the ANOVA section for details on the study. We will need to load the `lme4` and `multcomp` packages for this analysis.

```r
library(lme4)
read.csv("http://rstats4ag.org/data/FlumiBeans.csv")->flum.dat
flum.dat$yrblock  <- with(flum.dat, factor(year):factor(block))
```

First, we can look at group means (in plants per acre, and plants per hectare) using the `dplyr` package.

```r
library(dplyr)
beansum <- flum.dat %>%
  group_by(year, treatment) %>%
  summarize( Pop.plantsA  = round(mean(population.4wk),-2),
             Pop.plantsHa = round(mean(population.4wk*2.47),-2))
beansum
```

```
## # A tibble: 21 x 4
## # Groups:   year [3]
##     year treatment                    Pop.plantsA Pop.plantsHa
##    <int> <chr>                              <dbl>        <dbl>
##  1  2009 EPTC + ethafluralin                48800       120500
##  2  2009 flumioxazin + ethafluralin         38300        94700
##  3  2009 flumioxazin + pendimethalin        19700        48800
##  4  2009 flumioxazin + trifluralin          24400        60300
##  5  2009 Handweeded                        34800        86100
##  6  2009 imazamox + bentazon               49900       123400
##  7  2009 Nontreated                        46500       114800
##  8  2010 EPTC + ethafluralin               56600       139900
##  9  2010 flumioxazin + ethafluralin        15700        38700
## 10  2010 flumioxazin + pendimethalin       20000        49500
## # ... with 11 more rows
```

9.1 Mixed Effects Model using the lme4 Package

In the ANOVA section, we considered year, block, and treatment all as fixed effects. However, because the number of replicates was different by year, analyzing the combined data from all three years is problematic. The effect of year is unbalanced; we have more observations for 2010 and 2011 than for 2009. We can deal with the unbalanced nature of the data by using a mixed effects model. There are other good reasons for analyzing the data as a mixed effect model. The hope is that the results of this experiment can be generalized beyond the three years the study was conducted. The three years in which we conducted the experiment represent 3 possible years out of many when the experiment could have been done, and therefore, we can assume the impact of year to be a random variable. Same goes for the blocking criteria within a year. However, the treatments were selected specifically to determine their impact on the dry bean crop. Therefore, we will consider the treatments as fixed effects, but the year and blocking criteria within a year random effects.

```
# Mixed Effects ANOVA
lmer(population.4wk ~ treatment+(1|year/block), data=flum.dat, REML=F)->flum.lmer
```

```
## boundary (singular) fit: see ?isSingular
```

```
anova(flum.lmer)
```

```
## Analysis of Variance Table
##           npar     Sum Sq    Mean Sq F value
## treatment    6 1.1974e+10 1995718507  28.887
```

9.1.1 Model Comparison and Obtaining P-values

One of the most frustrating things to many researchers analyzing mixed models in R is a lack of p-values provided by default. The calculation of P-values for complex models with random effects and multiple experimental unit sizes is not a trivial matter. Certain assumptions must be made, and it is difficult to generalize these assumptions from experiment to experiment. Therefore, the authors of the `lme4` package have chosen not to print P-values by default. We can get a P-value for the fixed effect of treatment in two different ways. First, we can compare our model with a reduced model that does not contain the treatment effect. The P-value that results, then, is effectively the P-value for the treatment effect. When comparing `lmer()` models, it is important to specify the argument `REML=F` in both models so that the model is fit using full maximum likelihood (rather than the default restricted maximum likelihood).

```
lmer(population.4wk ~ (1|year/block), data=flum.dat, REML=F)->flum.null # reduced model
anova(flum.null, flum.lmer) # anova function to compare the full and reduced models
```

```
## Data: flum.dat
## Models:
```

```
## flum.null: population.4wk ~ (1 | year/block)
## flum.lmer: population.4wk ~ treatment + (1 | year/block)
##            npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## flum.null     4 1710.4 1719.8 -851.20   1702.4
## flum.lmer    10 1633.0 1656.5 -806.52   1613.0 89.359  6  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A second method for getting a p-value for fixed effects is to use the `Anova()` (note the different capitalization) from the `car` package. This will provide an anova table of the full model that is complete with p-values. By deault, the `Anova()` function will provide p-values from a Chi-square test.

```
# Mixed Effects ANOVA - with p-values from Chisq test:
library(car)
Anova(flum.lmer) # Chi-square
```

```
## Analysis of Deviance Table (Type II Wald chisquare tests)
##
## Response: population.4wk
##            Chisq Df Pr(>Chisq)
## treatment 173.32  6  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To get a p-value from an F-test, the model must be fit using restricted maximum likelihood.

```
# re-fit the model with REML, and get F-test:
lmer(population.4wk ~ treatment+(1|year/block),
     data=flum.dat, REML=T)->flumREML.lmer #note REML = TRUE
Anova(flumREML.lmer, test="F") # F-test
```

```
## Analysis of Deviance Table (Type II Wald F tests with Kenward-Roger df)
##
## Response: population.4wk
##                F Df Df.res    Pr(>F)
## treatment 26.537  6     60 3.518e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For this example, treatment effect produces a very low p-valueregardless of method we use to get it, indicating there appears to be a strong effect of treatment on dry bean emergence Therefore, we will want to look at the treatment means to determine which treatments affected dry bean stand. First, though, it is often of interest to look at the random effects.

9.1.2 Random Effects

The `summary()` function can be used to print most of the relevant information from the mixed model fit `summary(flum.lmer)`. We can selectively print only the certain parts of the model fit. Adding `$varcor` to the summary function of the fit will print out the variance components for the random terms as well as the residual variance.

```
summary(flum.lmer)$varcor
```

```
##  Groups     Name        Std.Dev.
##  block:year (Intercept)    0.0
##  year       (Intercept) 3145.8
##  Residual               8311.9
```

The random 'year' effect appears to be contributing substantially to the model, but the blocking criteria within a year does not. When reporting results of a random effects model for publication, it is important that these results be included so that a full accounting of variability is provided. The `ranef()` function returns estimates for the random effects of year:

```
ranef(flum.lmer)$year
```

```
##      (Intercept)
## 2009    1364.249
## 2010    2517.687
## 2011   -3881.935
```

On average, bean populations were lowest in 2011, and highest in 2010. Using `tapply()` results in a similar conclusion, and helps put the random effects estimates into context.

```
tapply(flum.dat$population.4wk, flum.dat$year, mean)
```

```
##     2009     2010     2011
## 37503.00 38830.64 30835.39
```

9.1.3 Fixed Effects & Mean Separation

In most designed agricultural experiments, the fixed effects (and specifically differences between fixed effects treatments) are of most interest to the researcher. We can use the `glht()` function in the `multcomp` package to separate treatment means if we conclude there is a significant treatment effect. The p-values below are adjusted using Tukey's method.

```
library(multcomp)
summary(glht(flum.lmer, linfct=mcp(treatment="Tukey")))->flum.glht
flum.glht # print all pairwise comparisons
```

Printing all pairwise comparisons is valuable to get P-values, estimates of differences, etc. for reporting (not shown here due to space). But we can also plot the 95% confidence intervals to allow better visualization of differences. In Figure 9.1, the comparison is listed on the left in the format (treatment1 - treatment2). If the confidence interval bars do not include zero (the vertical dotted line) then one could conclude that difference between treatments is significantly different according to the Tukey adjusted p-values (at the 5% confidence level).

```
par(mar=c(5,22,3,1), mgp=c(2,.7,0)) # set wide left margin
plot(flum.glht)
```

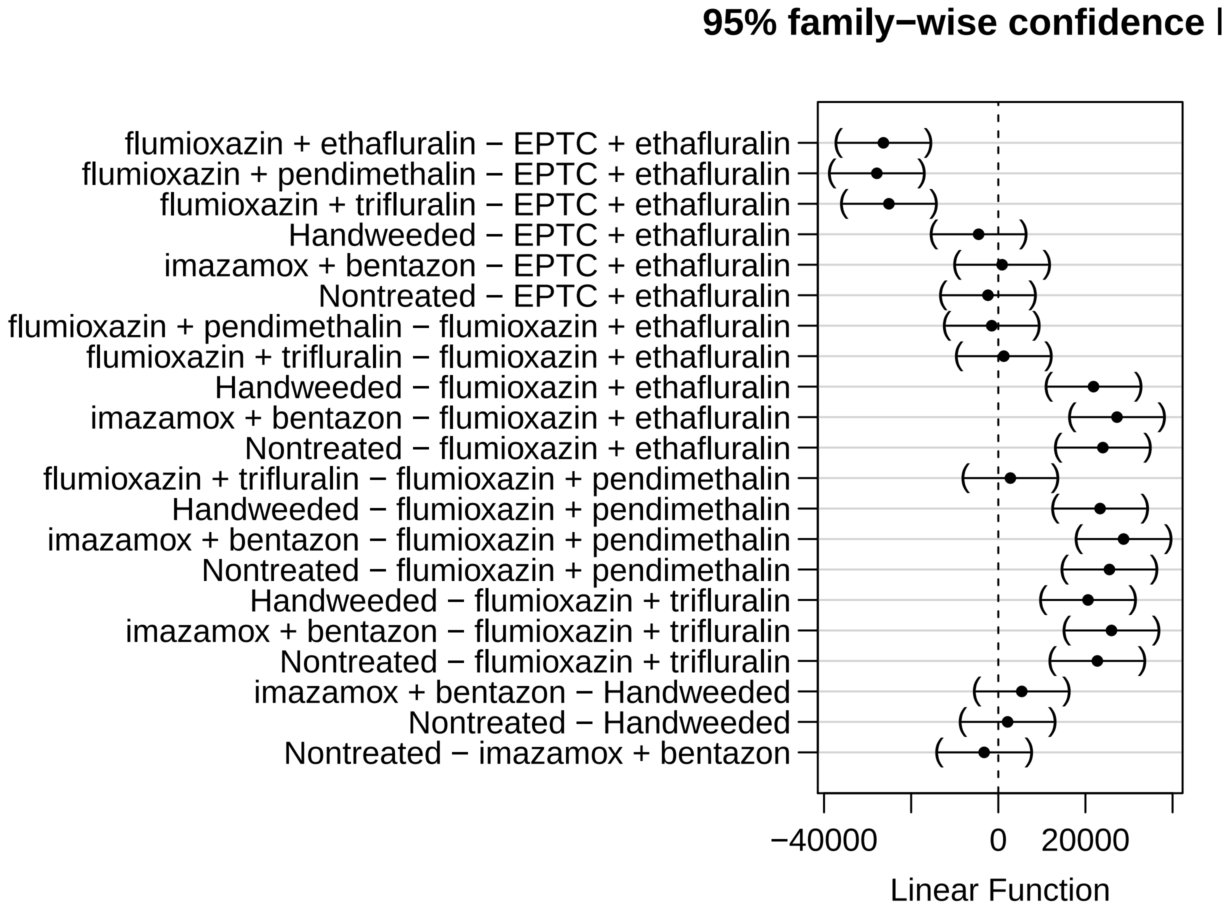


Figure 9.1: *Model estimates of treatment differences with 95% confidence intervals.*

The `glht()` function will not allow for unadjusted comparisons, since multiple testing tends to inflate the likelihood of a Type I error. If for some reason unadjusted Fisher LSD p-values are required (similar to the default in SAS) the `emmeans()` function in the `emmeans` package can be used. The estimated mrginal means (or least-square means) cna be printed along with the mean separation groups using the `cld()` function from the `multcomp` package, and the pairwise comparisons of estimated marginal means can be printed by calling the 'contrasts' of the estimated marginal means.

```
library(emmeans)
emmeans(flum.lmer, pairwise ~ treatment, data=flum.dat, adjust="none") -> flum.emm
multcomp::cld(flum.emm$emmeans)
```

```
##  treatment                    emmean   SE   df lower.CL upper.CL .group
##  flumioxazin + pendimethalin   20003 3483 20.9    12757    27250  1
##  flumioxazin + ethafluralin    21508 3483 20.9    14262    28754  1
##  flumioxazin + trifluralin     22775 3483 20.9    15529    30022  1
##  Handweeded                    43368 3483 20.9    36122    50614   2
##  Nontreated                    45506 3483 20.9    38260    52752   2
##  EPTC + ethafluralin           47882 3483 20.9    40636    55128   2
##  imazamox + bentazon           48753 3483 20.9    41507    55999   2
##
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
## P value adjustment: tukey method for comparing a family of 7 estimates
## significance level used: alpha = 0.05
```

```
flum.emm$contrasts
```

```
##  contrast                                                   estimate   SE   df t.ratio p.value
##  EPTC + ethafluralin - flumioxazin + ethafluralin              26374 3698 71.1  7.132  <.0001
##  EPTC + ethafluralin - flumioxazin + pendimethalin             27879 3698 71.1  7.539  <.0001
##  EPTC + ethafluralin - flumioxazin + trifluralin               25107 3698 71.1  6.790  <.0001
##  EPTC + ethafluralin - Handweeded                               4514 3698 71.1  1.221  0.2262
##  EPTC + ethafluralin - imazamox + bentazon                      -871 3698 71.1 -0.236  0.8144
##  EPTC + ethafluralin - Nontreated                               2376 3698 71.1  0.643  0.5226
##  flumioxazin + ethafluralin - flumioxazin + pendimethalin       1505 3698 71.1  0.407  0.6853
##  flumioxazin + ethafluralin - flumioxazin + trifluralin        -1267 3698 71.1 -0.343  0.7328
##  flumioxazin + ethafluralin - Handweeded                      -21859 3698 71.1 -5.911  <.0001
##  flumioxazin + ethafluralin - imazamox + bentazon             -27245 3698 71.1 -7.368  <.0001
##  flumioxazin + ethafluralin - Nontreated                      -23998 3698 71.1 -6.490  <.0001
##  flumioxazin + pendimethalin - flumioxazin + trifluralin       -2772 3698 71.1 -0.750  0.4559
##  flumioxazin + pendimethalin - Handweeded                     -23364 3698 71.1 -6.318  <.0001
##  flumioxazin + pendimethalin - imazamox + bentazon            -28750 3698 71.1 -7.775  <.0001
##  flumioxazin + pendimethalin - Nontreated                     -25503 3698 71.1 -6.897  <.0001
##  flumioxazin + trifluralin - Handweeded                       -20592 3698 71.1 -5.569  <.0001
##  flumioxazin + trifluralin - imazamox + bentazon              -25978 3698 71.1 -7.025  <.0001
##  flumioxazin + trifluralin - Nontreated                       -22731 3698 71.1 -6.147  <.0001
##  Handweeded - imazamox + bentazon                              -5386 3698 71.1 -1.456  0.1497
##  Handweeded - Nontreated                                       -2138 3698 71.1 -0.578  0.5649
##  imazamox + bentazon - Nontreated                               3247 3698 71.1  0.878  0.3828
##
## Degrees-of-freedom method: kenward-roger
```

Chapter 10

Mixed Models - Regression

10.1 Regression Models with Mixed Effects

When we conduct experiments over several years and/or at several locations, we have to decide if differences among years and/or differences among locations are of interest. In other words: are they fixed effects like the treatments we apply or we want to classify them random effects? If locations are picked at random it seems obvious to define locations as random effects. It means that the location contributes to the variation of the response that cannot be controlled. If the locations are picked according to their weed flora, soil type, and crop pattern, it would be wise to define locations as a fixed effect. Usually, one should use one's common sense and the knowledge of the experimental design and the objectivewhen determining whether locations are fixed or random. There is ongoing discussions among statisticians about what should be random and what should be fixed; but that discussion is somewhat outside this course.

10.1.1 Example 1: Sugarbeet yield

Within the many years that sugarbeet has been an economic crop, we incidentally did field experiments in 2006 and 2007, near Powell, Wyoming to determine the effects of duration of Salvia reflexa (lanceleaf sage) interference with sugarbeet (Odero et al. 2010). The experiment is also used in the ANCOVA chapter.

```
lanceleaf.sage<-read.csv("http://rstats4ag.org/data/lanceleafsage.csv")
head(lanceleaf.sage,n=3)
```

```
##   Replicate Duration Year Yield_Mg.ha Yield.loss.pct
## 1         1       36 2006       67.26          98.05
## 2         1       50 2006       60.34          87.96
## 3         1       64 2006       53.69          78.27
```

“

First of all we look at the distribution of data within years in Figure 10.1:

```
library(lattice)
xyplot(Yield_Mg.ha ~ Duration | factor(Year), data = lanceleaf.sage,
    xlab="Duration of Competition",
       panel = function(x,y)
         {panel.xyplot(x,y)
          panel.lmline(x,y)})
```

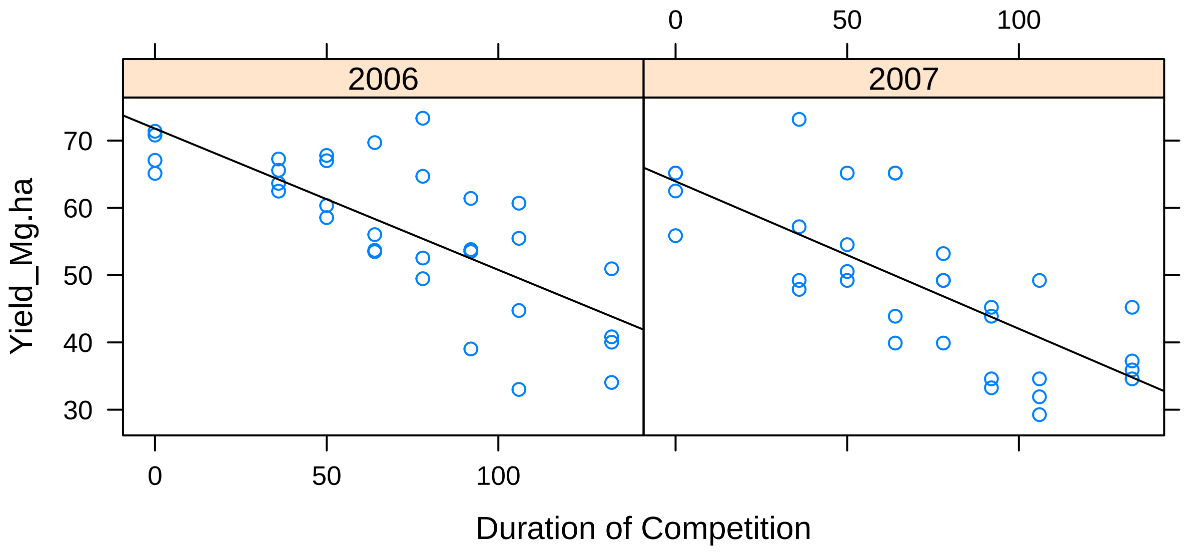


Figure 10.1: *The effect of the duration of weed competition on sugarbeet yield in two different years. It looks as if the relationships could be linear.*

The ANCOVA chapter analysis showed a linear regression was acceptable by checking the regression against the most general ANOVA model. The ANCOVA also showed that the regression slopes were the same independent of years, but the intercepts differed between years. In this chapter we now assume that the variation between years are random, because we cannot control the climate.

To analyze this we need to use the `lmer()`function in the package `lme4`. The syntax for `lmer()`is almost the same as for `lm()`, but there are some limitations and some important additional arguments. In `lm()` we can convert a continuous x-variable to a factor just by writing `factor(x)`. You can also do that with fixed effects in the `lmer()`function, but when it comes to random effects you cannot. In this particular experiment, there are two random effects: experimental years `(1|Year)` and Replicate because it is a block experiment. But the Replicate 1 in year 2006 is not the same as Replicate 1 in 2007. Consequently, we have to make the names for Replicates unambiguously defined with a unique name, so we get a total of 8 unique levels for Year and Replicate combination and the random effect takes the form `(1|New.Rep)` by defining `New.Rep` using the `with()` function.

```
lanceleaf.sage$New.Rep <- with(lanceleaf.sage, factor(Replicate):factor(Year))
levels(lanceleaf.sage$New.Rep)
```

```
## [1] "1:2006" "1:2007" "2:2006" "2:2007" "3:2006" "3:2007" "4:2006" "4:2007"
```

After this initial exercise, we are now ready to test whether we can assume linearity of Yield on duration of competition. We compare the regression model with the most general model, the ANOVA, as it is becoming standard practice in the course.

```
library(lme4)
Anova.m1 <- lmer(Yield_Mg.ha ~ factor(Duration) + (1|Year) + (1|New.Rep),
                 data=lanceleaf.sage, REML=FALSE)
Regression.m2 <- lmer(Yield_Mg.ha ~ Duration + (1|Year) + (1|New.Rep),
                      data=lanceleaf.sage, REML=FALSE)
anova(Anova.m1, Regression.m2) #No indication of nonlinearity
```

```
## Data: lanceleaf.sage
## Models:
## Regression.m2: Yield_Mg.ha ~ Duration + (1 | Year) + (1 | New.Rep)
## Anova.m1: Yield_Mg.ha ~ factor(Duration) + (1 | Year) + (1 | New.Rep)
##               npar    AIC    BIC  logLik deviance Chisq Df Pr(>Chisq)
## Regression.m2    5 454.77 465.56 -222.38   444.77
## Anova.m1        11 460.97 484.72 -219.49   438.97 5.791  6      0.447
```

The test for lack of fit is non-significant as was also seen in the ANCOVA chapter.

Notice, we have an argument that is not used in the general `lm()` and it is `REML` (Rstricted Maximum Likelihood). There are two options: Maximum Likelihood estimation (ML) REML. `REML=TRUE` is the default. ML produces downwards biased estimated variance components (over-optimistic precision). REML reduces the bias (similar to dividing by n - p rather than n for the residual standard error in linear regression). In other words REML agrees with `lm()` in case of no random effects. To make a long story short, we should use ML (`REML=FALSE`) when comparing models using the `anova()` function.

However, when we look at the summary of the mixed effect regression:

```
summary(Regression.m2)
```

```
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula: Yield_Mg.ha ~ Duration + (1 | Year) + (1 | New.Rep)
##    Data: lanceleaf.sage
##
##      AIC      BIC   logLik deviance df.resid
##    454.8    465.6   -222.4    444.8       59
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.71954 -0.65323 -0.09755  0.53285  2.36798
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  New.Rep  (Intercept)  8.993   2.999
##  Year     (Intercept) 14.092   3.754
##  Residual             52.217   7.226
## Number of obs: 64, groups:  New.Rep, 8; Year, 2
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept) 67.85642    3.40236  19.944
## Duration    -0.21451    0.02303  -9.314
##
## Correlation of Fixed Effects:
##          (Intr)
## Duration -0.473
```

it looks rather different from an ordinary regression summary with the `lm()` function. The first thing we see is that there are no significance levels like in `lm()` output. This is because there is still no consensus about how to calculate degrees of freedom for mixed effects models. There is fairly extensive discussion of this topic in the R community, that can be left out here, but we will point you to a breif explanation provided by Douglas Bates, the author of the `lmer()` function.

Variation between years (Year) and variation between reps within year (New.Rep) are both provided in the random effects portion of the output. What it means is that the variance between years and the variation among Replicates cannot be controlled by us; therefore we define them as random effect and they are now an integral part of the total variation we cannot explain.

Comparing the result above with the fixed year effect:

```
regression.fixed<-lm(Yield_Mg.ha ~ Duration + factor(Year), data=lanceleaf.sage)
summary(regression.fixed)
```

```
##
## Call:
## lm(formula = Yield_Mg.ha ~ Duration + factor(Year), data = lanceleaf.sage)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.3477  -5.2428  -0.7246   3.5724  17.9560
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      72.09580    2.20523  32.693  < 2e-16 ***
## Duration         -0.21451    0.02472  -8.678 3.02e-12 ***
## factor(Year)2007 -8.47875    1.93890  -4.373 4.86e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.756 on 61 degrees of freedom
## Multiple R-squared:  0.6075, Adjusted R-squared:  0.5947
## F-statistic: 47.21 on 2 and 61 DF,  p-value: 4.081e-13
```

We see that the slope for the mixed model is the same as for the fixed year model -0.21 with standard error of 0.02. The intercept of the mixed model is 67.9 (3.4). For the fixed effect model the intercepts were 72.1 and 72.1 -8.5 = 63.6with a standard error of 2.2 and 1.9, respectively. The changes of going from fixed year effect to random year effect is not dramatic when it comes to regression slopes, either. But if we decide year is random, then the mixed model is the one to use. We get a bonus, however by separating the variation of years from the residual. Reporting this information gives others who want to include our results in a meta-analysis knowledge of the actual variation in the experiments.

The illustration of the results is shown below. In order to make the plot neat we take the averages of the measurements by using the package `dplyr`, before plotting the data in Figure 10.2.

```
library(dplyr)
averages<-lanceleaf.sage %>%
  group_by(Duration,Year) %>%
  summarise(YIELD = mean(Yield_Mg.ha))
par(mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(YIELD ~ Duration, data = averages, bty="l",
     ylab = quote(Yield~(kg~ha^-1)), #Note how to use dimension accepted in some journals
     xlab = "Duration of competition",
     pch  = as.numeric(factor(Year)), ylim=c(10,80))
  abline(fixef(Regression.m2))
```

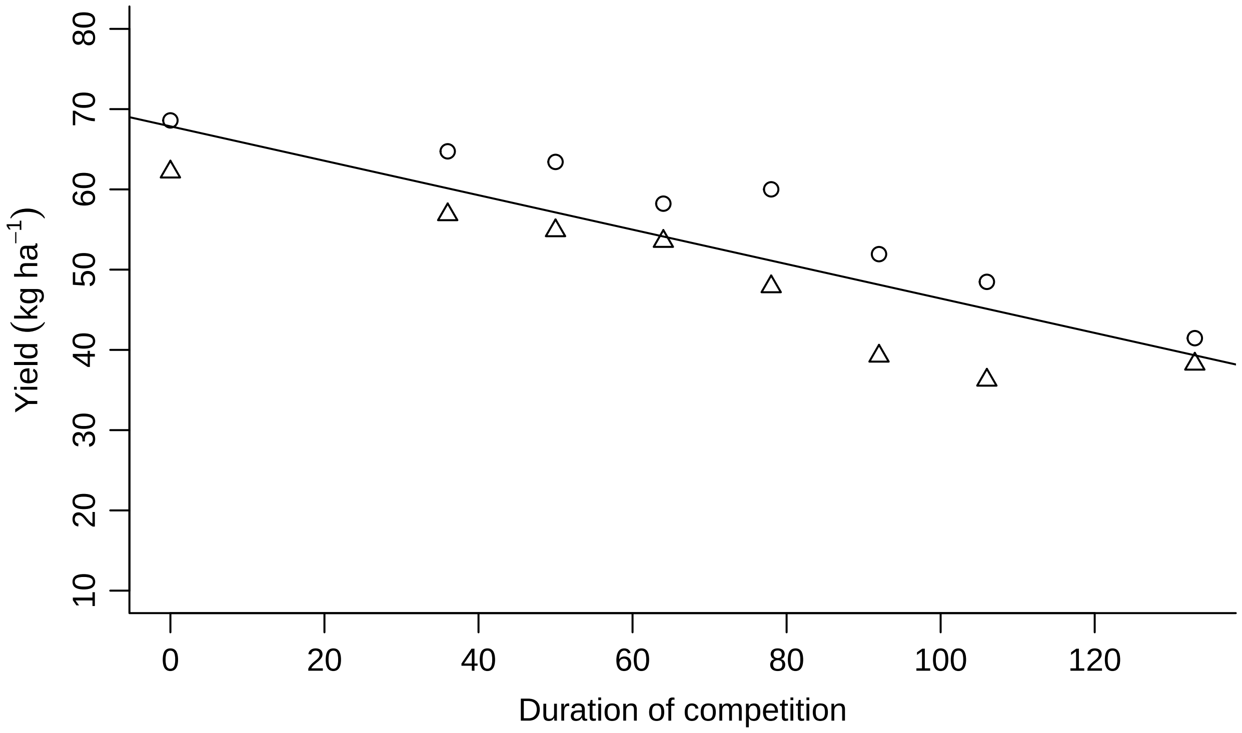


Figure 10.2: *The regression model with year as a random effect. Note that in this instance we have one slope and one intercept, and the regression line now is almost in the middle. Note that the `fixef(Regression.m2)` contains the fixed regressions parameters and the ordinary `abline()` can be used.*

10.1.2 Example 2: Fungicide toxicity

Another example is with a total of six concentration-response experiments evaluating the effect of a fungicide, vinclozolin on luminescence of ovary cells. Vinclozolin is a putative endogenous disruptor and therefore banned in Denmark. The concentration-response curves were run on separate days and it shows that the calibration of the instrument makes the response-curves somewhat different.

```
vinclozolin <- read.csv("http://rstats4ag.org/data/vinclozolin.csv")
head(vinclozolin)
```

```
##    conc effect experiment
## 1 0.025    908      10509
## 2 0.050    997      10509
## 3 0.100    744      10509
## 4 0.200    567      10509
## 5 0.390    314      10509
## 6 0.780    325      10509
```

Figure 10.3 shows the individual response curves illustrating the variation in the slope and intercept of the straight lines. However, the variation is kind of erratic because of daily calibration of the machine. Therefore, we consider days a random effect, the variation is intangible. As was the case with the duration of competition we will change the numeric name of each experiment to a factor:

```
vinclozolin$Experiment <- factor(vinclozolin$experiment)
```

And then run the analysis, but first we illustrate thevariation in data (Figure 10.3).

```
library(lattice)
xyplot(effect ~ log(conc) | Experiment, data = vinclozolin,
       xlab = "Log( Concentration of Vinclozolin)", ylab = "Luminescence",
       panel = function(x,y)
         {panel.xyplot(x,y)
          panel.lmline(x,y)})
```

The procedure to run the mixed regression model is the same as for the sugarbeet example. First we conduct a lack-of-fit test against the most general model - ANOVA.

```
library(lme4)
vinclo.mixed.ANOVA <- lmer(effect ~ factor(conc) + (1|Experiment),
                           data = vinclozolin, REML = FALSE)
vinclo.mixed.Regression <- lmer(effect ~ log(conc) + (1|Experiment),
                                data = vinclozolin, REML = FALSE)
anova(vinclo.mixed.ANOVA, vinclo.mixed.Regression)
```

```
## Data: vinclozolin
## Models:
## vinclo.mixed.Regression: effect ~ log(conc) + (1 | Experiment)
## vinclo.mixed.ANOVA: effect ~ factor(conc) + (1 | Experiment)
##                         npar    AIC    BIC  logLik deviance Chisq Df Pr(>Chisq)
## vinclo.mixed.Regression    4 627.06 634.46 -309.53   619.06
## vinclo.mixed.ANOVA        10 628.42 646.92 -304.21   608.42 10.64  6     0.1002
```

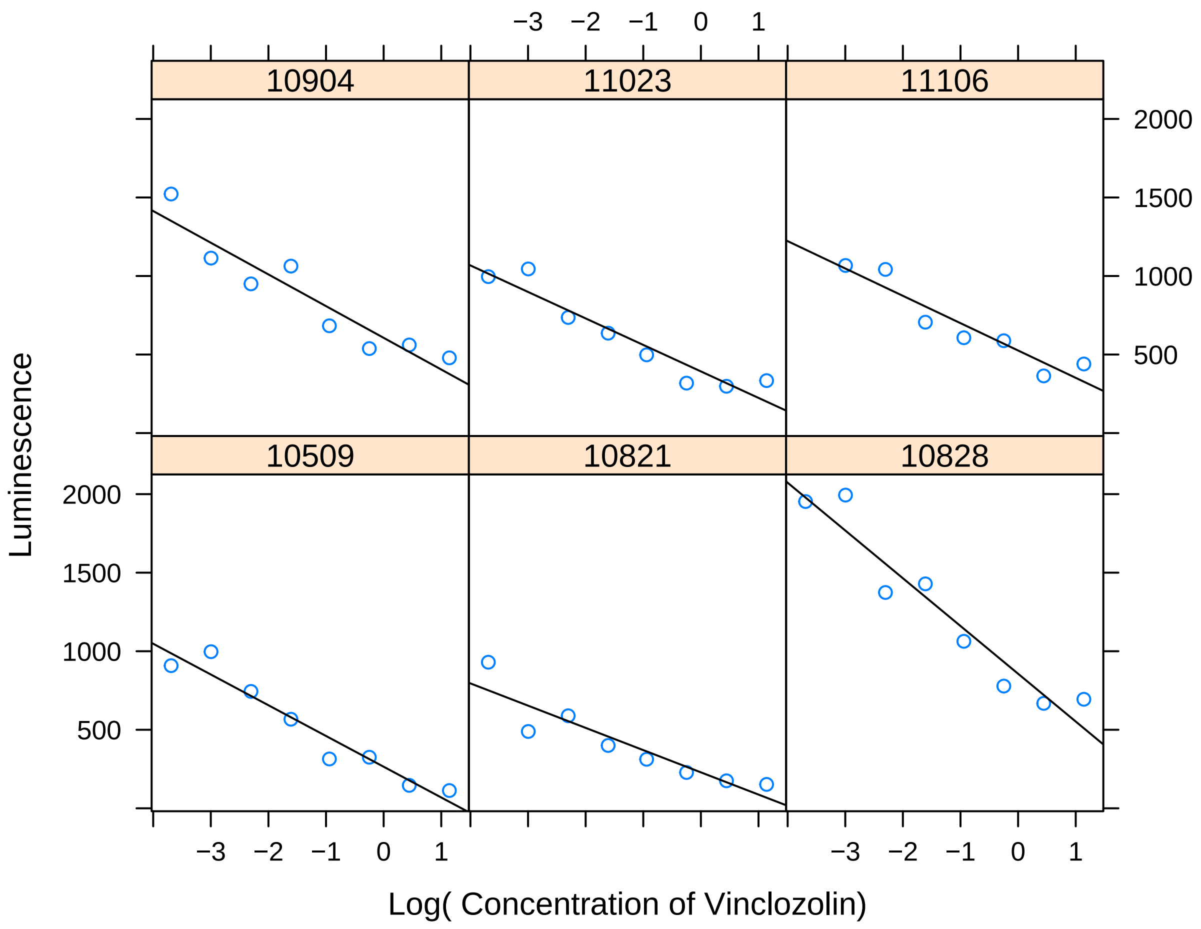


Figure 10.3: *Concentration-response curves for ovary cell luminescence on venclozolin.*

The test for lack of fit is non-significant so we can assume the relationship is linear, whatever the day. Please note that we only have one replication per regression per day. But the concentrations are the same, so a test for lack of fit is still appropriate.

```
summary(vinclo.mixed.Regression)
```

```
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula: effect ~ log(conc) + (1 | Experiment)
##    Data: vinclozolin
##
##      AIC      BIC   logLik deviance df.resid
##    627.1    634.5   -309.5    619.1       43
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.9356 -0.6270 -0.2354  0.6112  3.0031
##
## Random effects:
##  Groups     Name        Variance Std.Dev.
##  Experiment (Intercept) 72092    268.5
##  Residual               19987    141.4
## Number of obs: 47, groups:  Experiment, 6
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   478.77     112.71   4.248
## log(conc)    -199.03      13.31 -14.956
##
## Correlation of Fixed Effects:
##           (Intr)
## log(conc) 0.144
```

The variation among experimental days (268) is huge compared with the variation within days (141).

```
xyplot(effect~log(conc)|experiment, data=vinclozolin,
       xlab="Log( Concentration of Vinclozolin)", ylab="Luminescence",
       panel=function(...)
         {panel.xyplot(...);
          panel.abline(fixef(vinclo.mixed.Regression))})
```

If we had run an ordinary regression by pooling the six concentration-response curves then we would have gotten the result below.

```
vinclo.fixed.Regression<-lm(effect ~ log(conc), data = vinclozolin)
summary(vinclo.fixed.Regression)
```

```
##
## Call:
## lm(formula = effect ~ log(conc), data = vinclozolin)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

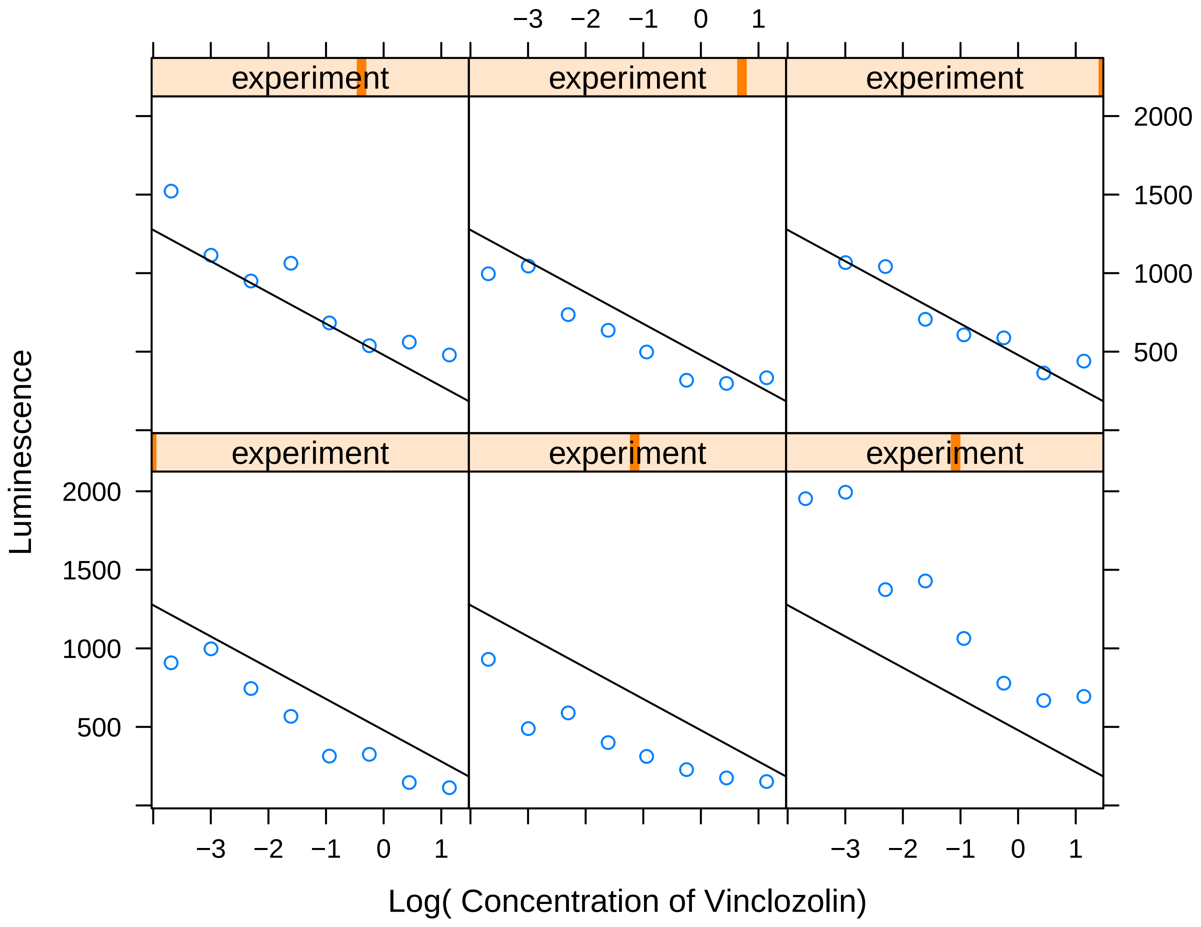


Figure 10.4: *Regression lines for all 6 days shown in six panels to give an idea of the variation among days. There appear to be particularly large differences between the responses and the regression lines for the three days in the lower panels.*

```
## -584.64 -212.74  -58.83  179.01  920.36
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   478.90      58.03   8.253 1.48e-10 ***
## log(conc)    -198.53      29.33  -6.768 2.25e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 312.6 on 45 degrees of freedom
## Multiple R-squared:  0.5044, Adjusted R-squared:  0.4934
## F-statistic:  45.8 on 1 and 45 DF,  p-value: 2.253e-08
```

The slope and the intercept are almost identical for the two `lmer()`and `lm()`. The standard errors for the intercept, though, roughly doubled, 58 for `lm()` to 113 for `lmer()`, whereas the slope was more that halved, 13, for the `lmer()`compared to 29 with `lm()`. Undoubtedly, the experimental dates improved the precision of the slope. We can compare the fixed and random effects models on these data using Akaike Information Criterion (AIC). For a given set of data, the model with the lowest AIC value is more likely to be the appropriate model.

```
AIC(vinclo.mixed.Regression, vinclo.fixed.Regression)
```

```
##                         df      AIC
## vinclo.mixed.Regression  4 627.0628
## vinclo.fixed.Regression  3 677.3501
```

As a gereral rule, differences in AIC values less than 10 indicate two models perform similarly in describing the data. For the vinclozolin data set, the mixed model has and AIC of 627 compared to 677 for the fixed model where the effect of day was not included. This indicates the mixed model is the best fit for these data.

