Tutorial: Using the R Environment for Statistical Computing
An example with the Mercer & Hall wheat yield dataset

D G Rossiter
University of Twente, Faculty of Geo-Information Science & Earth Observation (ITC)
Enschede (NL)

January 9, 2014

Contents

1 Introduction

This tutorial introduces the **R environment for statistical computing and visualisation** [21, 38] and its dialect of the **S language**. It is organized as a systematic analysis of a simple dataset: the Mercer & Hall wheat yield uniformity trial (Appendix A). After completing the tutorial you should:

- know the basics of the R environment;
- be able to use R at a beginning to intermediate level;
- follow a systematic approach to analyze a simple dataset.

The tutorial is organized as a set of **tasks** followed by **questions** to check your understanding; **answers** are at the end of each section. If you are ambitious, there are also some challenges: tasks and questions with no solution provided, that require the integration of skills learned in the section.

Not every section is of equal importance; you should pick and choose those of interest to you. Sections marked with an asterisk '*' are interesting "detours" or perhaps "scenic byways" developed to satisfy the author's curiosity or to answer a reader's question.

> **Note:** For an explanation of the R project, including how to **obtain and install** the software and documentation, see Rossiter [40]. This also contains an extensive discussion of the **S language**, **R graphics**, and many statistical methods, as well as a **bibliography** of texts and references that use R.

2 R basics

Before entering into the sample data analysis (§3), we first explain how to interact with R, and the basics of the S language.

Task 1 : Start R. •

How you do this is system-specific. Under Microsoft Windows, open the executable program `RGui.exe`; under Mac OS/X, open the application program `R.app`. Rossiter [40, §3] explains how to run the ITC network installation of R.

> **Note:** There are also some integrated development environments (IDE) for R, along with a source-code editor, object browser, help text, and graphics display. One that runs on Microsoft Windows, OS/X and Linux is RStudio[1].

After starting R, you will be looking at a **console** where you interact with R: giving commands and seeing numerical results; graphs are displayed in their own windows. You perform most actions in R by typing **commands** in response to a **command prompt**, which usually looks like this:

```
>
```

[1] http://rstudio.org/; there is a complete list of code editors and IDE's at http://www.sciviews.org/_rgui/projects/Editors.html

The `>` is a **prompt symbol** displayed by R, **not** typed by you. This is R's way of telling you it's waiting for you to enter a command.

Type your command[2] and press the `Enter` or `Return` keys; R will **execute** (carry out) your command.

Sometimes the command will result in numerical output listed on the console, other times in a graph displayed in a separate window, other times R will just do what you requested without any feedback.

If your entry is not a complete R command, R will prompt you to complete it with the **continuation prompt symbol**:

```
+
```

R will accept the command once it is **syntactically complete**; in particular any parentheses must balance. Once the command is complete, R will execute it.

Several commands can be given on the same line, separated by `;`. A command may be interrupted by pressing the `Esc` key.

To illustrate this interaction, we draw a sample of random numbers from the uniform probability distribution; this is a simple example of R's simulation repertoire.

> **Note:** The code in these exercises was tested with Sweave [27, 28] on R version 3.0.1 (2013-05-16), `sp` package Version: 1.0-9, `gstat` package Version: 1.0-16, and `lattice` package Version: 0.20-15 running on Mac OS X 10.7.5. The text and graphical output you see here was automatically generated and incorporated into LaTeX by running R source code through R and its packages. Then the LaTeX document was compiled into the PDF version you are now reading. Your output may be slightly different on different versions and on different platforms.

Task 2 : Draw 12 random numbers uniformly distributed from -1 to 1, rounded to two decimal places, and sort them from smallest to largest. •

In this tutorial we show the **R code** like this, with the prompt `>` and then then command:

```
> sort(round(runif(12, -1, 1), 2))
```

Note that the prompt `>` is *not* part of the command typed by the user; it is presented by the R console.

We show the **output** printed by R like this:

```
 [1] -0.87 -0.65 -0.57 -0.49 -0.30 -0.25 -0.22 -0.19  0.13  0.17  0.28
[12]  0.81
```

The numbers in brackets, like `[1]`, refer to the position in the output vector. In the example above, the 12^{th} element is `0.81`.

[2] or cut-and-paste from a document such as this one

This first example already illustrates several features of R:

1. It includes a large number of **functions** (here, `runif` to generate **r**andom numbers from the **unif**orm distribution; `round` to **round** them to a specified precision; and `sort` to **sort** them);
2. These functions have **arguments** that specify the exact behaviour of the function. For example, `round` has two arguments: the first is the object to be rounded (here, the vector returned by the `runif` function) and the second is the number of decimal places (here, `2`);
3. Many functions are **vectorized**: they can work on vectors (and usually matrices) as well as scalars. Here the `round` function is modifying the results of the `runif` function, which is a 12-element vector;
4. Values **returned** by a function can be immediately used as an argument to another function. Here the results of `runif` is the vector to be rounded by the `round` function; and these are then used by the `sort` function. To understand a complex expression, read it from the inside out.
5. R has a rich set of functions for **simulation** of **random processes**.

Q1 : *Your results will be different from the ones printed in this note; why?* *Jump to A1* •

To see how this works, we can do the same operation step-by-step.

1. Draw the random sample, and save it in a **local variable** in the **workspace** using the `<-` (assignment) operator; we also list it on the console with the `print` function:

```
> sample <- runif(12, -1, 1)
> print(sample)

 [1] -0.20929  0.90818  0.30273  0.88772 -0.61184 -0.54945  0.44606
 [8]  0.28284  0.62387 -0.80595 -0.90955  0.88406
```

2. Round it to two decimal places, storing it in the same variable (i.e. replacing the original sample):

```
> sample <- round(sample, 2)
> sample

 [1] -0.21  0.91  0.30  0.89 -0.61 -0.55  0.45  0.28  0.62 -0.81 -0.91
[12]  0.88
```

3. Sort it and print the results:

```
> (sample <- sort(sample))

 [1] -0.91 -0.81 -0.61 -0.55 -0.21  0.28  0.30  0.45  0.62  0.88  0.89
[12]  0.91
```

This example also shows three ways of printing R output on the console:

- By using the `print` function with the object name as argument;
- By simply typing the object name; this calls the `print` function;
- By enclosing any expression in parenthesis `( ... )`; this forces another evaluation, which prints its results.

R has an immense repertoire of **statistical methods**; let's see two of the most basic.

Task 3 : Compute the theoretical and empirical mean and variance of a sample of 20 observations from a uniformly-distributed random variable in the range $(0 \ldots 10)$, and compare them. •

The theoretical mean and variance of a uniformly-distributed random variable are [6, §3.3]:

$$\begin{aligned} \mu &= (b+a)/2 \\ \sigma^2 &= (b-a)^2/12 \end{aligned}$$

where a and b are the lower and upper endpoints, respectively, of the uniform interval.

First the theoretical values for the mean and variance. Although we could compute these by hand, it's instructive to see how R can be used as an **interactive calculator** with the usual operators such as `+`, `-`, `*`, `/`, and `^` (for exponentiation):

```
> (10 + 0)/2

[1] 5

> (10 - 0)^2/12

[1] 8.3333
```

Now draw a 20-element sample and compute the sample mean and variance, using the `mean` and `var` functions:

```
> sample <- runif(20, min = 0, max = 10)
> mean(sample)

[1] 4.57

> var(sample)

[1] 11.281
```

Q2 : *How close did your sample come to the theoretical value?* *Jump to A2* •

We are done with the local variable `sample`, so we **remove** it from the workspace with the `rm` ("**rm**ove") function; we can check the contents of the workspace with the `ls` ("**ls**t") function:

```
> ls()

[1] "sample"

> rm(sample)
> ls()

character(0)
```

On-line help If you know a function or function's name, you can get **help** on it with the `help` function:

```
> help(round)
```

This can also be written more simply as `?round`.

Q3 : *Use the `help` function to find out the three arguments to the `runif` function. What are these? Are they all required? Does the order matter?*
Jump to A3 •

Arguments to functions We can experiment a bit to see the effect of changing the arguments:

```
> runif(1)

[1] 0.81942

> sort(runif(12))

 [1] 0.054629 0.179545 0.221020 0.287711 0.379869 0.509107 0.509446
 [8] 0.638026 0.644253 0.803661 0.870994 0.884329

> sort(runif(12, 0, 5))

 [1] 0.99097 1.25497 1.44115 2.10537 2.93438 3.35891 3.40965 3.41952
 [9] 3.65709 3.68396 3.92879 4.03538

> sort(runif(12, min = 0, max = 5))

 [1] 0.0017379 0.3919713 0.3960378 0.4537320 0.4949257 0.5339417
 [7] 0.6730482 1.0922637 2.4902907 2.7513814 2.9691217 4.1510191

> sort(runif(max = 5, n = 12, min = 0))

 [1] 0.77907 1.47624 1.93221 2.36229 2.79788 3.29575 4.04648 4.08959
 [9] 4.36135 4.64995 4.78142 4.92621
```

Searching for a function If you don't know a function name, but you know what you want to accomplish, you can search for an appropriate function with the `help.search` function:

```
> help.search("principal component")
```

This will show packages and functions relevant to the topic:

```
stats::biplot.princomp   Biplot for Principal Components
stats::prcomp            Principal Components Analysis
stats::princomp          Principal Components Analysis
stats::summary.princomp  Summary method for Principal Components Analysis
```

Then you can ask for more information on one of these, e.g.:

```
> help(prcomp)

prcomp            package:stats                    R Documentation

Principal Components Analysis

Description:

Performs a principal components analysis on the given data matrix
and returns the results as an object of class 'prcomp'.

Usage: ...
```

2.1 Leaving R

At this point you should leave R and re-start it, to see how that's done.

Before leaving R, you may want to **save your console log** as a text file to document what you did, and the results, or for later re-use. You can edit this file in any plain-text editor, or include in a report,

To leave R, use the `q` ("quit") function; if you are running R with a GUI, you can use a menu command, or a "close" icon as in any GUI program.

```
> q()
```

You will be asked if you want to **save your workspace** in the current directory; generally you will want to do this[3]. The next time you start R in the same directory, the saved workspace will be automatically loaded.

In this case we haven't created anything useful for data analysis, so you should quit without saving the workspace.

2.2 Answers

A1 : *Random number generation gives a different result each time.*[4]. *Return to Q1* •

A2 : *This depends on your sample; see the results in the text for an example.* *Return to Q2* •

A3 : *There are three possible arguments: the number of samples* `n`*, the minimum*

[3] By default this file is named `.RData`

[4] To start a simulation at the same point (e.g. for testing) use the `set.seed` function

value `min` *and the maximum* `max`*. The last two are not required and* ***default*** *to* 0 *and* 1*, respectively. If arguments are* ***named*** *directly, they can be put in any order. If not, they have to follow the default order.* *Return to Q3* •

3 Loading and examining a data set

The remainder of this tutorial uses the Mercer & Hall wheat yield data set, which is described in Appendix A. Please read this now.

There are many ways to get data into R [40, §6]; one of the simplest is to create a **comma-separated values** ("CSV") file in a text editor[5]. For this example we have prepared file `mhw.csv` which is supplied with this note.

Task 4 : From the operating system, open the text file `mhw.csv` with a plain-text editor such as WordPad and examine its structure.

Note: *do not* examine it in Excel; this automatically splits the file into spreadsheet columns, obscuring its structure as a text file. •

The first four lines of the file should look like this:

```
"r","c","grain","straw"
1,1,3.63,6.37
2,1,4.07,6.24
3,1,4.51,7.05
```

Q4 : *What does the first line represent? What do the other lines represent, and what is their structure?* *Jump to A4* •

3.1 Reading a CSV file into an R object

Now we read the dataset into R.

Task 5 : Start R. •

Task 6 : If necessary, make sure R is pointed to the same **working directory** where you have stored `mhw.csv`. You can use the `getwd` function to check this, and `setwd` to change it; in the Windows GUI you can use the `File|Change directory...` menu command. •

```
> getwd()
```

Once the directory is changed, the contents of the file can be displayed with the `file.show` function:

```
> file.show("mhw.csv")
```

[5] A CSV file can also be prepared as a spreadsheet and exported to CSV format.

```
"r","c","grain","straw"
1,1,3.63,6.37
2,1,4.07,6.24
3,1,4.51,7.05
4,1,3.9,6.91
...
```

A CSV file can be read into R with the `read.csv` function and **assigned** to an object in the **workspace** using the `<-` **operator** (which can also be written as `=`):

```
> mhw <- read.csv("mhw.csv")
```

Q5 : *Why is nothing printed after this command?* *Jump to A5* •

3.2 Examining a dataset

The first thing to do with any dataset is to examine its **structure** with the `str` function.

```
> str(mhw)

'data.frame':        500 obs. of  4 variables:
 $ r    : int  1 2 3 4 5 6 7 8 9 10 ...
 $ c    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ grain: num  3.63 4.07 4.51 3.9 3.63 3.16 3.18 3.42 3.97 3.4 ...
 $ straw: num  6.37 6.24 7.05 6.91 5.93 5.59 5.32 5.52 6.03 5.66 ...
```

Q6 : *How many observations (cases) are there in this frame? How many fields (variables)? What are the field names?* *Jump to A6* •

We can extract the names for each field (matrix column) with the `names` function; this is equivalent to `colnames`:

```
> names(mhw)

[1] "r"     "c"     "grain" "straw"

> colnames(mhw)

[1] "r"     "c"     "grain" "straw"
```

Every object in R belongs to a **class**, which R uses to decide how to carry out commands.

Q7 : *What is the **class** of this object?* *Jump to A7* •

We can examine the class with the `class` function:

```
> class(mhw)

[1] "data.frame"
```

A **data frame** is used to hold most data sets. The **matrix rows** are the **observations** or **cases**; the **matrix columns** are the named **fields** or **variables**. Both matrix rows and columns have **names**.

Fields in the data frame are commonly referred to by their matrix column name, using the syntax `frame$variable`, which can be read as "extract the field named `variable` from the data frame named `frame`.

Task 7 : Summarize the grain and straw yields. •

```
> summary(mhw$grain)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.73    3.64    3.94    3.95    4.27    5.16

> summary(mhw$straw)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   4.10    5.88    6.36    6.51    7.17    8.85
```

A data frame is also a **matrix**; we can see this by examining its **dimensions** with the `dim` function and extracting elements.

The two dimensions are the numbers of matrix rows and columns:

```
> dim(mhw)

[1] 500   4
```

Q8 : *Which matrix dimension corresponds to the observations and which to the fields?* *Jump to A8* •

Matrix rows, columns, and individual cells in the matrix can be extracted with the `[]` operator; this is just like standard matrix notation in mathematics:

```
> mhw[1, ]

  r c grain straw
1 1 1  3.63  6.37

> length(mhw[, 3])

[1] 500

> summary(mhw[, 3])

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.73    3.64    3.94    3.95    4.27    5.16

> mhw[1, 3]

[1] 3.63
```

Matrix rows and columns can also be accessed by their **names**; here is the grain yield of the first plot:

```
> mhw[1, "grain"]

[1] 3.63
```

Q9 : *What is the grain yield of plot 64? Where is this located in the (experimental) field?* *Jump to A9* •

```
> mhw[64, "grain"]

[1] 4.04

> mhw[64, c("r", "c")]

   r c
64 4 4
```

Note the use of the `c` ("catenate", Latin for 'build a chain') function to build a **list** of two names.

Several adjacent rows or columns can be specified with the `:` "sequence" operator. For example, to show the row and column in the wheat field for the first three records:

```
> mhw[1:3, 1:2]

  r c
1 1 1
2 2 1
3 3 1
```

Rows or columns can be omitted with the `-` "minus" operator; this is shorthand for "leave these out, show the rest". For example to summarize the grain yields for all except the first field column[6]:

```
> summary(mhw[-(1:20), "grain"])

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.73    3.64    3.94    3.95    4.27    5.16

> summary(mhw$grain[-(1:20)])

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.73    3.64    3.94    3.95    4.27    5.16
```

An entire field (variable) can be accessed either by matrix column number or name (considering the object to be a matrix) or variable name (considering the object to be a data frame); the output can be limited to the first and last lines only by using the `head` and `tail` functions. By default they show the six first or last values; this can be over-ridden with the optional `n` argument.

[6] recall, the dataset is presented in field column-major order, and there are 20 field rows per field column

```
> head(mhw[, 3])

[1] 3.63 4.07 4.51 3.90 3.63 3.16

> tail(mhw[, "grain"], n = 10)

 [1] 3.29 3.83 4.33 3.93 3.38 3.63 4.06 3.67 4.19 3.36

> head(mhw$grain)

[1] 3.63 4.07 4.51 3.90 3.63 3.16
```

The `order` function is somewhat like the `sort` function shown above, but rather than return the actual values, it returns their position in the array. This position can then be used to extract other information from the data frame.

Task 8 : Display the information for the plots with the five *lowest* straw yields. •

To restrict the results to only five, we again use the `head` function.

```
> head(sort(mhw$straw), n = 5)

[1] 4.10 4.28 4.53 4.56 4.57

> head(order(mhw$straw), n = 5)

[1] 470 467 441 447 427

> head(mhw[order(mhw$straw), ], n = 5)

     r  c grain straw
470 10 24  2.84  4.10
467  7 24  2.78  4.28
441  1 23  2.97  4.53
447  7 23  3.44  4.56
427  7 22  3.05  4.57
```

Q10 : *What are the values shown in the first command, using* `sort`*? In the second, using* `order`*? Why must we use the results of* `order` *to extract the records in the data frame?* *Jump to A10* •

Task 9 : Display the information for the plots with the *highest* straw yields. •

One way is to use the `rev` command to reverse the results of the `sort` or `order` function:

```
> head(rev(sort(mhw$straw)), n = 5)

[1] 8.85 8.85 8.78 8.75 8.74
```

Another way is to use the optional `decreasing` argument to `sort` or `order`; by default this has the value `FALSE` (so the sort is ascending); by setting it to `TRUE` the sort will be descending:

```
> head(sort(mhw$straw, decreasing = T), n = 5)

[1] 8.85 8.85 8.78 8.75 8.74
```

And a final way is to display the end of the ascending order vector, instead of the beginning, with the `tail` function; however, this shows the last records but still in ascending order:

```
> tail(sort(mhw$straw), n = 5)

[1] 8.74 8.75 8.78 8.85 8.85
```

Records can also be selected with **logical criteria**, for example with **numeric comparison operators**.

Task 10 : Identify the plots with the highest and lowest grain yields and show their location in the field and both yields. •

There are two ways to do this. First, apply the `max` and `min` functions to the grain yield field, and use their values (i.e., the highest and lowest yields) as a row selector, along with the `==` "numerical equality" comparison operator.

We save the returned value (i.e., the row number where the maximum or minimum is found), and then use this as the row subscript selector:

```
> (ix <- which(mhw$grain == max(mhw$grain)))

[1] 79

> mhw[ix, ]

    r c grain straw
79 19 4  5.16  8.78

> (ix <- which(mhw$grain == min(mhw$grain)))

[1] 338

> mhw[ix, ]

     r  c grain straw
338 18 17  2.73  4.77
```

The easier way, in the case of the minimum or maximum, is to use the `which.max` (index of the maximum value in a vector) and `which.min` (index of the minimum value in a vector) function

```
> (ix <- which.max(mhw$grain))

[1] 79

> mhw[ix, ]
```

```
    r c grain straw
79 19 4  5.16  8.78

> (ix <- which.min(mhw$grain))

[1] 338

> mhw[ix, ]

     r  c grain straw
338 18 17  2.73  4.77
```

Q11 : *Why is there nothing between the comma ',' and right bracket ']' in the expressions* `mhw[ix, ]` *above?* *Jump to A11* •

The advantage of the first method is that `==` or other numeric comparison operators can be used to select; operators include `!=` (not equal), `<`, `>`, `<=` (≤), and `>=` (≥). For example:

Task 11 : Display the records for the plots with straw yield > 8.8 lb. per plot. •

```
> mhw[which(mhw$straw > 8.8), ]

    r c grain straw
15 15 1  3.46  8.85
98 18 5  4.84  8.85
```

Challenge: Extract all the grain yields from the most easterly (highest-numbered) column of field plots, along with the straw yields and field plot row number. Sort them from highest to lowest yields, also displaying the row numbers and straw yields. Does there seem to be any trend by field plot row? How closely are the decreasing grain yields matched by straw yields?

3.3 Saving a dataset in R format

Once a dataset has been read into R and possibly modified (for example, by assigning field names, changing the class of some fields, or computing new fields) it can be saved in R's internal format, using the `save` function. The dataset can then be read into the workspace in a future session with the `load` function.

Task 12 : Save the `mhw` object in R format. •

It is conventional to give files with R objects the `.RData` extension.

```
> save(mhw, file = "mhw.RData")
```

3.4 Answers

A4 : *The first line is a **header** with the **variable names**, in this case* `r`, `c`, `grain` *and* `straw`*. The following lines each represent one **plot**; there are four variables recorded for each plot, i.e. its row and column number in the field, and its grain and straw yield.* *Return to Q4* •

A5 : *Commands that store their results in an object (using the* `=` *or* `<-` *operators) do their work silently; if you want to see the results enclose the command in parentheses* `( ... )` *or just type the object name at the command prompt.* *Return to Q5* •

A6 : *There are 500 observations (cases), and for each 4 variables:* `r`, `c`, `grain` *and* `straw`*.* *Return to Q6* •

A7 : *It is in class* `data.frame`*.* *Return to Q7* •

A8 : *Matrix rows are observations, matrix columns are fields.* *Return to Q8* •

A9 : *Grain yield 4.04; this is located at field row 4, field column 4* *Return to Q9* •

A10 : *The* `sort` *function shows the actual values of straw yield;* `order` *shows in which records in the data frame these are found. The record numbers are the key into the data frame.* *Return to Q10* •

A11 : *So that all fields (matrix columns) are selected.* *Return to Q11* •

4 Exploratory graphics

Before beginning a data analysis, it is helpful to **visualise** the dataset. This is generally the first phase of **exploratory data analysis** (EDA) [44].

R is an excellent environment for visualisation; it can produce simple plots but also plots of great sophistication, information and beauty. We look first at single variables and then at the relation between two variables.

4.1 Univariate exploratory graphics

Task 13 : Visualise the **frequency distribution** of grain yield with a stem plot. •

A **stem-and-leaf** plot, displayed by the `stem` function, shows the numerical values themselves, to some precision:

```
> stem(mhw$grain)

  The decimal point is 1 digit(s) to the left of the |

  27 | 38
  28 | 45
  29 | 279
  30 | 144555557899
  31 | 4446678999
  32 | 2345589999
  33 | 002455666677789999
  34 | 00112233444445667777777888999
  35 | 0111233444455566667778999 9
  36 | 000111113333344444566666777778889999
  37 | 0001111112222223334444455555666666777789999 9
  38 | 00112222233344444555666677779999 99
  39 | 011111111222223333344444455566666677777777 7999
  40 | 01112233334455566666667777777888889999999 9
  41 | 000111112233344555777779999
  42 | 0000111111122233334444466677777788999999
  43 | 011122333356666677777888899999 9
  44 | 00111112222344455666667777899
  45 | 0112222234445667888899
  46 | 1344446678899
  47 | 3356677
  48 | 466
  49 | 12349
  50 | 279
  51 | 3336
```

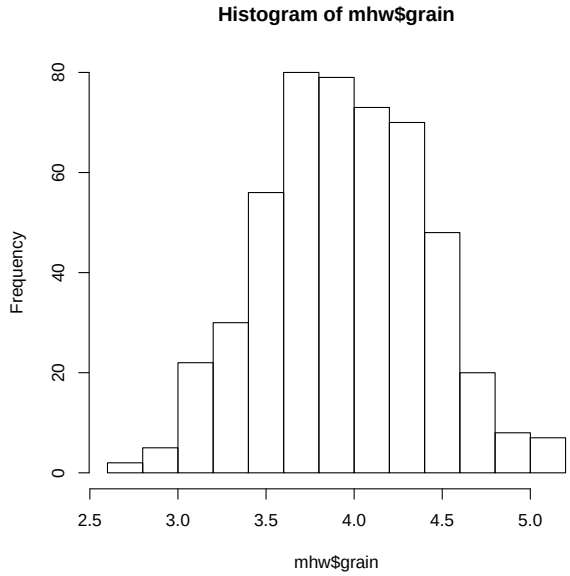
Q12 : *According to the stem-and-leaf plot, what are the approximate values of the minimum and maximum grain yields?* *Jump to A12* •

Q13 : *What is the advantage of the stem plot over the histogram?* *Jump to A13* •

Task 14 : Visualise the **frequency distribution** of grain yield with a frequency histogram. •

A histogram, displayed by the `hist` function, shows the distribution:

```
> hist(mhw$grain)
```



Saving graphic output

You can **save the graphics** window to any common graphics format. In the Windows GUI, bring the graphics window to the front (e.g. click on its title bar), select menu command `File | Save as ...` and then one of the formats.

Q14 : *What are the two axes of the default histogram?* *Jump to A14* •

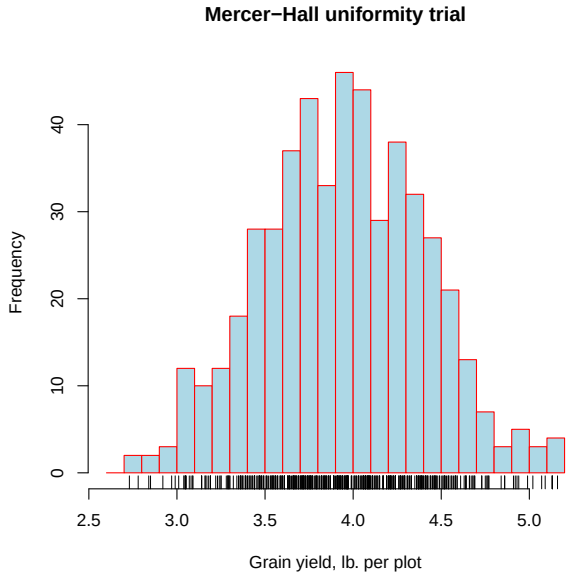
Q15 : *By examining the histogram, how many observations had grain yield below 3 lb. per plot?* *Jump to A15* •

4.1.1 Enhancing the histogram*

R graphics, including histograms, can be enhanced from their quite plain default appearance. Here we change the break points with the `breaks` argument, the colour of the bars with the `col` graphics argument, the colour of the border with the `border` graphics argument, and supply a title with the `main` graphics argument.

We then use the `rug` function to add a "rug" plot along the x-axis to show the actual observations. This is an example of a graphics function that **adds** to an existing plot; whereas `hist` creates a **new** plot. Which does which? Consult the help.

```
> hist(mhw$grain, breaks = seq(2.6, 5.2, by = 0.1), col = "lightblue",
+     border = "red", main = "Mercer-Hall uniformity trial",
+     xlab = "Grain yield, lb. per plot")
> rug(mhw$grain)
```



Note the use of the `seq` ("sequence") function to make a list of break points:

```
> seq(2.6, 5.2, by = 0.1)

 [1] 2.6 2.7 2.8 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1
[17] 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.0 5.1 5.2
```

In this example, the colours are from a list of known names. For more information on these names, and other ways to specify colours, see Appendix B.

4.1.2 Kernel density*

A **kernel density**, computed with `density` function, fits an empirical curve to a sample supposed to be drawn from a univariate probability distribution [46, §5.6]. It can be used to give a visual impression of the distribution or to **smooth** an empirical distribution.

In the context of EDA, the kernel density can suggest:

- whether the empirical distribution is **unimodal** or **multimodal**;
- in the case of a unimodal distribution, the **theoretical probability density function** from which it may have been drawn.

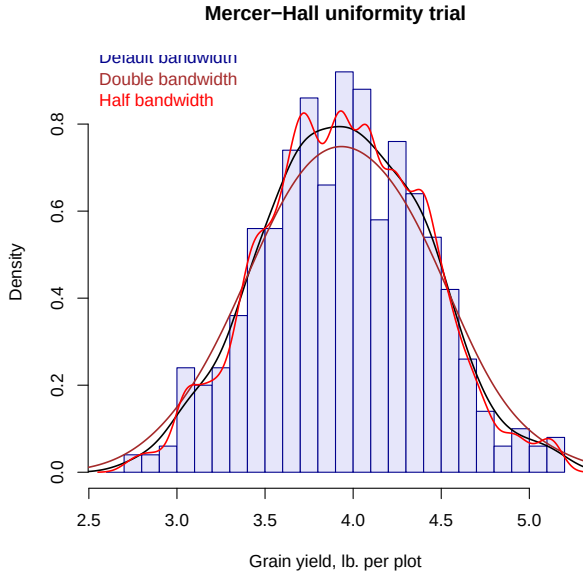
The kernel density is controlled by the `kernel` and `adjust` optional arguments to the `density` function; see `?density` for details. The default values of `"gaussian"` and `1` select a smoothing bandwidth based on the number of observations and a theoretical normal density.

A special case of the density is a histogram expressed as densities rather than frequencies; this is selected with the optional `freq` ("frequency") argument to the `hist` function set to `FALSE`. The total area under the histogram is then by definition `1`.

The `lines` function can be used to add the empirical density computed by `density` to a density histogram plotted with `hist`. Another interesting view is the kernel density with a rug plot to show the actual values of the sample.

Task 15 : Display a histogram of the grain yields as a density (proportion of the total), with the default kernel density superimposed, along with a double and half bandwidth kernel density. •

```
> hist(mhw$grain, breaks = seq(2.6, 5.2, by = 0.1), col = "lavender",
+     border = "darkblue", main = "Mercer-Hall uniformity trial",
+     freq = F, xlab = "Grain yield, lb. per plot")
> lines(density(mhw$grain), lwd = 1.5)
> lines(density(mhw$grain, adj = 2), lwd = 1.5, col = "brown")
> lines(density(mhw$grain, adj = 0.5), lwd = 1.5, col = "red")
> text(2.5, 0.95, "Default bandwidth", col = "darkblue",
+     pos = 4)
> text(2.5, 0.9, "Double bandwidth", col = "brown", pos = 4)
> text(2.5, 0.85, "Half bandwidth", col = "red", pos = 4)
```

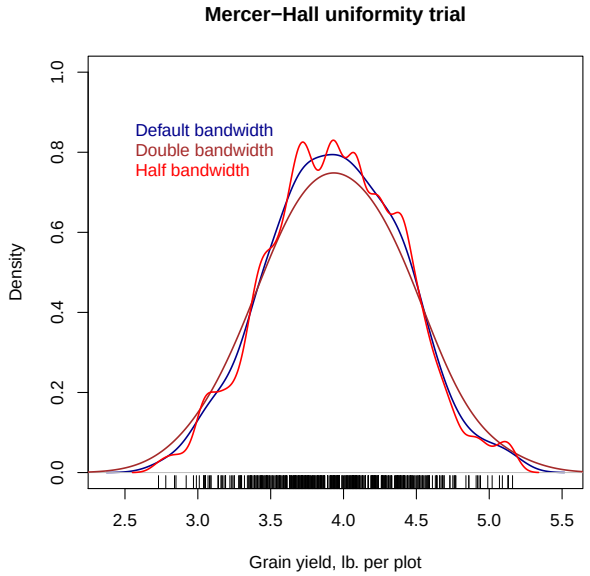


Task 16 : Repeat, but show just the kernel density with a rug plot (i.e. no histogram). •

Here the first plot must be of the density, because `rug` only adds to an existing plot.

```
> plot(density(mhw$grain) ,xlab="Grain yield, lb.\ per plot",
+      lwd=1.5, ylim=c(0,1), col="darkblue",
+      main="Mercer-Hall uniformity trial")
> rug(mhw$grain)
> lines(density(mhw$grain, adj=2), lwd=1.5, col="brown")
> lines(density(mhw$grain, adj=.5), lwd=1.5, col="red")
> text(2.5,0.85,"Default bandwidth", col="darkblue", pos=4)
```

```
> text(2.5,0.80,"Double bandwidth", col="brown", pos=4)
> text(2.5,0.75,"Half bandwidth", col="red", pos=4)
```



Q16 : *Which bandwidths give rougher or smoother curves? What does the curve for the default bandwidth suggest about the underlying distribution?* *Jump to A16* •

4.1.3 Another histogram enhancement: colour-coding relative frequency*

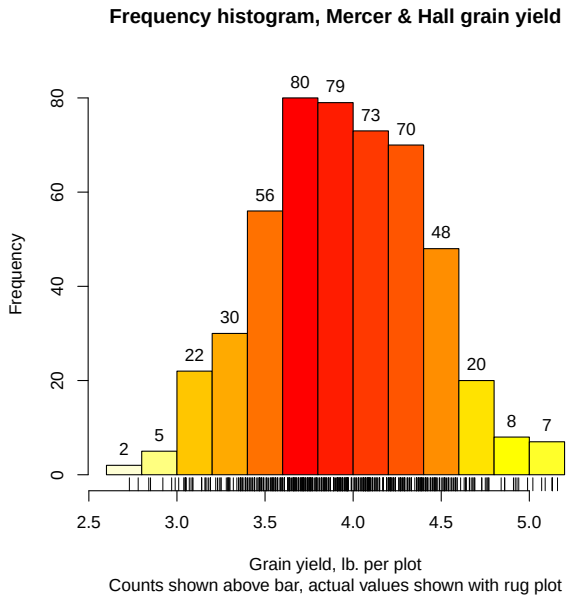
Task 17 : Display a histogram of the grain yield with break points every 0.2 lb., with the count in each histogram bin printed on the appropriate bar. Shade the bars according to their count, in a colour ramp with low counts whiter and high counts redder. •

The solution to this task depends on the fact that the `hist` function not only plots a histogram graph, it can also **return** an object which can be **assigned** to an object in the workspace; we can then examine the object to find the counts, breakpoints etc. We first compute the histogram but don't plot it (`plot=F` argument), then draw it with the `plot` command, specifying a colour ramp, which uses the computed counts, and a title. Then the `text` command adds text to the plot at `(x, y)` positions computed from the class mid-points and counts; the `pos=3` argument puts the text on top of the bar.

```
> h <- hist(mhw$grain, breaks = seq(2.6, 5.2, by = 0.2),
+     plot = F)
> str(h)

List of 6
 $ breaks  : num [1:14] 2.6 2.8 3 3.2 3.4 3.6 3.8 4 4.2 4.4 ...
 $ counts  : int [1:13] 2 5 22 30 56 80 79 73 70 48 ...
```

```
 $ density : num [1:13] 0.02 0.05 0.22 0.3 0.56 ...
 $ mids    : num [1:13] 2.7 2.9 3.1 3.3 3.5 3.7 3.9 4.1 4.3 4.5 ...
 $ xname   : chr "mhw$grain"
 $ equidist: logi TRUE
 - attr(*, "class")= chr "histogram"
```

```
> plot(h, col = heat.colors(length(h$mids))[length(h$count) -
+     rank(h$count) + 1], ylim = c(0, max(h$count) + 5),
+     main = "Frequency histogram, Mercer & Hall grain yield",
+     sub = "Counts shown above bar, actual values shown with rug plot",
+     xlab = "Grain yield, lb. per plot")
> rug(mhw$grain)
> text(h$mids, h$count, h$count, pos = 3)
> rm(h)
```



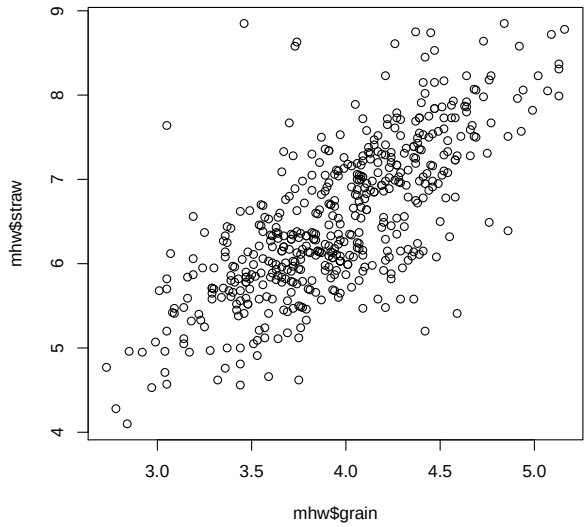
4.2 Bivariate exploratory graphics

When several variables have been collected, it is natural to compare them.

Task 18 : Display a **scatterplot** of straw vs. grain yield. •

We again use `plot`, but in this case there are two variables, so a scatterplot is produced. That is, `plot` is an example of a **generic** function: its behaviour changes according to the **class** of object it is asked to work on.

```
> plot(mhw$grain, mhw$straw)
```



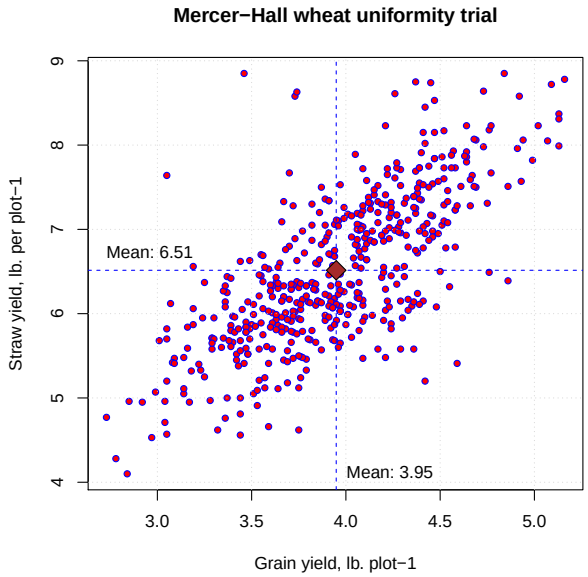
Q17 : *What is the relation between grain and straw yield?* *Jump to A17* •

This plot can be enhanced with much more information. For example:

- We add a grid at the axis ticks with the `grid` function;
- We specify the plotting character with the `pch` graphics argument,
- its colours with the `col` (outline) and `bg` (fill) graphics arguments,
- its size with the `cex` "character expansion" graphics argument,
- the axis labels with the `xlab` and `ylab` graphics arguments;
- We add a title with the `title` function, and
- mark the centroid (centre of gravity) with two calls to `abline`, one specifying a vertical line (argument `v=`) and one horizontal (argument `vh=`) at the means of the two variables, computed with the `mean` function;
- The two lines are dashed, using the `lty` "line type" graphics argument,
- and coloured red using `col`;
- The centroid is shown as large diamond, using the `points` function and the `cex` graphics argument;
- Finally, the actual mean yields are displayed with the `text` function, using the `pos` and `adj` graphic argument to position the text with respect to the plotting position.

```
> plot(mhw$grain, mhw$straw, cex=0.8, pch=21, col="blue",
+      bg="red", xlab="Grain yield, lb.\ plot-1",
+      ylab="Straw yield, lb.\ per plot-1")
> grid()
```

```
> title(main="Mercer-Hall wheat uniformity trial")
> abline(v=mean(mhw$grain), lty=2, col="blue")
> abline(h=mean(mhw$straw), lty=2, col="blue")
> points(mean(mhw$grain), mean(mhw$straw), pch=23, col="black",
+        bg="brown", cex=2)
> text(mean(mhw$grain), min(mhw$straw),
+      paste("Mean:",round(mean(mhw$grain),2)), pos=4)
> text(min(mhw$grain), mean(mhw$straw),
+      paste("Mean:",round(mean(mhw$straw),2)), adj=c(0,-1))
```



The advantage of this **programmed enhancement** is that we can store the commands as a script and reproduce the graph by running the script.

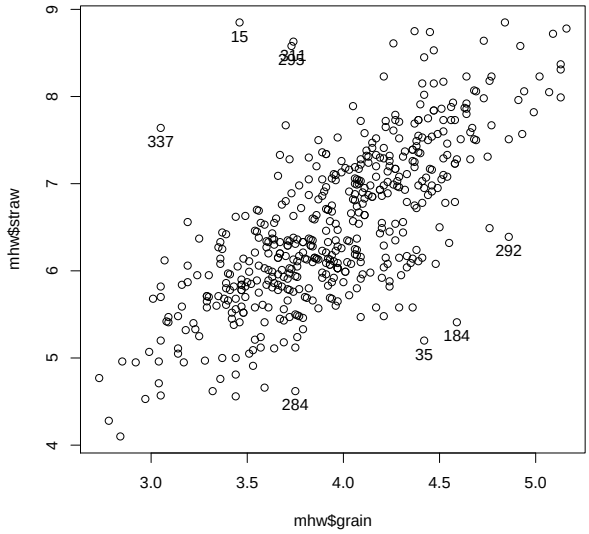
Some R graphics allow **interaction**.

Task 19 : Identify the plots which do not fit the general pattern. (In any analysis these can be the most interesting cases, requiring explanation.) •

For this we use the `identify` function, specifying the same plot coördinates as the previous `plot` command (i.e. from the plot that is currently displayed):

```
> plot(mhw$grain, mhw$straw)
> pts <- identify(mhw$grain, mhw$straw)
```

After `identify` is called, switch to the graphics window, **left-click** with the mouse on points to identify them, and **right-click** to exit. The plot should now show the row names of the selected points:



Q18 : *Which observations have grain yield that is much lower than expected (considering the straw) and which higher?* *Jump to A18* •

```
> tmp <- mhw[pts, ]
> tmp[order(tmp$grain), ]

     r  c grain straw
337 17 17  3.05  7.64
15  15  1  3.46  8.85
295 15 15  3.73  8.58
311 11 16  3.74  8.63
284  4 15  3.75  4.62
35  15  2  4.42  5.20
184  4 10  4.59  5.41
292 12 15  4.86  6.39

> rm(pts, tmp)
```

4.3 Answers

A12 : 2.73 *and* 5.16 *lb. per plot, respectively. Note the placement of the decimal point, as explained in the plot header. Here it is one digit to the left of the* |, *so the entry* 27 | 38 *is to be read as* 2.73, 2.78. *Return to Q12* •

A13 : *The stem plot shows the actual values (to some number of significant digits). This allows us to see if there is any pattern to the digits.* *Return to Q13* •

A14 : *The horizontal axis is the value of the variable being summarized (in this case, grain yield). It is divided into sections ("histogram bins") whose limits are*

shown by the vertical vars. The vertical axis is the count (frequency) of observations in each bin. *Return to Q14* •

A15 : *The two left-most histogram bins represent the values below 3 lb. per plot (horizontal axis); these appear to have 2 and 5 observations, respectively, for a total of 7; although it's difficult to estimate exactly. The stem plot, which shows the values to some precision, can show this exactly.* *Return to Q15* •

A16 : *The higher value of the* `adj` *argument to the* `density` *function gives a smoother curve. In this case with* `adj=2` *the curve is indistinguishable from a univariate normal distribution. The default curve is quite similar but with a slight asymmetry (peak is a bit towards the smaller values) and shorter tails. But, considering the sample size, it still strongly suggests a normal distribution.* *Return to Q16* •

A17 : *They are positively associated: higher grain yields are generally associated with higher straw yields. The relation appears to be linear across the entire range of the two measured variables. But the relation is diffuse and there are some clear exceptions.* *Return to Q17* •

A18 : *Plots 15, 337, 311 and 295 have grain yields that are lower than the general pattern; plots 308, 292, 184 and 35 the opposite.* *Return to Q18* •

5 Descriptive statistics

After visualizing the dataset, the next step is to compute some **numerical summaries**, also known as **descriptive statistics**. We can **summarize** all the variables in the dataset at the same time or individually with the `summary` function:

```
> summary(mhw)

       r                c          grain          straw
 Min.   : 1.00   Min.   : 1   Min.   :2.73   Min.   :4.10
 1st Qu.: 5.75   1st Qu.: 7   1st Qu.:3.64   1st Qu.:5.88
 Median :10.50   Median :13   Median :3.94   Median :6.36
 Mean   :10.50   Mean   :13   Mean   :3.95   Mean   :6.51
 3rd Qu.:15.25   3rd Qu.:19   3rd Qu.:4.27   3rd Qu.:7.17
 Max.   :20.00   Max.   :25   Max.   :5.16   Max.   :8.85

> summary(mhw$grain)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.73    3.64    3.94    3.95    4.27    5.16
```

Q19 : *What are the summary statistics for grain yield?* *Jump to A19* •

5.1 Other descriptive statistics*

The descriptive statistics in the summary all have their individual functions: `max`, `median`, `mean`, `max`, and `quantile`. Note this latter has a second argument, `probs`, with a single value or list (formed with the `c` function) of the probabilities for which the quantile is requested:

```
> max(mhw$grain)

[1] 5.16

> min(mhw$grain)

[1] 2.73

> median(mhw$grain)

[1] 3.94

> mean(mhw$grain)

[1] 3.9486

> quantile(mhw$grain, probs = c(0.25, 0.75))

   25%    75%
3.6375 4.2700
```

Other statistics that are often reported are the variance, standard deviation (square root of the variance) and inter-quartile range (IQR).

Task 20 : Compute these for grain yield. •

The `var`, `sd` and `IQR` functions compute these.

```
> var(mhw$grain)

[1] 0.21002

> sd(mhw$grain)

[1] 0.45828

> IQR(mhw$grain)

[1] 0.6325
```

Other measures applied to distributions are the skewness (deviation from symmetry; symmetric distributions have no skewness) and kurtosis (concentration around central value; a normal distribution has kurtosis of 3). Functions for these are not part of base R but are provided as the `skewness` and `kurtosis` functions of the curiously-named `e1071` package from the Department of Statistics, Technical University of Vienna[7].

[7] This may have been the Department's administrative code.

Task 21 : Load the `e1071` package and compute the skewness and kurtosis. Note that if `e1071` is not already installed on your system, you have to install it first. •

Optional packages are best loaded with the `require` function; this ensures that the package is not already loaded before loading it (to avoid duplication).

> **Note:** Since this is the only use we make of this package, we can unload it with the `detach` function. This is generally not necessary.

```
> require(e1071)
> skewness(mhw$grain)

[1] 0.035363

> kurtosis(mhw$grain)

[1] -0.27461

> detach(package:e1071)
```

Note that `kurtosis` computes the so-called "excess" kurtosis, i.e., the difference from the normal distribution's value (3). So a normally-distributed variable would have no excess kurtosis.

Q20 : *How do the skew and kurtosis of this distribution compare to the expected values for a normally-distributed variable, i.e., 0 (skew) and 0 (excess kurtosis)?* *Jump to A20*

•

5.2 Attaching a data frame to the search path

We can avoid the construction `frame$variable` by attaching the frame to the **search path** with the `attach` function.

```
> search()

 [1] ".GlobalEnv"        "package:spgwr"     "package:maptools"
 [4] "package:grid"      "package:foreign"   "package:nlme"
 [7] "package:lattice"   "package:gstat"     "package:sp"
[10] "package:MASS"      "package:class"     "package:stats"
[13] "package:graphics"  "package:grDevices" "package:utils"
[16] "package:datasets"  "package:methods"   "Autoloads"
[19] "package:base"

> summary(grain)

Error in summary(grain) : object "grain" not found

> attach(mhw)
```

```
> search()

 [1] ".GlobalEnv"        "mhw"               "package:spgwr"
 [4] "package:maptools"  "package:grid"      "package:foreign"
 [7] "package:nlme"      "package:lattice"   "package:gstat"
[10] "package:sp"        "package:MASS"      "package:class"
[13] "package:stats"     "package:graphics"  "package:grDevices"
[16] "package:utils"     "package:datasets"  "package:methods"
[19] "Autoloads"         "package:base"

> summary(grain)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.73    3.64    3.94    3.95    4.27    5.16
```

Notice how the `mhw` object was not on the search path prior to the `attach` command.

5.3 A closer look at the distribution

Q21 : *What is the range in grain yields? What proportion of the median yield is this? Does this seem high or low, considering that all plots were treated the same?* *Jump to A21* •

To answer this we can use the `diff` ("difference") and `median` functions:

```
> diff(range(grain))

[1] 2.43

> diff(range(grain))/median(grain)

[1] 0.61675
```

Q22 : *Which is the lowest-yielding plot? Does it also have low straw yield?* *Jump to A22* •

To answer this, use the `which.min` function to identify the record number with the lowest yield:

```
> ix <- which.min(grain)
> mhw[ix, "straw"]

[1] 4.77
```

We can **select** cases based on logical criteria, for example, to find the lowest-yielding plots.

Task 22 : Find all observations with grain yield less than 3 lb. per plot, and also those with grain yield in the lowest (first) percentile. •

We can use either the `subset` function or direct matrix selection. The `quantile` function returns a list with quantiles; here we illustrate the default, the case where we use the `seq` function to ask for the ten deciles, and finally just the 1% quantile:

```
> row.names(subset(mhw, grain < 3))

[1] "149" "336" "338" "339" "441" "467" "470"

> quantile(grain)

    0%    25%    50%    75%   100%
2.7300 3.6375 3.9400 4.2700 5.1600

> quantile(grain, seq(0, 1, 0.1))

   0%   10%   20%   30%   40%   50%   60%   70%   80%   90%  100%
2.730 3.370 3.558 3.700 3.820 3.940 4.070 4.210 4.362 4.520 5.160

> mhw[grain < quantile(grain, 0.01), ]

     r  c grain straw
336 16 17  2.92  4.95
338 18 17  2.73  4.77
339 19 17  2.85  4.96
467  7 24  2.78  4.28
470 10 24  2.84  4.10
```

Q23 : *Which plots have grain yield less than 3 lb.? Which are the lowest-yielding 1%? Are those close to each other?* *Jump to A23*
•

5.4 Answers

A19 : *Minimum 2.73, maximum 5.16, arithmetic mean 3.95, first quartile 3.64, third quartile 4.27, median 3.94.* *Return to Q19* •

A20 : *Both skew and excess kurtosis are quite close to the expected values (0). This strengthens the evidence of a normal distribution.* *Return to Q20* •

A21 : *The range in grain yields is 2.43, which is about 62% of the median. This seems quite high considering the "equal" treatment.* *Return to Q21* •

A22 : *The lowest-yielding plot is 338, with a grain yield of 4.77 lb.* *Return to Q22* •

A23 : *Plots with yield less than 3 lb. are 149, 336, 338, 339, 441, 467, and 470. The lowest percent are plots 336, 338, 339, 467, and 470. The first three are all in*

field column 17 and almost adjacent field rows (16, 18, 19); this seems definitely to be a 'low-yield "hot spot" in the experimental field. The last two are both in field column 24 but a few rows apart (7 and 10). *Return to Q23* •

6 Editing a data frame

If you need to fix up a few data entry errors, the data frame can be edited interactively with the `fix` function:

```
> fix(mhw)
```

In this case there is nothing to change, so just close the editor.

New variables are calculated in the local variable space. For example, the **grain-to-straw ratio** is an important indicator of how well the wheat plant has formed grain, relative to its size.

Task 23 : Compute the grain/straw ratio and summarize it. •

Note that arithmetic operations are performed on entire vectors; these are called **vectorized** operations:

```
> gsr <- grain/straw
> summary(gsr)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.391   0.574   0.604   0.611   0.642   0.850
```

Q24 : *What is the range in grain/straw ratio? Is it relatively larger or smaller than the range in grain?* *Jump to A24* •

```
> range(gsr)

[1] 0.39096 0.85000

> diff(range(gsr))/median(gsr)

[1] 0.75944

> diff(range(grain))/median(grain)

[1] 0.61675
```

For further analysis we would like to include this in the data frame itself, as an additional variable.

Task 24 : Add grain-straw ratio to the `mhw` data frame and remove it from the local workspace. •

For this we use the `cbind` ("column bind") function to add a new matrix column (data frame field):

```
'data.frame':        500 obs. of  5 variables:
 $ r    : int  1 2 3 4 5 6 7 8 9 10 ...
 $ c    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ grain: num  3.63 4.07 4.51 3.9 3.63 3.16 3.18 3.42 3.97 3.4 ...
 $ straw: num  6.37 6.24 7.05 6.91 5.93 5.59 5.32 5.52 6.03 5.66 ...
 $ gsr  : num  0.57 0.652 0.64 0.564 0.612 ...

> mhw <- cbind(mhw, gsr)
> str(mhw)
```

Note how the new matrix column took the name from the local variable.

Now we remove the local variable `gsr` so that we do not confuse it with the `gsr` field of the `mhw` data frame:

```
> ls()

[1] "gsr" "ix"  "mhw"

> rm(gsr)
> ls()

[1] "ix"  "mhw"
```

But now the new field is not found unless the frame is named explicitly:

```
> summary(gsr)

Error in summary(gsr) : object "gsr" not found

> summary(mhw$gsr)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.391   0.574   0.604   0.611   0.642   0.850
```

Task 25 : Detach and re-attach the `mhw` object, so that the field `gsr` can be found. •

```
> detach(mhw)
> attach(mhw)

> summary(gsr)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.391   0.574   0.604   0.611   0.642   0.850
```

Task 26 : Save the updated `mhw` object in R format. •

We use a different file name to distinguish this from the original file, without the added column.

```
> save(mhw, file = "mhw2.RData")
```

6.1 Answers

A24 : *The range is from 0.39 to 0.85, i.e. the ratio of grain to straw doubles. This is about 76% of the median ratio, which is considerably higher than the comparable figure for grain yield (about 62%).* *Return to Q24* •

7 Univariate modelling

After descriptive statistics and visualisation comes the attempt to build **statistical models** of the underlying processes. These are **empirical** mathematical relations that describe one variable in terms of some hypothetical underlying distribution of which it is a realisation, or describe several variables either as equals ("correlation") or where one is described in terms of others ("regression"). We explore these, from simple to complex.

The simplest kind of model is about the distribution of a single variable, i.e, **univariate** modelling.

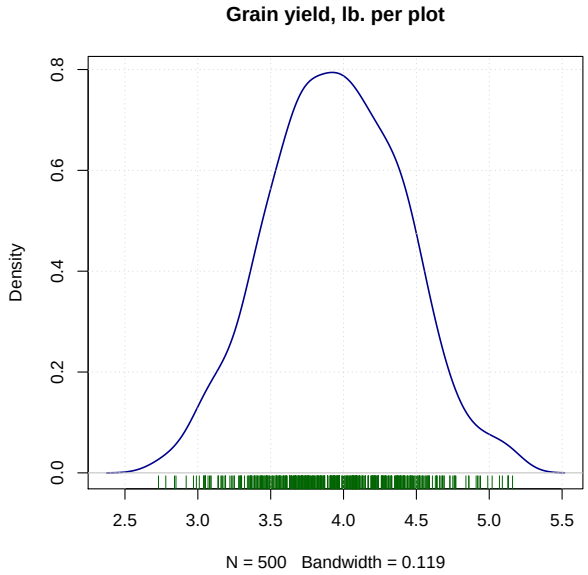
We suppose that the observed sample distribution is from an underlying probability distribution. This raises two questions: (1) what is the form of that distribution, and (2) what are its parameters?

To decide what **theoretical** distribution might fit, we first visualise the **empirical** distribution. This continues the ideas from §4.1.2.

Task 27 : Visualise an empirical continuous frequency distribution on the rug plot. •

We again use the `density` function, with default arguments:

```
> plot(density(mhw$grain), col="darkblue",
+      main="Grain yield, lb. per plot", lwd=1.5)
> rug(mhw$grain, col="darkgreen")
> grid()
```

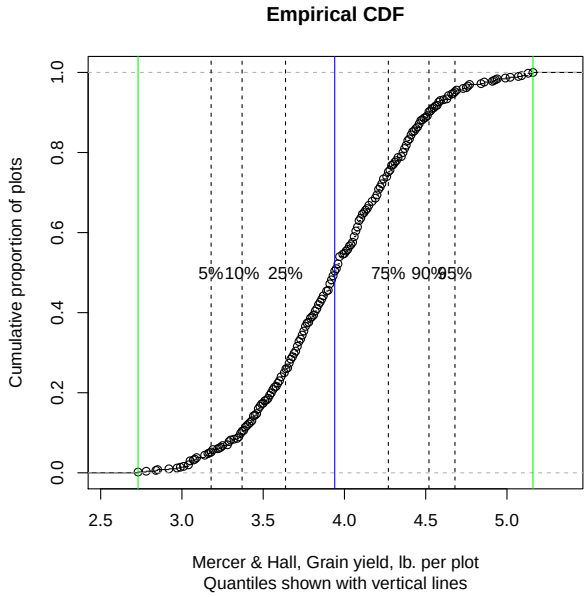


We can also view the distribution as a **cumulative** rather than **density** distribution.

Task 28 : Visualise the empirical **cumulative distribution** of grain yield. •

We use the `ecdf` ("**e**mpirical **c**umulative **d**istribution **f**unction") function to compute the distribution, then plot it with the `plot` function. Vertical lines are added to the plot with the `abline` ("add a straight line") function, at the median, extremes, and specified quantiles.

```
> plot(ecdf(grain), pch=1,
+      xlab="Mercer & Hall, Grain yield, lb. per plot",
+      ylab="Cumulative proportion of plots",
+      main="Empirical CDF",
+      sub="Quantiles shown with vertical lines")
> q <- quantile(grain, c(.05, .1, .25, .75, .9, .95))
> abline(v=q, lty=2)
> abline(v=median(grain), col="blue")
> abline(v=max(grain), col="green")
> abline(v=min(grain), col="green")
> text(q, 0.5, names(q))
> rm(q)
```

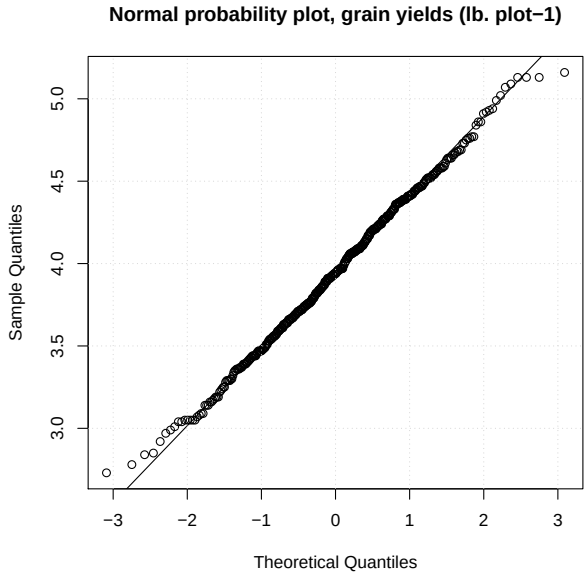


Q25 : *From the histogram, stem-and-leaf plot, empirical cumulative distribution, and theory, what probability distribution is indicated for the grain yield?* *Jump to A25* •

We can also visualise the distribution against the theoretical normal distribution computed with the sample mean and variance. There are (at least) two useful ways to visualise this.

First, compare the actual and theoretical normal distribution is with the `qqnorm` function to plot these against each other and then superimpose the theoretical line with the `qqline` function:

```
> qqnorm(grain, main = "Normal probability plot, grain yields (lb. plot-1)")
> qqline(grain)
> grid()
```



Q26 : *Does this change your opinion of normality?* *Jump to A26* •

The second way to visually compare an empirical and theoretical distribution is to display the empirical density plot, superimposing the normal distribution that would be expected with the sample mean and standard deviation.

Task 29 : Fit a normal probability distribution to the empirical distribution of grain yield. •

Q27 : *What are the best estimates of the parameters of a normal distribution for the grain yield?* *Jump to A27*

•

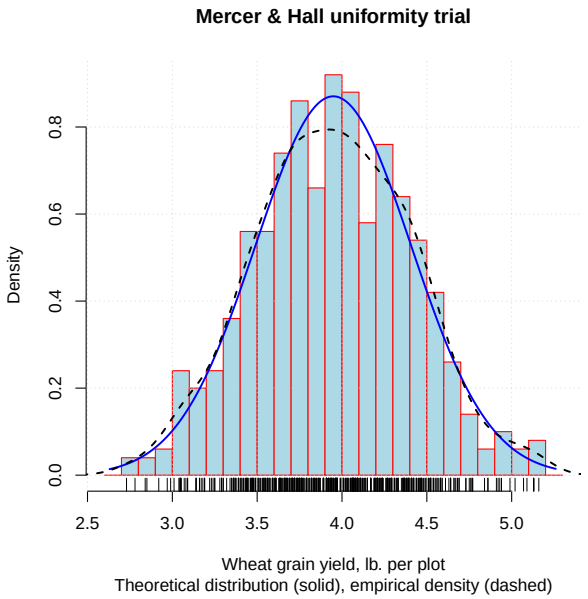
These are computed with the `mean` and `sd` functions:

```
> mean(grain)

[1] 3.9486

> sd(grain)

[1] 0.45828
```

With these in hand, we can plot the theoretical distribution against the empirical distribution:

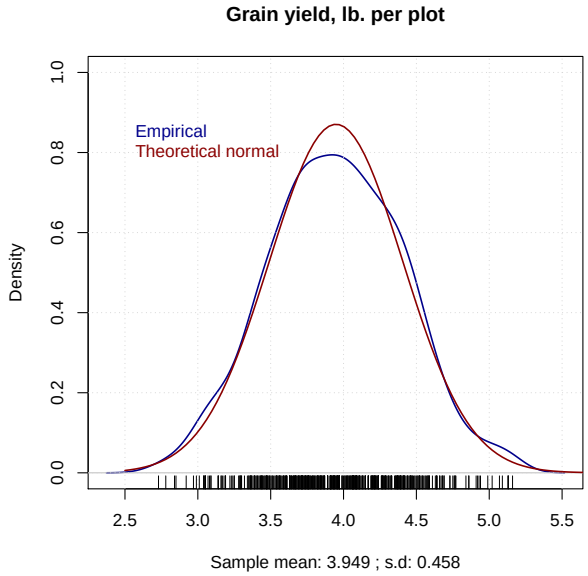
Task 30 : Graphically compare the theoretical and empirical distributions. •

```
> res <- 0.1
> hist(grain, breaks=seq(round(min(grain),1)-res,
+                round(max(grain),1)+res, by=res),
+      col="lightblue", border="red", freq=F,
+      xlab="Wheat grain yield, lb. per plot",
+      main="Mercer & Hall uniformity trial",
+      sub="Theoretical distribution (solid), empirical density (dashed)")
> grid()
> rug(grain)
> x <- seq(min(grain)-res, max(grain)+res, by=.01)
> lines(x, dnorm(x, mean(grain), sd(grain)), col="blue", lty=1, lwd=1.8)
> lines(density(grain), lty=2, lwd=1.8, col="black")
> rm(res, x)
```



We can also see this on the empirical density. This version also uses the `curve` method to draw the theoretical curve.

```
> plot(density(mhw$grain), col="darkblue",
+     main="Grain yield, lb. per plot", lwd=1.5, ylim=c(0,1),
+     xlab=paste("Sample mean:",round(mean(mhw$grain), 3),
+       "; s.d:", round(sd(mhw$grain),3)))
> grid()
> rug(mhw$grain)
> curve(dnorm(x, mean(mhw$grain), sd(mhw$grain)), 2.5, 6, add=T,
+       col="darkred", lwd=1.5)
> text(2.5, 0.85, "Empirical", col="darkblue", pos=4)
> text(2.5, 0.8, "Theoretical normal", col="darkred", pos=4)
```



There are several tests of normality; here we use the Shapiro-Wilk test, implemented by the `shapiro.test` function. This compares the empirical distribution (from the **sample**) with the theoretical distribution, and computes a statistic ("W") for which is known the probability that it could occur by change, **assuming** the sample is really from a normally-distributed **population**. The reported probability value is the chance that **rejecting the null hypothesis** H_0 that the sample *is* from a normal population is an **incorrect decision** (i.e. the probability of committing a Type I error).

```
> shapiro.test(grain)

        Shapiro-Wilk normality test

data:  grain
W = 0.997, p-value = 0.486
```

Q28 : *According to the Shapiro-Wilk test, what is the probability if we reject the null hypothesis that this empirical distribution is a realisation of a normal distribution with the sample mean and standard deviation as parameters, we would be wrong (i.e. commit a Type I error)? So, should we reject the null hypothesis of normality? Should we consider grain yield to be a normally-distributed variable?* *Jump to A28* •

Once we've established that the yields can be reasonably modelled by a normal distribution, we can compute **confidence limits** on the mean yield. Mercer and Hall used the 50% confidence level, which they called the **probable error**: there is equal chance for the true yield to be inside as outside this interval.

Task 31 : Compute the probable error for grain yield in a 1/500 acre plot.

•

Although we've fitted a normal distribution, both the mean and standard deviation were estimated from the sample. So, we should use Student's t-distribution (although with so many plots the difference with the normal z-distribution will be very small).

The probable error is computed from the limits of the interquartile range of the distribution; we get the quantiles of the t distribution with the `qt` function, specifying the degrees of freedom:

```
> (t.q13 <- qt(c(0.25, 0.75), length(mhw$grain) - 1))

[1] -0.67498  0.67498
```

This is for $\mu = 0, \sigma = 1$. We then scale this to the data:

```
> (pe <- mean(mhw$grain) + t.q13 * sd(mhw$grain))

[1] 3.6393 4.2580
```

And we can express it as a relative error; conventionally this is expressed as error ± relative to the mean:

```
> rel.error <- (diff(pe)/2)/mean(mhw$grain)
> round(100 * rel.error, 2)

[1] 7.83
```

Q29 : *If the true yield over the entire field is the observed mean yield, what yields can be expected in any one 1/500 acre plot, with 50% confidence that the mean is in this range?* *Jump to A29* •

Q30 : *What is the probable error expressed as a percentage of mean yield?* *Jump to A30* •

To put this in practical terms, since this was Mercer and Hall's main concern, we calculate the probable error in absolute kg ha^{-1}.

We saw above that the mean grain yield in this experiment was 3.95 lb. plot^{-1}. Scaling this up to a hectare basis (which is how yields are normally expressed):

```
> (yield.ha <- mean(mhw$grain) * 500/(0.40469)/(2.2046226))

[1] 2212.9
```

The two constants in this formula are `0.40469 ha acre`$^{-1}$ and `2.2046226 lb. kg`$^{-1}$, and there are 500 plots per acre.

Q31 : *If we try to estimate the yield in kg ha^{-1} from a single 1/500 acre plot in a variety trial, how much error (in absolute terms) can we expect,*

with 50% probability? What are the practical implications of this? *Jump to A31* •

```
> round(yield.ha * rel.error, 2)

[1] 173.35
```

Clean up:

```
> rm(t.q13, pe, rel.error, yield.ha)
```

7.1 Answers

A25 : *The Normal distribution. From theory: the addition of multiple independent sources of noise. From the plots: visual fit to simulated normal distributions.* *Return to Q25* •

A26 : *The distribution still looks normal, except at both tails: the highest yields are not as high, and the lowest not as low, as expected if the yields were normally distributed.* *Return to Q26* •

A27 : *Parameters* $\mu = 3.95, \sigma = 0.458$. *Return to Q27* •

A28 : *According to the Shapiro-Wilk test of normality, the probability that rejecting the* ***null hypothesis*** *of normality would be an* ***incorrect*** *decision (i.e. a* ***Type I error*** *is 0.49; this is quite high, so we* ***do not reject*** *the null hypothesis. So, we consider the grain yield to be a realisation of a normal distribution.* *Return to Q28* •

A29 : *From* 3.64 *to* 4.36 *lb. plot*$^{-1}$*; the observed mean yield is* 3.95 *lb. plot*$^{-1}$. *Return to Q29* •

A30 : ± *7.83%.* *Return to Q30* •

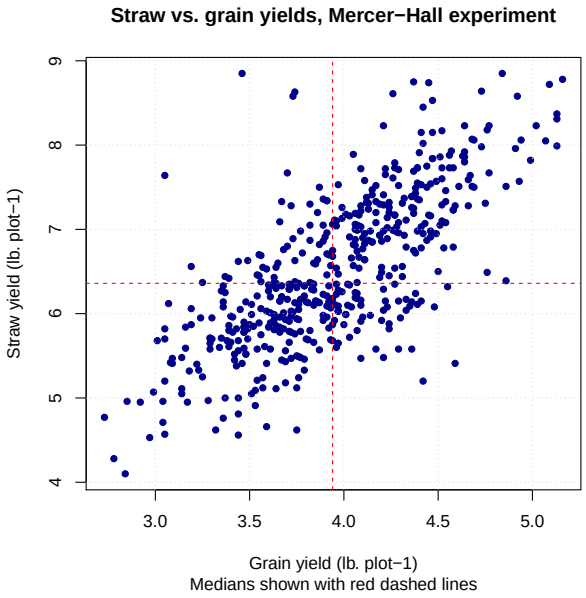
A31 : *From a true yield (estimated here from our mean) of 2213 kg wheat grain, the probable error is 173 kg, a substantial amount. There is a 50% chance of observing this much deviation if we estimate the per-hectare yield from any single 1/500 acre plot. Clearly, this is where the idea of* ***replicated*** *plots originated.* *Return to Q31* •

8 Bivariate modelling: two continuous variables

After modelling the distribution of a single variable, we now model the **joint** distribution of two variables, i.e., **bivariate** modelling.

In §4.2 we displayed a scatterplot of straw vs. grain yield, repeated here with some additional graphical elements:

```
> plot(mhw$straw ~ mhw$grain,
+      xlab="Grain yield (lb. plot-1)",
+      ylab="Straw yield (lb. plot-1)",
+      pch=20, col="darkblue", cex=1.2,
+      sub="Medians shown with red dashed lines")
> title(main="Straw vs. grain yields, Mercer-Hall experiment")
> grid()
> abline(v=median(mhw$grain), lty=2, col="red")
> abline(h=median(mhw$straw), lty=2, col="red")
```



Note the use of the ~ formula operator to indicate **statistical dependence**, here of straw on grain; we will explore this further just below. The centroid is shown by intersecting horizontal and vertical lines, added with the `abline` function.

Q32 : *From the scatterplot of straw vs. grain yield, what functional relation is suggested?* *Jump to A32* •

For a **bivariate linear** relation, as hypothesised here, we can view this four ways:

1. A bivariate **linear correlation** between the two variables (straw and grain yields) (§8.1);
2. A univariate **linear regression** of straw (dependent) on grain (independent) yield (§8.2);
3. A univariate **linear regression** of grain (dependent) on straw (independent) yield (§8.3).
4. A **linear structural relation** between the two yields (§8.3).

These will each be explored below.

Bivariate correlation, bivariate structural relations and univariate regression all compare two variables that refer to *the same observations*, that is, they are *paired*. This is the natural order in a data frame: each *row* represents *one observation* on which several *variables* were measured; in the present case, the row and column numbers of the plots, and the grain and straw yields.

Correlation and various kinds of regression are often misused. There are several good journal articles that explain the situation, with examples from earth science applications [30, 47]. A particularly understandable introduction to the proper use of regression is by Webster [48].

Note that in this case there is no evidence to suggest a **non-linear** or **piecewise** relation; but in many cases these are possibilities that must be compared with a linear model.

8.1 Correlation

Correlation measures the strength of the association between two variables measured on the same object, from -1 (perfect negative correlation), through 0 (no correlation), to $+1$ (perfect positive correlation). The two variables have logically equal status, so there is no concept of **causation**; in addition, there is no functional relation, so there is no way to **predict**.

There are several numerical measures of correlation; we will see two:

1. Parametric: Pearson's product moment correlation coefficient (PMCC) r (§8.1.1);
2. Non-parametric: Spearman's ρ (§11.3).

8.1.1 Parametric correlation

The PMCC should only be used if the two variables are distributed approximately **bivariate normally**. This is two normal distributions, but with some degree of correlation. So we first check whether the relation between grain and straw yield has this distribution.

Task 32 : Visualise a bivariate normal distribution with the parameters of the grain and straw yields; visually compare with the actual bivariate distribution. •

R has a `rnorm` function to simulate a random sample of a given size from the normal distribution, by analogy to the `runif` function presented above. However, to simulate a **correlated** sample of several variables, we turn to the `mvrnorm` function in the `MASS` ("Modern Applied Statistics with S") package which corresponds to the very useful advanced text of Venables and Ripley [46]. This function uses a vector of the variable means, along with the **variance-covariance** matrix of two or more variables.

The variable means are computed with the vectorized `colMeans` function, which finds the by-column mean all or some of the columns in a data frame:

```
> colMeans(mhw[, c("grain", "straw")])

 grain  straw
3.9486 6.5148
```

The variance-covariance matrix is computed with the vectorized `var` function:

```
> var(mhw[, c("grain", "straw")])

        grain   straw
grain 0.21002 0.30043
straw 0.30043 0.80696
```

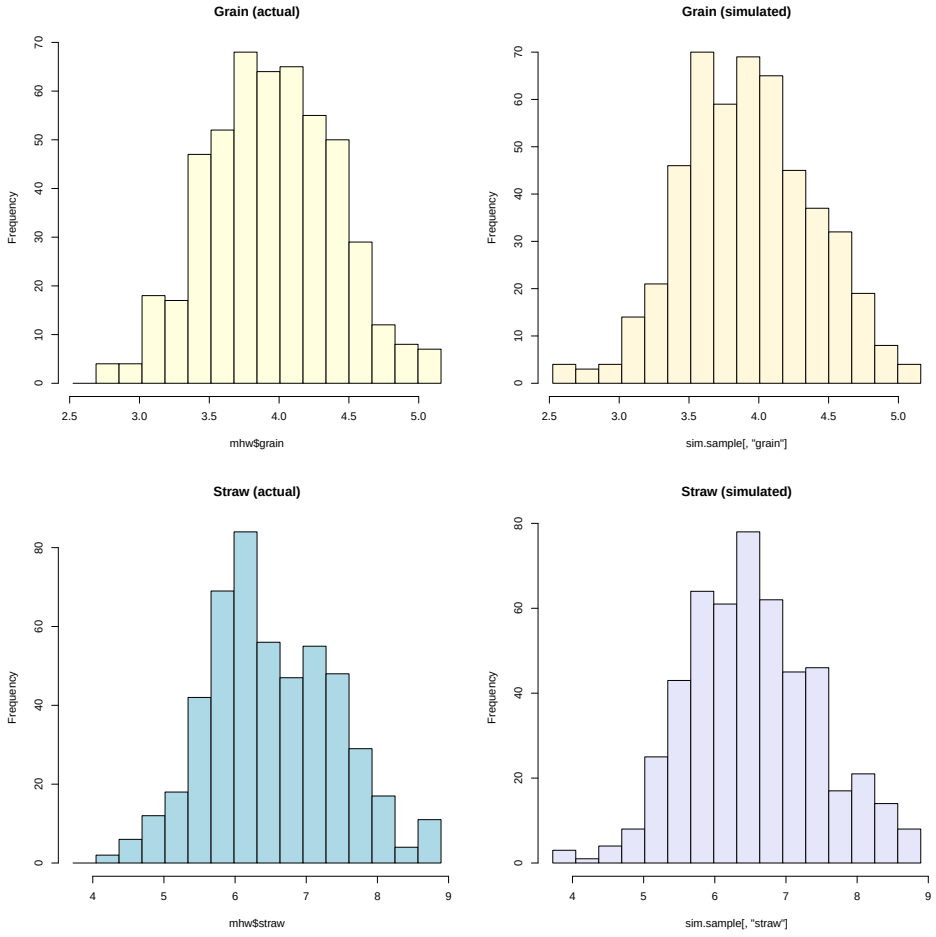
Q33 : *What do the diagonals and off-diagonals of this matrix represent? What are their units of measure?* *Jump to A33* •

From these the `mvrnorm` can draw a simulated random sample. We first load the optional `MASS` package with the `require` function; the `mvrnorm` function is then available.

```
> require(MASS)
> sim.sample <- mvrnorm(length(mhw$grain), mu = colMeans(mhw[,
+     c("grain", "straw")]), Sigma = var(mhw[, c("grain",
+     "straw")]))
> head(sim.sample)

      grain  straw
[1,] 4.1254 6.8756
[2,] 3.9784 6.6011
[3,] 3.8108 5.6451
[4,] 4.3392 6.7978
[5,] 4.3313 6.0042
[6,] 4.4293 7.4135

> summary(sim.sample)

     grain          straw
 Min.   :2.52   Min.   :3.72
 1st Qu.:3.61   1st Qu.:5.88
 Median :3.91   Median :6.51
 Mean   :3.92   Mean   :6.53
 3rd Qu.:4.23   3rd Qu.:7.12
 Max.   :5.15   Max.   :8.89

> summary(mhw[, c("grain", "straw")])

     grain          straw
 Min.   :2.73   Min.   :4.10
 1st Qu.:3.64   1st Qu.:5.88
 Median :3.94   Median :6.36
 Mean   :3.95   Mean   :6.51
 3rd Qu.:4.27   3rd Qu.:7.17
 Max.   :5.16   Max.   :8.85
```

We can also visualise these distributions with histograms. To ensure that the visualisation is comparable, we compute the overall minimum and maximum of both variables and use these to explicitly set the axis limits.

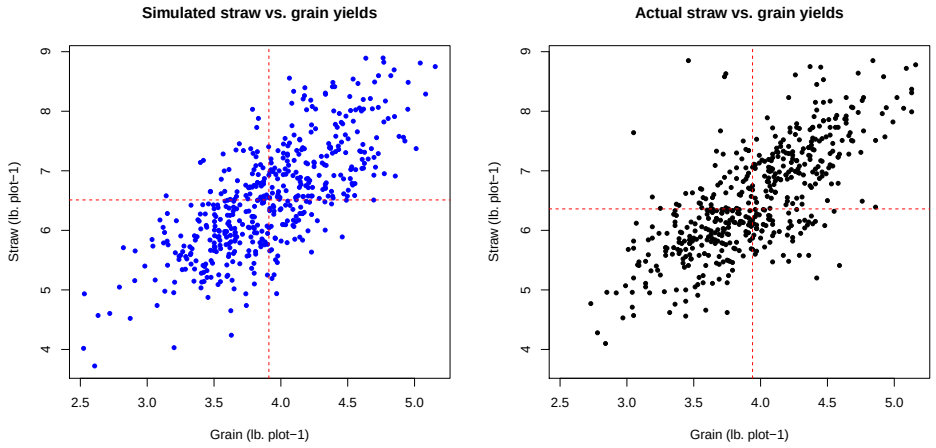
```
> par(mfrow = c(2, 2))
> grain.lim = c(min(sim.sample[, "grain"], mhw$grain),
+     max(sim.sample[, "grain"], mhw$grain))
> straw.lim = c(min(sim.sample[, "straw"], mhw$straw),
+     max(sim.sample[, "straw"], mhw$straw))
> hist(mhw$grain, xlim = grain.lim, main = "Grain (actual)",
+     col = "lightyellow", breaks = seq(grain.lim[1], grain.lim[2],
+         length = 17))
> hist(sim.sample[, "grain"], xlim = grain.lim, main = "Grain (simulated)",
+     col = "cornsilk", breaks = seq(grain.lim[1], grain.lim[2],
+         length = 17))
> hist(mhw$straw, xlim = straw.lim, main = "Straw (actual)",
+     col = "lightblue", breaks = seq(straw.lim[1], straw.lim[2],
+         length = 17))
> hist(sim.sample[, "straw"], xlim = straw.lim, main = "Straw (simulated)",
+     col = "lavender", breaks = seq(straw.lim[1], straw.lim[2],
+         length = 17))
> par(mfrow = c(1, 1))
```



Q34 : *How well do the two univariate simulations match the actual data?* *Jump to A34* •

So we can simulate a sample with the same mean and standard deviation as the grain and straw yields, and plot them against each other in a scatterplot. We display this side-by-side with the actual scatterplot.

```
> par(mfrow = c(1, 2))
> plot(sim.sample, main = "Simulated straw vs. grain yields",
+     xlab = "Grain (lb. plot-1)", ylab = "Straw (lb. plot-1)",
+     xlim = grain.lim, ylim = straw.lim, pch = 20, col = "blue")
> abline(v = median(sim.sample[, 1]), lty = 2, col = "red")
> abline(h = median(sim.sample[, 2]), lty = 2, col = "red")
> plot(mhw$grain, mhw$straw, main = "Actual straw vs. grain yields",
+     xlab = "Grain (lb. plot-1)", ylab = "Straw (lb. plot-1)",
+     xlim = grain.lim, ylim = straw.lim, pch = 20, col = "black")
> abline(v = median(mhw$grain), lty = 2, col = "red")
> abline(h = median(mhw$straw), lty = 2, col = "red")
> par(mfrow = c(1, 1))
```



Q35 : *Do the two relations (simulated vs. actual) look similar? Do they support the hypothesis of a bivariate linear relation?* *Jump to A35* •

We delete the temporary variables used in this visualisation:

```
> rm(sim.sample, grain.lim, straw.lim)
```

Challenge: The single simulation is only one realisation of the hypothetical process. Repeat the simulation several times and compare (1) the simulations with each other; (2) the simulations with the actual data.

With this evidence of bivariate normality, we are justified in computing the parametric correlation.

Task 33 : Compute the PMCC between grain and straw yield, and its 95% confidence limit. •

The `cor` function computes the correlation; the `cor.test` function also computes the confidence interval:

```
> cor(mhw$grain, mhw$straw)

[1] 0.72978

> cor.test(mhw$grain, mhw$straw)

        Pearson's product-moment correlation

data:  mhw$grain and mhw$straw
t = 23.821, df = 498, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.68599 0.76830
sample estimates:
    cor
0.72978
```

Q36 : *What are the lowest possible, most probable, and highest possible values of the correlation coefficient r, with 95% confidence? Do you consider this a very strong, strong, moderately strong, weak, or no correlation? Positive or negative?* *Jump to A36* •

8.2 Univariate linear regression

Univariate **linear regression** differs from bivariate correlation in that one of the two variables is considered mathematically **dependent** on the other.

An important distinction is made between predictors which are known without error, whether fixed by the experimenter or measured, and those that are not.

Fixed effects model Webster [48] calls the first type a "Gauss linear model"[8] because only the predictand has Gaussian or "normal" error, whereas the predictor is known without error. The regression goes in one direction only, from the mathematical predictor to the random response, and is modelled by a **linear model with error in the response**:

$$y_i = BX_i + \varepsilon_i \tag{8.1}$$

of which the simplest case is a line with intercept:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \tag{8.2}$$

[8] referring to the developer of least-squares regression

In this model there is no error associated with the predictors x_i, only with the predictand y_i. The predictors are known without error, or at least the error is quite small in comparison to the error in the model. An example is a designed agricultural experiment where the quantity of fertiliser added (the predictor) is specified by the design and the crop yield is measured (the predictand); there is random error ε_i in this response.

Further, the errors ε_i are considered to be **identically and independently distributed** (IID):

- no relation between the magnitude of the error and that of the predictor (**homoscedascity**);
- no **serial correlation** between errors (e.g., small errors systematically followed by another small errors) in the sequence of predictors.

These assumptions can be verified after the regression parameters are estimated, using feature-space **regression diagnostics** (§8.2.4) and spatial analysis (§18). In the present case we will see (§18) that the residuals are not independent and a more sophisticated model is needed to get a proper regression equation.

Random effects model In the present example both variables (grain and straw yields) were measured with error; they were not imposed by the experimenter. Both variables should have Gaussian error, with some correlation. This is modelled as a **bivariate normal distribution** of two random variables, X and Y with (unknown) population means μ_X and μ_Y, (unknown) population variances σ_X and σ_Y, and an (unknown) correlation ρ_{XY} which is computed as the standardised (unknown) covariance $\mathrm{Cov}(X, Y)$:

$$\begin{aligned} X &\sim \mathcal{N}(\mu_X, \sigma_X) \\ Y &\sim \mathcal{N}(\mu_Y, \sigma_Y) \\ \rho_{XY} &= \mathrm{Cov}(X, Y)/\sigma_X\sigma_Y \end{aligned}$$

This is exactly what was modelled in the previous section.

> **Note:** In practice, the distinction between the two models is not always clear. The predictor, even if specified by the experimenter, can also have some measurement error. In the fertiliser experiment, even though we specify the amount per plot, there is error in measuring, transporting, and spreading it. In that sense it can be considered a random variable. But, since we have some control over it, the experimental error can be limited by careful procedures. We can not limit the error in the response by the same techniques.

8.2.1 Fitting a regression line – 1

When we decide to consider one of the variables as as a response and the other as a predictor, we attempt to fit a line that best describes this relation. There are three types of lines we can fit, usually in this order:

1. Exploratory, non-parametric
2. Parametric

3. Robust

The first kind just gives a "smooth" impression of the relation. The second fits according to some optimality criterion; the classic *least-squares* estimate is in this class. The third is also parametric but optimises some criterion that protects against a few unusual data values in favour of the majority of the data.

In the present case, our analysis above provides evidence that the relation is indeed well-modelled by a bivariate normal distribution, so that a parametric approach can be taken. We will then examine regression diagnostics to see if a robust regression is needed; these methods are discussed below in §11.

Q37 : *Is there any reason to choose grain or straw as the dependent variable (predictand), and other as the independent (predictor)?* *Jump to A37* •

The simplest way to model a parametric linear regression is by **ordinary least-squares** (OLS). The next section explains the mathematical justification; the computation continues in §8.2.3.

8.2.2 Ordinary least-squares*

Note: This explanation is adapted from Lark and Cullis [24, Appendix].

In the general linear model, with any number of predictors, there is a **design matrix** of predictor values usually written as $\mathbf{X}$, with one row per observation (data point) and one column per predictor. So in the single-predictor with intercept case, it has two columns: (1) a column of 1 representing the intercept, and (2) a column of predictor values x_i. The predictand (response variable) is a column vector $\mathbf{y}$. The coefficient vector $\boldsymbol{\beta}$ is what multiplies the design matrix to produce the response:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{8.3}$$

where $\boldsymbol{\varepsilon}$ is a column vector of **residuals**, also called **errors**, i.e., the lack of fit. We know the values in the predictor matrix $\mathbf{X}$ and the response vector $\mathbf{y}$ from our observations, so the task is to find the optimum values of the coefficients vector $\boldsymbol{\beta}$.

To solve this we need an optimization criterion. The obvious criterion is to minimize the total error (lack of fit) as some function of $\boldsymbol{\varepsilon} = \mathbf{y} - \mathbf{X}\boldsymbol{\beta}$; the goodness-of-fit is then measured by the size of this error. A common way to measure the total error is by the sum of vector norms; in the simplest case the Euclidean distance from the expected value, which we take to be 0 in order to have an unbiased estimate. If we decide that both positive and negative residuals are equally important, and that larger errors are more serious than smaller, the vector norm is expressed as the sum of squared errors, which in matrix algebra can be written as:

$$S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \tag{8.4}$$

which expands to

$$S = \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\beta + \beta^T\mathbf{X}^T\mathbf{X}\beta \tag{8.5}$$

This is minimized by finding the partial derivative with respect the the unknown coefficients β, setting this equal to $\mathbf{0}$, and solving:

$$\begin{aligned} \frac{\partial}{\partial\beta}S &= -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\beta \\ 0 &= -\mathbf{X}^T\mathbf{y} + \mathbf{X}^T\mathbf{X}\beta \\ (\mathbf{X}^T\mathbf{X})\beta &= \mathbf{X}^T\mathbf{y} \\ \hat{\beta}_{\text{OLS}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \end{aligned} \tag{8.6}$$

which is the usual OLS solution.

The above solution depends on an important assumption: the errors must be identically and independently distributed (abbreviated *i.i.d.*). We did not consider the direction of the error, i.e., with which y_i a particular ε_i is associated; all errors are considered to be drawn from the same population. This assumption may not be tenable; we will return to this in §18.

8.2.3 Fitting a regression line – 2

Task 34 : Fit a least-squares prediction line of straw as predicted by grain; display the model summary. •

Q38 : *What is the purpose of fitting this relation?* *Jump to A38* •

For this we use the **linear models** function `lm`; this is the workhorse of modelling in R. The generic `summary` function now produces a model summary; the `coefficients` **access function** extracts the regression line coefficients from the model object:

```
> model.straw.grain <- lm(straw ~ grain)
> summary(model.straw.grain)

Call:
lm(formula = straw ~ grain)

Residuals:
    Min      1Q  Median      3Q     Max
-2.0223 -0.3529  0.0104  0.3734  3.0342

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.8663     0.2387    3.63  0.00031 ***
grain         1.4305     0.0601   23.82  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.615 on 498 degrees of freedom
```

```
Multiple R-squared:  0.533,        Adjusted R-squared:  0.532
F-statistic:  567 on 1 and 498 DF,  p-value: <2e-16

> coefficients(model.straw.grain)

(Intercept)       grain
    0.86628     1.43050
```

This example illustrates the simplest **S model formula**, here `straw ~ grain`, which is signalled by the `~` operator, which can be read "is modelled by". So here the formula `straw ~ grain` can be read "straw yield is modelled by grain yield".

Q39 : *What is the linear least-squares relation between straw and grain yields?* *Jump to A39* •

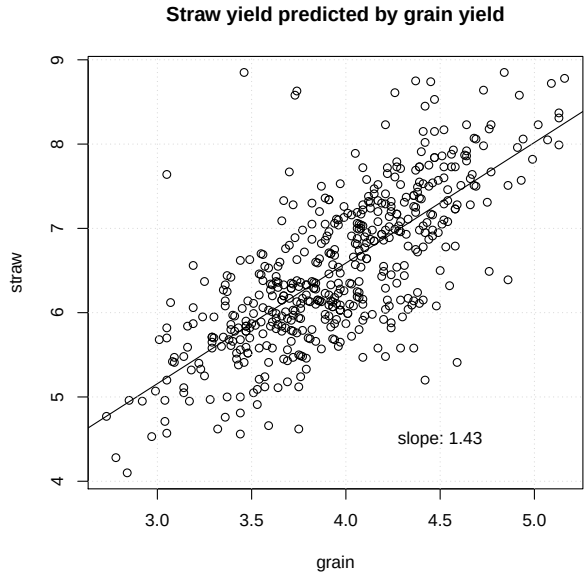
Q40 : *How much straw yield is expected if there is no grain yield? Does this seem realistic?* *Jump to A40* •

Q41 : *How much does the straw yield increase for each extra lb. of grain yield?* *Jump to A41* •

Q42 : *How much of the variability in straw yield is explained by this relation?* *Jump to A42* •

Task 35 : Display the scatterplot with the best-fit line superimposed. •

```
> plot(straw ~ grain)
> title("Straw yield predicted by grain yield")
> abline(model.straw.grain)
> grid()
> text(4.5, 4.5, paste("slope:",
+                      round(coefficients(model.straw.grain)[2], 2)))
```



Note the use of the `abline` function to add a line to the plot, the `text` function to place text on it, and the `title` function to add a title. Also note the use of the model formula syntax `straw ~ grain`. This is to be read "straw depends on grain".

8.2.4 Regression diagnostics

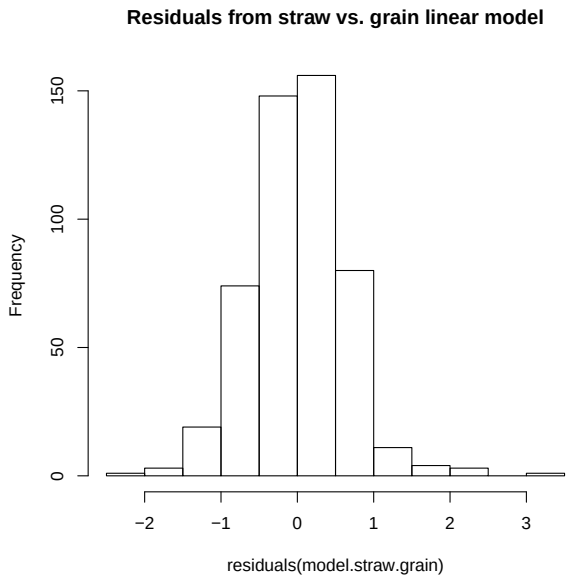
Of course, we have to treat any model with suspicion; for linear models there are some standard diagnostics. In particular, the hypothesis for the linear model is that the response is some deterministic linear function of the predictor, plus a **normally-distributed random error**:

$$y = \beta_0 + \beta_1 x + \epsilon$$

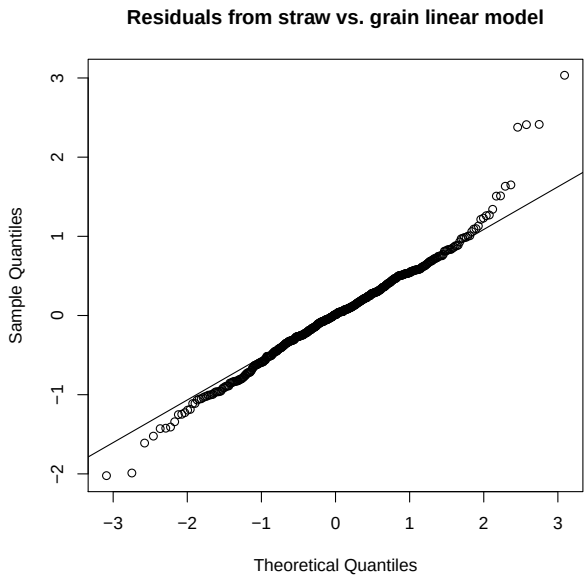
We will investigate whether the model we have just fit satisfies this criterion.

Task 36 : Display a histogram and quantile-quantile plot of the **regression residuals**; summarise these numerically. •

```
> hist(residuals(model.straw.grain),
+      main="Residuals from straw vs.\ grain linear model")
```




```
> qqnorm(residuals(model.straw.grain),
+        main="Residuals from straw vs.\ grain linear model")
> qqline(residuals(model.straw.grain))
> summary(residuals(model.straw.grain))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-2.0200 -0.3530  0.0104  0.0000  0.3730  3.0300
```



Q43 : *Do the residuals appear to be normally-distributed? Are they at least symmetrically-distributed?* *Jump to A43* •

We can test the normality of the residuals with a Shapiro-Wilks test:

```
> shapiro.test(residuals(model.straw.grain))

        Shapiro-Wilk normality test

data:  residuals(model.straw.grain)
W = 0.9767, p-value = 3.63e-07
```

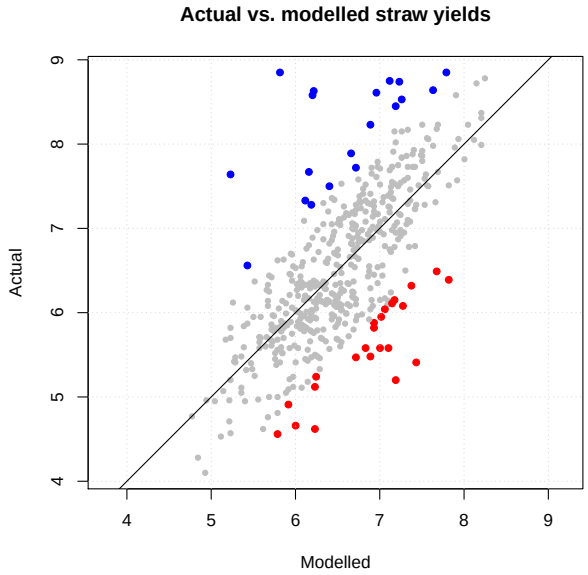
Q44 : *According to the Shapiro-Wilk test, should we reject the null hypothesis of normality for the residuals?* *Jump to A44*

•

Task 37 : Compare the fitted to actual values along a 1:1 line; highlight those that are more than 1 lb. plot-1in error. •

To colour points we use the `col` optional argument to the `plot` function, and the `ifelse` function to select a colour based on a **logical condition**:

```
> lim <- c(min(fitted(model.straw.grain), mhw$straw),
+          max(fitted(model.straw.grain), mhw$straw))
> plot(fitted(model.straw.grain), mhw$straw,
+      xlab="Modelled", ylab="Actual", asp=1,
+      xlim=lim, ylim=lim, pch=20,
+      col=ifelse(
+        (abs(fitted(model.straw.grain) - mhw$straw) < 1),
+        "gray",
+        ifelse(fitted(model.straw.grain) < mhw$straw, "blue","red")),
+      cex=ifelse(
+        (abs(fitted(model.straw.grain) - mhw$straw) < 1),1,1.3)
+ )
> title("Actual vs. modelled straw yields")
> grid()
> abline(0,1)
> rm(lim)
```



Task 38 : Some of the plots are poorly-fit by this relation; identify those with an absolute residual > 1.8 lb. straw and display their records, sorted by grain/straw ratio. •

This example illustrates the `abs` ("absolute value"), `which` ("which record numbers?"), `sort` ("sort values") and `order` ("order records") functions.

We first identify the records with the high absolute residuals, and show them:

```
> which.high.res <- which(abs(residuals(model.straw.grain)) >
+     1.8)
> sort(residuals(model.straw.grain)[which.high.res])

    184      35     295     337     311      15
-2.0223 -1.9891  2.3780  2.4107  2.4137  3.0342
```

Now we show these records in the data frame:

```
> high.res <- mhw[which.high.res, ]
> high.res[order(high.res$gsr), ]

     r  c grain straw     gsr
15   15  1  3.46  8.85 0.39096
337 17 17  3.05  7.64 0.39921
311 11 16  3.74  8.63 0.43337
295 15 15  3.73  8.58 0.43473
184  4 10  4.59  5.41 0.84843
35   15  2  4.42  5.20 0.85000

> rm(which.high.res, high.res)
```

Q45 : *Which plots have absolute residuals* > 1.8 *lb. straw? Which are too high and which too low, according to the relation?* *Jump to A45* •

Challenge: Repeat the analysis, but reversing the variables: model grain yield as a function of straw yield. Are the slopes inverses of each other? Why or why not?

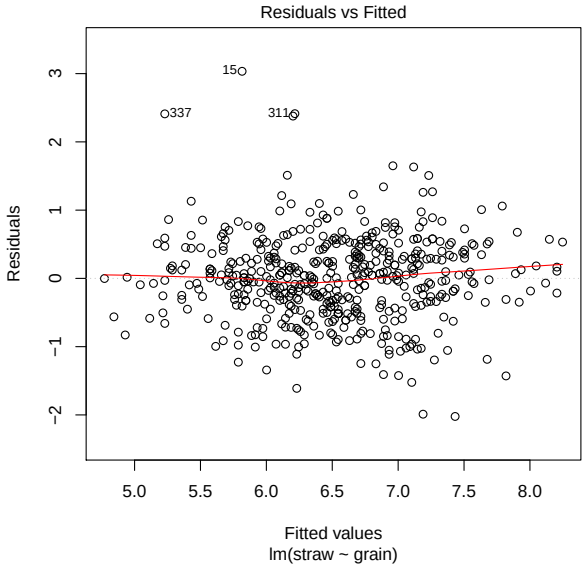
Further diagnostics The normality of residuals is one requirement for linear modelling; however there are other issues.

The generic `plot` function applied to a model result (i.e. object returned from a call to `lm`) gives a standard set of diagnostic plots, selectable with the `which` argument.

Plot type 1 is a plot of residuals vs. fitted values; there should be no relation between these. That is, whether a residual is high or low, positive or negative should not depend on the value fitted by the model. There should not be any trend; the smooth curve (fit by the `lowess` function) gives a visual impression of this – it should ideally be a horizontal line at 0.

This plot type also helps evaluate whether the variance of the residuals is constant across the range of the predictor, i.e., are they **homoscedastic** as required for fitting simple linear regression by OLS (Equation 8.1): looking vertically at any fitted value, the spread should be identical.

```
> plot(model.straw.grain, which = 1)
```



Plot type 2 is the Q-Q plot as we studied in the previous section.

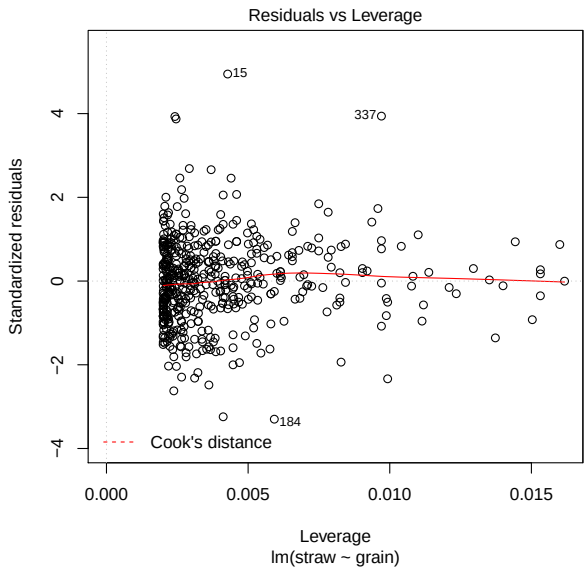
Plot type 5 shows the **leverage** of each observation and its corresponding residual.

Observations that are far from the centroid of the regression line can have a large effect on the estimated slope; they are said to have high *leverage*, by analogy with a physical lever. They are not necessarily in error, but they should be identified and verified; in particular, it is instructive to compare the estimated regression line with and without the high-leverage observations.

The leverage is measured by the *hat value*, which measures the overall influence of a single observation on the predictions. In diagnostic plot type 5 the abscissa ('x-axis') is the hat value. Two things are of interest:

- No extreme leverage, compared to other observations;
- Residuals at high-leverage observations should not be too much smaller than for other observations. Note that high-leverage observation "pull" the regression towards better fits, so their residuals are expected to be somewhat below average.

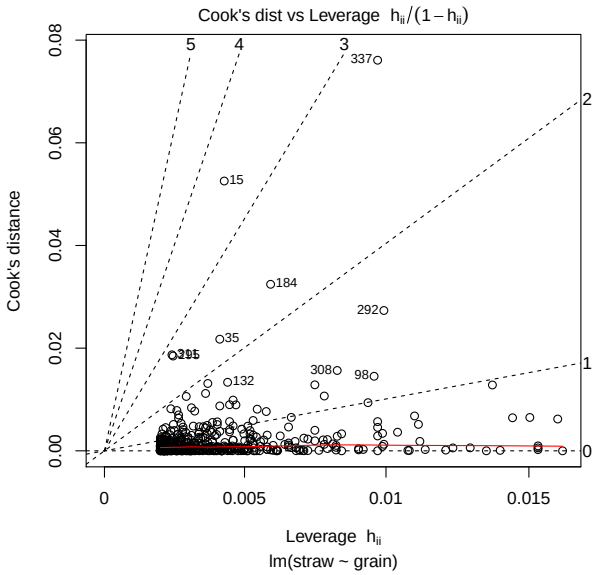
```
> plot(model.straw.grain, which = 5)
```



Plot type 6 is Cook's distance vs. leverage. Cook's distance measures how much the estimated parameter vector $\hat{\beta}$ shifts if a single observation is omitted. A high Cook's distance means that the observation has a large influence on the fitted model.

We also specify the number of extreme points to label with the `id.n` optional argument.

```
> plot(model.straw.grain, which = 6, id.n = 10)
```



This plot also shows contours of absolute standardised residuals, as labelled straight lines through the origin.

Q46 : *Which observations are flagged by their Cook's distance as having the most influence on the fit? Are these high-leverage observations?* *Jump to A46* •

We will compare this model to more complex ones in §12, below.

8.2.5 Prediction

Once the regression model is built, it can be used to **predict**: in this case, given a grain yield, what is the expected straw yield? And, how large or small could it be?

The `predict` generic method specialized to `predict.lm` for objects of class `lm`, such as the model we've just built. In addition to the model name, it requires a dataframe in which `predict` will look for variables with which to predict (in this case, a field named `grain`). In addition, `predict.lm` can return the **confidence interval** of the prediction, either of the fit itself or of values predicted from the equation.

There are two sources of prediction error:

1. The uncertainty of fitting the best regression line from the available data;
2. The uncertainty in the prediction, even with a perfect regression line, because of uncertainty in the process which is revealed by the regression (i.e. the inherent noise in the process) These correspond to the **confidence interval** and the **prediction interval**, respectively, both at

some level of risk of a Type I error, i.e., that the true value is not in the given range. Clearly, the second must be wider than the first.

The interpretation of these two intervals is as follows:

confidence : We are confident, with only a specified probability of being wrong, that the **expected** value of the response at the given value of the predictand is within this interval. In this case, if we sampled many plots with the same grain yield, this is the interval where the true mean value lies.

prediction : We are confident, with only a specified probability of being wrong, that any **single** value of the response at the given value of the predictand is within this interval. In this case, any one plot with the given grain yield should have a straw yield in this interval.

The estimation variance depends on (1) the variance of the regression $s^2_{Y.x}$ and (2) the distance $(x_0 - \bar{x})$ of the predictand at value x_0 from the centroid of the regression, $\bar{x}$. The further from the centroid, the more any error in estimating the slope of the line will affect the prediction:

$$s^2_{Y_0} = s^2_{Y.x}\left[1 + \frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}\right] \tag{8.7}$$

This shows that if we try to predict "too far" from the centroid, the uncertainty will be so large that any prediction is meaningless.

Note: The variance of the regression $s^2_{Y.x}$ is computed from the deviations of actual and estimated values at all the known points:

$$s^2_{Y.x} = \frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{8.8}$$

Task 39 : Compute the most likely value, and the 95% confidence and prediction intervals, of straw yields predicted by a grain yield of the mean, and the mean ± one and two standard deviations of the predictor. •

We first build a (very) small dataframe with the `data.frame` function, with a variable of the same name, and then use it as the `newdata` argument to `predict.lm`; the other arguments are the model name, the type of interval, and the risk of Type I error.

```
> t1 <- mean(mhw$grain); t2 <- sd(mhw$grain);
> p.frame <- data.frame(grain=seq(t1-2*t2, t1+2*t2, t2))
> predict(model.straw.grain, newdata=p.frame,
+         interval="confidence", level=0.95)

     fit    lwr    upr
1 5.2037 5.0828 5.3245
2 5.8592 5.7828 5.9357
3 6.5148 6.4608 6.5688
4 7.1704 7.0939 7.2468
5 7.8259 7.7051 7.9468

> predict(model.straw.grain, newdata=p.frame,
+         interval="prediction", level=0.95)
```

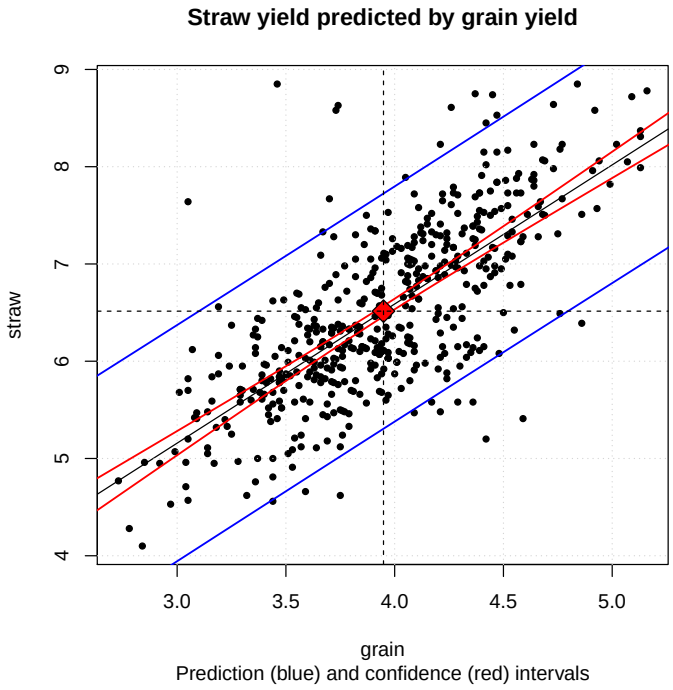
```
      fit    lwr    upr
1 5.2037 3.9898 6.4176
2 5.8592 4.6490 7.0695
3 6.5148 5.3057 7.7239
4 7.1704 5.9601 8.3806
5 7.8259 6.6120 9.0398
```

Q47 : *Which interval is wider? What happens to the width of the interval as the predictand is further from its centroid (mean)?* *Jump to A47* •

Task 40 : Display the scatterplot of straw vs. grain yields, with the best-fit line and the two confidence intervals, for grain yields from 2 to 6 lb. acre^{-1}, at 0.1 lb. resolution. •

The `seq` function builds the required sequence, and `data.frame` is again used to build a prediction frame. The `plot` function initiates the plot, and then the `title`, `grid`, `lines`, `points`, and `abline` functions add graphic or text elements.

```
> p.frame <- data.frame(grain = seq(from = 2, to = 6, by = 0.1))
> pred.c <- predict(model.straw.grain, newdata = p.frame,
+     interval = "confidence", level = 0.95)
> pred.p <- predict(model.straw.grain, newdata = p.frame,
+     interval = "prediction", level = 0.95)
> plot(straw ~ grain, data = mhw, pch = 20)
> title(main = "Straw yield predicted by grain yield",
+     sub = "Prediction (blue) and confidence (red) intervals")
> abline(model.straw.grain)
> grid()
> lines(p.frame$grain, pred.c[, "lwr"], col = 2, lwd = 1.5)
> lines(p.frame$grain, pred.c[, "upr"], col = 2, lwd = 1.5)
> lines(p.frame$grain, pred.p[, "lwr"], col = 4, lwd = 1.5)
> lines(p.frame$grain, pred.p[, "upr"], col = 4, lwd = 1.5)
> points(mean(grain), mean(straw), pch = 23, cex = 2, bg = "red")
> abline(h = mean(straw), lty = 2)
> abline(v = mean(grain), lty = 2)
```



Q48 : *Why is the confidence interval so narrow and the prediction interval so wide, for this relation?* *Jump to A48* •

Task 41 : Clean up from this subsection. •

```
> rm(t1, t2, p.frame, pred.c, pred.p)
```

8.3 Structural Analysis*

In §8.2.1 we modelled one of the two variables as as a response and the other as a predictor, and fit a line that best describes this relation. If we reverse the relation, what happens?

Task 42 : Compare the regression of strain yield on grain yield, with the regression of grain yield on straw yield. •

```
> model.grain.straw <- lm(grain ~ straw, data = mhw)
> summary(model.grain.straw)

Call:
lm(formula = grain ~ straw, data = mhw)

Residuals:
    Min      1Q  Median      3Q     Max
-1.3580 -0.2011  0.0004  0.1918  1.0527

Coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.5231     0.1028    14.8   <2e-16 ***
straw         0.3723     0.0156    23.8   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.314 on 498 degrees of freedom
Multiple R-squared:  0.533,        Adjusted R-squared:  0.532
F-statistic:  567 on 1 and 498 DF,  p-value: <2e-16

> summary(model.straw.grain)

Call:
lm(formula = straw ~ grain)

Residuals:
    Min      1Q  Median      3Q     Max
-2.0223 -0.3529  0.0104  0.3734  3.0342

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.8663     0.2387    3.63  0.00031 ***
grain         1.4305     0.0601   23.82  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.615 on 498 degrees of freedom
Multiple R-squared:  0.533,        Adjusted R-squared:  0.532
F-statistic:  567 on 1 and 498 DF,  p-value: <2e-16
```

Q49 : *Is the amount of variability explained by the two models the same? That is, does knowing straw yield give the same amount of information on grain yield as the reverse?* *Jump to A49* •

Intuitively it might seem that the slope of grain vs. straw would be the inverse of the slope of straw vs. grain. Is this the case?

Task 43 : Compute the slope of straw vs. grain as the inverse of the modelled slope of grain vs. straw, and compare with the modelled slope of straw vs. grain. •

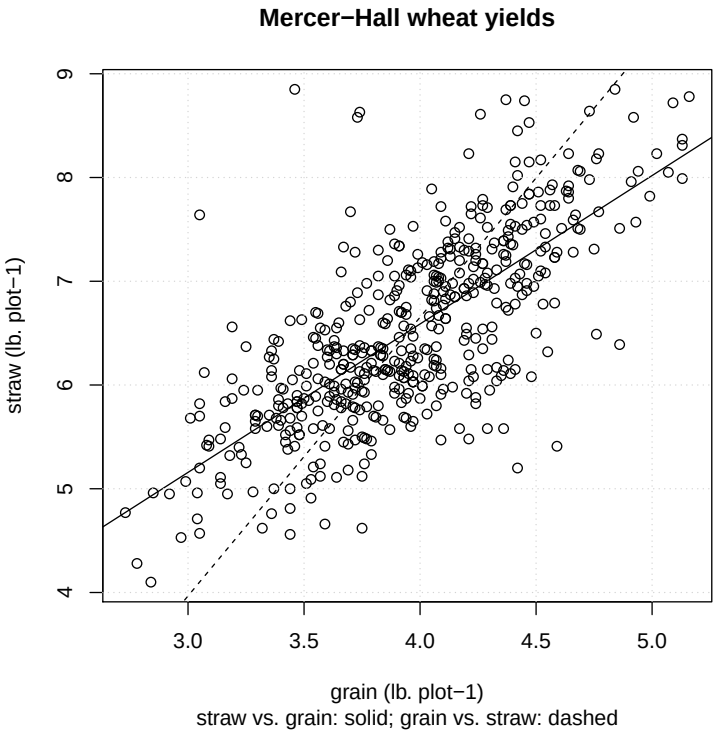
```
> coefficients(model.straw.grain)["grain"]

 grain
1.4305

> 1/coefficients(model.grain.straw)["straw"]

straw
2.686
```

We can visualise these on the scatterplot of straw vs. grain. The regression line of straw on grain can be directly plotted with `abline` on the model

object; the reverse regression must be inverted from coefficients extracted from its model object. The slope is just the inverse; the intercept is the straw yield corresponding to zero grain yield:

$$\begin{aligned} \text{grain} &= b_0 + b_1 \cdot \text{straw} \\ 0 &= \text{grain} \\ &\Downarrow \\ 0 &= b_0 + b_1 \cdot \text{straw} \\ &\Downarrow \\ \text{straw} &= -b_0/b1 \end{aligned}$$

```
> plot(straw ~ grain, pch = 1, main = "Mercer-Hall wheat yields",
+     xlab = "grain (lb. plot-1)", ylab = "straw (lb. plot-1)")
> title(sub = "straw vs. grain: solid; grain vs. straw: dashed")
> abline(model.straw.grain)
> beta <- coefficients(model.grain.straw)
> abline(-beta["(Intercept)"]/beta["straw"], 1/beta["straw"],
+     lty = 2)
> grid()
> rm(beta)
```



Q50 : *Do these two models give the same straw vs. grain relation? Why not?* *Jump to A50* •

So, the regression of two variables on each other depends on which variables is considered the predictor and which the predictand. If we are predicting,

this makes sense: we get the best possible prediction. But sometimes we are interested not in prediction, but in *understanding* a relation between two variables. In the present example, we may ask what is the true relation between straw and grain in this wheat variety? Here we assume that this relation has a common cause, i.e. plant growth processes affect the grain and straw in some systematic way, so that there is a consistent relation between them. This so-called **structural analysis** is explained in detail by Sprent [43] and more briefly by Webster [48] and Davis [8, pp. 218–219].