Chapter 11

Logistic Regression (Binary Response)

Determination of the effective dose of a herbicide under field, greenhouse, or laboratory conditions is a common goal of weed science experiments. Although the response variable in some cases is continuous (dry weight) or percent (visual injury), in many cases the response variable of interest is a binary response, such as mortality (the plant is alive or dead). This type of data is important in many types of toxicology and pest management, and weed science is no exception. In many ways the analysis of binary response data is analogous to using ANOVA followed by non-linear regression.

11.1 Generalized Linear Model

Instead of fitting a linear model using the lm() function, analysis of binary response data requires the use of a generalized linear model with the glm() function. For this example, the data set has mortality observations collected 21 days after treatment with six rates of imazamox with and without six different growth regulator herbicides.

```r
rye.dat <- read.csv("http://rstats4ag.org/data/2012_RyeGH.csv")
rye.glm <- glm(mort.21dat ~ imaz.rate + gr, data = rye.dat,
               family = binomial(link = "logit"))
anova(rye.glm, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: mort.21dat
##
## Terms added sequentially (first to last)
##
##
##           Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                        430     464.54
## imaz.rate  1   93.502       429     371.04 < 2.2e-16 ***
## gr         7   31.180       422     339.86 5.759e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Using `anova()` on the `glm()` fit results in an analysis of deviance table with a similar interpretation as an ANOVA table for continuous response data. Both the imazamox rate and growth regulator herbicide significantly affected the mortality of feral rye, which was the expected result. The drc package can be used to quantify the response of feral rye to imazamox in the presence and absence of the growth regulator herbicides.

```
library(drc)
rye.drc <- drm(mort.21dat ~ imaz.rate, gr, data = rye.dat, fct = LL.2(),
               type = "binomial", na.action = na.omit)
```

```
## Control measurements detected for level: Control
```

```
summary(rye.drc)
```

```
##
## Model fitted: Log-logistic (ED50 as parameter) with lower limit at 0 and upper limit at 1 (2 parms)
##
## Parameter estimates:
##
##               Estimate Std. Error t-value   p-value
## b:None         1.31358    0.52697  2.4927 0.0126766 *
## b:2,4-D amine  1.93438    0.63004  3.0702 0.0021389 **
## b:2,4-D ester  1.83033    0.45369  4.0343 5.476e-05 ***
## b:MCPA amine   2.00705    0.67980  2.9524 0.0031531 **
## b:MCPA ester   1.16337    0.34724  3.3503 0.0008071 ***
## b:dicamba      2.38035    0.79232  3.0043 0.0026623 **
## b:fluroxypyr   1.93821    0.63093  3.0720 0.0021264 **
## e:None        97.74085   48.66554  2.0084 0.0445987 *
## e:2,4-D amine 61.90448   15.27230  4.0534 5.048e-05 ***
## e:2,4-D ester 25.55234    4.80647  5.3162 1.059e-07 ***
## e:MCPA amine  66.09968   16.64430  3.9713 7.148e-05 ***
## e:MCPA ester  28.58611    7.85012  3.6415 0.0002711 ***
## e:dicamba     62.04112   12.76510  4.8602 1.173e-06 ***
## e:fluroxypyr  61.84463   15.21629  4.0644 4.816e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
EDcomp(rye.drc, c(50,50))
```

```
##
## Estimated ratios of effect doses
##
##                                  Estimate  Std. Error     t-value     p-value
## 2,4-D amine/2,4-D ester:50/50  2.4227e+00  7.5160e-01  1.8928e+00  5.8379e-02
## 2,4-D amine/dicamba:50/50      9.9780e-01  3.2054e-01 -6.8712e-03  9.9452e-01
## 2,4-D amine/fluroxypyr:50/50   1.0010e+00  3.4876e-01  2.7746e-03  9.9779e-01
## 2,4-D amine/MCPA amine:50/50   9.3653e-01  3.3015e-01 -1.9224e-01  8.4755e-01
## 2,4-D amine/MCPA ester:50/50   2.1655e+00  7.9943e-01  1.4580e+00  1.4485e-01
## 2,4-D amine/None:50/50         6.3335e-01  3.5194e-01 -1.0418e+00  2.9751e-01
## 2,4-D ester/dicamba:50/50      4.1186e-01  1.1482e-01 -5.1224e+00  3.0171e-07
## 2,4-D ester/fluroxypyr:50/50   4.1317e-01  1.2796e-01 -4.5860e+00  4.5186e-06
## 2,4-D ester/MCPA amine:50/50   3.8657e-01  1.2150e-01 -5.0487e+00  4.4489e-07
## 2,4-D ester/MCPA ester:50/50   8.9387e-01  2.9753e-01 -3.5669e-01  7.2132e-01
## 2,4-D ester/None:50/50         2.6143e-01  1.3915e-01 -5.3079e+00  1.1091e-07
## dicamba/fluroxypyr:50/50       1.0032e+00  3.2175e-01  9.8746e-03  9.9212e-01
## dicamba/MCPA amine:50/50       9.3860e-01  3.0521e-01 -2.0117e-01  8.4056e-01
## dicamba/MCPA ester:50/50       2.1703e+00  7.4473e-01  1.5715e+00  1.1607e-01
## dicamba/None:50/50             6.3475e-01  3.4197e-01 -1.0681e+00  2.8548e-01
## fluroxypyr/MCPA amine:50/50    9.3563e-01  3.2939e-01 -1.9543e-01  8.4506e-01
## fluroxypyr/MCPA ester:50/50    2.1635e+00  7.9769e-01  1.4585e+00  1.4470e-01
## fluroxypyr/None:50/50          6.3274e-01  3.5141e-01 -1.0451e+00  2.9598e-01
## MCPA amine/MCPA ester:50/50    2.3123e+00  8.6153e-01  1.5232e+00  1.2770e-01
## MCPA amine/None:50/50          6.7627e-01  3.7733e-01 -8.5793e-01  3.9093e-01
## MCPA ester/None:50/50          2.9247e-01  1.6630e-01 -4.2545e+00  2.0950e-05
```

Please note that the test for significant relative potency is not a null hypothesis, but that the relative potency SI is different from unity.

```r
par(mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(rye.drc, legendPos=c(10,0.6), col=T, xt=unique(rye.dat$imaz.rate),
     broken=T, bp=1.5, xlim=c(0,100),xlab="Imazamox rate", bty="l",
     ylab="Probability of survival 21 DAT")
```

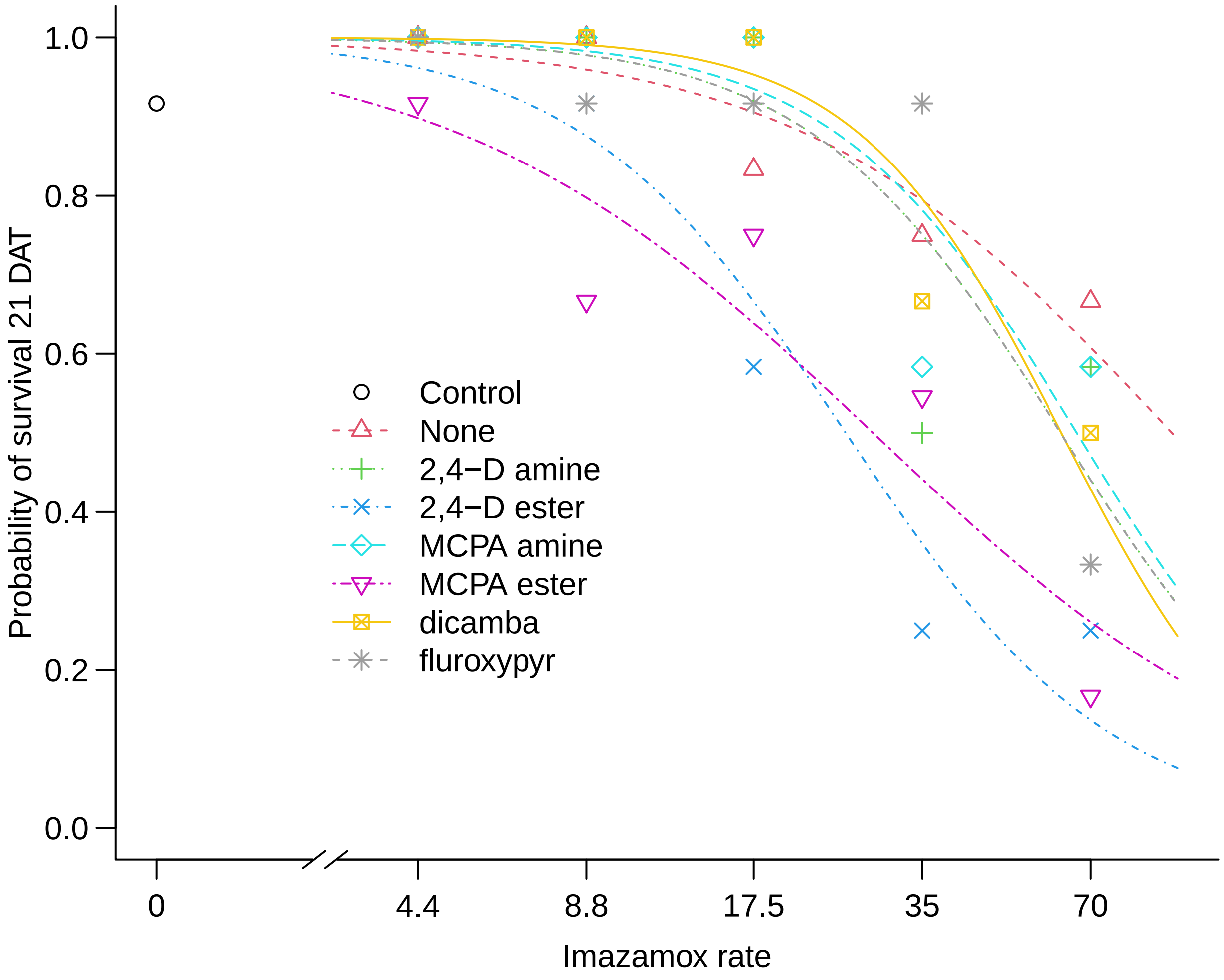


Figure 11.1: *The effect of imazamox rate with and without auxin-mimic herbicides on rye survival.*

Chapter 12

Dose-Response curves

We are surrounded by synthetic and natural substances that have both positive and negative effects upon humans, other animals, and the environment. It is difficult to estimate effects of a substance without testing it on living organisms. Chemical and physical properties of a substance can tell us about the molecules themselves, but living organisms may react in unpredictable ways. Therefore, we use biological assay to estimate effects. It means the design of bioassay experiments and the modelling of the response as a function of the dose is crucial for the classification of chemical compounds. In order to understand the principles of bioassay it is imperative to link results with the real world exposure to chemicals. In this chapter, chemicals used to illustrate bioassay are pesticides, but the principles are the same whatever the chemical.

Finney 1976 wrote: "As a branch of applied statistics, biological assay appears to have a somewhat specialized appeal. Although a few statisticians have worked in it intensively, to the majority it appears as a topic that can be neglected, either because of its difficulties or because it is trivial……. I am convinced that many features of bioassay are out-standingly good for concentrating the mind on important parts of biometric practice and statistical inference."

In biology there are four general curves that describe many biological phenomena. Two of them, the rectangular hyperbola (of which Michaelis-Menten is a notable example) and sigmoid curves have either one or two asymptotes; that is, they have upper and/or lower limits as the independent variable approaches very small or very large values.

In most text books, non-linear regression is not very well explained although it is easy for practitioners to implement. The difference between linear and non-linear regression can often be boiled down to the very fact: linear regression can be solved analytically by least square analysis, whereas the nonlinear regression cannot.

Before fitting a nonlinear regression model, one has to do some guesstimates about the initial regression parameters before fitting. In some cases a nonlinear relationship can be converted to a linear one. The most common one is the exponential curve (Figure 1 that turns into a linear relationship when taking the logarithm of the `y`. For the rectangular hyperbola it is usually with no lower limit and then it can also be linearize by manipulating as in ezymeology and biochemistry.The rectangular hyperbola is in other disciplines called the Michaelis Menten model.

This chapter will be devoted to the use of the add-on package ‘drc? and will illustrate some of the facilities built in the package so far.

For the sigmoid curves, which can take several forms, we have to use nonlinear regression. One of the most common curves is the symmetric log-logistic model:

$$y = c + \frac{d - c}{1 + exp(b(log(x) - log(ED50)))}.$$

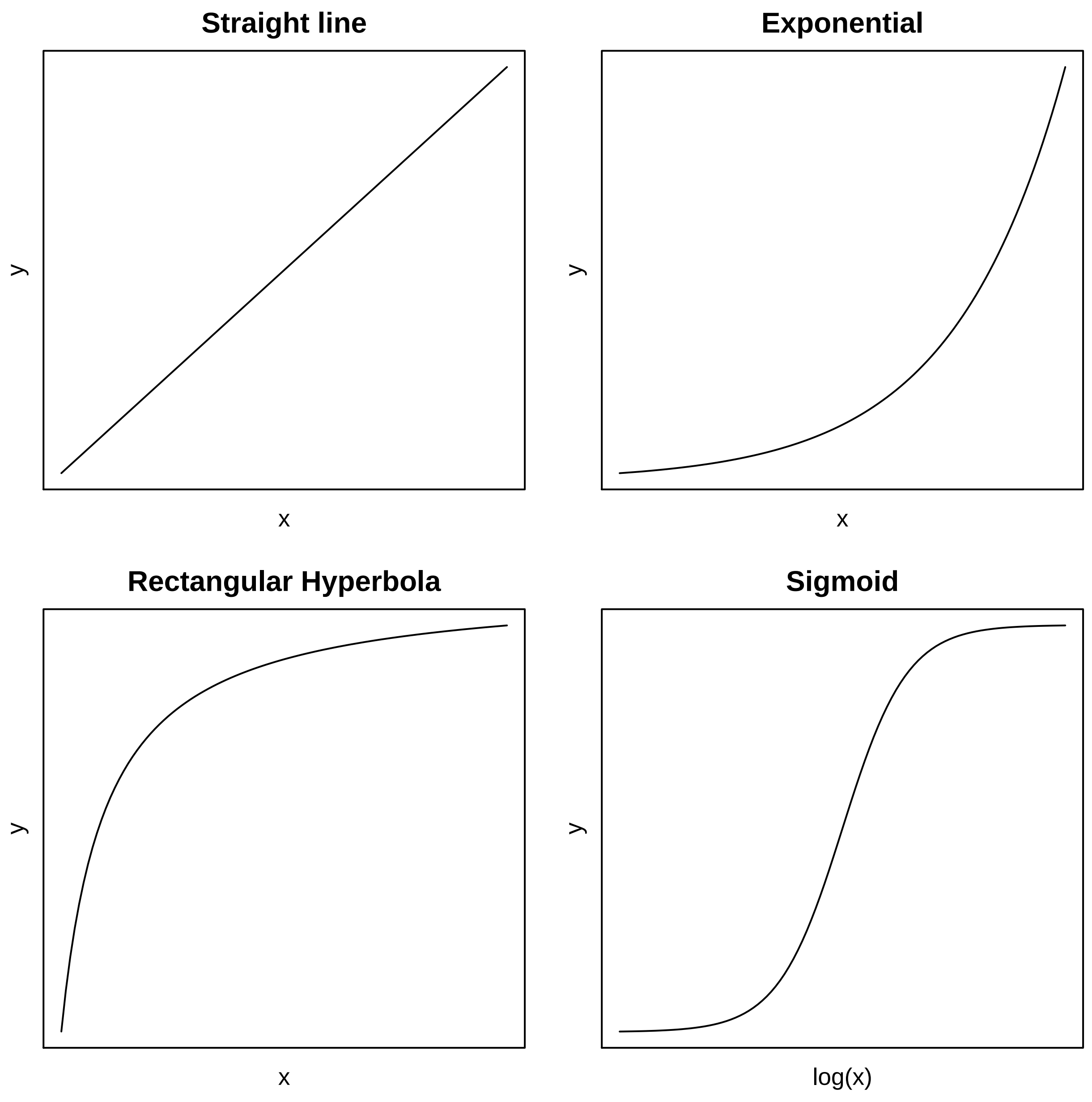


Figure 12.1: *Common regressions models in the biological sciences.*

`y` is the response, `c` denotes the lower limit of the response when the dose `x` approaches infinity; `d` is the upper limit when the dose `x` approaches 0. `b` denotes the slope around the point of inflection, which is the ED_{50}, i.e. the dose required to reduce the response half-way between the upper and lower limit.

Guessing the parameters of a sigmoid curve can, for the less experienced person, be tiresome and frustrating. This has now been overcome by the development of a package named `drc`, which was developed at the University of Copenhagen. The purpose of the `drc` package was to ease the analysis of dose-response curves from greenhouse and field experiments in Weed and Pesticide Science courses and to assess the selectivity of herbicides and efficacy of pesticides. Similar dose-response curves are commonly used in ecotoxicology and general toxicology, and the `drc` package is used in various other disciplines.

Gradually, `drc` has been extended with numerous nonlinear curves commonly used in the life sciences and elsewhere. The engine of `drc` is the `drm(y~x, fct=...)` function with self-starter that automatically calculates initial parameters. The `fct=`argument identifies the particular dose-response curve to be used. An overview of the various functions available can be accessed, after loading the `drc` package, by typing `?getMeanFunctions()`.

12.1 One Dose-Response Curve

Let us look at how to fit a dose-response curve using a training data set in the `drc` package.

```r
library(drc)
```

Before curve fitting, it is important to get an idea of how the data look. Below we create a plot with the x variable both on original scale and on log scale.

```r
head(ryegrass,3)
```

```
##      rootl conc
## 1 7.580000    0
## 2 8.000000    0
## 3 8.328571    0
```

```r
op <- par(mfrow = c(1, 2), mar=c(3.2,3.2,2,.5), mgp=c(2,.7,0)) #make two plots in two columns
plot(rootl ~ conc, data = ryegrass, main="Original Dose Scale")
plot(rootl ~ log(conc+.1), data = ryegrass, main="Logarithmic Dose Scale")
```

Below we fit a four-parameter log-logistic model with user-defined parameter names. The default names of the parameters (b, c, d, and e) included in the `drm()` function might not make sense to many weed scientists, but the `names=c()` argument can be used to facilitate sharing output with less seasoned `drc` users. The four parameter log-logistic curve has an upper limit, `d`, lower limit, `c`, the ED_{50} is denoted `e`, and finally the slope, `b`, around the ED_{50}.

```r
ryegrass.m1 <- drm(rootl ~ conc, data = ryegrass,
                   fct = LL.4(names = c("Slope", "Lower Limit", "Upper Limit", "ED50")))
summary(ryegrass.m1)
```

```
##
## Model fitted: Log-logistic (ED50 as parameter) (4 parms)
##
## Parameter estimates:
```

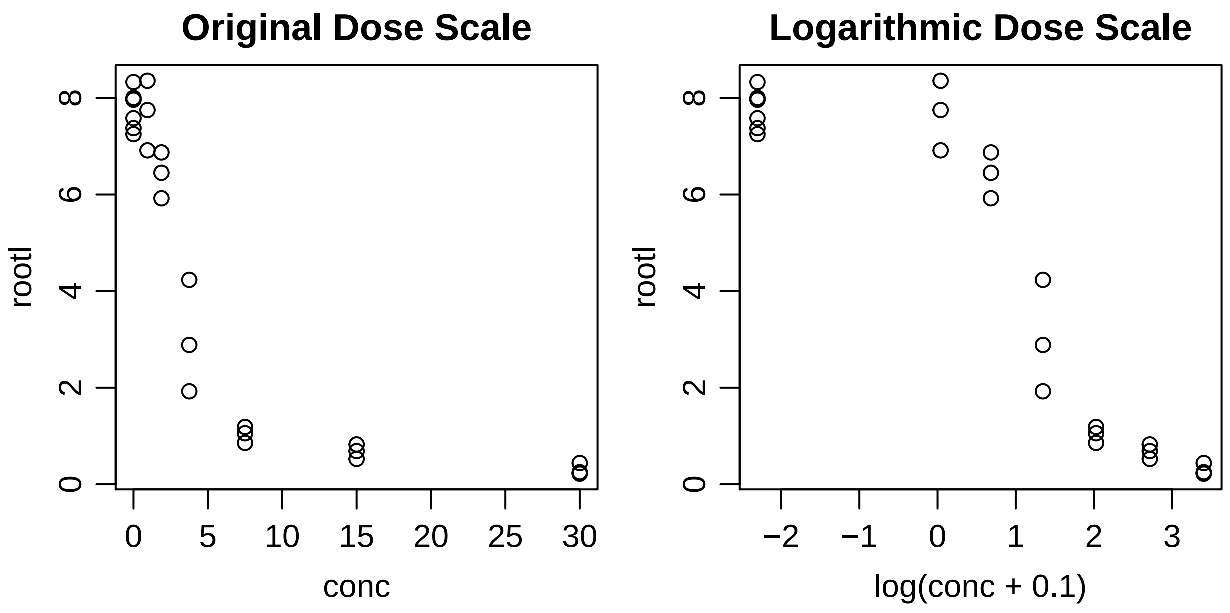


Figure 12.2: *There are two ways of plotting dose-response data, the default in the* **drc** *package is to use a logarithmic scale for x. Note with the logarithmic dose scale the untreated control, 0, is not defined. That is why we add 0.1 to the concentration before taking the logarithm.*

```
##
##                          Estimate Std. Error t-value   p-value
## Slope:(Intercept)         2.98222    0.46506  6.4125 2.960e-06 ***
## Lower Limit:(Intercept)   0.48141    0.21219  2.2688   0.03451 *
## Upper Limit:(Intercept)   7.79296    0.18857 41.3272 < 2.2e-16 ***
## ED50:(Intercept)          3.05795    0.18573 16.4644 4.268e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error:
##
##  0.5196256 (20 degrees of freedom)
```

The summary of the curve fitting shows the estimates of each of the four parameters and their standard errors. The p-values tell us if the parameters are different from zero. In this instance all four parameters are significantly different from zero and as seen on the graph the log-logistic curve seems to fit well to data.

```
op <- par(mfrow = c(1, 2), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(ryegrass.m1, broken=TRUE, bty="l",
     xlab="Concentration of Ferulic Acid", ylab="Length of Roots")
plot(ryegrass.m1, broken=TRUE, bty="l",
     xlab="Concentration of Ferulic Acid", ylab="Length of Roots",type="all")
```

The slope of the dose-response curve at ED_{50} has the opposite sign as compared to the sign of the parameter `b`. This is the consequence of the parameterization used in `drc` for the log-logistic model, a choice that is in part rooted in what was commonly used in the past. The actual slope of the tangent of the curve at ED_{50} is determined :

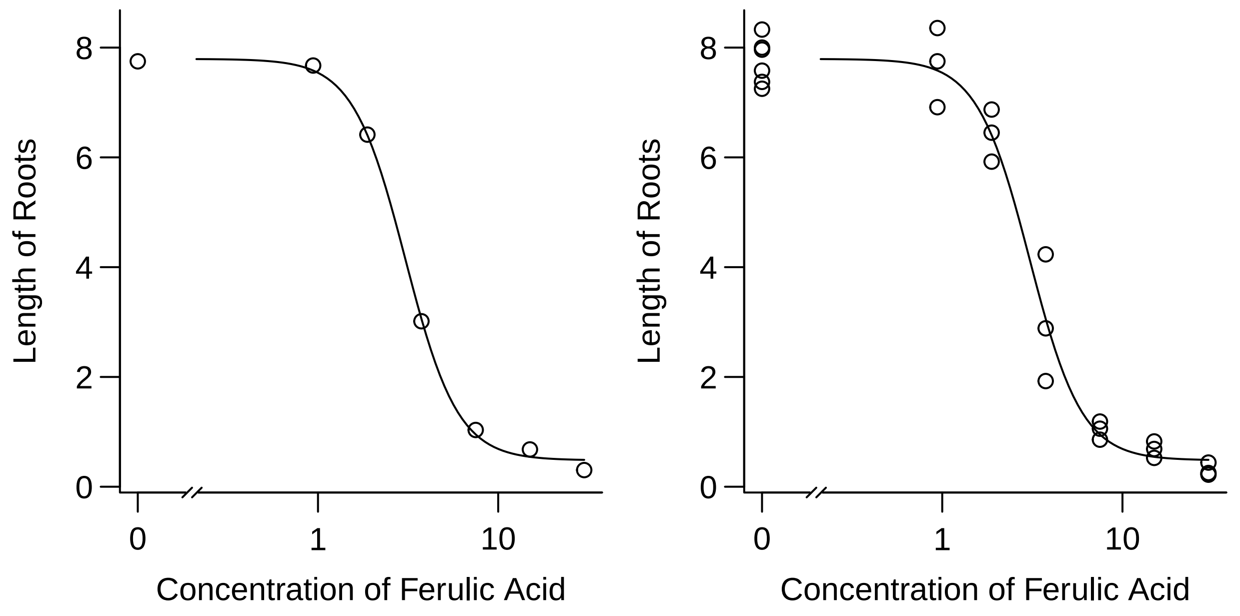


Figure 12.3: *Plot of regression and averages of observations within each concentration (left), and all observation (right).*

$$\frac{-b}{(d-c)/(4*e)}$$

So apart from the sign there is a scaling factor that converts the parameter `b` into the slope, Therefore, the use of the word "relative". Note that the scaling factor may be viewed as a kind of normalization factor involving the range of the response. As a consequence estimated `b` values often lie in the range 0.5 to 20 in absolute value, regardless of the assay or experiment generating the data.

Note at the end of the ryegrass help file `?ryegrass` you can see examples, which give you some ideas of what can be done. The ryegrass help provides examples of various sigmoid curves to show the differences among them (see later). Note that we use the argument `broken=TRUE` to break the x-axis, because on a logarithmic scale, which is the default in `drc`, there is no such thing as a zero dose.

There are various ways to check if a regression model gives a satisfactory picture of the variation in data. Before doing further analysis, the best way to judge it is to look at the graph to see if there are some serious discrepancies between the regression and mean values. Being satisfied with the results of the fit, one way of checking is to test for lack of fit against the most general model, an ANOVA. It is done automatically in drc with the function `modelFit()`.

```
#Test for lack of fit
modelFit(ryegrass.m1)
```

```
## Lack-of-fit test
##
##           ModelDf    RSS Df F value p value
## ANOVA          17 5.1799
## DRC model      20 5.4002  3  0.2411  0.8665
```

Obviously, the test for lack of fit is non-significant and therefore we can tentatively entertain the idea that the regression analysis was just as good to describe the variation in data as was an ANOVA. The test for lack of fit is not enough to ensure the fit is reasonable, because of few degrees of freedom, in this instance only 3 degrees of freedom. Another issue is that the more doses we use the less likely it is that the test for lack of fit would become non-significant; the mere fact being that the log-logistic curve, like any other sigmoid curve, is just an approximation of the real relationship.

The distribution of the residuals and the hopefully normal distribution of residuals are informative to make sure that the prerequisites for doing a regression in the first place are not violated. The assumptions we must consider include:

1. Correct regression model
2. Variance Homogeneity
3. Normally distributed measurement errors
4. Mutually independent measurement error ε

Number 1 can always be discussed and we will not go too much into detail. This is generally determined by researchers familiar with the phenomena being described. Number 2 is important if you want to infer on the variability of parameters. Normally, the parameter estimates do not change much whether there are homogeneous variance or not; it is the standard errors that change and that is why we want to come so close to homogeneity of variance as possible. This is important when comparing curves, or evaluating the certainty we have about various parameters (like the effective dose or upper limit). Number 3 can be graphically shown and is only in some instances critical. Number 4 is important because we assume the observations for doses are independent of each other, a question that also can be debated. When we do dose-response experiments we often use the same stock solution. It means that any systematic error can be carried all the way though the dilution process and be part of the dose. Below we illustrate items 2 and item 3.

```
#Graphical analysis of residuals
op <- par(mfrow = c(1, 2), mar=c(3.2,3.2,2,.5), mgp=c(2,.7,0)) #put two graphs together
    plot(residuals(ryegrass.m1) ~ fitted(ryegrass.m1), main="Residuals vs Fitted")
abline(h=0)
qqnorm(residuals(ryegrass.m1))
 qqline(residuals(ryegrass.m1))
```

The residual plot indicates that the model was able to catch the variation in data. The distribution of residuals should look like a shotgun shot on a wall. Apparently, there are some systematic violations in the residual plot, At small fitted values the residuals are close to zero than at high fitted values the variation is much more pronounced. This is called a funnel type pattern. We also see it in the graphs with the fit and all the observations (Figure 12.4). The normal Q-Q plot looks all right, no serious deviation from the assumption of normal distribution of residuals. Later we will look at this problem, which is a statistical problem not a biological one.

12.2 Note on use of R^2

One of the most popular ways of assessing goodness of fit in regression, be it linear or nonlinear, is to use R^2. The higher the better, but the R^2 does *not* tell you how well the curve fits to the data. In fact, if there are large differences between the maximum and the minimum observation, the R^2 will be larger than if the differences are small. It is also important to note that the most common interpretation of R^2 for linear regression does not hold true for nonlinear regression; in the nonlinear case, the R^2 value is *not* the amount of variability in Y explained by X. At its core, the R^2 is a comparison of 2 models; in linear regression the two models being compared are the regression model and a null model indicating no relationship between X and Y. For nonlinear regression, the choice of a ‘null’ model is not as simple, and therefore the R^2 type measure will be different depending on the chosen reduced model. The appropriate ‘null’ model may differ

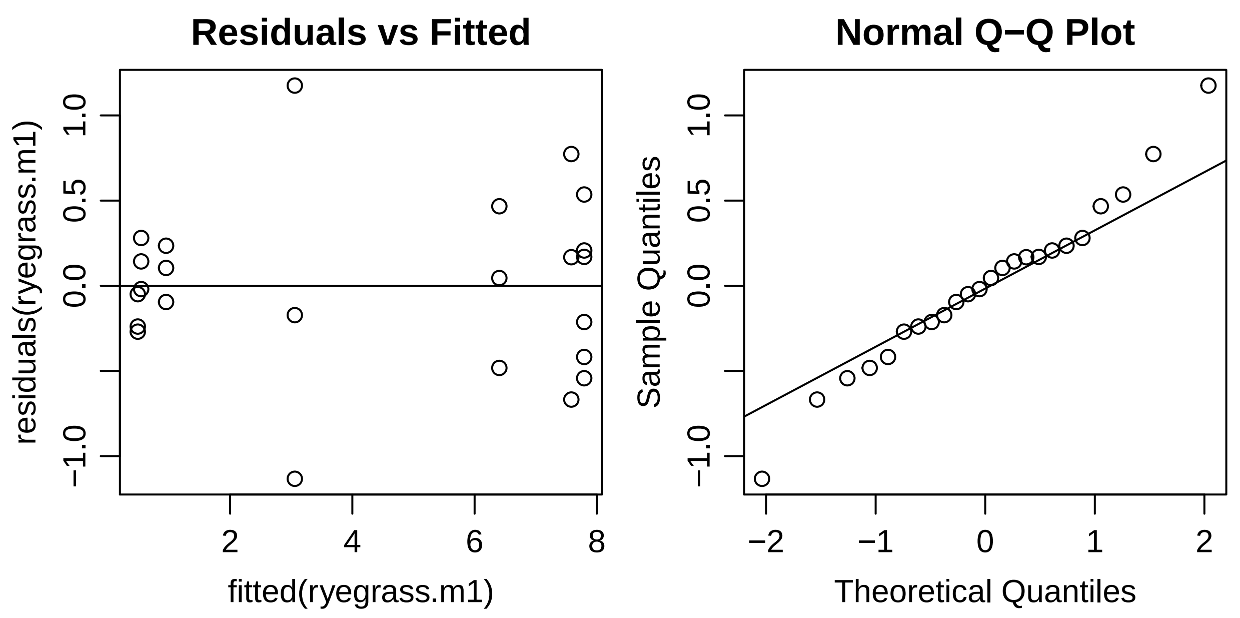


Figure 12.4: *Model esiduals diagnostic plots: Left: distribution of residuals against the fitted value (variance homogeneity); Right: normal q-q plot illustrating whether the residuals are normally distributed (normally distributed measurement errors).*

(and perhaps *should* differ) depending on the nonlinear model being used. Therefore, the R^2 is not part of the output from `drc`. To give an example we fit a linear model to the ryegrass data.

```
op <- par(mfrow = c(1, 2), mar=c(3.2,3.2,0,.5), mgp=c(2,.7,0)) #put two graphs together
plot(rootl ~ conc, data = ryegrass, bty="l")
linear.m1 <- lm(rootl ~ conc, data = ryegrass)
summary(linear.m1)
```

```
##
## Call:
## lm(formula = rootl ~ conc, data = ryegrass)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4399 -1.7055  0.9623  1.7532  2.3575
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.24176    0.52973  11.783 5.64e-11 ***
## conc        -0.25929    0.04326  -5.994 4.94e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.07 on 22 degrees of freedom
## Multiple R-squared:  0.6202, Adjusted R-squared:  0.603
## F-statistic: 35.93 on 1 and 22 DF,  p-value: 4.939e-06
```

```
abline(linear.m1)
plot(linear.m1, which=1, bty="l")
```

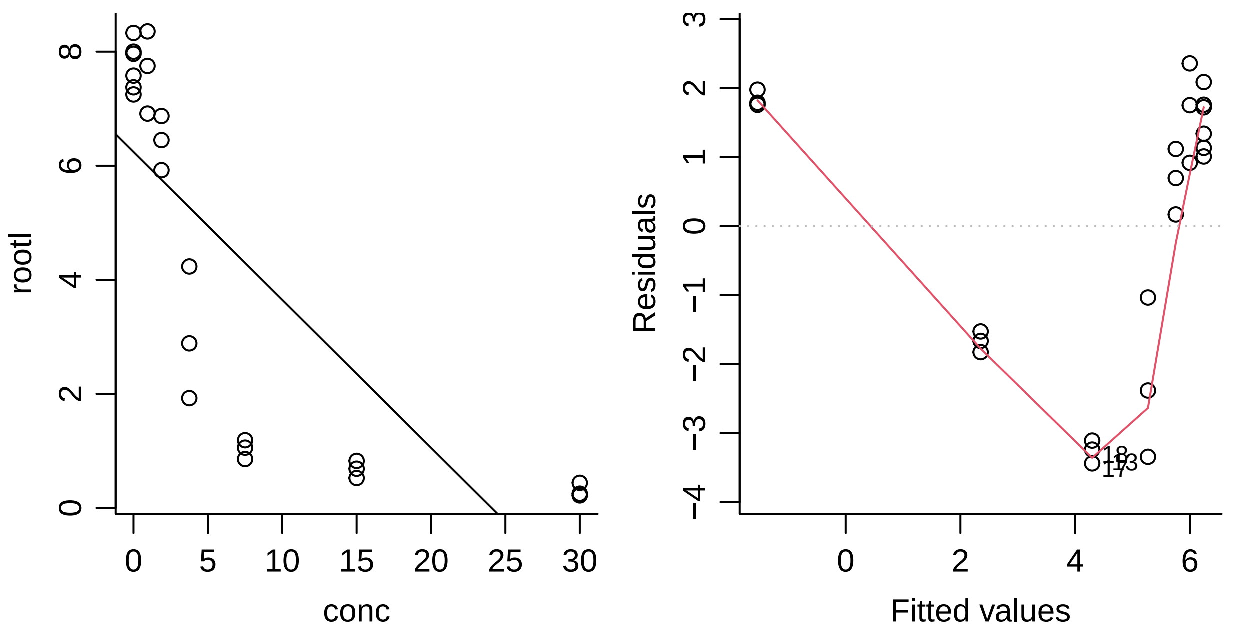


Figure 12.5: *Linear regression y=-.25x+6.24 with highly significant parameters (left) and residual plot (right). The R^2 is not negligable (R^2 = 0.6), but the model is obviously not correct.*

The regression parameters are highly significantly different from zero so it looks good as does R^2 of 0.62 . This is not a small value, but obviously the goodness of fit is not good, let alone the systematic distribution of residuals. When trying to determine whether the nonlinear model is better or worse than the linear model, Akaike Information Criterion (AIC) can be used. Smaller values of AIC indicate better fit to the data.