In structural analysis we are trying to establish the best estimate for a **structural** or **law-like** relation, i.e. where we hypothesise that $y = \alpha + \beta x$, where both x and y are mathematical variables. This is appropriate when there is no need to predict, but rather to understand. This depends on the prior assumption of a true linear relation, of which we have a noisy sample.

$$X = x + \xi \tag{8.9}$$
$$Y = y + \eta \tag{8.10}$$

That is, we want to observe X and Y, but instead we observe x with random error ξ and y with random error η. These errors have (unknown) variances σ^2_ξ and σ^2_η, respectively; the ratio of these is crucial to the analysis, and is symbolised as λ:

$$\lambda = \sigma^2_\eta / \sigma^2_\xi \tag{8.11}$$

Then the maximum-likelihood estimator of the slope $\hat{\beta}_{Y.X}$, taking Y as the predictand for convention, is:

$$\hat{\beta}_{Y.X} = \frac{1}{2s_{XY}}\left\{(s^2_Y - \lambda s^2_X) + \sqrt{(s^2_Y - \lambda s^2_X)^2 + 4\lambda s^2_{XY}}\right\} \tag{8.12}$$

Equation 8.12 is only valid if we can assume that the **errors in the two variables are uncorrelated**. In the present example, it means that a large random deviation for a particular sample of the observed straw yield from its "true" value does *not* imply anything about the random deviation of the observed grain yield from *its* "true" value.

The problem is that we don't have any way of knowing the true error variance ratio λ (Equation 8.11), just as we have no way of knowing the true population variances, covariance, or parameters of the structural relation. We estimate the **population** variances σ^2_X, σ^2_Y and covariance σ_{XY} from the sample variances s^2_x, s^2_y and covariance s_{xy}, but there is nothing we've measured from which we can estimate the **error** variances or their ratio. However, there are several plausible methods to estimate the ratio:

- If we can assume that the two error variances are **equal**, $\lambda = 1$. This may be a reasonable assumption if the variables measure the same property (e.g. both measure clay content in different soil layers), use the same method for sampling and analysis, and there is an *a priori* reason to expect them to have similar variability (heterogeneity among samples). However in this case there is no reason to expect equal variances.

- The two **error** variances may be estimated by the ratio of the **sample** variances: $\lambda \approx s_y^2/s_z^2$. That is, we assume that the ratio of variability in the measured variable is also the ratio of variability in their errors. For example, if the set of straw yields in a sample is twice as variable as the set of grain yields in the same sample, we would infer that the error variance of straw yields is also twice as much that for grain yields, so that $\lambda = 2$. But, these are two completely different concepts! One is a sample variance and the other the variance of the error in some random process. However, this ratio at least normalizes for different units of measure and for different process intensities. Using this value of λ computes the **Reduced Major Axis** (RMA), which is popular in biometrics.
- The variance ratio may be known from previous studies.

Task 44 : Compute the variance ratio of straw and grain yields. •

```
> var(straw)/var(grain)

[1] 3.8423
```

Q51 : *Is straw or grain yield more variable across the 500 plots? What is the ratio?* *Jump to A51* •

8.3.1 A user-defined function

Task 45 : Write an R function to compute $\hat{\beta}_{Y.X}$, given the two structural variables in the order predictand, predictor and the ratio of the error variances λ. •

This gives us an opportunity to see how to write a **user-defined function** in the S language. A function has:

1. a **name**, like any R object; this is written at the left of the `<-` assignment operator;
2. the command `function`;
3. a list of named **arguments** immediately following the function name, written within matched parentheses `( )`; if there are more than one argument, these are separated by commas (`,`);
4. the function **body** between matched braces `{` and `}`; this is R code which can refer to the named arguments and any other object defined in the workspace at the time the function is called;
5. an optional `return` command, whose argument is evaluated and returned as the value of the function; if no `return` command is given the

value at the end of the function is returned.

The `function` command creates a function, and we know it must have three arguments: the two structural variables and the ratio of the error variances. We can name the arguments as we wish. Here we choose `y` and `x` for the variables, and `lambda` for the ratio, and refer to these names in the body of the function. We also have to choose a name for the function object; here we choose a meaningful name, `struct.beta`.

```
> struct.beta <- function(y, x, lambda) {
+   a <- var(y)-lambda*var(x);
+   c <- var(x,y);
+   return((a + sqrt(a^2 + 4 * lambda * c^2))/(2*c))
+ }
```

This function is now defined in the workspace and available to be called with the required three arguments.

Task 46 : Apply this function to the straw vs. grain yields:

1. with $\lambda = 1$; this is the **orthogonal** estimate;
2. with λ as the variance ratio of straw and grain yields (assuming the error variance ratio equals the variables' variance ratio); this is the **proportional** estimate.

Compare with the slopes of the forward and reverse regressions. •

```
> print(paste("Forward:", round(coefficients(model.straw.grain)["grain"],
+     4)))

[1] "Forward: 1.4305"

> print(paste("Proportional:", round(struct.beta(straw,
+     grain, var(straw)/var(grain)), 4)))

[1] "Proportional: 1.9602"

> print(paste("Inverse proportional:", round(1/struct.beta(grain,
+     straw, var(grain)/var(straw)), 4)))

[1] "Inverse proportional: 1.9602"

> print(paste("Orthogonal:", round(struct.beta(straw, grain,
+     1), 4)))

[1] "Orthogonal: 2.4031"

> print(paste("Inverse orthogonal:", round(1/struct.beta(grain,
+     straw, 1), 4)))

[1] "Inverse orthogonal: 2.4031"

> print(paste("Reverse:", round(1/coefficients(model.grain.straw)["straw"],
+     4)))

[1] "Reverse: 2.686"
```

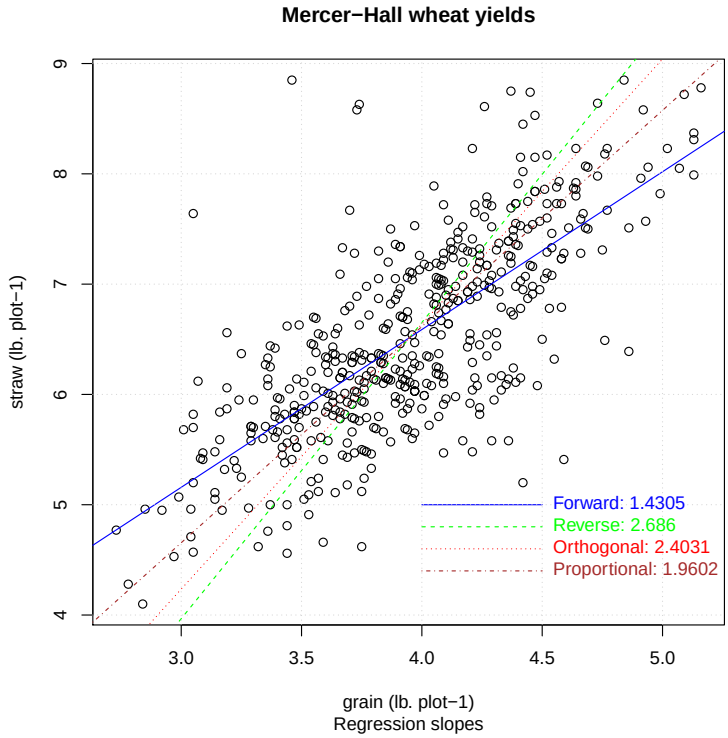
Note that all the estimates made with the `struct.beta` function are numerically between the slopes of the forward and inverse regressions, which can be considered the extremes (where all error is attributed to one or the other variable).

Task 47 : Plot the forward, reverse, orthogonal and proportional regression lines on one scatterplot of straw vs. grain yield. •

For the models fit with `lm` we can extract the coefficients; for the structural models we compute the slopes with our user-written function `struct.beta` and then their intercepts with the relation:

$$\hat{\beta}_0 = \hat{\mu}_y - \hat{\beta}_{y.x}\hat{\mu}_x \tag{8.13}$$

```
> plot(straw ~ grain, main="Mercer-Hall wheat yields",
+      sub="Regression slopes", xlab="grain (lb. plot-1)",
+      ylab="straw (lb. plot-1)")
> abline(model.straw.grain, col="blue")
> beta <- coefficients(model.grain.straw)
> abline(-beta["(Intercept)"]/beta["straw"]  , 1/beta["straw"],
+        lty=2, col="green")
> beta <- struct.beta(straw,grain,1)
> abline(mean(straw)-beta*mean(grain), beta, lty=3, col="red")
> beta <- struct.beta(straw,grain,var(straw)/var(grain))
> abline(mean(straw)-beta*mean(grain), beta, lty=4, col="brown")
> lines(c(4,4.5),c(5,5), lty=1, col="blue")
> lines(c(4,4.5),c(4.4,4.4), lty=4, col="brown")
> lines(c(4,4.5),c(4.6,4.6), lty=3, col="red")
> lines(c(4,4.5),c(4.8,4.8), lty=2, col="green")
> grid()
> text(4.5,5,paste("Forward:",
+              round(coefficients(model.straw.grain)["grain"],4)),
+      col="blue", pos=4)
> text(4.5,4.4,paste("Proportional:",
+                round(struct.beta(straw,grain,var(straw)/var(grain)),4)),
+      col="brown", pos=4)
> text(4.5,4.6, paste("Orthogonal:",
+                 round(struct.beta(straw,grain,1),4)),
+      col="red", pos=4)
> text(4.5,4.8,paste("Reverse:",
+                round(1/coefficients(model.grain.straw)["straw"],4)),
+      col="green", pos=4)
```



Q52 : *What do you conclude is the best numerical expression of the structural relation between straw and grain yields in this variety of wheat, grown in this field?* *Jump to A52* •

Challenge: Modify function `struct.beta` to return both the intercept and slope of the structural line[9], and use this to simplify the display of lines on the scatterplot.

8.4 No-intercept model*

In the simple linear regression of §8.2 the model is:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \tag{8.14}$$

This has two parameters: the slope β_1 and the intercept (value of the predictand when the predictor is 0) β_0. It is also possible to fit the model without an intercept, i.e., the linear relation is forced through the origin $(0, 0)$. The equation becomes:

$$y_i = \beta x_i + \varepsilon_i \tag{8.15}$$

There is then only a slope to be estimated, since the intercept is fixed at 0. These are termed **no-intercept** models.

Q53 : *Why might this have some logic in the case of predicting straw yield from grain yield?* *Jump to A53* •

[9] Hint: use the `c` "make a list" function

There are some mathematical implications of a no-intercept model.

- The mean residual is (in general) not zero;
- The residual sum-of-squares is (in general) larger than for a model with intercept;
- The usual formula for goodness-of-fit is not appropriate (§8.4.2).

Even if we know from nature that the relation must include $(0, 0)$, this takes away a degree of freedom from the fit, and gives a poorer fit in the range of observed responses, if this does not include $y = 0$.

A no-intercept model may be appropriate when:

1. There are physical reasons why the relation must include $(0, 0)$; e.g., no straw → no grain is possible (but not vice-versa!);
2. If a negative prediction should be avoided; e.g., it is impossible to have negative straw or grain in a plot[10];
3. If the range of the observations covers or approaches $(0, 0)$; otherwise we are assuming a linear form from the origin to the range of our data, when it may have some other form, e.g., exponential, power . . . ; there is no evidence for choosing a linear form **near the origin**;
4. If, after fitting a with-intercept model, the null hypothesis $H_0 : \beta_0 = 0$ in a linear regression with intercept can not be disproven (t-test of the coefficient), and we want to simplify the relation, we may then choose to re-fit with a no-intercept model.

8.4.1 Fitting a no-intercept model

In a no-intercept model, the slope $\hat{\beta}_{Y.X}$ can not be estimated from the sample covariance s_{XY} and variance of the predictand s_X^2, because the (co)variances are relative to means, which we can not compute; this is because the fixed intercept removes this degree of freedom.

Instead, the slope is computed by minimizes the RSS, again by orthogonal projection: $\mathbf{b} = [\mathbf{x}'\mathbf{x}]^{-1}[\mathbf{x}'\mathbf{y}]$, where the design matrix $\mathbf{x}$ here does *not* have an initial column of 1's, just a column of x_i. In the univariate case this reduces to $\sum x_i y_i / \sum x_i^2$.

Task 48 : Fit a no-intercept model of straw yield predicted by grain yield and summarize it. •

In the R model formulas, absence of the intercept is symbolized by the term `-1` in the formula supplied to the `lm` function:

```
> model.straw.grain.0 <- lm(straw ~ grain - 1)
> summary(model.straw.grain.0)
```

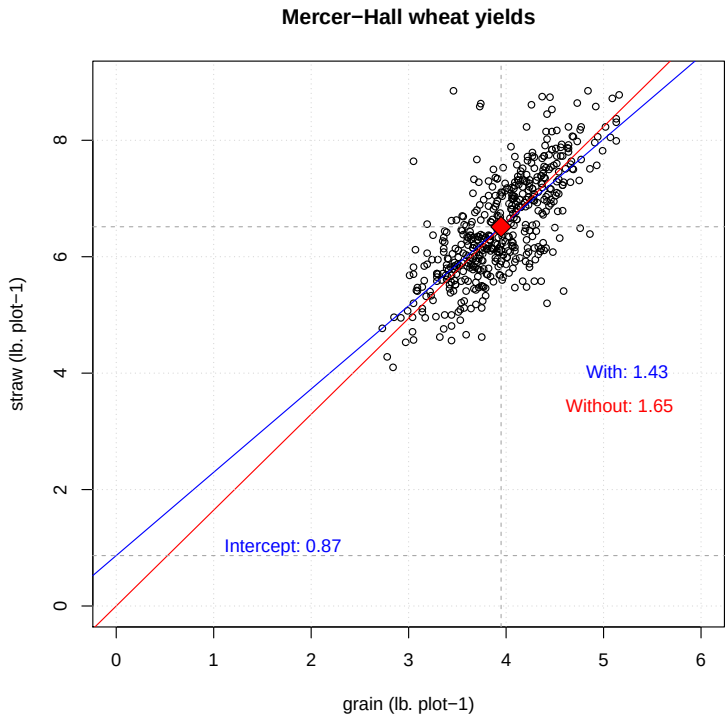
[10] But this can also be avoided by setting any negative predictions to zero.

```
Call:
lm(formula = straw ~ grain - 1)

Residuals:
    Min      1Q  Median      3Q     Max
-2.1496 -0.3660  0.0292  0.3657  3.1515

Coefficients:
      Estimate Std. Error t value Pr(>|t|)
grain    1.647      0.007     235   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.622 on 499 degrees of freedom
Multiple R-squared:  0.991,        Adjusted R-squared:  0.991
F-statistic: 5.54e+04 on 1 and 499 DF,  p-value: <2e-16
```

Q54 : *What is the slope of the relation? Does this differ from the* β_1 *coefficient of the with-intercept model? Why?* *Jump to A54* •

Task 49 : Display a scatterplot of the straw vs. grain yields, with the with- and no-intercept lines superimposed. Show the origin $(0, 0)$. Also show the centroid of the points. •

```
> plot(straw ~ grain, main="Mercer-Hall wheat yields",
+      xlab="grain (lb. plot-1)", ylab="straw (lb. plot-1)",
+      xlim=c(0,ceiling(max(grain))),
+      ylim=c(0, ceiling(max(straw))), cex=0.8)
> abline(model.straw.grain, col="blue")
> abline(model.straw.grain.0, col="red")
> grid()
> text(4.5,4, paste("    With:",
+              round(coefficients(model.straw.grain)["grain"],2)),
+      col="blue", pos=4)
> text(4.5,3.4,paste("Without:",
+               round(coefficients(model.straw.grain.0)["grain"],2)),
+      col="red", pos=4)
> abline(v=mean(grain), col="darkgray", lty=2)
> abline(h=mean(straw), col="darkgray", lty=2)
> points(mean(grain), mean(straw), cex=2, pch=23, bg="red")
> abline(h=coefficients(model.straw.grain)["(Intercept)"],
+        col="darkgray", lty=2)
> text(1,1,paste("Intercept:",
+           round(coefficients(model.straw.grain)["(Intercept)"],2)),
+      col="blue", pos=4)
```



Task 50 : Confirm that the mean residual of the no-intercept model is not zero, whereas that for the with-intercept model is. •

```
> mean(residuals(model.straw.grain.0))

[1] 0.011491

> mean(residuals(model.straw.grain))

[1] 1.7643e-16
```

Q55 : *Is the no-intercept model appropriate in this case?* *Jump to A55* •

8.4.2 Goodness-of-fit of the no-intercept model

The coefficient of determination R^2 for no-intercept model is in general *not* a good measure of fit, and is usually massively inflated, for the following reason.

Since there is no intercept in the design matrix, the total sum of squares (TSS) must be computed relative to zero: $TSS = \sum_{i=1}^{n}(y_i - 0)^2$, rather than relative to the sample mean $\bar{y}$: $TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2$. We still define R^2 as:

$$R^2 \quad = \quad 1 - \frac{\text{RSS}}{\text{TSS}}$$

But since the TSS is computed relative to zero, it tends to be quite high (no compensation for the sample mean), so even though the residual sum of squares (RSS) is larger than if an intercept is included, the R^2 tends to be very high.

Task 51 : Compute the coefficient of determination R^2 and the root mean squared error for the no- and with-intercept models. •

First, we compute R^2 directly from the definitions:

```
> (TSS <- sum((straw - mean(straw))^2))

[1] 402.67

> (TSS0 <- sum(straw^2))

[1] 21624

> (RSS <- sum(residuals(model.straw.grain)^2))

[1] 188.22

> (RSS0 <- sum(residuals(model.straw.grain.0)^2))

[1] 193.19

> (R2 <- 1 - (RSS/TSS))

[1] 0.53258

> (R20 <- 1 - (RSS0/TSS0))

[1] 0.99107

> rm(TSS, TSS0, RSS, RSS0, R2, R20)
```

Notice how the total sums of squares is much higher for the no-intercept model, because it is relative to 0 rather than the sample mean. The residual sum of squares is a bit higher, because the fit through the points is not so close when an intercept is not allowed; however, in this case (and in general) the RSS is only a bit higher.

Second, we show the R^2 computed with the model; note that this is adjusted for the number of model parameters and sample size.

```
> summary(model.straw.grain.0)$adj.r.squared

[1] 0.99105

> summary(model.straw.grain)$adj.r.squared

[1] 0.53164
```

Q56 : *(1) What is the relation between the adjusted and raw R^2 for both models? Compare the amount of adjustment; are they the same? Why not?*

(2) What happens to the R^2 in this case when the intercept is removed from the model? Is this a realistic view of the success of the model? *Jump to A56* •

We also compute the root mean squared error (RMSE), i.e., lack of fit, from the RSS and the number of observations, for both models.

```
> sqrt(sum(residuals(model.straw.grain)^2)/(length(straw)))

[1] 0.61354

> sqrt(sum(residuals(model.straw.grain.0)^2)/(length(straw)))

[1] 0.6216
```

Q57 : *What happens to the RMSE when the model is forced through* $(0, 0)$*? Why?* *Jump to A57* •

8.5 Answers

A32 : *Linear.* *Return to Q32* •

A33 : *The diagonals are the variances of the two variables, both in (lb. plot^{-1}) squared; the off-diagonals are the covariances between the two variables, in this case also in (lb. $^{-1}$) squared, because the two variables have the same units of measure.* *Return to Q33* •

A34 : *The summary statistics are quite similar, for both variables; the simulation reproduces the statistics of the actual data.* *Return to Q34* •

A35 : *The relations look quite similar; this supports the hypothesis. However, the bivariate simulation seems to have a slightly steeper slope than the actual data.* *Return to Q35* •

A36 : *The most probable value is 0.73 lb. plot^{-1}; the lower and upper confidence limits are 0.686 and 0.768 lb. plot^{-1}, respectively.*

Assessment of the strength is subjective and depends on the application field; the author would call this a moderate positive correlation. *Return to Q36* •

A37 : *No; both are caused by the same underlying process (plant growth in response to the environment), and neither is more under control of the experimenter. However, see the next question.* *Return to Q37* •

A38 : *It can be used to* **understand** *plant physiology: is grain yield a direct result of straw yield, or at least do large plants (lots of straw) tend to have high*

yield (lots of grain)? Practically, we could use this relation to predict straw yield on other plots where only grain was weighed; it is much easier to collect and weigh grain than straw. *Return to Q38* •

A39 : straw = 0.866 + 1.43 · grain *Return to Q39* •

A40 : 0.866 *lb.; maybe wheat this small would indeed have no grain, because the plants would be too weak to form grain.* *Return to Q40* •

A41 : 1.43 *lb. of straw increase for each lb. of grain increase.* *Return to Q41* •

A42 : *53.2% (the value of the adjusted* R^2*).* *Return to Q42* •

A43 : *It's clear that the highest residuals are too low and vice-versa; the histogram is somewhat peaked. The median residual is slightly biased (*−0.01*). The range is quite high, from* −2 *to* +3 *lb. plot*$^{-1}$*.* *Return to Q43* •

A44 : *The p-value (probability that rejecting the null hypothesis would be an error) is almost zero, so we should reject the null hypothesis: these residuals are* not *normally-distributed. This is due to the deviations at both tails.* *Return to Q44* •

A45 : *Plots 15, 337, 311 and 295 have very low grain/straw ratios, so the linear relation predicts too much grain; for plots 35 and 184 it's the reverse.* *Return to Q45* •

A46 : *Observations 292, 184, 15, and especially 337. Plots 337 and 292 also have high leverage. In the previous answer we saw that plots 15 and 337 have very low grain/straw ratios, so the linear relation predicts too much grain; for plot 184 it's the reverse. Plot 292 has high leverage and fairly high Cook's distance, but its standardised residual is not so high (*< 2*).* *Return to Q46* •

A47 : *The prediction intervals are much wider at all values of the predictand. Intervals further away from the centroid are increasingly wide.* *Return to Q47* •

A48 : *The confidence interval is narrow because the average linear relation is very consistent across its range (although, see §18 for some exceptions at the extreme values), so the estimate of the best-fit line is quite good. The prediction interval is wide because there is poor correlation in the sample set, i.e., a wide spread in straw yields for any observed grain yield. So this same uncertainty must appear in the prediction.* *Return to Q48* •

A49 : *The models explain the same proportion of the total variability, 0.532.*

Return to Q49 •

A50 : *The slopes are very different. The forward model gives a slope of 1.43 of straw vs. grain, whereas the inverse model gives a slope of 2.686. The reason is that the two regressions minimise different error sums-of-squares: in the forward model, residual straw yield, in the inverse model, residual grain yield. Each is the best predictive relation for its target variable.* Return to Q50 •

A51 : *The variance ratio is 3.842, that is, straw yields are almost four times as variable as grain yields. This is partly explained by the higher absolute values of all the yields: the medians are 6.36 for straw vs. 3.94 for grain.* Return to Q51 •

A52 : *The best estimate of the error variance ratio is the variable variance ratio, so the structural relation is 1.96 lb. straw for each lb. grain; or equivalently 0.51 lb. grain for each lb. straw. This is the best estimate of the plant morphology.* Return to Q52 •

A53 : *Physically, if there is no grain, there is no straw. Thus the point* $(0, 0)$ *is by definition part of the straw vs. grain response relation.* Return to Q53 •

A54 : *The slope is 1.65, considerably steeper than the slope of the with-intercept model, 1.43. This compensates for the intercept (here, forced to* 0*) being smaller than the fitted intercept of the full model, 0.87, which allows the line to have a shallower slope while passing through the centroid.* Return to Q54 •

A55 : *No, for three reasons. (1) The intercept from the full model is highly unlikely to be zero, so the no-intercept model is not appropriate; (2) the range of the observations is far from* $(0, 0)$ *so there is no reason to guard from negative predictions; (3) we have no evidence for the model form near the origin; the closest points are around* $(2.8, 4)$*.* Return to Q55 •

A56 : *(1) For both models the adjusted* R^2 *is lower than the raw* R^2*, because of the adjustment for the number of parameters used in the model. The difference for the no-intercept model is less, because only one, rather than two, parameters are used in the model.*

(2) The adjusted R^2 *increases from 0.53 to 0.99, i.e., almost* 1*. This is an artefact of the calculation and does not reflect the success of the no-intercept model.* Return to Q56 •

A57 : *The RMSE increases from 0.614 (with-intercept) to 0.622 (no-intercept) lb. acre*$^{-1}$*; this shows that the no-intercept line does not come as close, on average, to the points in the scatterplot. This is because the slope is not free to float at both ends (find the optimum intercept); instead it is forced through* $(0, 0)$ *as one point, and the other point is then the centroid of the point cloud in the scatterplot.* Return to Q57 •

9 Bivariate modelling: continuous response, classified predictor

A continuous variable can also be modelled based on some **classified** (discrete) predictor. In the present example we will consider the field half (North & South) to be such a predictor.

It has been suggested [45] that the North and South halves of the field had been treated differently prior to Mercer & Hall's experiment. To test this suggestion, we first have to code each plot according to its location in the field (N or S half); this is a **logical variable** (True or False, written in S as `TRUE` or `FALSE`, abbreviated `T` and `F`). Then we use this variable to split the field statistically and compare the yields for each half. Each plot is thus in one or the other **class**, in this case field half.

Task 52 : Add a logical field to the data frame to codes whether each plot falls in the north half or not. •

We first use a **logical expression** that evaluates to either `T` or `F` to create a **logical variable**, here named `in.north`, as a field in the data frame. This field codes whether teach plot falls in the north half or not. We then detach and re-attach the data frame to make the name `in.north` visible.

Recall from the description of the dataset in Appendix A that the rows ran W to E, with 25 plots in each row, beginning at 1 on the W and running to 25 at the E, and the columns run N to S with 20 plots in each, beginning at 1 on the N and running to to 20 at the S. So the N half of the field consists of the plots with row numbers from 1 to 10, inclusive.

```
> mhw <- cbind(mhw, in.north = (r < 11))
> str(mhw)

'data.frame':        500 obs. of  6 variables:
 $ r       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ c       : int  1 1 1 1 1 1 1 1 1 1 ...
 $ grain   : num  3.63 4.07 4.51 3.9 3.63 3.16 3.18 3.42 3.97 3.4 ...
 $ straw   : num  6.37 6.24 7.05 6.91 5.93 5.59 5.32 5.52 6.03 5.66 ...
 $ gsr     : num  0.57 0.652 0.64 0.564 0.612 ...
 $ in.north: logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
```

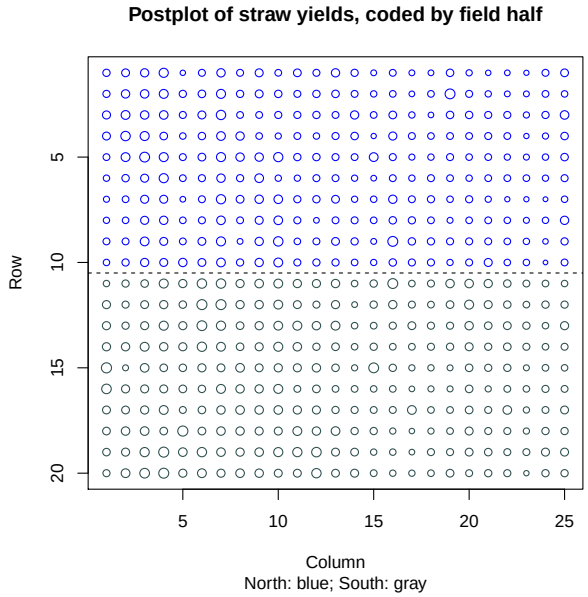
To make the new field `in.north` visible without having to specify the data frame name, we `detach` the frame, and then re-`attach` it:

```
> detach(mhw)
> attach(mhw)

> summary(in.north)

   Mode   FALSE    TRUE    NA's
logical     250     250       0
```

Task 53 : Display a post-plot of grain yields with the plots in the North

half coloured blue, those in the South coloured grey[11]. •

```
> plot(c, r, col = ifelse(in.north, "blue", "darkslategrey"),
+     cex = 1.3 * straw/max(straw), pch = 1, xlab = "Column",
+     ylab = "Row", ylim = c(20, 1), sub = "North: blue; South: gray")
> title(main = "Postplot of straw yields, coded by field half")
> abline(h = 10.5, lty = 2)
```



Note: Note the use of the `ylim` graphics argument, to specify that the labels of the y-axis run from 20 (lower left corner) to 1 (upper left corner); by default scatterplots drawn by the `plot` function assume the lowest-numbered row is the lower left. This is the usual case for scatterplots, but here we know the lowest-numbered row is at the N side.

9.1 Exploratory data analysis

We first compare the two halves with exploratory graphics; a suitable graph is the **boxplot**, created with the `boxplot` function.

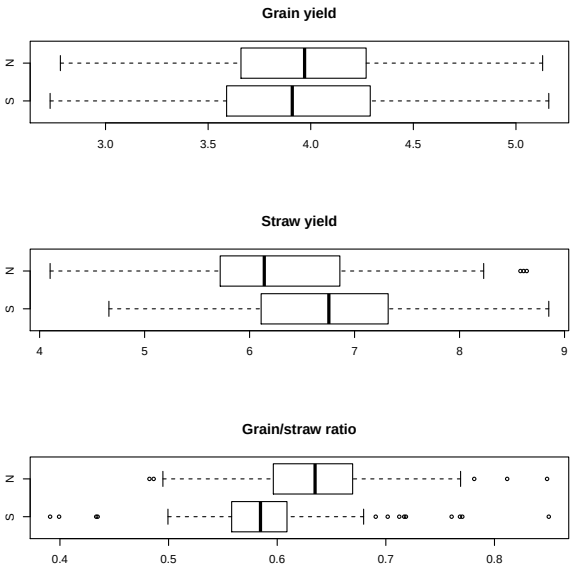
Task 54 : Compare the two field halves with box plots. •

To compare these on one graph, we split the graphics frame by specifying the number of rows and columns with the `mfrow` argument to the `par` ("graphics parameters") command. These plots look better if they are displayed horizontally, using the optional `horizontal` argument with the value `TRUE`. Note the use of the optional `names` argument to label the plots; these are `S` and `N` to represent the internal values `FALSE` and `TRUE` of the `in.north` classifier.

```
> par(mfrow=c(3,1))
> boxplot(grain ~ in.north, names=c("S", "N"),
+         main="Grain yield", horizontal=T)
```

[11] An obvious historical reference.

```
> boxplot(straw ~ in.north, names=c("S", "N"),
+         main="Straw yield", horizontal=T)
> boxplot(gsr ~ in.north, names=c("S", "N"),
+         main="Grain/straw ratio", horizontal=T)
> par(mfrow=c(1,1))
```



Q58 : *Do the two halves appear to have different ranges? medians? spreads? Comment for all three variables.* *Jump to A58* •

We then compare the two halves numerically:

Task 55 : Compare the summary statistics of the two halves for all the variables. Also compare their variances. •

Any function can be applied to subsets of a data frame with the `by` function. The first argument is the argument to the function, the second is the subset classifier, and the third the function to be applied:

```
> by(grain, in.north, summary)

in.north: FALSE
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.73    3.59    3.91    3.93    4.29    5.16
------------------------------------------------------
in.north: TRUE
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.78    3.66    3.97    3.96    4.27    5.13

> by(straw, in.north, summary)

in.north: FALSE
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   4.66    6.11    6.76    6.75    7.32    8.85
```

```
------------------------------------------------------
in.north: TRUE
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   4.10    5.73    6.14    6.28    6.86    8.64

> by(gsr, in.north, summary)

in.north: FALSE
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.391   0.558   0.585   0.586   0.609   0.850
------------------------------------------------------
in.north: TRUE
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.482   0.596   0.635   0.636   0.669   0.848

> by(grain, in.north, var)

in.north: FALSE
[1] 0.22162
------------------------------------------------------
in.north: TRUE
[1] 0.19876

> by(straw, in.north, var)

in.north: FALSE
[1] 0.74777
------------------------------------------------------
in.north: TRUE
[1] 0.75985

> by(gsr, in.north, var)

in.north: FALSE
[1] 0.0026123
------------------------------------------------------
in.north: TRUE
[1] 0.0030585
```

Q59 : *Do the two halves have different summary statistics? Is one half more variable than the other?* *Jump to A59* •

From the boxplots, it appears that the straw yield is, on average, higher in the S half; can we confirm this with a statistical test?

Task 56 : Test whether the straw yield is higher in the N half. •

There are two approaches that give the same answer for a binomial classified predictor: a two-sample t-test, and a one-way ANOVA.

9.2 Two-sample t-test

The simplest way to do this is with a **two-sample unpaired t test** of the difference between means, with the default **null hypothesis** that they are

identical. This only works when the classified variable is binary.

```
> t.test(straw[in.north], straw[!in.north])

        Welch Two Sample t-test

data:  straw[in.north] and straw[!in.north]
t = -6.0152, df = 497.97, p-value = 3.481e-09
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.61969 -0.31455
sample estimates:
mean of x mean of y
   6.2812    6.7484
```

Q60 : *Is there a significant difference in the means? What is the probability that this apparent difference is only by chance (i.e. a Type I error would be committed if we reject the null hypothesis)? What is the 95% confidence interval of the difference?* *Jump to A60* •

9.3 One-way ANOVA

Another way to analyse this is with a **one-way Analysis of Variance** (ANOVA). This can also deal with multivalued classified predictors, although in this case we only have a binary predictor.

Task 57 : Compute a one-way ANOVA for straw yield between field halves. •

This illustrates another use of the `lm` function, i.e. modelling a continuous response variable from a categorical (here, binary) predictor:

```
> model.straw.ns <- lm(straw ~ in.north, data = mhw)
> summary(model.straw.ns)

Call:
lm(formula = straw ~ in.north, data = mhw)

Residuals:
   Min     1Q Median     3Q    Max
-2.181 -0.608 -0.108  0.572  2.359

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    6.7484     0.0549  122.90  < 2e-16 ***
in.northTRUE  -0.4671     0.0777   -6.02  3.5e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.868 on 498 degrees of freedom
Multiple R-squared:  0.0677,        Adjusted R-squared:  0.0659
F-statistic: 36.2 on 1 and 498 DF,  p-value: 3.48e-09
```

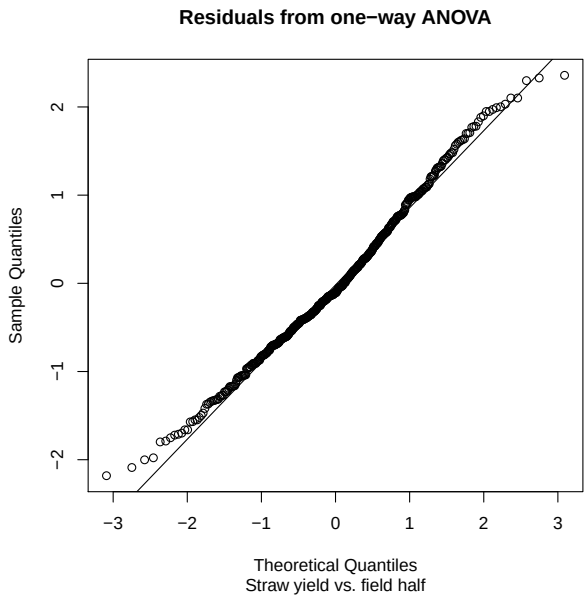
Q61 : *How much of the total variability in straw yield is explained by field half?* *Jump to A61* •

We can also see the results with a traditional ANOVA table:

```
> anova(model.straw.ns)

Analysis of Variance Table

Response: straw
           Df Sum Sq Mean Sq F value  Pr(>F)
in.north    1     27   27.28    36.2 3.5e-09 ***
Residuals 498    375    0.75
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

And of course we must check the regression diagnostics:

```
> qqnorm(residuals(model.straw.ns),
+        main="Residuals from one-way ANOVA",
+        sub="Straw yield vs. field half")
> qqline(residuals(model.straw.ns))
```



Q62 : *Are the model residuals normally-distributed? What does this imply about the process in the field?* *Jump to A62* •

We will compare this model to more complex ones in §12, below.

Challenge: Repeat the analysis of this section, but splitting the field into E-W halves, rather than N-S halves.[12] Do you reach similar conclusions about the differences between the field halves?

[12] Make sure to pick an appropriate colour scheme for the classified postplot.

9.4 Answers

A58 : *Grain yields appear to be almost identically distributed, although the S half is slightly more variable. Straw yields appear slightly higher in the S. The grain/straw ratio appears higher in the N. Variability between field halves seems similar for grain and straw, but the grain/straw ratio in the S half appears to have more total spread and boxplot outliers.* Return to Q58 •

A59 : *Grain: Almost identical summary statistics; Straw: The S is somewhat higher in all summary statistics, but the variability is almost the same; Grain/straw ratio: the N is higher in all summary statistics except the maximum, which is almost the same; the N is also somewhat more variable.* Return to Q59 •

A60 : *Yes, very highly significant; the N has a higher ratio than the S. the probability of a Type I error is almost 0. The 95% confidence interval is -0.62, -0.315, i.e. there is only a 2.5% chance that the difference in yields is not at least 0.315.* Return to Q60 •

A61 : *In the summary, the* `Adjusted R^2` *gives this proportion; here it is 0.066, a very low proportion. This shows that a model can be highly significant – the difference between class means is almost surely not zero – but numerically not important.* Return to Q61 •

A62 : *For the most part yes, but the tails (highest and lowest ratios) are not well-modelled: the absolute residuals are too high at both tails, especially the lower tail. This suggests that the extreme values are caused by some process that is beyond what is causing most of the "random" variability. This is sometimes called a "contaminated" process.* Return to Q62 •

10 Bootstrapping*

Most of the estimates of **population** statistics based on **sample** estimates, as computed in the previous sections, rely on assumptions that are difficult to verify.

For example, we saw in §8.1.1 that the parametric correlation coefficient is only justified for the case of a bivariate normal distribution. Although in this case the simulation based on the sample variance-covariance matrix seemed to support this assumption, we did notice several observations well outside the "envelope" expected if the distribution of the two variables is in fact bivariate normal. Also in the univariate modelling of §7 we could see that the distribution of grain yield was not completely normal: the highest yields are not as high, and the lowest not as low, as expected if the yields were normally distributed.

Further, any **confidence intervals** for both parametric and non-parametric statistics rely on a major assumption: that the **sample** estimate approaches the true **population** value **asymptotically**; that is, as the sample gets larger,

the estimate gets more precise in a smooth manner. All the classical confidence intervals depend on this assumption, which by its nature can not be proven. Further, the smaller the sample (typical of many studies), the less the asymptotic assumption is justified.

Another approach has been made possible by the advent of fast computers. This is **bootstrapping**, first proposed by Efron in the late 1970's [12]. Suppose we could repeat whatever experiment gave rise to the one dataset we have – in this case, it would be another uniformity trial with the same design as Mercer and Hall's original trial and under the same conditions (wheat variety, weather, soil, cultural practices ...). If we repeated the trial a large number of times, we'd have a direct estimate of the confidence interval: "Out of 1 024 trials, 95% of the correlation coefficients between grain and straw were above 0.67." This is another way of saying that we have 95% confidence that any future trial would show a correlation that high or higher.

But of course, we can't repeat most trials many times – either it is too expensive, would take too long, or is logistically impractical – in the present example we couldn't reproduce the same weather as in summer 1910 and even in the same field the soil properties could have changed due to the previous trial.

Efron's idea was to **simulate** many trials by **resampling** the **existing** trial. This is a subtle idea which at first seems like a trick. But, recall that the actual sample in hand is in fact the best non-parametric information about the true distribution in a larger population. In the present case we have 500 valid observations of grain and straw yield.

As a simple example, suppose we want to know the worst-case grain yield of a small plot (as in the experiment), say the yield with only 1% chance that a given yield would be smaller. With our current sample we can simply take the 1% quantile, symbolised as $q_{0.01}$; in this case with 500 observations, this is the mean of the 5$^{\text{th}}$ and 6$^{\text{th}}$-smallest values:

```
> quantile(mhw$grain, p = 0.01, type = 5)

   1%
2.945

> mean(sort(mhw$grain)[5:6])

[1] 2.945
```

> **Note:** The `type=5` specifies a piecewise linear function through the values; the default `type=7` uses a slightly different computation; see Hyndman and Fan [20] for an extensive discussion of different ways to compute quantiles of a continuous distribution from an empirical distribution.

But of course this is based on a single experiment with 500 observations. What if we could repeat this experiment many times?

Efron proposed a **non-parametric bootstrap** which uses the sample in hand as the population, and **simulates** a new sample from this "population" by picking a number of observations with equal probability, **with replacement**.

That is, a given observation can be picked multiple times[13]. Then this simulated sample is used to estimate the statistic of interest. We can do this 100's or 1000's of times and then summarise the statistic.

Bootstrapping is nicely introduced, without formulas, by Shalizi [41] and explained in more detail in the texts of Efron [11] and Davison and Hinkley [9]. The `boot` R package provides functions related to bootstrapping.

The basic function of the `boot` package is also named `boot`. The default is for a **non-parametric** bootstrap, specified with the optional `sim` argument as `sim="ordinary"` (the default).

Setting up the bootstrap can be quite complicated. The user must write a function that computes the statistic of interest for each bootstrap replicate. This function has a different form depending on the `boot` optional arguments; in this simplest case it has two arguments:

1. the **data**, typically a data frame;
2. a list of the **indices** of the rows (observations) selected for the bootstrap. Note that these may be repeated; the bootstrap samples **with replacement**.

10.1 Example: 1% quantile of grain yield

We begin with a simple example: the 1% quantile (i.e., estimate of the lowest grain yield, with 99% being greater). We estimated this from the single sample as:

```
> quantile(mhw$grain, p = 0.01, type = 5)

   1%
2.945
```

The bootstrapped estimate will compute this many times, each with a different simulated sample.

Task 58 : Write a function to return the statistic "1% quantile". •

We already wrote a user-defined function in §8.3.1. Again, we use the `function` command to create a function in the workspace. In this case it must have two arguments: the data frame and the selected indices. We choose meaningful names `data` and `indices`, and refer to these names in the body of the function. We choose a meaningful name for the function object: `boot.q01`.

```
> boot.q01 <- function(data, indices) {
+     obs <- data[indices, ]
+     return(quantile(obs$grain, p = 0.01, type = 5))
+ }
```

[13] otherwise we'd just get the original sample

In the function body, the line `obs <- data[indices,]` makes a data frame with rows corresponding to the bootstrap sample, with the same structure as the original frame. So then the function call to `quantile` refers to field `grain` in the resampled dataframe, which has the same structure as the original frame `mhw` but a different set of 500 observations from the resampling.

A function typically ends with the `return` command, which specifies the value to return to the caller; here it's the statistic of interest, computed on the replicate.

Task 59 : Estimate the population statistic "1% quantile" with a non-parametric bootstrap. •

We call `boot` with three arguments: the data, the function to compute the statistic on the replicate (i.e., the function we just wrote), and the number of replicates (argument `R`). Before calling `boot`, we must first load the optional `boot` package, by using the `require` function.

```
> require(boot)
> b.q01 <- boot(mhw, boot.q01, R = 1024)
> print(b.q01)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = mhw, statistic = boot.q01, R = 1024)


Bootstrap Statistics :
    original     bias    std. error
t1*    2.945 -0.0049512    0.071391
```

The output of `boot` shows:

`original` : the statistic (here,the 1% quantile) applied to the original dataset; in this example, this is the same as the result of the (non-bootstrapped) R command `quantile(mhw$grain, p=0.01, type=5)`;

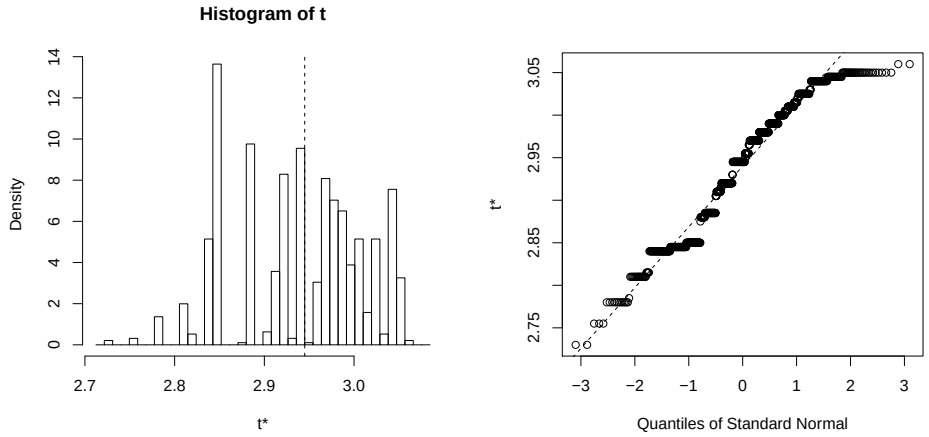
`bias` : the average difference of the bootstrapped estimates from the original value; this should be zero;

`standard error` : of the replicated statistic; the lower this is, the more consistent is the estimate.

Note that each time that `boot` is called, a **random** set of replicates is generated, so the statistics will vary.

The bootstrapped estimates can be summarised graphically with the `plot` method; this recognises the object of class `boot` and produces two plots: a histogram of the estimate (with the non-bootstrapped estimate shown as a dashed vertical line) and its normal Q-Q plot.

```
> plot(b.q01)
```



Q63 : *Describe the histogram of the 1 024 bootstrapped estimates of $q_{0.01}$, also with respect to the single (non-bootstrapped) estimate.* *Jump to A63*
•

Q64 : *Explain the "discontinuous" form of the histogram and Q-Q plot.* *Jump to A64* •

Bootstrapped confidence intervals With these in hand we can compute the confidence intervals of the statistic. This is one of the main applications of bootstrapping.

There are various ways to compute bootstrapped confidence intervals; the two most used are the **normal** approximation and the **basic** bootstrapped estimate; see Davison and Hinkley [9, Ch. 5] for a lengthy discussion.

Normal : Assumes that the empirical distribution of the statistic is asymptotic to a normal distribution, so the bias b and standard error s computed from the empirical bootstrap estimates can be used to compute a normal confidence interval for the population statistic t:

$$t - b - s \cdot z_{1-\alpha},\ t + b + s \cdot z_{\alpha} \tag{10.1}$$

where t is the statistic of interest and α specifies the $(1-2\alpha)$ interval; e.g., $\alpha = 0.025$ specifies the 0.95 (95%) interval.

Basic : When there is evidence that the empirical distribution of the statistic not asymptotic to normal (e.g., as revealed by the Q-Q normal probability plot of the estimates $t_1^*, t_2^*, \ldots t_n^*$), the normal approximation is not justified. Instead, the value of the quantile is extracted directly from the empirical distribution of the statistic.

Task 60 : Compute bootstrap estimates of the of the 1% quantile of grain

yield and the bootstrap estimate of the normal approximation and basic 95% confidence intervals. •

The original estimate of any bootstrap statistic is found in the `t0` field of the object returned by `boot`; the bootstrap estimates are in field `t`. So to get the best estimate we average the bootstrap estimates in field `t`:

```
> mean(b.q01$t)

[1] 2.94

> b.q01$t0

   1%
2.945
```

Q65 : *How does the average bootstrapped estimate of the 1% quantile compare to the estimate from the original sample?* *Jump to A65* •

The `boot.ci` function computes confidence intervals; the `conf` argument gives the probability 2α and the `type` argument specifies the type of computation.

```
> (b.q01.ci <- boot.ci(b.q01, conf = 0.95, type = c("norm",
+     "basic")))

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1024 bootstrap replicates

CALL :
boot.ci(boot.out = b.q01, conf = 0.95, type = c("norm", "basic"))

Intervals :
Level      Normal              Basic
95%   ( 2.81,  3.09 )   ( 2.84,  3.08 )
Calculations and Intervals on Original Scale
```

Q66 : *What are the bootstrapped estimates of the 95% confidence interval for the 1% quantile?* *Jump to A66* •

Q67 : *State the basic interval in terms of probability.* *Jump to A67* •

10.2 Example: structural relation between grain and straw

In §8.3 we investigated the structural relation between grain and straw; this should be an inherent property of the wheat variety.

Recall that the function to compute a structural relation is:

```
> struct.beta <- function(y, x, lambda) {
+   a <- var(y)-lambda*var(x);
```

```
+   c <- var(x,y);
+   return((a + sqrt(a^2 + 4 * lambda * c^2))/(2*c))
+ }
```

where the variance ratio `lambda` partitions the error between the two variables. For the reduced major axis (RMA) structural relation, the ratio of error variances is estimated as the ratio of sample variances; then the structural relation is estimated as a slope and intercept of:

```
> beta <- struct.beta(straw, grain, var(straw)/var(grain))
> alpha <- mean(straw) - beta * mean(grain)
```

Task 61 : Determine the most likely value and 95% confidence interval for the slope and intercept of the structural relation, with a non-parametric bootstrap. •

We first write the function to compute the statistics; note that the function `struct.beta` to compute the slope of the structural relation must be already defined. This function returns a **list** of two values; the `boot` function will then record both of these in field `t` of the `boot` object.

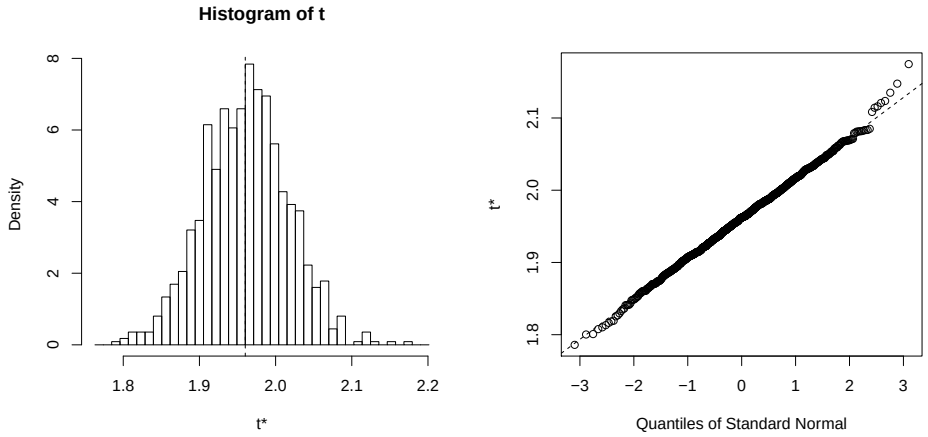
```
> boot.sr <- function (data, indices) {
+         obs <- data[indices,]
+         beta <- struct.beta(obs$straw,obs$grain,
+                             var(obs$straw)/var(obs$grain))
+         alpha <- mean(obs$straw)-beta*mean(obs$grain)
+         return(c(beta, alpha))
+ }
```

Then the bootstrap:

```
> b.sr <- boot(mhw, boot.sr, R = 1024)
> print(b.sr)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = mhw, statistic = boot.sr, R = 1024)


Bootstrap Statistics :
    original      bias    std. error
t1*   1.9602  0.00081772    0.055807
t2*  -1.2252 -0.00337799    0.218023
```

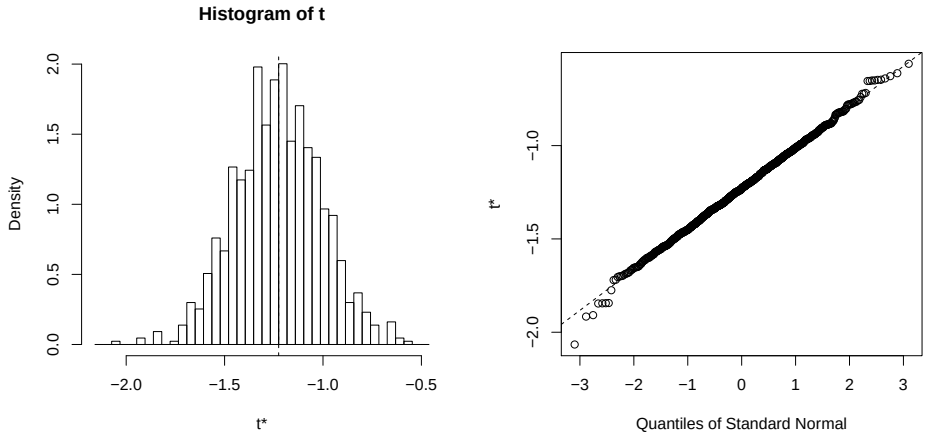
Visualise the bootstrap; first the slope:

```
> plot(b.sr, index = 1)
```



and then the intercept:

```
> plot(b.sr, index = 2)
```



Note the use of the optional `index` argument to select each of the two parameters in the `boot` object created by `boot`.

Q68 : *Do the bootstrap estimates of the two parameters of the structural relation appear to be normally-distributed?* *Jump to A68* •

Finally, from this the normal and basic confidence intervals, along with the mean (best estimate):

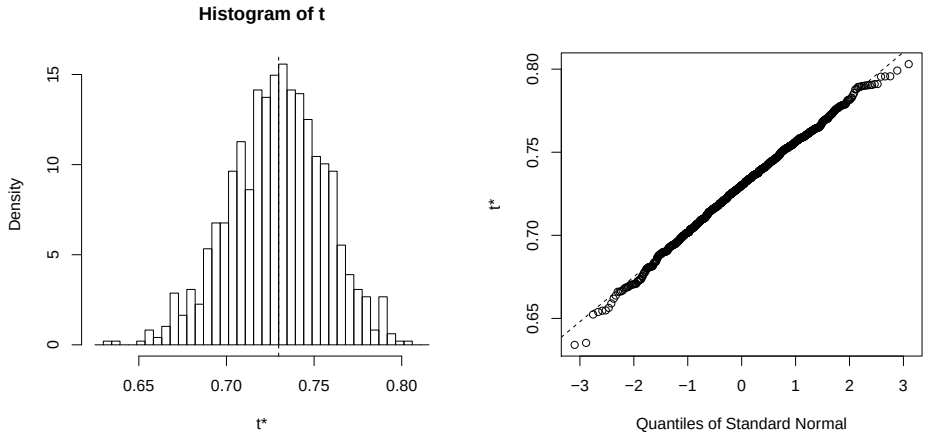
```
> mean(b.sr$t[, 1])

[1] 1.961

> (b.sr.ci <- boot.ci(b.sr, conf = 0.95, type = c("norm",
+     "basic"), index = 1))
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1024 bootstrap replicates

CALL :
boot.ci(boot.out = b.sr, conf = 0.95, type = c("norm", "basic"),
    index = 1)

Intervals :
Level      Normal              Basic
95%   ( 1.850,  2.069 )   ( 1.852,  2.069 )
Calculations and Intervals on Original Scale

> mean(b.sr$t[, 2])

[1] -1.2286

> (b.sr.ci <- boot.ci(b.sr, conf = 0.95, type = c("norm",
+     "basic"), index = 2))

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1024 bootstrap replicates

CALL :
boot.ci(boot.out = b.sr, conf = 0.95, type = c("norm", "basic"),
    index = 2)

Intervals :
Level      Normal              Basic
95%   (-1.649, -0.795 )   (-1.667, -0.799 )
Calculations and Intervals on Original Scale
```

Q69 : *How well do the two types of confidence intervals agree?* *Jump to A69* •

Challenge: Use non-parametric bootstrapping to estimate the 95% confidence interval and best estimate of the correlation coefficient between grain and straw yields for this wheat variety grown in the conditions of Mercer & Hall's experiment.

You should obtain the following plot of the bootstrap:



You should obtain these results for the basic confidence interval and best estimate of the population correlation coefficient ρ: $\hat{\rho} = 0.7291$, $\rho \in (0.6786, 0.7887)$. Compare this to the parametric (Pearson's) estimate: $\hat{\rho} = 0.7298$, $\rho \in (0.686, 0.7683)$. What can you conclude about the appropriateness of the parametric test and its confidence interval computed on the basis of a theoretical bivariate normal distribution, in this experiment?

10.3 Answers

A63 : *The single (non-bootstrapped) estimate is 2.945, shown by the dashed vertical line, is close to the middle of the histogram. The histogram of the bootstrapped estimates of $q_{0.01}$ is quite irregular. At low values there is a high frequency of some values and none of others; at high values a fairly uniform distribution. This is because of the few low values in the sample.* *Return to Q63* •

A64 : *The resampling only uses the known values; there are only 500 of these and so are not continuous.* *Return to Q64* •

A65 : *The bootstrapped estimate is 2.94, while the one-sample estimate is 2.945. They are quite close, although the bootstrap estimate is a bit lower (i.e., more conservative, predicting a lower value for the 1%-lowest grain yields).* *Return to Q65* •

A66 : *The normal approximation is 2.81 . . . 3.09; the basic bootstrap estimate is 2.84 . . . 3.08. In this case the basic estimate is a bit narrower than the normal approximation, probably because the very low values are not as likely as predicted by the normal distribution see the Q-Q normal probability plot of §7 where the lower tail is above the Q-Q line.* *Return to Q66* •

A67 : *There is only a 5% chance that in the population of all possible small plots grown according to the Mercer & Hall experimental protocol, under the same*

conditions (weather, soil ...), the lowest 1% of grain yields would be lower than 2.84 or higher than 3.08. *Return to Q67* •

A68 : *Yes, the parameters do seem normally-distributed. Thus the basic and normal confidence intervals should be almost the same.* *Return to Q68* •

A69 : *The two types of confidence intervals agree very closely; this is because the the bootstrap estimates are almost normally-distributed.* *Return to Q69* •

We are done with these models, some variables, and the boot package, so clean up the workspace:

```
> rm(model.grain.straw)
> rm(boot.q01, b.q01, b.q01.ci, boot.sr, b.sr, b.sr.ci)
> detach(package:boot)
```

11 Robust methods*

A **robust** inference is one that is not greatly disturbed by:

- a few unusual observations in the dataset, so-called **outliers**; or
- a "small" violation of model assumptions.

An example of the first case is a **contaminated** dataset, where some observations result from a **different process** then that which produced the others. In the present example this could be small areas of the wheat field with extreme stoniness, where a large amount of animal manure was stored, or where pests attacked. The issue here is that the observations do *not* all come from the same population, i.e., result of a single process. But of course that is not known prior to the experiment.

The second case (violation of model assumptions) can be the result of the first case (outliers) but also because an incorrect model form was selected to represent the underlying process.

Recall that in estimating linear regression parameters by least-squares, the assumed model is:

$$y_i = BX_i + \varepsilon_i$$

where the errors are identically and independently distributed (IID). If the regression diagnostic plots suggest that this assumption is violated, robust methods should be used.

11.1 A contaminated dataset

To illustrate the application of robust methods, we purposely add some **contamination** to our dataset. That is, we **simulate** the case where the wheat plants in some of the 500 plots were subject to some process other than "normal" growth as affected by soil, weather, management, and small attacks by a variety of pests. The contamination is that plots in one corner

of the field were attacked by deer, who ate most of the grain but did not affect the straw[14].

Q70 : *What could be some analysis options for the experimenter who observes different processes in the field?* *Jump to A70* •

In the present example, we suppose that we receive the dataset without having any opportunity to determine *a priori* whether several processes were active; we need to deal with the dataset as-is. Recall, the purpose of this experiment is to investigate the distribution of many replications of grain and straw yields when grown under identical conditions.

- Can we determine whether the conditions were "identical" except for identical random "noise"?
- How can we estimate true values for the "typical" situation when there is unknown contamination from another process?

Task 62 : Make a "contaminated" version of the dataset by setting the 16 northwest-most plots' grain yields (3.2%) to one-quarter of their actual yields. •

First, make a copy the "true" dataset:

```
> mhw.c <- mhw
```

Second, modify the NW corner. The dataset description (§A) states that the rows ran W to E, with 25 plots in each row, beginning at 1 on the W and running to 25 at the E, so that columns run S to N with 20 plots in each, running from 1 at the N to 20 at the S. We can find the 4 x 4 NW-most plots by selecting on row and column number:

```
> ix <- (mhw.c$r < 5) & (mhw.c$c < 5)
> rownames(mhw.c[ix, ])

 [1] "1"  "2"  "3"  "4"  "21" "22" "23" "24" "41" "42" "43" "44" "61"
[14] "62" "63" "64"

> subset(mhw.c, ix)

   r c grain straw     gsr in.north
1  1 1  3.63  6.37 0.56986     TRUE
2  2 1  4.07  6.24 0.65224     TRUE
3  3 1  4.51  7.05 0.63972     TRUE
4  4 1  3.90  6.91 0.56440     TRUE
21 1 2  4.15  6.85 0.60584     TRUE
22 2 2  4.21  7.29 0.57750     TRUE
23 3 2  4.29  7.71 0.55642     TRUE
24 4 2  4.64  8.23 0.56379     TRUE
41 1 3  4.06  7.19 0.56467     TRUE
```

[14] This happened to the one of author's field experiments during his MSc research at the Pennsylvania State University in the mid 1970's.

```
42 2 3  4.15  7.41 0.56005     TRUE
43 3 3  4.40  7.35 0.59864     TRUE
44 4 3  4.05  7.89 0.51331     TRUE
61 1 4  5.13  7.99 0.64205     TRUE
62 2 4  4.64  7.80 0.59487     TRUE
63 3 4  4.69  7.50 0.62533     TRUE
64 4 4  4.04  6.66 0.60661     TRUE
```

Note how the **logical** vector `ix` is a list of `TRUE` and `FALSE`, stating whether a given case (row, observation) in the dataframe is in the NW corner or not.

Now adjust the grain yields:

```
> mhw.c[ix, "grain"] <- mhw.c[ix, "grain"]/4
```

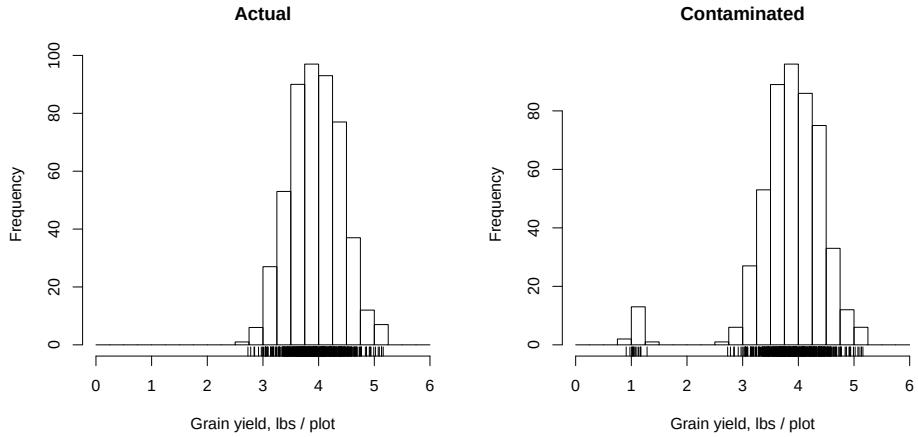
Task 63 : Summarize the effect of the contamination both numerically and graphically. •

First the numeric summary, also the standard deviation:

```
> summary(mhw$grain)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.73    3.64    3.94    3.95    4.27    5.16

> sd(mhw$grain)

[1] 0.45828

> summary(mhw.c$grain)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.908   3.590   3.920   3.850   4.260   5.160

> sd(mhw.c$grain)

[1] 0.67636
```

Second, side-by-side histograms “before” and “after”:

```
> par(mfrow=c(1,2))
> hist(mhw$grain, xlab="Grain yield, lbs / plot",
+      main="Actual", breaks=seq(0,6, by=.25))
> rug(mhw$grain)
> hist(mhw.c$grain, xlab="Grain yield, lbs / plot",
+      main="Contaminated", breaks=seq(0,6, by=.25))
> rug(mhw.c$grain)
> par(mfrow=c(1,1))
```



11.2 Robust univariate modelling

Typical parameters estimated for a single variable are some measure of the **center** and **spread**. For a normally-distributed variable the appropriate measures are the **mean** and **variance** (or **standard deviation**). But these are not robust. The robust measures include the **median** for central tendency and the **inter-quartile range** (IQR) for spread.

Task 64 : Compute the mean and standard deviation of the actual and contaminated grain yields. Also compute the robust measures. For all, compute the percent change due to contamination. •

The `mean`, `sd`, `median`, and `IQR` functions compute these:

```
> mean(mhw$grain)

[1] 3.9486

> mean(mhw.c$grain)

[1] 3.8458

> (mean(mhw.c$grain) - mean(mhw$grain))/mean(mhw$grain) *
+     100

[1] -2.6044

> sd(mhw$grain)

[1] 0.45828

> sd(mhw.c$grain)

[1] 0.67636

> (sd(mhw.c$grain) - sd(mhw$grain))/sd(mhw$grain) * 100

[1] 47.586
```

```
> median(mhw$grain)

[1] 3.94

> median(mhw.c$grain)

[1] 3.915

> (median(mhw.c$grain) - median(mhw$grain))/median(mhw$grain) *
+     100

[1] -0.63452

> IQR(mhw$grain)

[1] 0.6325

> IQR(mhw.c$grain)

[1] 0.67

> (IQR(mhw.c$grain) - IQR(mhw$grain))/IQR(mhw$grain) *
+     100

[1] 5.9289
```

Q71 : *How do the changes in the robust measures compare to those in the non-robust measures?* *Jump to A71* •

11.3 Robust bivariate modelling

In §8.1.1 we estimated the correlation between grain and straw yield. Recall that the parametric correlation (Pearson's) should only be used if the two variables are distributed approximately bivariate normally. In the original dataset this seemed to be a reasonable assumption.

In case the relation can not be assumed to be bivariate normal, methods must be used that do not depend on this assumption. These are called **non-parametric**; for correlation a widely-used metric is Spearman's ρ, which is the PPMC between the **ranks** of the observations.

Task 65 : Compare the ranks of the grain and straw yields for the first eight plots in the original data frame. •

The `rank` function returns the ranks of the values in a vector, from low to high. Ties can be handled in several ways; the default is to average the ranks. We display the data values and their ranks in a table:

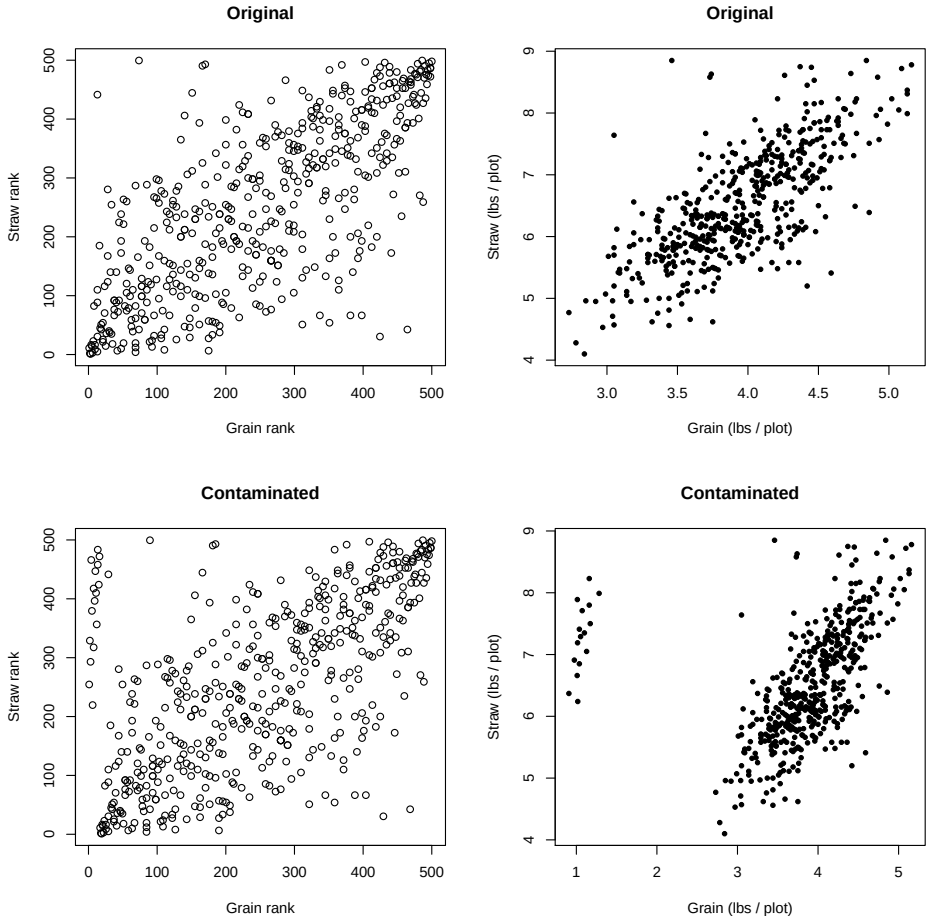
```
> head(cbind(mhw[, c("grain", "straw")], rank(mhw$grain),
+     rank(mhw$straw)), n = 8)

  grain straw rank(mhw$grain) rank(mhw$straw)
1  3.63  6.37           123.0           254.5
2  4.07  6.24           299.0           219.5
```

```
3  4.51  7.05          445.5          356.5
4  3.90  6.91          228.0          329.0
5  3.63  5.93          123.0          136.0
6  3.16  5.59           23.5           70.5
7  3.18  5.32           26.0           36.0
8  3.42  5.52           62.5           59.0
```

Q72 : *What are the data values and ranks of the first plot? Do the ranks of the two variables generally match?* *Jump to A72* •

Task 66 : Display a scatterplot of the ranks, alongside a scatterplot of the values of the **original** dataset, and below this, the same for the **contaminated** dataset •

```
> par(mfrow = c(2, 2))
> plot(rank(mhw$grain), rank(mhw$straw), xlab = "Grain rank",
+     ylab = "Straw rank", pch = 1, main = "Original")
> plot(mhw$grain, mhw$straw, xlab = "Grain (lbs / plot)",
+     ylab = "Straw (lbs / plot)", pch = 20, main = "Original")
> plot(rank(mhw.c$grain), rank(mhw.c$straw), xlab = "Grain rank",
+     ylab = "Straw rank", pch = 1, main = "Contaminated")
> plot(mhw.c$grain, mhw.c$straw, xlab = "Grain (lbs / plot)",
+     ylab = "Straw (lbs / plot)", pch = 20, main = "Contaminated")
> par(mfrow = c(1, 1))
```



Q73 : *How similar are the rank and value scatterplots in both cases? Which scatterplot type (rank or value) is more affected by contamination?* *Jump to A73* •

Q74 : *Does the scatterplot of values (straw vs. grain yields) of the contaminated dataset appear to be bivariate normal?* *Jump to A74*
•

Task 67 : Compute the numerical value of the Spearman's correlation between grain and straw yield, and compare it to the PPMC. Do this for both the original and contaminated datasets. •

The `cor` and `cor.test` functions have an optional `method` argument; this defaults to `pearson` but can be set explicitly:

```
> (c.p <- cor(mhw$grain, mhw.c$straw, method = "pearson"))

[1] 0.72978
```

```
> (cc.p <- cor(mhw.c$grain, mhw.c$straw, method = "pearson"))

[1] 0.35968

> (c.s <- cor(mhw$grain, mhw.c$straw, method = "spearman"))

[1] 0.71962

> (cc.s <- cor(mhw.c$grain, mhw.c$straw, method = "spearman"))

[1] 0.61684
```

Q75 : *Which method is most affected by the contamination?* *Jump to A75* •

11.4 Robust regression

In §8.2.1 we computed the linear regression of straw yield as modelled by grain yield.

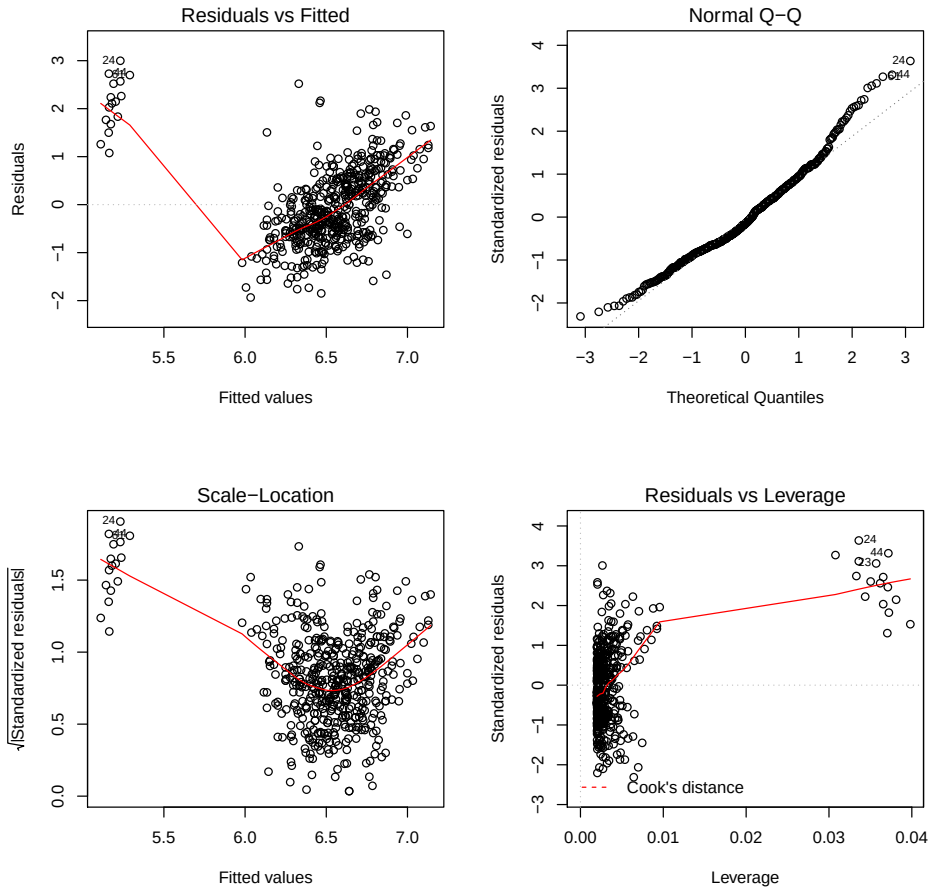
Task 68 : Repeat the regression fit for the contaminated dataset, and compare it with the original estimates of the regression model parameters. •

```
> print(model.straw.grain)

Call:
lm(formula = straw ~ grain)

Coefficients:
(Intercept)        grain
      0.866        1.430

> (model.straw.grain.c <- lm(straw ~ grain, data = mhw.c))

Call:
lm(formula = straw ~ grain, data = mhw.c)

Coefficients:
(Intercept)        grain
      4.678        0.478
```

Q76 : *How much did the regression parameters change?* *Jump to A76* •

Task 69 : Display the regression diagnostic plots for the regression fit for the contaminated dataset. Compare these with the plots from §8.2.4. •

```
> par(mfrow = c(2, 2))
> plot(model.straw.grain.c)
> par(mfrow = c(1, 1))
```

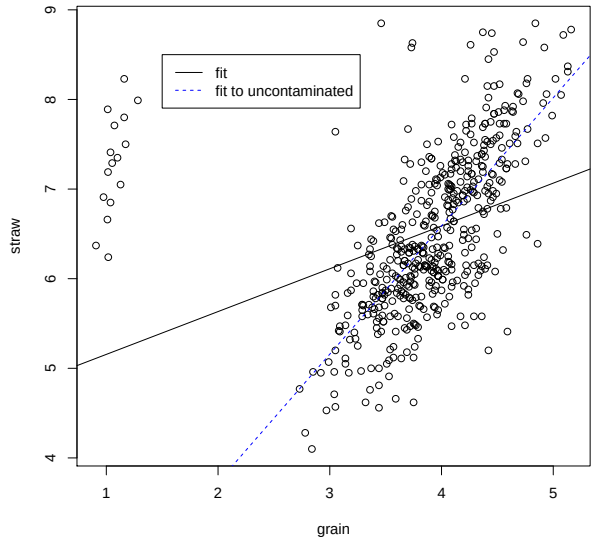


Q77 : *Are the assumptions for a least-squares linear fit justified? Which assumptions are violated?* *Jump to A77* •

Task 70 : Visualize the poor quality of the linear fit on a scatterplot of straw vs. grain yield of the contaminated dataset. Also show the linear fit to the original dataset. •

We use the generic `plot` method with a formula to specify the scatterplot, the `abline` function to add lines extracted from the linear models, and the `legend` function to place a legend on the graph. Note the use of the `with` function to specify an environment for the `plot` method.

```
> with(mhw.c, plot(straw ~ grain))
> abline(model.straw.grain.c)
> abline(model.straw.grain, lty=2, col="blue")
> legend(1.5, 8.5, legend=c("fit", "fit to uncontaminated"),
+        lty=1:2, col=c("black","blue"))
```



Q78 : *Looking at this scatterplot and the two lines, explain how the contamination affects the linear model fit.* *Jump to A78*

•

Many of the problems with parametric regression can be avoided by fitting a so-called "robust" regression line. There are many variants of this, well-explained by Birkes and Dodge [4] and illustrated with S code by Venables and Ripley [46]. Fox and Weisberg [16] is a good introduction to the concepts, illustrated by R code.

Here we just explore one method: `lqs` in the `MASS` package; this fits a regression to the "good" points in the dataset (as defined by some criterion), to produce a regression estimator with a high "breakdown" point. This method has several tuneable parameters; we will just use the default.

This is the so-called "least trimmed squares" (LTS) estimate of the slope vector β, by the criterion of minimizing:

$$\sum_{i=1}^{q} |y_i - \mathbf{x}_i\beta|^2_{(i)}$$

that is, the squared absolute deviations over some subset of the residuals, indicated by the subscript (i) and the summand q in the above formula. The smallest q residuals are chosen as some proportion, by default:

$$q = \lfloor (n + p + 1)/2 \rfloor$$

where p is the number of predictors and n the number of observations (cases). In this case these are 2 and 500, so $q = 251$. There is no analytical solution, the method **iterates**: first fitting a regular linear model, then examining the residuals and selecting the q smallest, then refitting

(thus obtaining a new set of residuals), selecting the smallest q of these, refitting, and so on until the fit converges.

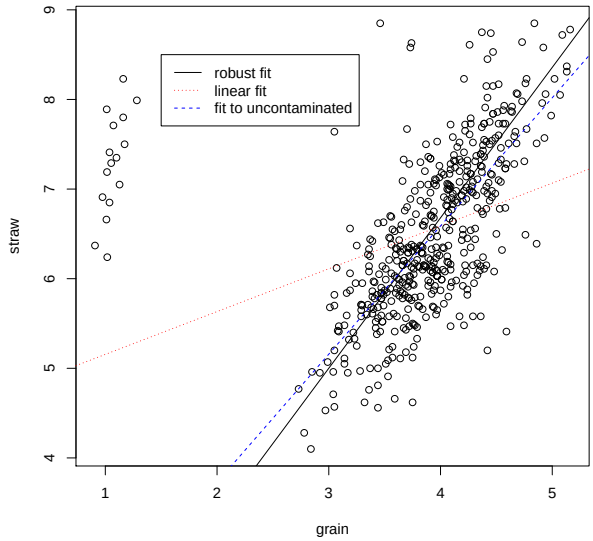
Task 71 : Load the `MASS` package and compute a robust regression of straw on grain yield. Compare the fitted lines and the coefficient of determination (R^2) of this with those from the least-squares fit. •

```
> require(MASS)
> (model.straw.grain.c.r <- lqs(straw ~ grain, data = mhw.c))

Call:
lqs.formula(formula = straw ~ grain, data = mhw.c)

Coefficients:
(Intercept)        grain
    -0.0351       1.6786

Scale estimates 0.548 0.531

> sqrt(mean(residuals(model.straw.grain)^2))

[1] 0.61354
```

The scale estimates are the scale of the error, similar to the residual mean square in a least-squares fit (shown above for comparison). There are two scale estimates, the first is based on the fit and is more conservative (larger).

Task 72 : Visualize the robust fit on a scatterplot of straw vs. grain yield of the contaminated dataset. Also show the linear fit to the contaminated and original datasets. •

```
> with(mhw.c, plot(straw ~ grain))
> abline(model.straw.grain.c.r)
> abline(model.straw.grain.c, lty = 3, col = "red")
> abline(model.straw.grain, lty = 2, col = "blue")
> legend(1.5, 8.5, legend = c("robust fit", "linear fit",
+     "fit to uncontaminated"), lty = c(1, 3, 2), col = c("black",
+     "red", "blue"))
```



Q79 : *Describe the effect of the robust fit.* *Jump to A79* •

Challenge: Compute the robust regression for the original (uncontaminated) dataset. How much does the fit change? What does this imply about the "outliers" identified in §4.2? Are they contamination (i.e., from another process) or just unusual (extreme) observations?

Challenge: Repeat the analysis of this section with a larger contaminated portion of the observations. At what point do the robust estimates also become unreliable?

11.5 Answers

A70 : *Some possibilities:*

(1) Remove the known contamination and analyze the "typical" case; state that the results only apply to these.

(2) Mark observations as "typical" or "contaminated" and use this factor in modelling. *Return to Q70*

•

A71 : *Since the contaminated observations are smaller than the original, the central tendency will be lower. The contaminated mean is lower by -2.6% whereas the contaminated median is hardly affected, only -0.6% lower.*

The measure of spread is much more affected by contamination: the standard deviation increases dramatically, by 48%, whereas the IQR only increases by 6%. *Return to Q71* •

A72 : *The first plot has grain and straw yields of 3.63 and 6.37, respectively; these are ranked 123 and 254.5 in their respective vectors. The grain yield is thus just under the first quartile (123 out of 500) whereas the straw yield is higher, just under the median (254 out of 500). For this plot, the straw yield ranks considerably higher than the grain yield.*

Overall, in the eight displayed ranks, there is a clear correspondence, but not very close. Return to Q72 •

A73 : *In both datasets the scatterplot of ranks is more diffuse; this is because the ranks throw away much information (the actual numeric values). In the case of bivariate normal distributions, the tails (extreme values) contribute disproportionately to the correlation.*

The rank plots are much less affected by contamination than the value plots; thus we expect a non-parametric correlation (based on ranks) to be less affected. Return to Q73 •

A74 : *The relation of straw and grain yields is definitely not bivariate normal. The group of observations at low grain yields (around 1 lb plot^{-1}) has very little relation with straw yield, and also is far from the scatter of the "normal" plots.* Return to Q74 •

A75 : *The parametric estimate of correlation (Pearson's) is greatly affected, by -51%; whereas the non-parametric estimate (Spearman's) is only changed by -14%.* Return to Q75 •

A76 : *The regression parameters (slope and intercept) both change drastically. The slope decreased from 1.43 to 0.48, i.e., from well above unity (straw yield increases more than grain yield) to well below (the reverse). The intercept then is adjusted to maintain the centroid of the regression; for the original dataset this is 0.87 lb plot^{-1} straw, i.e., a grain yield of zero corresponds to a straw yield of a bit under 1 lb plot^{-1} straw, but to compensate for the greatly reduced slope in the model of the contaminated dataset this increases to 4.68 lb plot^{-1} straw.* Return to Q76 •

A77 : *Several assumptions are clearly violated:*

(1) The residuals have a clear relation with the fits: low fits have very high positive residuals (these are the plots where the deer ate most of the grain); to compensate there is then a linear relation between residuals and fits from about 6 to 7.5 lbs plot^{-1} straw, i.e, the unaffected plots.

(2) The distribution of the residuals is not normal; this is especially clear at high residuals where the actual residuals are too high, i.e., large over-predictions (because the model is fitting mostly the ordinary relation, the low-grain plots are over-predicted).

(3) The high residuals are also associated with high leverage, i.e., the low-grain plots disproportionately affect the regression parameters (which is why the slope is

so low). Return to Q77 •

A78 : *This scatterplot clearly shows what we mean by "leverage": the 16 contaminated points "pull" the original line (shown as a dashed line on the plot) towards them. The further from the original centroid, the more the leverage, exactly as with a physical lever.* Return to Q78 •

A79 : *The robust fit models the uncontaminated portion of the dataset and completely ignores the contaminated observations. Thus it gives a true model of the dominant process of wheat growth. Note that the robust fit is a bit different from the fit to the uncontaminated dataset, because it also ignores the outliers in the original dataset.* Return to Q79 •

12 Multivariate modelling

In §8.2 we determined that the regression of straw yield on grain yields, over the whole field, was significant: i.e. straw yield does vary with grain yield. In §9 we determined that straw yields of the two field halves differ significantly. This raises the question whether it is possible and useful to combine these two findings, i.e. whether straw yield can be modelled as a function of both field half and grain yield.

There are four possibilities, from simplest to most complex:

1. Straw yields can be modelled by field half only;
2. Straw yields can be modelled by grain yield only;
3. Straw yields can be modelled by field half and grain yield, additively (i.e. with these as independent predictors);
4. Straw yields must be modelled by field half and grain yield, considering also the interaction between them (i.e. the relation between straw and grain yield differs in the two field halves).

These are tested with increasingly-complex linear models:

1. One-way ANOVA (or, two-sample t-test) of straw yield; following the evidence in the boxplot of §9.1 that seemed to show slightly higher yields in the north half; this was computed in §9.3 above;
2. Linear model of straw yield predicted by grain yield only, following the evidence of the scatterplot, ignoring field halves; this was computed in §8.2 above;
3. Linear model of straw yield predicted by field half (a categorical variable) and grain yield (a continuous variable), with an additive model (§12.1, below);
4. Linear model of straw yield predicted by field half (a categorical variable) and grain yield (a continuous variable), with an interaction model (§12.3, below);

We now compute all of these and compare the proportion of the total variation in straw yield that each explains.

Task 73 : Display the results of the field-half and bivariate regression models. •

Straw yield vs. field half was already computed in §9.3 above and saved as model object `model.straw.ns`:

```
> summary(model.straw.ns)

Call:
lm(formula = straw ~ in.north, data = mhw)

Residuals:
   Min     1Q Median     3Q    Max
-2.181 -0.608 -0.108  0.572  2.359

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    6.7484     0.0549  122.90  < 2e-16 ***
in.northTRUE  -0.4671     0.0777   -6.02  3.5e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.868 on 498 degrees of freedom
Multiple R-squared:  0.0677,        Adjusted R-squared:  0.0659
F-statistic: 36.2 on 1 and 498 DF,  p-value: 3.48e-09
```

Q80 : *Is there evidence that average straw yield differs in the two field halves? What proportion of the total variation in straw yield is explained by field half?* *Jump to A80* •

Straw yield vs. grain yield was computed in §8.2 above and saved as model object `model.straw.grain`:

```
> summary(model.straw.grain)

Call:
lm(formula = straw ~ grain)

Residuals:
    Min      1Q  Median      3Q     Max
-2.0223 -0.3529  0.0104  0.3734  3.0342

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.8663     0.2387    3.63  0.00031 ***
grain         1.4305     0.0601   23.82  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.615 on 498 degrees of freedom
```

```
Multiple R-squared:  0.533,        Adjusted R-squared:  0.532
F-statistic:  567 on 1 and 498 DF,  p-value: <2e-16
```

Q81 : *Is there evidence that straw yield varies with grain yield? What proportion of the total variation in straw yield is explained by grain yield?* *Jump to A81* •

Since both of these predictors (field half and grain yield) do explain some of the straw yield, it seems logical that a combination of the two, i.e. a **multivariate** model, might explain more than either separately. So, we now model straw yield vs. grain yield, also accounting for the overall difference between field halves.

12.1 Additive model: parallel regression

The simplest multivariate model is an **additive** model, also called **parallel regression** because it fits one regression line, but with the intercept at different levels, one for each field half.

Task 74 : Model straw yield as the combined effect of two independent predictors: field half and grain yield. •

We use the `lm` function, naming both predictors on the right-hand side of the model formula, combined with the `+` "additive effects" formula operator. Note this is *not* an arithmetic `+` (addition).

```
> model.straw.ns.grain <- lm(straw ~ in.north + grain,
+     data = mhw)
> summary(model.straw.ns.grain)

Call:
lm(formula = straw ~ in.north + grain, data = mhw)

Residuals:
    Min      1Q  Median      3Q     Max
-2.2548 -0.3189 -0.0276  0.3042  2.7871

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    1.0461     0.2178     4.8  2.1e-06 ***
in.northTRUE  -0.5132     0.0500   -10.3  < 2e-16 ***
grain          1.4499     0.0546    26.5  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.559 on 497 degrees of freedom
Multiple R-squared:  0.614,        Adjusted R-squared:  0.613
F-statistic:  396 on 2 and 497 DF,  p-value: <2e-16
```

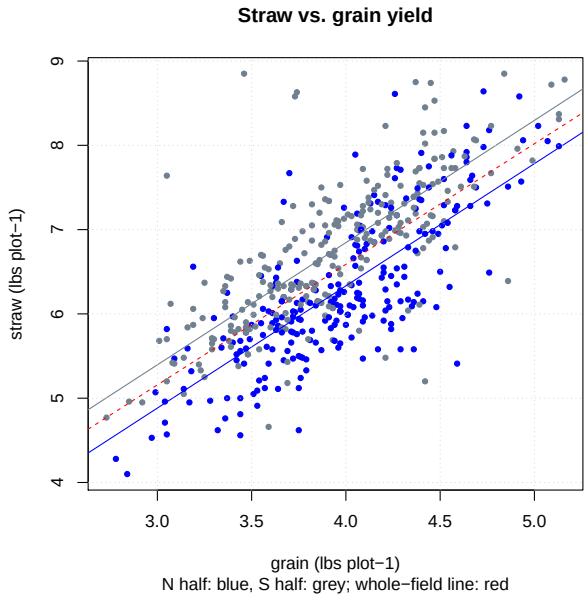
Q82 : *Is there evidence that both field half and grain yield are needed to*

predict straw yield? What proportion of the total variation in straw yield is explained by this model? *Jump to A82* •

Q83 : *According to the model summary, what is the difference in overall yields between the S and N halves? What is the slope of the regression line for straw vs. grain yield? Is this the same as the slope for the model that does not include field half?* *Jump to A83* •

In parallel regression (additive effects of a continuous and discrete predictor) there is only one regression line, which is displaced up or down for each class of the discrete predictor. Even though there are two predictors, we can visualize this in a 2D plot by showing the displaced lines.

```
> plot(straw ~ grain, col = ifelse(in.north, "blue", "slategray"),
+     pch = 20, xlab = "grain (lbs plot-1)", ylab = "straw (lbs plot-1)")
> title(main = "Straw vs. grain yield")
> title(sub = "N half: blue, S half: grey; whole-field line: red")
> abline(coefficients(model.straw.ns.grain)["(Intercept)"],
+     coefficients(model.straw.ns.grain)["grain"], col = "slategray")
> abline(coefficients(model.straw.ns.grain)["(Intercept)"] +
+     coefficients(model.straw.ns.grain)["in.northTRUE"],
+     coefficients(model.straw.ns.grain)["grain"], col = "blue")
> abline(model.straw.grain, lty = 2, col = "red")
> grid()
```



In the code for this plot, note the use of the `coefficients` function to extract the model coefficients.

12.2 Comparing models

Is a more complex model better than a simpler one? There are several ways to answer this, among which are:

- Compare the adjusted R^2 of the two models: this is the proportion of the variance explained;
- Directly compare hierarchical models with an Analysis of Variance.

Task 75 : Compare the adjusted R^2 of the three models. •

We've already seen these in the model summaries; they can be accessed directly as field `adj.r.squared` of the model summary:

```
> summary(model.straw.ns)$adj.r.squared

[1] 0.065864

> summary(model.straw.grain)$adj.r.squared

[1] 0.53164

> summary(model.straw.ns.grain)$adj.r.squared

[1] 0.61268
```

Q84 : *Which model explains the most variability?* *Jump to A84* •

Another way to compare two **hierarchical** models (i.e. where the more complex model has all the predictors of the simpler one) is with an analysis of variance: comparing variance explained vs. degrees of freedom. This is a statistical test, so we can determine whether the more complex model is provably better.

Task 76 : Compare the additive multivariate model to the two univariate models. •

The `anova` function can be used to compare the models:

```
> anova(model.straw.ns.grain, model.straw.ns)

Analysis of Variance Table

Model 1: straw ~ in.north + grain
Model 2: straw ~ in.north
  Res.Df RSS Df Sum of Sq   F Pr(>F)
1    497 155
2    498 375 -1      -220 704 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> anova(model.straw.ns.grain, model.straw.grain)

Analysis of Variance Table

Model 1: straw ~ in.north + grain
Model 2: straw ~ grain
```

```
  Res.Df RSS Df Sum of Sq   F Pr(>F)
1    497 155
2    498 188 -1     -32.9 105 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The `anova` function compares the **residual sum of squares** (RSS) of the two models; this is the amount of variability not explained by the model, so a lower RSS is better. It then computed the F-ratio between the two variances, and the probability that this large an F-value, with the **degrees of freedom** (d.f.) could occur by chance, if the null hypothesis of no model improvement is true. The probability of a Type I error (falsely rejecting a true null hypothesis) is reported as field `Pr(>F)`; lower is better.

Q85 : *Is the multivariate additive model provably better than either univariate model?* *Jump to A85*

•

12.3 Interaction model

We saw that the additive model is superior to either single-predictor model. However, there is also the possibility that both field half and grain yield help predict straw yield, but that the relation between straw and grain is different in the two halves; this is known as an **interaction**. This allows a different linear regression in each field half, rather than a parallel regression.

Q86 : *What is the difference between an additive and interaction model, with respect to processes in the field?* *Jump to A86* •

Task 77 : Model straw yield as the combined effect of two interacting predictors: field half and grain yield. •

We use the `lm` function, naming both predictors on the right-hand side of the model formula, combined with the `*` "interactive effects" formula operator. This is *not* an arithmetic `*` (multiplication).

```
> model.straw.ns.grain.i <- lm(straw ~ in.north * grain,
+     data = mhw)
> summary(model.straw.ns.grain.i)

Call:
lm(formula = straw ~ in.north * grain, data = mhw)

Residuals:
    Min      1Q  Median      3Q     Max
-2.2242 -0.3169 -0.0398  0.3136  2.7574

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)          1.2930     0.2979    4.34  1.7e-05 ***
in.northTRUE        -1.0375     0.4350   -2.39    0.017 *
grain                1.3872     0.0752   18.44  < 2e-16 ***
in.northTRUE:grain   0.1328     0.1094    1.21    0.225
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.559 on 496 degrees of freedom
Multiple R-squared:  0.615,        Adjusted R-squared:  0.613
F-statistic:  265 on 3 and 496 DF,  p-value: <2e-16
```

Q87 : *What do the four coefficients in the regression equation represent?* *Jump to A87* •

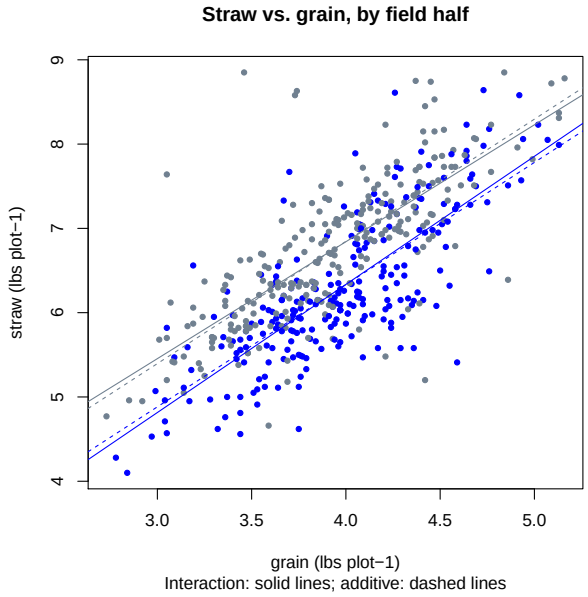
Q88 : *Are the two slopes (one for each field half) significantly different? Is an interaction model indicated? What does that imply about the processes in the field?* *Jump to A88* •

Even though the slopes are not significantly different, we show them graphically, to visualize how different they are.

Task 78 : Plot the two regressions (one for each field half). •

To do this, we use the `subset=` optional argument to the `lm` method to select just some observations, in this case, those in a zone. We plot each regression line (using the `abline` function on the model object returned by `lm`) and its associated points in different colours, using the `col` graphics argument. Dashed lines (using the `lty` graphics argument) show the parallel regression for the two field halves.

```
> plot(straw ~ grain, col = ifelse(in.north, "blue", "slategray"),
+     pch = 20, xlab = "grain (lbs plot-1)", ylab = "straw (lbs plot-1)")
> title(main = "Straw vs. grain, by field half")
> title(sub = "Interaction: solid lines; additive: dashed lines")
> abline(lm(straw ~ grain, subset = in.north), col = "blue")
> abline(lm(straw ~ grain, subset = !in.north), col = "slategray")
> abline(coefficients(model.straw.ns.grain)["(Intercept)"],
+     coefficients(model.straw.ns.grain)["grain"], col = "slategray",
+     lty = 2)
> abline(coefficients(model.straw.ns.grain)["(Intercept)"] +
+     coefficients(model.straw.ns.grain)["in.northTRUE"],
+     coefficients(model.straw.ns.grain)["grain"], col = "blue",
+     lty = 2)
```



Is this more complex interaction model significantly better than the additive model?

Task 79 : Compare the interaction and additive models by their adjusted R^2 and with an analysis of variance. •

```
> summary(model.straw.ns.grain)$adj.r.squared

[1] 0.61268

> summary(model.straw.ns.grain.i)$adj.r.squared

[1] 0.61305

> anova(model.straw.ns.grain.i, model.straw.ns.grain)

Analysis of Variance Table

Model 1: straw ~ in.north * grain
Model 2: straw ~ in.north + grain
  Res.Df RSS Df Sum of Sq    F Pr(>F)
1    496 155
2    497 155 -1     -0.46 1.47   0.23
```

Q89 : *Does the more complex model have a higher proportion of variance explained? Is this statistically significant?* *Jump to A89* •

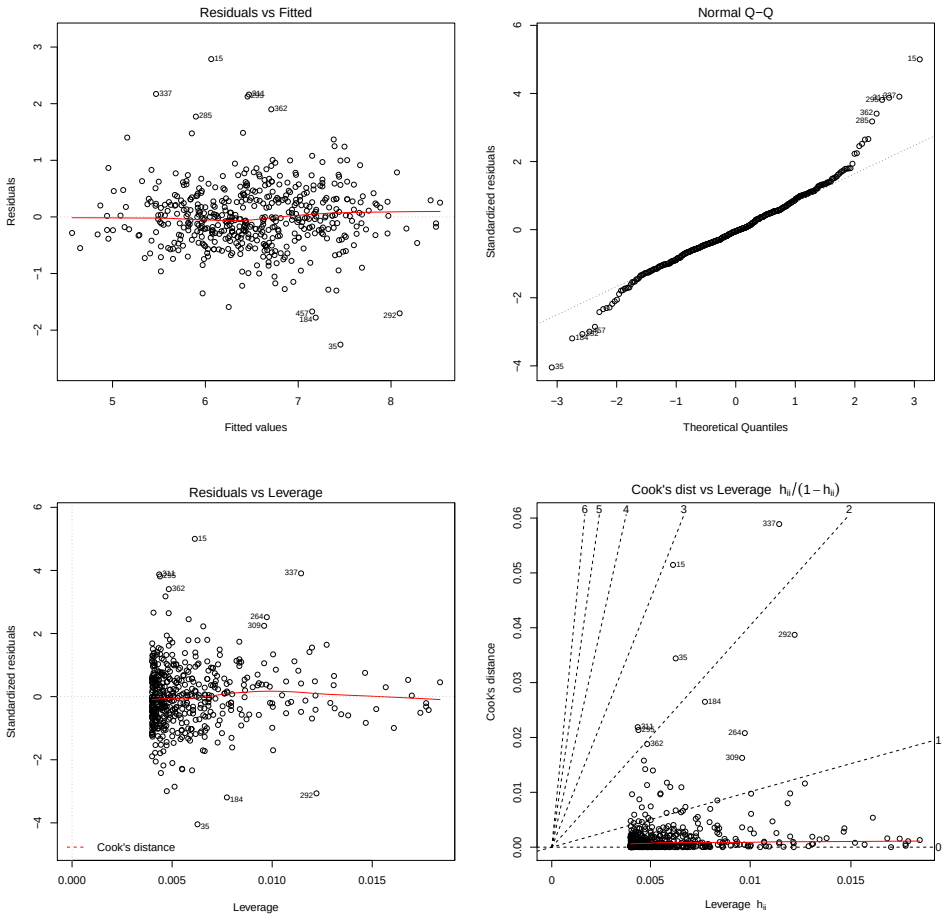
12.4 Regression diagnostics

As with univariate regression, multivariate regression models must be examined to see if they meet modelling assumptions.

Task 80 : Display the diagnostic plots for the additive model: (1) residuals vs. fits; (2) normal Q-Q plot of the residuals; (3) residuals vs. leverage; (4) Cook's distance vs. leverage. •

These are plot types 1, 2, 5, and 6, respectively, selected with the `which` optional argument to the `plot.lm` function. We also specify the number of extreme points to label with the `id.n` optional argument.

```
> par(mfrow = c(2, 2))
> plot.lm(model.straw.ns.grain, which = c(1, 2, 5, 6),
+     id.n = 10)
> par(mfrow = c(1, 1))
```



Q90 : *Which observations (plots) are marked on the plots as being potential problems?* *Jump to A90* •

We can identify these numerically. First, we examine plots that were not well-modelled. We use the criterion of model residuals more than three standard deviations from zero; we want to see if there is any pattern to these.

We identify the plots with the large residuals, using the `rstandard` "standardized residuals" function, and show just these records in the data frame, using the `which` function to identify their row (record) numbers:

```
> (selected <- which(abs(rstandard(model.straw.ns.grain)) >
+     3))

 15  35 184 285 292 295 311 337 362
 15  35 184 285 292 295 311 337 362

> rstandard(model.straw.ns.grain)[selected]

     15      35     184     285     292     295     311     337
 5.0007 -4.0459 -3.1930  3.1776 -3.0646  3.8105  3.8741  3.9068
    362
 3.4074
```

Second, build a data frame with all the information for these plots, along with the residuals, using the `cbind` function to add a column:

```
> mhw.hires <- cbind(mhw[selected,],
+                    sres = rstandard(model.straw.ns.grain)[selected])
> rm(selected)
> str(mhw.hires)

'data.frame':        9 obs. of  7 variables:
 $ r       : int  15 15 4 5 12 15 11 17 2
 $ c       : int  1 2 10 15 15 15 16 17 19
 $ grain   : num  3.46 4.42 4.59 3.7 4.86 3.73 3.74 3.05 4.26
 $ straw   : num  8.85 5.2 5.41 7.67 6.39 8.58 8.63 7.64 8.61
 $ gsr     : num  0.391 0.85 0.848 0.482 0.761 ...
 $ in.north: logi  FALSE FALSE TRUE TRUE FALSE FALSE ...
 $ sres    : num  5 -4.05 -3.19 3.18 -3.06 ...
```

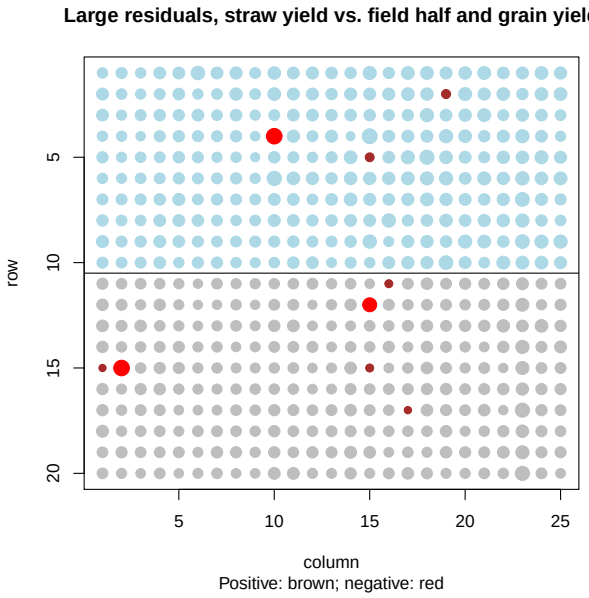
Finally, order the selected plots by the residual, using the `order` function:

```
> mhw.hires[order(mhw.hires$sres), ]

     r  c grain straw     gsr in.north    sres
35  15  2  4.42  5.20 0.85000    FALSE -4.0459
184  4 10  4.59  5.41 0.84843     TRUE -3.1930
292 12 15  4.86  6.39 0.76056    FALSE -3.0646
285  5 15  3.70  7.67 0.48240     TRUE  3.1776
362  2 19  4.26  8.61 0.49477     TRUE  3.4074
295 15 15  3.73  8.58 0.43473    FALSE  3.8105
311 11 16  3.74  8.63 0.43337    FALSE  3.8741
337 17 17  3.05  7.64 0.39921    FALSE  3.9068
15  15  1  3.46  8.85 0.39096    FALSE  5.0007
```

We can also visualize the locations of these in the field: high positive residuals green, high negative residuals red, symbol size proportional to the grain/straw ratio:

```
> plot(c, r, ylim = c(20, 1), cex = 3 * gsr/max(gsr), pch = 20,
+     col = ifelse(rstandard(model.straw.ns.grain) > 3,
+         "brown", ifelse(rstandard(model.straw.ns.grain) <
+             (-3), "red", ifelse(in.north, "lightblue",
```

```
+             "gray"))), xlab = "column", ylab = "row")
> abline(h = 10.5)
> title(main = "Large residuals, straw yield vs. field half and grain yield")
> title(sub = "Positive: brown; negative: red")
```



Note: Note the use of the nested `ifelse` functions to select the four colours.

Q91 : *Are the high positive or negative residuals concentrated in one part of the field? Is there anything else unusual about these? Hint: look at the most negative and positive residuals.* *Jump to A91* •

The plot of leverage vs. Cook's distance shows which plots most affect the fit: high leverage and high distance means that removing that plot would have a large effect on the fit.

Q92 : *Do the two highest-residual plots identified in the previous question have high leverage? Which high-residual plots also have high leverage?* *Jump to A92* •

We can examine the effect of these on the fit by re-fitting the model, leaving out one or more of the suspect plots.

Task 81 : Fit the model without the two adjacent plots where we hypothesize sloppy field procedures and compare the goodness-of-fit and regression equations to the original model. •

To exclude some observations, we use the `-` operator on a list of dataframe row numbers created with the `c` function:

```
> model.straw.ns.grain.adj <- lm(straw ~ in.north + grain,
+     data = mhw[-c(15, 35), ])
> summary(model.straw.ns.grain.adj)

Call:
lm(formula = straw ~ in.north + grain, data = mhw[-c(15, 35),
    ])

Residuals:
    Min      1Q  Median      3Q     Max
-1.7926 -0.3106 -0.0274  0.3017  2.1942

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.9528     0.2094    4.55  6.8e-06 ***
in.northTRUE  -0.5118     0.0481  -10.64  < 2e-16 ***
grain          1.4731     0.0525   28.04  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.536 on 495 degrees of freedom
Multiple R-squared:  0.64,        Adjusted R-squared:  0.638
F-statistic:  440 on 2 and 495 DF,  p-value: <2e-16

> summary(model.straw.ns.grain)

Call:
lm(formula = straw ~ in.north + grain, data = mhw)

Residuals:
    Min      1Q  Median      3Q     Max
-2.2548 -0.3189 -0.0276  0.3042  2.7871

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    1.0461     0.2178     4.8  2.1e-06 ***
in.northTRUE  -0.5132     0.0500   -10.3  < 2e-16 ***
grain          1.4499     0.0546    26.5  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.559 on 497 degrees of freedom
Multiple R-squared:  0.614,        Adjusted R-squared:  0.613
F-statistic:  396 on 2 and 497 DF,  p-value: <2e-16
```

Q93 : *Does the model without the two plots fit the remaining plots better than the original model? How different are the model coefficients?* *Jump to A93* •

Challenge: Compare the four diagnostic plots for the adjusted additive regression model (i.e., leaving out the "suspect" points) with the diagnostic plots for the additive regression model with all points, above (§12.4). Display the two sets of diagnostic plots together and evaluate them visually. What,

if anything, has improved? Does the model now meet the assumptions of linear regression?

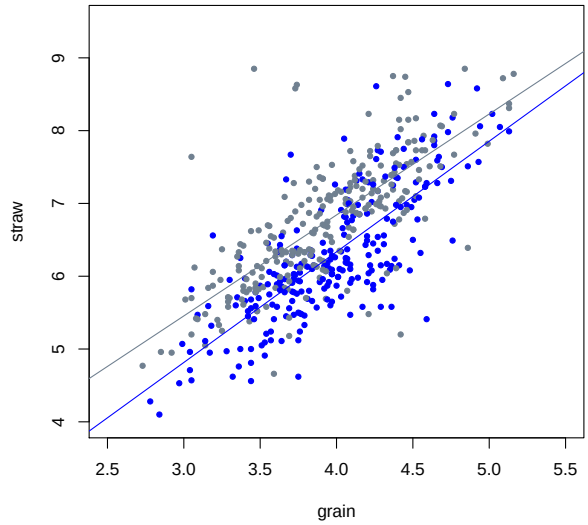
12.5 Analysis of covariance: a nested model*

In the parallel-lines model there is only one regression line between the continuous predictor and predictand, which can be moved up and down according to different class means; this is an **additive** model. In the **interaction** model there is both an overall line and deviations from it according to class, allowing different slopes, as well as differences in class means.

Another way to look at this is to abandon the idea of a single regression altogether, and fit a separate line for each class. This is a **nested** model: the continuous predictor is measured only *within* each level of the classified predictor. There is no interest in the whole-field relation between straw and grain, only the overall difference between classes (here, the field halves), and then the best fit of the straw vs. grain relation in each half separately.

A nested model is specified with the `/` formula operator (this is *not* mathematical division). This is to be read as "fit the relation after the `/` separately for the two values of the classified variable".

```
> model.straw.ns.grain.nest <- lm(straw ~ in.north/grain,
+     data = mhw)
> summary(model.straw.ns.grain.nest)

Call:
lm(formula = straw ~ in.north/grain, data = mhw)

Residuals:
    Min      1Q  Median      3Q     Max
-2.2242 -0.3169 -0.0398  0.3136  2.7574

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           1.2930     0.2979    4.34  1.7e-05 ***
in.northTRUE         -1.0375     0.4350   -2.39    0.017 *
in.northFALSE:grain   1.3872     0.0752   18.44  < 2e-16 ***
in.northTRUE:grain    1.5199     0.0794   19.14  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.559 on 496 degrees of freedom
Multiple R-squared:  0.615,        Adjusted R-squared:  0.613
F-statistic:  265 on 3 and 496 DF,  p-value: <2e-16

> plot(straw ~ grain, data = mhw, col = ifelse(mhw$in.north,
+     "blue", "slategray"), pch = 20, xlim = c(2.5, 5.5),
+     ylim = c(4, 9.5))
> coef <- coef(model.straw.ns.grain.nest)
> abline(coef[1], coef[3], col = "slategray")
> abline(coef[1] + coef[2], coef[4], col = "blue")
```



Note that the nested model does *not* compute an overall slope of straw vs. grain; instead each half has its own regression line (intercept and slope). The coefficient `in.northTRUE` gives the *difference* between the intercept of the N-half regression line from the intercept of the S-half regression line. The two coefficients `in.northFALSE:grain` and `in.northTRUE:grain` give the computed slopes of the two regression lines of straw yield vs. grain yield.

Q94 : *What are the two slopes of straw vs. grain? Are they different? Do they differ from the single slope found in the parallel regression model?* *Jump to A94* •

Q95 : *This model has the same adjusted R-squared as the interaction model. Why?* *Jump to A95* •

Challenge: Compare the regression lines of this nested model with the regression lines implied by the interaction model. Are they the same? Why or why not?

12.6 Answers

A80 : *Yes, straw yield most likely ($p \approx 0$) is higher in the South half; but this explains very little (0.066) of the total variance.* *Return to Q80* •

A81 : *Yes, straw yield almost surely ($p \approx 0$) varies with grain yield; this explains about half (0.532) of the total variance.* *Return to Q81* •

A82 : *Both coefficients are highly significant (probability that they are really*

zero almost nil). Coefficient `in.northTRUE` *represents the difference between field halves, and* `grain` *the regression between straw and grain.*

This explains over 60% (0.613) of the total variance. *Return to Q82* •

A83 : *Coefficient* `in.northTRUE` *represents the difference between field halves; the fitted value is -0.5132 lb. plot^{-1}. Coefficient* `grain` *is the regression between straw and grain; the fitted value is 1.4499 lb. straw increase per plot for each lb grain increase per plot. This is* not *the same as the best-fit univariate line (ignoring field half), which is 1.4305* *Return to Q83* •

A84 : *The multivariate additive model is clearly best; it explains about two-thirds (66%) of the variability, whereas the whole-field straw vs. grain model only explains just more than half (53%) and the field-half model very little (6%).* *Return to Q84* •

A85 : *Yes, in both cases the probability that we'd be wrong by rejecting the null hypothesis of no difference is practically zero. The RSS decreases from 375.4 for the field-half model, and 188.2 for the whole-field straw vs. grain model, to 155.3 for the combined model. That is, much less variability in straw yield remains unexplained after the combined model.* *Return to Q85* •

A86 : *The parallel regression models the case where one half of the field is more productive, on average, than the other, but the relation between grain and straw is the same in both cases (i.e. the grain/straw ratio is the same in both field halves). There is no difference in plant morphology, just overall size.*

By contrast, the interaction model allows that the two field halves may also differ in the grain/straw ratio, i.e. the relation between grain and straw yield – different plant morphology. *Return to Q86* •

A87 : *The coefficients are:*

1. `(Intercept)`*: estimated straw yield at* ***zero*** *grain yield and in the S field half;*
2. `in.northTRUE`*:* ***difference*** *in* ***average*** *yield in the N vs. the S;*
3. `grain`*:* ***increase*** *in straw yield for each unit increase in grain yield, in the S field half;*
4. `in.northTRUE:grain`*:* ***difference*** *in* ***slope*** *(increase in straw yield for each unit increase in grain yield) in the N vs. the S.*

Return to Q87 •

A88 : *The coefficient* `in.northTRUE:grain` *is not significant; the probability of falsely rejecting the null hypothesis is quite high, 0.2255, so we should accept the hypothesis that this difference is really 0.*

According to this test, the interaction is not significant. Plants grow with the same morphology in the two field halves. *Return to Q88* •

A89 : *The interaction model has slightly higher adjusted R^2: 0.61305 vs. 0.61268. However, the ANOVA table shows that this increase has a p=0.225 probability of occurring by chance, so we conclude the interaction model is not justified. This is the same conclusion we reached from the model summary, where the interaction term (coefficient) was not significant.* *Return to Q89* •

A90 : *Plots labelled on the diagnostic graphs are 15, 337, 311, 295, 362, 285 (positive residuals) and 35, 184, 292, 457 (negative residuals). These have residuals more extreme than expected by theory (normal Q-Q plot of residuals). Plots 292, 337, 264, 184 and 309 are large residuals with high leverage.* *Return to Q90* •

A91 : *The highest four positive residuals are all in the S half, but otherwise do not seem clustered. The most negative residual is from plot 35 (r=15, c=2) and the most positive from the immediately adjacent plot 15 (r=15, c=1). Could some of the grain from plot 15 have been accidentally measured as part of the yield of plot 35? If these two are combined, the grain/straw ratio is 0.56085, close to the mean grain/straw ratio of the whole field, 0.61054.* *Return to Q91* •

A92 : *Plots 15 and 35 do not have high leverage, i.e. their removal would not change the equation very much. The high-leverage plots that also have high residuals are 292 and 184 (negative residuals) and 337, 264 and 309 (positive residuals).* *Return to Q92* •

A93 : *The fit is considerably better without these badly-modelled plots: 0.63841 without the two plots vs. 0.61268 with them. Another 2.5% of the variation is explained.*

The coefficient for field half hardly changes, but the regression line changes substantially: higher intercept: (0.9528 vs. 1.0461) and shallower slope (1.4731 vs. 1.4499 . As predicted by the high leverage, removing these points changes the functional relation. *Return to Q93* •

A94 : *The slope in the S half is 1.3872, in the N half 1.5199. These differ considerably from each other, and from the parallel regression slope: 1.4499. The slope in the S half is less steep, in the N half steeper, than the parallel regression slope.* *Return to Q94* •

A95 : *Both models have four parameters to fit the same dataset. Both model difference between levels (either means or intercepts) and slopes.* *Return to Q95* •

We are done with these models and some other variables, so clean up the workspace:

```
> rm(model.straw.ns, model.straw.grain, model.straw.ns.grain,
+     model.straw.ns.grain.adj, model.straw.ns.grain.i,
```

```
+     model.straw.ns.grain.nest)
> rm(struct.beta, beta, mhw.hires)
```

13 Principal Components Analysis

In §8.1.1 and §11.3 we computed the correlation between the grain and straw yields from each plot, i.e., the strength of their association. This showed that they are highly-correlated. Thus there is **redundancy**: the two variables are not fully **independent**.