```
AIC(ryegrass.m1, linear.m1)
```

```
##              df       AIC
## ryegrass.m1  5  42.31029
## linear.m1    3 106.95109
```

The AIC value for the nonlinear model `ryegrass.m1` is much lower than the linear fit `linear.m1` that definitely violated item 1. This confirms that the nonlinear model is a better fit to the data compared with the linear model.

12.3 Which Dose-Response Model?

There exists numerous sigmoid curves of which the log-logistic is the most common one. It is symmetric and the ED_{50} is a parameter of the model. Several other models exist, particularly, the non symmetric ones where ED_{50} is not a "natural" parameter. For example the Weibull-1 model.

$$y = c + (d - c)exp(-exp(b(log(x) - log(e)))).$$

The parameter `b` is the steepness of the curve and the parameter `e` is the dose where the inflection point of the dose- response curve is located, which is *not* the ED_{50}.The Weibull-1 descends slowly from the upper limit `d` and approach the lower limit `c` rapidly.

Another model, Weibull-2 is somewhat different from Weibull-1, because it has a different asymmetry, with a rapid descent from the upper limit `d` and a slow approach toward the lower limit (Figure 12.6).

$$y = c + (d - c)(1 - exp(-exp(b(log(x) - log(e))))).$$

Ritz 2009 has in detailed described those asymmetric dose-response curves and also illustrated it as shown in Figure 12.6.

```
library(drc)
## Comparing log-logistic and Weibull models
## (Figure 2 in Ritz (2009))
ryegrass.m0 <- drm(rootl ~ conc, data = ryegrass, fct = LL.4())
ryegrass.m1 <- drm(rootl ~ conc, data = ryegrass, fct = W1.4())
ryegrass.m2 <- drm(rootl ~ conc, data = ryegrass, fct = W2.4())

par(mfrow=c(1,1), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(ryegrass.m0, broken=TRUE, xlab="Dose (mM)", ylab="Root length (cm)", lwd=2,
     cex=1.2, cex.axis=1.2, cex.lab=1.2, bty="l")
plot(ryegrass.m1, add=TRUE, broken=TRUE, lty=2, lwd=2)
plot(ryegrass.m2, add=TRUE, broken=TRUE, lty=3, lwd=2)

arrows(3, 7.5, 1.4, 7.5, 0.15, lwd=2)
text(3,7.5, "Weibull-2", pos=4, cex=1.2)

arrows(2.5, 0.9, 5.7, 0.9, 0.15, lwd=2)
text(3,0.9, "Weibull-1", pos=2, cex=1.2)
```

The ED_{50} for the Weibull-1 and Weibull-2 model has to be derived from the fit by using the function`ED()`.

```
 ED(ryegrass.m1,50,interval="delta")
```

```
##
## Estimated effective doses
##
##         Estimate Std. Error   Lower   Upper
## e:1:50  3.08896    0.17331 2.72744 3.45048
```

A relevant questions would be how do the ED_x change when fitting the different sigmoid curves?

To illustrate this we use the ggplot2 package and a function to combine graphs. First we have to calculate the ED_x and combine the results in a data.frame as done below.

```
edLL<-data.frame(ED(ryegrass.m0,c(10,50,90),interval="delta",
                    display=FALSE),ll="Log-logistic")
edW1<-data.frame(ED(ryegrass.m1,c(10,50,90),interval="delta",
                    display=FALSE),ll="Weibull 1")
edW2<-data.frame(ED(ryegrass.m2,c(10,50,90),interval="delta",
                    display=FALSE),ll="Weibull 2")
CombED<-rbind(edLL,edW1,edW2)
```

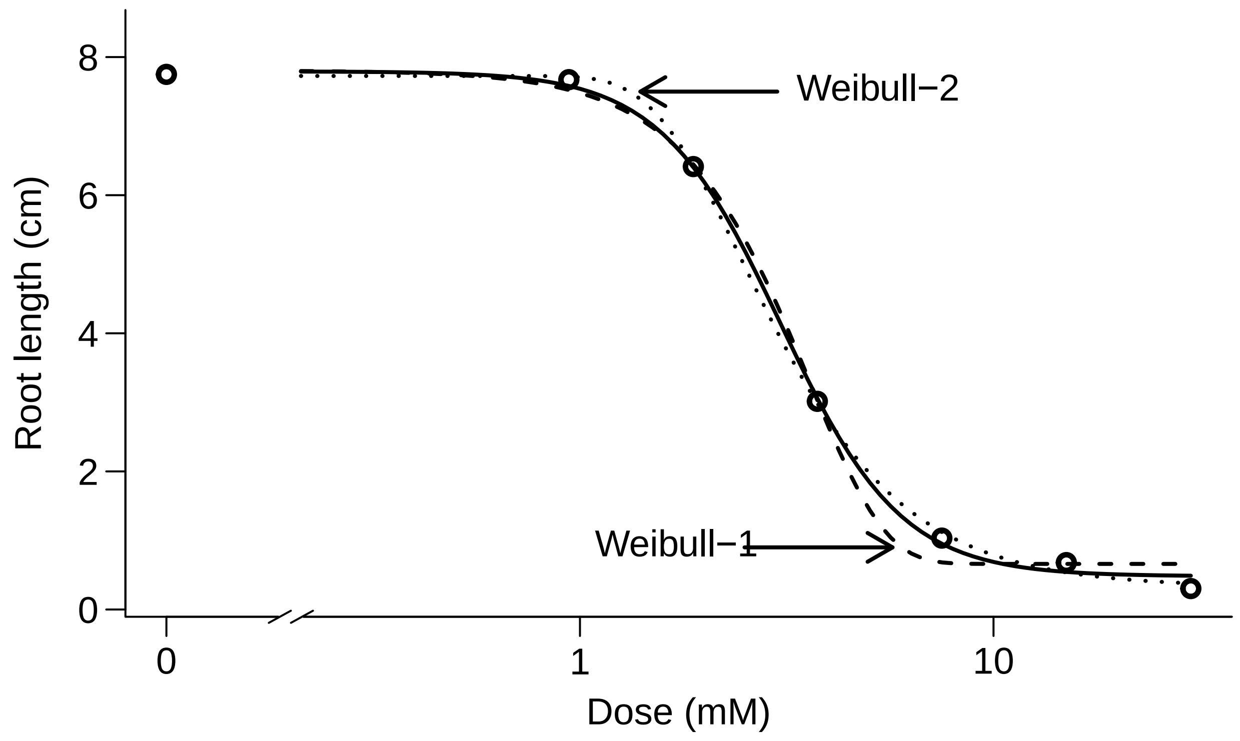


Figure 12.6: *Comparison among the two asymmetric Weibull models and the log-logistic model. The differences in this instance are at either the upper limit or the lower limit of the curves.*

Then we can use `plot_grid()` from the `cowplot` package, which is a function that combines various independent ggplots into one plot.

```
library(ggplot2)
library(cowplot)

p1 <- ggplot(data=CombED[c(1,4,7),], aes(x=ll, y=Estimate))+
 geom_bar(stat="identity", fill="lightgreen", colour="black")+
  geom_errorbar(aes(ymin=Lower, ymax=Upper), width=0.1) +
  ylab("ED10")+
  xlab("")
p2 <- ggplot(data=CombED[c(2,5,8),], aes(x=ll, y=Estimate))+
  geom_bar(stat="identity", fill="lightgreen", colour="black")+
  geom_errorbar(aes(ymin=Lower, ymax=Upper), width=0.1) +
  ylab("ED50")+
  xlab("")

p3 <- ggplot(data=CombED[c(3,6,9),], aes(x=ll, y=Estimate))+
geom_bar(stat="identity", fill="lightgreen", colour="black")+
  geom_errorbar(aes(ymin=Lower, ymax=Upper), width=0.1) +
  ylab("ED90")+
  xlab("Sigmoid four parameter models")

plot_grid(p1,p2,p3, ncol=1)
```

Obviously, the ED-levels in Figure 12.7 in this instance did not change much among the three sigmoid curves.

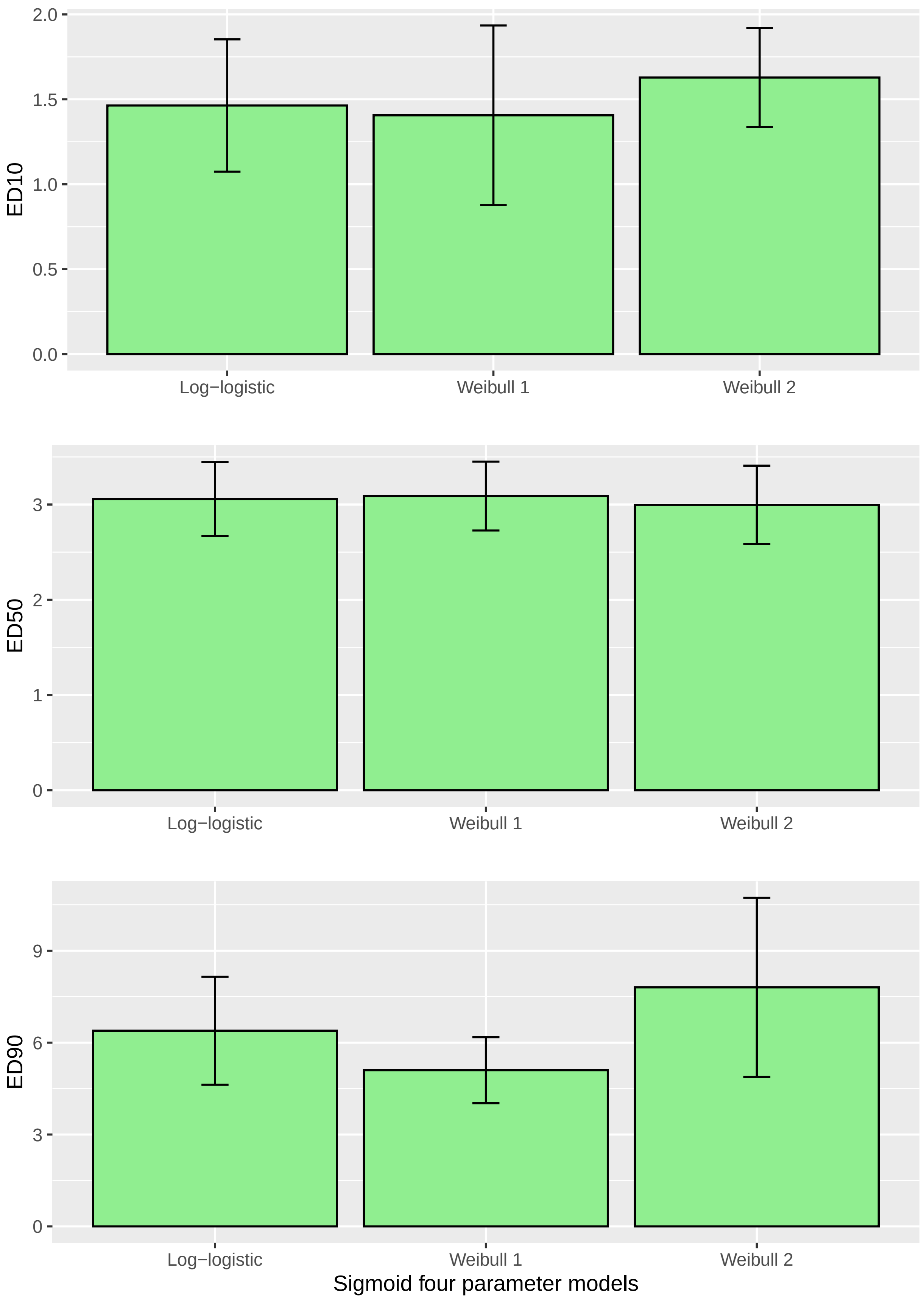


Figure 12.7: *Comparison of ED_{10}, ED_{50} and ED_{90} for various sigmoid dose-response curves with associated 95% confidence intervals.*

Sometimes they do and residual plots show that one of the asymmetrical curves is more appropriate than the common log-logistic, although it can not be substantiated by an ordinary test for lack of fit.`drc`also has a function where you can test the differences between the ED_x as shown below

```
comped(CombED[c(1,4),1],CombED[c(1,4),2],log=F,operator = "-")
```

```
##
## Estimated difference of effective doses
##
##       Estimate Std. Error     Lower Upper
## [1,]  0.057726   0.314927 -0.559519 0.675
```

```
comped(CombED[c(1,7),1],CombED[c(1,7),2],log=F,operator = "-")
```

```
##
## Estimated difference of effective doses
##
##      Estimate Std. Error    Lower  Upper
## [1,] -0.16457    0.23335 -0.62193 0.2928
```

```
comped(CombED[c(3,6),1],CombED[c(3,6),2],log=F,operator = "-")
```

```
##
## Estimated difference of effective doses
##
##      Estimate Std. Error    Lower  Upper
## [1,]  1.28762    0.99032 -0.65337 3.2286
```

```
comped(CombED[c(6,9),1],CombED[c(6,9),2],log=F,operator = "-")
```

```
##
## Estimated difference of effective doses
##
##      Estimate Std. Error   Lower  Upper
## [1,]  -2.7048     1.4939 -5.6327 0.2231
```

```
comped(CombED[c(3,9),1],CombED[c(3,9),2],log=F,operator = "-")
```

```
##
## Estimated difference of effective doses
##
##      Estimate Std. Error   Lower Upper
## [1,]  -1.4172     1.6368 -4.6253 1.791
```

```
comped(CombED[c(6,9),1],CombED[c(6,9),2],log=F,operator = "-")
```

```
##
## Estimated difference of effective doses
##
##      Estimate Std. Error   Lower  Upper
## [1,]  -2.7048     1.4939 -5.6327 0.2231
```

None of the differences are significantly different form zero, lower limits are negative in all instance and upper limits of the differences are positive.

12.4 More Dose-Response Curves

When it comes to assessment of herbicide selectivity, one dose-response curve does not tell you anything. Selectivity of a herbicide is always dose-dependent and a bioassay with a pre-fixed harvest day only gives a snapshot of the effect of a herbicide. When comparing dose-response curves of say two herbicides on the same plant species or one herbicide on two plant species, it is imperative that the curves you compare were run at the same point in time or at least close to and at the same stage of plant development. The training data set `S.alba` in the `drc` package consists of two dose-response curves, one for bentazon and one for glyphosate.

```
head(S.alba,n=2)
```

```
##   Dose  Herbicide DryMatter
## 1    0 Glyphosate       4.7
## 2    0 Glyphosate       4.6
```

```
tail(S.alba,n=2)
```

```
##    Dose Herbicide DryMatter
## 67  640 Bentazone       0.6
## 68  640 Bentazone       0.8
```

```
## Fitting a log-logistic model with
S.alba.m1 <- drm(DryMatter ~ Dose, Herbicide, data=S.alba, fct = LL.4())
```

Note that the the variable, Herbicide, identifies the herbicides, Therefore, the third argument in the `drm()` is the classification variable `Herbicide`.

```
modelFit(S.alba.m1)
```

```
## Lack-of-fit test
##
##           ModelDf    RSS Df F value p value
## ANOVA          53 8.0800
## DRC model      60 8.3479  7  0.2511  0.9696
```

```
summary(S.alba.m1)
```

```
##
## Model fitted: Log-logistic (ED50 as parameter) (4 parms)
##
## Parameter estimates:
##
##               Estimate Std. Error t-value   p-value
## b:Glyphosate  2.715409   0.748279  3.6289 0.0005897 ***
## b:Bentazone   5.134810   1.130949  4.5403 2.760e-05 ***
## c:Glyphosate  0.891238   0.194703  4.5774 2.421e-05 ***
## c:Bentazone   0.681845   0.095111  7.1689 1.288e-09 ***
## d:Glyphosate  3.875759   0.107463 36.0661 < 2.2e-16 ***
## d:Bentazone   3.805791   0.110341 34.4911 < 2.2e-16 ***
## e:Glyphosate 62.087606   6.611444  9.3909 2.184e-13 ***
## e:Bentazone  29.268444   2.237090 13.0833 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error:
##
##  0.3730047 (60 degrees of freedom)
```

The test for lack of fit shows non-significance, and furthermore, the summary shows that the upper limits `ds` are very close to each other and the same applies to the lower limit `cs`. Consequently, we could entertain the idea that they are similar for both curves. We use the argument in the `drm()` function `pmodels=data.frame(..)`to make the upper limit and the lower limit similar for both curves. The argument follows the alphabetical order of the names of the parameters, `b,c,d,e`. Below, we will allow the `b` parameter to vary freely between the two herbicides, whereas `c` and `d` will be held the same for both herbicides (they have the same identification `1,1`). Finally the `e` parameters can vary freely.

```
##  common lower and upper limits
S.alba.m2 <- drm(DryMatter ~ Dose, Herbicide, data=S.alba, fct = LL.4(),
                 pmodels=data.frame(Herbicide, 1, 1, Herbicide))
summary(S.alba.m2)
```

```
##
## Model fitted: Log-logistic (ED50 as parameter) (4 parms)
##
## Parameter estimates:
##
##                 Estimate Std. Error t-value   p-value
## b:Bentazone    5.046141   1.040135  4.8514 8.616e-06 ***
## b:Glyphosate   2.390218   0.495959  4.8194 9.684e-06 ***
## c:(Intercept)  0.716559   0.089245  8.0291 3.523e-11 ***
## d:(Intercept)  3.854861   0.076255 50.5519 < 2.2e-16 ***
## e:Bentazone   28.632355   2.038098 14.0486 < 2.2e-16 ***
## e:Glyphosate  66.890545   5.968819 11.2067 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error:
##
##  0.3705151 (62 degrees of freedom)
```

```
par(mfrow=c(1,1), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(S.alba.m1, broken=TRUE, bty="l")
plot(S.alba.m2, col="red", broken=TRUE, add=TRUE, legend=FALSE)
```

```
anova(S.alba.m1, S.alba.m2) # Test for lack of fit not signifiancant
```

```
##
## 1st model
##  fct:     LL.4()
##  pmodels: Herbicide, 1, 1, Herbicide
## 2nd model
##  fct:     LL.4()
##  pmodels: Herbicide (for all parameters)
```

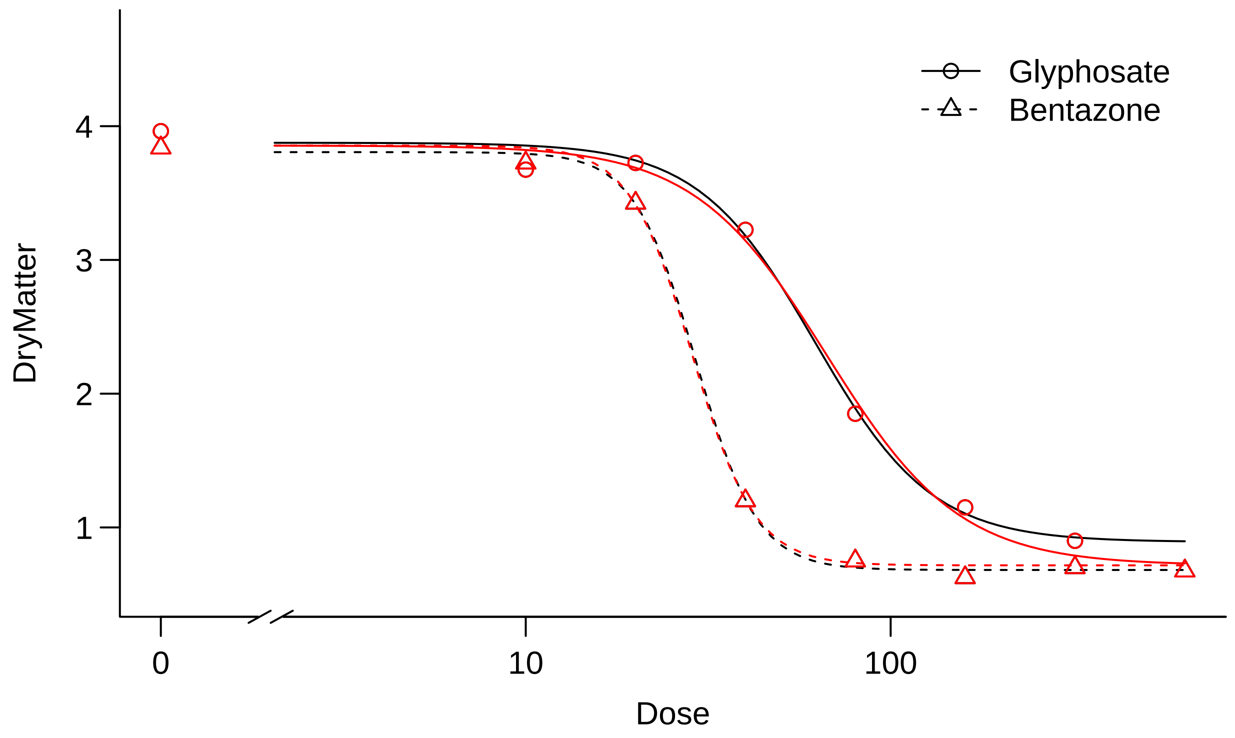


Figure 12.8: *The two fits, one without assuming common upper and lower limits for the two curves (black) and one where the curves have the same upper and lower limits (red).*

```
## ANOVA table
##
##            ModelDf    RSS Df F value p value
## 2nd model      62 8.5114
## 1st model      60 8.3479  2  0.5876  0.5588
```

A test for lack of fit was non-significant and, therefore, we presume that the curves have similar upper an lower limits. Now we have the ideal framework for comparing the relative potency between the two herbicides with similar upper and lower limit, we have the same reference for both curves.

Comparing the fit without freely estimated upper and lower limit and with common upper and lower limit did not change the ED_{50} much but did reduce the standard errors somewhat.

The next step could be to assume that the two curves also have the same slope, i.e. the two curves are similar and only differ in their relative displacement along the x-axis.

```
##  common lower and upper limits and slope
S.alba.m3 <- drm(DryMatter~Dose, Herbicide, data=S.alba, fct = LL.4(),
                 pmodels=data.frame(1, 1, 1, Herbicide))
anova(S.alba.m2, S.alba.m3)
```

```
##
## 1st model
##  fct:     LL.4()
##  pmodels: 1, 1, 1, Herbicide
## 2nd model
##  fct:     LL.4()
##  pmodels: Herbicide, 1, 1, Herbicide

## ANOVA table
##
##           ModelDf    RSS Df F value p value
## 2nd model      63 9.4946
## 1st model      62 8.5114  1  7.1614  0.0095
```

Although the ‘parallel’ curves in Figure 12.9 seem reasonable, the test for lack of fit now is significant, so the assumption of similar curves except for the ED_{50} is not supported. We should prefer, instead, the model with independent slopes (Figure 12.8).

```
par(mfrow=c(1,1), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(S.alba.m3, broken=TRUE, bty="l")
```

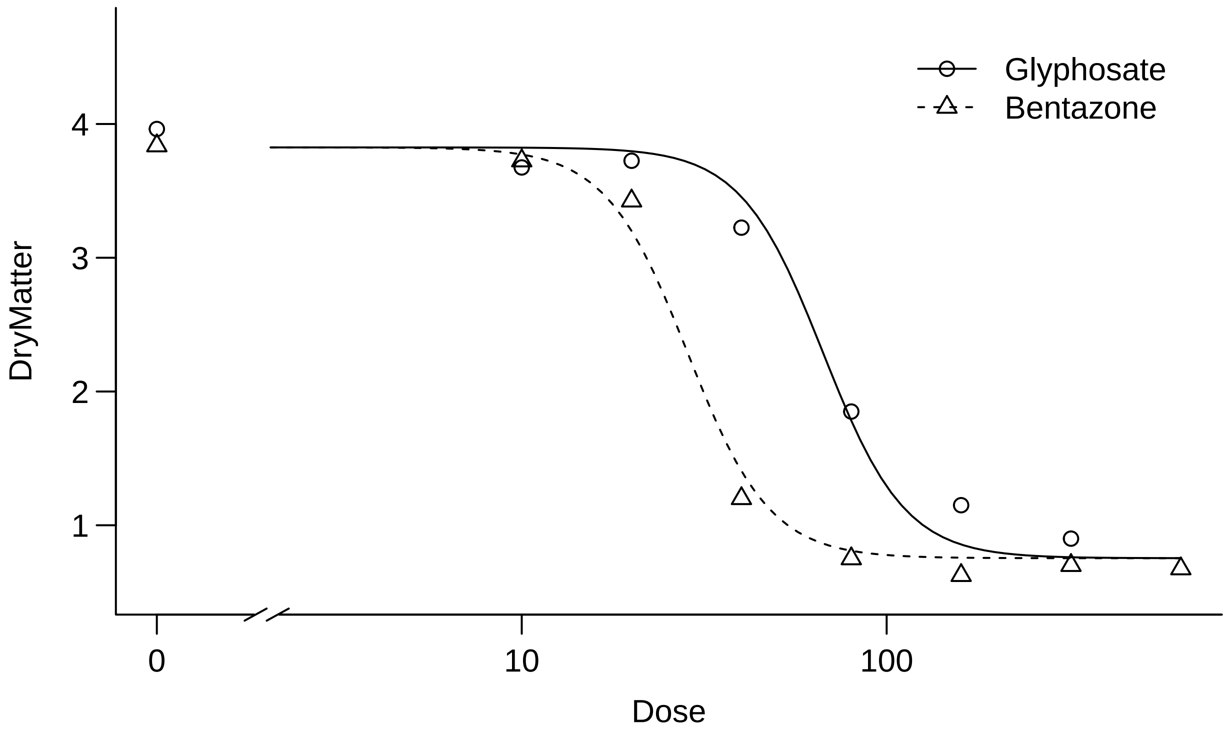


Figure 12.9: *The two curves are similar except for their relative displacement on the x-axis - this is often called a parallel slopes model.*

Comparing Figures 12.8 and 12.9 shows that the fits are forced so they do not capture the responses properly at the lower limits. As seen below the relative potency with common upper and lower limit but different slopes yield different relative potencies: ED_{10}, ED_{50}, and ED_{90}.

```
#EDcomp(S.alba.LL.4.1, c(10, 50, 50), interval = "delta")
EDcomp(S.alba.m2, c(10, 10), interval="delta", reverse=TRUE)
```

```
##
## Estimated ratios of effect doses
```

```
##
##                               Estimate   Lower   Upper
## Glyphosate/Bentazone:10/10  1.44007 0.81836 2.06177
```

```
EDcomp(S.alba.m2, c(50, 50), interval="delta", reverse=TRUE)
```

```
##
## Estimated ratios of effect doses
##
##                               Estimate  Lower  Upper
## Glyphosate/Bentazone:50/50   2.3362 1.8571 2.8153
```

```
EDcomp(S.alba.m2, c(90, 90), interval="delta", reverse=TRUE)
```

```
##
## Estimated ratios of effect doses
##
##                               Estimate  Lower  Upper
## Glyphosate/Bentazone:90/90   3.7899 2.0793 5.5006
```

It is noted above that the relative potency varies depending of the ED-level. This is of course due to the fact that the two curves have different slopes, `b` parameters. From a biological point of view the herbicides have different mode of action and experience shows that contact PS II herbicides, such as bentazon, would have a steeper slope than does the EPSP inhibitor, glyphosate.

In `drc` there is a function `relpot()` that can illustrate the relationship between changes of the relative potency and the predicted responses.

```
par(mfrow=c(1,1), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
relpot(S.alba.m2, interval = "delta", bty="l")
```

The most precise ED-level is at the ED_{50}; towards the extremes close to the upper and lower limit the variation becomes huge. This is a general trend and explains why we often wish to quantify toxicity of a compound by using ED_{50}.

12.5 When Upper and Lower Limits are not similar

In real life, e.g., when testing herbicide tolerant/resistant biotypes of weeds and different weed species, we cannot expect that the upper and lower limits are similar among curves and that the curves have the same slopes. On the contrary, if there is a penalty for acquiring resistance (fitness cost) then the untreated control for each biotype would be expected to have different biomass. Depending on the time from spraying and ambient growth condition at harvest; the lower limit may also differ.

Whatever the purpose of assessing potencies among plants the problem of different upper and lower limits is often "solved" in the literature by scaling the raw data to percentage of untreated control. This "solution" often results in smaller but incorrect standard error of ED_{50} being inflated by the scaling.

In general, we do not want to go for low standard error, but go for the *correct* standard error. Therefore, we recommend to do all the curve-fitting on original raw data!

Below we have two accessions of *Echonichlora oryzoides* where both the upper limit and the lower limits are different.

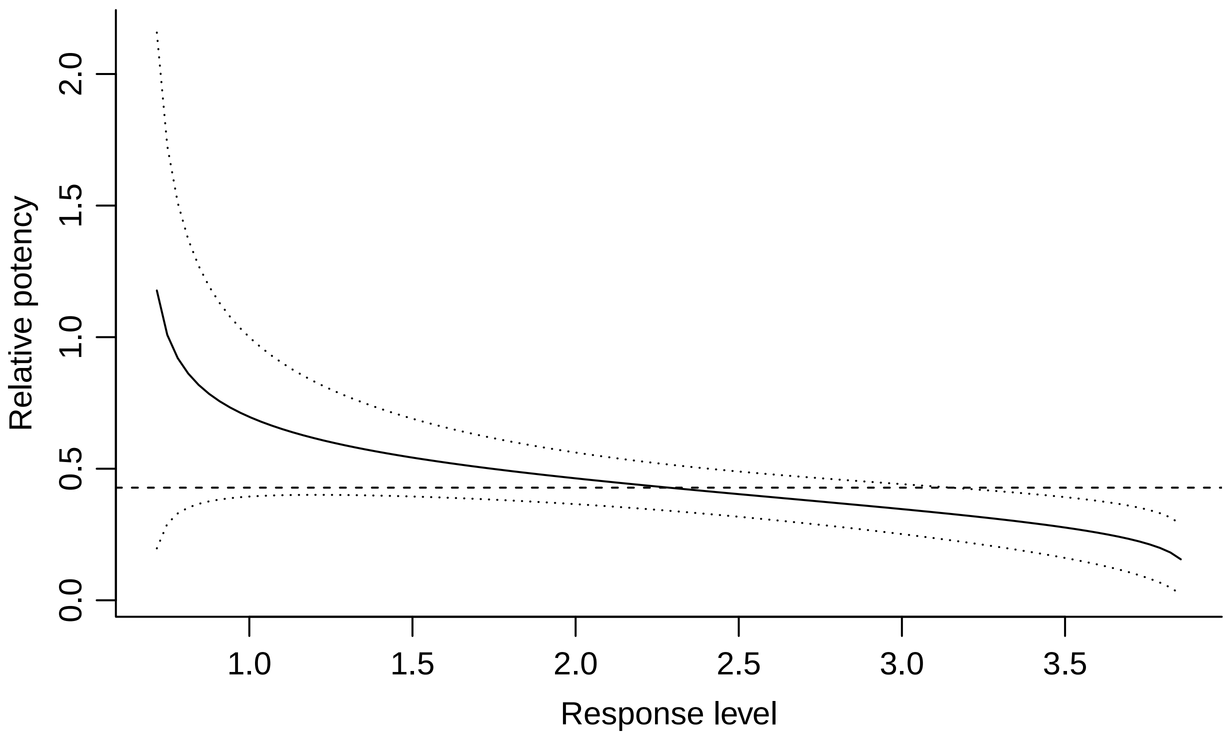


Figure 12.10: *The relative potency varies depending of response level, with the 95% confidence intervals. The dashed horizontal line shows the relative potency at the ED_{50} for glyphosate and bentazon.*

```r
op <- par(mfrow = c(1, 1), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
TwoCurves<-read.csv("http://rstats4ag.org/data/twoCurves.csv")
head(TwoCurves,3)
```

```
##   xn Dose Accesion Dry.weight.per.pot  X
## 1 33    0  SAM-E-8              1.968 NA
## 2 34    0  SAM-E-8              2.030 NA
## 3 35    0  SAM-E-8              2.129 NA
```

```r
Bispyribac.sodium.m1 <- drm(Dry.weight.per.pot ~ Dose, Accesion,
                            data = TwoCurves, fct=LL.4())
modelFit(Bispyribac.sodium.m1)
```

```
## Lack-of-fit test
##
##           ModelDf    RSS Df F value p value
## ANOVA          48 1.7678
## DRC model      56 2.2799  8  1.7384  0.1137
```

```r
summary(Bispyribac.sodium.m1)
```

```
##
## Model fitted: Log-logistic (ED50 as parameter) (4 parms)
```

```
##
## Parameter estimates:
##
##               Estimate Std. Error t-value   p-value
## b:SAM-E-8    2.047171   0.327585  6.2493 5.957e-08 ***
## b:TEK-E-10   0.876716   0.315014  2.7831  0.007327 **
## c:SAM-E-8    0.245913   0.073938  3.3259  0.001560 **
## c:TEK-E-10 -0.053077   0.091626 -0.5793  0.564726
## d:SAM-E-8    2.242014   0.081575 27.4841 < 2.2e-16 ***
## d:TEK-E-10   1.827553   0.101102 18.0764 < 2.2e-16 ***
## e:SAM-E-8   21.702693   2.276642  9.5328 2.514e-13 ***
## e:TEK-E-10   2.666463   1.026905  2.5966  0.012004 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error:
##
##  0.2017751 (56 degrees of freedom)
```

```r
plot(Bispyribac.sodium.m1, broken=TRUE, bp=.1, ylab="Dry Matter(g/pot)",
     xlab="Bispyribac sodium (g/ha)", bty="l")

# drawing lines on the plot
ED.SAM.E.8 <- 0.24591 + (2.24201 - 0.24591) * 0.5
ED.TEK.E.10 <- -0.05308 + (1.82755 +- 0.05308) * 0.5
arrows(0.1, ED.SAM.E.8, 21.7, ED.SAM.E.8, code=0)
arrows(21.7, ED.SAM.E.8, 21.7, 0, code=0)
arrows(0.1, ED.TEK.E.10, 2.67, ED.TEK.E.10, code=0, lty=2)
arrows(2.67, ED.TEK.E.10, 2.67, 0, code=0, lty=2)
```

The fit looks reasonable and the test for lack of fit is non-significant and the ED_{50}s are shown. Everything seems to be all right, but we notice that distribution of observations for TEK-E-10 could be better to describe the mid range of the responses. The summary above reveals that the lower limit, `c`, for TEK-E-10 is less than zero, not much and not different from zero though, but still illogical. We can re-fit the model using the `lowerl=` argument which will allow us to restrict model parameters to biologically relevant estimates. In this case, we will restrict the lower limit to values of zero or greater. All other parameters will still be allowed to vary without restriction.

```r
Bispyribac.sodium.m2 <- drm(Dry.weight.per.pot ~ Dose, Accesion,
                            data = TwoCurves, fct=LL.4(),
                            lowerl=c(NA,0,NA,NA))
summary(Bispyribac.sodium.m2)
```

```
##
## Model fitted: Log-logistic (ED50 as parameter) (4 parms)
##
## Parameter estimates:
##
##               Estimate Std. Error t-value   p-value
## b:SAM-E-8    2.042454   0.329052  6.2071 6.984e-08 ***
## b:TEK-E-10   1.015226   0.313919  3.2340  0.002049 **
## c:SAM-E-8    0.243411   0.074596  3.2631  0.001881 **
## c:TEK-E-10   0.000000   0.071617  0.0000  1.000000
```

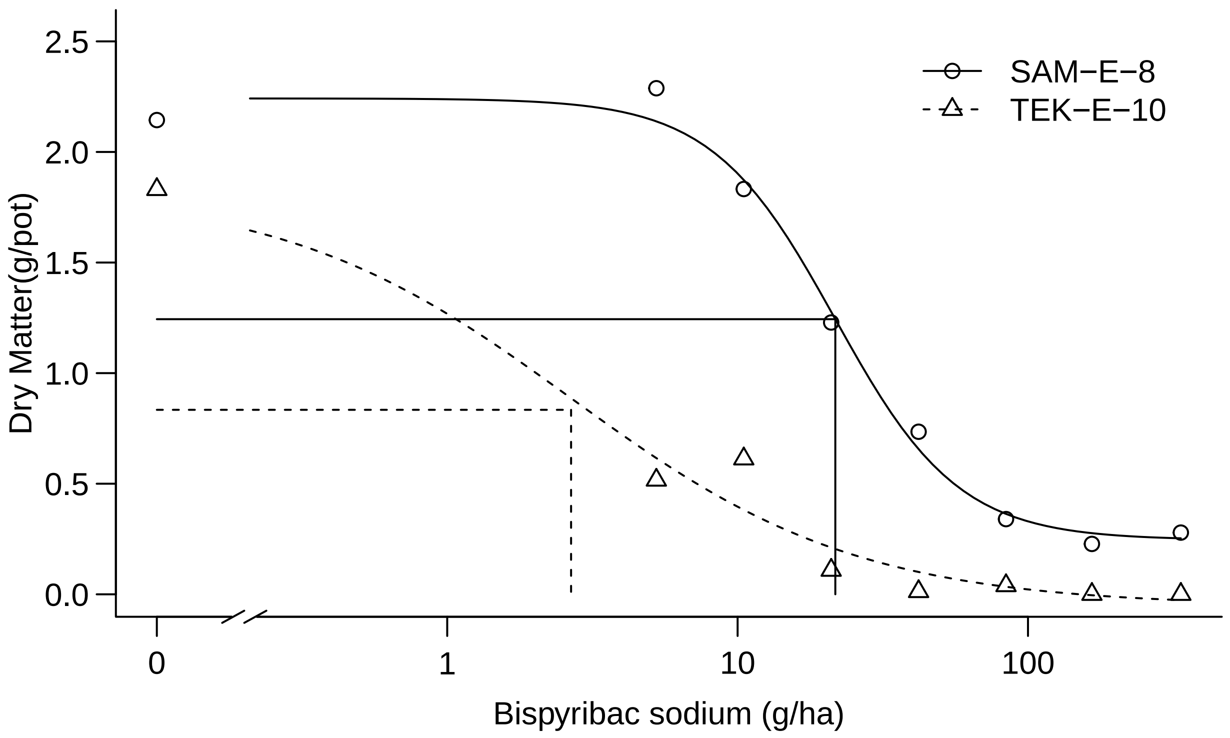


Figure 12.11: *Fit of regression for two accessions.*

```
## d:SAM-E-8   2.242077   0.082057 27.3234 < 2.2e-16 ***
## d:TEK-E-10  1.826855   0.101703 17.9627 < 2.2e-16 ***
## e:SAM-E-8  21.740911   2.295727  9.4702 3.166e-13 ***
## e:TEK-E-10  2.730186   0.961385  2.8398  0.006280 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error:
##
##  0.2028892 (56 degrees of freedom)
```

We see that the standard error dropped and now we will use the fit and calculate the selectivity factor at ED_{50}:

```
EDcomp(Bispyribac.sodium.m2, c(50, 50),interval="delta")
```

```
##
## Estimated ratios of effect doses
##
##                         Estimate   Lower   Upper
## SAM-E-8/TEK-E-10:50/50   7.9632  2.0988 13.8275
```

The resistance factor is about 8 and significantly greater than 1. Note this is one of the few times we do not test the null hypothesis but the hypothesis, that the potency is the same for the two accessions; that is, the relative potency is 1.0. Another important issue is that the dry matter for the ED_{50} differs because of the different upper and lower limits. This will not be solved by scaling the observation to the individual accessions' untreated control.

Obviously, the ED_{50} is not at the same response level, and the relative potency of 8 is significantly different form 1.00.

When using absolute dry matter of say 1.24, the response at ED_{50} for SAM-E-8 and TEK-E-10 is seen below.