Principal components analysis (PCA) is a data reduction technique. It finds a new set of variables, equal in number to the original set, where these so-called **synthetic** variables are uncorrelated. In addition, the first synthetic variable represents as much of the common variation of the original variables as possible, the second variable represents as much of the residual variation as possible, and so forth. This technique thus reveals the structure of the data. The transformation itself and the synthetic variables produced by it can be **interpreted** by the analyst to understand the underlying processes.

In the present example, Mercer & Hall measured two variables: grain and straw yield. However, these measured quantities are the outcomes of **processes** which we can not directly observe: (1) plant growth; (2) partition of plant growth between grain and straw. PCA can be used to gain insight into these.

PCA is often used for data reduction in datasets with many variables; good examples are image processing or spectroscopy with many bands, and geochemical datasets with many measured elements. It is explained by many textbooks on remote sensing [e.g. 1, 29]

> **Note:** In terms of mathematics, the vector space made up of the original variables is projected onto another space such that the projected variables are orthogonal, with descending variances. The mathematics are well-explained in many texts, e.g., [8, 26].

PCA can be **standardized** or not. If so, the variables are centred to zero mean and then scaled to unit variance, thus giving them equal importance mathematically. In the second, the original units are used after centring; thus variables with more variance are more important. In the present case, although the two variables are computed in the same units of measure (lb. plot^{-1}), they are of intrinsically different magnitude (there is much more straw than grain). To reveal the relation between the variables they should be standardised before computing the principal components.

Task 82 : Compute the standardized PCA of grain and straw yields. •

The `prcomp` function computes the PCA; with the optional `scale` argument the variables are first scaled, resulting in standardized PCA. The output is an object of class `prcomp`.

```
> pc <- prcomp(mhw[, c("grain", "straw")], scale = T)
> summary(pc)
```

```
Importance of components:
                          PC1    PC2
Standard deviation     1.3152 0.5198
Proportion of Variance 0.8649 0.1351
Cumulative Proportion  0.8649 1.0000
```

The standard deviations shown here are of the synthetic variables (principal components); these re-apportion the standard deviations of the original variables according to the redundancy.

Q96 : *How much of the variance is explained by each component? How much data redundancy is in the original two variables?* *Jump to A96* •

Sometimes PCA is used just for data reduction, but here our interest is in the interpretation. One way to interpret is to examine the **loadings**: these show how the synthetic variables are created from the original ones.

> **Note:** Mathematically, these are the eigenvectors (in the columns) which multiply the original (scaled and centred) variables to produce the synthetic variables (principal components).

Task 83 : Examine the loadings of the two PC's. •

These are in the `rotation` field of the PCA object.

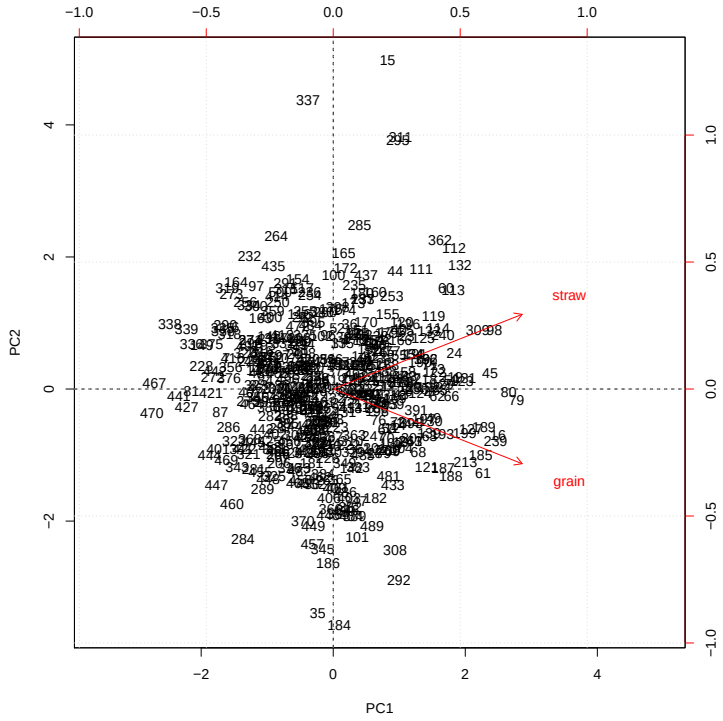
```
> pc$rotation

             PC1        PC2
grain 0.7071068 -0.7071068
straw 0.7071068  0.7071068
```

These can be visualized on a so-called **biplot**, which shows the loadings as vectors in the space spanned by two PC's. It also shows the location of the 500 observations in this space, i.e., the values of the synthetic variables (called the principal component "scores").

Task 84 : Display the biplot. •

This is displayed with the `biplot` method, which when presented with an object of class `prcomp` specializes to the `biplot.prcomp` function. We specify the `pc.biplot` argument to be `TRUE`, to produce the biplot as proposed by Gabriel [17], which scales the scores of the observations and rotations by the components' standard deviations (eigenvalues).

```
> biplot(pc, pc.biplot = T)
> abline(h = 0, lty = 2)
> abline(v = 0, lty = 2)
> grid()
```



The biplot can be interpreted in several ways:

1. The **orientation** (direction) of the vector, with respect to the PC space. The more a vector, which represents an original variable, is parallel to a PC axis, the more it contributes to that PC.

2. The **length** in the space defined by the displayed PCs; the longer the vector, the more variability of this variable is represented by the two displayed PCs.

3. The **angles between vectors** of different variables show their correlation in the space spanned by the two displayed PC's: small angles represent high positive correlation, right angles represent lack of correlation, opposite angles represent high negative correlation.

   The scaled values of the eigenvectors are shown as the top (PC1) and right (PC2) axes.

   Because of the scaling applied, the inner (vector) product between two variables approximates their covariance (if standardized, their correlation). Thus two vectors that are almost parallel will have a very high covariance, and two orthogonal vectors will have no covariance.

4. The **location of observations** in the biplot space, defined by their scores scaled by the standard deviation of each PC, i.e., its eigenvalue: this shows the relation of observations to each other and which observations are unusual. These values are shown on the bottom (PC1) and left (PC2) axes.

   Because of the scaling applied, the distances between observations plot-

ted in this space approximate the Mahalanobis distance between them; this is a common multivariate measure of similarity between observations.

Q97 : *Considering the biplot and loadings matrix, what does the first PC represent?* *Jump to A97* •

To help answer this question, we can examine the field plots with the highest and lowest scores of PC1. The scores (synthetic variables) are returned in the `x` field of the `prcomp` object; this is a matrix whose columns are the PC scores.

The `which.max` and `which.min` functions find the positions of the maximum and minimum values in a vector. We then extract the record from the dataframe, and also show the PC scores:

```
> summary(pc$x)

      PC1                PC2
 Min.   :-3.61141   Min.   :-1.8592
 1st Qu.:-0.93213   1st Qu.:-0.3474
 Median :-0.09962   Median : 0.0119
 Mean   : 0.00000   Mean   : 0.0000
 3rd Qu.: 0.92202   3rd Qu.: 0.3028
 Max.   : 3.65214   Max.   : 2.5921

> mhw[ix.max <- which.max(pc$x[, "PC1"]), ]

    r c grain straw       gsr in.north
79 19 4  5.16  8.78 0.5876993    FALSE

> pc$x[ix.max, ]

        PC1         PC2
 3.65214277 -0.08601565

> mhw[ix.min <- which.min(pc$x[, "PC1"]), ]

     r  c grain straw       gsr in.north
470 10 24  2.84   4.1 0.6926829     TRUE

> pc$x[ix.min, ]

       PC1        PC2
-3.6114084 -0.1902355
```

Q98 : *Which are the plots with the largest positive and negative scores for PC1? How is this explained?* *Jump to A98* •

If you have looked closely at the biplot, you have surely noticed that the lower and left axes, which are related to the PC scores, do not correspond exactly to the scores (read the values for PC1 off the graph for the highest/lowest scores discovered just above). This is because the default biplot is scaled

by the standard deviation of each PC, i.e., its eigenvalue. If we reverse the scaling, we see the value on the plot:

```
> pc$sdev

[1] 1.3152116 0.5198253

> pc$x[ix.max, ]/pc$sdev

       PC1        PC2
 2.7768480 -0.1654703

> pc$x[ix.min, ]/pc$sdev

       PC1        PC2
-2.7458762 -0.3659605
```

Now you can properly locate the PC scores for field plots 79 and 470 on the biplot.

Q99 : *What is the interpretation of the second PC?* *Jump to A99* •

To help answer this question we can again look at high– and low–scoring observations, but this time for PC2.

```
> mhw[ix.max <- which.max(pc$x[, "PC2"]), ]

    r c grain straw       gsr in.north
15 15 1  3.46  8.85 0.3909605    FALSE

> pc$x[ix.max, ]/pc$sdev

      PC1       PC2
0.8243637 4.9865139

> mhw[ix.min <- which.min(pc$x[, "PC2"]), ]

    r  c grain straw       gsr in.north
184 4 10  4.59  5.41 0.8484288     TRUE

> pc$x[ix.min, ]/pc$sdev

        PC1         PC2
 0.09119713 -3.57666566
```

Q100 : *Interpret the two PCs and the proportion of variance explained by each in terms of this experiment.* *Jump to A100* •

So far we have only looked at the observations in PC space, shown by the bottom (PC1) and left (PC2) axes. There is another set of axes on the graph, and vector arrows (drawn in red). The top (PC1) and right (PC2) axes show the eigenvector components ("rotations", "loadings") for the original variables, multiplied by the standard deviation (eigenvalue) of the corresponding component. The vector arrows, beginning at the origin and ending

in labels, show the eigenvectors for the two original variables, scaled as just explained. The value is at the centre of the label.

```
> pc$rotation[, 1] * pc$sdev[1]

    grain     straw
0.9299951 0.9299951

> pc$rotation[, 2] * pc$sdev[2]

    grain     straw
-0.367572  0.367572
```

The cosine of the angle between the vectors is proportional to their correlation.

Q101 : *What does the biplot reveal about the correlation between grain and straw yield?* *Jump to A101* •

Challenge: Repeat the analysis with *unstandardized* principal components. Explain the differences with standardized PCA for the proportion of variance explained per component, the rotations, the biplot, and the interpretation of the two PC's.

13.1 Answers

A96 : *PC1 explains 86.5% and PC2 13.5% of the total variance. That is, about 85% of the information is in the first component; this is a measure of the redundancy.* *Return to Q96* •

A97 : *The first PC is made up of large contributions from the standardized variables representing grain yield and straw yield. This is interpreted as general plant size: high yielding-plants score higher on PC1.* *Return to Q97* •

A98 : *The maximum score is from plot 79, the minimum from plot 470. The first has both very high grain and straw yield, the second both very low. The interpretation that PC1 represents overall yield is confirmed.* *Return to Q98* •

A99 : *The second PC represents a contrast between grain and straw yield,* ***independent*** *of the overall yield level. Positive values have higher-than-average straw yields, at a given level of grain yield. In other words, the grain-to-straw ratio (GSR) is low. The highest– and lowest– scoring plots for PC2 confirm this: plot 15 has a very low GSR: 0.391, whereas plot 184 has a very high GSR: 0.848.* *Return to Q99* •

A100 : *In summary, the first PC, accounting for about 85% of the variance in both grain and straw yield, represents the overall yield level, whereas the second PC, accounting for about 15% of the variance, represents the grain/straw ratio, i.e.,*

plant morphology independent of yield. The great majority of the overall variability in this field is due to variable yield. However, there is still a fair amount of variation in plant morphology that does not depend on overall plant size. Since this is one variety of wheat (i.e., no genetic variation) and one management, we conclude that local environmental factors affect not only yield but also plant morphology. *Return to Q100* •

A101 : *The angle between the vectors representing the two yields is small, showing that the yields are highly-correlated.* *Return to Q101* •

14 Model evaluation

In §8, §9 and §12 models were evaluated by the goodness-of-fit (coefficient of determination, expressed as the adjusted R^2) and compared by ANOVA. These are *internal* measures of model quality: the data used to evaluate the model is also used to build it.

Of course, the main use of a model is to predict; and what we really would like is some measure of the **predictive success** of a model, i.e., an *external* measures of model quality, using **independent** data to evaluate the model, not the data that was used to build it. The modelling steps so far were designed to best **calibrate** the model; now we want to **evaluate** it.

> **Note:** What we call **evaluation** is often termed **validation**; we prefer the first term because we can never know if the model is valid, we can only evaluate how well it matches reality.

A common approach is to **split** the dataset into a calibration and an evaluation (often called a validation) set.

1. The model is developed using only the observations in the **calibration** set;
2. This model is used to **predict** at the the observations in the **evaluation** set, using the actual (measured) values of the **predictor** (independent) variable(s);
3. These predicted values are compared to the actual (measured) values of the **response** (dependent) variable in the **evaluation** set.

14.1 Splitting the dataset

Splitting the dataset has two constraints:

1. The calibration set must be large enough reliable modelling;
2. The evaluation set must be large enough for reliable evaluation statistics.

A common split in a medium-size dataset such as this one is 3 to 1, i.e., 3/4 for calibration and 1/4 for evaluation.

The next issue is how to select observations for each set. This can be:

- **random**: select at random (without replacement); this requires no assumptions about the sequence of items in the dataset;
- **systematic**: select in sequence; this requires absence of **serial correlation**, i.e., that observations listed in sequence be **independent**;
- **stratified**: first divide the observations by some factor and then apply either a random or systematic sampling within each stratum, generally proportional to stratum size.

To decide which strategy to use, we need to know how the dataset is ordered, and if there are any useful stratifying factors.

The sequence of observations is given by the `row.names` function; to see just the first few we use the `head` function with the optional `n` argument to specify the number of items to view:

```
> head(row.names(mhw), n = 10)

 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
```

The observations correspond to these are:

```
> head(mhw, n = 10)

    r c grain straw       gsr in.north
1   1 1  3.63  6.37 0.5698587     TRUE
2   2 1  4.07  6.24 0.6522436     TRUE
3   3 1  4.51  7.05 0.6397163     TRUE
4   4 1  3.90  6.91 0.5643994     TRUE
5   5 1  3.63  5.93 0.6121417     TRUE
6   6 1  3.16  5.59 0.5652952     TRUE
7   7 1  3.18  5.32 0.5977444     TRUE
8   8 1  3.42  5.52 0.6195652     TRUE
9   9 1  3.97  6.03 0.6583748     TRUE
10 10 1  3.40  5.66 0.6007067     TRUE
```

Q102 : *What is the sequence of observations? Could there be serial correlation?* *Jump to A102*
•

Q103 : *What are possible stratifying factors?* *Jump to A103* •

Q104 : *Which of these strategies is best to apply to the Mercer-Hall dataset, and why?* *Jump to A104* •

Task 85 : Select a random sample of 3/4 of the 500 observations as the calibration set, and the rest as the evaluation set. •

The `sample` function selects a random sample from a vector (here, the row numbers of the dataframe); the `size` argument gives the size of the sample and the `replace` argument is logical: should sampling be with replacement?

Although we know there are 500 observations, it's more elegant to extract the number programatically with the `dim` "dimensions of a matrix" function.

Finally, we use `set.seed` so your results will be the same as ours; in practice you would not use this, instead accept the random seed.

> **Note:** The argument 123 to `set.seed` is arbitrary, it has no meaning. Any number can be used, the only purpose is to get the same "random" result in the subsquent call to `sample`.

```
> dim(mhw)

[1] 500   6

> (n <- dim(mhw)[1])

[1] 500

> set.seed(123)
> head(index.calib <- sort(sample(1:n, size = floor(n * 3/4), replace = F)),
+     n = 12)

 [1]  1  2  3  4  5  6  7 11 12 13 14 15

> length(index.calib)

[1] 375
```

Task 86 : Assign the remaining observations to the evaluation set. •

The very useful `setdiff` function selects the subset of a set that is *not* in another subset:

```
> head(index.valid <- setdiff(1:n, index.calib), n = 12)

 [1]  8  9 10 18 20 25 29 30 32 33 37 40

> length(index.valid)

[1] 125
```

Check that the two subsets together equal the original set; this gives a chance to introduce the `union` function and the `setequal` "are the sets equal?" logical function.

```
> setequal(union(index.calib, index.valid), 1:n)

[1] TRUE
```

14.2 Developing the model

From §12.2 we know that the multivariate additive model (straw yield explained by the additive effect of grain yield and field half) is clearly best in this particular experiment. However, in any other situation the field would be different and so would any effect of field half. We'd like a measure of how good is a model to predict straw yield from grain yield in general. So we work with the single-predictor model.

Task 87 : Re-fit this model on the calibration set. •

```
> cal.straw.grain <- lm(straw ~ grain, data = mhw, subset = index.calib)
> summary(cal.straw.grain)

Call:
lm(formula = straw ~ grain, data = mhw, subset = index.calib)

Residuals:
     Min       1Q   Median       3Q      Max
-1.96424 -0.34779  0.01022  0.40281  3.02782

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.06985    0.26884   3.979  8.3e-05 ***
grain        1.37351    0.06737  20.387  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.609 on 373 degrees of freedom
Multiple R-squared:  0.527,        Adjusted R-squared:  0.5258
F-statistic: 415.6 on 1 and 373 DF,  p-value: < 2.2e-16
```

Q105 : *How well do the model coefficients agree with the model based on all observations?* *Jump to A105* •

To answer this, fit the model based on all observations, then compare the absolute and relative differences of the coefficients, extracted from the model with the `coef` (or, `coefficients` function:

```
> model.straw.grain <- lm(straw ~ grain, data=mhw)
> (coef(cal.straw.grain) - coef(model.straw.grain))

(Intercept)       grain
 0.20357413 -0.05699242

> ((coef(cal.straw.grain) - coef(model.straw.grain))
+                     /coef(model.straw.grain))*100

(Intercept)       grain
  23.499816   -3.984098
```

14.3 Predicting at the evaluation observations

Now we have a model, it can be used to predict at the observations held out from calibration, i.e., the evaluation set.

Task 88 : Predict the straw yield for the observation set. •

The `predict.lm` function, which can also be called as `predict`, uses an object returned by `lm` to predict from a dataframe specified by the `newdata` argument. Here the data should be the rows (cases) of the `mhw` dataframe that are part of the evaluation set; these row numbers are in the `index.valid` vector.

```
> pred <- predict.lm(cal.straw.grain, newdata = mhw[index.valid,
+     ])
```

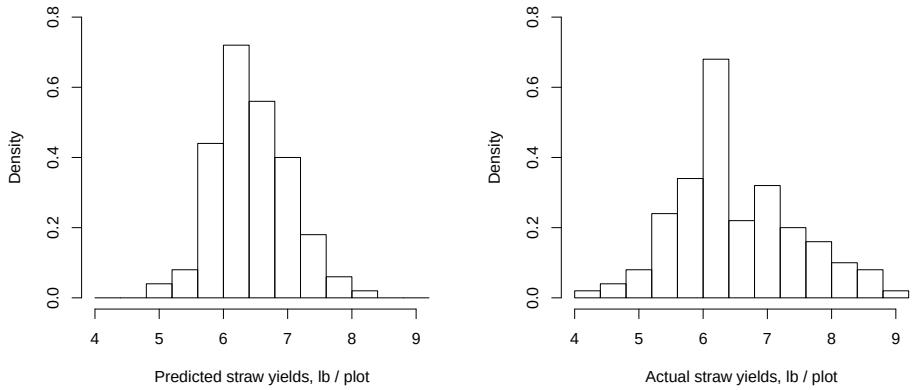
Task 89 : Compare them with the actual yields in this set, both numerically and graphically. •

For convenience, we first extract the vector of actual yields from the evaluation data frame:

```
> actual <- mhw[index.valid, "straw"]
```

Now compare the numeric summaries and side-by-side histograms; note the common scales.

```
> summary(pred); summary(actual)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.820   6.028   6.330   6.434   6.839   8.116

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   4.10    5.95    6.33    6.52    7.08    8.85

> par(mfrow=c(1,2))
> hist(pred, main="", xlab="Predicted straw yields, lb / plot",
+      breaks=seq(4,9.2,by=0.4), freq=F, ylim=c(0,.8))
> hist(actual, main="", xlab="Actual straw yields, lb / plot",
+      breaks=seq(4,9.2,by=0.4), freq=F, ylim=c(0,.8))
> par(mfrow=c(1,1))
```



Q106 : *What are the major differences in the distributions of the modelled and actual evaluation observations?* *Jump to A106* •

14.4 Measures of model quality*

A systematic approach to model quality was presented by Gauch Jr. et al. [18]. This is based on a comparison of the model-based predictions and the measured values. This should be a 1:1 relation: each model prediction should equal the true value. Of course this will rarely be the case; the decomposition proposed by Gauch Jr. et al. [18] shows the source of the disagreement and allows interpretation.

> **Note:** Another commonly-used approach was proposed by Kobayashi and Salam [23], who also wrote a letter contrasting their approach to Gauch's [22]; a comprehensive review of model evaluation, citing many applications and interpretations, is presented by Bellocchi et al. [3].

Gauch Jr. et al. [18] distinguish:

MSD : Mean Squared Deviation. This shows how close, on average the prediction is to reality. Its square root is called the Root Mean Squared Error of Prediction (RMSEP), expressed in the same units as the modelled variable.

SB : Squared bias. This shows if the predictions are systematically higher or lower than reality.

NU : Non-unity slope. This shows if the relation between predicted and actual is proportional 1:1 throughout the range of values; if not, there is either an under-prediction at low values and corresponding over-prediction at high variables (slope > 1), or vice-versa (slope < 1).

LC : Lack of correlation. This shows how scattered are the predictions about the 1:1 line.

It is quite common to report the RMSEP; this indeed is the single number that tells how closely, on average, the model predicted the evaluation points; however the decomposition shows the reason(s) for lack of agreement.

Some notation:

- there are n total evaluation observations;
- y_i is the true (measured) value of evaluation observation i;
- $\hat{y}_i$ is the predicted value of evaluation observation i;
- the overbar $\overline{y}$ indicates the arithmetic mean of the y_i, etc.

Then MSD, SB, NU, and LC are computed as:

$$\text{MSD} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{14.1}$$

$$\text{SB} = \left(\overline{\hat{y}} - \overline{y}\right)^2 \tag{14.2}$$

$$\text{NU} = (1 - b^2)\frac{1}{n}\sum_{i=i}^{n}\left(\hat{y}_i - \overline{\hat{y}}\right)^2 \tag{14.3}$$

$$\text{LC} = (1 - r^2)\frac{1}{n}\sum_{i=i}^{n}\left(y_i - \overline{y}\right)^2 \tag{14.4}$$

where:

- b is the slope of the least-squares regression of actual values on the predicted values, i.e., $\sum y_i\hat{y}_i / \sum \hat{y}_i^2$; this is also called the **gain**.
- r^2 is the square of the correlation coefficient $r_{1:1}$ between actual and predicted, i.e., $(\sum y_i\hat{y}_i)^2 / (\sum y_i)^2(\sum \hat{y}_i)^2$.

Thus NU is the non-unity slope in the regression of actual on predicted, scaled by the sums of squares of the predicted values of the evaluation observations, and LC is the lack of correlation in the regression of actual on predicted, scaled by the sums of squares of the actual values of the evaluation observations.

These have the relation:

$$\text{MSD} = \text{SB} + \text{NU} + \text{LC} \tag{14.5}$$

That is, the total evaluation error consists of bias, gain, and lack of correlation. These are distinct aspects of model quality and can be interpreted as **translation** (SB), **rotation** (NU), and **scatter** (LC), respectively:

Translation : The model systematically over- or under-predicts;

Rotation : The relation between actual and predicted value is non-linear, that is, not a constant relation throughout the range of values;

Scatter : The model is not precise.

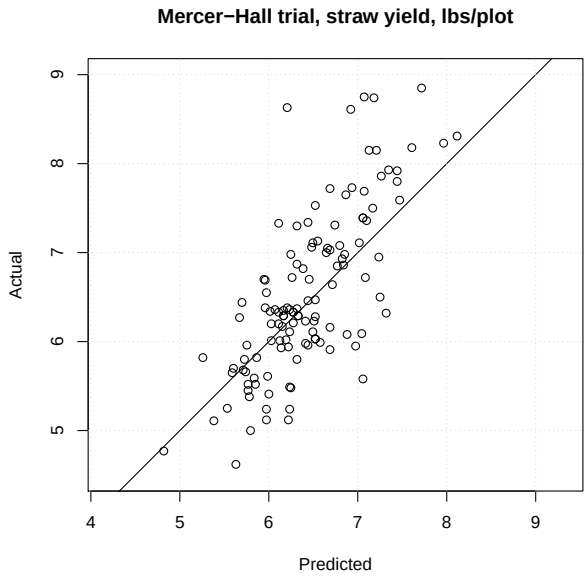
If there is significant translation or rotation, this indicates the model **form** is not correct; we show an example of this below in §14.5. If there is significant scatter, this indicates that the model does not well-describe the system; perhaps there are missing factors (predictors) or perhaps the system has a lot of noise that can not be modelled. Thus by examining the model evaluation decomposition the analyst can decide on how to improve the model.

We begin by visualizing the predictive success of the model on a 1:1 plot. Note that by convention this plot has the actual (measured) value on the y-axis (dependent) and the predicted (modelled) value on the x-axis (independent). That is, it shows how the prediction predicts the actual.

Task 90 : Plot the actual vs. predicted grain yields of the evaluation set, with a 1:1 line. •

The plot:

```
> plot(actual ~ pred, ylab="Actual", xlab="Predicted", asp=1,
+      main="Mercer-Hall trial, straw yield, lbs/plot",
+      xlim=c(4.5,9), ylim=c(4.5,9));
> abline(0,1); grid()
```



Q107 : *How well does the calibrated model predict the evaluation observations?* *Jump to A107*

•

14.4.1 MSD

Task 91 : Compute the MSD and RMSEP. •

```
> (valid.msd <- sum((actual - pred)^2)/length(index.valid))

[1] 0.4034063

> (valid.msd <- mean((actual - pred)^2))

[1] 0.4034063

> (valid.rmsep <- sqrt(valid.msd))

[1] 0.6351428
```

Q108 : *How does the RMSEP compare to the RMSE (root mean squared error) in the calibration linear model? What is the practical interpretation of the RMSE?* *Jump to A108* •

The RMSE of the linear model is the mean of the squared residuals, extracted from the linear model with the `residuals` function:

```
> (rmse <- sqrt(mean(residuals(cal.straw.grain)^2)))

[1] 0.6074227
```

14.4.2 SB

Task 92 : Compute the bias and its square (SB). •

```
> (valid.bias <- (mean(pred) - mean(actual)))

[1] -0.08587374

> (valid.sb <- valid.bias^2)

[1] 0.0073743

> valid.sb/valid.msd * 100

[1] 1.828008
```

Q109 : *What is the bias? Is it positive or negative? What does this imply about the model? What proportion of the MSD is attributed to the SB?* *Jump to A109* •

14.4.3 NU

The next component of the error the non-unity slope (NU) factor. To compute this we first need to compute b, the slope of the least-squares regression of actual values on the predicted values. This is an interesting number in its own right, the **gain**, which we hope will be 1. The gain can be directly with the least-squares formula $\sum y_i \hat{y}_i / \sum \hat{y}_i^2$, but in practice it's easier to

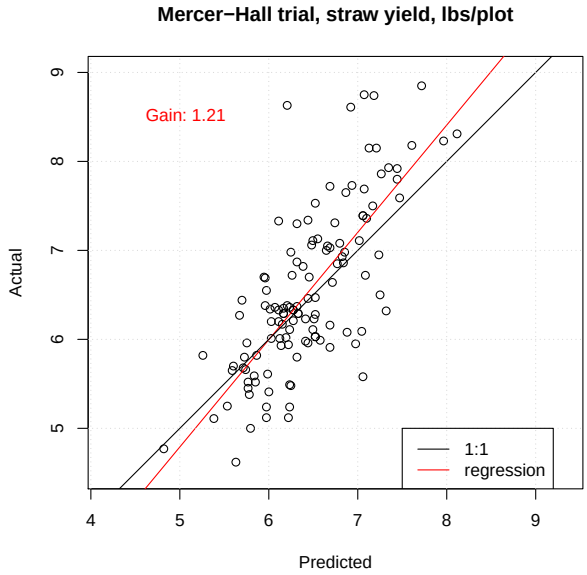
use the `lm` function to fit the slope and then extract that coefficient with the `coef` function:

Task 93 : Compute the regression of actual straw yield on predicted grain yield in the evaluation set, and display, along with the 1:1 line, on a scatterplot of actual vs. predicted. •

As usual, we fit the model with the `lm` function, summarize it with the `summary` function (to see the coefficients and scatter), and produce the scatterplot with `plot`. We add text to the plot with `text`, and (this is new) produce a legend with `legend`, also specifying the legend symbols (here, lines using the `lty` "line type" argument) and colours (here, matching the line colours, both specified with the `col` "colour" argument):

Now compute, summarize and plot the regression:

```
> regr.actual.pred <- lm(actual ~ pred)
> summary(regr.actual.pred)

Call:
lm(formula = actual ~ pred)

Residuals:
     Min       1Q   Median       3Q      Max
-1.69278 -0.34682 -0.00838  0.32601  2.38413

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.23880    0.61383  -2.018   0.0458 *
pred         1.20589    0.09501  12.692   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6226 on 123 degrees of freedom
Multiple R-squared:  0.567,        Adjusted R-squared:  0.5635
F-statistic: 161.1 on 1 and 123 DF,  p-value: < 2.2e-16

> plot(actual ~ pred, ylab="Actual", xlab="Predicted", asp=1,
+      main="Mercer-Hall trial, straw yield, lbs/plot",
+      xlim=c(4.5,9), ylim=c(4.5,9));
> abline(regr.actual.pred, col="red")
> abline(0,1); grid()
> text(4.5, 8.5, paste("Gain:", round(coef(regr.actual.pred)[2], 2)),
+      pos=4, col="red")
> legend(7.5, 5, c("1:1","regression"), lty=1, col=c("black","red"))
```

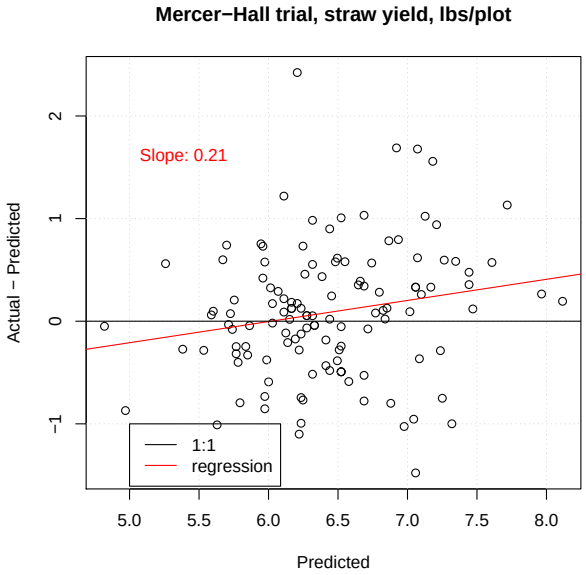


Q110 : *What is the gain? Is it greater than or less than* 1*? What does this say about the model?* *Jump to A110* •

The model summary shows that the gain is significantly different from zero, which is no surprise. To assess it against a null hypothesis of $\beta = 1$, one method is to remove the 1:1 line from the regression and then compare the regression slope to zero. The 1:1 line is simply the line where actual yield equals predicted yield, so first we subtract the predicted from the actual; this would reduce an exact 1:1 slope to 0. Then we re-fit and summarize the model. We also visualize the hypothesized 0 slope and the actual negative gain.

```
> regr.actual.pred.0 <- lm(I(actual - pred) ~ pred)
> summary(regr.actual.pred.0)

Call:
lm(formula = I(actual - pred) ~ pred)

Residuals:
     Min       1Q   Median       3Q      Max
-1.69278 -0.34682 -0.00838  0.32601  2.38413

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.23880    0.61383  -2.018   0.0458 *
pred         0.20589    0.09501   2.167   0.0322 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6226 on 123 degrees of freedom
Multiple R-squared:  0.03677,        Adjusted R-squared:  0.02894
F-statistic: 4.696 on 1 and 123 DF,  p-value: 0.03216
```

```
> plot(I(actual - pred) ~ pred, ylab="Actual - Predicted",
+      xlab="Predicted",
+      main="Mercer-Hall trial, straw yield, lbs/plot")
> grid(); abline(regr.actual.pred.0, col="red"); abline(h=0)
> text(5, 1.6, paste("Slope:",
+                    round(coef(regr.actual.pred.0)[2], 2)),
+      pos=4, col="red")
> legend(5, -1, c("1:1","regression"), lty=1, col=c("black","red"))
```



Note here the use of the `I` "as-is" function to specify that the expression `(actual-pred)` is an arithmetic operation, *not* a model formula. This is necessary because `-` is a formula operator, meaning to remove a term from a model.

The slope coefficient here is exactly 1 less than the slope coefficient from the original regression of actual vs. predicted, as expected:

```
> coef(regr.actual.pred)[2] - coef(regr.actual.pred.0)[2]

pred
   1
```

Q111 : *Is the gain of the regression of actual vs. predicted significantly different from 1?* *Jump to A111* •

Task 94 : Compute the non-unity slope (NU) factor. •

First, extract the slope from the fitted linear model:

```
> b <- coef(regr.actual.pred)[2]
> names(b) <- NULL
> print(b)
```

```
[1] 1.20589
```

> **Note:** The `names` function retrieves the name attribute; here it was assigned the coefficient name `pred` (from the independent variable's name) by the `lm` function. We don't need the name, so we assign it `NULL`.

Then we use equation (14.3). Note that the factor $1/n \cdot \sum_{i=i}^{n}(x_i - \overline{x})^2$ is the mean squared deviation of the predicted values from their mean:

```
> (valid.msd.pred <- mean((pred - mean(pred))^2))

[1] 0.3435529
```

and this is multiplied by the squared deviation of the slope of the regression of actual on predicted from 1:1:

```
> (valid.nu <- (1 - b)^2 * valid.msd.pred)

[1] 0.01456352

> valid.nu/valid.msd * 100

[1] 3.610135
```

Q112 : *What is the magnitude of the non-unity factor NU? What proportion of the MSD is attributed to the NU?* *Jump to A112* •

14.4.4 LC

The final aspect of model quality is the lack of correlation, LC.

Task 95 : Compute LC. •

We use equation (14.4). Note that the factor $1/n \cdot \sum_{i=i}^{n}(y_i - \overline{y})^2$ is the mean squared deviation of the measured values from their mean:

```
> (valid.msd.actual <- mean((actual - mean(actual))^2))

[1] 0.8810535
```

and then this is multiplied by the lack of fit in the regression of actual on predicted, $(1 - r^2)$. The fit is a field in the model summary; it can also be computed directly with `cor` function for this bivariate model:

```
> (r2 <- summary(regr.actual.pred)$r.squared)

[1] 0.5670314

> (r2 <- cor(actual, pred)^2)

[1] 0.5670314

> (valid.lc <- (1 - r2) * valid.msd.actual)
```

```
[1] 0.3814685

> valid.lc/valid.msd * 100

[1] 94.56186
```

Q113 : *What proportion of the MSD is attributed to LC?* *Jump to A113* •

Check that the three components add up to the MSD:

```
> print(valid.msd - (valid.sb + valid.nu + valid.lc))

[1] 3.330669e-16
```

Yes, within rounding error.

Q114 : *What is the weak point of the model? Is the linear form justified?* *Jump to A114* •

Challenge: Repeat the above analysis for several more random selections (splits 3/4 – 1/4) of subsets. Collect the statistics for MSD and its components and summarize them (minimum, maximum, IQR, range, mean). How much do MSD and its components change? How reliable then is information from a single split?

Challenge: Repeat the above analysis for another (or several more) split sizes, e.g., 1/20, 1/10, 1/5, 1/3, 1/2 evaluation sets. Again, take several selections at each split, and summarize them as in the previous challenge. At what split are the results least variable?

14.5 An inappropriate model form*

The main use of the evaluation analysis is to discover an inappropriate model form. We can see how this works by fitting such a model. The example we choose is a linear model *without* an intercept. That is, we force a zero grain yield to also have a zero straw yield. This was discussed above in §8.4, where its deficiencies in this case were revealed; here we analyze these with the model evaluation components of the previous section.

Q115 : *Would a plot with no grain yield necessarily have no straw yield?* *Jump to A115* •

Task 96 : Fit a linear model of straw predicted by grain, without an intercept, to the evaluation subset, and summarize it. •

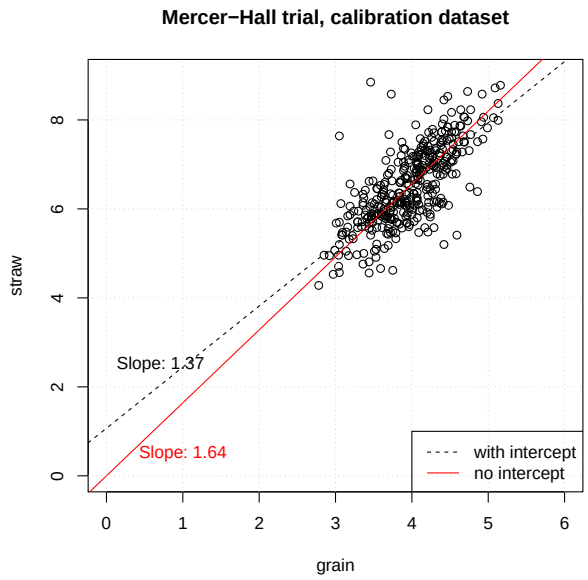
By default models fit with `lm` include an intercept; to remove it use the `-` formula operator to remove the intercept, symbolized by the term `1`.

```
> cal.straw.grain.00 <- lm(straw ~ grain - 1, data = mhw, subset = index.calib)
> summary(cal.straw.grain.00)

Call:
lm(formula = straw ~ grain - 1, data = mhw, subset = index.calib)

Residuals:
    Min      1Q  Median      3Q     Max
-2.1165 -0.3619  0.0450  0.3801  3.1764

Coefficients:
      Estimate Std. Error t value Pr(>|t|)
grain 1.639767   0.008036     204   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.621 on 374 degrees of freedom
Multiple R-squared:  0.9911,        Adjusted R-squared:  0.9911
F-statistic: 4.163e+04 on 1 and 374 DF,  p-value: < 2.2e-16
```

Task 97 : Plot the straw vs. grain yield of the evaluation set, with the with-intercept and no-intercept models shown as lines; include the $(0, 0)$ point. •

The graphing limits are specified with the `xlim` and `ylim` arguments to `plot`:

```
> plot(straw ~ grain, data = mhw, subset = index.calib, xlim = c(0,
+     6), ylim = c(0, 9))
> title("Mercer-Hall trial, calibration dataset")
> abline(cal.straw.grain, lty = 2)
> abline(cal.straw.grain.00, col = "red")
> grid()
> legend(4, 1, c("with intercept", "no intercept"), lty = c(2,
+     1), col = c("black", "red"))
> text(0, 2.5, paste("Slope:", round(coef(cal.straw.grain)[2],
+     2)), pos = 4)
> text(1, 0.5, paste("Slope:", round(coef(cal.straw.grain.00)[1],
+     2)), col = "red")
```



Q116 : *What is the effect of forcing the regression through* $(0, 0)$*? Can you determine by eye which one fits better the observations?* *Jump to A116* •

Task 98 : Compute the predicted values for the evaluation observations, using the no-intercept model. •

```
> pred <- predict.lm(cal.straw.grain.00, newdata = mhw[index.valid,
+     ])
> summary(pred)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.477   5.920   6.280   6.404   6.887   8.412
```

Task 99 : Plot the actual vs. predicted on a 1:1 graph, along with the 1:1 line and a regression of actual vs. predicted. •

```
> regr.actual.pred.00 <- lm(actual ~ pred)
> plot(actual ~ pred, ylab="Actual", xlab="Predicted", asp=1,
+      main="Mercer-Hall trial, straw yield, lbs/plot",
+      xlim=c(4.5,9), ylim=c(4.5,9));
> abline(regr.actual.pred.00, col="red")
> abline(0,1); grid()
> text(4.5, 8.5, paste("Gain:",
+                      round(coef(regr.actual.pred.00)[2], 2)),
+      pos=4, col="red")
> legend(7.5, 5, c("1:1","regression"), lty=1,
+        col=c("black","red"))
```

Task 100 : Compute the evaluation statistics. •

```
> (msd.00 <- mean((actual - pred)^2))

[1] 0.3949466

> (rmsep.00 <- sqrt(msd.00))

[1] 0.6284478

> (sb.00 <- (mean(pred) - mean(actual))^2)

[1] 0.01342833

> (nu.00 <- (1 - coef(regr.actual.pred.00)[2])^2 * mean((pred -
+     mean(pred))^2))

        pred
4.976105e-05

> (lc.00 <- (1 - cor(actual, pred)^2) * mean((actual - mean(actual))^2))

[1] 0.3814685
```

Task 101 : Compute the relative contribution of the model evaluation elements to the overall quality. •

```
> sb.00/msd.00 * 100

[1] 3.400036

> nu.00/msd.00 * 100

      pred
0.01259944
```

```
> lc.00/msd.00 * 100

[1] 96.58736

> msd.00 - (sb.00 + nu.00 + lc.00)

        pred
5.551115e-17
```

Q117 : *How do these statistics compare with those for the with-intercept model? Interpret them geometrically – what could be wrong with this model?* Jump to A117 •

14.6 Answers

A102 : *By row, within each column in the field. Adjacent plots, i.e., adjacent observations in the data frame, may not be independent. Thus there could be serial correlation.* Return to Q102 •

A103 : *Possible stratifying factors are field half (`in.north`), row, and column. But we saw in §9.3 that field half only explains a small amount of the variance, well under 10%; below in §17 we will see the same result for row and column.* Return to Q103 •

A104 : *A random sampling is indicated: (1) there is no useful stratifying factor; (2) there may be serial autocorrelation.* Return to Q104 •

A105 : *The coefficients agree fairly well; the percentage change from the full model to the calibration model is 23.5% for the intercept, -6.6% for the grain yield,* Return to Q105 •

A106 : *The actual yields are spread over a wider range, especially the maximum, and less concentrated at the central value (about 6.5 lbs plot^{-1}).* Return to Q106 •

A107 : *For the most part fairly well (points are bunched around the 1:1 line) but several points are very badly predicted. There does not appear to be any gain.* Return to Q107 •

A108 : *The RMSEP of evaluation is 0.635 lbs plot^{-1}; the residual mean square error of the model is 0.607 lbs plot^{-1}. These are similar but in this evaluation the RMSEP is a somewhat higher. The RMSEP is the mean prediction error of straw yield that we expect for the "same" experimental structure where we measure grain yield and use it to predict straw yield.* Return to Q108 •

A109 : *The bias is very small: -0.0859. This negative bias shows that the actual is*

a bit greater than the predicted, i.e., on average the model slightly under-predicts. But the squared bias SB is a very small proportion of the total error, 1.828%, i.e., less than 1%. This is not a weak point of the model. *Return to Q109* •

A110 : *The gain is slightly less than* 1*: 1.21. Thus high straw yields are (very slightly) under-predicted, and low straw yields are (very slightly) over-predicted.* *Return to Q110* •

A111 : *No, the probability that it would be a Type I error to reject the null hypothesis that the coefficient of the 1:1 model is in fact* 1 *(i.e., the coefficient of the 0:1 model is in fact* 0*) is 0.03, i.e., about 70% chance of an incorrect statement! So the hypothesis of 1:1 model fit stands.* *Return to Q111* •

A112 : *The non-unity factor NU is 0.01456352. This is again a very small part of the MSD, namely 3.61%, much less than 1%. This is not a weak point of the model.* *Return to Q112* •

A113 : *The lack of correlation LC is 0.3815; this is almost all of MSD, namely 94.562% of it.* *Return to Q113* •

A114 : *The only problem with the model is lack of correlation (LC), i.e., the prediction precision. There is almost no bias or gain. Thus the linear model form is well-justified.* *Return to Q114* •

A115 : *No, a plant could grow vegetatively but never flower or produce grain.* *Return to Q115* •

A116 : *The line through* $(0, 0)$ *has a steeper slope to reach the "cloud" of points, whereas the line with intercept has a positive intercept and so can have a gentler slope. It seems difficult to tell by eye which better fits the point cloud.* *Return to Q116* •

A117 : *The no-intercept model has higher overall error: its RMSEP is 0.628 compared to 0.635 for the full model. But, this is not due to lack of correlation (LC): these are 0.381 and 0.381 respectively, i.e., identical! This shows nicely the value of the decomposition – the problem with the no-intercept model is not its lack of precision, this is exactly the same as for the with-intercept model.*

Both squared biases (SB): are quite small: 0.013428 (no-intercept) and 0.007374 (intercept). Thus neither systematically over- or under-predicts.

This means the problem with the no-intercept model is in the non-unity slope (NU): 4.976e-05 (no-intercept) vs. 0.01456352. In the no-intercept model this is now a recognizable proportion (0.013%) of the total error. This is interpreted as a ***rotation****; this is in fact what happens when we forced the regression through the origin.* *Return to Q117* •

We are done with these models and some other variables, except for the main model of straw yield and its RMSEP (see §15), so clean up the workspace:

```
> rm(n, index.valid, index.calib, actual)
> rm(cal.straw.grain, pred)
> rm(valid.msd, rmse)
> rm(regr.actual.pred, regr.actual.pred.0, valid.bias, valid.sb,
+     valid.lc, b, valid.nu, valid.msd.pred, valid.msd.actual,
+     r2)
> rm(cal.straw.grain.00, regr.actual.pred.00, msd.00, rmsep.00,
+     sb.00, nu.00, lc.00)
```

15 Cross-validation*

In §14 the predictive success of a model was evaluated by evaluation against an independent dataset. Since we only had one dataset (the 500 plots), we were forced to create this set by a random split into calibration and evaluation sets. There are several problems here:

1. We lose precision in the model, because it's based on fewer observations;
2. The split is random, so that a different split (with the same proportions) would give different results.

An approach to evaluation that uses the insight of point (2) but retains precision is **leave-one-out cross validation**, abbreviated LOOCV. This is also known as leave-one-out **jackknifing** [10, 12], where our main interest is in the accuracy of the prediction.

> **Note:** The term "cross-validation" is used consistently in texts, papers and computer programs, so we use the term; however we consider it a form of evaluation, as explained in the previous §14.

The concept is simple:

1. For each observation:
   (a) remove it from the dataset, i.e., "leave one out";
   (b) compute the model parameters (e.g., slope and intercept of a simple linear regression);
   (c) use this model to predict at the left-out point;
   (d) calculate the prediction error for this one point.
2. Compute the evaluation statistics for the set of prediction errors (one for each observation), as in §14.4.

These evaluation statistics are assumed to apply to the single equation (parameterization) computed from all observations.

Task 102 : Write a function to compute the LOOCV fits for the linear

model of straw yield predicted by grain yield. This function should take as arguments: (1) a model to cross-validate and (2) the dataset on which to cross-validate it. It should return (1) the LOOCV predictions and (2) the coefficients for each LOOCV model fit. •

We use the `function` command to define a function; this was explained in §8.3.1. We define two arguments: (1) the model form to be cross-validated, named `model` within the function; (2) the dataset where the model should be applied, named `dset`.

This is the most complex function we've defined; here are some points of interest:

- The function is defined with the `function` command and stored in a workspace object with a name of our choosing; this name is then used to call the function;
- At the end of the function we use the `return` function to return a **list**, build with the `list` function, of (1) the LOOCV predictions as a vector named `pred`, (2) the coefficients of each of the models built omitting one observation as a matrix with one column per model coefficient, named `coef`. The former is used for the cross-validation, the latter to evaluate the robustness of the model to any unusual single observations.
- Both of the returned variables must be initialized before the `for` loop, which will fill in the rows one-by-one as each observation is omitted.
- The `colnames` function is used to assign names to the columns of the coefficients vector, using the `paste` function to create a list of coefficient names of the correct length, depending on the number of model parameters.
- Also note the use of the type conversion function `as.character` to convert a sequence of numbers into a character vector, suitable to be used in `paste`.
- The `for` flow control structure defines a **for-loop**: the first expression after the `for` is the list of **indices** which are used inside the loop. Here we specify the sequence `1: nrow(dset)`; this uses the `:` operator, shorthand for a continuous integer sequence which could also be specified with the `seq` command, i.e., `seq(1, nrow(dset), by=1)`.
- The value from this sequence is assigned to variable which we name `i`. This is only defined in the loop, and is used to specify observation numbers, both for omitting a row from the dataframe, i.e., `[-i,]`, and selecting just the one row, i.e., `[i,]`; both use the `[]` matrix selection operator.
- The results of each model fit (prediction and coefficients) are stored in the initialized vectors, at the correct slot, again using the loop index `i`.

Here is the function:

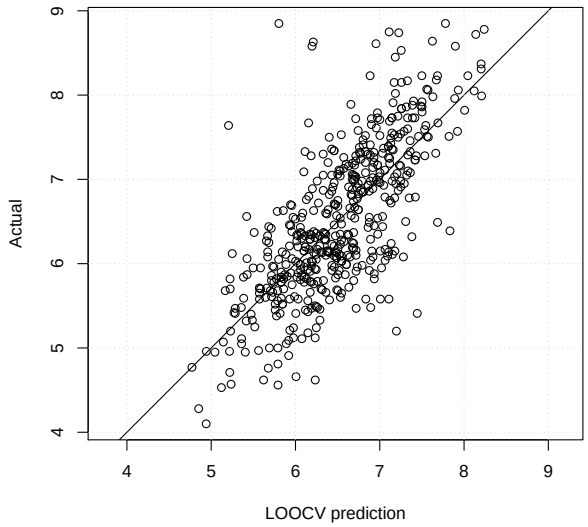
```
> Xval <- function(model, dset) {
+     pred <- rep(0, nrow(dset))
+     n <- length(coefficients(model))
+     coef <- matrix(0, nrow = nrow(dset), ncol = n)
+     colnames(coef) <- paste("b", as.character(0:(n - 1)), sep = "")
+     for (i in 1:nrow(dset)) {
+         m <- lm(formula(model), data = dset[-i, ])
+         pred[i] <- predict(m, newdata = dset[i, ])
+         coef[i, ] <- coefficients(m)
+     }
+     return(list(pred = pred, coef = coef))
+ }
```

Task 103 : Apply this function to the model of straw yield predicted by grain yield, and display the structure of the returned object. •

```
> xval.fit <- Xval(model.straw.grain, mhw)
> str(xval.fit)

List of 2
 $ pred: num [1:500] 6.06 6.69 7.32 6.44 6.06 ...
 $ coef: num [1:500, 1:2] 0.862 0.865 0.861 0.864 0.868 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "b0" "b1"
```

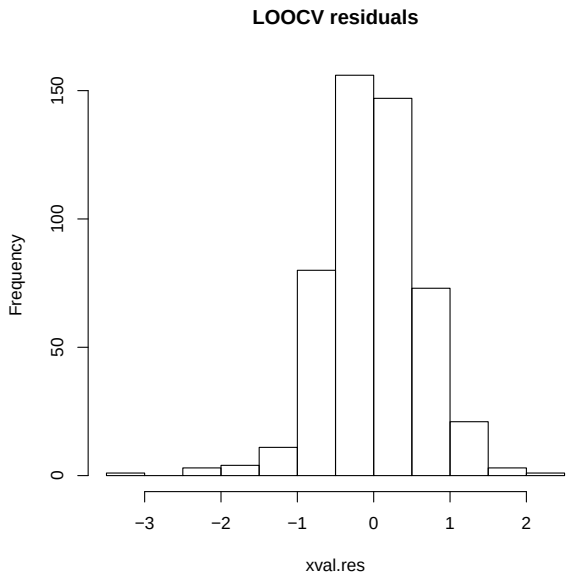
Task 104 : Display the actual observations of straw yield against the LOOCV fits, on a 1:1 line. •

```
> lim <- range(xval.fit$pred, mhw$straw)
> plot(mhw$straw ~ xval.fit$pred, asp = 1, xlim = lim, ylim = lim,
+     xlab = "LOOCV prediction", ylab = "Actual")
> abline(0, 1)
> grid()
```



Task 105 : Compute the cross-validation residuals, i.e., the actual straw yields less the LOOCV fits; summarize them and display as a histogram. •

```
> xval.res <- xval.fit$pred - mhw$straw
> summary(xval.res)

      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-3.0470000 -0.3748000 -0.0104200 -0.0000946  0.3552000  2.0340000

> hist(xval.res, main = "LOOCV residuals")
```



Recall, the RMSEP is the square root of the sum of squared residuals divided by the number of observations.

Q118 : *What is the LOOCV RMSEP? How does this compare to the RMSE of the fit (the internal measure) and the independent evaluation RMSE?* *Jump to A118* •

Recall, the single estimate of independent evaluation RMSE was computed in §14.4.

Here are (1) RMSEP from cross-validation; (2) RMSE from the internal fit; (3) RMSEP from the single evaluation:

```
> sqrt(sum(xval.res^2)/nrow(mhw))

[1] 0.6159743

> sqrt(sum(residuals(model.straw.grain)^2)/(model.straw.grain$df.residual))

[1] 0.614771

> print(valid.rmsep)

[1] 0.6351428
```

Challenge: Compute and interpret the measures of model quality developed in §14.4, i.e., the RMSEP broken down into bias, gain and scatter.

The function also returned the LOOCV model coefficients.

Task 106 : Summarize the LOOCV model coefficients, and compare with the best-fit coefficients (using all observations). •

```
> summary(xval.fit$coef, digits = 5)

       b0                b1
 Min.   :0.77899   Min.   :1.4214
 1st Qu.:0.86293   1st Qu.:1.4296
 Median :0.86674   Median :1.4304
 Mean   :0.86628   Mean   :1.4305
 3rd Qu.:0.87003   3rd Qu.:1.4313
 Max.   :0.90307   Max.   :1.4514

> coefficients(model.straw.grain)

(Intercept)       grain
  0.8662797   1.4304977
```

Note the use of the optional `digits` argument to `summary`, to show more than the default three significant digits.

Q119 : *How consistent are the LOOCV coefficients? Which one varied more? Why?* *Jump to A119* •

15.1 Answers

A118 : *The LOOCV RMSEP is 0.616 lb. plot^{-1}. The single-estimate evaluation RMSEP is 0.6351 lb. plot^{-1}. The internal RMSE of the fit is 0.6148 lb. plot^{-1}.*

The single evaluation RMSEP is the most conservative; this is because the evaluation is based on a large number of observations and a single model. Adjusting the model for each omitted observation (LOOCV) reduces the RMSEP to just a bit greater than the internal estimate.

Note that there is only one LOOCV estimate, only one internal fit estimate, whereas the single evaluation by splitting the dataset could be done many times, each with a different random split. This is similar to ***K-fold cross-validation****, where K is the proportion of the observations omitted each time.* *Return to Q118* •

A119 : *The slopes are very consistent; that is, leaving out any one observation hardly changes it: the total range is 0.03 compared to the single "best" value 1.4305. The intercepts are less consistent: the total range is 0.1241 compared to the single "best" value 0.8663. This shows that leaving out one very high or very low straw yield can move the line up or down.* *Return to Q119* •

16 Spatial analysis

To this point we have only considered the wheat yields in **feature** space (also known as **attribute** or **property** space). For example, the grain and straw yields form a two-dimensional 'space'. But we have ignored an additional piece of information: the **relative location** of the plots in the field. Mercer & Hall clearly stated that there could be hot spots or geographic trends, meaning that the plots are not necessarily **spatially independent**. We now investigate this.