```
ED(Bispyribac.sodium.m2, 1.24, type="absolute", interval="delta")
```

```
##
## Estimated effective doses
##
##                  Estimate Std. Error     Lower     Upper
## e:SAM-E-8:1.24  21.799437   2.302921 17.186132 26.412741
## e:TEK-E-10:1.24  1.306694   0.698047 -0.091663  2.705051
```

Looking at the distribution of observation for TEK-E-10, there are no observations to determine the shape of the curve above the ED_{50}.

If this is the right way to compare absolute response level we could use $ED_{1.24}$ (Figure 12.12).

```
par(mfrow = c(1, 1), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(Bispyribac.sodium.m2, broken=TRUE, bp=.1, bty="l",
     ylab="g Dry Matter/pot",
     xlab="Dose (g Bispyribac sodium/ha)")
arrows(  0.1, 1.24, 21.79, 1.24, code=0, lty=1, col="red")
arrows(21.79, 1.24, 21.79,    0, code=0, lty=1, col="red")
arrows(  1.31, 1.24,   1.31,    0, code=0, lty=2, col="red")
```

Now the resistance factor below becomes huge, but so does the confidence interval and consequently the resistance factor is not statistically different from 1.0 at an alpha ov 0.05. The villain is the poor precision of TEK-E-10 $ED_{1.24}$ that is not significantly different form 0.0.

```
EDcomp(Bispyribac.sodium.m2, c(1.24, 1.24), type="absolute",interval="delta")
```

```
##
## Estimated ratios of effect doses
##
##                           Estimate  Lower  Upper
## SAM-E-8/TEK-E-10:1.24/1.24   16.683 -1.516 34.882
```

The problems described above is virtually never mentioned in the weed science journals, even though it is instrumental to understand what we actually compare. The researcher must choose, and the choice has nothing to do with statistics but with biology.

On the basis of the fit above we can actually scale data on the basis of the regression parameters of the `d` and the `c` parameters.

```
SAM.E.8.C<-coef(Bispyribac.sodium.m1)[3]
TEK.E.10.C<-coef(Bispyribac.sodium.m1)[4]
SAM.E.8.d<-coef(Bispyribac.sodium.m1)[5]
```

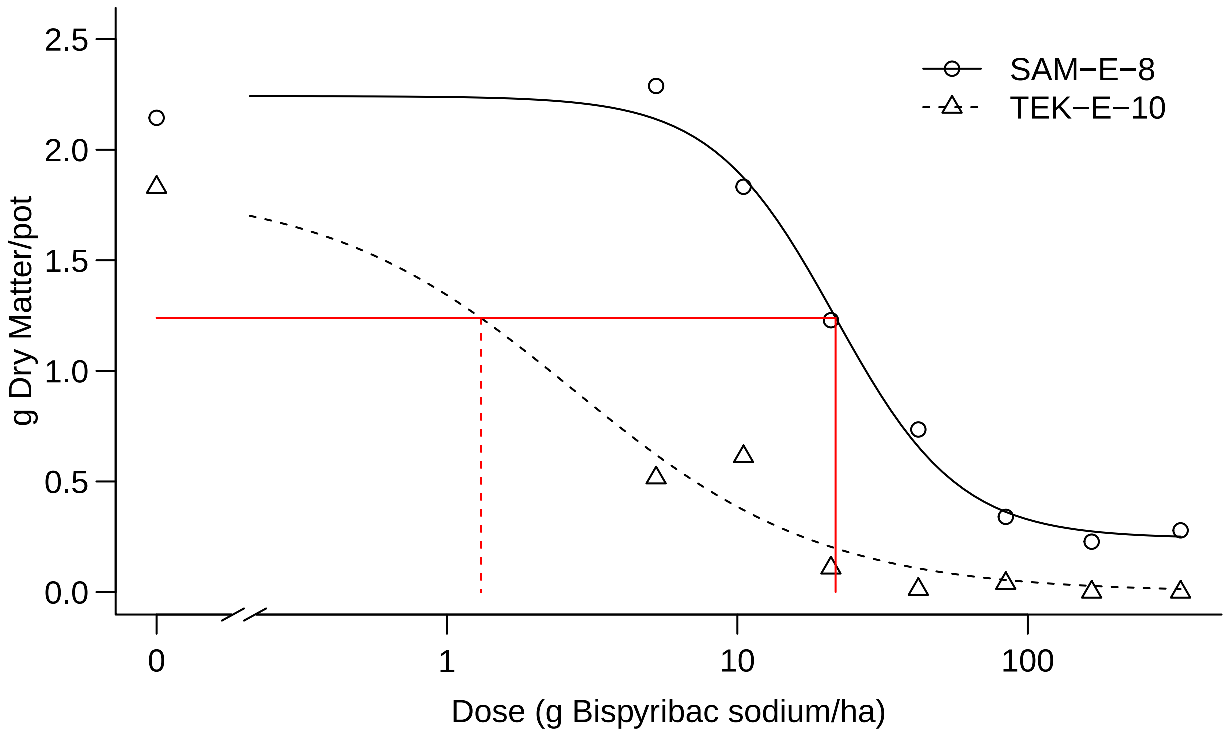


Figure 12.12: *Comparison of the two dose-response curves; red vertical lines show the dose of bispyribac sodium resulting in 1.24 g of dry matter (horizontal line) for the two biotypes.*

```
TEK.E.10.d<-coef(Bispyribac.sodium.m1)[6]

Rel.Y<-with(TwoCurves,
            ifelse(Accesion=="SAM-E-8",
                   ((Dry.weight.per.pot-SAM.E.8.C)/(SAM.E.8.d-SAM.E.8.C)),
                   ((Dry.weight.per.pot-TEK.E.10.C)/(TEK.E.10.d-TEK.E.10.C))))

Scaled.m1<-drm(Rel.Y ~ Dose, Accesion, pmodels=data.frame(Accesion, 1, Accesion),
        data=TwoCurves, fct=LL.3())
par(mfrow = c(1, 1), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(Scaled.m1, broken=TRUE, bp=.1, bty="l",
     ylab="Relative Dry Matter", ylim=c(0,1.2),
     xlab="Bispyribac-sodium dose (g/ha)")
arrows(  0.1, .5, 21.7,.5, code=0, lty=1, col="red")
arrows(21.79, 0.5, 21.79,    0, code=0, lty=1, col="red")
arrows(  2.66, .5,   2.66,    0, code=0, lty=2, col="red")
```

The summary of the fit in Figure 12.13 shows that we get almost the same parameters for the scaled ED_{50}'s and the selectivity factor is the same but what is concealed by fitting the scaled responses is that the dry matter production is not the same for the two curves, because the scales are originally different. In essence, scaling the data in this way removes the valuable information contained in the d parameter without adding benefit.

```
summary(Scaled.m1)
```

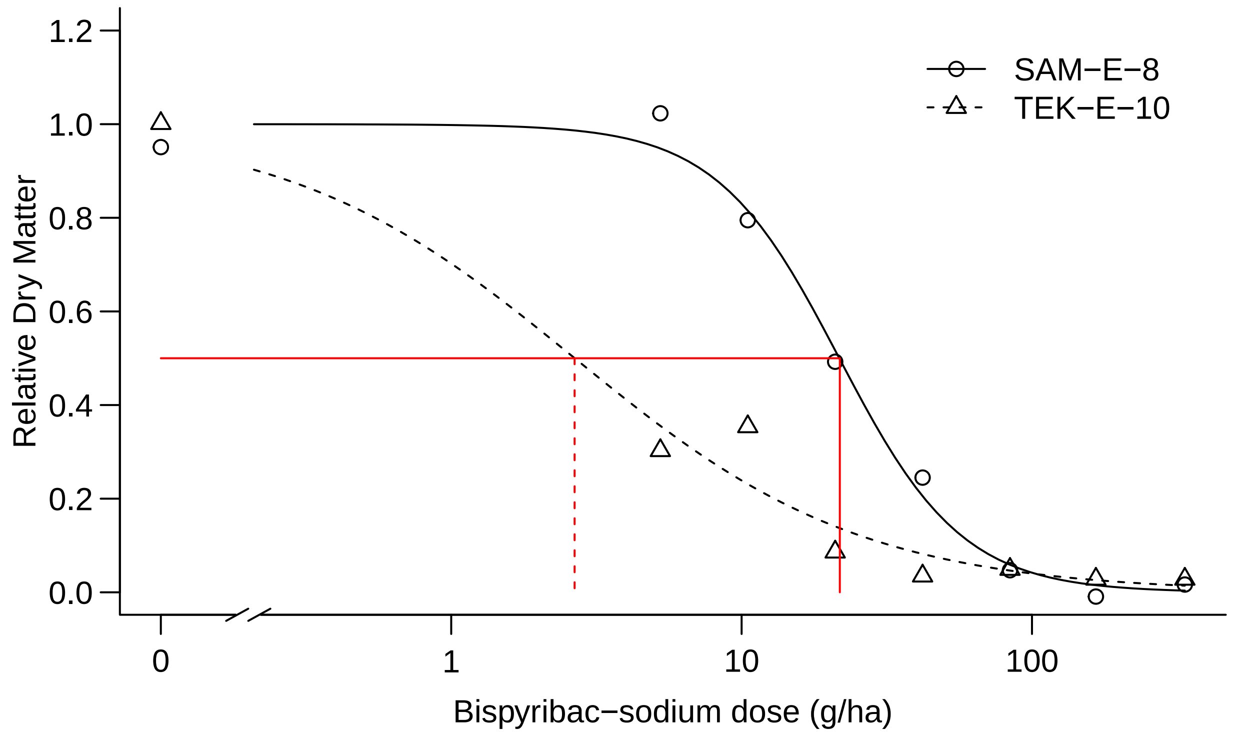


Figure 12.13: *Regression fit of scaled responses with similar upper limit,* `d`*, and lower limit set to zero.*

```
##
## Model fitted: Log-logistic (ED50 as parameter) with lower limit at 0 (3 parms)
##
## Parameter estimates:
##
##                Estimate Std. Error t-value   p-value
## b:SAM-E-8      2.048548   0.264669  7.7400 1.507e-10 ***
## b:TEK-E-10     0.875897   0.197672  4.4311 4.133e-05 ***
## d:(Intercept)  0.999972   0.031495 31.7504 < 2.2e-16 ***
## e:SAM-E-8     21.697160   1.825621 11.8848 < 2.2e-16 ***
## e:TEK-E-10     2.662759   0.924541  2.8801  0.005533 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error:
##
##  0.1005159 (59 degrees of freedom)
```

```
EDcomp(Scaled.m1, c(50, 50),interval="delta")
```

```
##
## Estimated ratios of effect doses
##
##                        Estimate   Lower   Upper
## SAM-E-8/TEK-E-10:50/50   8.1484  2.4492 13.8475
```

In summary, fitting on raw data the relative potency is 7.96 (2.93) with a coefficient of variation of 37% with the fit on scaled data on the basis of the upper limits of the regression fit on raw data the relative potency also is 8.14 (2.84) with a coefficient of variation of 32%. In this instance not dramatically different, except that for the raw data the relative potency is significantly different from zero on the 2% level whereas for the fit on relative data it is significant on the 1% level. However, if we go for an absolute ED at the dry matter of 1.24 the the relative potency is 16, but not significantly different from zero. In the literature the differences between untreated controls among accessions can be much larger than here.

12.6 Remedy for heterogeneous variance

Turning back to the ryegrass dataset the plot of the fit with all observations and the residual plot suggest we should do some kind of transformation or weighing. Because the log-logistic relationship is on the original scale, we cannot just make a transformation of response without doing the same on the right hand side of the equation. `drc` has a function that can do a a BoxCox transform-both-side regression. You can get more information by executing `?boxcox.drc`.

- Fit the original model
- update the fit with the function boxcox()

```r
par(mfrow = c(1, 1), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
ryegrass.BX <- boxcox(ryegrass.m1,method="anova")
```

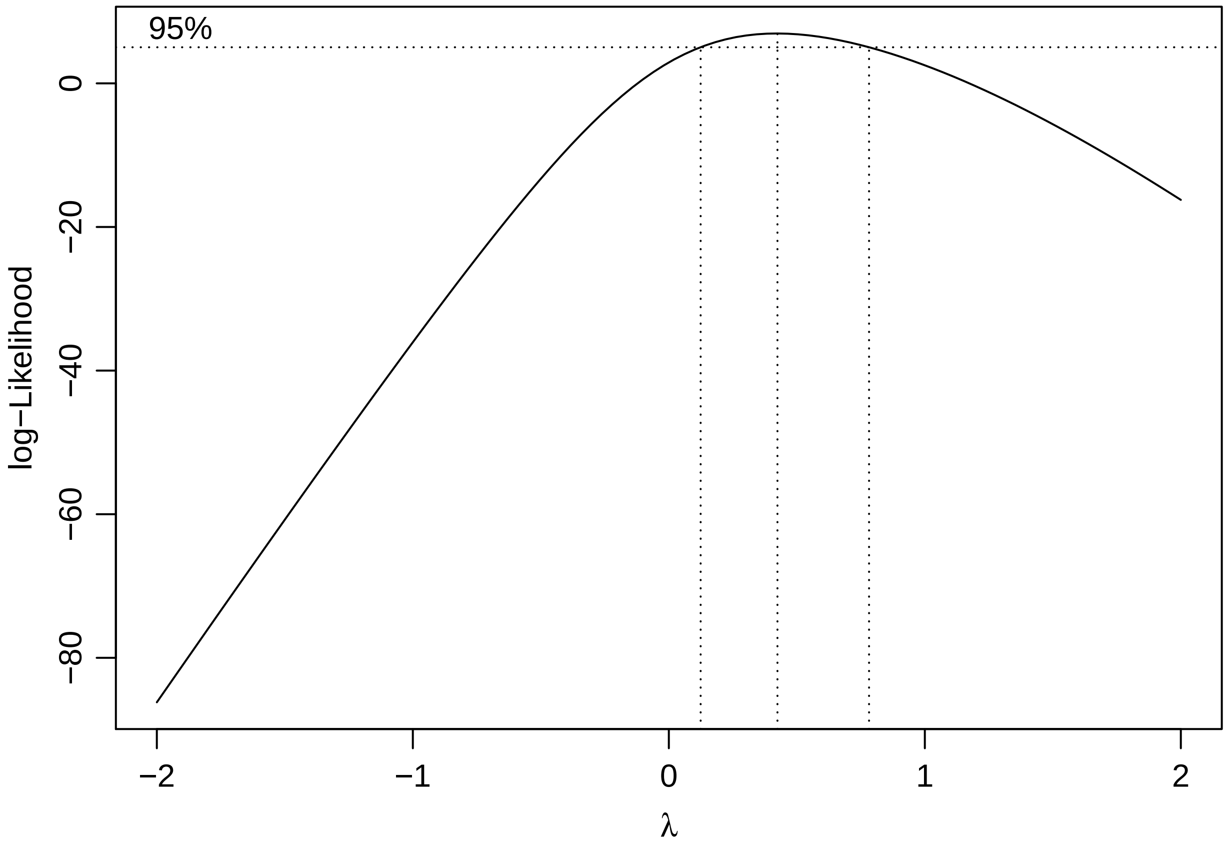


Figure 12.14: *Search for optimum λ with confidence intervals. The chosen λ is 0.42.*

The summary below give the parameters with the BoxCox transformation.

```
summary(ryegrass.BX)
```

```
##
## Model fitted: Weibull (type 1) (4 parms)
##
## Parameter estimates:
##
##                Estimate Std. Error t-value   p-value
## b:(Intercept) 1.573865   0.223277  7.0489 7.772e-07 ***
## c:(Intercept) 0.476866   0.090536  5.2672 3.731e-05 ***
## d:(Intercept) 8.007031   0.393446 20.3510 7.759e-15 ***
## e:(Intercept) 3.845926   0.308134 12.4813 6.764e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error:
##
##  0.3258055 (20 degrees of freedom)
##
## Non-normality/heterogeneity adjustment through optimal Box-Cox transformation
##
## Estimated lambda: 0.424
## Confidence interval for lambda: [0.124,0.782]
```

```
ED(ryegrass.BX,50,interval="delta")
```

```
##
## Estimated effective doses
##
##        Estimate Std. Error   Lower   Upper
## e:1:50  3.04695    0.29106 2.43981 3.65409
```

The changes in parameters and their standard deviations in relation to the original fit are not dramatic. Normally, if the heterogeneity of variance is moderate and the response curve is well described by responses covering the response range, the changes in the parameters would be small, but perhaps there will be changes in the standard errors. Consequently, if the objective is to compare potency among compounds the prerequisite for constant variance may be important to draw the optimal conclusions.

The residuals after the BoxCox update look a bit better now, compared to the original one.

```
#Graphical analysis of residuals
op <- par(mfrow = c(1, 2), mar=c(3.2,3.2,2,.5), mgp=c(2,.7,0)) #put two graphs together
    plot(residuals(ryegrass.BX) ~ fitted(ryegrass.BX), main="Residuals vs Fitted")
abline(h=0)
qqnorm(residuals(ryegrass.BX))
 qqline(residuals(ryegrass.BX))
```

12.7 Dose-responses with binomial data

Survival of plant or ability to germinate in response to herbicides is a classical underlying construct in toxicology. In weed science it also is important because *the only good weed is a dead weed.* Particularly, in

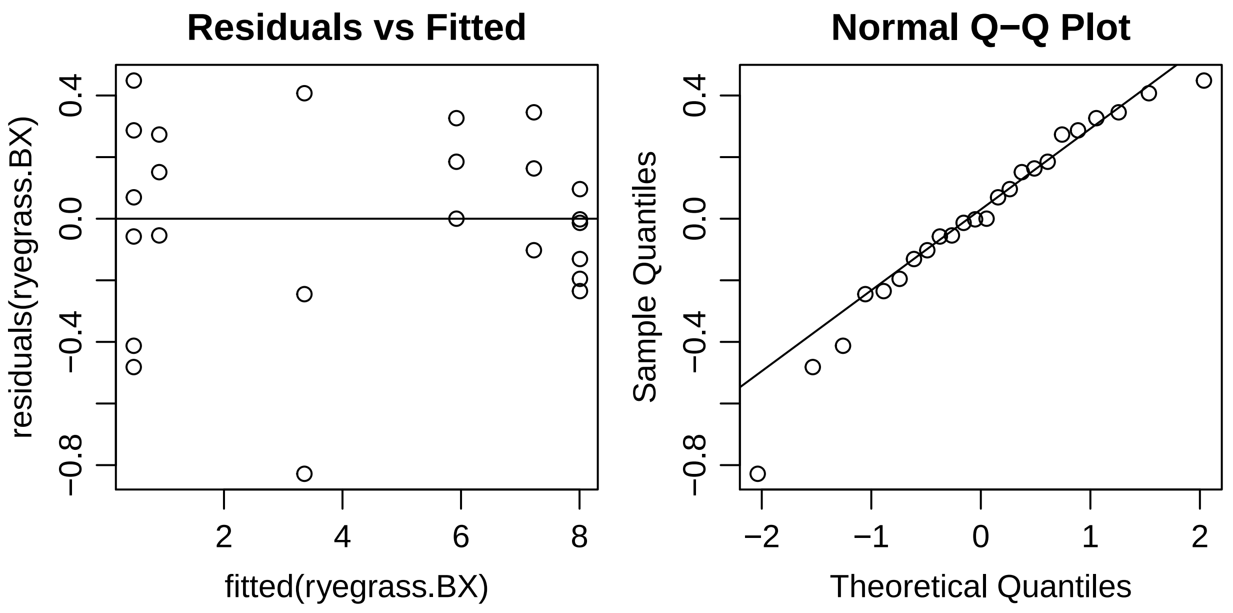


Figure 12.15: *Residual plots of BoxCox transformed-both-sides.The funnel type distribution of the residuals now has disappeared.*

these ages of detecting herbicide resistance/tolerance the dead or alive response is important. In insecticide research and in general toxicology the binomial response, dead or alive or affected not affected is fundamental for classifying toxic compounds.

A prerequisite is that we know how to properly classify the demise of a plant. We are far away from the normal distribution. The data, however, can still be analysed with the function `drm()` but a new argument must be included in the `drm()` function, viz `type="binomial`.

12.7.1 Example 1: Herbicide susceptibility

Below we have an experiment with putative glyphosate sensitive and tolerant biotypes of *Chenopodium album* and the survival as response, which takes two values: 0 means dead and 1 means alive. In fact, every plants is an experimental unit. We assume that all plants survived in the untreated control and at infinite high rates all plants die. It is important to note, though, that this assumption may not always hold; for example, some triazine-resistant biotypes may not be killed by atrazine, even at exceptionally high doses. But for this example, the data suggest that even the tolerant biotype can be killed at high doses.

```
library(drc)
read.csv("http://rstats4ag.org/data/chealglyph.csv")->cheal.dat
head(cheal.dat,n=3)
```

```
##   block biotype glyph.rate survival
## 1     A       S        426        0
## 2     A       T       1680        0
## 3     A       T          0        1
```

```
drm(survival ~ glyph.rate, biotype, fct=LL.2(names=c("Slope", "LD50")),
    type=("binomial"), data=cheal.dat, na.action=na.omit)->cheal.drm
```

```
par(mfrow = c(1, 1), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(cheal.drm, broken=TRUE, xlim=c(0,10000), bty="l",
     xlab="glyphosate rate", ylab="P(survival)")
```

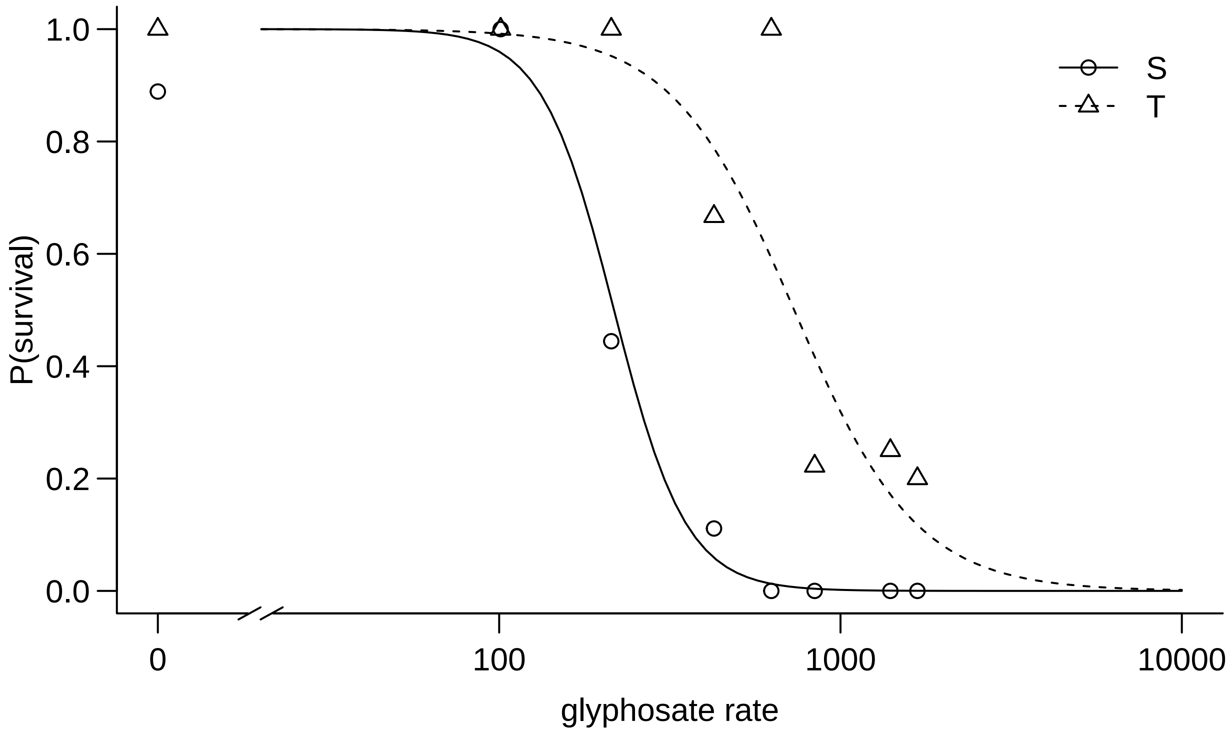


Figure 12.16: *Dose-response curves for survival of* Chenopodium album *classified* ***a priori*** *as either sensitive or tolerant to glyphosate based on field history.*

```
modelFit(cheal.drm)
```

```
## Goodness-of-fit test
##
##             Df Chisq value p value
##
## DRC model   89      63.247  0.9823
```

```
summary(cheal.drm)
```

```
##
## Model fitted: Log-logistic (ED50 as parameter) with lower limit at 0 and upper limit at 1 (2 parm
##
## Parameter estimates:
##
##           Estimate Std. Error t-value   p-value
## Slope:S    4.10418    1.40025  2.9310 0.0033783 **
## Slope:T    2.42592    0.72357  3.3527 0.0008003 ***
## LD50:S   216.98961   30.29692  7.1621 7.945e-13 ***
## LD50:T   731.56946  122.16134  5.9886 2.117e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The data from the tolerant biotypes are not as ‘clean’ as for the sensitive biotype, and this is reflected in the standard error of the LD_{50}. This variability is often to be expected when a population is still segregating for a resistance trait, and individual organisms are being used as experimental units. The ?EDcomp()‘ function can be used to test whether the LD_{50} is statistically different between biotypes.

```
EDcomp(cheal.drm,c(50,50), reverse=TRUE)
```

```
##
## Estimated ratios of effect doses
##
##             Estimate Std. Error   t-value   p-value
## T/S:50/50 3.3714492  0.7338531 3.2315040 0.0012314
```

The two biotypes exhibit an approximately 3-fold difference in susceptibility to glyphosate, and this difference is statistically different from 1.

Another important issue here, is that the response range goes between 0 to 1.0, which is in contrast to the use of continuous response variables such as biomass. Generally, you get more information using biomass or other continuous responses, but also more difficult choices for interpretation, as mentioned before about, e.g. how to deal with different upper limit and lower limits among dose response curves.

12.7.2 Example 2: Earthworm toxicity

The next example is from an ecotoxicological experiment with earthworms. A number of earthworms is distributed uniformly in a container where one half of the container surface is contaminated by a toxic substance (not disclosed). If the earthworms do not like the substance they will flee to the uncontaminated part of the container.

In this dataset the arrangement for data is somewhat different from the one with the sensitive and tolerant Chenopodium album. For each container we have a dose, a total number of earthworms, and the number of earthworms that migrated away form the contaminated part of the container. Consequently, we now need to use an additional argument in `drm()`, a weighing argument `weights=total`.

```
head(earthworms,n=3)
```

```
##   dose number total
## 1    0      3     5
## 2    0      3     5
## 3    0      3     4
```

```
## Fitting a logistic regression model
earthworms.m1 <- drm(number/total ~ dose, weights = total, data = earthworms,
                     fct = LL.2(), type = "binomial")
modelFit(earthworms.m1)  # a crude goodness-of-fit test
```

```
## Goodness-of-fit test
##
##              Df Chisq value p value
##
## DRC model    28      35.236  0.1631
```

```
summary(earthworms.m1)
```

```
##
## Model fitted: Log-logistic (ED50 as parameter) with lower limit at 0 and upper limit at 1 (2 parm
##
## Parameter estimates:
##
##                 Estimate Std. Error t-value   p-value
## b:(Intercept) 1.260321   0.246707  5.1086 3.246e-07 ***
## e:(Intercept) 0.145140   0.036797  3.9444 8.000e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
par(mfrow = c(1, 1), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(earthworms.m1, broken=TRUE, bty="l")
```

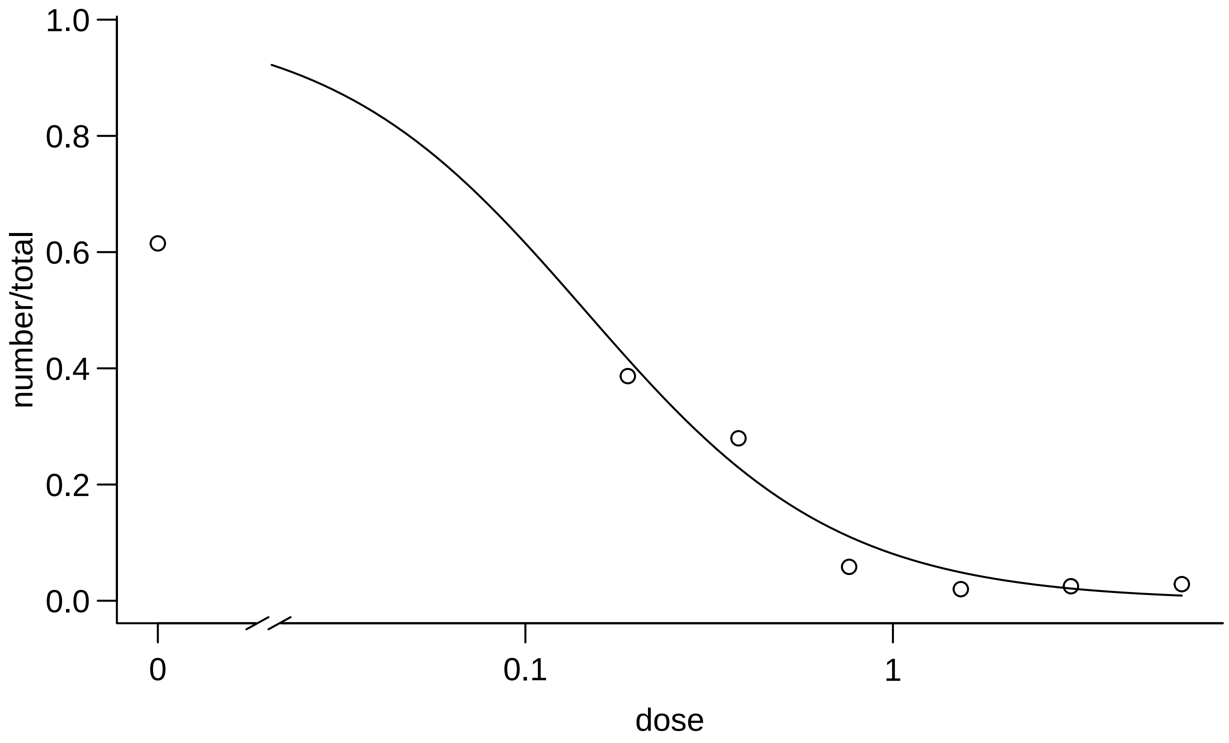


Figure 12.17: *Standard 2 parameter log-logistic dose-response regression with upper and lower limits set to 1 and 0, respectively.*

The regression above is a standard the with an upper limit of 1 and apparently the goodness of fit is not significant, even though the untreated control is around 0.6. From a biological point of view the earthworms can move freely after half of the container is contaminated with toxic materials. In the untreated control with no contamination we expect that around 0.5 would be present in the part where we count. Therefore, a 2-parameter model with the upper limit fixed at 1.0 does not seem appropriate here. Consequently, we will introduce a three parameter log-logistic model with an upper limit being estimated on the basis of data.

```r
## Fitting an extended logistic regression model
##  where the upper limit is estimated
earthworms.m2 <- drm(number/total ~ dose, weights = total, data = earthworms,
                     fct = LL.3(), type = "binomial")
modelFit(earthworms.m2)  # goodness-of-fit test
```

```
## Goodness-of-fit test
##
##              Df Chisq value p value
##
## DRC model   32       43.13  0.0905
```

```r
summary(earthworms.m2)
```

```
##
## Model fitted: Log-logistic (ED50 as parameter) with lower limit at 0 (3 parms)
##
## Parameter estimates:
##
##                Estimate Std. Error t-value   p-value
## b:(Intercept) 1.505679   0.338992  4.4416 8.928e-06 ***
## d:(Intercept) 0.604929   0.085800  7.0505 1.783e-12 ***
## e:(Intercept) 0.292428   0.083895  3.4856  0.000491 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
par(mfrow = c(1, 1), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(earthworms.m2, broken=TRUE, bty="l")
```

The goodness of fit was also here acceptable, so on the basis of the statistics we cannot pick the "best model". However, we could compare the AIC for the two models and pick the lowest value.

```r
AIC(earthworms.m1, earthworms.m2)
```

```
##                df       AIC
## earthworms.m1  2 699.10026
## earthworms.m2  3  78.31036
```

The model with the estimated upper limit has an AIC that is 9 times smaller than the 2-parameter model. This also corresponds with what we biologically expected.

The LD_{50} changed somewhat because the upper limit changed, not because there is much difference of the mid part of the curves as snow below.

```r
par(mfrow = c(1, 1), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(earthworms.m1, broken=TRUE,  col="red", lty=2, bty="l")
plot(earthworms.m2, broken=TRUE, add=TRUE)
legend("topright",c("2-parameter model (d = 1)", "3-parameter model (d = estimated)"),
       lty=2:1, col=c("red",1))
```

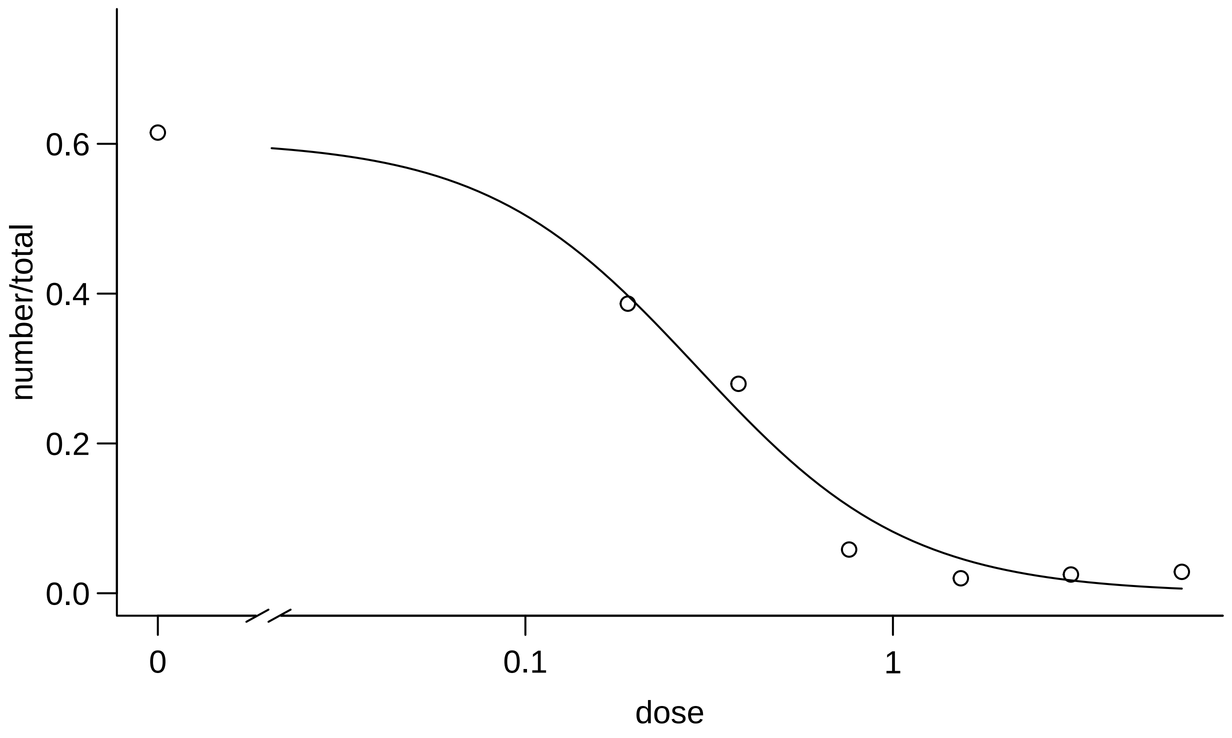


Figure 12.18: *Regression fit with the upper limit being estimated instead of fixed at 1.0.*

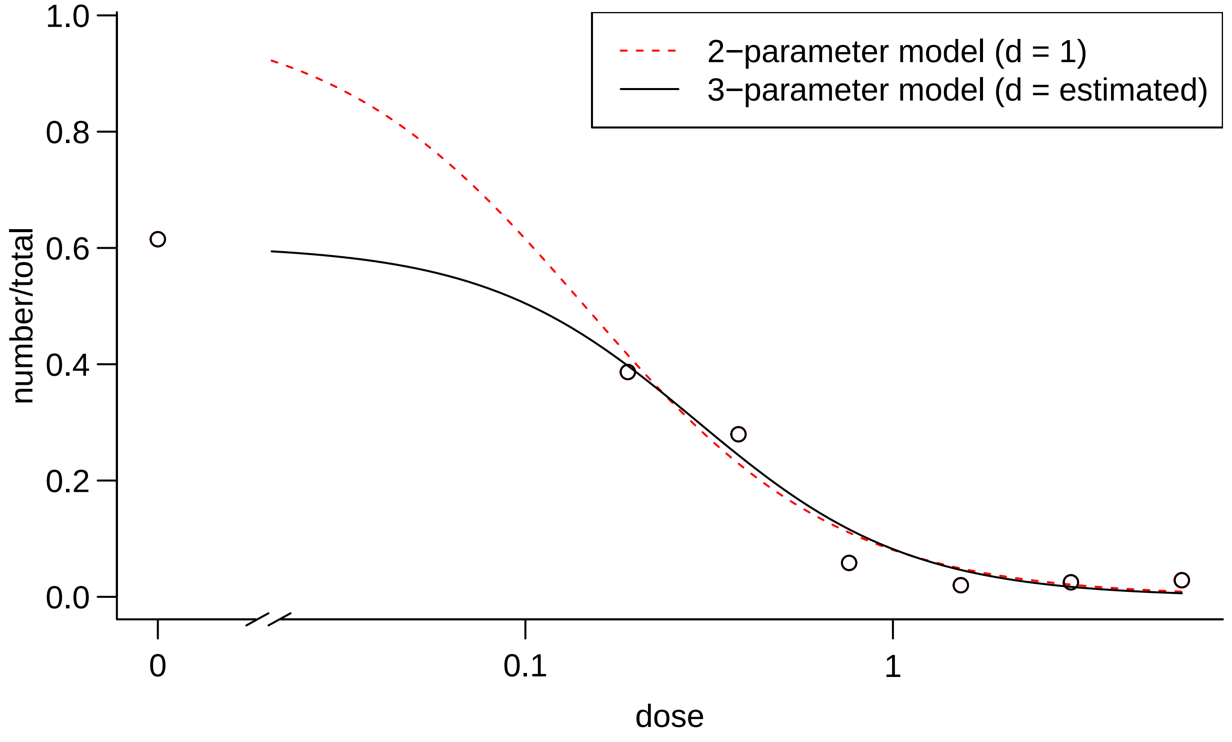


Figure 12.19: *Comparison between the two curves fittings on with standard* `d`*=1 and one with a* `d` *determined by data.*

The earthworm example is a good one, because the biological knowledge of the set up of the experiment is instrumental to understanding which model to use. In fact the LD_{50} ranged from 0.04 with the 2 parameter log-logistic fit to 0.29 while the 3 parameter log-logistic fit, which was the correct one from a biological point of view. We do expect that the earthworms would evenly distribute themselves in the untreated control with about 50% in each part of the box.