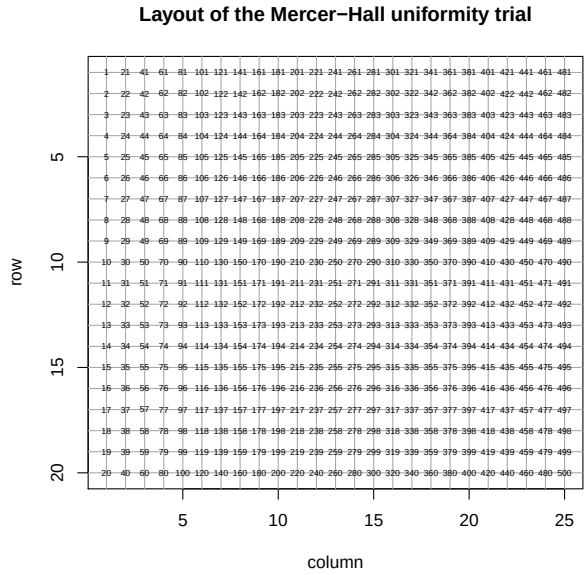
16.1 Geographic visualisation

We begin by visualisaing the agricultural field:

Task 107 : Make a correctly-scaled map of the plot locations in the field, showing the plot numbers. •

The plots are rectangular (longer N-S than wide E-W), so that by plotting the 25 columns and 20 rows in a square (the shape of the field, and the default for a bivariate `plot`), we get a geometrically-correct map. However, the default plotting function is from low to high indices, so that row 1 would be plotted at the bottom, when in fact it is at the top. We can specify the axis with the `ylim` argument, reversing the row order:

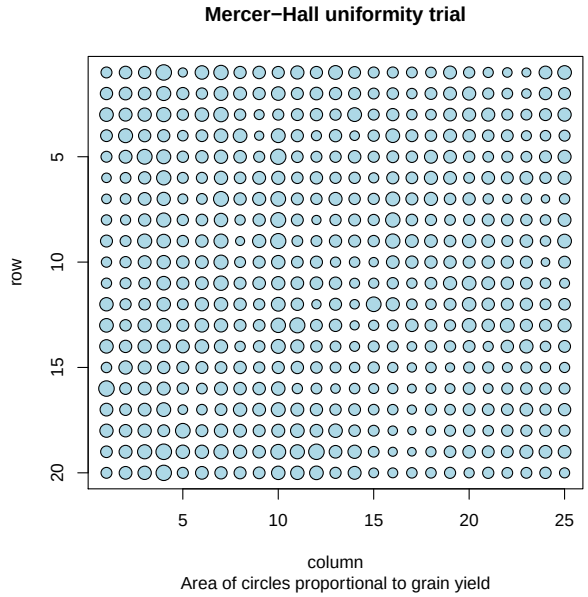
```
> plot(c, r, type = "n", xlab = "column", ylab = "row",
+     ylim = c(20, 1), main = "Layout of the Mercer-Hall uniformity trial")
> abline(v = 1:25, lty = 1, col = "darkgray")
> abline(h = 1:20, lty = 1, col = "darkgray")
> text(c, r, rownames(mhw), cex = 0.5)
```



Note the use of the `xlab` and `ylab` graphics parameter to specify the axis names. Also, the type of plot is specified with the `type` graphics argument, here set to `"n"`, meaning "no plot" (yet) – this just sets up the plot area and axes. The actual plotting is then done with the `text` function, using the `rownames` function to extract the plot number from the dataframe.

Task 108 : Make a **post-plot** of the grain yield, i.e. a map of the plot locations with symbol size proportional to the data value. •

```
> plot(c, r, pch=21, col="black", bg="lightblue", ylim=c(20,1),
+      xlab="column", ylab="row",
+      main="Mercer-Hall uniformity trial",
+      sub="Area of circles proportional to grain yield",
+      cex=2*grain/max(grain))
```



We can visualise this better by displaying each point in a colour ramp. First we classify the observations into octiles (eight groups) with the `cut` function, using the `quantile` function to compute the octiles.

```
> (q8 <- quantile(grain, seq(0, 1, length = 9)))

    0%  12.5%    25%  37.5%    50%  62.5%    75%  87.5%   100%
2.7300 3.4238 3.6375 3.7812 3.9400 4.0900 4.2700 4.4700 5.1600

> grain.c <- cut(grain, q8, include.lowest = T, labels = F)
> sort(unique(grain.c))

[1] 1 2 3 4 5 6 7 8
```

So the 500 yields have been grouped into eight classes.

A **colour ramp** is a list of colours in some visually-meaningful sequence. One example is produced by the `terrain.colors` function; the colours are given as hexidecimal numbers from `00` (absence of colour) to `FF` (saturation with the colour), for the three primary colours Red, Green and Blue:

```
> terrain.colors(8)

[1] "#00A600FF" "#3EBB00FF" "#8BD000FF" "#E6E600FF" "#E9BD3AFF"
[6] "#ECB176FF" "#EFC2B3FF" "#F2F2F2FF"
```

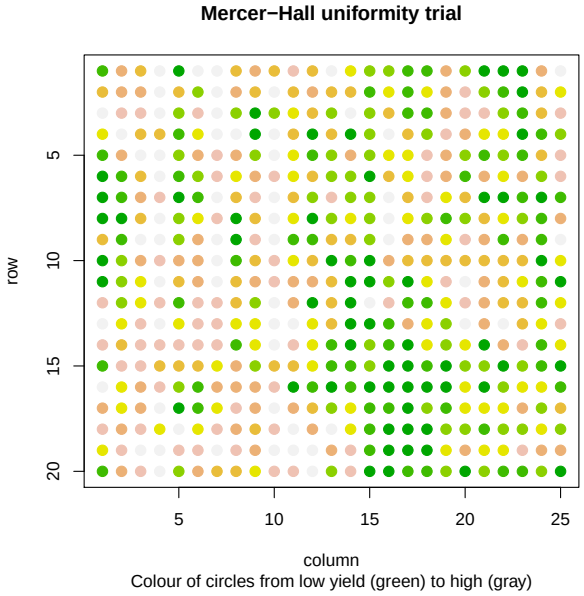
For example, the final colour in this ramp is `#F2F2F2`, which is a dark gray: equal saturations of the three primaries, and each of these has `#F2/#FF`, i.e. 95% saturation:

```
> 0xf2/0xff

[1] 0.94902
```

This example shows that hexidecimal numbers may be used in R; they are indicated with the prefix `0x`.

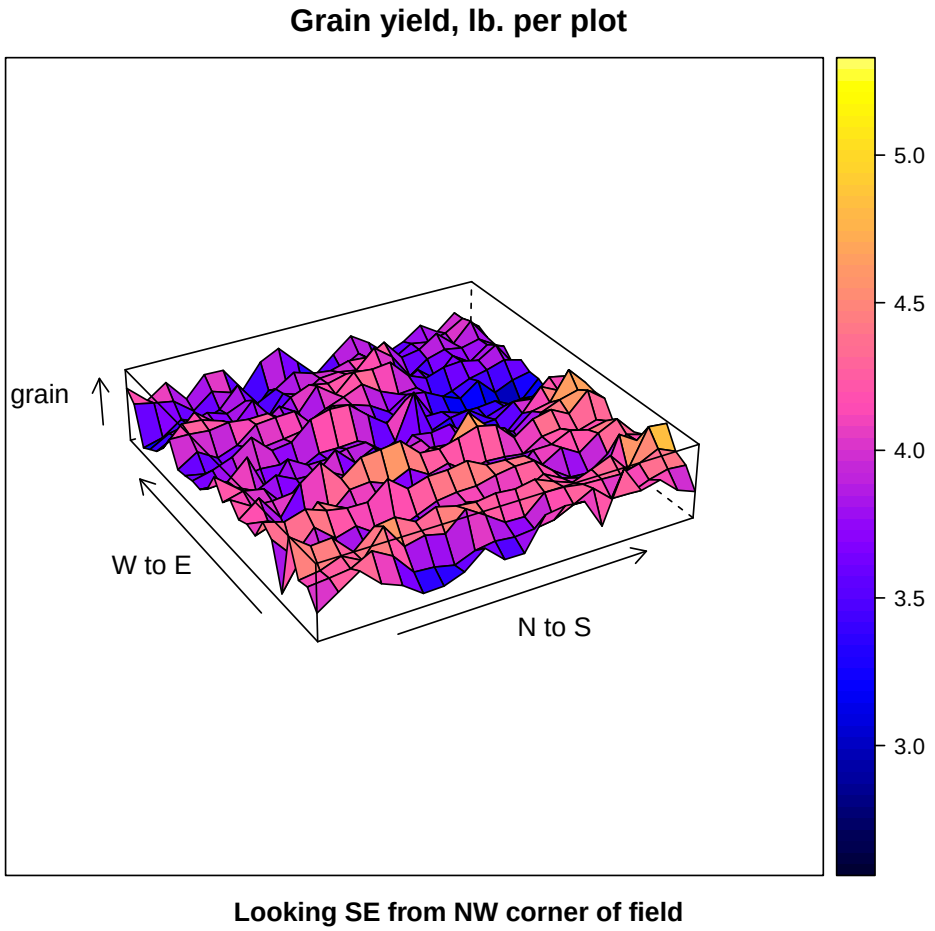
Now we use this colour ramp, selecting the appropriate colour for each quantile: dark green is lowest, white is highest.

```
> plot(c, r, pch=20, cex=2, bg="lightblue", ylim=c(20,1),
+      xlab="column", ylab="row",
+      main="Mercer-Hall uniformity trial",
+      sub="Colour of circles from low yield (green) to high (gray)",
+      col=terrain.colors(8)[grain.c])
```



Another way to visualize the field is with a **3D plot** using the `wireframe` graphics function of the `lattice` package. The optional `aspect` argument controls ratio between the horizontal axes, as well as the vertical exaggeration; the optional `screen` argument controls the viewing angle; this is a list of rotation from looking across the rows, and the rotation from vertical viewing. We use a different colour ramp, obtained by a call to the `bpy.colors` function of the `sp` package, which we load with `require`:

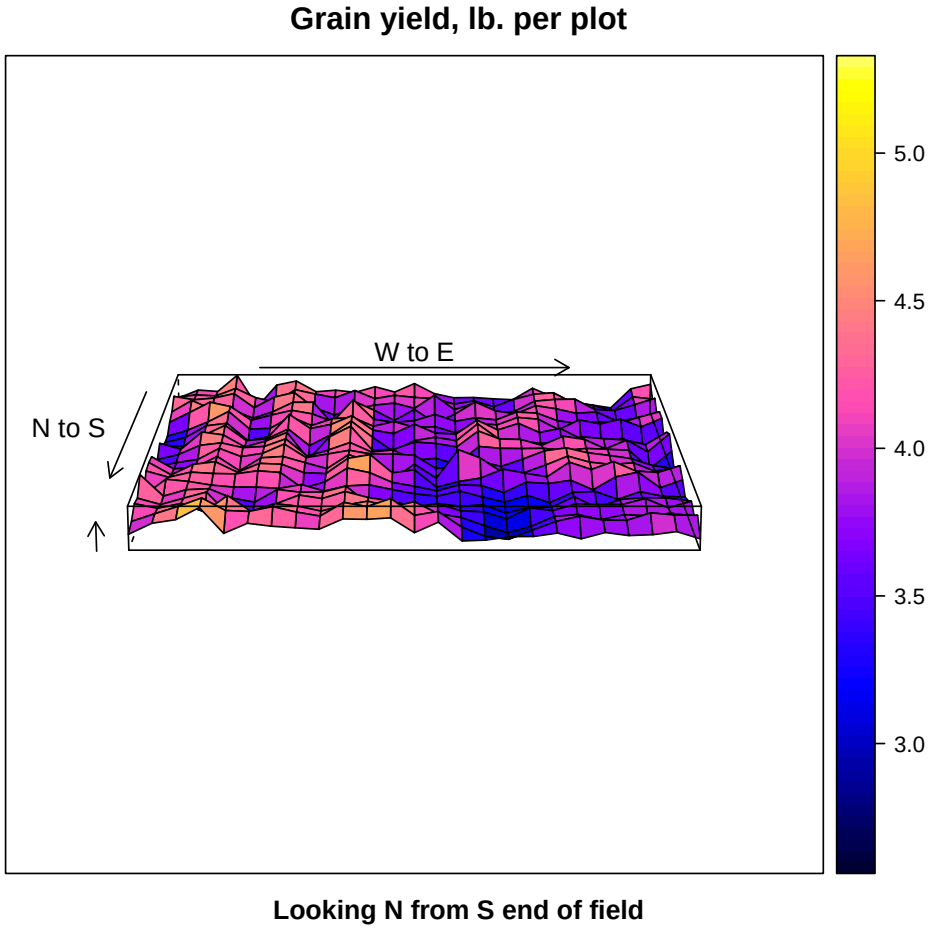
```
> require(sp)
> plot(wireframe(grain ~ r + c, data=mhw, drape=T,
+                aspect=c(1,.2), col.regions=bpy.colors(128),
+                main="Grain yield, lb. per plot",
+                screen= c(z=30, x=-60),
+                xlab="N to S", ylab="W to E",
+                sub="Looking SE from NW corner of field"))
```



Q120 : *Does there appear to be* ***local spatial dependence****, i.e. similar values near each other? Does this appear to vary over the field?* *Jump to A120* •

We view this from another perspective: from the S (so, W to the left, E to the right), from somewhat a lower viewing angle, and with less vertical exaggeration; this corresponds to figure 1 of McBratney and Webster [31]:

```
> plot(wireframe(grain ~ r + c, data=mhw, drape=T,
+                aspect=c(1,.08), col.regions=bpy.colors(128),
+                main="Grain yield, lb. per plot",
+                screen= c(z=270, x=-75), zlab="",
+                xlab="N to S", ylab="W to E",
+                sub="Looking N from S end of field"))
```



16.2 Setting up a coördinate system

Analysis of the **spatial structure** requires metric coördinates, rather than row and column numbers, since the plots are not square.

Task 109 : Determine the field size and the plot dimensions. •

The original experiment was in English units, but we will use metres. So, we start with some conversions to get the dimensions of each plot:

First, some conversion factors; note that 1 ha = 10 000 m²:

```
> ha2ac <- 2.471054
> ft2m <- 0.3048
> (field.area <- 10000/ha2ac)

[1] 4046.9
```

Then we divide the side of the 1-acre field evenly into 20 rows and 25 columns to obtain the dimensions in meters and the area in m²:

```
> (plot.area <- field.area/500)
```

```
[1] 8.0937

> (plot.len <- sqrt(field.area)/20)

[1] 3.1807

> (plot.wid <- sqrt(field.area)/25)

[1] 2.5446

> rm(ha2ac, ft2m, field.area)
```

Task 110 : Compute the total length and width in metres, confirm they are equal (because the field is a square, and confirm that they multiply to 1 ha (4 045.9 m^2). •

```
> (tot.len <- plot.len * 20)

[1] 63.615

> (tot.wid <- plot.wid * 25)

[1] 63.615

> tot.len * tot.wid

[1] 4046.9

> rm(tot.len, tot.wid)
```

Task 111 : Compute coördinates for the centre of each plot. •

Coördinates are assigned from an arbitrary origin of $(0, 0)$ at the SW corner of the field, so that the coördinates of the centre of plot $[1, 1]$ are half the plot size in both directions:

```
> plot.wid/2

[1] 1.2723

> plot.len/2

[1] 1.5904
```

Now we build a data frame of coordinates; first with the `seq` function to make vectors of the midpoints of the E and N directions, respectively; and then with the `expand.grid` function to make a dataframe with one row per combination:

```
> nrow <- length(unique(r))
> ncol <- length(unique(c))
> sx <- seq(plot.wid/2, plot.wid/2 + (ncol - 1) * plot.wid,
+     length = ncol)
> sy <- seq(plot.len/2 + (nrow - 1) * plot.len, plot.len/2,
+     length = nrow)
```

```
> xy <- expand.grid(x = sx, y = sy)
> rm(nrow, ncol, sx, sy)
```

Note the order of the y sequence: starting with the highest coordinate for row 1 (which is at the top of the plot).

We keep `plot.wid`, `plot.len`, and `plot.area` to be used later.

16.3 Loading add-in packages

For most of the spatial analysis we will use two **add-in packages**; these are representative of the hundreds which have been implemented by practising statisticians. Here we will use `sp` package [35], which is a foundation for spatially-explicit analysis in R, and the `gstat` package [34], which is an R implementation of the **gstat** geostatistics program [36]. Both of these are extensively discussion and illustrated in the textbook "Applied Spatial Data Analysis with R" by Bivand et al. [5].

When R starts, a number of basic packages are loaded; we can see these with the `search` function. Additional packages may be **loaded** with the `library` or `require` functions; both of these ensures that the package isn't already loaded.

Task 112 : Load the `sp` and `gstat` packages, and also the `lattice` graphics package. •

```
> require(sp); require(gstat); require(lattice)
```

16.4 Creating a spatially-explicit object

The `sp` package adds a number of **spatial data types**, i.e. new **object classes**.

Task 113 : Copy the `mhw` dataframe to a new object and convert the copy to class `SpatialPointsDataFrame`. •

By making a copy we have the data in two forms, spatial and non-spatial, so we don't have to keep converting between them.

We do this by adding the computed coordinates to the data frame with the `coordinates` function; this automatically converts to the spatial data type defined by the `sp` package:

```
> mhw.sp <- mhw
> coordinates(mhw.sp) <- xy
> summary(mhw.sp)

Object of class SpatialPointsDataFrame
Coordinates:
     min    max
x 1.2723 62.343
y 1.5904 62.025
Is projected: NA
```

```
proj4string : [NA]
Number of points: 500
Data attributes:
       r               c          grain          straw
 Min.   : 1.00   Min.   : 1   Min.   :2.73   Min.   :4.10
 1st Qu.: 5.75   1st Qu.: 7   1st Qu.:3.64   1st Qu.:5.88
 Median :10.50   Median :13   Median :3.94   Median :6.36
 Mean   :10.50   Mean   :13   Mean   :3.95   Mean   :6.51
 3rd Qu.:15.25   3rd Qu.:19   3rd Qu.:4.27   3rd Qu.:7.17
 Max.   :20.00   Max.   :25   Max.   :5.16   Max.   :8.85
      gsr          in.north
 Min.   :0.391   Mode :logical
 1st Qu.:0.574   FALSE:250
 Median :0.604   TRUE :250
 Mean   :0.611   NA's :0
 3rd Qu.:0.642
 Max.   :0.850
```

Q121 : *What is the data type of the* `mhw.sp` *object? What is the* ***bounding box****, i.e. limits of the the coördinates?* *Jump to A121* •

Now that we've built the spatial object, we can save it for later use in another session:

```
> save(mhw.sp, file = "mhw_spatial.RData")
```

16.5 More geographic visualisation

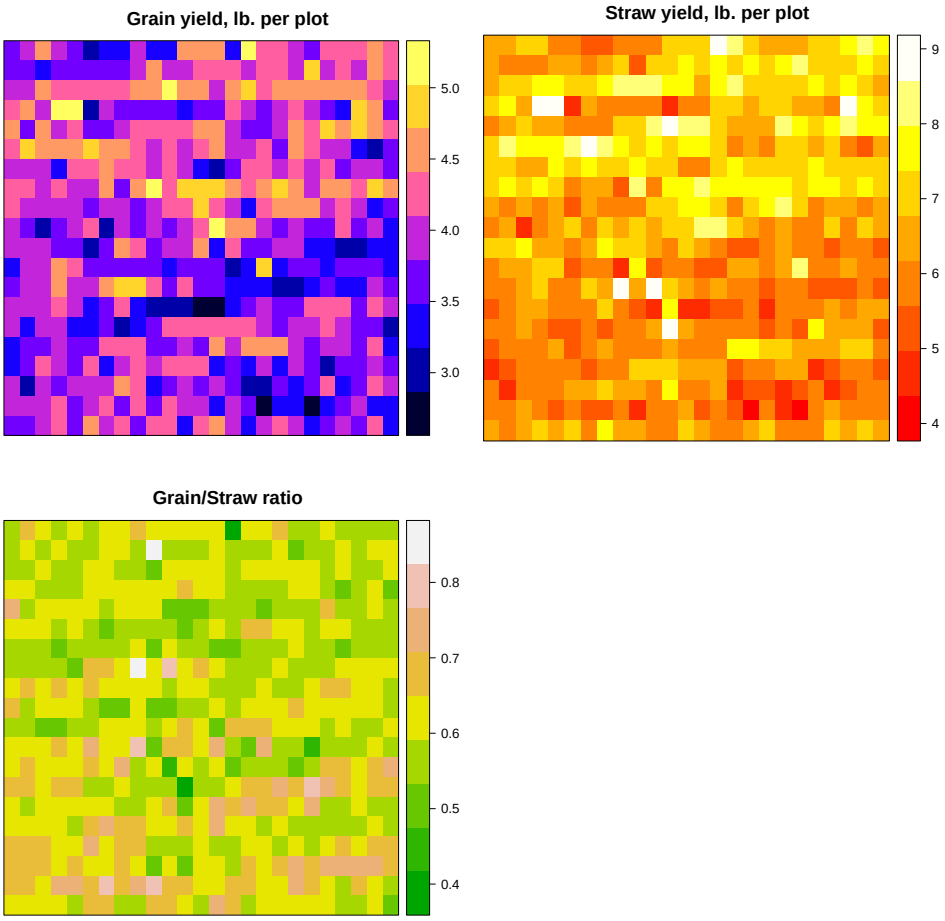
Once an object is in a spatial class, the `spplot` function can be used to make a nicely-coloured post plot:

Task 114 : Plot the grain, straw, and their ratio, coloured by their octile. •

We take this chance to illustrate some more colour ramps, produced by the `bpy.colors` and `heat.colors` functions, as well as the `terrain.colors` function we saw before. To put several plots on the same page, we create each plots with `spplot` and save it in a local variable; we then use the generic `print` function, with the optional `more` argument. Further, to give the impression of plots rather than points, we use the `as` generic method to convert the `mhw.sp` object to a spatial object of class `SpatialPixelsDataFrame`.

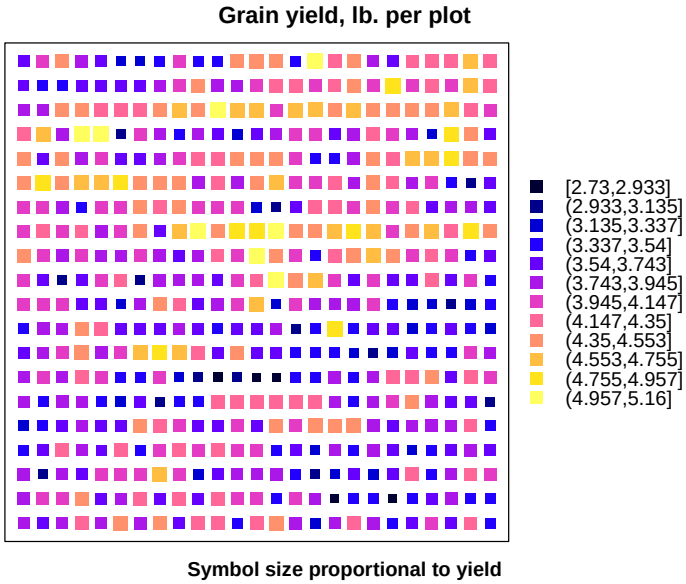
```
> mhw.sp.pix <- as(mhw.sp,"SpatialPixelsDataFrame")
> f1 <- spplot(mhw.sp.pix, zcol="grain", cuts=8,
+              col.regions=bpy.colors(64),
+              main="Grain yield, lb. per plot", key.space="right")
> f2 <- spplot(mhw.sp.pix, zcol="straw", cuts=8,
+              col.regions=heat.colors(64),
+              main="Straw yield, lb. per plot", key.space="right")
> f3 <- spplot(mhw.sp.pix, zcol="gsr", cuts=8,
+              col.regions=terrain.colors(64),
+              main="Grain/Straw ratio", key.space="right")
```

```
> print(f1, split=c(1,1,2,2), more=T)
> print(f2, split=c(2,1,2,2), more=T)
> print(f3, split=c(1,2,2,2), more=F)
> rm(f1, f2, f3)
```



And we can make a postplot with both colour and size, although its value for visualization is questionable:

```
> print(spplot(mhw.sp, zcol="grain", pch=15, cex=1.6*grain/max(grain),
+              cuts=12,
+              col.regions=bpy.colors(64), main="Grain yield, lb. per plot",
+              sub="Symbol size proportional to yield",
+              key.space="right"))
```



Challenge: Display grain and straw yields on adjacent plots, using a gray-scale colour ramp. (Hint: see §B). What are the advantages and disadvantages, for visualization, of using the gray scale vs. colour ramps?

16.6 Answers

A120 : *There are clear clusters of similar values, e.g. the patch of low values centred near (16, 18). Clusters seem more obvious in the north than the south.* *Return to Q120* •

A121 : `SpatialPointsDataFrame`*; the bounding box is 1.2723 to 62.3426 (W-E) and 1.5904 to 62.0245 (S-N)* *Return to Q121* •

17 Spatial structure

Now that we have a spatially-explicit object, we can examine it for its **spatial structure**. This can be of two kinds: a **trend** across the entire area or a **local structure** that does not depend on absolute location.

17.1 Spatial structure: trend

One possibility for spatial structure is a **trend** across the field.

Task 115 : Explore whether there is any trend in grain yield by row or column. •

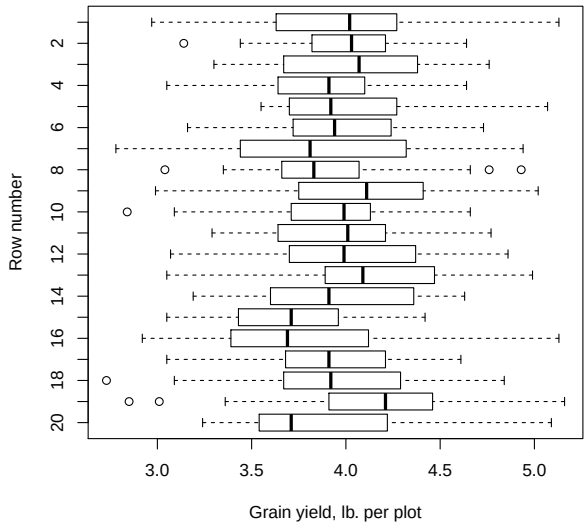
One way to do this is to compute the row and column mean yields, and then sort them from lowest to highest with the `sort` function:

```
> sort(by(grain, r, mean), decreasing = FALSE)

r
    15     16      7      8     20     17     10      4     18     14
3.7072 3.7432 3.8708 3.8796 3.8816 3.9012 3.9136 3.9144 3.9164 3.9352
     6     11      1     12      2      5      3      9     13     19
3.9536 3.9540 3.9576 3.9848 4.0072 4.0276 4.0420 4.0788 4.1160 4.1880

> sort(by(grain, c, mean), d = F)

c
    17     15     24     14     18     22     23      5     21     16
3.5280 3.6075 3.6565 3.7390 3.7545 3.7585 3.7925 3.8150 3.8165 3.8635
    19     12     13      1      9     25      8      2     20      6
3.8740 3.8955 3.8970 3.9165 3.9420 3.9450 3.9635 3.9650 4.0025 4.0570
    11      3      7     10      4
4.1125 4.2820 4.4630 4.5280 4.5410
```

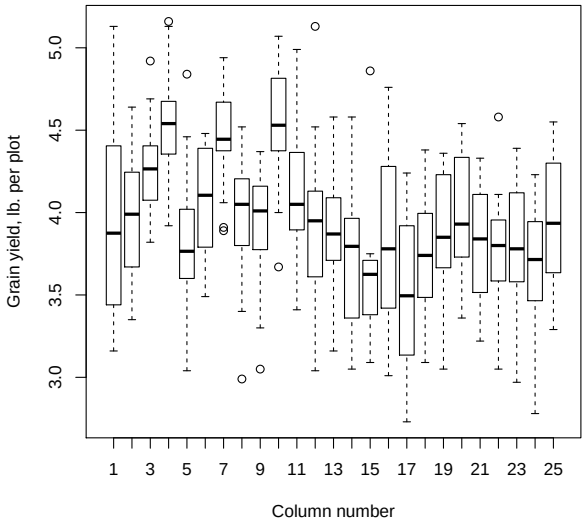
Q122 : *Does there appear to be any trend or pattern in the sequence of row or column numbers?* *Jump to A122* •

We can see both the means and variability with a grouped boxplot, first by row and then by column. Note the use of the `xlim` argument to display the row boxplots in correct geographical order.

```
> boxplot(grain ~ r, horizontal = T, data = mhw, xlim = c(20,
+     1), ylab = "Row number", xlab = "Grain yield, lb. per plot")
```




```
> boxplot(grain ~ c, data = mhw, xlab = "Column number",
+     ylab = "Grain yield, lb. per plot")
```



Q123 : *Does there appear to be any pattern by row or column, either in the median yield or the variability with each row or column?* *Jump to A123* •

Although only the columns show a slight trend, there could be a trend not oriented with these.

Task 116 : Compute a **first-order trend surface** of grain yield. •

```
> ts1 <- lm(grain ~ coordinates(mhw.sp))
> summary(ts1)

Call:
lm(formula = grain ~ coordinates(mhw.sp))

Residuals:
    Min      1Q  Median      3Q     Max
-1.1352 -0.2936  0.0069  0.3140  1.1711

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           3.731260   0.051850   71.96  < 2e-16 ***
coordinates(mhw.sp)x -0.000664   0.001067   -0.62     0.53
coordinates(mhw.sp)y  0.007498   0.001068    7.02  7.2e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.438 on 497 degrees of freedom
Multiple R-squared:  0.0909,        Adjusted R-squared:  0.0873
F-statistic: 24.9 on 2 and 497 DF,  p-value: 5.14e-11
```

Q124 : *Is there a significant trend? How much of the variance is explained?* *Jump to A124* •

This is not a promising approach, so we remove the trend surface object from the workspace:

```
> rm(ts1)
```

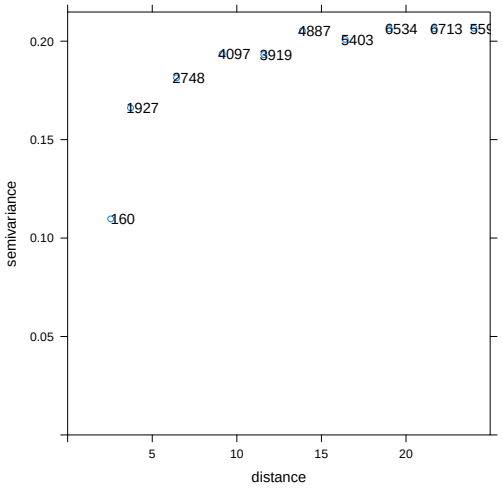
17.2 Spatial structure: local

There is only a very weak trend in grain yield; but are there hotspots?

Task 117 : Compute and display the variogram of the grain yield. •

We use the `variogram` function of the `gstat` package to analyze the local spatial structure. We also specify the optional `plot.numbers = T` argument to print the number of point-pairs next to the variogram values; the optional `width` argument to specify the bin size (here, the plot width), and the optional `cutoff` argument (by default it is 1/3 of the largest distance between point pairs); here it is 10 plot widths.

```
> v <- variogram(grain ~ 1, mhw.sp,
+                cutoff=plot.wid*10, width=plot.wid)
> print(plot(v, plot.numbers=T))
```



Q125 : *Describe the shape of the variogram. Is there evidence of local spatial structure? What is the approximate* ***range*** *of local spatial dependence, i.e. the separation at which the experimental variogram reaches its* ***sill*** *(maximum)?* *Jump to A125* •

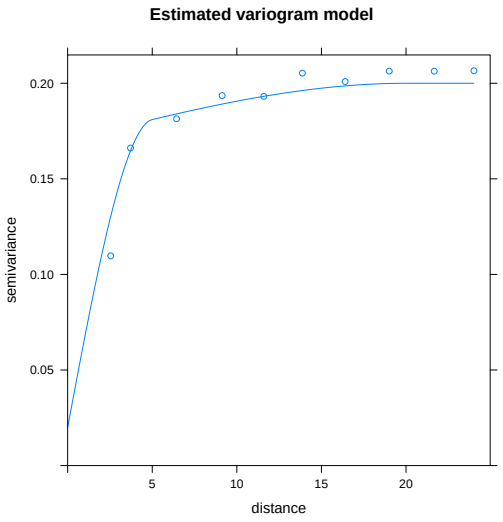
Q126 : *Across how many adjacent plots is there expected to be some spatial dependence?* *Jump to A126* •

We now try to fit a theoretical variogram model to this empirical variogram. There appear to be two structures:

1. a short-range to about 5 m with rapid increase in semivariance with separation; this section is difficult to model because of the few point-pairs in the closest bin;
2. a gradual increase to a sill at about 18 m.

The total sill appears to be about 0.20, of which the nugget is about 0.02, the first partial sill about 0.15 and the second about 0.03. We initialize the variogram with these parameters. We use the the `vgm` (specify a variogram) function twice, with the `add.to` argument the second time to combine variogram models.

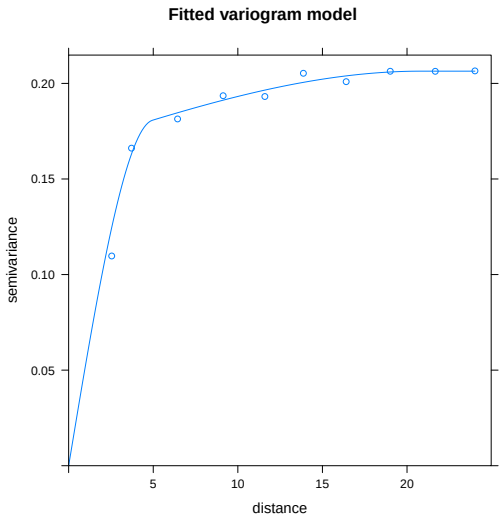
```
> (vm <- vgm(0.15, "Sph", 5, 0.02))

  model psill range
1   Nug  0.02     0
2   Sph  0.15     5

> (vm <- vgm(0.03, "Sph", 20, add.to = vm))

  model psill range
1   Nug  0.02     0
2   Sph  0.15     5
3   Sph  0.03    20

> print(plot(v, model = vm, main = "Estimated variogram model"))
```



We then adjust the variogram with the `fit.variogram` function:

```
> (vmf <- fit.variogram(v, vm))

  model    psill   range
1   Nug 0.000000  0.0000
2   Sph 0.166884  4.9488
3   Sph 0.039522 20.8112
```

```
> print(plot(v, model = vmf, main = "Fitted variogram model"))
```



The fit has reduced the nugget to zero; so at each point (inside a plot) there should be, as theory predicts, no spatial dependence. The nested structure clearly suggests that most of the spatial variability is within the first 5 m, so grouping a few plots should greatly reduce the between-plot variability of an experiment (Mercer & Hall's objective).

Q127 : *The zero nugget also implies no measurement error. Is that a valid assumption in this case? Why or why not? What should be the minimum nugget variance?* *Jump to A127* •

17.3 Absence of spatial structure*

In this section we show that spatial data does not necessarily have spatial structure. This also is a chance to investigate some of R's facilities for simulation. We can see how this field might look if there were no spatial dependence, i.e. if the variation in yields analyzed in §7 were **randomly** distributed across the field. We do this by applying the `sample` function to the vector of yields, to take a sample of the size as the original vector (extracted with the `length` function) without replacement. This ensures that each recorded yield appears once in the new vector.

Task 118 : Permute the vector of grain yields into a random order, compare to the original vector. •

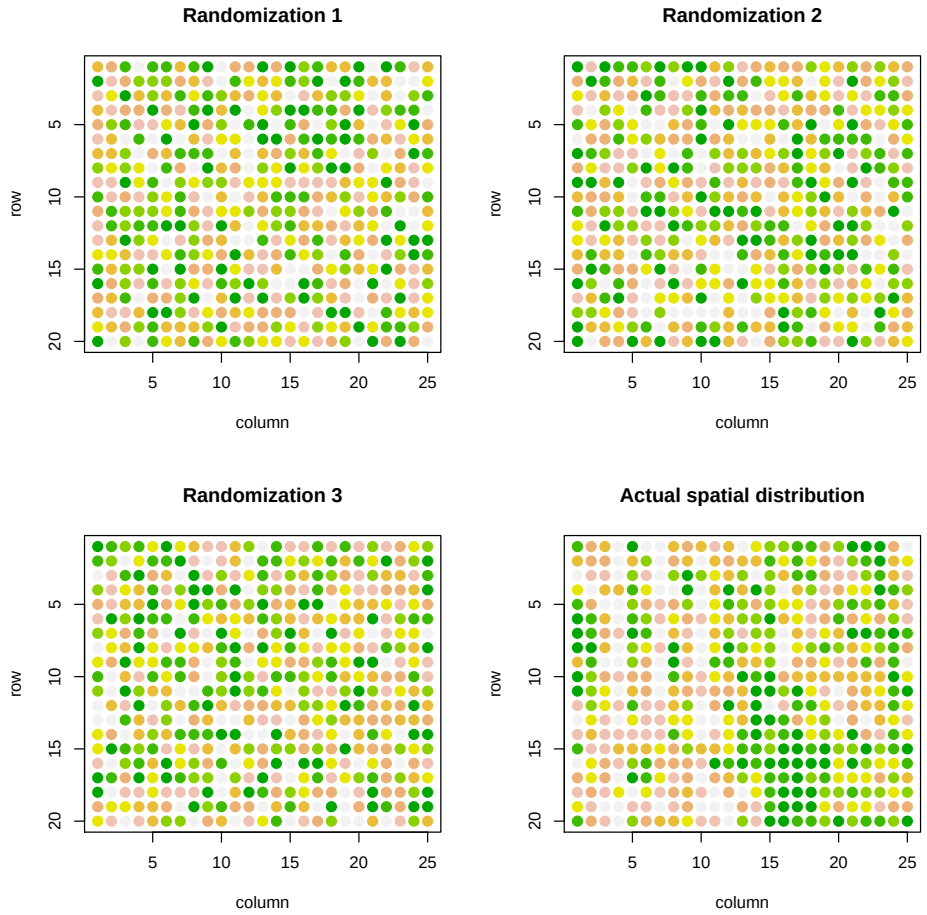
We do this several times to see the effect of randomization. The `head` function displays the first few elements of a vector; the `sort` method sorts them. The `set.seed` function ensures that your results match those presented here; of course you can experiment with other randomizations. We show the first few records of the samples, both unsorted and sorted, with the `head` function.

```
> set.seed(4502)
> head(mhw$grain)

[1] 3.63 4.07 4.51 3.90 3.63 3.16

> head(s1 <- sample(mhw$grain, length(mhw$grain), replace = FALSE))

[1] 4.06 2.97 4.47 4.06 4.20 4.29

> head(s2 <- sample(mhw$grain, length(mhw$grain), replace = FALSE))

[1] 3.05 4.27 3.92 4.46 3.55 4.49

> head(s3 <- sample(mhw$grain, length(mhw$grain), replace = FALSE))

[1] 3.42 3.46 5.07 3.67 4.10 4.46

> head(s1)

[1] 4.06 2.97 4.47 4.06 4.20 4.29

> head(s2)

[1] 3.05 4.27 3.92 4.46 3.55 4.49

> head(s3)

[1] 3.42 3.46 5.07 3.67 4.10 4.46
```

Q128 : *Do the permutations have the same elements as the original vector? What is different?* *Jump to A128* •

Task 119 : Display the spatial pattern of the randomized yields. •

```
> par(mfrow=c(2,2))
> plot(mhw$c, mhw$r, pch=20, cex=2, bg="lightblue", ylim=c(20,1),
+      xlab="column", ylab="row", main="Randomization 1",
+      col=terrain.colors(8)[cut(s1, q8, include.lowest=T, labels=F)])
> plot(mhw$c, mhw$r, pch=20, cex=2, bg="lightblue", ylim=c(20,1),
+      xlab="column", ylab="row", main="Randomization 2",
+      col=terrain.colors(8)[cut(s2, q8, include.lowest=T, labels=F)])
> plot(mhw$c, mhw$r, pch=20, cex=2, bg="lightblue", ylim=c(20,1),
+      xlab="column", ylab="row", main="Randomization 3",
+      col=terrain.colors(8)[cut(s3, q8, include.lowest=T, labels=F)])
> plot(mhw$c, mhw$r, pch=20, cex=2, bg="lightblue", ylim=c(20,1),
+      xlab="column", ylab="row", main="Actual spatial distribution",
+      col=terrain.colors(8)[grain.c])
> par(mfrow=c(1,1))
```

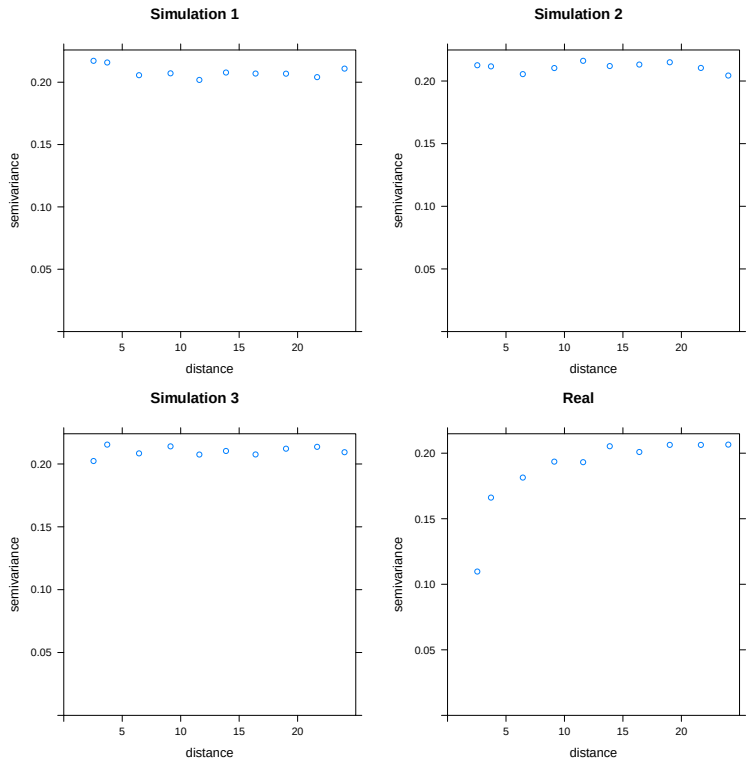


Q129 : *How do the randomizations differ from each other and the original spatial pattern?* *Jump to A129* •

It may be difficult to see that the randomizations have no spatial structure. So, we examine this with variograms, as in §17.2, after converting the simulated objects to spatial object.

```
> s1 <- data.frame(s1); coordinates(s1) <- xy
> s2 <- data.frame(s2); coordinates(s2) <- xy
> s3 <- data.frame(s3); coordinates(s3) <- xy
> pv <- plot(variogram(grain ~ 1, mhw.sp, cutoff=plot.wid*10,
+                      width=plot.wid), main="Real")
> p1 <- plot(variogram(s1 ~ 1, s1, cutoff=plot.wid*10,
+                      width=plot.wid), main="Simulation 1")
> p2 <- plot(variogram(s2 ~ 1, s2, cutoff=plot.wid*10,
+                      width=plot.wid), main="Simulation 2")
> p3 <- plot(variogram(s3 ~ 1, s3, cutoff=plot.wid*10,
+                      width=plot.wid), main="Simulation 3")

> print(p1, split = c(1, 1, 2, 2), more = T)
> print(p2, split = c(2, 1, 2, 2), more = T)
```

```
> print(p3, split = c(1, 2, 2, 2), more = T)
> print(pv, split = c(2, 2, 2, 2), more = F)
```



Q130 : *Are the variograms of simulated fields similar to each other? To the variogram of the actual spatial arrangement of plots? What is the evidence that the simulated fields (actual yields randomly assigned to plots) has no spatial dependence?* *Jump to A130* •

The conclusion from this section must be that spatial dependence is not always present! It must be verified by variogram analysis or similar (e.g. spatial autocorrelograms) or trend surface analysis.

Remove the temporary variables from the simulation displays:

```
> rm(xy, q8, grain.c, s1, s2, s3, pv, p1, p2, p3)
```

17.4 Spatial structure of field halves*

In §9 we computed an indicator variable to show which half of the field each plot is in. In a spatial analysis we may now ask whether these two halves have different spatial structures.

Task 120 : Separate the dataset into two halves, one for each field half. •

To get a suitable data structure we use the `split` function to create one object with a list of two data frames, one for each half. We then assign their coordinates.

```
> mhw.sp.ns <- split(as.data.frame(mhw.sp), in.north)
> coordinates(mhw.sp.ns$T) <- ~x + y
> coordinates(mhw.sp.ns$F) <- ~x + y
> summary(mhw.sp.ns)

      Length Class                  Mode
FALSE   8    data.frame             list
TRUE    8    data.frame             list
T     250    SpatialPointsDataFrame S4
F     250    SpatialPointsDataFrame S4

> summary(mhw.sp.ns$T)

Object of class SpatialPointsDataFrame
Coordinates:
     min    max
x 1.2723 62.343
y 1.5904 62.025
Is projected: NA
proj4string : [NA]
Number of points: 250
Data attributes:
       r               c           grain           straw
 Min.   : 1.0   Min.   : 1   Min.   :2.78   Min.   :4.10
 1st Qu.: 3.0   1st Qu.: 7   1st Qu.:3.66   1st Qu.:5.73
 Median : 5.5   Median :13   Median :3.97   Median :6.14
 Mean   : 5.5   Mean   :13   Mean   :3.96   Mean   :6.28
 3rd Qu.: 8.0   3rd Qu.:19   3rd Qu.:4.27   3rd Qu.:6.86
 Max.   :10.0   Max.   :25   Max.   :5.13   Max.   :8.64
      gsr         in.north
 Min.   :0.482   Mode:logical
 1st Qu.:0.596   TRUE:250
 Median :0.635   NA's:0
 Mean   :0.636
 3rd Qu.:0.669
 Max.   :0.848
```

Task 121 : Compute the variograms for each half, and plot these along with the combined variogram. •

We first compute the variograms for the two field halves; we already have the variogram for the entire field.

```
> v.n <- variogram(grain ~ 1, mhw.sp.ns$T, cutoff = 30)
> v.s <- variogram(grain ~ 1, mhw.sp.ns$F, cutoff = 30)
```

We now compute the figures, but do not print them right away; instead we store them as plotting objects:

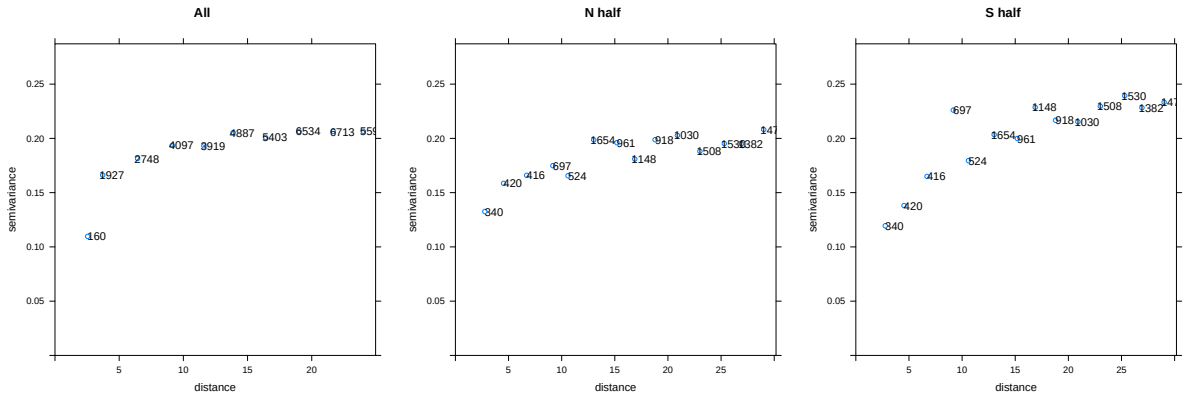
```
> g.max = max(v$gamma, v.n$gamma, v.s$gamma)*1.2
> plot.vgm.all <- plot(v, plot.numbers=T,
+                      main="All", ylim=c(0,g.max))
> plot.vgm.N <- plot(v.n, plot.numbers=T,
+                    main="N half", ylim=c(0,g.max))
```

```
> plot.vgm.S <- plot(v.s, plot.numbers=T,
+                    main="S half", ylim=c(0,g.max))
```

Note how we compute a common vertical axis from the maximum value of all three variograms, so we can compare them side-by-side.

Now we print these on one screen, specifying their positions in the plot:

```
> print(plot.vgm.all, split = c(1, 1, 3, 1), more = T)
> print(plot.vgm.N, split = c(2, 1, 3, 1), more = T)
> print(plot.vgm.S, split = c(3, 1, 3, 1), more = F)
```

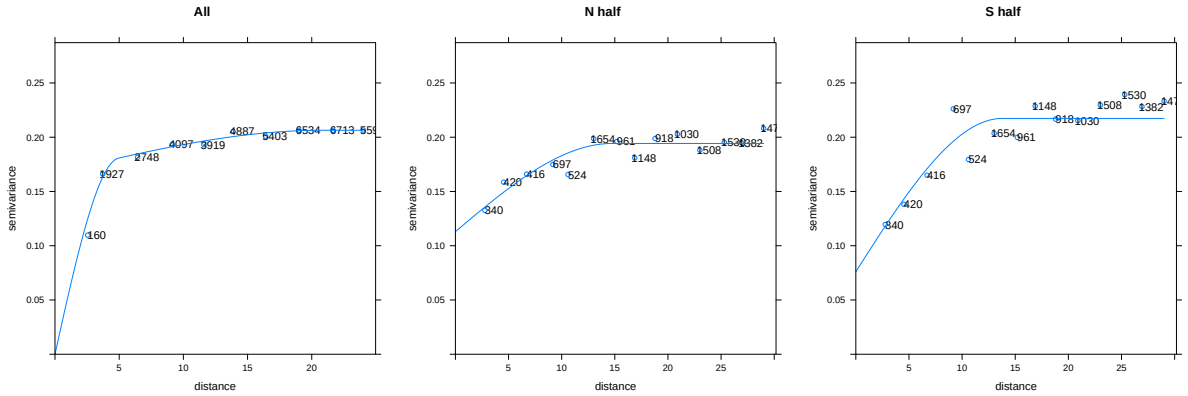


We now try to model the half-field variograms, as we did for the whole field in the previous section. The half-field variograms do not seem to show the nested structure of the whole-field variogram.

```
> (vmS <- vgm(0.14, "Sph", 20, 0.09))

  model psill range
1   Nug  0.09     0
2   Sph  0.14    20

> (vmN <- vgm(0.08, "Sph", 13, 0.11))

  model psill range
1   Nug  0.11     0
2   Sph  0.08    13

> (vmSf <- fit.variogram(v.s, vmN))

  model    psill  range
1   Nug 0.076024  0.000
2   Sph 0.141284 13.787

> (vmNf <- fit.variogram(v.n, vmS))

  model    psill  range
1   Nug 0.112956  0.000
2   Sph 0.081287 14.747
```

```
> plot.vgm.all <- plot(v, plot.numbers = T, main = "All",
+     model = vmf, ylim = c(0, g.max))
> plot.vgm.N <- plot(v.n, plot.numbers = T, main = "N half",
+     model = vmNf, ylim = c(0, g.max))
> plot.vgm.S <- plot(v.s, plot.numbers = T, main = "S half",
+     model = vmSf, ylim = c(0, g.max))
```

```
> print(plot.vgm.all, split = c(1, 1, 3, 1), more = T)
> print(plot.vgm.N, split = c(2, 1, 3, 1), more = T)
> print(plot.vgm.S, split = c(3, 1, 3, 1), more = F)
```



Remove the plotting objects and scale:

```
> rm(g.max, plot.vgm.all, plot.vgm.N, plot.vgm.S)
```

Q131 : *Do the two halves appear to have different local spatial structure?*
Jump to A131 •

Remove the variograms and models from the workspace:

```
> rm(v, v.n, v.s, vm, vmf, vmN, vmNf, vmS, vmSf)
```

We are also done with the field halves:

```
> rm(mhw.sp.ns)
```

Challenge: Repeat the analysis of this section with E-W field halves, instead of N-S field halves. Do you reach similar conclusions about the differences between the spatial structure of the field halves?

17.5 Answers

A122 : *The row and column numbers don't seem to show any pattern or trend.*
Return to Q122 •

A123 : *Although there are differences among rows and columns, there does not appear to be a trend. The higher-numbered columns (East half of the field) appear to be slightly lower (as a group) than the lower-numbered columns. There appears to be some short-range periodicity at a one-plot range (high followed by low) in both dimensions, although this is not regular.* Return to Q123 •

A124 : *Only about 9% of the variance is explained; only the x-coördinate (E-W, across the columns from low- to high-numbered) is significant; it shows a slight trend towards the East; this agrees with the by-column boxplot. This trend was also noted by Patankar [33].* Return to Q124 •

A125 : *There is evidence of spatial structure: the semivariance increases up to about 13 m separation; the semivariance then fluctuates around a total sill of about* $\gamma = 0.21 \text{lb}^2$ *There appears to be a "nugget" effect (semivariance at zero separation) of about* $\gamma = 0.05 \text{lb}^2$. Return to Q125 •

A126 : *Plot size is (3.18 m long x 2.55 m wide) (§A), so the range of about 13 m corresponds to about four adjacent plots column-wise and five adjacent plots row-wise.* Return to Q126 •

A127 : *Strictly speaking, this is not a valid assumption: we know from the description of the experimental protocol that the measurement precision was 0.01 lb (§A), so the minimum nugget should be 0.0001 lb². However, this is very close to zero, in relation to the total sill of about 0.20 lb². This reported precision assumes that all other operations were carried out perfectly: the plot was exactly delineated, all grain and straw in the plot was harvested, the air-drying brought all samples to the same moisture level, and the hand-threshing did not lose any grain. The zero fitted nugget suggests that the experimental protocol was very carefully carried out.* Return to Q127 •

A128 : *The elements of the permuted and original vectors are the same, all that changes is their sequence.* Return to Q128 •

A129 : *The randomizations have a similar pattern but different locations of high and low values; there is no apparent spatial pattern. The actual spatial distribution shows clear concentrations of high and low values ("hot" and "cold" spots) with a spatial dependence of about 4 plots.* Return to Q129 •

A130 : *The three simulations have very similar variograms: they all fluctuate around the sill (representing the total variability in the field), which is the same as for the variogram of actual data. This latter is quite different from the simulations, and shows clear spatial structure up to 15 m.*

The lack of spatial dependence in the "random assignment" fields is proven by the variograms: a pure nugget effect, where the nugget variance is the same as the sill. Return to Q130 •

A131 : *There is a big difference between the structures in the two halves. The S half is more variable overall (higher total sill, about* $\gamma = 0.22\text{lb}2^2$*), with a longer range around 20 m; the N half reaches a lower total sill (about* $\gamma = 0.19\text{lb}^2$*) at a shorter range, about 12 m. Both have a nugget effect, about* $\gamma = 0.075\text{lb}^2$ *in the S and* $\gamma = 0.011\text{lb}^2$ *in the N.* *Return to Q131* •

18 Generalized least squares regression*

In §8.2 we computed a relation of straw yield modelled from grain yield, which could then be applied to predict the straw yield of any plot, under similar conditions, with a measured grain yield:

```
> coef(model.straw.grain <- lm(straw ~ grain, data = mhw.sp))

(Intercept)       grain
    0.86628     1.43050
```

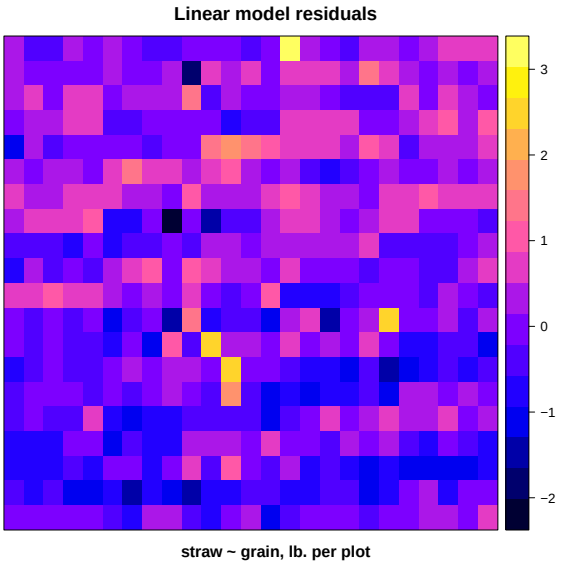
Clearly, however, this relation is not the same at each plot: that's why there are non-zero residuals from the linear regression. But is there a **spatial** relation here? That is, are the residuals from the regression **spatially correlated**? If they are, they violate one of the assumptions of ordinary least-squares (OLS) linear regression, namely, the independence of and identical distribution (*i.i.d*) of the residuals. In that case, the solution by OLS presented in §8.2.2 is not valid and must be modified.

So, we now determine whether there is any evidence of non-independence of the residuals due to spatial correlation. The first step is visualization; afterwards we will model the suspected spatial correlation.

Task 122 : Add the OLS model residuals to the spatial points data frame and show as a post-plot. •

Again, as in §16.5, we also make a pixel version for easier visualization.

```
> mhw.sp$msg.res <- residuals(model.straw.grain)
> mhw.sp.pix <- as(mhw.sp,"SpatialPixelsDataFrame")
> spplot(mhw.sp.pix, zcol="msg.res", col.regions=bpy.colors(64),
+        main="Linear model residuals",
+        sub="straw ~ grain, lb. per plot")
```



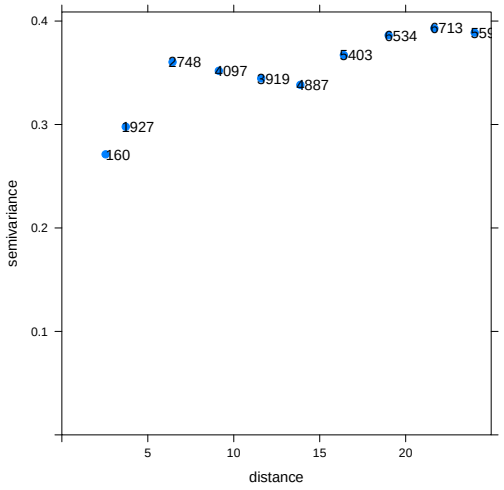
Q132 : *Does there appear to be spatial correlation among the residuals? At what scale?* *Jump to A132* •

The post-plot suggests that the residuals are spatially-correlated. In §17.2 we saw how to reveal spatial structure of a variable with a variogram; here we apply that method to the residuals, to see if they are spatially-correlated.