Chapter 13

Nonlinear Regression - Selectivity of Herbicides

Herbicides are unique in that they are designed to kill plants. Sufficiently high doses will kill both crops and weeds, while small doses have no effect on crops and weeds. For the selective herbicides there are dose-range windows that control some weeds without harming the crop too much. The action of a herbicide is usually determined by its chemical and physical properties, its effect on plant metabolism, the stage of development of the plant and the environment. The purpose of this chapter is to give an overview of the basic principles of how to quantitatively assess herbicide selectivity.

The chapter is based upon general toxicology commonly used in many disciplines of the biological sciences and it is also used to classify xenobiotics according to their toxic profile; the first of which is the dose required to kill 50% of some test animals (e.g., rats, mice, hamsters). However, the principles in the pesticide science have focus not only on general toxicity, but also on selectivity, particularly when it comes to herbicides.

13.1 Dose-Response Curves

The difference between tolerance and control of a plant is determined by the size of the dose. The term size of a dose, however, is rather vague in that for some herbicides, only few g/ha are needed to control weeds (e.g., many sulfonylureas) whereas for others we must apply several kg to obtain the same level of control (e.g., phenoxy acids). If we want to determine the potency or selectivity of a herbicide it is not enough only to look at one dose-response curve, as the proper assessment of selectivity should be stated in relative terms depending on the herbicide, crops and weeds in question. In order to avoid ambiguity in assessment of selectivity the sigmoid log-logistic dose-response curve is a good starting point.

$$y = c + \frac{d - c}{1 + \left(\frac{x}{ED_{50}}\right)^b}$$

Where `y` is the response, `d` denotes the upper limit, `c` the lower limit, ED_{50} denotes the dose, `x`, required to half the response 50% between `d` and `c`. Finally `b` denotes the relative slope of the curve around ED_{50}.

The ordinary dose-scale looks almost as an exponential decay curve apart form the upper part where there is a small bend. However, one of the nice properties of the log-logistic curves is that it is symmetric on the log(dose) scale (Figure 13.1). The inflection point is the ED_{50} whatever the dose scale. The broken lines in Figure 13.1 show the location of ED_{50}, and in equation 1 it is a natural parameter of the model. If zero-dose is used as control we must break the x-axis to indicate that the logarithm to zero does not exist.

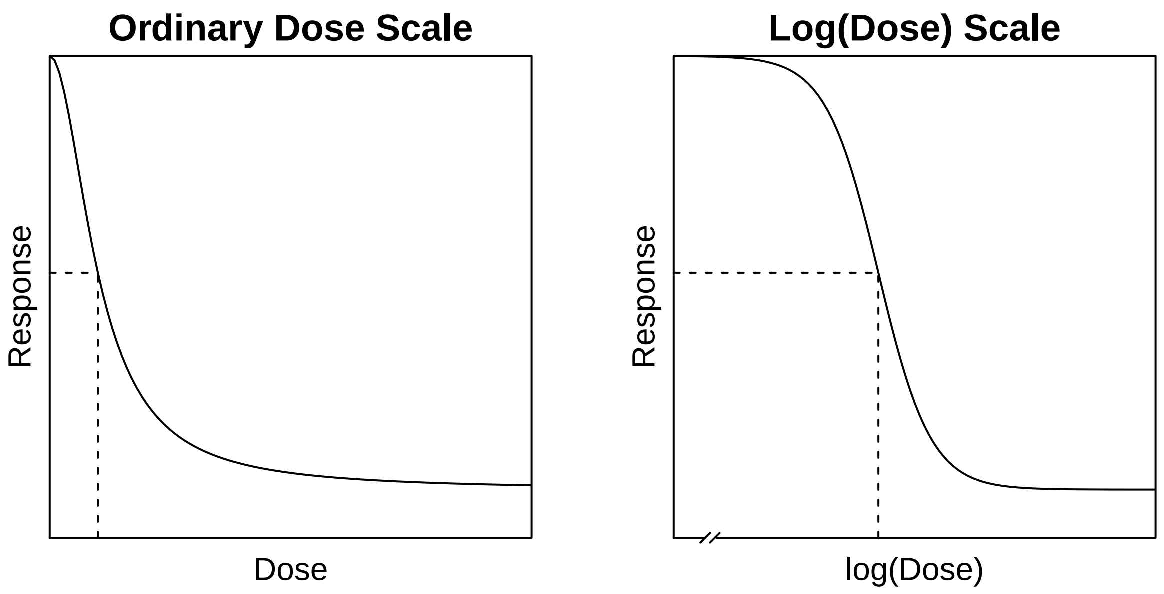


Figure 13.1: *The log-logistic curve plotted on ordinary dose axis and log(Dose) axis. The broken lines indicate the ED_{50} on the x-axis and on the y-axis. The broken line on the log(Dose axis) indicates that a zero-dose does not exist on a logarithmic axis.*

13.2 Assessment of Selectivity

If we want to compare the phytotoxicity of two or more herbicides on the same plant species or maybe the selectivity of a herbicide upon a crop and a weed species, then we must compare several dose-response curves simultaneously and quantify our findings in relative terms. We must define a standard herbicide and/or a standard species.

To make things simple we assume that we compare the action of two herbicides on the same plant species or at different plant species but with the same upper limit, `d`, and lower limit `c`.

The distance between the two dose-response curves at ED_{50} is a measure of the biological exchange rate between herbicides (Figure 13.2), analogous to the more common practice of exchanging currencies, when traveling to foreign land. In toxicology and herbicide selectivity, we do not call it exchange rate, but the relative potency or relative strength, and when studying herbicide resistant biotypes of weeds, the resistance factor or R:S ratio.

For example, we know that to get adequate control of a common weed flora we need x_B g of a test herbicide and x_A g of the standard herbicide per unit area. Some will argue that ED_{50} is not at all of interest when it comes to controlling weeds, since we are never interested in only achieving 50% control. While true from a practical standpoint, for the time being we will look at ED_{50} because it is a natural parameter of the log-logistic curve, and therefore it is an important parameter to classify toxic material. Later, we will look at the two additional ED-levels - ED_{10} to assess the crop tolerance and ED_{90} to assess the weed control. In general terms, we can define the relative potency as a response half way between the upper limit,`d`, and the lower limit `c`.

$$R = \frac{x_A}{x_B}$$

Where the relative potency, `R`, can be defined at any ED_x level of which ED_{10}, ED_{50}, and ED_{90} are the most common levels in Weed Science. In ecotoxicology ED_{05} is common to define an no-effect level.

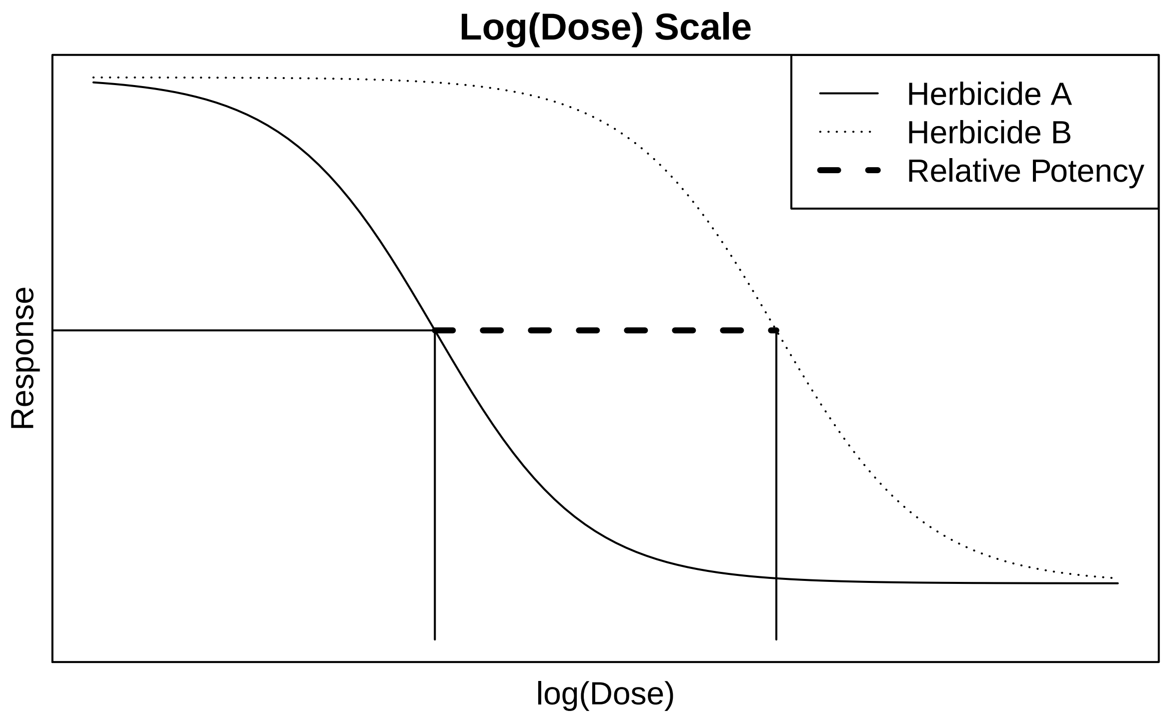


Figure 13.2: *The relative potency at ED_{50} is the distance between the two curves. Note that the doses are on log scale and therefore the distance shown at the graph is the ratio between the two doses. In this instance the two curves are similar, i.e. that they have the same upper and lower limit and slope. It is only the ED_{50} that differs.*

If the relative potency, R=1.0, then there is no difference between the effect of the two herbicides, i.e. $x_A = x_B$, if `R<1` the herbicide dose of B is less effective than herbicide A and if `R>1` the herbicide A is more effective than herbicide B. So every time we calculate the relative potency, then we must know if the sheer ratio is different from 1.0.

In Figure 13.2, the two dose response curves are similar in that the upper and lower limit are the same for the two curves as are the relative slopes, `b`. But what if the two curves are not similar (and this more commonly the rule rather than the exception in real life)?

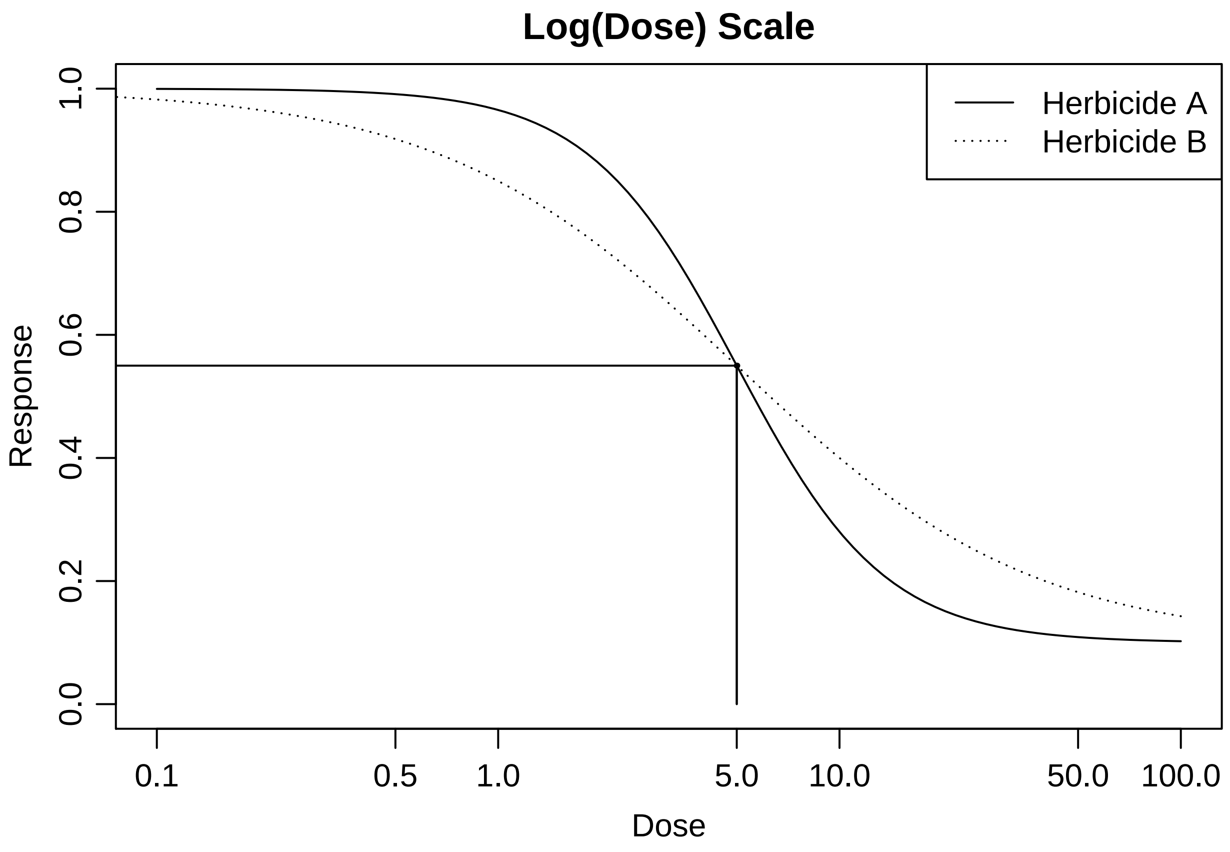


Figure 13.3: *The two herbicides have similar upper and lower limit and ED_{50}, but have different slopes.*

It is evident that in some cases ED_{50} does not give the right picture of toxicity when curves are not similar (Figure 13.3). Herbicide A is less potent above ED_{50} and more potent below ED_{50}. Consider the example in Figure 13.3 where the slopes are different but the ED_{50} s are identical. How should we report this? One way of doing it is to report the relative potency at relevant response levels. By defining Herbicide A as being the standard, it means that its ED_x is in the denominator. We get the following relative potencies:

```
##
## Estimated effective doses
##
##           Estimate Std. Error
## e:1:10 1.6667e+00 3.6316e-07
## e:1:50 5.0000e+00 5.1752e-07
## e:1:90 1.5000e+01 3.5880e-06
```

```
##
```

```
## Estimated effective doses
##
##          Estimate Std. Error
## e:1:10 5.5556e-01 1.7824e-07
## e:1:50 5.0000e+00 1.2011e-06
## e:1:90 4.5000e+01 2.6206e-05
```

```
##    e:1:10    e:1:50    e:1:90
## 2.9999968 1.0000014 0.3333346
```

- Herbicide A the doses are 1.67, 5, 15‘ for ED_{10}, ED_{50} and ED_{90}, respectively.
- Herbicide B the doses are 0.56, 5, 45 for ED_{10}, ED_{50} and ED_{90},respectively.
- And the relative potencies are 3, 1, 0.33 at ED_{10}, ED_{50} and ED_{90}, respectively when Herbicide A is the standard.

Note how the relative potency changes from 3.0 at ED_{10} to 0.33 at ED_{90}. Above ED_{50} herbicide B is more potent than herbicide A, but this changes below ED_{50} where the herbicide A now is most potent. This problem is looked at in more detailed in the previous chapter on dose response curves.

The ED_{50} somewhat resembles the LD_{50} (or dose required to cause 50% mortality in a test population) as an estimate in acute toxicity in toxicology, it does not tell the full story in Weed science. Particularly, when we wish to compare the selectivity of a herbicide on a crop and a weed flora, be it predominant weed species or not. ED_{50} contains much more information than do LD_{50}, but becomes more difficult to interpret because we typically deal with continuous responses in Weed Science.

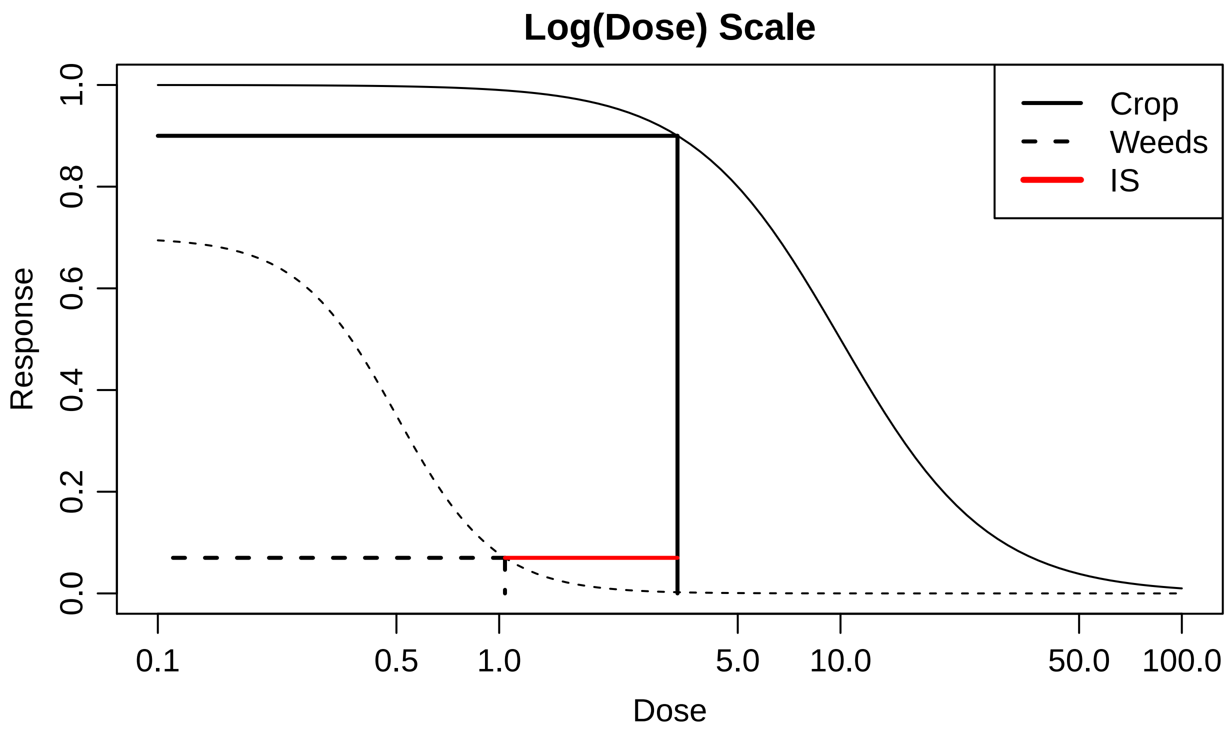


Figure 13.4: *The assessment of the Selectivity Index for a crop and a weed or a group of weeds.*

13.3 Selectivity index

Since herbicides may have an effect on any plant, be it crop or weeds, we sometimes have to accept a small decrease in the yield of the crop growing in a weed free environment. For example a 10% decrease might be tolerated, ED_{10}, while a 90% control of the weed, ED_{90}, is considered a reasonable control level of the weeds. The ED_{10} for the crop is 3.33 and the ED_{90} for the weed is 1.04. Consequently the Selectivity Index is 3.20. As seen in Figure 13.4, the upper limits for the crop and the weed are different.

In the development of herbicides the companies screen virtually 10,000's of compounds, only a small fraction of which may have some activity. In order to handle this immense amount of information, they must unambiguously define the selectivity of herbicides in the development phase. By accepting a 10% yield loss of the crop growing in weed free environment and by being satisfied with a 90% effect on the weeds growing in crop free environment, we get an Index of Selectivity, IS:

$$IS = \frac{[ED_{10}]_{crop}}{[ED_{90}]_{weeds}}$$

However, the same situation arises when we compare herbicide resistant biotypes with susceptible biotypes. Obviously, if there is target site resistance and a yield penalty we expect difference in response in untreated control. If it is metabolic resistance where no yield penalty has been observed, we can still have different untreated control response. This fact of differences in upper limit among curves are not being solved by making the untreated control equal to 100 and adjust the responses within biotypes accordingly. This often-ignored problem is dealt with in the previous chapter on dose-response curves.

13.4 Vertical and Horizontal Assessment

Vertical and horizontal assessment of herbicide effect is shown graphically in Figure 13.5.

13.4.1 Vertical assessment

Vertical assessment compares plant response at some preset dose levels. This is the most common method for evaluating herbicides in the field. If doses were chosen close to the upper or lower limit of the curves, differences between treatments would be less than if they were chosen in the middle part of the curve. If we are only working at dose-ranges in the middle part of the curves then, in this particular instance, differences would be almost independent of dose-levels. Consequently, the middle region is obviously the optimal part of the curve to obtain information about differences of effects in this particular graph where the curves are looking very similar. This, however, does not apply when the curves have different slopes as shown in Figure 13.4.

The curves show that if a herbicide is tested with or without an adjuvant in a factorial experiment, we may get significant interactions, because the differences of effects are not constant. As this interaction is dose dependent due to the S-shaped curves, it may be considered trivial and of little biological significance, when we appraise the selective action of the herbicide. If we choose only dose-ranges in the middle part of the curves, then differences are constant and independent of dose-levels and the interaction would disappear, i.e. the effects are additive.

13.4.2 Horizontal assessment

Horizontal assessment is the method we used in the herbicide selectivity section and it answers the question many farmers are interested in: which dose should I use of herbicide A to get the same effect as with herbicide B (the biological exchange rate of two herbicide products). Usually, herbicides are tested in two to three doses and their efficacy is compared with either the untreated control or with some standard herbicide treatment. Rarely, we have so many doses in the field that we actually are able to fit the entire dose response curve.

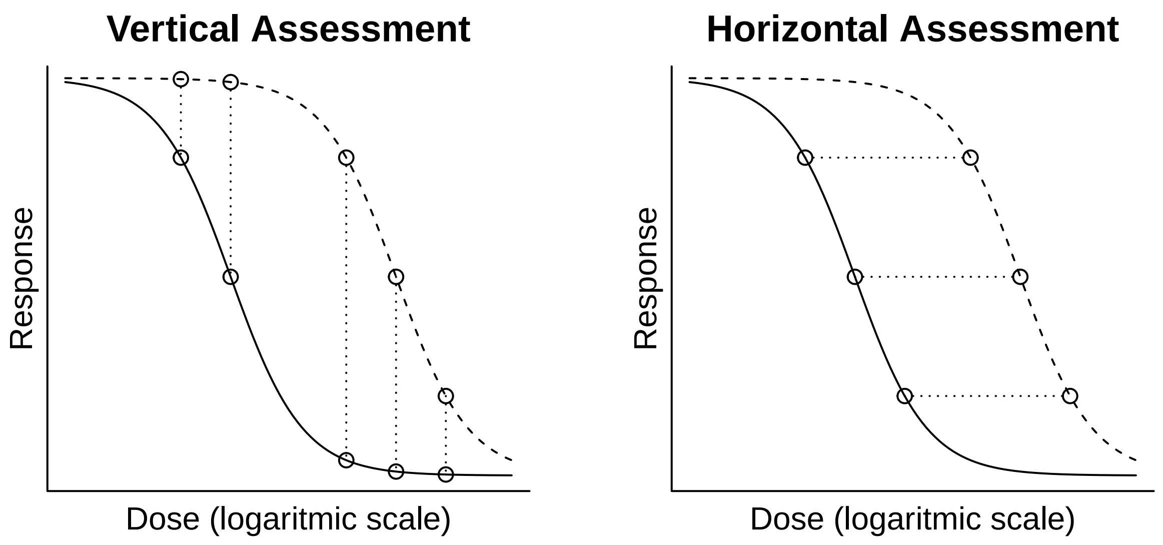


Figure 13.5: *Vertical and horizontal comparison of efficacy of say two herbicide on the same plant species.*

Even though the horizontal assessment above seems straight forward, it is not always the case. With continuous response variables comparing dose-response curves with different upper and lower limits is not just a trivial task to compare curves. The differences in upper and lower limits of different dose-response curves are sometimes important as they tell a story about the experiments. In the Dose-Response Curve chapter we briefly discuss it. The bottom line here is that comparison is not a matter of statistics, but a matter of biology, and thus rather complicated.

Chapter 14

Plant Competition Experiments

Competition experiments are a staple of weed science. Typically, we often want to assess the effect of weed density or duration of competition on crop yield. There is an ongoing debate about the appropriateness of using density and not for example plant cover. We will not go into this debate, but stick to density of plant, because the methods of analyzing data remain the same whether the independent variable, `x`, is density or plant cover.

One of the important questions is,how do we assess competition and when does it start. In principle competition starts at germination and is a question of the resources:

- Light
- Nutrients
- Water
- "Space"

The three first factors are rather easy to quantify, but the fourth one is more intangible it is dependent on the growth habits of the weeds, e.g. prostrate, erect and the ability of the crop to outgrow the weeds. Whatever the reason for competition, it often boils down to the relationship in Figure 14.1; when will the relationship divert from a straight line.

14.1 Yield, Density and Yield Loss

The general plant competition and crop yield loss relationships are consider the same, a rectangular hyperbola. In Figure 14.1 we have a classical intra-specific competition relationship (A), and a yield loss relationship (B). Competition materializes when the curve diverts from the straight line. The initial straight line means that putting a new plant into the system just increases the yield the same way as all the other individuals contribute initially. When individual plants begin compete with each other for resources, because of high density, then the curves diverts from the straight line.

In Figure 14.1B, the percentage yield loss is based upon the yield without the presence of weeds. The competition (inter-specific competition) for resources materializes itself immediately. When the line diverts from the straight line relationship there will also be some intra-specific competition among the weeds. If there is no competition between crop and weeds at all, then the slope of the curve in Figure 14.1B would be zero, or no change in yield whatever the density of weeds.

If the yield is a crop and the density is weeds per unit area then the the competition (inter-specific competition) materializes in exactly the same way. But now the competition begins from the very start. If there is no competition between crop and weed then the slope of the curve would be zero, viz no change in yield whatever the density of weeds.

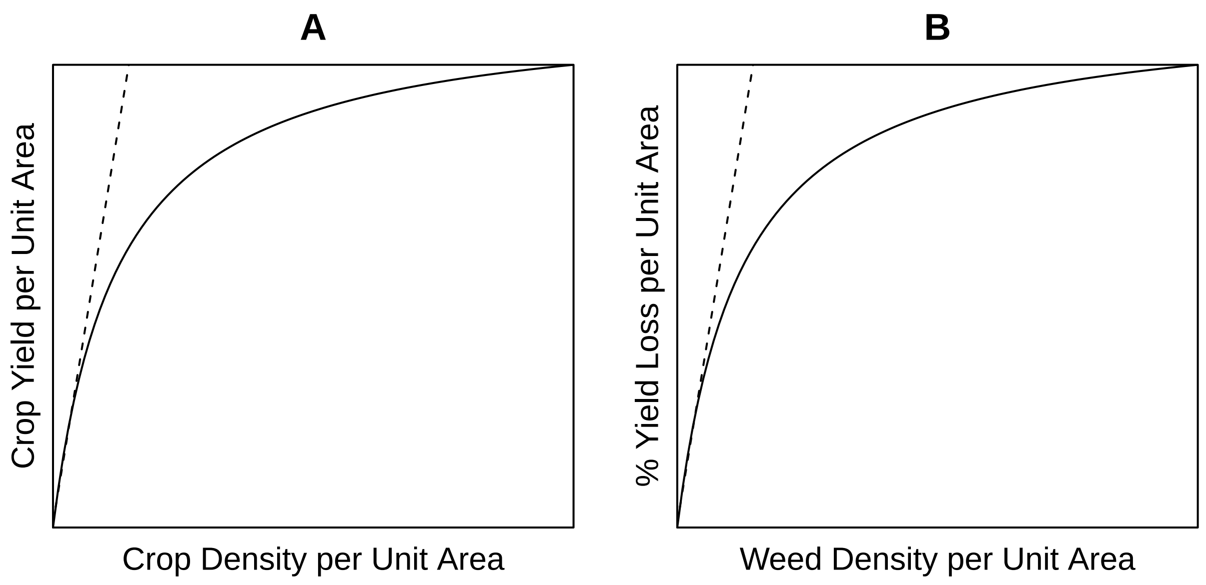


Figure 14.1: *Competition within the same species, often denoted intra-specific competition (A). Yield loss function based on the percentage yield loss relative to the yield in weed free environment (B).*

The first example is a study conducted near Lingle, Wyoming over two years. In this study, volunteer corn densities ranging from 0 to 2.4 plants/m^2 were planted along with dry edible beans to document the bean yield loss from increasing volunteer corn density.

```
read.csv("http://rstats4ag.org/data/volcorn.csv")->VolCorn
VolCorn$yr<-as.factor(VolCorn$yr)
VolCorn$reps<-as.factor(VolCorn$reps)

head(VolCorn,n=3)
```

```
##     yr reps dens    y.pct y.kg
## 1 2009    1    0 3.283996 6273
## 2 2009    2    0 9.466543 5872
## 3 2009    3    0 4.625347 6186
```

The variable 'yr' is the year the study was completed (either 2008 or 2009), `reps` denotes the replicate (1 through 4), 'dens' is the volunteer corn density in plants/m^2 (0 to 2.4), 'y.pct' is the percentage dry bean yield loss as compared with the zero volunteer corn density, and 'y.kg' is the dry bean yield in kg/ha.

Cousens 1985 proposed a re-parameterization of a rectangular hyperbola (perhaps better known as Michaelis-Menten) model as a tool to analyze competition experiments, and the `drc` is well suited for this type of analysis Ritz and Streibig 2005. This package must be loaded with the code:

```
library(drc)
```

The `drm()` function can be used to fit a variety of non-linear models, including the Michaelis-Menten model. The code to fit the Michaelis-Menten model to the volunteer corn data is for one of the two years, 2009, by using the argument `data=VolCorn, subset=yr==2009,`

```r
drm(y.pct ~ dens, fct=MM.2(names=c("Vmax","K")), data=VolCorn,
    subset=yr==2009, na.action=na.omit)->y2009.MM2
summary(y2009.MM2)
```

```
##
## Model fitted: Michaelis-Menten (2 parms)
##
## Parameter estimates:
##
##                    Estimate Std. Error t-value p-value
## Vmax:(Intercept)    66.7415    73.8753  0.9034  0.3782
## K:(Intercept)        2.6490     4.8209  0.5495  0.5894
##
## Residual standard error:
##
##  13.32239 (18 degrees of freedom)
```

Obviously, the `Vmax` and `K` parameter of the Michaelis-Menten model were non-significant, the reason is that the range of density of weeds were not large enough, we only catch the linear part (Figure 14.2).

```r
par(mfrow = c(1, 1), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(y2009.MM2,log="",
     ylab= "Percent Sugarbeet Yield Loss",
     xlab="Volunteer Corn Density",
     ylim=c(0,70), xlim=c(0,10), bty="l")
abline(h=66.7, lty=2)
```

All functions in the `drc` package are defined in the `getMeanFunctions()` by writing `?MM.3` or `?MM.2` you can see the help on the curve fitting function.

$$y = c + \frac{d - c}{1 + (e/x)}.$$

The suffix 3 or 2 defines how many asymptotes we use. The most common one is `MM.2` where there is only one upper limit d, in this context often referred to as *Vmax*. If you compare the model above with the log-logistic models, used in the selectivity and dose-response chapter, they look almost identical except that b in the log-logistic does not exist in `MM.2` or `MM.3`, because for this case b=1. The standard error of the parameter estimates reveals that none of the two parameters are significantly different from zero. This is due to the fact that only the first part of the curve is supported by experimental data as seen in Figure 14.2; there is no data to support the upper limit of the curve.

The tradition in weed science, as mentioned above, is to reparametrizise the Michaelis-Menten model and use:

$$Y_L = \frac{Ix}{1 + Ix/A}.$$

which was proposed by Cousens (1985), where `A` now is the upper limit and `I` is the initial slope of the curve as shown Figure 14.1. This reparametrization is available in the 'drc' package by using the 'yieldLoss()' function as shown below:

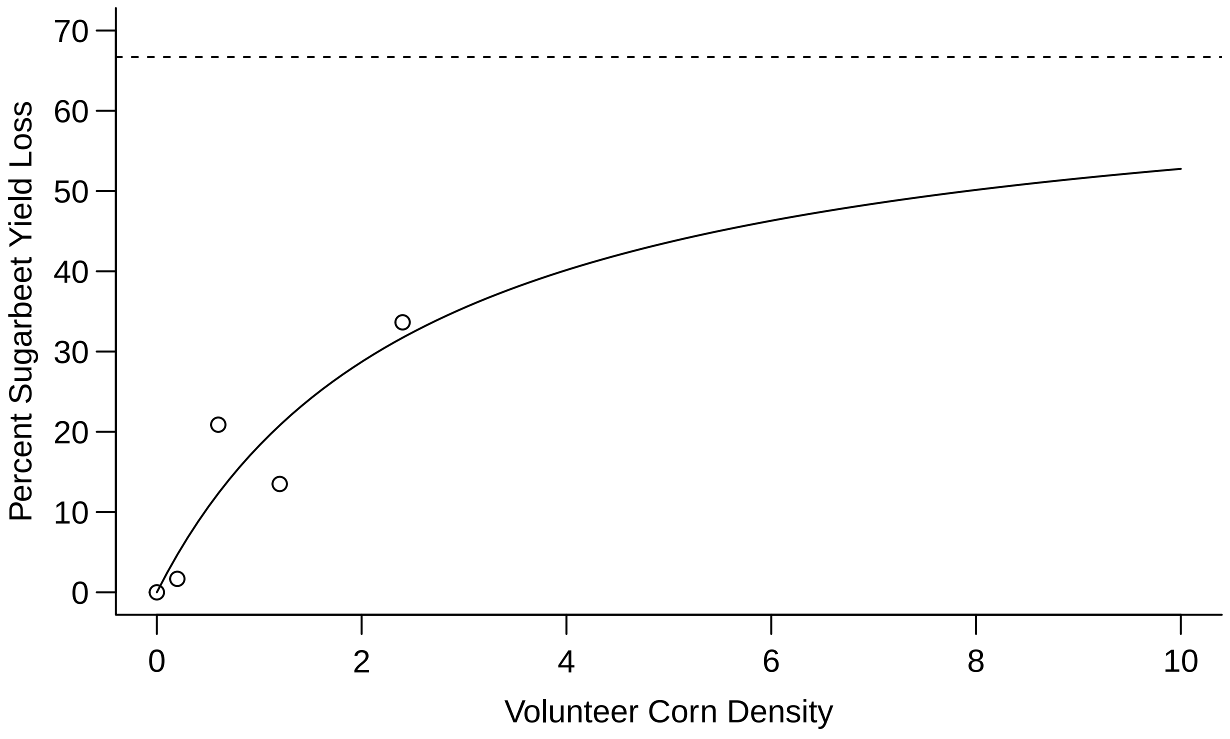


Figure 14.2: *Yield loss curve with a two parameter Michaelis-Menten's curve (the argument in `drm()` is `fct=MM.2()`. The `Vmax` is the upper limit, shown with the broken horizontal line, and `k` is the rate constant.*

```
yieldLoss(y2009.MM2, interval="as")
```

```
##
## Estimated A parameters
##
##   Estimate Std. Error   Lower   Upper
## 1   66.741     73.875 -88.465 221.948
##
##
## Estimated I parameters
##
##   Estimate Std. Error   Lower   Upper
## 1   25.195     18.749 -14.196  64.585
```

The upper limit, which is called `Vmax` in the Michaelis-Menten and `A` in the yieldLoss function is the same 67% and the rate constant in the Michaelis-Menten is 2.64 (corresponding to ED50 in the Log-logistic), but for the Cousens rectangular hyperbola the initial slope is 25. Of course the parameters of the `yieldLoss()` function were not different from zero either.