Task 123 : Compute and display their empirical variogram. •

As in §17.2 we compute the variogram to a radius of 10 plot widths, with bins of plot width size:

```
> vr <- variogram(msg.res ~ 1, loc = mhw.sp, cutoff = plot.wid *
+     10, width = plot.wid)
> plot(vr, pl = T, pch = 20, cex = 1.5)
```



Q133 :

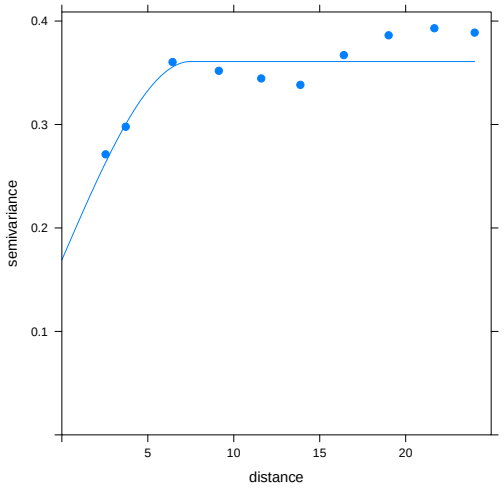
(1) How does the empirical variogram support the inference from the post-plot that there is spatial correlation among the residuals?

(2) Approximately how much of the residual variance is spatially-correlated at the shortest separation?

(3) What is the approximate range of spatial correlation? *Jump to A133* •

Task 124 : Model the variogram. •

```
> (vgmr <- fit.variogram(vr, model = vgm(0.15, "Sph", 20,
+     0.05)))

  model   psill  range
1   Nug 0.16902 0.0000
2   Sph 0.19183 7.4717

> plot(vr, model = vgmr, pch = 20, cex = 1.5)
```



Q134 : *Describe the spatial structure of the model residuals. Is this evidence of spatial correlation?* *Jump to A134* •

We conclude that the model residuals are *not* independent – they are spatially correlated. Although the regression coefficients computed in §8.2 are unbiased, the standard errors are too small and so the significance is overestimated, since there are effectively fewer degrees of freedom (plots partially duplicate each other's information). Further, the coefficients may not be optimal.

The key difference here is that in the linear model, the residuals ε are *independently* and *identically* distributed with the same variance σ^2:

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon,\ \varepsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I}) \tag{18.1}$$

Whereas, now the residuals are considered themselves a random variable η that has a covariance structure:

$$\mathbf{y} = \mathbf{X}\beta + \eta,\ \eta \sim \mathcal{N}(0, \mathbf{V}) \tag{18.2}$$

where $\mathbf{V}$ is a positive-definite variance-covariance matrix of the model residuals.

Continuing with the derivation from §8.2.2, Lark and Cullis [24, Appendix] point out that the issue here is that the error vectors can now not be assumed to be spherically distributed around the 0 expected value, but rather that error vectors in some directions are longer than in others. So, the measure of distance (the vector norm) is now a so-called "generalized" distance[15], taking into account the covariance between error vectors:

$$S = (\mathbf{y} - \mathbf{X}\beta)^T\mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\beta) \tag{18.3}$$

Comparing this to the OLS equivalent (Equation 8.4), we see here the variance-covariance matrix of the residuals $\mathbf{V} = \sigma^2 C$, where σ^2 is the variance of the residuals and $\mathbf{C}$ is the correlation matrix. This reduces to the OLS formulation of Equation 8.4 if there is no covariance, i.e., $\mathbf{V} = \mathbf{I}$.

[15] This is closely related to the Mahalanobis distance

Expanding Equation 18.3, taking the partial derivative with respect to the parameters, setting equal to zero and solving we obtain:

$$\begin{aligned} \frac{\partial}{\partial \beta} S &= -2\mathbf{X}^T\mathbf{V}^{-1}\mathbf{y} + 2\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X}\beta \\ 0 &= -\mathbf{X}^T\mathbf{V}^{-1}\mathbf{y} + \mathbf{X}^T\mathbf{V}^{-1}\mathbf{X}\beta \\ \hat{\beta}_{\text{GLS}} &= (\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{V}^{-1}\mathbf{y} \end{aligned} \tag{18.4}$$

This reduces to the OLS estimate $\hat{\beta}_{\text{OLS}}$ of Equation 8.6 if there is no covariance, i.e., $\mathbf{V} = \mathbf{I}$.

In the case of spatial correlation, we ensure positive-definiteness (i.e., always a real-valued solution) by using an authorized covariance function C and assuming that the entries are completely determined by the vector distance between points $\mathbf{x}_i - \mathbf{x}_j$:

$$\mathbf{C}_{i,j} = C(\mathbf{x}_i - \mathbf{x}_j) \tag{18.5}$$

In this formulation C has a three-parameter vector θ, as does the corresponding variogram model: the range a, the total sill σ^2, and the proportion of total sill due to pure error, not spatial correlation s[16].

In modelling terminology, the coefficients β are called **fixed** effects, because their effect on the response variable is fixed once the parameters are known. By contrast the covariance parameters η are called **random** effects, because their effect on the response variable is stochastic, depending on a random variable with these parameters.

Models with the form of Equation 18.2 are called **mixed** models: some effects are fixed (here, the relation between the straw and grain yields) and others are random (here, the error variances) but follow a known structure; these models have many applications and are extensively discussed in Pinheiro and Bates [37]. Here the random effect η represents both the spatial structure of the residuals from the fixed-effects model, and the unexplainable (short-range) noise. This latter corresponds to the noise σ^2 of the linear model of Equation 18.1.

Q135 : *If $s = 1$, what does this imply?* *Jump to A135* •

To solve Equation 18.4 we first need to compute $\mathbf{V}$, i.e., estimate the variance parameters $\theta = [\sigma^2, s, a]$, use these to compute $\mathbf{C}$ with equation 18.5 and from this $\mathbf{V}$, after which we can use equation 18.4 to estimate the fixed effects β. But θ is estimated from the residuals of the fixed-effects regression, which has not yet been computed. How can this "chicken-and-egg"[17] computation be solved?

The answer is to use **residual** (sometimes called "restricted") **maximum likelihood** (REML) to maximize the likelihood of the random effects θ independently of the fixed effects β. We continue with this in §18.2 below.

[16] In variogram terms, this is the nugget variance c_0 as a proportion of the total sill $(c_0 + c_1)$.

[17] from the question "which came first,the chicken or the egg?"

18.1 A detour into Maximum Likelihood*

To understand REML, we first explain the basics of ML, the **Maximum Likelihood** estimation, which is the derivation of the most "likely" values of model parameters, consistent with a set of observations.

There are three steps:

1. Specify a model with parameters.

   For example, the linear model with i.i.d. residuals of Equation 18.1: $\mathcal{Y} = \mathbf{X}\beta + \varepsilon,\ \varepsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$. In the single-predictor case the parameter vector is $\beta = (\beta_0, \beta_1)$, the intercept and slope of the linear model.

   All model forms are **assumptions**: we fit parameters given observations, but the model form is set by us. For this linear model, we are assuming that the observed values are the result of a linear deterministic process with coefficients β, with stochastic errors with zero mean and a given variance, uncorrelated with each other. If this assumption is not valid, the rest of the analysis is invalid.

   > **Note:** We can specify several models, fit their parameters, and compare their likelihoods.

   We can rewrite the single-predictor linear model as the residuals expressed as the fits $\varepsilon = \mathbf{y} - \mathbf{X}\beta$. That is, for any chosen regression parameters β, these are the errors $\varepsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$. Clearly, the we want to minimize the errors; we express this as the squared error $(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)$ to give equal weight to the positive and negative residuals.

   Now, the **probability** of observing a specific response $\mathcal{Y}_i$ from the random variable Y, once β is fixed, is given by the normal probability of the associated residual, which we assume is normally-distributed with mean 0 and standard deviation σ:

   $$\Pr(Y = \mathcal{Y}_i | \beta, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(\mathcal{Y}_i - \mathbf{X_i}\beta)^T(\mathcal{Y}_i - \mathbf{X_i}\beta)} \tag{18.6}$$

   This probability depends on the residual $\varepsilon_i = (\mathcal{Y}_i - \mathbf{X_i}\beta)$, which can be directly calculated from the chosen regression parameters β, but also on the standard deviation of the errors σ, which must also be estimated from the observations.

   The matrix product in the final term can be written as the squared residual:

   $$(\mathcal{Y}_i - \mathbf{X_i}\beta)^T(\mathcal{Y}_i - \mathbf{X_i}\beta) = (\mathcal{Y}_i - \beta_0 - \beta_1 x_i)^2 = \varepsilon_i^2 \tag{18.7}$$

   so Equation 18.6 can be written as:

   $$\Pr(Y = \mathcal{Y}_i | \beta, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}\varepsilon_i^2} \tag{18.8}$$

This probability can be computed with the `dnorm` function; here the mean residual is 0.

Task 125 : For a fixed (assumed) value of the standard deviation, say the default 1, compute some probabilities of observed residuals. •

```
> dnorm(x = c(-1.96, -1, -0.5, 0, 0.5, 1, 1.96), mean = 0,
+     sd = 1)

[1] 0.058441 0.241971 0.352065 0.398942 0.352065 0.241971 0.058441
```

Task 126 : For a fixed (assumed) observed residual, say 1, compute the probability of observing that value, for several assumed values of the population standard deviation σ. •

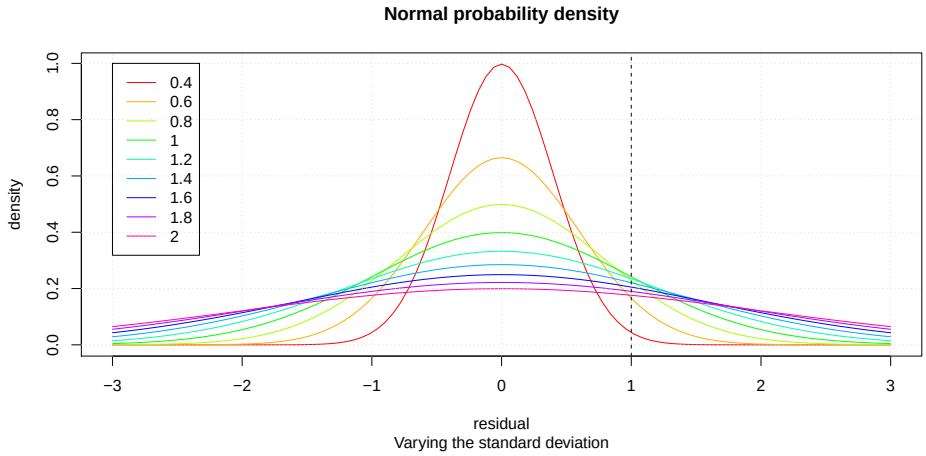
```
> s <- seq(0.4, 2, by = 0.2)
> data.frame(sd = s, p = dnorm(x = 1, mean = 0, sd = s))

   sd        p
1 0.4 0.043821
2 0.6 0.165795
3 0.8 0.228311
4 1.0 0.241971
5 1.2 0.234927
6 1.4 0.220797
7 1.6 0.205101
8 1.8 0.189940
9 2.0 0.176033
```

The probability of observing this fixed value of the residual (i.e., deviation from zero) increases until standard deviation 1 and then decreases, as the normal "bell-shaped" curve becomes flatter.

Task 127 : Visualize the normal curves for the different standard deviations, and show the probability of the selected residual for each. •

We can visualize this by applying the `curve` plotting function to values calculated with `dnorm`, with different standard deviations:

```
> tmp <- rainbow(length(s))
> curve(dnorm(x, mean=0, s[1]), -3, 3, col=tmp[1],
+       main="Normal probability density",
+       sub="Varying the standard deviation",
+       ylab="density",xlab="residual")
> for (i in 2:length(s))
+      curve(dnorm(x, mean=0, sd=s[i]), -3, 3,
+            col=tmp[i], add=T)
> grid()
> abline(v=1, lty=2)
> legend(-3, 1, s, lty=1, col=tmp)
```



2. Write an equation to compute the **likelihood** of observing the known values, given specific values of the parameters. This is the same conditional probability, but this time considering the observations $\mathbf{y}$ as fixed and the β as unknowns, to be solved for.

   In the case where the observations are independent, the likelihood of the set of observations $\mathbf{y}$, given fixed β, is defined as the product of their individual probabilities from Equation 18.6. This is because independent observations implies independent residual errors, so the joint probability of observing the actual vector $\mathbf{y}$ is the product of the probability of observing each one:

$$\begin{aligned} \mathcal{L}(\beta, \sigma^2|\mathbf{y}) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\varepsilon_i^2}{2\sigma^2}} \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} \prod_{i=1}^{n} e^{-\frac{\varepsilon_i^2}{2\sigma^2}} \end{aligned} \tag{18.9}$$

   This is now a function of the model parameters; as we vary these, the regression residuals ε change, so the the likelihood changes; when it is a maximum, these are the most probable values of the parameters. The variance σ^2 also can vary, and also affects the likelihood.

   The likelihood function is usually written as a logarithm to allow easy differentiation; recall that $\log(ab) = \log(a) + \log(b)$ and $\log(a^c) = c\log(a)$, so that the product becomes a sum and an exponentiation becomes a product.

   > **Note:** The choice of parameters that maximizes the log-likelihood also maximizes the likelihood, because the logarithm is a monotonically increasing function.

Taking logarithms of both sides of Equation 18.9 we obtain:

$$\begin{aligned} \ell(\beta, \sigma^2|\mathbf{y}) &= \log\left(\frac{1}{(2\pi\sigma^2)^{n/2}}\right) - \sum_{i=1}^{n} \frac{\varepsilon_i^2}{2\sigma^2} \\ &= -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\varepsilon_i^2 \end{aligned} \tag{18.10}$$

The first term depends only on the sample size; the second term depends on the sample size and the population variance; and the third term depends on the population variance and the regression coefficients (because they are used to compute ε_i).

3. Solve for the maximum value of the log-likelihood, either analytically or numerically.

   Since this is an optimization problem, the obvious way to solve analytically is to differentiate Equation 18.10 with respect to each parameter and set to zero; then check if this is a maximum by the second derivative.

   > **Note:** Many likelihood functions can be solved analytically, in which case numerical optimization methods must be used, systematically varying the parameters, computing the likelihood, and looking for a maximum..

   Here we have three partial derivatives, with respect to each of the three parameters to be estimated:

$$\frac{\partial \ell}{\partial \beta_0} = \frac{1}{\sigma^2}\sum_i (y_i - \beta_0 - \beta_1 x_i) \tag{18.11}$$

$$\frac{\partial \ell}{\partial \beta_1} = \frac{x_i}{\sigma^2}\sum_i x_i (y_i - \beta_0 - \beta_1 x_i) \tag{18.12}$$

$$\frac{\partial \ell}{\partial \sigma^2} = -\frac{1}{\sigma^2} + \frac{1}{n}\sum_i (y_i - \beta_0 - \beta_1 x_i)^2 \tag{18.13}$$

   Setting these equal to zero and simplifying:

$$\sum_i (y_i - \beta_0 - \beta_1 x_i) = 0 \tag{18.14}$$

$$\sum_i x_i (y_i - \beta_0 - \beta_1 x_i) = 0 \tag{18.15}$$

$$\sum_i (y_i - \beta_0 - \beta_1 x_i)^2 = n\sigma^2 \tag{18.16}$$

   Solving these three equations, we obtain the familiar least-squares estimators for the slope β_1 and the intercept β_0:

$$\hat{\beta}_{1,ML} = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} = \frac{S_{XY}}{S_{XX}} \tag{18.17}$$

$$\hat{\beta}_{0,ML} = \bar{y} - \hat{\beta}_{1,ML}\bar{x} \tag{18.18}$$

The most likely variance of the residuals is:

$$\hat{\sigma}^2_{ML} = \frac{1}{n}\sum_i e_i^2 = \frac{1}{n}\sum_i (y_i - \beta_0 - \beta_1 x_i)^2 \tag{18.19}$$

18.1.1 Numerical solution

In this case the solution can be found analytically; in general a numerical solution is required. For illustration we show a brute-force approach, trying various values of the three parameters and then finding the maximum log-likelihood. In practical computations, gradient methods are used, e.g., a multivariate version of Newton-Raphson root finding. In complicated problems the likelihood surface may not be convex and there is a danger of finding only a local maximum.

Task 128 : Write a function to compute log-likelihood for simple linear regression according to Equation 18.10; its arguments should be the parameter values and the observations (both predictor and predictand). •

```
> like <- function(beta0, beta1, sigma, x, y) {
+     s2 <- sigma^2
+     n <- length(y)
+     pred <- beta0 + beta1 * x
+     loglike <- -(n/2) * (log(2 * pi)) - (n/2) * (log(s2)) -
+         (1/(2 * s2)) * (sum((y - pred)^2))
+     return(loglike)
+ }
```

Task 129 : Set up arrays of possible parameter values to be tested for their likelihood. •

These should be based on guesses from the scatterplot or previous information; here we use the results of the linear model fit and vary them by ±10%:

```
> coefficients(lm(straw ~ grain, data = mhw))

(Intercept)       grain
    0.86628     1.43050

> summary(lm(straw ~ grain, data = mhw))$sigma

[1] 0.61477
```

We create the three vectors and then all possible combinations, using the `expand.grid` function, and add a placeholder column for the computed log-likelihood:

```
> coef <- round(coefficients(lm(straw ~ grain, data = mhw)),
+     5)
> (beta0 <- coef[1] * seq(0.9, 1.1, by = 0.02))

 [1] 0.77965 0.79698 0.81430 0.83163 0.84895 0.86628 0.88361 0.90093
 [9] 0.91826 0.93558 0.95291

> (beta1 <- coef[2] * seq(0.9, 1.1, by = 0.02))

 [1] 1.2875 1.3161 1.3447 1.3733 1.4019 1.4305 1.4591 1.4877 1.5163
[10] 1.5449 1.5736

> (sigma <- round(summary(lm(straw ~ grain, data = mhw))$sigma,
+     5) * seq(0.9, 1.1, by = 0.02))

 [1] 0.55329 0.56559 0.57788 0.59018 0.60247 0.61477 0.62707 0.63936
 [9] 0.65166 0.66395 0.67625

> beta <- expand.grid(beta0 = beta0, beta1 = beta1, sigma = sigma)
> beta$loglik <- 0
> dim(beta)

[1] 1331    4

> rm(coef, beta0, beta1, sigma)
```

Task 130 : Compute the log-likelihood for each combination of parameters, given the observed wheat yield data, and find the maximum. •

This is a "brute-force" computation of all combinations, using a `for` loop and then finding the maximum with the `which.max` function:

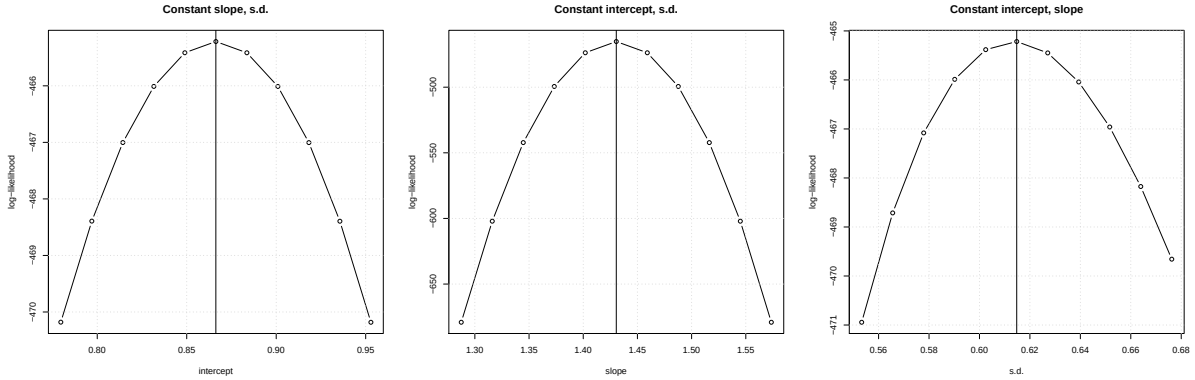
```
> for (i in 1:length(beta$loglik)) beta$loglik[i] <- like(beta[i,
+     "beta0"], beta[i, "beta1"], beta[i, "sigma"], mhw$grain,
+     mhw$straw)
> summary(beta$loglik)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   -821    -614    -530    -556    -484    -465

> (beta.mle <- beta[which.max(beta$loglik), ])

      beta0  beta1   sigma  loglik
666 0.86628 1.4305 0.61477 -465.22
```

As expected, these optimal parameter values are identical to those computed by `lm` using least squares.

Task 131 : Visualize the likelihood surface, first 1D for each parameter separately and then 2D for a combination of two parameters. •

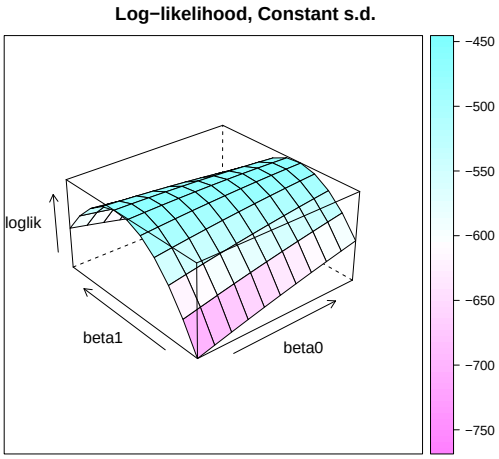
First, the three per-parameter one-dimensional plots, holding the other two parameters constant:

```
> par(mfrow = c(1, 3))
> tmp <- beta[(beta$sigma == beta.mle$sigma), ]
> tmp <- tmp[(tmp$beta1 == beta.mle$beta1), ]
> plot(tmp$loglik ~ tmp$beta0, type = "b", xlab = "intercept",
+     ylab = "log-likelihood", main = "Constant slope, s.d.")
> grid()
> abline(v = beta.mle$beta0)
> tmp <- beta[(beta$sigma == beta.mle$sigma), ]
> tmp <- tmp[(tmp$beta0 == beta.mle$beta0), ]
> plot(tmp$loglik ~ tmp$beta1, type = "b", xlab = "slope",
+     ylab = "log-likelihood", main = "Constant intercept, s.d.")
> grid()
> abline(v = beta.mle$beta1)
> tmp <- beta[(beta$beta1 == beta.mle$beta1), ]
> tmp <- tmp[(tmp$beta0 == beta.mle$beta0), ]
> plot(tmp$loglik ~ tmp$sigma, type = "b", xlab = "s.d.",
+     ylab = "log-likelihood", main = "Constant intercept, slope")
> grid()
> abline(v = beta.mle$sigma)
> par(mfrow = c(1, 3))
```



Second, a two-dimensional surface, holding one parameter constant, using the `wireframe` function of the `lattice` package:

```
> tmp <- beta[(beta$sigma == beta.mle$sigma), ]
> wireframe(loglik ~ beta0 + beta1, data = tmp, aspect = c(1,
+     0.5), drape = T, main = "Log-likelihood, Constant s.d.")
```




```
> rm(like, beta, i, beta.mle, tmp)
```

18.2 Residual Maximum Likelihood

We first complete the theoretical derivation of REML (§18.2.1), and then show how to compute the GLS relation using this method (§18.2.2).

18.2.1 REML – theory

Returning now to Equation 18.2, where we can not assume i.i.d. residuals as in Equation 18.1, Lark and Cullis [24, Eq. 12] show that the likelihood of the parameters is now expanded to include the spatial dependence implicit in the variance-covariance matrix $\mathbf{V}$, rather than a single residual variance σ^2. The log-likelihood is then:

$$\ell(\beta, \theta|\mathbf{y}) = c - \frac{1}{2}\log|\mathbf{V}| - \frac{1}{2}(\mathbf{y} - \mathbf{X}\beta)^T\mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\beta) \tag{18.20}$$

where c is a constant (and so does not vary with the parameters) and $\mathbf{V}$ is built from the variance parameters θ and the distances between the observations. By assuming second-order stationarity[18], the structure can be summarized by the covariance parameters $\theta = [\sigma^2, s, a]$, i.e., the total sill, nugget proportion, and range.

However, maximizing this likelihood for the random-effects covariance parameters θ also requires maximizing in terms of the fixed-effects regression parameters β, which in this context are called *nuisance parameters* since at this point we don't care about their values; we will compute them after determining the covariance structure.

Both the covariance and the nuisance parameters β must be estimated, it seems at the same time ("chicken and egg" problem) but in fact the technique of REML can be used to first estimate θ without having to know the nuisance parameters. Then we can use these to compute $\mathbf{C}$ with equation 18.5 and

[18] that is, the covariance structure is the same over the entire field, and only depends on the distance between pairs of points

from this $\mathbf{V}$, after which we can use equation 18.4 to estimate the fixed effects β.

The maximum likelihood estimate of θ is thus called "restricted", because it only estimates the covariance parameters (random effects). Conceptually, REML estimation of the covariance parameters θ is ML estimation of both these and the nuisance parameters β, with the later integrated out [37, §2.2.5]:

$$\ell(\theta|\mathbf{y}) = \int \ell(\beta, \theta|\mathbf{y}) \, d\beta \tag{18.21}$$

Pinheiro and Bates [37, §2.2.5] show how this is achieved, given a likelihood function, by a change of variable to a statistic sufficient for β.

Lark and Cullis [24, Eq. 15], following Smyth and Verbyla [42], show that the log-likelihood of θ is then conditional on the sufficient statistic $\mathbf{t}$ for β:

$$\begin{aligned} \ell(\theta|\mathbf{t}) &= c - \frac{1}{2}\log|\mathbf{V}| - \frac{1}{2}\log|\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X}| \\ &\quad -\frac{1}{2}\mathbf{y}^T\mathbf{V}^{-1}(\mathbf{I}-\mathbf{Q})\mathbf{y} \\ \text{where } \mathbf{Q} &= \mathbf{X}(\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{V}^{-1} \end{aligned} \tag{18.22}$$

Since the nuisance parameters β are not present in Equation 18.22, the likelihood of the covariance parameters θ, which determine the variance-covariance matrix $\mathbf{V}$, can be maximized independently of the nuisance (regression) parameters.

18.2.2 REML – computation

Equation 18.22 must be maximized by varying the three parameters in θ which determine $\mathbf{V}$, there is no analytic solution. This is is a non-linear optimization problem over a large parameter space, and there is no guarantee of finding the true optimum. This is in contrast to ordinary least squares, which is a direct computation from the model matrix and observation values. This problem can be partially addressed by starting the solution with reasonable covariance parameters, for example, those inferred from a variogram fit to the OLS residuals. But even this is no guarantee; Lark and Cullis [24] used simulated annealing, which can escape local minima. We use a gradient method, a multivariate version of Newton-Raphson minimization.

In the R environment REML by the gradient method is implemented with the `gls` "Generalized Least Squares" function of the `nlme` "Linear and Nonlinear Mixed Effects Models" package, based on the text by Pinheiro and Bates [37]. See Bates [2] for a simple introduction to the concepts, and how to use them in the R environment.

Task 132 : Load the `nlme` package and examine the help for the `gls` function. •

```
> require(nlme)
> help(gls)
```

As with the `lm` function, this requires a formula (here called a 'model') and a dataframe in which to find variables. In addition, it requires either a correlation structure or a weights matrix. In our case we have a known spatial correlation, so we need to specify how the residuals may be correlated.

The `nlme` package provides a number of constructors of correlation structures for both time and spatial correlation. The spatial correlation models are similar to familiar variogram models: exponential, Gaussian, and Spherical (the structure we used to model the residuals, above). All that is required is a starting value for the range and nugget, which we extract from the fitted variogram model and put together in a list, and the formula for spatial coordinates. These constructors are functions named `corSpher`, `corExp` and so forth; see `?corClasses` for details.

A small complication is that `nlme` does not depend on the `sp` structures, so we need to convert to a dataframe with the `as.data.frame` function, so that the coordinate names `x` and `y` become visible.

Task 133 : Fit the regression model by GLS, using a spherical correlation structure based on the variogram analysis. •

The model fitting takes a bit of time; we can see how much by enclosing the call to `gls` in the `system.time` function:

```
> system.time(
+             model.gls.straw.grain <-
+             gls(model=straw ~ grain,
+                 data=as.data.frame(mhw.sp),
+                 correlation=corSpher(
+                   value=c(vgmr[2,"range"],vgmr[1,"psill"]),
+                   form=~x+y, nugget=T))
+             )

   user  system elapsed
 11.389   0.936  13.158
```

Task 134 : Summarize the model. •

```
> summary(model.gls.straw.grain)

Generalized least squares fit by REML
  Model: straw ~ grain
  Data: as.data.frame(mhw.sp)
     AIC    BIC  logLik
  853.54 874.59 -421.77

Correlation Structure: Spherical spatial correlation
 Formula: ~x + y
 Parameter estimate(s):
  range  nugget
8.02312 0.37426
```

```
Coefficients:
             Value Std.Error t-value p-value
(Intercept) 1.5605  0.240165  6.4977       0
grain       1.2557  0.059545 21.0879       0

 Correlation:
      (Intr)
grain -0.978

Standardized residuals:
      Min        Q1       Med        Q3       Max
-3.114035 -0.637046 -0.010328  0.619166  4.790892

Residual standard error: 0.61467
Degrees of freedom: 500 total; 498 residual
```

There is quite a bit of information in the model summary:

1. the fixed-effects model form;
2. the fitting information: Akaike's Information Criterion (AIC), Bayes' Information Criterion (BIC) and the log-likelihood of the final REML fit;
3. the random-effects model form, i.e., the correlation structure, with estimates of its parameters (here, range and proportional nugget);
4. the fitted model coefficients for the fixed effects, their standard errors and significance;
5. the correlation between parameters;
6. the residuals and residual standard error (i.e., lack of fit).

Q136 :

(1) How do the GLS linear regression coefficients compare with those estimated by OLS?;

(2) What parameters for the specified spatial correlation structure were fit by REML?

(3) How do these spatial structure parameters compare with the fitted variogram model? *Jump to A136*

•

```
> coefficients(model.gls.straw.grain)

(Intercept)       grain
     1.5605      1.2557

> coefficients(model.straw.grain)

(Intercept)       grain
    0.86628     1.43050
```

We now have two models to compare; which is better? One way to answer this is to compute each model's log-likelihood with the `logLik` "log-likelihood" function, which can be applied to most models; but this is not corrected for degrees of freedom. For that, we use the `AIC` "Akaike's An Information Criterion" function; this is defined as:

$$\text{AIC} = -2\ \log(\text{likelihood}) + 2\ p \tag{18.23}$$

where p is the number of model parameters. Thus the AIC penalizes models with many parameters, similarly to the adjusted R^2 for linear models. Because of the change in sign, the lower AIC is better.

```
> logLik(model.gls.straw.grain)

'log Lik.' -421.77 (df=5)

> logLik(model.straw.grain)

'log Lik.' -465.21 (df=3)

> AIC(model.gls.straw.grain)

[1] 853.54

> AIC(model.straw.grain)

[1] 936.43
```

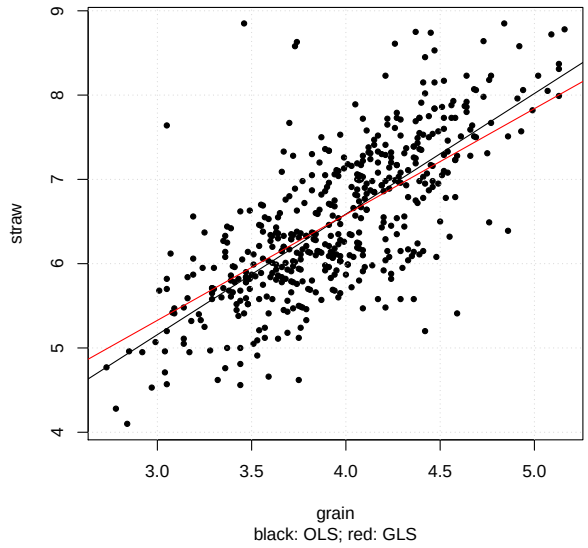
This comparison can only be applied to hierarchical models, that is, where one is an extension of the other. In this case the GLS model has the same form for the regression parameters but an extended form for the residual structure, so they can be compared.

Q137 : *Is the GLS model more likely than the OLS model? Which model has more parameters? Which model is better according to the AIC?* *Jump to A137* •

Finally, we visualize the effect on the regression model fit of using GLS instead of OLS.

Task 135 : Plot the straw vs. grain yield, with the OLS and GLS regression lines. •

```
> plot(straw ~ grain, data=mhw, pch=20,
+      sub="black: OLS; red: GLS")
> grid()
> abline(model.straw.grain)
> abline(model.gls.straw.grain, col="red")
```



Q138 : *What may account for the shallower slope of the GLS line?* *Jump to A138* •

Challenge: Recompute the GLS regression, but using an exponential variogram model form – this is often used for these sorts of problems, because it has a simple structure and interpretation.

You will first have to re-fit the residual variogram with an exponential model to obtain starting values for `gls`.

Compare the GLS regression line with that obtained with the spherical model; compare the fitted correlation structure with that from the spherical model. Plot all three lines on the scatterplot and comment on the differences. How much difference did the choice of correlation structure make in the estimates?

Compare the log-likelihoods of the two GLS models (exponential and spherical variogram forms); which is more likely?

> **Note:** Recall that the reported range parameter of an exponential model is one-third of its effective range, i.e., when the correlation is reduced to 5% of the total.

18.3 Answers

A132 : *Yes, there are definitely many high (orange to yellow) and low (blue) patches of a few adjacent plots.* *Return to Q132* •

A133 : *(1) Yes, the variance is less at closer separations; (2) about* (0.35 –

0.2)/0.35 ≈ 40% *of the total sill appears to be structural (not nugget variance); (3) the range is about 6 to 7 m, i.e., about two to three plots.* *Return to Q133* •

A134 : *There is a fairly high nugget (theoretically, the same as for the residuals of straw yield) but clear spatial structure to about 6 meters (a bit more than two plot widths) and then some fluctuation around a sill, suggestive of periodicity; see §20.* *Return to Q134* •

A135 : *A proportion of pure noise* $s = 1$ *means that the nugget* $c_0 = \sigma^2$*, i.e., is equal to the total sill* ($c_0 + c_1$ *in variogram terms), so the residuals have no spatial correlation and the OLS solution is valid; there is no need to account for correlation in the residuals by GLS.* *Return to Q135* •

A136 :

(1) The intercept is higher (1.56 vs. 0.87) and the slope is shallower (1.26 vs. 1.43).

(2) A range of just over 8 m and a nugget of about 0.37*.*

(3) The range is quite comparable to the variogram model fit, just a little longer; but the nugget has a different meaning. Here it is the proportion of the total sill, which for the variogram fit is 0.4684; the REML fit decreases this a bit.

Return to Q136 •

A137 : *The GLS model is substantially more likely than the OLS model; its log-likelihood is -421.8 compared to -465.2 for the OLS model. The GLS model has two more parameters, because three parameters are needed for the covariance structure, but only one for the variance assuming i.i.d. residuals. Still, the AIC for the GLS model, 853.5, is much superior to that for the OLS model, 936.4* *Return to Q137* •

A138 : *If the very high- and very low-yielding plots, with the highest leverage on the regression line, are spatially-correlated (which seems likely), the clusters in the scatterplot at extreme upper-right (above the average line) and extreme lower-left (below) will have less effect; because of spatial correlation they will be effectively clustered by the GLS fit, to some degree sharing their weights. So the line will not be pulled so much by these high-leverage plots.* *Return to Q138* •

19 Geographically-weighted regression*

Another approach to spatial correlation in regression is to use it to reveal an underlying process. In §8.2 and §18 we implicitly assumed that the process by which straw and grain yields are related (i.e., the plant physiology in response to environmental factors) is the same everywhere. The environmental factors could result in higher or lower yields (linear model intercept), but the relation of straw to grain is constant (linear model slope). But is this true?

A recently-developed technique for investigating this is "geographically weighted regression" (GWR). This computes the coefficients of the regression equa-

tion for each plot, in a local window which moves across a spatial field. Further, it weights plots near to the target plot more than those further away, using a density kernel. Thus the equation reported at each point can be thought of as the local linear relation between the response and predictor, which is now allowed to vary across the field. The main interpretive interest in GWR is the spatial pattern of the coefficients, which is taken as evidence of a **spatially-varying process**.

GWR is comprehensively described by Fotheringham et al. [15] and briefly by Fotheringham et al. [14, §5.4.3]; these same authors maintain a web page[19] with tutorial material.

One implementation of GWR in the R environment is the `spgwr` package[20], which builds on the `sp` package loaded in §16.3.

Task 136 : Load the `spgwr` package. •

```
> library(spgwr)
```

A key issue in GWR is the size and shape of the window to be moved over the field, and centered on each point in turn. This is closely related to kernel density estimation of a single variable (not a regression relation). The most common kernel shape is Gaussian (bivariate normal, the familiar bell-curve), and the bandwidth is often chosen to minimize the average cross-validation error of all the points predicted by their own local regression.

Task 137 : Compute the optimal bandwidth for GWR of the straw vs. grain relation. •

The `gwr.sel` function does this:

```
> (bw <- gwr.sel(straw ~ grain, data = mhw.sp, adapt = F,
+     verbose = F))

[1] 7.7937
```

Q139 : *To the range of the variogram of which variable should this bandwidth correspond?* *Jump to A139*
•

Task 138 : Compute and model the empirical variogram of the grain/straw ratio. •

```
> (vmf.gsr <- fit.variogram(v.gsr <- variogram(gsr ~ 1, loc=mhw.sp),
+                           model=vgm(0.004, "Sph", 10, 0.002)))
```

[19] http://ncg.nuim.ie/ncg/GWR/whatis.htm

[20] Although the package loads with the somewhat of-putting disclaimer "NOTE: This package does not constitute approval of GWR as a method of spatial analysis"!

```
  model     psill  range
1   Nug 0.0018000 0.0000
2   Sph 0.0016304 8.4351
```

Q140 : *Does the effective range of the fitted variogram model of the grain/straw ratio match the fitted bandwidth for GWR?* *Jump to A140* •

Task 139 : Compute the GWR for the straw vs. grain regression. •

The `gwr.sg` function computes this:

```
> (gwr.sg <- gwr(straw ~ grain, data = mhw.sp, bandwidth = bw))

Call:
gwr(formula = straw ~ grain, data = mhw.sp, bandwidth = bw)
Kernel function: gwr.Gauss
Fixed bandwidth: 7.7937
Summary of GWR coefficient estimates:
               Min. 1st Qu.  Median 3rd Qu.    Max. Global
X.Intercept. -0.0923  1.1600  1.6200  2.0600  3.4400   0.87
grain         0.8950  1.1000  1.2500  1.4000  1.7100   1.43
```

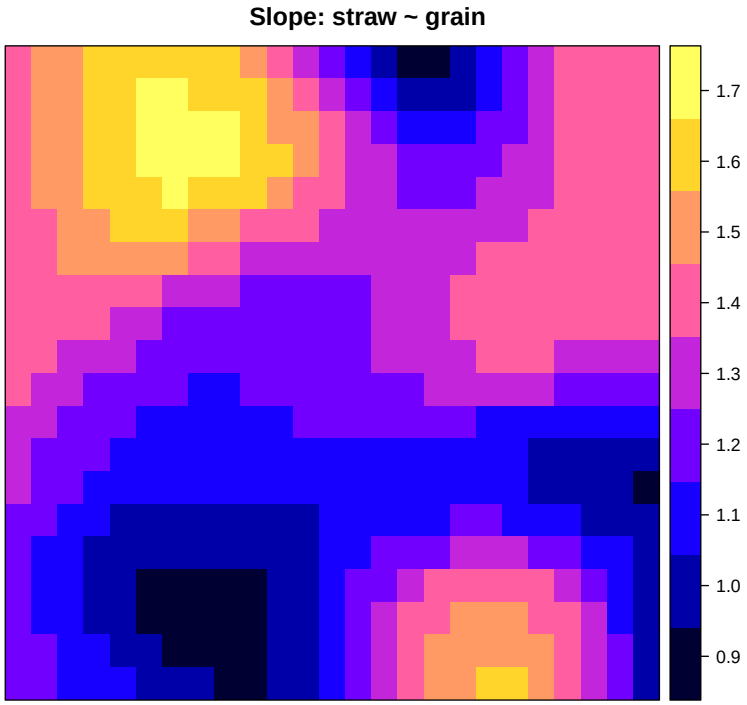
Q141 : *How much do the local slopes vary? How do they compare with the slope computed for the relation over the whole field?* *Jump to A141* •

Task 140 : Plot the GWR slopes (coefficient of straw vs. grain) across the field. •

The `SDF` field of the fitted object contains the fitted coefficients, the prediction from the local model, the residuals, and the local goodness-of-fit. The `grain` field within the `SDF` field is the slope coefficient.

For easier visualization, we convert the points representing plots into pixels.

```
> gwr.coef <- as(gwr.sg$SDF,"SpatialPixelsDataFrame")
> print(spplot(gwr.coef, zcol="grain",
+              col.regions=bpy.colors(64),
+              key.space="right", cuts=8,
+              main="Slope: straw ~ grain"))
```

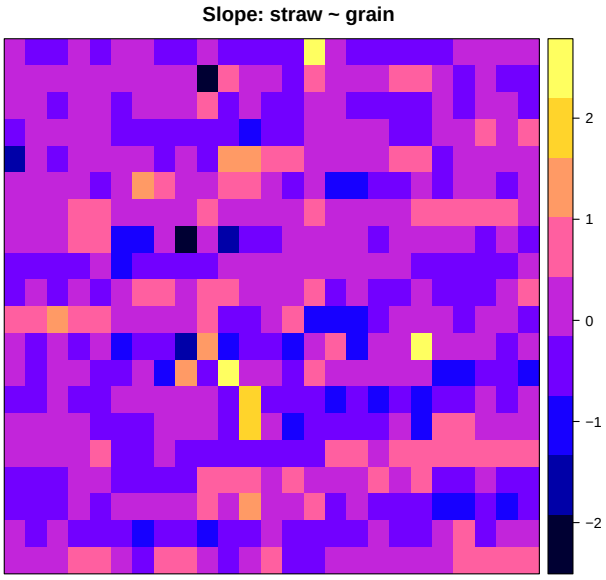


Q142 : *Does the relation of straw vs. grain appear to vary across the field? What is the interpretation?* *Jump to A142* •

Task 141 : Plot the GWR residuals and investigate if they have any spatial structure. •

The `gwr.e` field within the `SDF` field contains the residuals.

```
> print(spplot(gwr.coef, zcol="gwr.e",
+              col.regions=bpy.colors(64),
+              key.space="right", cuts=8,
+              main="Slope: straw ~ grain"))
```



To check for spatial structure, we compute the empirical variogram of these GWR residuals, and model it. A technical point: the `variogram` method can not handle non-square grid cells, so we have to convert to spatial points.

```
> vr.gwr <- variogram(gwr.e ~ 1,
+                     loc=as(gwr.coef, "SpatialPointsDataFrame"))
> (vmf.r.gwr <- fit.variogram(vr.gwr,
+                             model=vgm(0.1, "Sph", 5, 0.2)))

  model   psill  range
1   Nug 0.20118 0.0000
2   Sph 0.11740 8.5787
```

Q143 : *Is there any spatial structure in the GWR residuals?* *Jump to A143* •

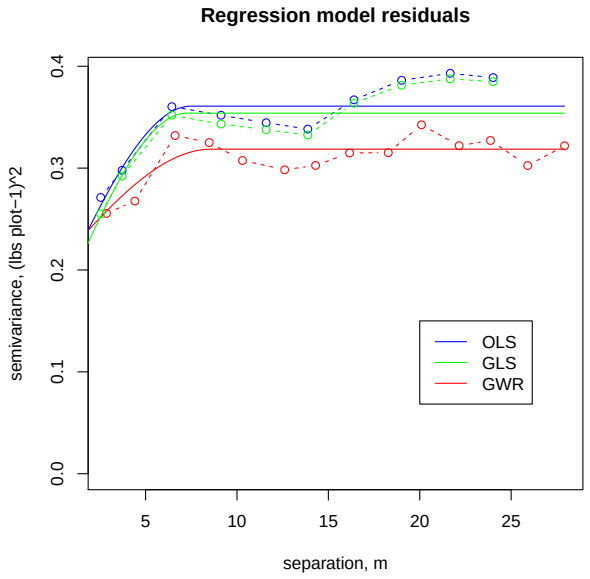
Task 142 : Plot the empirical variogram and fitted variogram model for these residuals, along with those from the OLS global fit (§8.2.1) and the REML fit (§18) to the entire dataset. •

We first must compute and model the variogram for the REML fit:

```
> mhw.sp$gls.res <- residuals(model.gls.straw.grain)
> vr.gls <- variogram(gls.res ~ 1, loc = mhw.sp, cutoff = plot.wid *
+     10, width = plot.wid)
> (vmf.r.gls <- fit.variogram(vr.gls, model = vgm(0.1,
+     "Sph", 5, 0.2)))

  model   psill  range
1   Nug 0.14959 0.0000
2   Sph 0.20438 7.2492
```

To put these on one plot it's easiest to use the base graphics `plot` method to establish the plotting axes and show one of the point sets; we then use the `lines` and `points` functions to add more point sets and lines. The lines are computed from the fitted variogram models with the `variogramLine` function.

```
> ylim.plot=c(0, max(vr.gwr$gamma, vr.gls$gamma, vr$gamma))
> plot(gamma ~ dist, data=vr.gwr, ylim=ylim.plot,
+      type="b", lty=2, col="red", xlab="separation, m",
+      ylab="semivariance, (lbs plot-1)^2",
+      main="Regression model residuals")
> lines(variogramLine(vmf.r.gwr, maxdist=max(vr.gwr$dist)),
+       col="red")
> points(gamma ~ dist, data=vr, type="b", lty=2, col="blue")
> lines(variogramLine(vgmr, maxdist=max(vr.gwr$dist)),
+       col="blue")
> points(gamma ~ dist, data=vr.gls, type="b", lty=2, col="green")
> lines(variogramLine(vmf.r.gls, maxdist=max(vr.gwr$dist)),
+       col="green")
> legend(20,0.15,c("OLS", "GLS", "GWR"), lty=1,
+        col=c("blue","green","red"))
```



Q144 : *Explain the change in spatial structure from (1) OLS residuals, (2) GLS residuals, (3) GWR residuals.* *Jump to A144* •

19.1 Answers

A139 : *In this case of a bivariate relation, the bandwidth might be expected to correspond to the variogram range for the grain/straw ratio, which also represents*

the same process as the straw vs. grain regression. Return to Q139 •

A140 : *The bandwidth 7.8 m is a bit less than the range for grain/straw with a spherical model, 8.4; but they are comparable.* Return to Q140 •

A141 : *There is quite a wide range of both coefficients; the IQR is also wide. The global coefficients are nowhere near the medians of the GWR coefficient distributions.* Return to Q141
•

A142 : *Most definitely. There is a very high straw yield per unit grain yield in the NW and SE, very low in the SW. The process by which carbohydrates are partitioned between grain and straw appears to be systematically different in patches of about 1/5 field size, not just random noise plot-to-plot.* Return to Q142 •

A143 : *Yes, the fitted variogram model shows a range of 8.6 m and a structural sill to total sill ratio of 37%.* Return to Q143 •

A144 : *The OLS fit has the highest residuals; these are slightly lowered by the REML fit, presumably because it better accounted for very high and low values close to each other. The GWR residuals are considerably lower and are fit with a slightly longer range. The lower residuals are because the GWR fit is local and so can better adjust to local relations between straw and grain.* Return to Q144 •

20 Periodicity*

Mercer & Hall asserted that the field was uniform; however others [31, 39, 45] have suggested that the field was not as uniform as assumed, and that there were systematic patterns of previous management that affected yield. In §9 we determined that the field halves had significantly different straw yields; in §17.4 we determined that the two halves had different spatial structures, i.e., local spatial dependence. These two findings support the idea that the two halves of the field had in the past been managed differently. However, there is another possible management effect: periodic variation across rows or columns due to previous ploughing or ridging. Had this field previously been used with a ridge-furrow system or divided into 'lands'?

> **Note:** There is also some evidence of periodicity in the analysis of local spatial structure of grain yields in §17.2. The variogram shows a dip at the 5^{th} bin, at about 11.6 m, then an increase in the 6^{th} (13.9 m), then again a dip in the 7^{th} (16.4 m).

Here we follow the analysis of McBratney and Webster [31] to investigate this possibility. The hypothesis is that in either the W–E (column-wise) or N–S (row-wise) direction that there is either positive or negative autocorrelation in grain or straw yields, at some spacing between columns or rows.

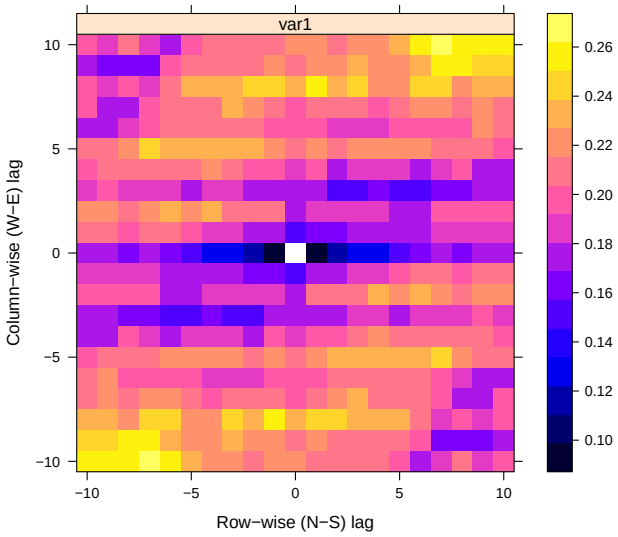
Note: The field layout is explained in §A.

To investigate this, we use the tools of one- and two-dimensional spatial correlation analysis. Thus we analyze the W–E dimension (columns along single rows) and the N–S dimension (rows along single columns), in all combinations. Note that 1D and 2D spatial correlations are symmetric; the same correlation is found E–W and W–E, and the same S–N as N–S; for the 2D case the correlations are radially symmetric.

20.1 Visualizing periodicity

In this section we attempt to reproduce Fig. 3 from McBratney and Webster [31], which shows an autocorrelation surface of grain yields, i.e., autocorrelations at all combinations of row and column lags. Autocorrelation is closely related to semivariance (see Eq. 20.1, just below), and fortunately a semivariogram surface can computed by the `variogram` "compute empirical variogram" method of the `gstat` package, specifying the `map` argument as `TRUE`; we apply this to a spatial version of the dataset, with coördinates specified by row and column; i.e., we compute lags in terms of rows and columns:

```
> mhw.rc <- mhw
> coordinates(mhw.rc) <- ~r + c
> v.map <- variogram(grain ~ 1, loc = mhw.rc, map = TRUE,
+     cutoff = 10, width = 1)
> summary(v.map$map$var1)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
 0.0986  0.1830  0.2040  0.2010  0.2170  0.2620       1

> class(v.map)

[1] "variogramMap" "list"

> plot(v.map, col.regions = bpy.colors(64), xlab = "Row-wise (N-S) lag",
+     ylab = "Column-wise (W-E) lag")
```



To convert semivariances γ to covariances C, and then to correlations ρ, we use the relations:

$$C(h) = C(0) - \gamma(h) \quad (20.1)$$

$$\rho(h) = C(h)/C(0) \quad (20.2)$$

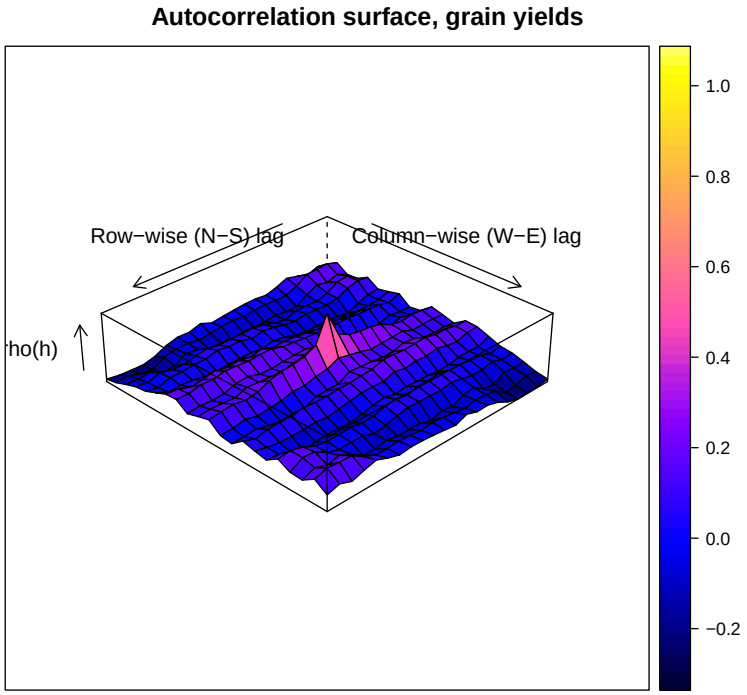
By definition the autocorrelation at the origin is 1. The covariance at a point $C(0)$ is estimated as as the variance over the field, using the `var` function. We also replace the "not applicable" semivariance at the origin with the known autocorrelation, i.e., 1, finding the proper location in the matrix with the `is.na` and `which` functions:

```
> c0 <- var(mhw$grain)
> v.map$map$cov <- c0 - v.map$map$var1
> v.map$map$cor <- v.map$map$cov/c0
> v.map$map$cor[which(is.na(v.map$map$cor))] <- 1
> summary(v.map$map$cor)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.2480 -0.0329  0.0298  0.0442  0.1320  1.0000
```

To view as a 3D plot, we convert the autocorrelations to a matrix using the `matrix` function, since this is the form required by the `wireframe` function. To get a perspective view, we use `aspect` (vertical and horizontal axis ratio) and `screen` (rotation and tilt) arguments.

```
> str(v.map$map@data)

'data.frame':        441 obs. of  4 variables:
 $ var1   : num  0.202 0.189 0.205 0.19 0.179 ...
 $ np.var1: num  150 165 180 195 210 225 240 255 270 285 ...
 $ cov    : num  0.00776 0.02126 0.00469 0.01954 0.03102 ...
 $ cor    : num  0.037 0.1012 0.0223 0.0931 0.1477 ...
```

```
> n <- sqrt(length(v.map$map$cor))
> v.map.mat <- matrix(v.map$map$cor, nrow=n, ncol=n)
> plot(wireframe(v.map.mat, drape=T, aspect=c(1,.25),
+                screen=c(z=225, x=-60), ylab="Column-wise (W-E) lag",
+                xlab="Row-wise (N-S) lag", zlab="rho(h)",
+                main="Autocorrelation surface, grain yields",
+                col.regions=bpy.colors(72)))
```



Note that this plot is diagonally symmetric. The front (positive, row-wise) half resembles Fig. 3 from McBratney and Webster [31], except at the $(+10, +10)$ corner where our figure shows a strong low autocorrelation (corresponding to a large semivariance in the variogram map) and the corresponding figure has another positive peak.

Q145 : *Does there appear to be any periodicity in one or both directions? What does this imply about the uniformity of the field?* *Jump to A145* •

Task 143 : Plot the average autocorrelation along rows and columns. •

The variogram map (surface) has this information in the central row and column of the matrix. Since the map is symmetric, we only need to plot from the centre out.

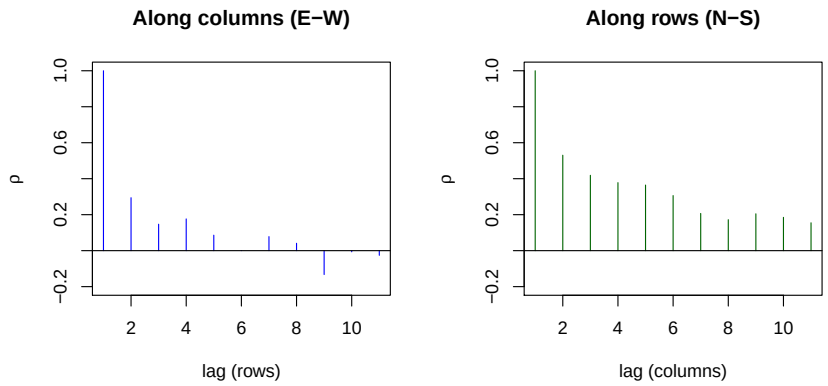
FIrst, compute the correlations along the central row (E–W) and column (N–S):

```
> (c <- ceiling(n/2))
```

```
[1] 11

> (v.map.mat[c:n, c])

 [1] 1.00000 0.53053 0.41881 0.37809 0.36462 0.30621 0.20680 0.17216
 [9] 0.20485 0.18515 0.15488

> (v.map.mat[c, c:n])

 [1]  1.00000000  0.29431834  0.14698574  0.17645418  0.08599151
 [6]  0.00029433  0.07806734  0.04102077 -0.13276224 -0.00651483
[11] -0.02592755
```

Plot the autocorrelations in the two directions:

```
> par(mfrow = c(1, 2))
> str(v.map.mat)

 num [1:21, 1:21] 0.037 0.1012 0.0223 0.0931 0.1477 ...

> plot(v.map.mat[c, c:n], type = "h", ylim = c(-0.2, 1),
+     main = "Along columns (E-W)", ylab = expression(rho),
+     col = "blue", xlab = "lag (rows)")
> abline(h = 0)
> plot(v.map.mat[c:n, c], type = "h", ylim = c(-0.2, 1),
+     main = "Along rows (N-S)", ylab = expression(rho),
+     col = "darkgreen", xlab = "lag (columns)")
> abline(h = 0)
> par(mfrow = c(1, 1))
```



Q146 : *Does there appear to be any periodicity in one or both directions?*
Jump to A146 •

20.2 Spectral analysis

McBratney and Webster [31] also examine the spatial autocorrelation in the *frequency* domain (as opposed to the *spatial* domain); that is, they consider the signal as a sum of periodic components (so-called Fourier analysis), and examine the *power spectrum*, that is, the relative contribution of each period to the overall signal. This should reveal the periodicity. This technique is

widely used in time-series analysis, but is equally applicable to data organized in space. The frequency is relative to a *cycle*, which in time series is some natural cycle such as a year or day. Here the cycle is the single field.