Looking at the yield loss curve one could suspect that a straight line relationship applies. We test the regression against the most general model an ANOVA:

```
lm.Y2009<-lm(y.pct ~ dens,data=VolCorn, subset=yr==2009)
ANOVA.Y2009<-lm(y.pct ~ factor(dens), data=VolCorn, subset=yr==2009)
anova(lm.Y2009,ANOVA.Y2009)
```

```
## Analysis of Variance Table
##
## Model 1: y.pct ~ dens
## Model 2: y.pct ~ factor(dens)
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     18 3241.9
## 2     15 2637.2  3    604.71 1.1465 0.3625
```

The test for lack of fit is non-significant (p=0.36) so we can confidently assume the straight line gives a good description of the relationship.

```r
summary(lm.Y2009)
```

```
##
## Call:
## lm(formula = y.pct ~ dens, data = VolCorn, subset = yr == 2009)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -30.013  -6.501   1.457   7.370  27.645
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.498      4.285   0.583  0.56717
## dens          13.002      3.475   3.741  0.00149 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.42 on 18 degrees of freedom
## Multiple R-squared:  0.4374, Adjusted R-squared:  0.4062
## F-statistic:    14 on 1 and 18 DF,  p-value: 0.001495
```

Sugarbeet yield loss increases by 13% with each volunteer corn plant/m^2 that is added into the system.

```r
library(dplyr)
Yield.m<-subset(VolCorn,yr==2009) %>%
  group_by(dens) %>%
  summarise(Yieldloss=mean(y.pct))

par(mfrow = c(1, 1), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(Yieldloss ~ dens,data=Yield.m,
     ylab="Percent Sugarbeet Yield Loss",
     xlab="Volunteer corn density", bty="l")
abline(lm.Y2009)
plot(y2009.MM2,log="", add=T, lty=2)
```

The assumption of a straight line relationship in Figure 14.3 is justified by the test for lack of fit and we can conclude we loose 13% yield per each volunteer corn plant. However, it is important to recognize that the further we extrapolate beyond the volunteer corn densities used in the study, the more likely the linear fit is to provide nonsensical yield loss estimates.

One issue with expressing yield loss as a function of the weed-free is it relies heavily on the weed-free control treatments from a particular study. When regression is used, it is possible to use the density relationship (in

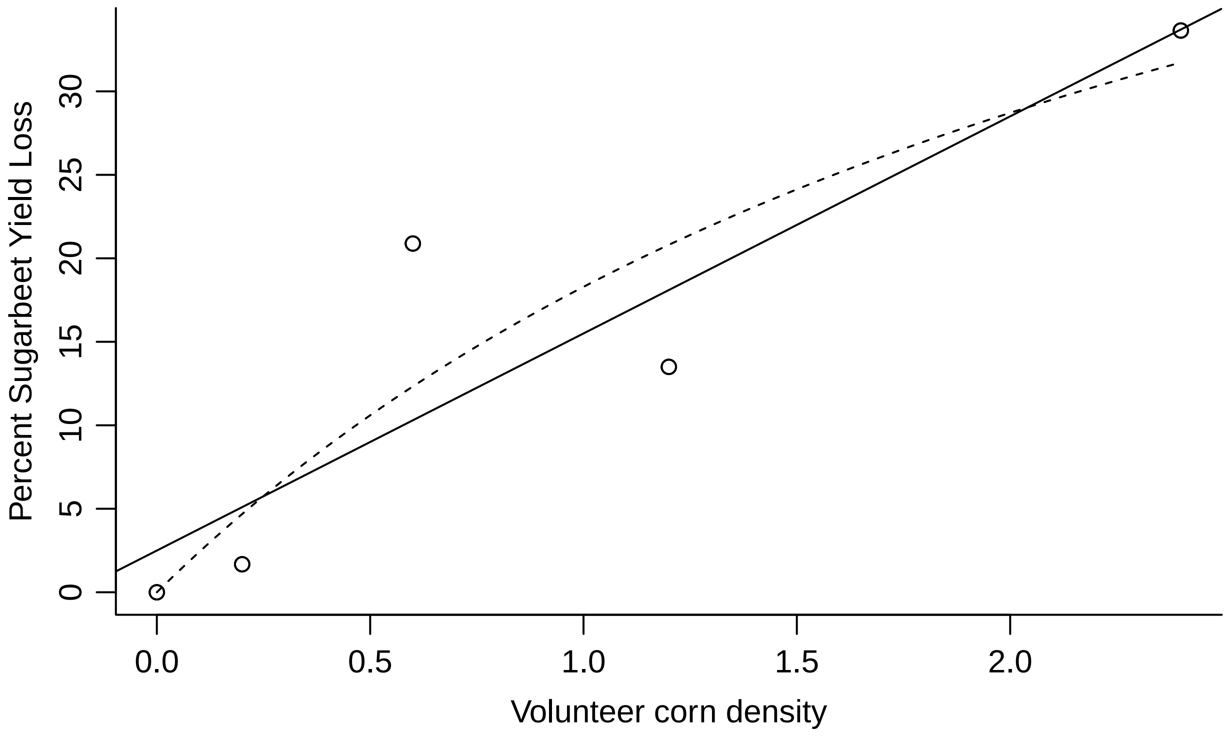


Figure 14.3: *The linear model fit with a slope of 13 meaning that for every unit density of corn we get a yield decrease of 13%. The broken line is the nonlinear fit from shown in Figure 14.2.*

addition to the weed-free control treatment) to provide a more robust estimate of crop yield in the absence of weed competition. The weed-free yield can be estimated by using the following re-parameterization of the rectangular hyperbolic model (also proposed by Cousens 1985):

$$Yield = Y_0(1 - (\frac{Ix}{100(1 + Ix/A)}))$$

where Y_0 is the intercept with the yield axis when weed density is zero. Of course it requires that the curve is well described over the range of relevant weed densities Ritz, Kniss and Streibig 2015.

14.2 Replacement Series

The replacement series can assess interference, niche differentiation, resource utilization, and productivity in simple mixtures of two species. Of course more than two species can be used as long as the total density remains the same, but the interpretation of results becomes very difficult. The species are growing at the same total density, but the proportion between the two species vary. One of the good things about replacement series is that if the replacement graphs looks like the one in Figure 5, it could be the reference, because with linear relationships in Figure 5 shows no competition; the two species do not interfere with each others growth. As discuss earlier, when there is a straight line relationship between yield and density of a species (Figure 1), the second species does not interfere. If there is a curved relationship there is intraspecific and/or inter specific competition. The intraspecific competition can only be assessed if a species is grown in pure stand.

The density dependence, maximum density determined by experimenter, impedes generalization for a replacement series. And as always, we only get a snapshot of what is going on in an otherwise dynamic competition scenario. In order to avoid criticisms, however, researchers should appreciate the assumptions and limitations of this methodology Jollife 2000

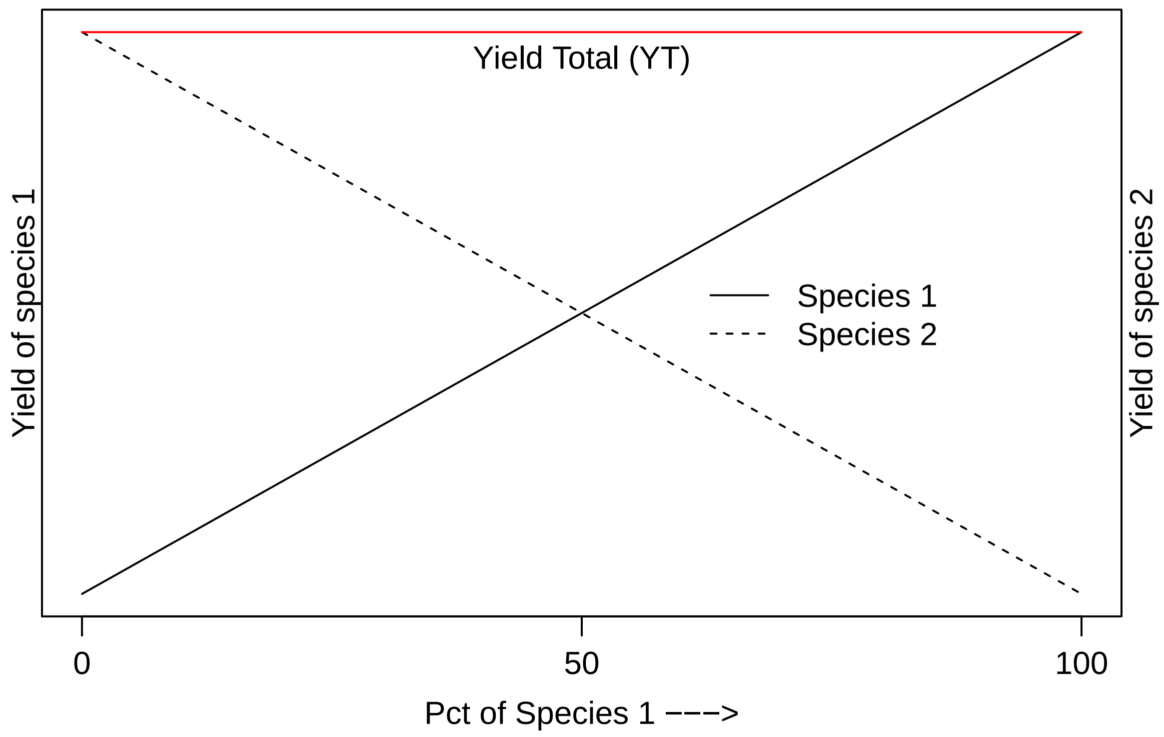


Figure 14.4: *The frame of reference for replacement series with two species where there is no competition between species and the Yield Total (YT) does not change which ever the combination of the two species is. The two species do not need to have the same maximum yield in monoculture.*

The philosophy of the replacement series is that the carrying capacity, in terms of say biomass, on a unit of land is constant whatever the proportion of the species.

The example uses a barley crop grown together with the weed Amsinckia menziesii. The experiment was run in greenhouse with the intention of having 20 plants in total in pots of 20 cm in diameter. After 20 days the plants were harvested and the actual number of plants were counted and the biomass per species measured.

The dataset `Replacement series.csv` is a mixture of `csv` and `csv2` files, because the students who did the experiments came form continental Europe or Australia. It means the continental Europeans use the semicolon as variable separator mixed with the Australian's decimal separator of dot. This mess is taken care of by using the `read.table()` function.

As we have to use the percent of either species as independent variable to fit regression models we have to define new variables `Pct.Amsinckia` and `Pct.Barley`.

```
Replace.1<-read.table("http://rstats4ag.org/data/ReplacementSeries.csv",
                      header=TRUE, sep=";", dec=".")
Replace.1$Pct.Amsinckia<-with(Replace.1,D.Amsinckia*100/(D.Amsinckia+D.Barley))
Replace.1$Pct.Barley<-   with(Replace.1,D.Barley*100/(D.Amsinckia+D.Barley))
head(Replace.1,n=3)
```

```
##   D.Amsinckia D.Barley Harvest.time B.Amsinckia B.Barley Pct.Amsinckia Pct.Barley
## 1           0       14   14.05.2014           0     98.1             0        100
## 2           0       32   14.05.2014           0    112.7             0        100
## 3           0       21    14.05.2014            0      94.5              0         100
```

First we try straight line relationships and illustrate the fit and with an analysis of residuals.

```
lm.Amsinckia<-lm(B.Amsinckia ~Pct.Amsinckia,data=Replace.1)
lm.Barley<-lm(B.Barley ~Pct.Barley,data=Replace.1)

par(mfrow = c(2, 2), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,.0))
plot(B.Amsinckia ~Pct.Amsinckia,data=Replace.1, ylab="Yield of Amsinckia",
     xlab="Percent of Amsinckia")
  legend("topleft", "A", bty="n")
abline(lm.Amsinckia)
plot(B.Barley ~Pct.Barley,data=Replace.1,ylab="Yield of Barley",
     xlab="Percent of Barley",pch=16)
  legend("topleft", "B", bty="n")
abline(lm.Barley,lty=2)
plot(lm.Amsinckia,which=1, caption=NA)
  legend("topleft", "C", bty="n")
plot(lm.Barley,which=1, caption=NA)
  legend("topleft", "D", bty="n")
```

Obviously, the relationships in Figure 14.5 for both species look like a curved relationship. We are not sure of which relationship to use and resort to a second degree polynomial.

$$Yield = a + bx + cx^2.$$

where a is the intercept with the y-axis and b and c are parameters for the x and the x^2. The biological meanings of polynomial parameters in general are not often of interest because they can be hard to interpret. To fit the polynomial we use the `lm()`function because it is essentially a linear model we are fitting by adding a parameter for the x^2 by writing `I(x^2)`. However, we cannot use the `abline()` function since there is now more than one 'slope' parameter.

To plot a line, first we generate predicted values using the polynomial model to get smooth fits. It is done with the `predict()`function `predict(Pol.B.Amsinckia,data.frame(Pct.Amsinckia=seq(0,100,by=1)))` where

- `Pol.B.Amsinckia` the results of the second degree polynomial fit for Amsinckia
- `data.frame(Pct.Amsinckia=seq(0,100,by=1))` generates a sequence of 'x' values from 0 to 100 by 1 (for a total of 101 values).

The same procedure applies to barley.

```
Pol.B.Amsinckia<-lm(B.Amsinckia ~Pct.Amsinckia+I(Pct.Amsinckia^2),data=Replace.1)
line.Pol.B.Amsinckia<-predict(Pol.B.Amsinckia,data.frame(Pct.Amsinckia=seq(0,100,by=1)))

Pol.B.Barley<-lm(B.Barley ~Pct.Barley+I(Pct.Barley^2),data=Replace.1)
line.Pol.B.Barley<-predict(Pol.B.Barley,data.frame(Pct.Barley=seq(0,100,by=1)))

par(mfrow = c(1, 2), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,.0))
plot(B.Amsinckia ~Pct.Amsinckia,data=Replace.1,ylim=c(0,100), bty="l",
     ylab="Amsinckia yield", xlab="Percent Amsinckia")
lines(line.Pol.B.Amsinckia~seq(0,100,by=1),col="red")
plot(B.Barley ~Pct.Barley,data=Replace.1,pch=16,ylim=c(0,100), bty="l",
```

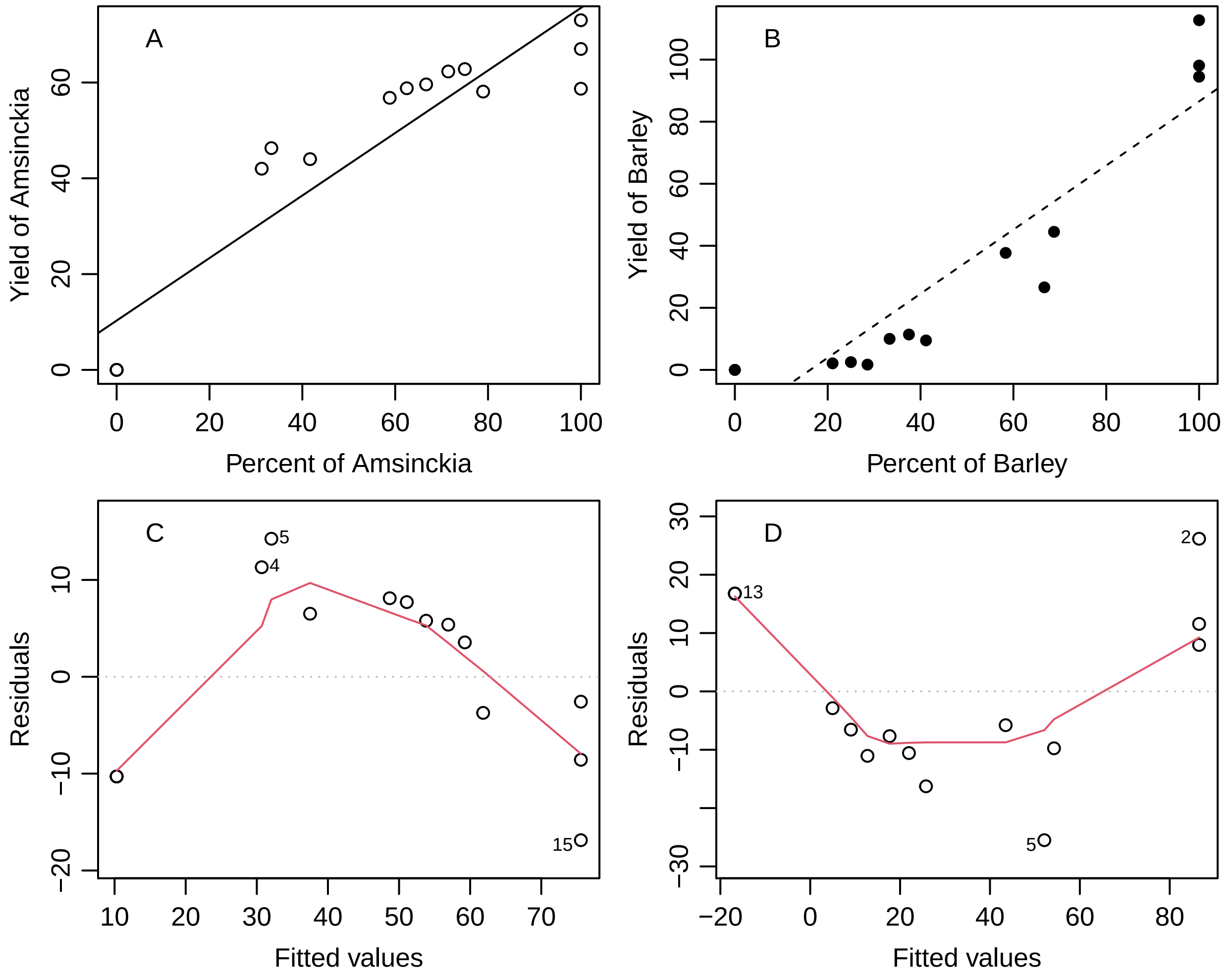


Figure 14.5: *Straight line relationships do not appear to capture the variation in the species. The analysis of residuals shows systematic departure form the expected shotgun distribution of residuals.*

```
     ylab="Barley yield", xlab="Percent Barley")
lines(line.Pol.B.Barley~seq(0,100,by=1),col="red",lty=2)
```

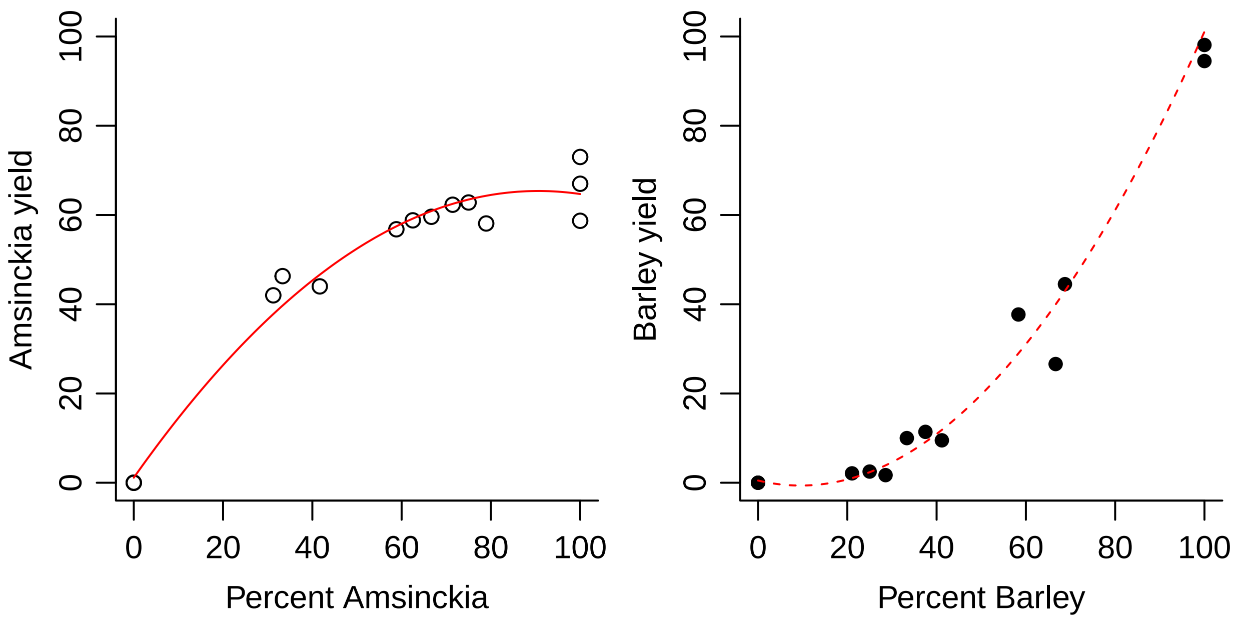


Figure 14.6: *Fitting second degree polynomials to data.*

Overall the second degree polynomials describe the variation reasonably well (Figure 14.6). However, there is a catch to it, the minimum for the Amsinckia second degree polynomial is at low percent of Amsinckia and the maximum for the second degree polynomial for barley is at rather high proportion of barley. If we can live with that we can use the fit to summarize the experiment by calculation the Yield Total (YT) as shown in the graph in Figure 14.4.

This is a good example of the problem with polynomials. A Second degree polynomial is symmetric with either a minimum or a maximum depending of the parameters. A third degree polynomial does not have this symmetric property, it becomes much more complex, and it is definitely not advisable to use with interpolations. In order to summarize the experiment on the basis of the fits, we can combine the two curves for the two species and calculate the YT.

We want to use the `line.Pol.B.Amsinckia` and the `line.Pol.B.Barley` that define the smooth lines and calculate YT. In order to calculate the YT we must sum the two predicted yields, but we must be carevul to reverse the order of one of the species (in this case we'll reverse using the function `rev(line.Pol.B.Barley)`. For this example, the maximum yield is 102, which occurs when the percentage of *Amsinckia* is 0% (found by using the `which.max()` function).

```
max(line.Pol.B.Amsinckia + rev(line.Pol.B.Barley))
```

```
## [1] 102.2132
```

```
which.max(line.Pol.B.Amsinckia + rev(line.Pol.B.Barley))-1
```

```
## 1
## 0
```

```
par(mfrow = c(1, 1), mar=c(5.2,3.2,0.5,.5), mgp=c(2,.7,.0))
plot(c(line.Pol.B.Amsinckia[101],line.Pol.B.Barley[101])~c(100,0),type="l",
     ylim=c(0,120), xlab="", ylab="Yield", bty="l")
mtext("Amsinckia (%) --->",side=1,line=2)
mtext("<--- Barley (100-%)",side=1,line=3)
Pred.RYT<-line.Pol.B.Amsinckia[1:101]+line.Pol.B.Barley[seq(101,1,by=-1)]
lines(Pred.RYT~seq(0,100,by=1),lty=3,lwd=2)
lines(line.Pol.B.Barley~seq(100,0,by=-1),col="blue",lty=2)
lines(line.Pol.B.Amsinckia~seq(0,100,by=1),col="red",lty=1)
points(B.Amsinckia ~Pct.Amsinckia,data=Replace.1,pch=19,ylim=c(0,100), col="red")
points(B.Barley ~Pct.Amsinckia,data=Replace.1,pch=1,ylim=c(0,100), col="blue")
```

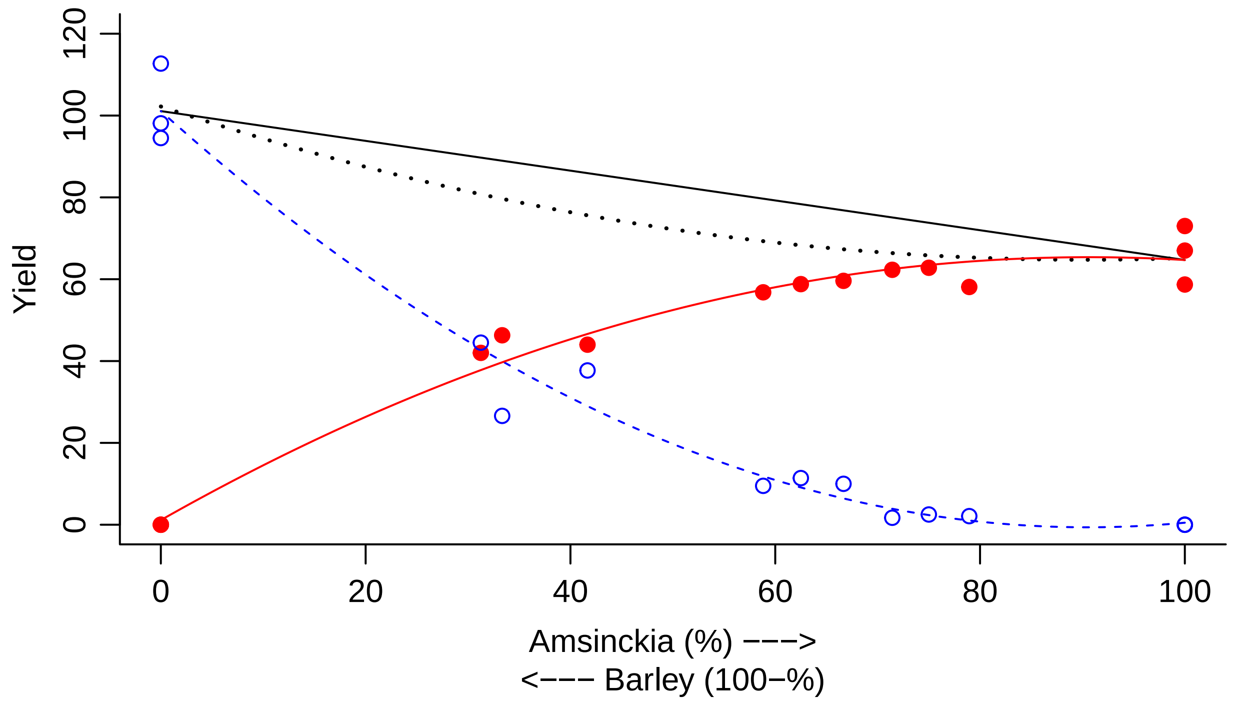


Figure 14.7: *Summary of the replacement series experiment with barley and* Amsinckia. *Solid and dotted black lines represent theoretical linear and estimated actual YT, respectively; red solid line and filled circles represent Amsinckia yield; blue dashed line and open circles represent barley yield.*

It seems in Figure 14.7 that if Amsinckia gains with its convex curve, and barley suffers with its concave curve, but gain and suffering still leave the YT below the theoretical value of YT.

Another issue is that that we do not test the regressions statistically, but use the fit to illustrate the relationships. In order to apply test statistics it requires more systematic designs with fixed number of plants per unit area, which unfortunately was not the case here. In the first example we had genuine replication with several replicates of the number of volunteer corn per unit area and therefore we could test which model could be used.

It also shows that when fitting curves we can derive predicted values, which can be used to calculate derived parameters such as YT. In replacement series analysis, one often scales the results so that the theoretical maximum yield is equal to 1.0 and then calculate the Relative yield Total (RYT), which in mixed cropping research is called Land Equivalent Ratio (LER).

Chapter 15

Seed Germination

Most weeds multiply and spread by seeds and understanding how seeds germinate, and how ambient factors affect seed dormancy, germination, vigor and early growth is instrumental for our continuous struggle to deplete soils of weed seeds. On agricultural land in Denmark we find about 30,000 viable weed seeds/m^2. It has dropped to half of what we found 45 years ago. This drop is due to many factors, but probably the most important one is the continuous use of herbicides.

The weed's ability to set seed and the seeds' ability to germinate is therefore important for us to try understand and counteract seed soil bank enrichment with agronomic means. Therefore, seed testing experiments to assess germination ability and vigor should support future action of depleting weed pressure.

The International Seed Testing Association ISTA produces internationally agreed rules for seed sampling and testing and disseminates knowledge in seed science and technology. This facilitates seed trading nationally and internationally and is focusing on agricultural species. The principles of studying agricultural species and weed species are the same, but for weed species we are not interested in certifying their germinability, but to discover how to counteract their germination ability and vigor. In crop science and breeding the aim is the opposite to select for germinability and vigor.

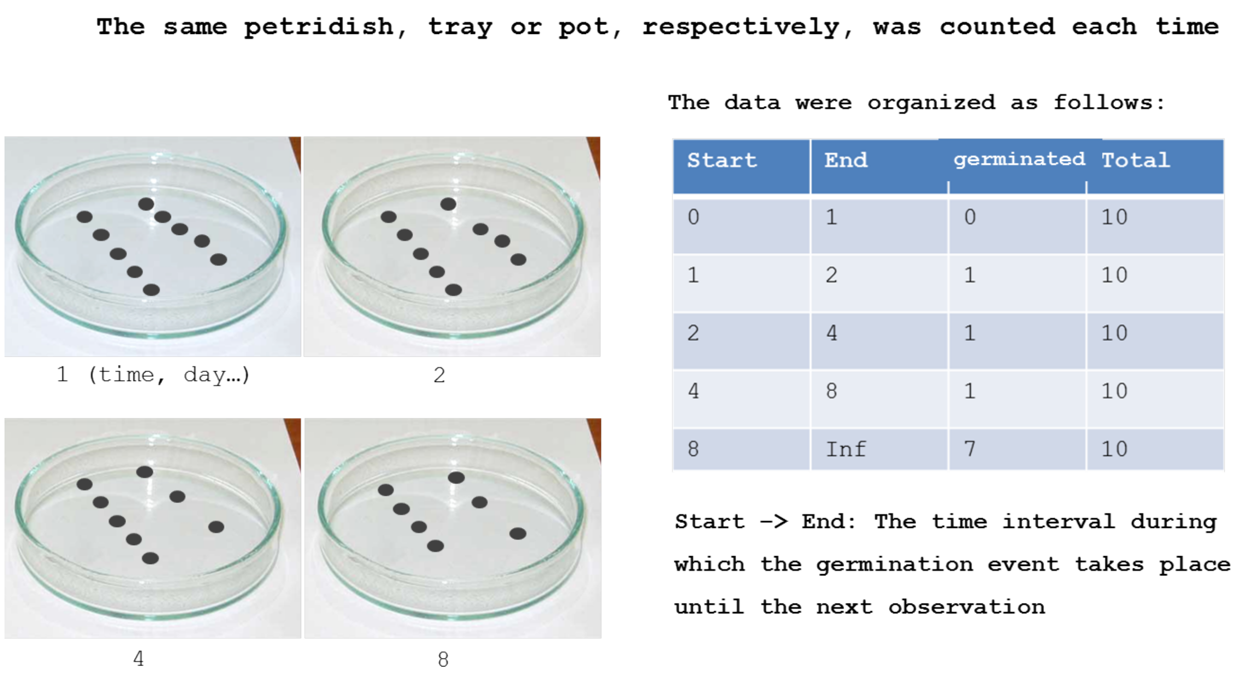
Nonlinear regression is commonly used to evaluate cumulative germination in cause of time, temperature or other external gradients. This approach, however, is problematic as it ignores that successive observations on the germination curve are highly correlated. The total number of seeds that have germinated at a particular time is highly dependent on the number of seeds that already have germinated. Moreover, variation in the proportions of germinated seeds will vary with time, being largest at intermediate monitoring intervals and smallest at the initial and final intervals, where germination activity is low. This means that the fundamental assumptions implicitly underlying nonlinear regression (independence among proportions and variance homogeneity) are not met.

In experiments, where new batches of seed are used for each inspection time, data can be analyzed by well-established models for binomial data such as log-logistic regression or other generalized linear models.

In experiments where the same batch of seeds is followed over time, the same seeds are observed repeatedly for some pre-specified duration of an experiment. We observe the waiting time until the event of interest occurs and the resulting data are often referred to as event-time or time-to-event data.

In this chapter we will concentrate on time-to-event data Ritz et al. 2013. Figure 14.1 shows the basic arrangement of this kind of experiment and how we arrange the data in R.

For various reasons germination need not occur at all during the experiment. We refer to this as right-censoring. Consequently, a plausible statistical model has to incorporate that the event of interest, seed germination, may occur after termination of the experiment, assigning a positive probability to the situation that germination is not observed. This is why the final inspection time has an `Inf` value that means infinity concludes the experiment.



The same petridish, tray or pot, respectively, was counted each time

The data were organized as follows:

Start	End	germinated	Total
0	1	0	10
1	2	1	10
2	4	1	10
4	8	1	10
8	Inf	7	10

Start -> End: The time interval during which the germination event takes place until the next observation

Figure 15.1: Example of how an experiment set up with four petridishes and how the data are arranged for being prepared for time-to-event analysis

Also, the germination is obviously not observed exactly at the very point in time, when the event took place. For instance, seeds in petridishes may only be inspected on a daily or weekly basis and in case a seed has germinated the only information we get: because germination took place between two inspection intervals. This type of time-to-event data is often referred to as grouped data or interval-censored. As a consequence an appropriate model should be specified by means of the probability of germination within the monitoring intervals. This probability will depend heavily on the monitoring intervals. Ignoring this grouped-data structure may lead to biased estimates. The finer the time grid the smaller the error caused by ignoring discretization.

15.1 Example 1: Chickweed

```r
library(drc)
head(chickweed,3)
```

```
##   start end count
## 1     0  12     0
## 2    12  22     0
## 3    22  30     0
```

```r
tail(chickweed,3)
```

```
##    start   end count
## 33 266.5 276.5     1
## 34 276.5 281.5     0
## 35 281.5   Inf   160
```

```r
chickweed.m1 <- drm(count~start+end, data = chickweed, fct = LL.3(names=c("b","MaxGerm","T50")), typ
```

Notice the model `count~start+end` operates with two independent variables the `start` and `end`. It means that the intervals between inspection can vary; there is not need to have the same inspection intervals for the whole period of the experiment.

```r
summary(chickweed.m1)
```

```
##
## Model fitted: Log-logistic (ED50 as parameter) with lower limit at 0 (3 parms)
##
## Parameter estimates:
##
##                       Estimate Std. Error t-value   p-value
## b:(Intercept)        -20.76747    2.94423 -7.0536 1.743e-12 ***
## MaxGerm:(Intercept)    0.20011    0.02830  7.0711 1.537e-12 ***
## T50:(Intercept)      196.05308    2.50570 78.2427 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The output from the `drm()`is the same as for other non-linear regressions, and the three parameter log-logistic model, `LL.3()`, is also familiar in dose-response work. What is new here is that the model needs `start+end,...,type = "event"`. The regression is illustrated in Figure 15.2.

```r
par(mfrow=c(1,1), mar=c(3.6,3.6,.5,.5), mgp=c(2.5,.7,0))
plot(chickweed.m1, xlab = "Time (hours)", ylab = "Proportion germinated",
     xlim=c(0, 340), ylim=c(0, 0.25), log="", lwd=2, cex=1.2, bty="l")
```

On the basis of the regression parameters we can see that the maximum germination rate is not that high and the $T_{50}$ is 196 hours, meaning that it takes 196 days to reach 0.20/2= 0.1 or 10% germination. The interesting biological parameters are the upper limit (maximum germination) 0.20 (0.03) and the time to 50% germination, $T_{50}$, 196 (2.5).

As was the case in the dose-response chapter we can also estimate how long it takes to reach 10 or 90 percent germination.

```r
## Calculating t10, $T_{50}$, t90 for the distribution of viable seeds
ED(chickweed.m1, c(10, 50, 90), interval="delta")
```

```
##
## Estimated effective doses
##
##         Estimate Std. Error    Lower    Upper
## e:1:10  176.3700     3.4930 169.5239 183.2161
## e:1:50  196.0531     2.5057 191.1420 200.9642
## e:1:90  217.9328     4.2730 209.5578 226.3078
```

If a cumulative germination curve had been analysed by a nonlinear regressions, the parameters would be similar, but their standard errors would be overly precise. If we are interested only in the curve fitting and not the uncertainties of parameters, either way of fitting may suffice. But usually we want to get the correct standard error of parameters to compare germination ability and/or vigor among species, biotypes, or environments, and the time-to-event regression is the appropriate way to better estimate the uncertainty in germination estimates, and may prevent us from drawing incorrect conclusions.

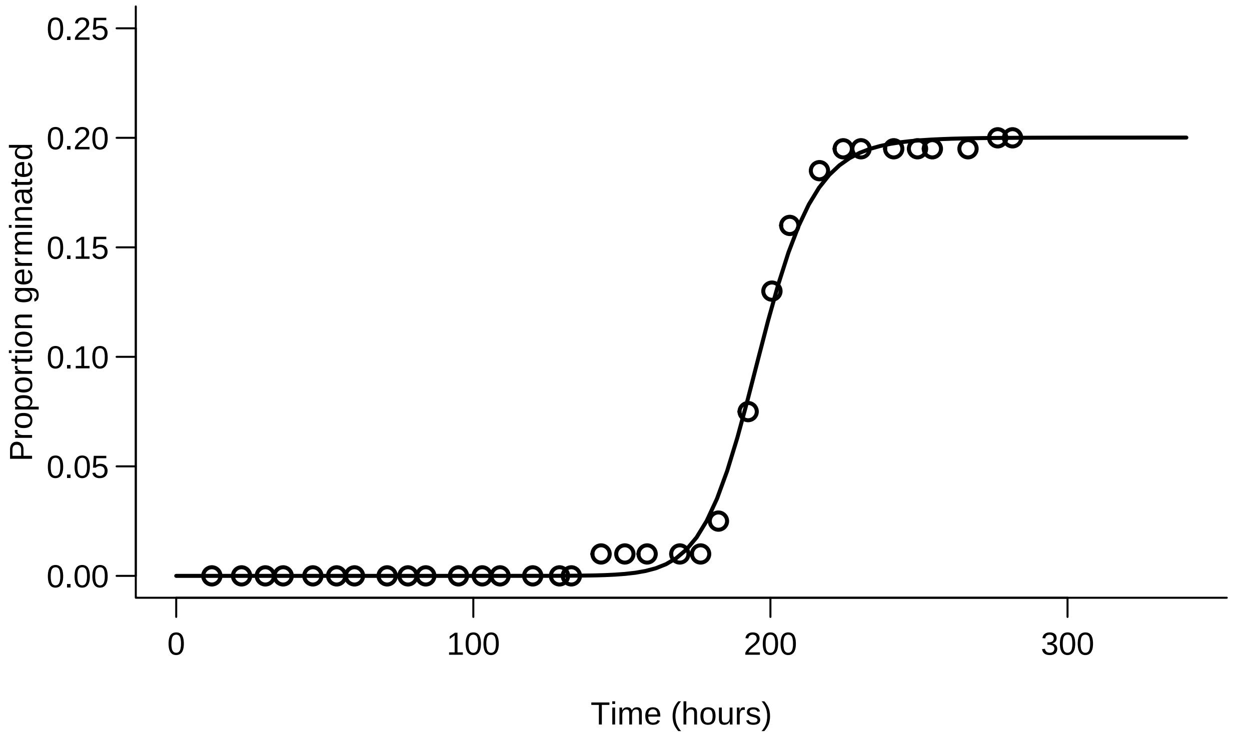


Figure 15.2: *Germination curves of ALS resistant* Stellaria media *from Danish fields in 2001. The code is from the example data set chickweed in the* `drc` *package Ritz et al. 2013. Note that we now use the actual time on the x-axis and not the logarithm as we did in the dose-response curve chapter.*

15.2 Example 2: *Bromus tectorum*

Another example uses germination data for three different biotypes of *Bromus tectorum* collected in Wyoming. Seeds were placed in petri dishes on moist filter paper in a germination chamber. The petri dishes were checked for germinated seeds once daily from the initiation of the experiment for 20 days. Each day the number of newly germinated seeds was recorded.

```
germ.dat<-read.csv("http://rstats4ag.org/data/broteGerm.csv")
head(germ.dat,3)
```

```
##   Source starttime endtime REP Daily.germ Cum.Germ
## 1     S1         0       1   1          0        0
## 2     S1         0       1   2          0        0
## 3     S1         0       1   3          0        0
```

There are 6 columns in the data set, the `Source` is the *B. tectorum* biotype, `starttime` and `endtime` denote the beginning and end of the interval represented by each recording time. This is similar to the chickweed data set. The `Daily.germ` column represents the number of germination events that occurred since the last observation. The `Cum.germ` column is a cumulative total that is often used (incorrectly) for analyzing germination over time.

The final observation in this experiment occurred 20 days after it began. Therefore, we have an interval in the data set from time period 20 to `Inf`. Many of the petri dishes had not germinated all seeds at the final observation. We don't know for certain that these seeds *never* would have germinated. All we can say is that they had not yet germinated at the final observation of day 20. In the data set, we incorporate this uncertainty by recording the number of non-germinated seeds in the interval 20 to `Inf`.

```
event.drc<-drm(Daily.germ ~ starttime+endtime, Source, data=germ.dat,
               fct=LL.3(names=c("Slope","Max","T50")), type="event",
               upperl=c(NA,1,NA))
summary(event.drc)
```

```
##
## Model fitted: Log-logistic (ED50 as parameter) with lower limit at 0 (3 parms)
##
## Parameter estimates:
##
##             Estimate Std. Error t-value   p-value
## Slope:S1   -9.037575   0.673091 -13.427 < 2.2e-16 ***
## Slope:S2 -21.195415   1.478692 -14.334 < 2.2e-16 ***
## Slope:S3  -6.881944   0.487129 -14.128 < 2.2e-16 ***
## Max:S1     0.723603   0.031958  22.642 < 2.2e-16 ***
## Max:S2     0.861457   0.024301  35.449 < 2.2e-16 ***
## Max:S3     0.823935   0.027511  29.949 < 2.2e-16 ***
## T50:S1    11.159361   0.183671  60.757 < 2.2e-16 ***
## T50:S2    10.207564   0.067115 152.091 < 2.2e-16 ***
## T50:S3     9.481163   0.189789  49.956 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
par(mar=c(3.6,3.6,.5,.5), mgp=c(2,.7,0))
plot(event.drc, log="", #ylim=c(0,1), xlim=c(0,20),  bty="l",
     ylab="Proportion of germinated seed", xlab="Time (d)",
     legendPos=c(20,.40), pch=1:3)
```

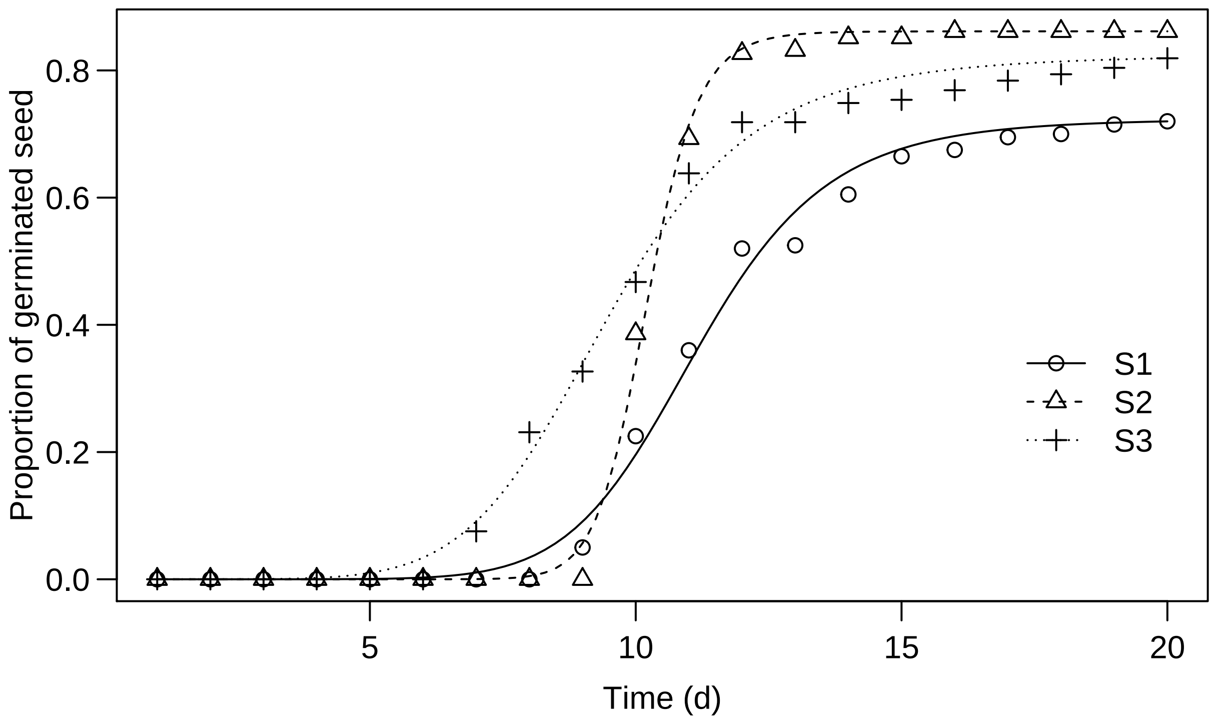


Figure 15.3: *Regression fits for the time-to-event 3 parameter log-logistic regression. Please note that the curves for the time to event do stop at 20 days, because we do not know about future germination beyond 20 days.*

In this code, we have set the `names` argument within the `drc` fitting function `LL.3()`. By default, the three parameters are named `b, d`, and `e`. For users who deal with these models on a regular basis, it is certainly acceptable to leave the default parameter names and interpret them accordingly. However, renaming the parameters to `Slope`, `Max`, and T_{50} provides a useful reminder of how to interpret the parameters in the model output. This is especially helpful when presenting the results to someone unfamiliar with `R` syntax.

In germination studies, all three parameters have some biological interpretation. The upper limit being the maximal germination, with an upper limit set at 1 (or 100% of the seed); T_{50} is the time taken to reach 50% germination relative to the upper limit. The slope could give an indication of how quickly seed germination reaches completion once the process has begun. The T_5 or T_{95} (the time required for 5 and 95 percent of the seeds to germinate, respectively) can also be estimated in relation to the upper limit.

```
#The ED5,ED50,and ED95 for the the germination
ED(event.drc,c(5,50,95), interval="delta")
```

```
##
## Estimated effective doses
##
##           Estimate Std. Error     Lower     Upper
## e:S1:5    8.056499   0.223117  7.619198  8.493799
## e:S1:50  11.159361   0.183671 10.799373 11.519350
## e:S1:95  15.457253   0.477020 14.522312 16.392194
## e:S2:5    8.883630   0.102078  8.683561  9.083700
## e:S2:50  10.207564   0.067115 10.076021 10.339106
## e:S2:95  11.728803   0.139904 11.454596 12.003010
## e:S3:5    6.180858   0.216678  5.756177  6.605539
## e:S3:50   9.481163   0.189789  9.109183  9.853142
## e:S3:95  14.543684   0.545492 13.474539 15.612829
```

Usually the T_{50} is the most precise estimate, this does also apply here when we look at the 95% confidence interval or the coefficient of variation.

```
#Coefficient of variation for ED50
T50<-ED(event.drc,c(50),display=FALSE)[,2]*100/ED(event.drc,c(50),display=FALSE)[,1]
T50
```

```
##   e:S1:50   e:S2:50   e:S3:50
## 1.6458918 0.6574998 2.0017478
```

```
#Coefficient of variation for ED5
T5<-ED(event.drc,c(5),display=FALSE)[,2]*100/ED(event.drc,c(5),display=FALSE)[,1]
T5
```

```
##   e:S1:5   e:S2:5   e:S3:5
## 2.769398 1.149057 3.505629
```

```
#Coefficient of variation for ED95
T95<-ED(event.drc,c(95),display=FALSE)[,2]*100/ED(event.drc,c(95),display=FALSE)[,1]
T95
```

```
##  e:S1:95  e:S2:95  e:S3:95
## 3.086056 1.192825 3.750715
```

Direct comparison of model parameters can be done by using the `compParm()` function in `drc`. Because we have used the `names` argument above, we need to specify those names in the `compParm()` syntax. If we had not used the `names` argument, we would compare slopes by specifying the b parameter, maximum asymptote d parameter, and T_{50} the e parameter.

```
compParm(event.drc, "Slope")
```

```
##
## Comparison of parameter 'Slope'
##
##       Estimate Std. Error  t-value   p-value
## S1/S2 0.426393   0.043513 -13.1825 < 2.2e-16 ***
## S1/S3 1.313230   0.134932   2.3214   0.02027 *
## S2/S3 3.079859   0.306093   6.7949 1.084e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Slopes are different between S2 and the rest as also seen in Figure 15.3. It means that the vigor of the germination process for S3 is greater that for the other two biotypes. Looking at the upper limit:

```
compParm(event.drc, "Max")
```

```
##
## Comparison of parameter 'Max'
##
##       Estimate Std. Error t-value   p-value
## S1/S2 0.839976   0.044020 -3.6353 0.0002777 ***
## S1/S3 0.878229   0.048625 -2.5043 0.0122691 *
## S2/S3 1.045540   0.045702  0.9965 0.3190268
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The maximum germination for biotypes S2 and S3 is similar, but biotype S1 has a lower maximum germination than do the other two biotypes. Finally, the time required for 50% germination can also be compared in a similar manner.

```
compParm(event.drc, "T50")
```

```
##
## Comparison of parameter 'T50'
##
##       Estimate Std. Error t-value   p-value
## S1/S2 1.093244   0.019376  4.8123 1.492e-06 ***
## S1/S3 1.177003   0.030502  5.8030 6.515e-09 ***
## S2/S3 1.076615   0.022684  3.3775 0.0007314 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The T_{50} (which is also be associated with speed of germination) are different form each other. Looking at Figure 15.3, it is noted that the logistic regression does not seem to be that good to capture the upper limits for S2 and S3, and perhaps the symmetric sigmoid curves would could be replaced by an asymmetric one.

By using the Weibull 1 model, `W1.3()`, which is asymmetric (i.e. T_{50} is not a parameter in the model). We may be better to model the upper limit. However, the Weibull 1 is rather sensitive to too many zeros so we must get rid of those in order to get a proper fit.This is done by omitting observations with the functions `subset(germ.dat, Daily.germ!=0)`. This function gives some warnings that we can ignore for the time being.

```
#Fitting LL.3() with the same conditions as for the Weibull 1 model
event.drc.1<-drm(Daily.germ ~ starttime+endtime, Source,
                 fct=LL.3(names=c("Slope","Max","T50")),
                 data=subset(germ.dat, Daily.germ!=0), type="event")

#Fitting W1.3()
event.drc.W1 <- drm(Daily.germ ~ starttime+endtime, Source,
                    fct=W1.3(),
                    data=subset(germ.dat, Daily.germ!=0), type="event")

#To extract the T50
ED(event.drc.1,50) # LL.3 model
```

```
##
## Estimated effective doses
##
##          Estimate Std. Error
## e:S1:50 11.159333   0.183699
## e:S2:50 10.207537   0.067106
## e:S3:50  9.481139   0.189743
```

```
ED(event.drc.W1,50) # W1.3 model
```

```
##
## Estimated effective doses
##
##          Estimate Std. Error
## e:S1:50 11.043969   0.183815
## e:S2:50 10.151655   0.061806
## e:S3:50  9.352818   0.211253
```

The T_{50} did not change much but in Figure 15.4, there are subtle changes in the curve on the right side by going from a log-logistic (`LL.3()`) to an asymmetric (`W1.3()`) curve. There are some differences in the way the upper limit is described by the two models. The Weibull 1 seems to do a better job than the log-logistic.

```
par(mfrow=c(1,1), mar=c(3.2,3.2,0.5,0.5), mgp=c(2,.7,0))
plot(event.drc.W1, log="", ylim=c(0,1), xlim=c(0,20), bty="l",
     ylab="Proportion of germinated seed", xlab="Time (d)",
     legendPos=c(20,.40))
plot(event.drc.1, log="", ylim=c(0,1), xlim=c(0,20),
     type="none",
     add=TRUE, col=c(2), legendPos=c(20,.40))
```

Even though the changes are not dramatic the AIC does drop somewhat when going form a log-logistic to an Weibull 1, which indicates that overall the asymmetric `W1.3()` fits best to the data.

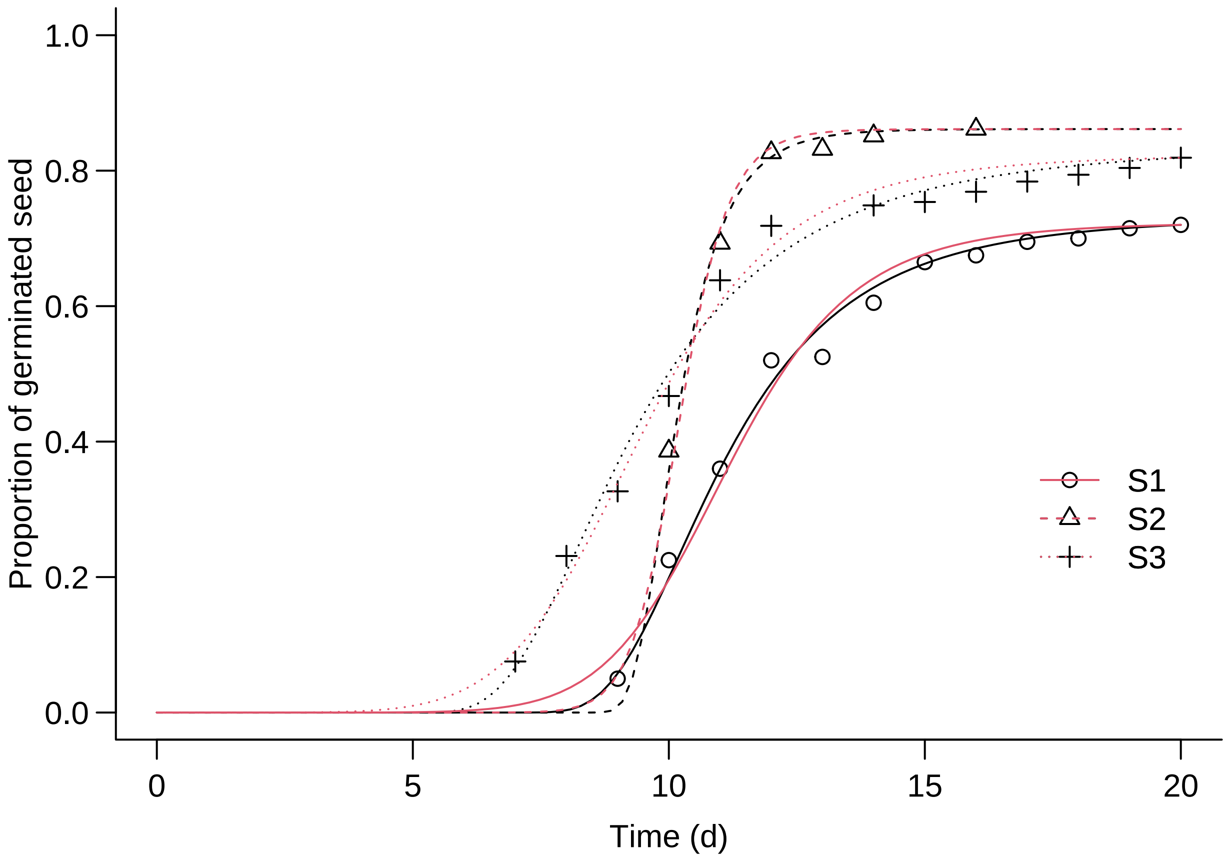


Figure 15.4: *An assymetric Weilbull 1 model(`W1.3()`) shown with black lines, compared with the symmetric log-logistic model (`LL.3()`) in red.*

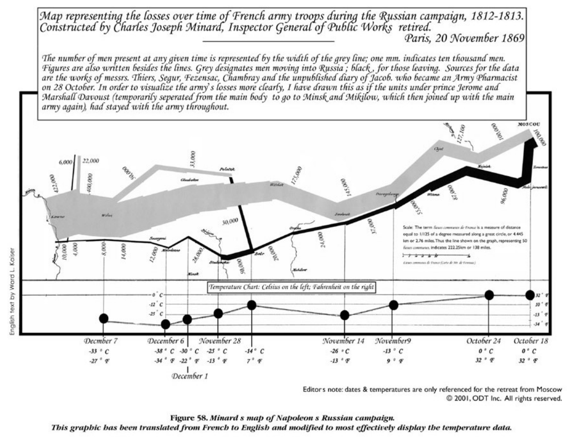
```
AIC(event.drc.1, event.drc.W1)
```

```
##               df      AIC
## event.drc.1  79 2600.177
## event.drc.W1 79 2526.731
```

Chapter 16

Visualization

Pictures and graphs may say more than 1000 words. After collection of data, the hard work of analyzing them according to the experimental design plans begins. But before you dig too much into statistics and jump to conclusions on the basis of the statistical tests, you should actually look at the data graphically. You must try to illustrate the data in such a way that it is informative not only to you, but also to potential readers of the project or the paper you wish to publish. In a way you could look at the visualization as the interface between science and art. One of the best examples of this merger of science and art is the campaign of Napoleon in Russia 1812-1813.


Map representing the losses over time of French army troops during the Russian campaign, 1812-1813. Constructed by Charles Joseph Minard, Inspector General of Public Works retired.
Paris, 20 November 1869

The number of men present at any given time is represented by the width of the grey line; one mm. indicates ten thousand men. Figures are also written besides the lines. Grey designates men moving into Russia ; black , for those leaving. Sources for the data are the works of messrs. Thiers, Segur, Fezensac, Chambray and the unpublished diary of Jacob. who became an Army Pharmacist on 28 October. In order to visualize the army's losses more clearly, I have drawn this as if the units under prince Jerome and Marshall Davoust (temporarily seperated from the main body to go to Minsk and Mikilow, which then joined up with the main army again), had stayed with the army throughout.

Editor s note: dates & temperatures are only referenced for the retreat from Moscow
© 2001, ODT Inc. All rights reserved.

Figure 58. ***Minard s map of Napoleon s Russian campaign.***
This graphic has been translated from French to English and modified to most effectively display the temperature data.

Figure 16.1: *Charles Joseph Minard (1781-1870) is most widely known for a single work, his illustration of the fate of Napoleon's Grand Army in Russia.* Source: http://www.datavis.ca/gallery/re-minard.php

Minard's work (Figure 16.1) has been called one of the best graphs ever produced in thematic cartography and statistical graphics. In short, it shows the size of the Army at the beginning of the attack and at the final retreat, some of the more important battles and the temperature during the retreat from Moscow. It contains so many dimensions and so much information and history that usually fill hundreds of pages in history books. Our work in weed science is rarely that exciting for the general public; although our discipline can be an esoteric at times, it plays a great role in the food production and sustainability of the farming and environment in the world.

16.1 Base R Graphics

Visualization of data is an important step to get a first impression of the quality of data and to see if some observations stand out from the rest. It can also help us observe whether the data show systematic relationships. Base R graphics are a quick and extensible way to do both this preliminary data inspection, but also for creating publication quality figures for manuscripts and presentations. We will begin this chapter with some basic examples of how to customize base R graphics to create high-quality figures.

16.1.1 Scatterplots

One of the most common figures is a scatter plot or x-y plot - typically used to see the relationship between an independent variable (x-axis) and a response variable (y-axis), although often used to show other relationships as well. Without much work, we can get a very simple scatterplot suitable for inspecting the data in the `cars` data set that is part of the base `R` installation (Figure 16.2).

Code to produce Figure 16.2:

```r
plot(dist ~ speed, data = cars, main="Recorded in the 1920's")
```

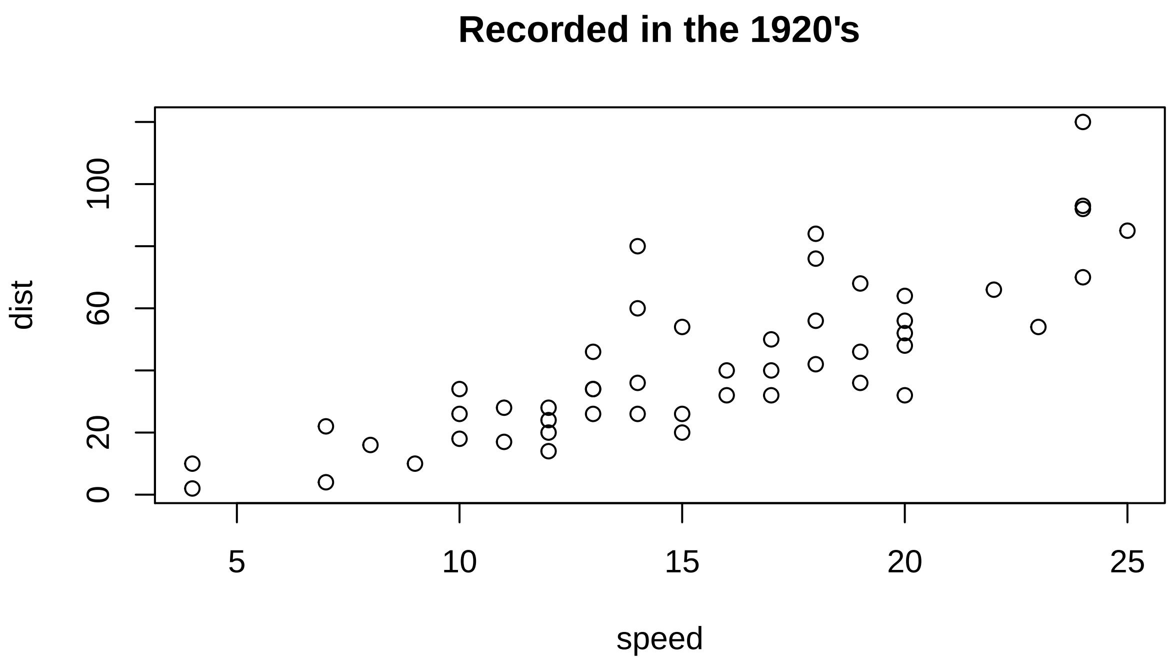


Figure 16.2: *A most basic scatterplot in R, showing the speed of cars and the distance required for the car to stop. The tilde ~ can be difficult to find on some keyboards, depending on you language, but it is the defined ASCII code 126. In Windows you can press the ALT key, activate Num. Lock and press 126.*

While Figure 16.2 is adequate for preliminary inspection of one's own data, it is woefully insufficient for submission to a journal. We can improve the figure greatly by making some simple additions to the basic plot code. To create Figure 16.3, the `par()` function is called to change some of the default graphing parameters. In this case we adjust the margins so that the bottom and left margins are set to 3.2, the top and right margins are set to 0.5. We have removed the main title (since that information would typically be contained in a figure caption according to most journal instructions), and also adjusted the axis labels, by setting them closer to the axis compared to default values using `mgp`. The `mgp` argument requires 3 numbers, which set the distance from the axis for the axis title, axis labels, and axis line, respectively. The defaults for `mgp` are 3, 1, 0 - so in this code we've removed some of the axis white space by moving the labels closer to the axis. We have also added axis labels and a main title, changed the plotting character (`pch=19`, which is a closed circle), and added some transparency to the points using the `rgb()` function and the `alpha` argument. Finally, we have, adjusted the x-axis to include 0, and removed the box around the figure by setting `bty="n"`. Setting `bty="l"` will leave the bottom and left lines of the box, and `bty="o"` will plot the entire box (the default).

Code to produce Figure 16.3:

```
par(mar=c(3.2,3.2,0.5,.5), mgp=c(2,.7,0))
plot(dist ~ speed, data = cars,
     xlab="Speed of car (mph)",
     ylab="Distance required to stop (ft)",
     pch=19, col=rgb(red=.3, blue=.3, green=.3, alpha=0.7),
     bty="n", xlim=c(0,25))
```

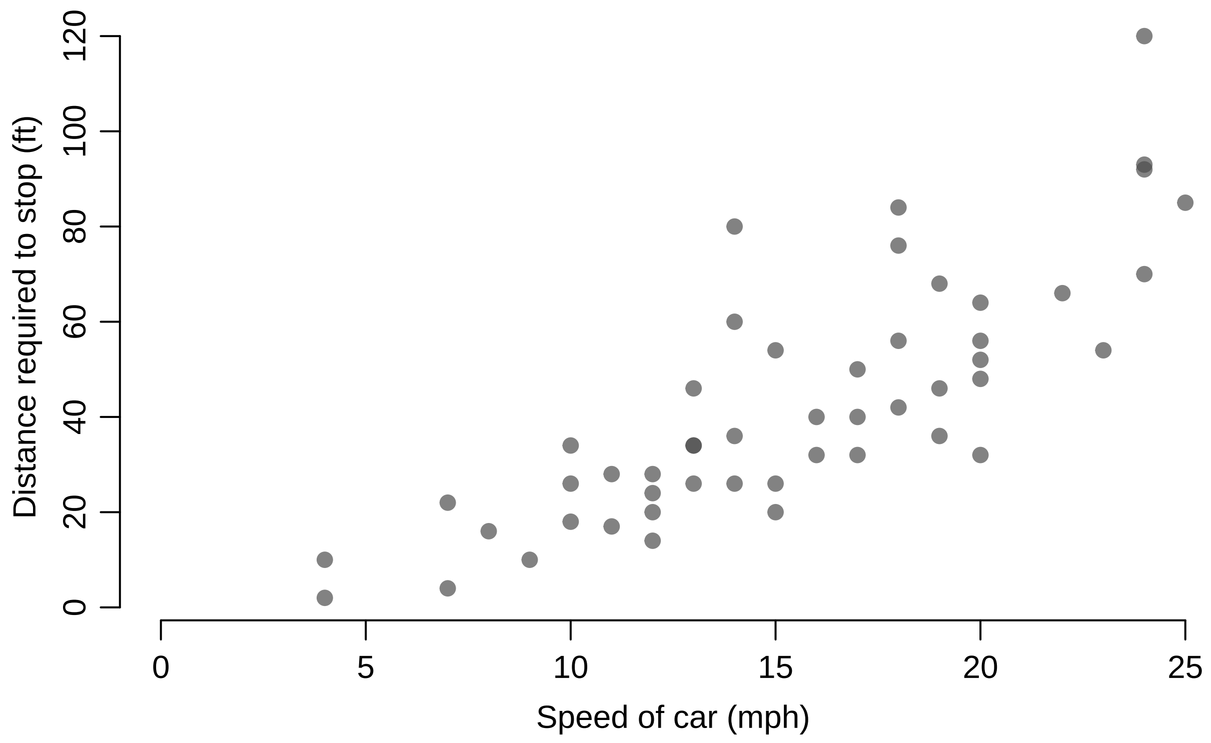


Figure 16.3: *An improved scatter plot showing the cars data.*

16.1.2 Box and Whisker Plots

If we have some data where the "treatment" is not based upon a continuous x variable, then we can illustrate the variation with a boxplot. The data for this example `InsectSprays` are also included with the basic R installation.

Code to produce Figure 16.4:

```
par(mfrow = c(1,2), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(count ~ spray, data = InsectSprays)
  legend("topleft","(a)", bty="n")
plot(count ~ as.numeric(spray), data = InsectSprays, pch=19)
  legend("topleft","(b)", bty="n")
```

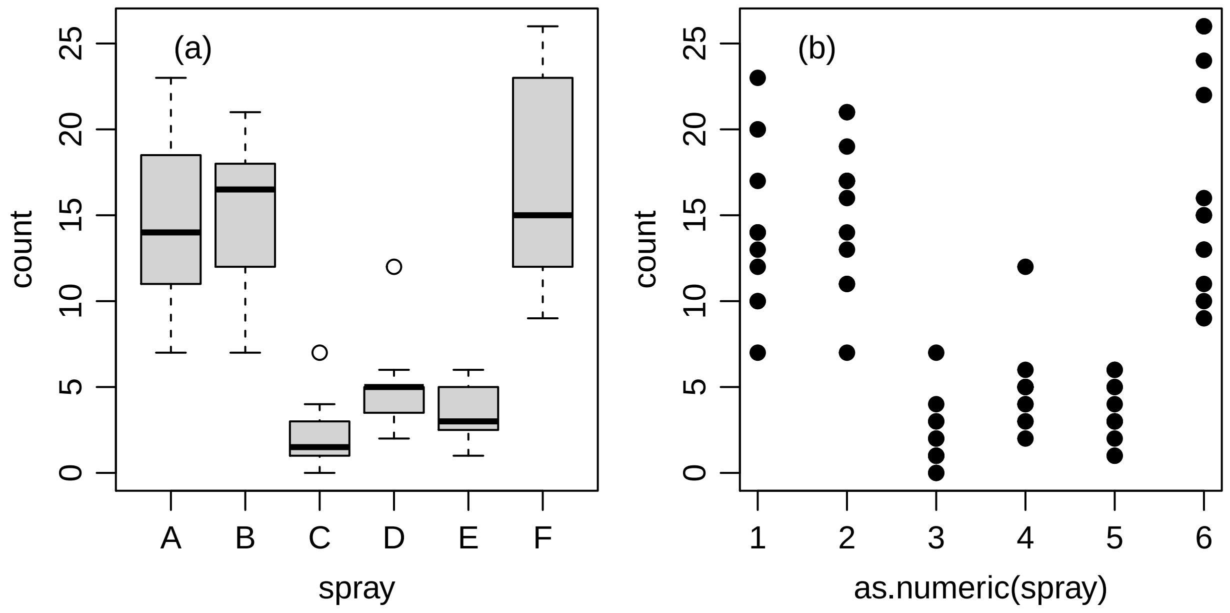


Figure 16.4: *Counts of insects in agricultural experimental units treated with different insecticides.*

Figure 16.4 illustrates how the default `plot()` function works in R; if the x-axis is based on factors (A through F) you automatically get a boxplot (Figure 16.4a), where the median and the quartiles are shown, as well as an indication of extreme observations. On the other hand, if we convert the factors A through F to numerical values of x by using function `as.numeric()`, we get an ordinary continuous x variable (Figure 16.4b). Factor A becomes 1 and factor B becomes 2 and so on. In this particular data we can change the names (e.g., F is becoming A and A becoming F) of the various sprays and it will just change the order on the x-axis. In doing so it could have a great influence on perceived relationship.

It is possible to combine these two panels in Figure 16.4 into a single panel, to more concisely show a high level of detail on the variance within and among treatments. To do so, we modify the code for Figure 16.4 by using `points()` instead of `plot()` for the second plot so that the points are added to the first plot instead of creating a new plotting space; jittering the points, or offsetting them slightly, in the 'x' direction so that counts that are similar or the same can be more easily seen; adding some transparency to the points so that we can see if multiple points are on top of one another; and by removing the outlier points from the initial boxplot since all points will be added later.

Code to produce Figure 16.5:

```
par(mfrow = c(1,1), mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(count ~ spray, data = InsectSprays, pch=NA)
points(count ~ jitter(as.numeric(spray)), data = InsectSprays,
       pch=19, col=rgb(.1,.4,.8, alpha=0.7))
```

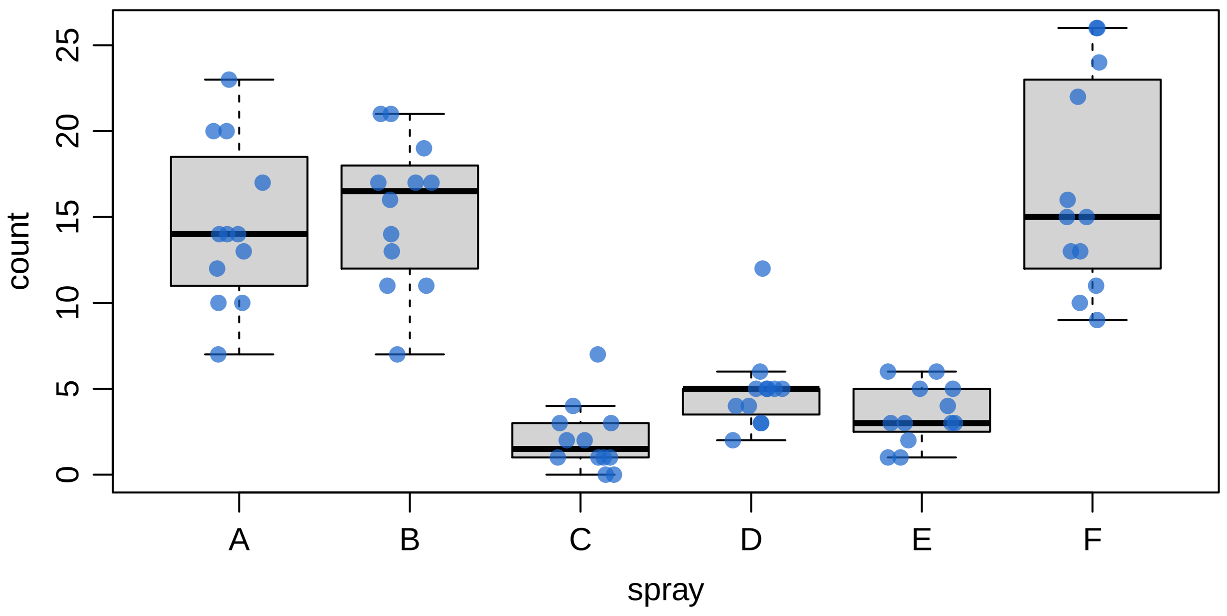


Figure 16.5: *Counts of insects in agricultural experimental units treated with different insecticides - shown with points on top of the boxplots.*

For data with continuous x we can also directly call the `boxplot()` function rather than relying on `R` to correctly identify the independent variable type. We can also use some additional arguments within the `boxplot()` function to modify the look of our boxplot (Figure 16.6).

Code to produce Figure 16.6:

```r
par(mfrow = c(1,1), mar=c(3.2,8.2,.5,.5), mgp=c(2,.7,0))
boxplot(count ~ as.numeric(spray), data = InsectSprays, horizontal=TRUE, pch=NA,
        names=c("Antibug","Bug Kill","Clear","Destroy","Endgame","Fuhgeddaboudit"),
        las=1, xlab="Insect counts", ylab="")
points(jitter(as.numeric(spray)) ~ count, data = InsectSprays,
       pch=19, col=rgb(.1,.4,.8, alpha=0.7))
```

16.1.3 More than Two Variables

One of the classical examples in statistics are the Fisher's or Anderson's iris data that give measurements of the sepal length and width and petal length and width, respectively, for each of three species of Iris. The `pairs()` function is a powerful (and quick) way to show the pairwise relationships between many variables at the same time.

Code to produce Figure 16.7:

```r
head(iris,3)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

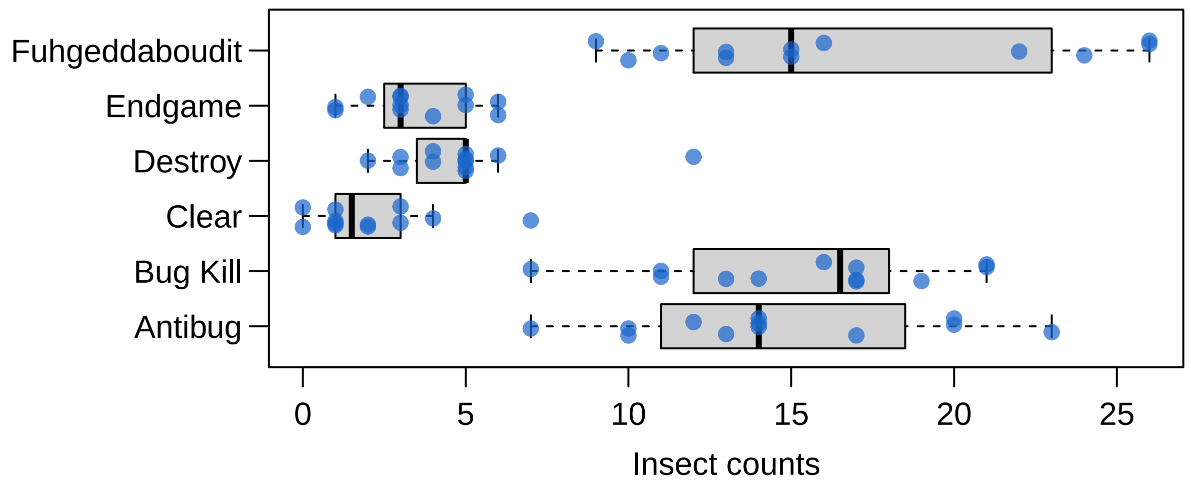


Figure 16.6: *The counts of insects in agricultural experimental units treated with different insecticides. Setting the option `horizontal=TRUE` creates a horizontal boxplot instead of vertical; we can use the `names` argument to add product names, and the `las` argument rotates the axis labels to horizontal.*

```
par(mar=c(2.2,2.2,2.2,2.2), mgp=c(2,.7,0))
pairs(~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data=iris)
```

It is obvious from Figure 16.7 that the relationships sometimes seem to be separated into two groups. However, knowing that the data are separated into three *Iris* species, we could once more use `as.numeric()` to identify which data belong to which species, and make each species a different color (Figure 16.8).

Code to produce Figure 16.8:

```
pairs(~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data=iris,
     col = as.numeric(iris$Species))
```

The `pairs()` function has even more power when used together with a function called `panel.cor()` that can be found at the end of the `?pairs` help page. If you find outliers in data and wish to omit them, then by using the original panel function you must omit the whole observation. Here, the function panel.cor has been improved by using the argument (`use = "pairwise.complete.obs"`).

```
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...) {
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
 r <- abs(cor(x, y, use = "pairwise.complete.obs"))
  txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- paste(prefix, txt, sep = "")
  if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
  text(0.5, 0.5, txt, cex = cex.cor * r)
}
```

For the average R user (and even the authors of this chapter), this looks very complicated and it probably is; but by running it before you run the `pairs()` you get a fantastic overview of relationships among variables. In soils and plant sciences you often have numerous variables, i.e. elements or chemicals or morphological attributes. Before launching the statistics weaponry it would be nice to see the relationships between these variables, as we will show using the *Nitrate* data set.

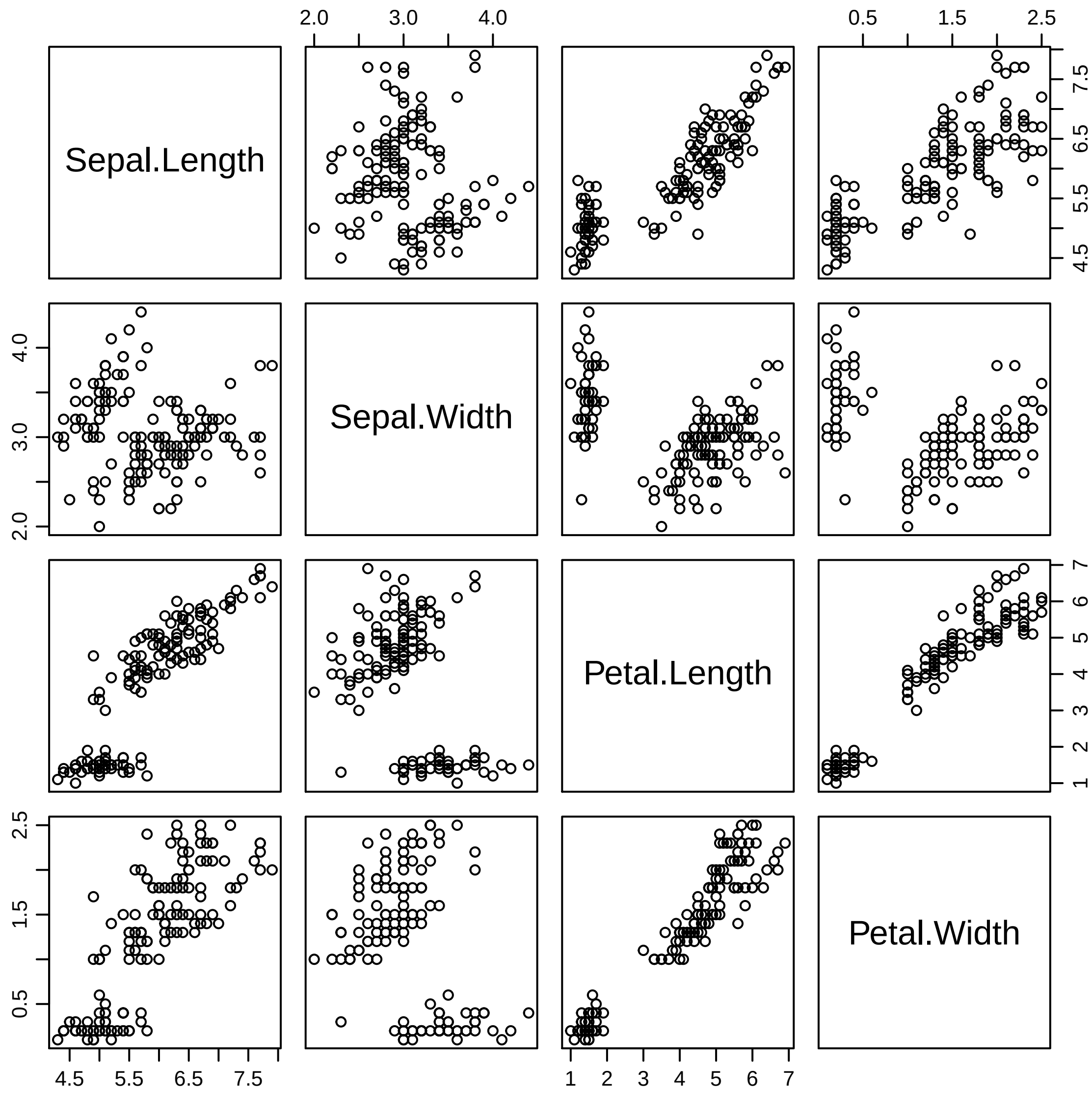


Figure 16.7: *Sepal and petal measurements on 50 flowers from each of three species of Iris*: I. setosa, I. versicolor, *and* I. virginica.

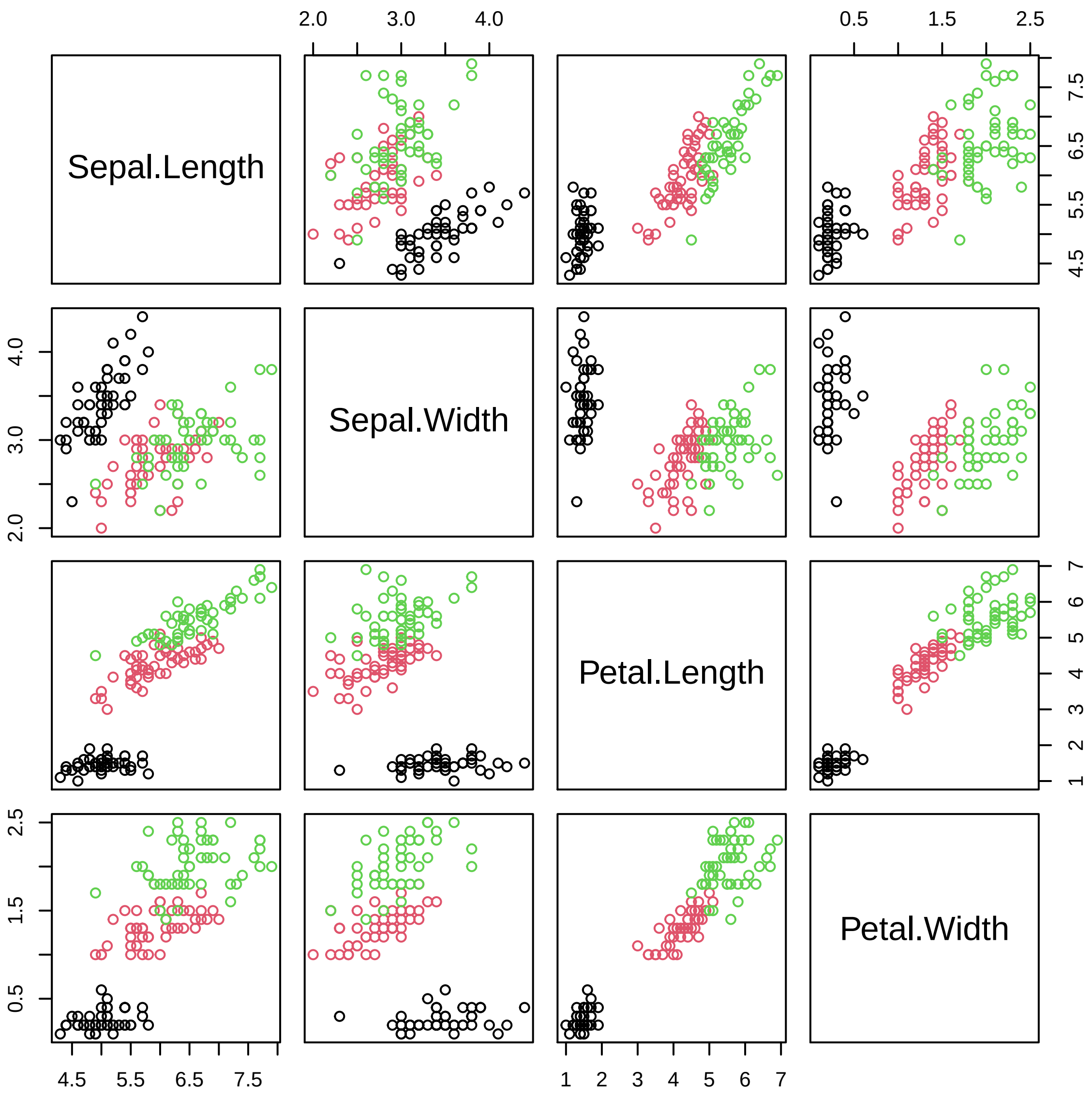


Figure 16.8: *By using the argument* `col = as.numeric(iris$Species)` *the colors separate the various species, making the differences and relationships among species obvious.*

```r
Nitrate <- read.table("http://rstats4ag.org/data/nitrate.txt", header=T)
head(Nitrate,3)
```

```
##   age no3n   pH   Na    K
## 1   0 0.13 4.91 4.36 0.14
## 2   0 0.15 5.00 3.91 0.21
## 3   0 0.05 5.37 3.98 0.16
```

We execute the `panel.cor()` function and call it within the `pairs()` function to see the relationships between all variables in the data set. Note that the correlations are absolute.

Code to produce Figure 16.9:

```r
pairs(Nitrate, lower.panel = panel.smooth, upper.panel = panel.cor)
```

It is clear from Figure 16.9 that K has virtually no correlation with the other variables. This is solely due to an observation with a rather high K concentration, seen in the last panel row. In a case like this, it would certainly be wise to determine what is going on with that particular observation - measurement or data entry errors could be potential explainers. For now, we will exclude that observation to look at the overall trends. To do so, we can simply find the observation with the largest K-value:

```r
which.max(Nitrate[["K"]])
```

```
## [1] 10
```

The maximum value is in the 10th observation. Secondly, we can change this putative outlier to a missing value (expressed as 'NA' in `R`). To do this, we define a new data set (`Nitrate1`) in order to not change the original data we have. We will then select the value of K that matches the maximum value of K (the 10th observation), and replace that value with 'NA':

```r
Nitrate1<-Nitrate
Nitrate1$K[Nitrate1$K==max(Nitrate1$K)] <- NA # This is missing value in R
```

We can then re-run the `pairs()` function on the new data set to see how removal of the outlier changes the relationship between K and the other variables.

Code to produce Figure 16.10:

```r
pairs(Nitrate1, lower.panel = panel.smooth, upper.panel = panel.cor)
```

A field experiment, with mechanical control of weeds in cereals, was done by measuring the effect on leaf cover as a function of number of harrow runs. The harrowing was done either along the crop rows or across the crop rows at a certain stage of cereal development.

```r
leafCover<-read.csv("http://rstats4ag.org/data/LeafCover.csv")
head(leafCover,3)
```

```
##         Leaf Intensity Direction
## 1 0.18623052         0     Along
## 2 0.12696533         1     Along
## 3 0.09006582         2     Along
```

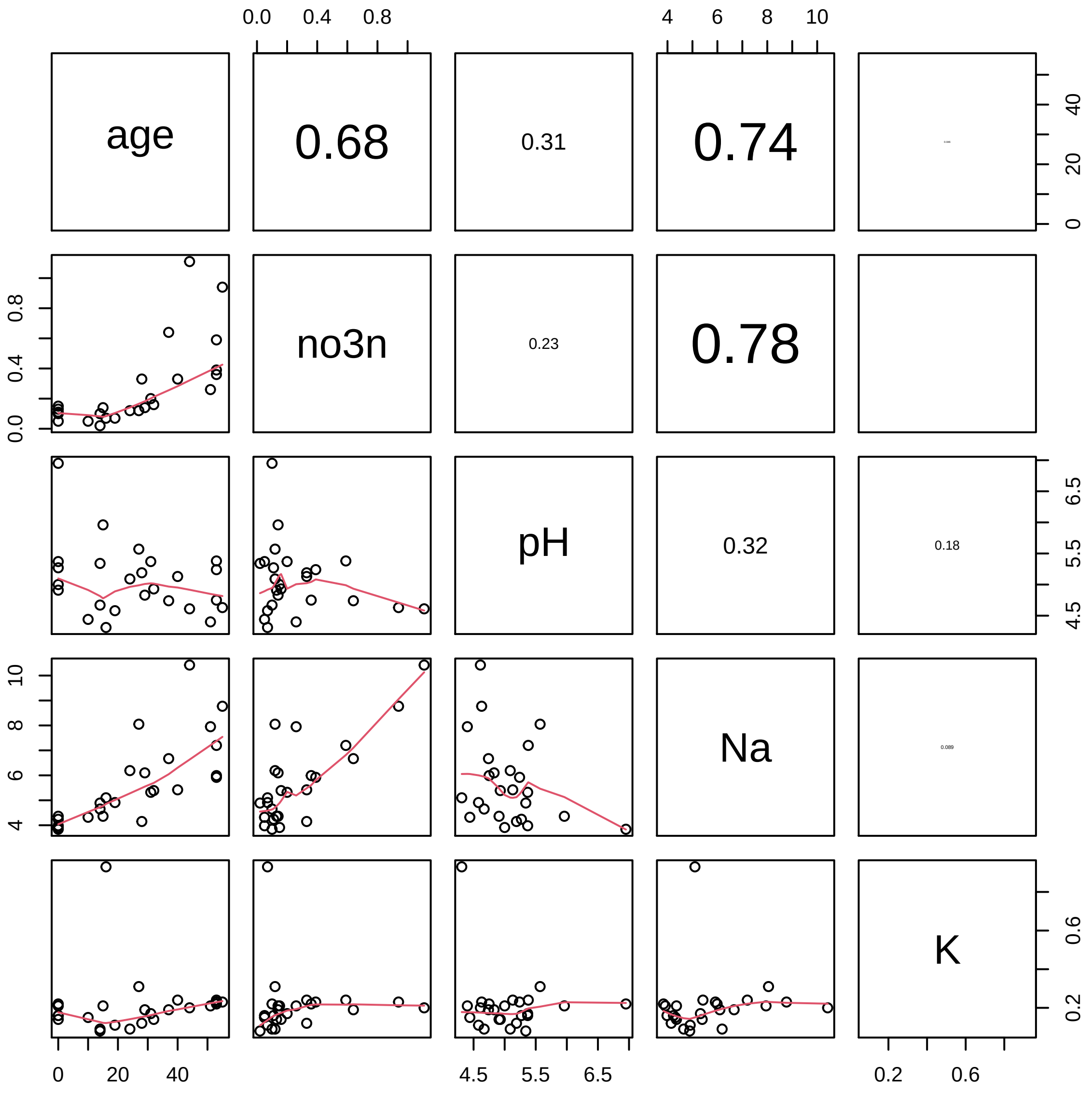


Figure 16.9: *The lower part of the panel shows the relationships soil variables. The upper part shows the absolute correlation coefficients, the font sizes of which are based upon the magnitude of the correlation.*

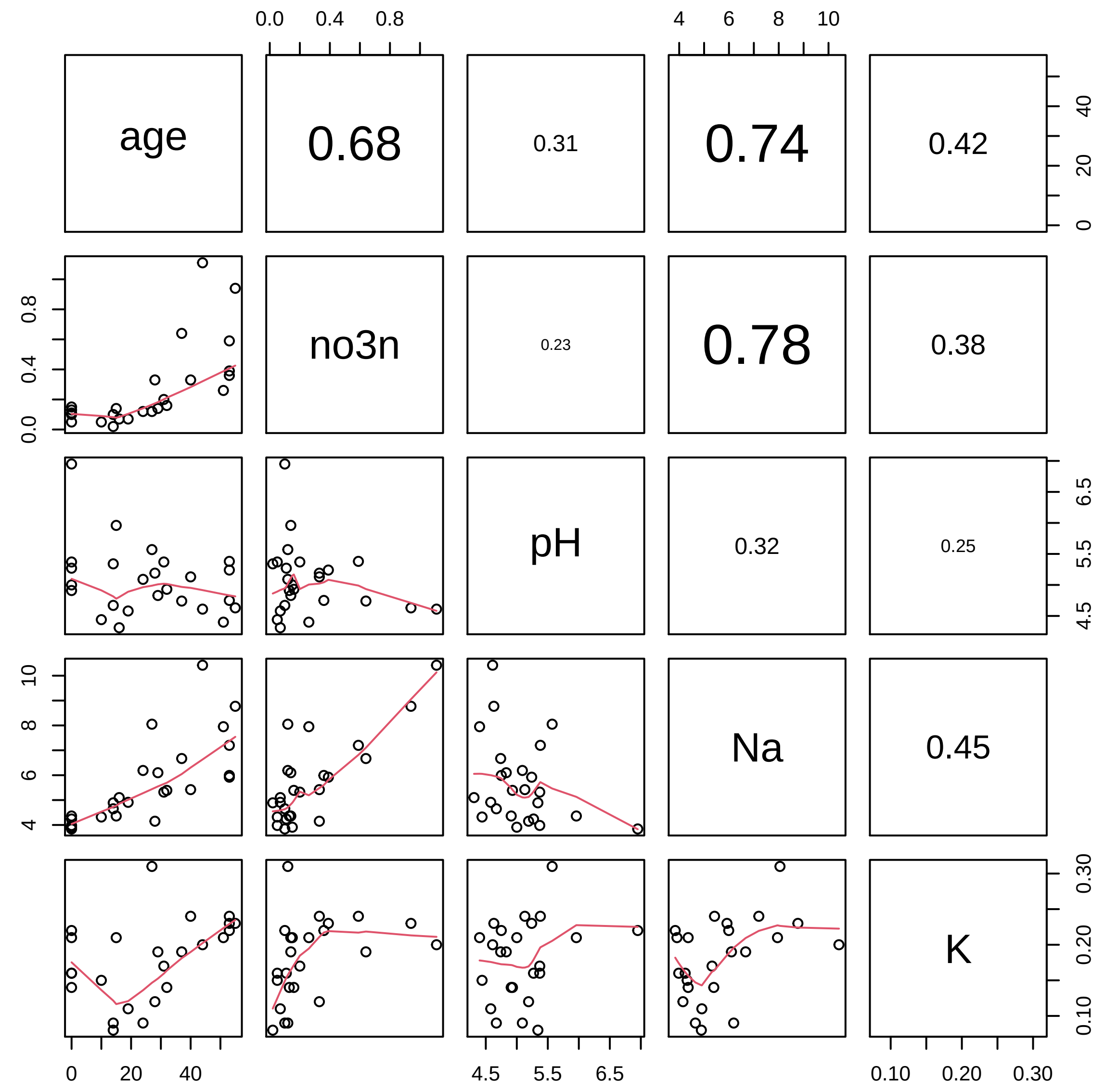


Figure 16.10: *K has now gotten a higher correlation with each variable because the concentration in th 10th observation has been set to missing value. Notice the other correlations have not changed, only K.*

If we have more than one relationship within a data set, we can separate them as done below. There is a classification with the factors levels: Across and Along. In Figure 16.11 you can see how you can identify the observation within harrowing Along and Across

Code to produce Figure 16.11:

```
par(mar=c(3.2,3.2,.5,.5), mgp=c(2,.7,0))
plot(Leaf~Intensity, data = leafCover, bty="l",
     pch = as.numeric(as.factor(Direction)),
     xlab="Harrow intensity", ylab="Leaf area")
legend("bottomleft", levels(factor(leafCover$Direction)), pch=c(1,2))
```

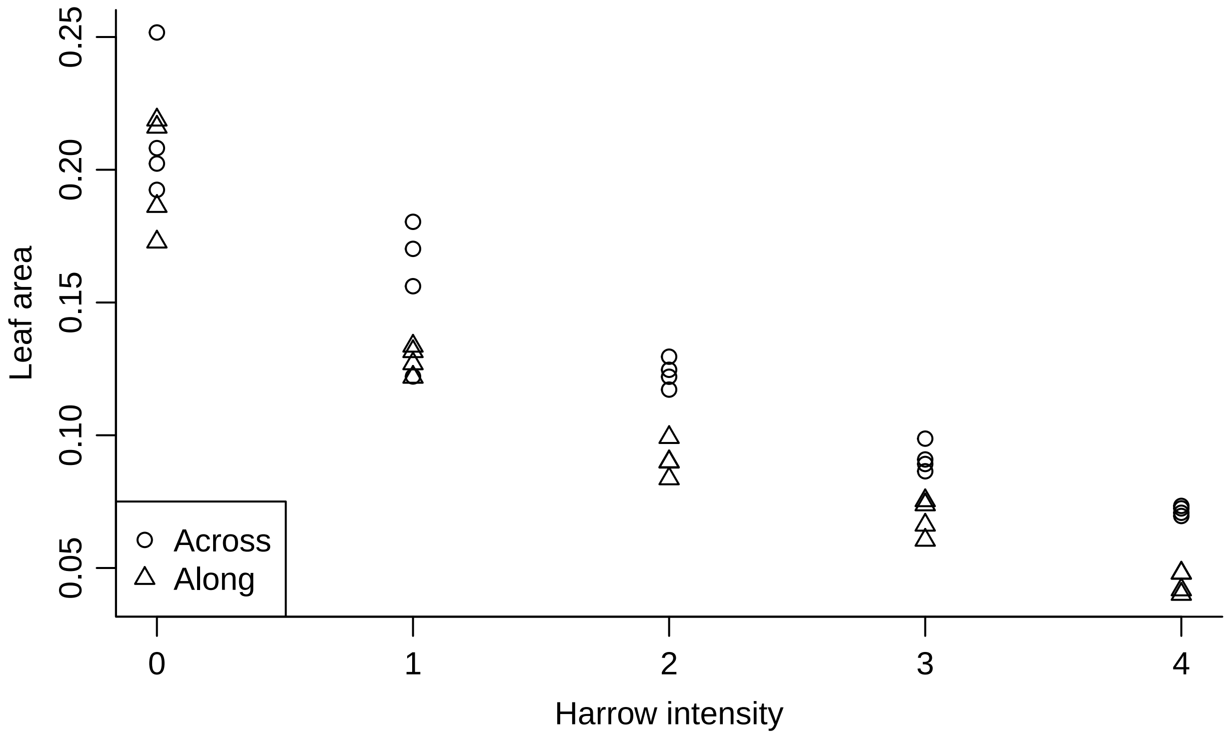


Figure 16.11: *The leaf area of weeds and crop in relation of the intensity of harrowing either across or along the crop rows. Note how we changed Across and Along to numerical values by using the function `as.numeric()` in order to get different symbols.*

Obviously, there is a relationship, and this will be properly addressed in the ANCOVA chapter. There are also other plotting facilities that can be used the `xyplot()` in the package,lattice, which comes with basic R.

Code to produce Figure 16.12

```
library(lattice)
xyplot(Leaf ~ Intensity | Direction, data = leafCover)
```

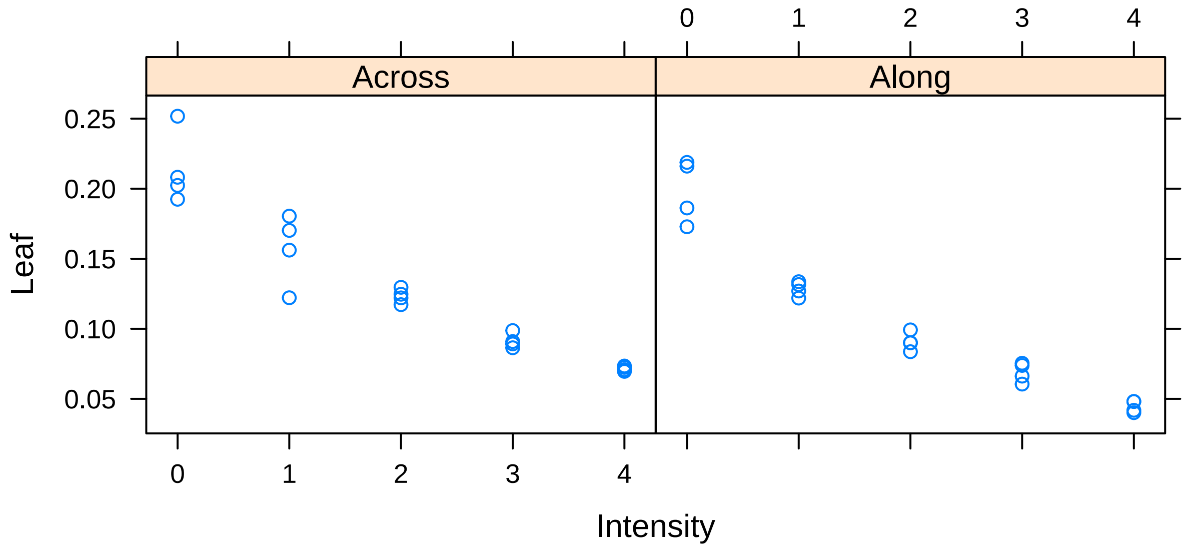


Figure 16.12: *The individual plots for each classification variable. Notice that the variable after the* | *defines the classification.*

16.2 ggplot2

Within the R community there is an array of versatile packages to visualize data, we have barely scratched the surface showing a few examples we felt may be of immediate interest. The numbers of graphic interfaces are numerous and increasing, some are easier to use than others, and sometimes it takes a rather large amount of time to get the hang of it. One package that has received a great deal of (much-deserved) praise is `ggplot2`. The 'gg' branding is "the grammar of graphics."

We will not spend too much time discussing the syntax of `ggplot2`, but we will provide several examples of plots created in this package to illustrate its powerful and elegant capabilities. We can begin by using the same `leafCover` data in Figures 16.11 and 16.12. In the 'gg' way of doing things, we first provide the data set and the primary variables of interest, then continue adding on additional layers to the plot, separated by a `+`.

Code to produce Figure16.13

```
library(ggplot2)

ggplot(leafCover, aes(x=Intensity, y=Leaf)) +
  geom_point(aes(color=Direction, shape=Direction)) +
  stat_smooth(aes(color=Direction, linetype=Direction), method="lm", se=TRUE) +
  ylab("Leaf area") + xlab("Harrow intensity") +
  theme_classic() + theme(legend.position="right")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

One of the strengths of the `ggplot2` package is the ability to use linear or local regression models using the `stat_smooth()` function, as shown in Figure 16.13. In this example, the linear model fit is drawn in the

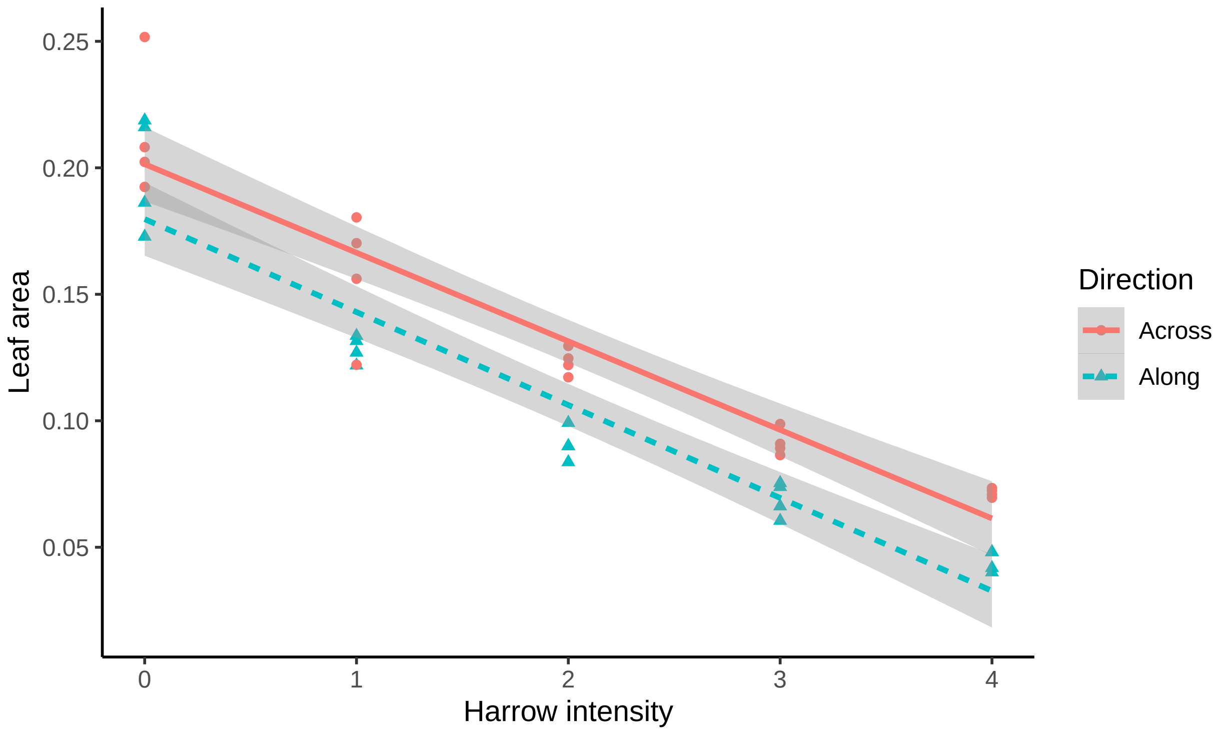


Figure 16.13: *Harrowing intensity data shown using ggplot2.*

same colors as the points, and a gray shaded area is added that represents the standard error. All of these options can be customized (for example, by changing to `se=FALSE` to remove the error shading).

To create a barplot with confidence intervals from a one way analysis of variance, ANOVA, we can again use the `InsectSprays` data set, and start by generating the linear model. The next step is to turn the relevant results from the ANOVA into a data.frame, for use in plotting, then passing that information along to the `geom_bar()` and `geom_errorbar()` functions within `ggplot`.

```
m1<-lm(count~spray-1,data=InsectSprays)
#Note the -1 presents the parameteres as sheer means not as differences
coef(m1) #the means
```

```
##    sprayA    sprayB    sprayC    sprayD    sprayE    sprayF
## 14.500000 15.333333  2.083333  4.916667  3.500000 16.666667
```

```
confint(m1) #the conficence intervals
```

```
##             2.5 %    97.5 %
## sprayA 12.2395786 16.760421
## sprayB 13.0729119 17.593755
## sprayC -0.1770881  4.343755
## sprayD  2.6562453  7.177088
## sprayE  1.2395786  5.760421
## sprayF 14.4062453 18.927088
```

```
d.frame.m1<-data.frame(cbind(coef(m1),confint(m1)),pest=rownames(confint(m1))) #Combining the two
head(d.frame.m1,2)
```

```
##              V1   X2.5..   X97.5..   pest
## sprayA 14.50000 12.23958 16.76042 sprayA
## sprayB 15.33333 13.07291 17.59375 sprayB
```

The results are provided in Figure 16.14. When doing an ANOVA to separate treatment effect it is not correct to give the standard error of the replicates within a treatment. It is based upon too few observations and essentially is a waste of ink. The only correct way is to use the ANOVA standard error. The trained weed and crop scientists will know this rather simple use of ANOVA results is not obeyed in a large proportion of papers with tables that give the individual treatments standard errors. But because this redundant presentation of data is generally accepted in most journals, whatever their impact factors, it should be discouraged.

Code to produce Figure16.14

```
ggplot(d.frame.m1, aes(x=pest, y=V1)) +
  geom_bar(stat="identity", fill="lightgreen",
           colour="black") +
  geom_errorbar(aes(ymin=pmax(X2.5..,0),
                    ymax=X97.5..), width=0.5) +
  xlab("Pesticide Spray")+
  ylab("Insect Counts")
```

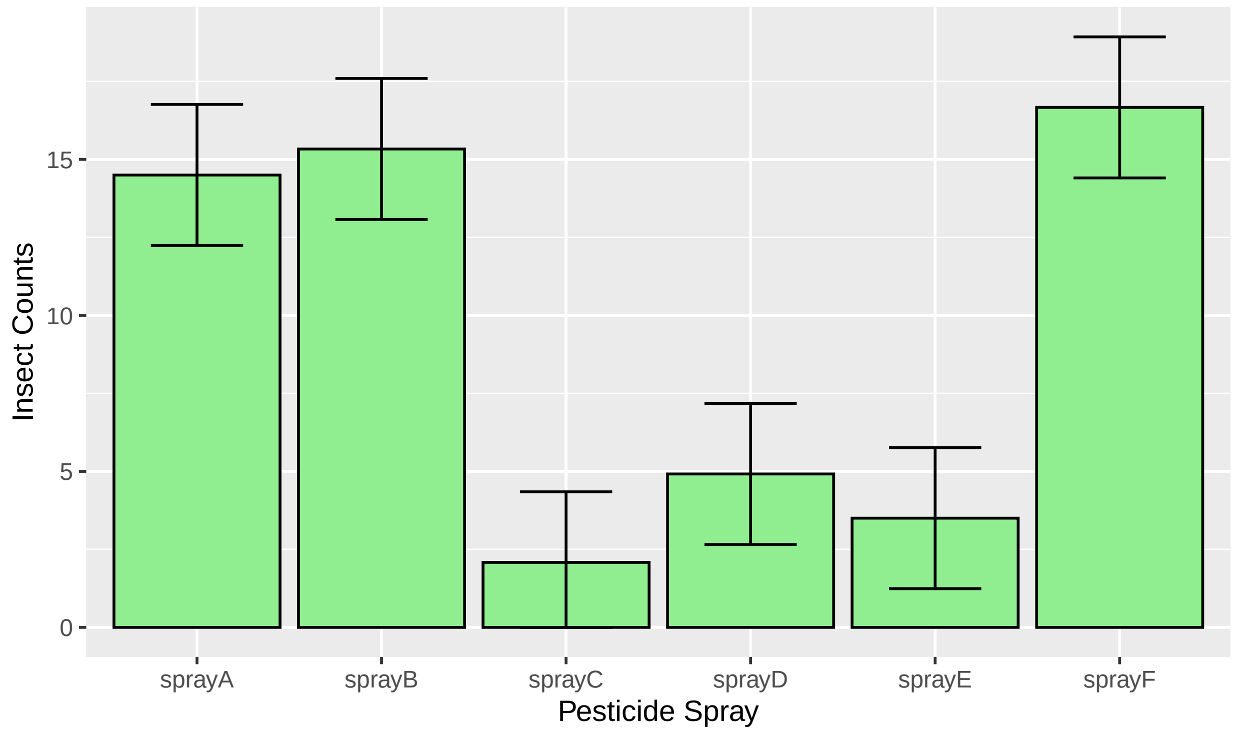


Figure 16.14: *InsectSpray data in a barplot with 95 % confidence intervals using the ggplot2 package.*

Paired barplots are another (more complicated, but often useful) way of arranging barplots. This is relevant for some factorial or split-plot designs. For example using data from a study looking at five crop species each grown in two different row orientations (east-west vs north-south), it makes sense to group the data by crops to look for consistent trends of row orientation (Figure 16.15).

Code to produce Figure16.15

```r
CropRowOrientation<-read.csv("http://rstats4ag.org/data/CropRowOrientation.csv")
head(CropRowOrientation)
```

```
##          Crop         Row Yield
## 1      Barley   East-west  1149
## 2       Wheat   East-west  1195
## 3      Canola   East-west   626
## 4   Field pea   East-west   500
## 5      Lupine   East-west   493
## 6      Barley North-south   856
```

```r
c <- ggplot(CropRowOrientation)
c + aes(x = Crop, y = Yield, fill = Row, color = Row) +
      geom_bar(stat="identity", width=.5, position="dodge")
```

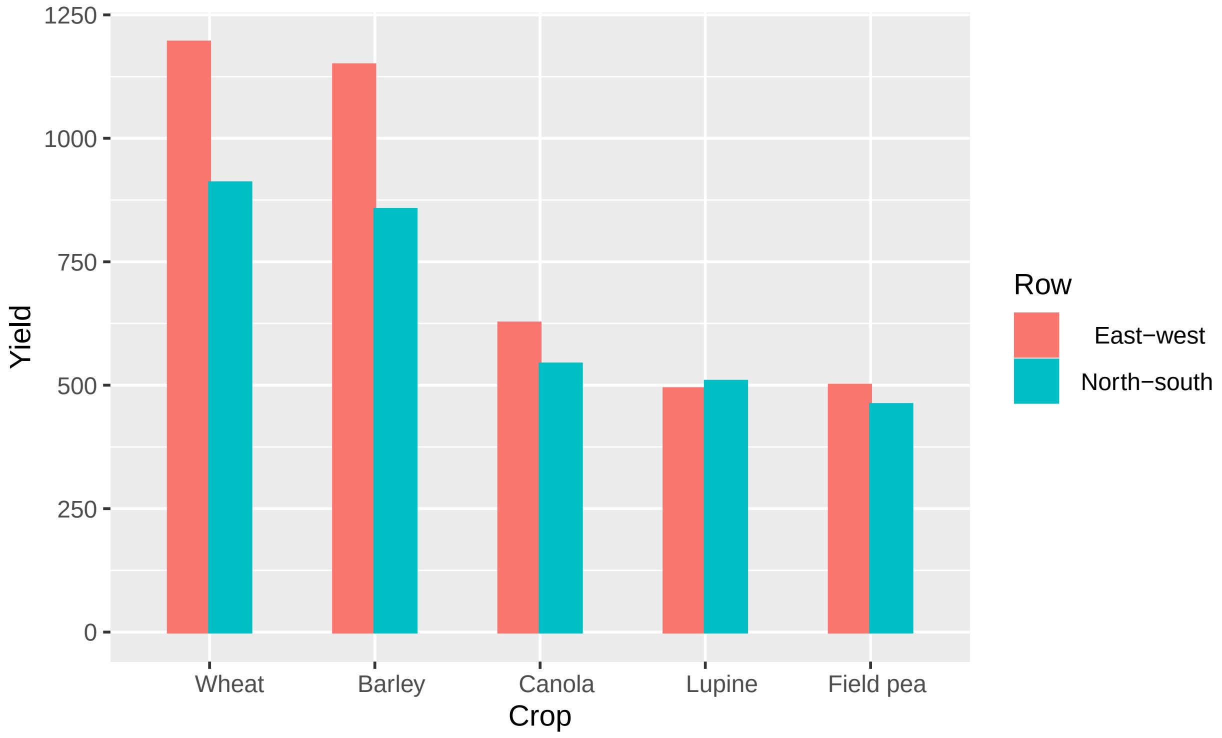


Figure 16.15: The effect of the orientation of crop sowing on the yield five crops. The illustration is based on means not raw data, so there are no error bars.