20.2.1 Theory

The theory of power spectra and the Fourier transform from and to the spatial domain are explained in the text of Webster and Oliver [49, Ch. 7]; we present enough theory here to motivate and interpret our application. We consider the 1D case, i.e., observations arranged in a line, to develop the theory, and then extend it to the 2D case.

The fundamental transformation from covariances at each lag $C(h)$ to the power at each frequency $R(f)$ is:

$$R(f) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \cos(fh) C(h) \mathrm{d}h \tag{20.3}$$

In words, the integral is of the covariances, each multiplied by the cosine at the given frequency, over all lags. As the frequency increases, the period of the cosine function gets shorter. If the higher covariances coincide with that period, the power at that frequency is higher.

Eqn. 20.3 assumes a 2nd–order stationary process in $\mathbb{R}^1$, in which as the lag h gets wider, the covariance $C(h)$ approaches 0.

This ideal equation can not be computed, for several reasons. First, we do not have an infinite number of lags (i.e., infinite-length sequence of observations), so the integral is restricted to some window L, after which the covariances ≈ 0. Second, the experimental covariances are increasingly unreliable at large lags, so a rule of thumb is to limit L to about one-fifth of a 1D series length. Finally, observations are not made at infintesimal lags, rather they have a minimum spacing, and higher-frequency components, if any, can not be estimated.

So, the spectrum is estimated for frequencies f, ranging from -0.5 to 0.5 cycles, from the empirical covariances $\hat{C}(k)$ estimated from the observations. So the spectrum is estimated as:

$$\hat{R}(f) = \frac{1}{2\pi} \left\{ \hat{C}(0) + 2 \sum_{k=1}^{L-1} \hat{C}(k) \cos(\pi f k) \right\} \tag{20.4}$$

Notice that the integral of Eqn. 20.3 is replaced by a sum over the chosen window. For correlations rather than covariances, replace C with c and omit the term with $\hat{C}(0)$. These sums estimate the relative contribution, also called *spectral density* or *power*, of each frequency to the overall signal in the 2nd-order stationary series.

Note that the cosine term varies from 0 to k for $f = 0 \ldots 0.5$, i.e., $\pi f = 0 \ldots \pi/2$. For example, at $f = 0$ (i.e., at the centre of a full cycle), in terms of correlations c:

$$\hat{R}(0) = \frac{1}{\pi} \sum_{k=1}^{L-1} \hat{c}(k)$$

At $f = 0.5$ (i.e., half a cycle):

$$\hat{R}(0.5) = \frac{1}{\pi} \sum_{k=1}^{L-1} \hat{c}(k) \cos(\frac{\pi}{2} k)$$

There is a further refinement to the estimate of Eq. 20.4: using a window that emphasizes the more reliable shorter lags, without discarding too much information from longer lags. Webster and Oliver [49, §7.3.1] propose several window functions $w(k)$ to multiply the covariances in Equation 20.4; the one used by McBratney and Webster [31, Eq. 9] is the Bartlett window:

$$w(k) = \begin{cases} 1 - (|k|/L) & \text{if } 0 \leq |k| \leq L \\ 0 & \text{if } |k| > L \end{cases} \tag{20.5}$$

> **Note:** Fig. 7.3 of Webster and Oliver [49] shows the relative weights of lags within a window for the rectangular, Bartlett, and Parzen window; these authors favour the latter.

There is only one quantity left to estimate; this is the auto-covariance at each lag h, including zero; this follows directly from the definition of covariance:

$$\hat{C}(h) = \frac{1}{N-h} \sum_{i=1}^{N-h} \{(z(i) - \bar{z})(z(i+h) - \bar{z})\} \tag{20.6}$$

where z is the data value.

We now apply this theory to the 2D case of the Mercer & Hall wheat field.

20.2.2 Covariance surface

We first compute the auto-covariances $c_{p,q}$ in two dimensions as [31, Eq. 6]:

$$c_{p,q} = \frac{1}{(m-p)(n-q)} \sum_{i=1}^{(m-p)} \sum_{j=1}^{(n-q)} (x_{i,j} - \bar{x})(x_{i+p,j+q} - \bar{x}) \tag{20.7}$$

where $\bar{x}$ is the mean of the $m \times n$ matrix, and the lags in the two dimensions are p and q. Thus, all possible combinations cells that contribute to the two-dimensional lag are included in the weighted sum. Auto-correlations are obtained by dividing the auto-covariances by $c_{0,0}$, i.e., the overall variance.

> **Note:** This should give identical results to the variogram surface, converted to an auto-correlation surface, which was computed by the `variogram` method in §20.1, above. However, its calculation is a good example of the use of the `for` flow control operator.[21]

To compute this sum, we use the `for` flow control operator in a so-called "for-loops". These are not much used in R, since in many cases vectorized operators can be used, but in this case we need to step through the matrix, and at each position compute a covariance of selected neighbour cells.

[21] This was covered in more detail in §15.

We first convert the grain yields to a matrix matching the field shape, and centre the yields on the mean, since the covariances are based on differences, not absolute values:

```
> m <- max(mhw$r)
> n <- max(mhw$c)
> str(mhw.grain.matrix <- matrix(data = (mhw.rc$grain -
+     mean(mhw.rc$grain)), nrow = m, ncol = n))

 num [1:20, 1:25] -0.3186 0.1214 0.5614 -0.0486 -0.3186 ...
```

Task 144 : Write a function to implement Eqn. 20.7. •

The following functions compute covariances $c_{p,q}$ and $c_{p,-q}$, respectively, when given the the row and column lags, and the data matrix, as arguments. The `return` function returns the computed covariance to the caller.

```
> cpq <- function(p, q, mat) {
+     s <- 0
+     for (i in 1:(m - p)) for (j in 1:(n - q)) s <- s +
+         (mat[i, j]) * (mat[i + p, j + q])
+     return((1/((m - p) * (n - q))) * s)
+ }
> cpmq <- function(p, q, mat) {
+     s <- 0
+     for (i in 1:(m - p)) for (j in (q + 1):n) s <- s +
+         (mat[i, j]) * (mat[i + p, j - q])
+     return((1/((m - p) * (n - q))) * s)
+ }
```

Task 145 : Apply these functions the desired combination of row and column lags. •

Following McBratney and Webster [31] we compute up to fourteen lags in both directions, although this is considerably more than the recommended one-fifth of the overall dimension (i.e., 4 rows and 5 columns). Note that the matrix is radial symmetric, so we can fill in the other two quadrants from the first two.

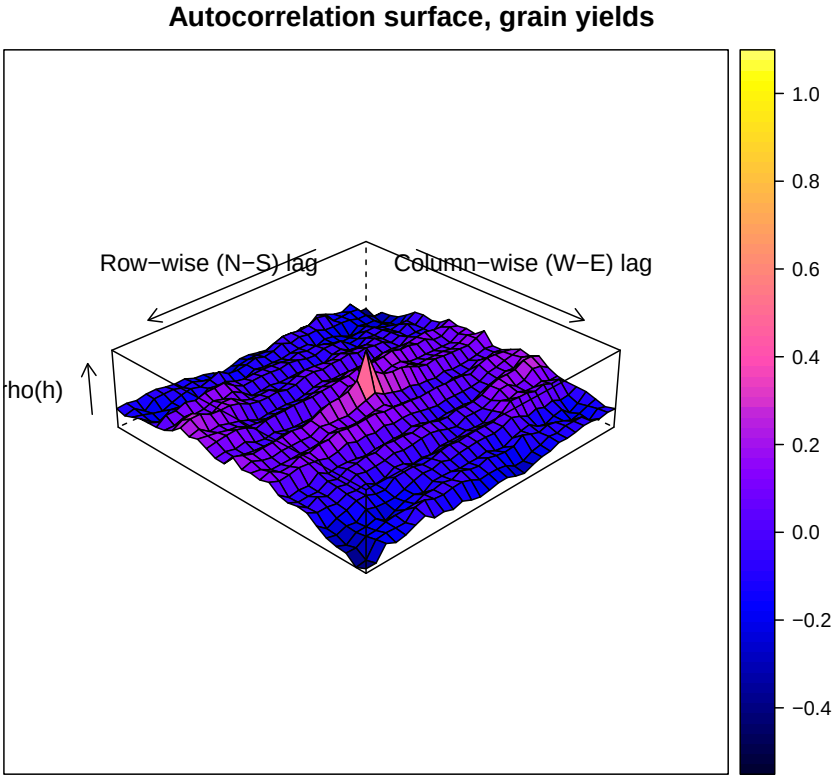
```
> max.l <- 14
> d <- 2 * max.l + 1
> ch <- matrix(0, d, d)
> for (lag.r in 1:max.l) for (lag.c in 1:max.l) {
+     ch[(max.l + 1) + lag.r, (max.l + 1) - lag.c] <- (cpmq(lag.r,
+         lag.c, mhw.grain.matrix))
+     ch[(max.l + 1) - lag.r, (max.l + 1) + lag.c] <- ch[(max.l +
+         1) + lag.r, (max.l + 1) - lag.c]
+ }
> for (lag.r in 0:max.l) for (lag.c in 0:max.l) {
+     ch[(max.l + 1) + lag.r, (max.l + 1) + lag.c] <- (cpq(lag.r,
+         lag.c, mhw.grain.matrix))
+     ch[(max.l + 1) - lag.r, (max.l + 1) - lag.c] <- ch[(max.l +
```

```
+         1) + lag.r, (max.l + 1) + lag.c]
+ }
> ch <- ch/var(mhw$grain)
> summary(as.vector(ch))

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.45000 -0.10400 -0.01260 -0.00802  0.09190  0.99800
```

Task 146 : Plot the autocorrelation surface. •

Again we use the `wireframe` function:

```
> plot(wireframe(ch, drape=T, aspect=c(1,.25),
+                screen=c(z=225, x=-60), ylab="Column-wise (W-E) lag",
+                xlab="Row-wise (N-S) lag", zlab="rho(h)",
+                main="Autocorrelation surface, grain yields",
+                auto.key=T,
+                col.regions=bpy.colors(72)))
```



Indeed, this is the same surface computed in §20.1, but extended to maximum lag 14, so the stretch is somewhat different, here reaching -0.45 at lag $(-14, +14)$. The periodicity in the N-S direction is obvious

20.2.3 Power spectrum

The 1D spectral calculation of Eqn. 20.4 can be extended into 2D, by considering the lags, correlations, and weights in 2D. For frequencies from -0.5 to 0.5 cycles per sampling interval, the 2D power spectrum (in the frequency domain) is estimated from the auto-covariances (in the spatial domain) as [31, Eq. 8]:

$$G_{r,s} = K^{-1} \sum_{q=-L}^{L} \sum_{p=-L}^{L} c_{p,q} w_{p,q} \cos\left\{(\pi/T)(rp + sq)\right\} \tag{20.8}$$

where:

- K is a normalizing constant, taken here as $(2\pi)^2 = 4\pi^2$;[22]
- L is the maximum lag, i.e., window size, to be included in the estimate;
- p, q are the lags in two dimensions (rows or columns);
- $r, s = -T, -T+1, \ldots, 0, 1, T-1, T$, i.e., the number of frequency estimates in the interval $-0.5 \ldots 0.5$ cycles;
- $w_{p,q}$ is Bartlett's smoothing function (Equation 20.5), which damps the cosine function's amplitude at greater lags, computed in 2D as:

$$w_{p,q} = \begin{cases} 1 - |\mathbf{h}|/L & \text{if } 0 \le |\mathbf{h}| \le L \\ 0 & \text{if } |\mathbf{h}| > L \end{cases}$$

  where $|\mathbf{h}| = \sqrt{p^2 + q^2}$, i.e., the Euclidean distance between plots, and L is the maximum lag at which we want compute the spectral density.

That is, the spectral estimate for a given combination of frequencies is a weighted sum of lagged covariances multiplied by the appropriate point in the period, i.e., the argument to the cosine function.

The number of frequency estimates T is not dependent on the number of lags; it is set by the analyst to obtain a sufficiently fine-resolution estimate. There is no disadvantage to a high value of T other than computation time; McBratney and Webster [31] used $T = 50$ to obtain their Fig. 5.

However, the choice of L is critical: small L are reliable but do not reveal high-frequency components; large L may be unreliable. The usual procedure is to compute for a variety of L and examine the spectra to see where sufficient detail, without excessive noise, is found. It is also possible to compute confidence intervals; see Webster and Oliver [49, §7.3.3]; we have not (yet) implemented that here.

Task 147 : Write a function to implement Eqn. 20.8, i.e., to convert the autocorrelation surface into a spectrum $G_{r,s}$ for each combination of frequency

[22] This constant only affects the absolute values, not the relative magnitudes, which are of primary interest.

estimates, from $-t, -t+1, \ldots, 0, 1, \ldots t-1, t$, where t is the number of frequency estimates. •

We use the `function` command to write a function from the lag combination `r` and `s`, the number of frequency estimates `t`, the maximum lag `L`, and the matrix of correlation coefficients `cor.mat`:

> **Note:** Note how this function also includes an internal function (i.e., only visible inside the function) to compute Barlett's weighting.

```
> grs <- function(r = 0, s = 0, t, L, cor.mat) {
+     w <- function(x, y) {
+         h <- sqrt(x^2 + y^2)
+         return(ifelse(h <= L, 1 - (h/L), 0))
+     }
+     max.p <- dim(cor.mat)[1]
+     max.q <- dim(cor.mat)[2]
+     centre.p <- ((max.p - 1)/2) + 1
+     centre.q <- ((max.q - 1)/2) + 1
+     sum <- 0
+     for (q in -L:L) {
+         if (centre.q + q + 1 > max.q)
+             break
+         for (p in -L:L) {
+             if (centre.p + p + 1 > max.p)
+                 break
+             s1 <- (cor.mat[centre.p + p, centre.q + q] *
+                 w(p, q) * cos((pi/t) * ((r * p) + (s *
+                 q))))
+             sum <- sum + s1
+         }
+     }
+     return(sum/(4 * pi^2))
+ }
```

Task 148 : Apply this function at a resolution of 50 estimates per cycle, first along the 1D W-E axis, with no N-S offsets and for a window size $L = 10$. •

Since the spectrum is symmetric, we only need to compute the positive half, i.e., from $s = 0 \ldots T$:

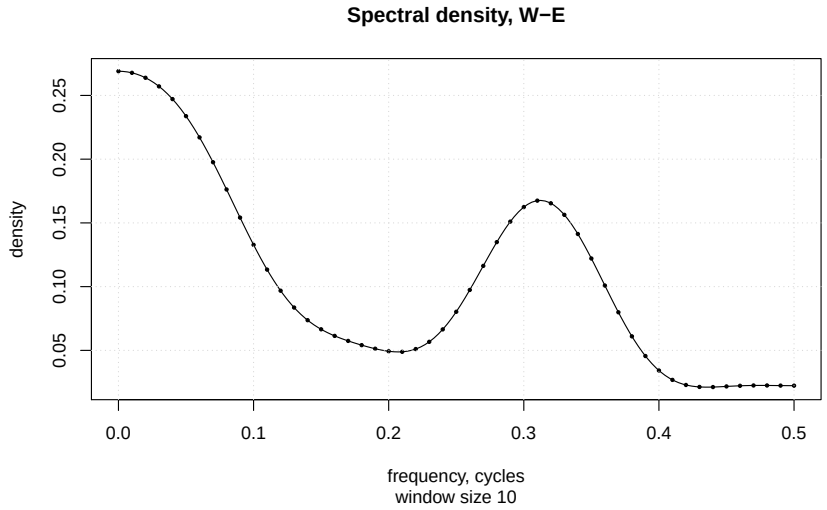
```
> theta <- 50
> dens <- rep(0, theta + 1)
> for (s in 0:theta) dens[s + 1] <- grs(0, s, t = theta,
+     L = 10, cor.mat = ch)
```

Task 149 : Plot the spectral density as a function of frequency. •

We use the interpolating spline function `spline` to smooth the graph; note that the estimates at each knot are not changed by the spline:

```
> plot(dens ~ seq(0, 0.5, length = theta + 1), type = "p",
+     main = "Spectral density, W-E", sub = "window size 10",
```

```
+     ylab = "density", xlab = "frequency, cycles", pch = 20,
+     cex = 0.6)
> dens.smooth <- spline(dens)
> lines(dens.smooth$y ~ seq(0, 0.5, length = length(dens.smooth$x)),
+     lty = 1)
> grid()
```



Recall that the selection of the window width L is subjective, so we now look for the width that best reveals the spectral characteristics.

Task 150 : Compute the power spectrum along the E-W dimension for window sizes $L = 4, 6, 8, 10, 12, 14$ and plot their spectral densities vs. frequency on one graph. •

We set up a matrix whose rows are the window size and whose columns are the densities for that window size, at the chosen resolution:

```
> l.s <- seq(4,14,by=2)
> theta <- 50
> dens <- matrix(rep(0, length(l.s)*(theta+1)),
+                nrow=length(l.s))
```

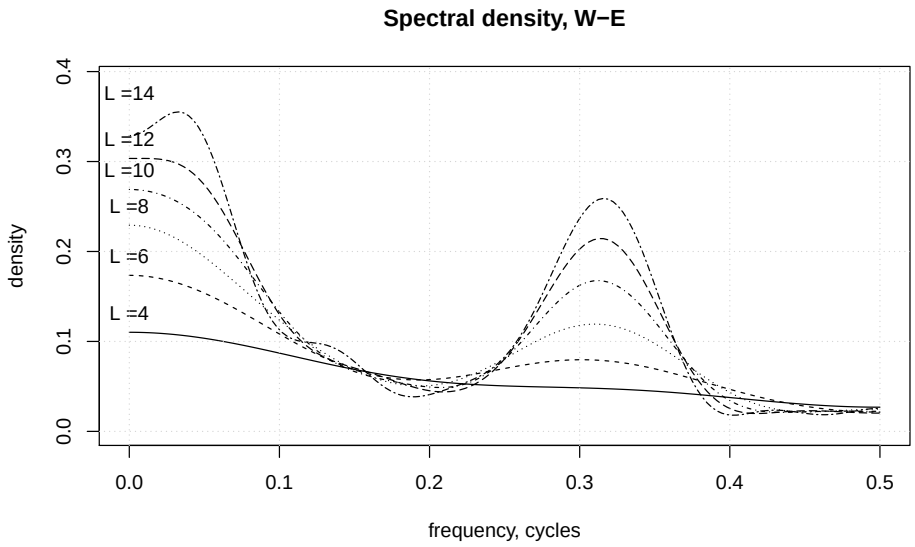
We then compute the spectral density for each window size, again using a `for` loop to step through the window sizes, i.e., rows of the results matrix:

```
> for (i in 1:length(l.s)) {
+     for (s in 0:theta) {
+         dens[i, s + 1] <- grs(0, s, theta, l.s[i], ch)
+     }
+ }
```

Finally, we plot them on one graph, first setting up the axes and then using a `for` loop to place each curve in the figure. This corresponds to Fig. 5 in McBratney and Webster [31].

```
> plot(dens[1, ] ~ seq(0, 0.5, length = theta + 1), type = "n",
+     main = "Spectral density, W-E", ylab = "density",
```

```
+     xlab = "frequency, cycles", ylim = c(min(0, dens),
+         max(dens) * 1.1))
> for (i in 1:length(l.s)) {
+     dens.smooth <- spline(dens[i, ])
+     lines(dens.smooth$y ~ seq(0, 0.5, length = length(dens.smooth$x)),
+         lty = i)
+     text(0, max(dens[i, ]), paste("L =", l.s[i], sep = ""),
+         pos = 3)
+ }
> grid()
```



Note: This figure is slightly different from Fig. 5 in McBratney and Webster [31]; the reason is not clear. However the major features are similar.

Q147 : *What are the outstanding features of the power spectrum? What do they imply about periodicity in this direction?* *Jump to A147* •

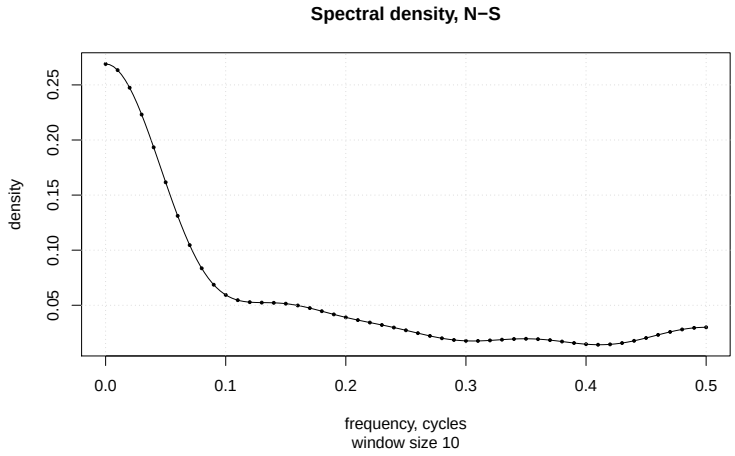
Q148 : *Which window size best reveals the features?* *Jump to A148* •

In §20.1 we concluded that there was periodicity in the W–E direction (column-wise along rows) but not the N–S direction (row-wise along column); the W–E power spectrum confirms the first; now we examine the second.

Task 151 : Compute and plot the N–S power spectrum for window width $L = 10$. •

```
> theta <- 50
> dens <- rep(0, theta + 1)
```

```
> for (r in 0:theta) dens[r + 1] <- grs(r, 0, t = theta,
+     L = 10, cor.mat = ch)
> plot(dens ~ seq(0, 0.5, length = theta + 1), type = "p",
+     main = "Spectral density, N-S", sub = "window size 10",
+     ylab = "density", xlab = "frequency, cycles", pch = 20,
+     cex = 0.6)
> dens.smooth <- spline(dens)
> lines(dens.smooth$y ~ seq(0, 0.5, length = length(dens.smooth$x)),
+     lty = 1)
> grid()
```



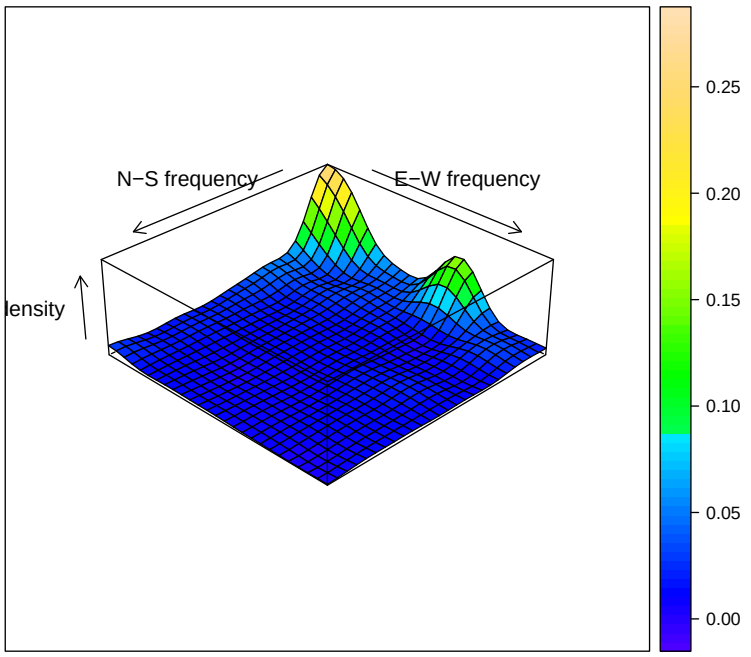
Q149 : *Is there any evidence of periodicity in this direction?* *Jump to A149* •

We can visualize the power in both directions simultaneously, and also discover if there are any interactions of the directions; for example if the periodicity were not along rows or columns.

Task 152 : Compute and plot the two-dimensional power spectrum for window width $L = 10$; this corresponds to Fig. 7 in McBratney and Webster [31]. •

This requires $O(\theta^2)$ density calculations, so we reduce the frequency resolution to 25 to speed up the computation:

```
> theta = 25
> dens <- matrix(rep(0, (theta + 1)^2), nrow = theta +
+     1)
> for (s in 0:theta) for (r in 0:theta) dens[r + 1, s +
+     1] <- grs(r, s, theta, 10, ch)
> wireframe(dens, drape = T, aspect = c(1, 0.35), screen = c(z = 225,
+     x = -60), xlab = "N-S frequency", ylab = "E-W frequency",
+     zlab = "density", auto.key = T, col.regions = topo.colors(72))
```



Task 153 : Remove the variables created in this section. •

```
> rm(mhw.rc, v.map, c0, v.map.mat, m, n, c, max.l, lag.r,
+     lag.c, r, s, theta, dens, dens.smooth, l.s, grs,
+     ch)
```

Challenge: Repeat this analysis for straw yields. Does this confirm, weaken, or strengthen the conclusions from the analysis of grain yields?

Challenge: Recall (§17.4) that the local spatial structure is different in the N and S field halves. Is the periodicity in the E–W direction present in both field halves? If so, does it show the same pattern?

Challenge: Use anisotropic variogram analysis to confirm the periodic effect.

20.3 Answers

A145 : *There are clear "ripples" on the autocorrelation surface in the E–W direction (columns along rows); these appear to be at 3-plot intervals. However, this pattern varies somewhat with increasing separation in the N–S direction. There seems to be no such a pattern in the N–S direction, just the expected decreasing autocorrelation with separation.* Return to Q145 •

A146 : *The autocorrelation graphs show clearly the pattern of the previous answer, i.e., along rows there is a clear dip in autocorrelation at lags 3 and 6, and even*

a negative autocorrelation at lag 9. No such pattern is seen along columns. This implies a periodic structure along the E-W (row-wise) direction, at an interval of approximately three rows; recalling that the row length is 3.30 *m* (§A)*, the periodicity appears to have a fundamental period of about* 10 *m; this is also the conclusion of McBratney and Webster [31, §4], who examine the likely causes.* *Return to Q146* •

A147 : *The highest power is at the origin, i.e., the overall mean, representing a full cycle. However there is significant power at approximately* $0.3\bar{3}$ *cycles, i.e.,* $20/3 = 6.\bar{6}$ *plots.* *Return to Q147* •

A148 : $L = 10$ *is sufficiently detailed to clearly see the spike at* $0.3\bar{3}$ *cycles without the small fluctuations for* $L > 10$*. At* $L < 4$ *the feature becomes increasingly vague.* *Return to Q148* •

A149 : *No.* *Return to Q149* •

21 The effect of plot size*

Mercer and Hall's original research objective was to determine how within-plot variability is affected by plot size. To investigate this, they grouped the 1 acre field into plots of 1/500 acre (the original plots), 1/250 acre, 1/125 acre, 1/50 acre, 1/25 acre, and finally 1/10 acre; they measured the variability in the resulting samples and graphed this by plot size. They then could determine how large a plot would be necessary to reduce the variability to an acceptable level. We will repeat their analysis here.

Q150 : *Based on the geostatistical analysis (§17.2), what size plot would be expected to remove most of the* ***local*** *variation?* *Jump to A150* •

Before proceeding, we need an operational definition of heterogeneity, so we can compare the variability between different plot sizes. In §7 we used the **probable error**, but that required modelling a normal distribution. A simpler, and commonly-used, measure of variability for samples that are approximately normally-distributed is the *coefficient of variation* (CV), which is defined as:

$$\text{CV} \quad = \quad s/\bar{x}$$

This normalizes the sample standard deviation s by the sample mean $\bar{x}$. It is commonly expressed in percent. This measure was also used by Mercer and Hall[23].

There is no R function to compute the CV; we can compute it on any sample by first using the `mean` function and then the `sd` function, and then taking their ratio:

[23] although they mistakenly refer to it as "standard deviation"

```
> round(100 * sd(mhw$grain)/mean(mhw$grain), 1)

[1] 11.6
```

To avoid writing this out for each vector whose CV we want to compute, we can write a small **function** to do this computation on any sample; this uses the `function` function to define the algorithm.

Task 154 : Write a **function** to compute the CV of a vector. •

```
> cv <- function(x) {
+     round(100 * sd(x)/mean(x), 1)
+ }
> class(cv)

[1] "function"
```

The object `cv` in the workspace is a function which can now be applied to any vector, just like a built-in R function.

Q151 : *What is the CV of the grain yields of the entire set of 500 plots?* *Jump to A151* •

```
> cv(mhw$grain)

[1] 11.6
```

Now we group the plots into increasingly-larger plots and see how the CV of the set is affected. Mercer and Hall compared six sizes: 1/250, 1/125, 1/100, 1/50, 1/25 and 1/10 acre; these are all possible groupings of rows and columns to this size, given the field layout.

Task 155 : Determine how adjacent plots can be grouped in increasingly-large blocks, up to 1/10 acre. •

We are restricted to divisors of 20 (rows), i.e. 2, 2 and 5, and 25 (columns), i.e. 5:

1/500 acre : Original layout; grid is (20 x 25);

1/250 acre : Combine the plots in each two adjacent rows; resulting grid is (10 x 25);

1/125 acre : Combine the plots in each four adjacent rows; resulting grid is (5 x 25);

1/100 acre : Combine the plots in each five adjacent columns; resulting grid is (20 x 5);

1/50 acre : Combine the plots in each five adjacent columns and two adjacent rows; resulting grid is (10 x 5);

1/25 acre : Combine the plots in each five adjacent columns and four adjacent rows; resulting grid is (5 x 5);

1/10 acre : Combine plots in each five adjacent columns and ten adjacent rows; resulting grid is (2 x 5);

Three larger sizes are possible: 1/5, 1/4 and 1/2 acre; however these do not have enough plots to evaluate variability.

Task 156 : Make a data structure with each plot size and its dimensions. •

From §16.2 we have the plot length, width, and area:

```
> plot.len; plot.wid; plot.area

[1] 3.1807

[1] 2.5446

[1] 8.0937
```

We can use these to compute dimensions for each combinations.

We first make a data frame with information about the combinations, using the `data.frame` function to combine three lists, each made with the `c` function; we also name the rows with the `row.names` function for easy reference. We name each field (column) explicity with the `field.name = ...` syntax.

```
> plots <- data.frame(acre.fraction = c(500, 250, 125,
+     100, 50, 25, 10), adj.rows = c(1, 2, 4, 1, 2, 4,
+     10), adj.col = c(1, 1, 1, 5, 5, 5, 5))
> row.names(plots) <- c("1/500", "1/250", "1/125", "1/100",
+     "1/50", "1/25", "1/10")
> str(plots)

'data.frame':        7 obs. of  3 variables:
 $ acre.fraction: num  500 250 125 100 50 25 10
 $ adj.rows     : num  1 2 4 1 2 4 10
 $ adj.col      : num  1 1 1 5 5 5 5
```

Now we can compute dimensions and add them to the frame with the `cbind` function:

```
> plots <- cbind(plots, len = plot.len * plots$adj.row)
> plots <- cbind(plots, wid = plot.wid * plots$adj.col)
> plots <- cbind(plots, area = plots$len * plots$wid)
> plots

      acre.fraction adj.rows adj.col     len     wid     area
1/500           500        1       1  3.1807  2.5446   8.0937
1/250           250        2       1  6.3615  2.5446  16.1874
1/125           125        4       1 12.7230  2.5446  32.3748
1/100           100        1       5  3.1807 12.7230  40.4686
1/50             50        2       5  6.3615 12.7230  80.9371
1/25             25        4       5 12.7230 12.7230 161.8742
1/10             10       10       5 31.8075 12.7230 404.6856
```

Note how the row names are shown when the entire frame is printed; this allows us to identify each combination.

Q152 : *What are the dimensions of these in meters, and their areas in m^2?* *Jump to A152* •

Task 157 : Compute the grain yields for the 1/250 acre plots made up of two adjacent rows, and its CV. •

To group in pairs, we make use of the **modulus** arithmetic operator `%%` to identify the odd and even rows:

```
> head(mhw$r%%2, 20)

 [1] 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
```

Now we split the plots into two groups using the `unstack` function and sum the two adjacent plots:

```
> tmp <- unstack(mhw, grain ~ r%%2)
> str(tmp)

'data.frame':        250 obs. of  2 variables:
 $ X0: num  4.07 3.9 3.16 3.42 3.4 4.43 4.46 5.13 4.38 3.61 ...
 $ X1: num  3.63 4.51 3.63 3.18 3.97 3.39 4.52 3.46 4.23 3.85 ...

> grain.250 <- tmp$X0 + tmp$X1
> rm(tmp)
> str(grain.250)

 num [1:250] 7.7 8.41 6.79 6.6 7.37 7.82 8.98 8.59 8.61 7.46 ...

> cv(grain.250)

[1] 10.1
```

Q153 : *Is the variation reduced, as expected, when plot size is doubled? By how much?* *Jump to A153* •

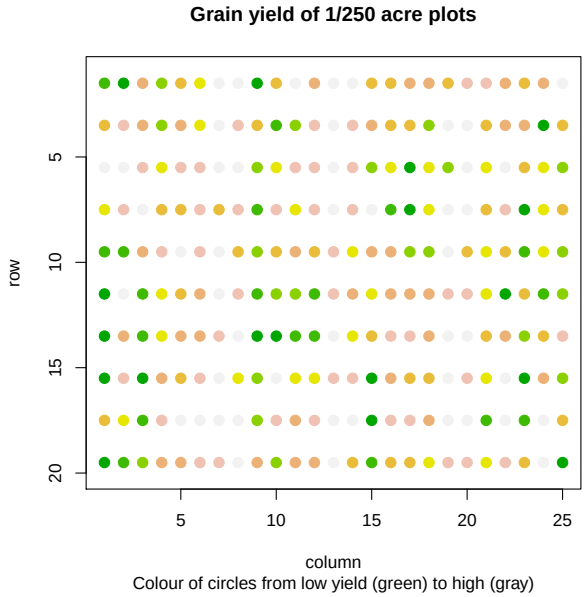
We now build a data frame of the combined plots, using the `data.frame` function, labeling each with its original column number and the average of the two rows:

```
> plots.250 <- data.frame(r = seq(1.5, 19.5, by = 2), c = rep(1:25,
+     each = 10), grain = grain.250)
> str(plots.250)

'data.frame':        250 obs. of  3 variables:
 $ r    : num  1.5 3.5 5.5 7.5 9.5 11.5 13.5 15.5 17.5 19.5 ...
 $ c    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ grain: num  7.7 8.41 6.79 6.6 7.37 7.82 8.98 8.59 8.61 7.46 ...
```

Task 158 : Visualise the variation across the field with the 1/250 acre plot size. •

We visualize by colour ramp:

```
> plot(plots.250$c, plots.250$r, pch=20, cex=2,
+      bg="lightblue", xlab="column", ylab="row",
+      main="Grain yield of 1/250 acre plots",
+      sub="Colour of circles from low yield (green) to high (gray)",
+      xlim=c(1, 25), ylim=c(20, 1),
+      col=terrain.colors(8)[cut(grain,
+        quantile(grain, seq(0, 1, length=9)),
+        include.lowest=T, labels=F)])
```



These figures can be compared to those for the 1/500 acre plots in §16.1.

Q154 : *Does the field appear more homogeneous with 250 vs. 500 plots? What about the pattern of spatial dependence?* *Jump to A154* •

Task 159 : Repeat this process for the other combinations. •

First, for 1/125 acre. We introduce the very useful `apply` function, which applies any other function (here, the `sum`) across an array margin; here the `1` as second argument specifies that the sum is across rows of the matrix. Since the results of `unstack` are organized into a set of rows, this will add the four plots.

```
> tmp <- unstack(mhw, grain ~ r%%4)
> str(tmp)

'data.frame':        125 obs. of  4 variables:
 $ X0: num  3.9 3.42 4.43 5.13 3.61 4.64 3.35 3.7 3.89 4.22 ...
```

```
 $ X1: num  3.63 3.63 3.97 4.52 4.23 4.15 4.27 3.61 3.79 3.87 ...
 $ X2: num  4.07 3.16 3.4 4.46 4.38 4.21 3.55 3.71 4.09 4.12 ...
 $ X3: num  4.51 3.18 3.39 3.46 3.85 4.29 3.5 3.64 4.42 4.28 ...

> grain.125 <- apply(tmp, 1, sum)
> rm(tmp)
> str(grain.125)

 num [1:125] 16.1 13.4 15.2 17.6 16.1 ...

> cv(grain.125)

[1] 8.9

> plots.125 <- data.frame(r = seq(2, 18, by = 4), c = rep(1:25,
+     each = 10), grain = grain.125)
> str(plots.125)

'data.frame':        250 obs. of  3 variables:
 $ r    : num  2 6 10 14 18 2 6 10 14 18 ...
 $ c    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ grain: num  16.1 13.4 15.2 17.6 16.1 ...
```

For the 1/100 acre plots the combination is by column:

```
> tmp <- unstack(mhw, grain ~ c%%5)
> grain.100 <- apply(tmp, 1, sum)
> rm(tmp)
> cv(grain.100)

[1] 7

> plots.100 <- data.frame(r = rep(1:20, each = 5), c = seq(3,
+     23, by = 5), grain = grain.100)
> str(plots.100)

'data.frame':        100 obs. of  3 variables:
 $ r    : int  1 1 1 1 1 2 2 2 2 2 ...
 $ c    : num  3 8 13 18 23 3 8 13 18 23 ...
 $ grain: num  20 21.1 21.7 20.1 21.2 ...
```

The 1/50 acre plots are the first where both rows and columns are combined. So we have to repeat the unstacking process twice. However, we can start from the 1/100 acre frame which already has combined the columns.

```
> tmp <- unstack(plots.100, grain ~ r%%2)
> grain.50 <- apply(tmp, 1, sum)
> rm(tmp)
> cv(grain.50)

[1] 5.9

> plots.50 <- data.frame(r = rep(seq(1.5, 19.5, by = 2),
+     each = 5), c = seq(3, 23, by = 5), grain = grain.50)
> str(plots.50)
```

```
'data.frame':        50 obs. of  3 variables:
 $ r    : num  1.5 1.5 1.5 1.5 1.5 3.5 3.5 3.5 3.5 3.5 ...
 $ c    : num  3 8 13 18 23 3 8 13 18 23 ...
 $ grain: num  39.1 39.7 40.9 40.6 41.1 ...
```

The 1/25 acre plots are constructed similarly, but combining four instead of two rows:

```
> tmp <- unstack(plots.100, grain ~ r%%4)
> grain.25 <- apply(tmp, 1, sum)
> rm(tmp)
> cv(grain.25)

[1] 4.8

> plots.25 <- data.frame(r = rep(seq(2.5, 18.5, by = 4),
+     each = 5), c = seq(3, 23, by = 5), grain = grain.25)
> str(plots.25)

'data.frame':        25 obs. of  3 variables:
 $ r    : num  2.5 2.5 2.5 2.5 2.5 6.5 6.5 6.5 6.5 6.5 ...
 $ c    : num  3 8 13 18 23 3 8 13 18 23 ...
 $ grain: num  79.9 79.8 83.8 84.6 82.4 ...
```

The 1/10 acre plots are constructed similarly, but combining ten rows:

```
> tmp <- unstack(plots.100, grain ~ r%%10)
> grain.10 <- apply(tmp, 1, sum)
> rm(tmp)
> cv(grain.10)

[1] 3.9

> plots.10 <- data.frame(r = rep(seq(5.5, 15.5, by = 10),
+     each = 5), c = seq(3, 23, by = 5), grain = grain.10)
> str(plots.10)

'data.frame':        10 obs. of  3 variables:
 $ r    : num  5.5 5.5 5.5 5.5 5.5 15.5 15.5 15.5 15.5 15.5
 $ c    : num  3 8 13 18 23 3 8 13 18 23
 $ grain: num  203 204 205 207 202 ...
```

Now we attempt to answer Mercer & Hall's research question.

Q155 : *What is the trend of the summary statistics (extremes, mean, median, IQR) as the plot size increases, normalized to a 1/500 acre basis?*
Jump to A155 •

> **Note:** To compare these we have to divide the summary by the number of 1/500 acre plots making up the larger plots:

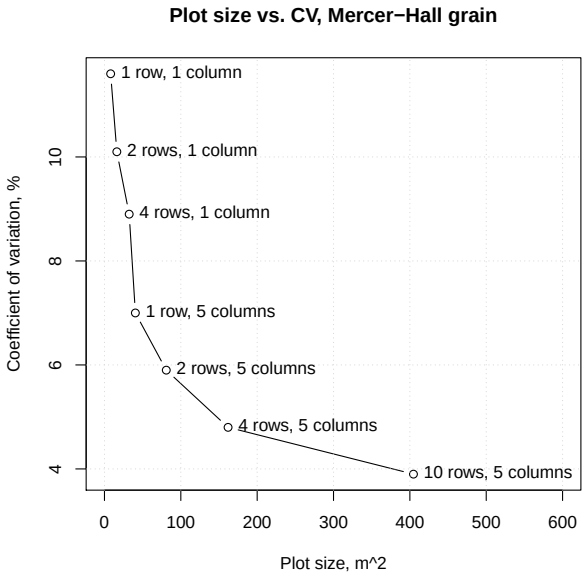
```
> summary(mhw$grain)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.73    3.64    3.94    3.95    4.27    5.16
```

```
> summary(plots.250$grain)/2

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.890   3.695   3.945   3.950   4.205   5.100

> summary(plots.125$grain)/4

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.025   3.675   3.925   3.950   4.150   4.800

> summary(plots.100$grain)/5

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   3.22    3.78    3.98    3.94    4.14    4.50

> summary(plots.50$grain)/10

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   3.40    3.82    3.95    3.95    4.11    4.40

> summary(plots.25$grain)/20

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.665   3.780   3.900   3.950   4.130   4.250

> summary(plots.10$grain)/50

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   3.72    3.82    3.98    3.94    4.08    4.14
```

Q156 : *What is the trend in the CV as the plot size increases?* *Jump to A156* •

```
> print(size.cv <- data.frame(area = plots$area, cv = c(
+    cv(mhw$grain), cv(plots.250$grain), cv(plots.125$grain),
+    cv(plots.100$grain), cv(plots.50$grain), cv(plots.25$grain),
+    cv(plots.10$grain))))

      area   cv
1   8.0937 11.6
2  16.1874 10.1
3  32.3748  8.9
4  40.4686  7.0
5  80.9371  5.9
6 161.8742  4.8
7 404.6856  3.9

> plot(size.cv$area, size.cv$cv, xlab="Plot size, m^2",
+      ylab="Coefficient of variation, %",
+      main="Plot size vs. CV, Mercer-Hall grain", type="b",
+      xlim=c(0,600))
> grid()
> text(size.cv$area, size.cv$cv, pos=4,
+      paste(
+         plots$adj.rows,
+         " row",
```

```
+          ifelse(plots$adj.rows==1,"","s"),
+          ", ",
+          plots$adj.col,
+          " column",
+           ifelse(plots$adj.col==1,"","s"),
+           sep=""))
```



Q157 : *What plot size do you recommend?* *Jump to A157* •

We now clean up from this section:

```
> rm(cv, plot.len, plot.wid, plot.area, plots, plots.250,
+     plots.125, plots.100, plots.50, plots.25, plots.10,
+     grain.250, grain.125, grain.100, grain.50, grain.25,
+     grain.10, size.cv)
```

21.1 Answers

A150 : *The range of local spatial dependence was about 8 m; plot size is (3.30 m long x 2.45 m wide) (§A); grouping six plots into one plot of (6.60 m long x 7.35 m wide) would remove most of this structure; grouping 12 plots into one plot (9.90 m long x 9.70 m wide) would remove all of it.* *Return to Q150* •

A151 : *11.6%.* *Return to Q151* •

A152 : *The dimensions are:*

```
> plots[, c("len", "wid", "area")]
```

	len	wid	area
1/500	3.1807	2.5446	8.0937
1/250	6.3615	2.5446	16.1874
1/125	12.7230	2.5446	32.3748
1/100	3.1807	12.7230	40.4686
1/50	6.3615	12.7230	80.9371
1/25	12.7230	12.7230	161.8742
1/10	31.8075	12.7230	404.6856

Return to Q152 •

A153 : *Yes, it is reduced somewhat, from 11.6%. to 10.1%. .* *Return to Q153* •

A154 : *There are half the plots so the detailed spatial structure is lost. However there are still clear patches of higher and lower yields.* *Return to Q154* •

A155 : *The means are almost identical (3.95 to 3.95), and the medians close (3.94 to 3.98); however the extremes (and hence the range) are reduced as plot size increases (from* $2.73 \ldots 5.16$ *in the full set to* $3.72 \ldots 4.14$ *in the largest plot) and the IQR is somewhat narrower (from* $3.64 \ldots 4.27$ *in the full set to* $3.82 \ldots 4.08$ *in the largest plot).* *Return to Q155* •

A156 : *The CV decreases rapidly at first, from 11.6% (500 plots) to 10.1% (250 plots) to 8.9% (125 plots) to 7% (100 plots), and then less dramatically, to 5.9% (50 plots), 4.8% (25 plots), and 3.9% (10 plots). The graph has a hyperbolic shape and shows a clear inflection point around 80* m^2 *plot size (50 plots per acre).* *Return to Q156* •

A157 : *This depends on the precision required, which depends on the purpose of the experiment. However, the apparent inflection point of the CV vs. plot size curve is at two rows, five columns, i.e., plots of 1/50 acres, or about 80* m^2*: 6.36 m long by 12.72 m wide. Put another way, one acre could be used for a trial of ten treatments (e.g., crop varieties or fertilizer combinations) with five replications, with a CV due to random error (not treatment effects) of 5.9%.* *Return to Q157* •

References

[1] E C Barrett and L F Curtis. *Introduction to environmental remote sensing*. Stanley Thornes Publishers, Cheltenham, Glos., UK, 4th edition, 1999. 118

[2] D Bates. Fitting linear mixed models in R. *R News*, 5(1):27–30, 2005. 184

[3] G Bellocchi, M Rivington, M Donatelli, and K Matthews. Validation of biophysical models: issues and methodologies. a review. *Agronomy for Sustainable Development*, 30(1):109–130, 2010. doi: 10.1051/agro/2009001. 129

[4] D Birkes and Y Dodge. *Alternative methods of regression*. Wiley series in probability and mathematical statistics. Wiley, 1993. ISBN 0471568813. 98

[5] R S Bivand, E J Pebesma, and V GÃşmez-Rubio. *Applied Spatial Data Analysis with R*. UseR! Springer, 2008. URL http://www.asdar-book.org/. 155

[6] G Casella and R L Berger. *Statistical inference*. Duxbury, Pacific Grove, CA, 2nd edition, 2002. 4

[7] W G Cochran. A catalogue of uniformity trial data. *Supplement to the Journal of the Royal Statistical Society*, 4(2):233–253, 1937. 227

[8] J C Davis. *Statistics and data analysis in geology*. John Wiley & Sons, New York, 3rd edition, 2002. 61, 118

[9] A C Davison and D V Hinkley. *Bootstrap methods and their application*. Cambridge University Press, 1997. ISBN 0521573912. 81, 83

[10] B Efron. *The jackknife, the bootstrap, & other resampling plans*. Society for Industrial & Applied Mathematics, 1982. 143

[11] B Efron. *An introduction to the bootstrap*. Chapman & Hall, 1993. 81

[12] B Efron and G Gong. A leisurely look at the bootstrap, the jackknife & cross-validation. *American Statistician*, 37:36–48, 1983. 80, 143

[13] H Fairfield Smith. An empirical law describing heterogeneity in the yields of agricultural crops. *The Journal of Agricultural Science (Cambridge)*, 28:1–3, 1938. 227

[14] A S Fotheringham, C Brunsdon, and M Charlton. *Quantitative geography: perspectives on spatial data analysis*. Sage Publications, 2000. 190

[15] A S Fotheringham, C Brunsdon, and M Charlton. *Geographically weighted regression: the analysis of spatially varying relationships*. Wiley, 2002. ISBN 0471496162. 190

[16] J Fox and S Weisberg. Robust regression in R: An appendix to "An R companion to applied regression, second edition". Dec

2010. URL http://socserv.mcmaster.ca/jfox/Books/Companion/appendix/Appendix-Robust-Regression.pdf. 98

[17] K. R. Gabriel. The biplot graphic display of matrices with application to principal component analysis. *Biometrika*, 58(3):453–467, 1971. doi: 10.2307/2334381. 119

[18] H G Gauch Jr., J T G Hwang, and G W Fick. Model evaluation by comparison of model-based predictions and measured values. *Agronomy Journal*, 95(6):1442–1446, 2003. 129

[19] J C Gower. Statistics and agriculture. *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, 151(1):179–200, 1988. 227

[20] R J Hyndman and Y N Fan. Sample quantiles in statistical packages. *American Statistician*, 50(4):361–365, Nov 1996. 80

[21] R Ihaka and R Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996. 1

[22] K Kobayashi. Comments on another way of partitioning mean squared deviation proposed by Gauch et al. (2003). *Agronomy Journal*, 96(4): 1206–1207, Aug 2004. 129

[23] K Kobayashi and M U Salam. Comparing simulated and measured values using mean squared deviation and its components. *Agronomy Journal*, 92(2):345–352, Apr 2000. 129

[24] R M Lark and B R Cullis. Model based analysis using REML for inference from systematically sampled data on soil. *European Journal of Soil Science*, 55(4):799–813, 2004. 46, 174, 183, 184

[25] R M Lark and J V Stafford. Information on within-field variability from sequences of yield maps: multivariate classification as a first step of interpretation. *Nutrient Cycling in Agroecosystems*, 50(1 - 3):277–281, 1998. 227

[26] P Legendre and L Legendre. *Numerical ecology*. Elsevier Science, 1998. 118

[27] F Leisch. Sweave, part I: Mixing R and LaTeX. *R News*, 2(3):28–31, December 2002. URL http://CRAN.R-project.org/doc/Rnews/. 2

[28] F Leisch. *Sweave User's Manual*. TU Wein, Vienna (A), 2.1 edition, 2006. URL http://www.ci.tuwien.ac.at/~leisch/Sweave. 2

[29] T M Lillesand, R W. Kiefer, and J W Chipman. *Remote sensing and image interpretation*. John Wiley & Sons, Hoboken, NJ, 6th edition, 2007. 118

[30] D M Mark and M Church. On the misuse of regression in earth science. *Mathematical Geology*, 9(1):63–77, 1977. 40

[31] A B McBratney and R Webster. Detection of ridge and furrow patterns by spectral analysis of crop yield. *International Statistical Review*, 49:

45–52, 1981. 152, 195, 196, 198, 199, 201, 202, 204, 206, 207, 208, 210, 227, 228

[32] W B Mercer and A D Hall. The experimental error of field trials. *The Journal of Agricultural Science (Cambridge)*, 4:107–132, 1911. 227

[33] V N Patankar. The goodness of fit of frequency distributions obtained from stochastic processes. *Biometrika*, 41:450–462, 1954. 170, 227

[34] E J Pebesma. Multivariable geostatistics in S: the gstat package. *Computers & Geosciences*, 30(7):683–691, 2004. 155

[35] E J Pebesma and R S Bivand. Classes and methods for spatial data in R. *R News*, 5(2):9–13, November 2005. URL http://CRAN.R-project.org/doc/Rnews/. 155

[36] E J Pebesma and C G Wesseling. Gstat: a program for geostatistical modelling, prediction and simulation. *Computers & Geosciences*, 24(1): 17–31, 1998. URL http://www.gstat.org/. 155

[37] J C Pinheiro and D M Bates. *Mixed-effects models in S and S-PLUS*. Springer, 2000. ISBN 0387989579. 175, 184

[38] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2004. URL http://www.R-project.org. ISBN 3-900051-07-0. 1

[39] B D Ripley. *Spatial statistics*. John Wiley and Sons, New York, 1981. 195, 227, 228

[40] D G Rossiter. *Introduction to the R Project for Statistical Computing for use at ITC*. International Institute for Geo-information Science & Earth Observation (ITC), Enschede (NL), 4.0 edition, 2012. URL http://www.itc.nl/personal/rossiter/teach/R/RIntro_ITC.pdf. 1, 7

[41] C Shalizi. The bootstrap. *American Scientist*, 98(3):186–190, 2010. 81

[42] G K Smyth and A P Verbyla. A conditional likelihood approach to residual maximum likelihood estimation in generalized linear models. *Journal of the Royal Statistical Society B*, 58(3):565–572, 1996. 184

[43] P Sprent. *Models in regression and related topics*. Methuen's monographs on applied probability and statistics. Methuen, London,, 1969. 61

[44] J W Tukey. *Exploratory data analysis*. Addison-Wesley, New York, 1977. 14

[45] H M van Es and C L van Es. Spatial nature of randomization and its effects on the outcome of field experiments. *Agronomy Journal*, 85: 420–428, 1993. 73, 195, 227, 228

[46] W N Venables and B D Ripley. *Modern applied statistics with S*. Springer-Verlag, New York, fourth edition, 2002. 17, 40, 98

[47] R Webster. Is regression what you really want? *Soil Use & Management*, 5(2):47–53, 1989. 40

[48] R Webster. Regression and functional relations. *European Journal of Soil Science*, 48(3):557–566, 1997. 40, 44, 61

[49] Richard Webster and Margaret A. Oliver. *Geostatistics for environmental scientists*. John Wiley & Sons Ltd., 2nd edition, 2008. ISBN 9780470517260. 200, 201, 204

[50] P Whittle. On stationary processes in the plane. *Biometrika*, 41:434–449, 1954. 227

[51] P Whittle. On the variation of yield variance with plot size. *Biometrika*, 43:337–343, 1956. 227

Index of R Concepts

A Example Data Set

In the early days of scientific agriculture, Mercer and Hall [32] were trying to determine the optimum plot size for agricultural yield trials:

- Plots that are too *small* will be too variable;
- Plots that are too *large* waste resources (land, labour, seed); if the land area is limited, the number of treatments will be unnecessarily small.

So, they performed a very simple experiment: an apparently homogeneous field was selected, prepared as uniformly as possible and planted to the same variety of wheat. They attempted to treat all parts of the field field exactly the same in all respects during subsequent farm operations. When the wheat had matured, the field was divided into 500 equally-size plots. Each plot was harvested separately. Both grain and straw were air-dried, then hand-threshed and weighed to a precision of 0.01 lb (= 4.54 g). The reported values are thus air-dry weight, lb plot^{-1}.

The field was a square of 1 acre[24], which is 0.40469 ha or 4,046.9 m^2, which was divided into a 20 rows by 25 columns, giving 500 plots, each of 1/500 acre, which is about 8.09 m^2 (3.30 m long x 2.45 m wide). We do not have records of the original orientation of the field, so we assume that the rows ran W to E, with 25 plots in each row, beginning at 1 on the W and running to 25 at the E, so that columns run N to S with 20 plots in each, running from 1 at the N to 20 at the S. Thus the NW corner (1,1) is plot 1, the NE corner (1, 25) is plot 481, the SE corner (25, 20) is plot 500, and the SW corner (1, 20) is plot 20.

Research questions This experiment was one of a series of so-called **uniformity trials** which were conducted early in the 20$^{\text{th}}$ century [7, 13], mainly to determine optimum plot sizes [51], field layouts and numbers of replications[25].

This data set has attracted many statisticians since 1911 [13, 25, 31, 33, 39, 45, 50] because of a simple fact: although the yields should be identical, they are not; in fact they vary considerably. How is this to be explained?

Mercer and Hall distinguished two possible causes:

> "If we consider the causes of variation in the yield of a crop it seems that broadly speaking they are divisible into two kinds. The first are random, occurring at haphazard all over the field. Such would be attacks by birds, the incidence of weeds or the presence of lumps of manure. The second occur with more regularity, increasing from point to point or having centres from which they spread outwards; we may take as instances of this kind changes of soil, moist patches over springs or the presence of rabbit holes along a hedge."

[24] 2.471054 acres = 1 ha = 10 000 m^2

[25] The intimate relation between the development of applied statistics and scientific agriculture is given in fascinating detail by Gower [19]

The first we recognize as multiple small random errors, with no spatial pattern, which should be **normally-distributed** (Gaussian) errors.

The second, however, may be evidence of **local spatial autocorrelation**, which can be investigated by *geo-statistical* analysis.

Others [31, 39, 45] have suggested that the field was not as uniform as assumed, so that there are both random effect as mentioned by Mercer and Hall but also systematic effects from previous management. These effects could include a regional trend, management blocks, or periodic effects due to, e.g., ridge-and-furrow or previous use as an orchard.

The CSV file The data has been prepared as the **comma-separated values** ("CSV") file `mhw.csv` in a plain-text editor. The first line gives the four field names:

```
"r","c","grain","straw"
```

These represent:

`r` : Row number in the field

`c` : Column number in the field

`grain` : Grain yield, lbs plot^{-1}

`straw` : Straw yield, lbs plot^{-1}

The following 500 lines each represent a plot; the four fields are separated by commas. For example, the first line is:

```
1,1,3.63,6.37
```

If you can not find the CSV file in digital form, here it is to copy-and-paste into a text file, which you should name `mhw.csv`. Note this is presented here in three columns to save space; the file should be one long column of four fields.

```
"r","c","grain","straw"
1,1,3.63,6.37
2,1,4.07,6.24
3,1,4.51,7.05
4,1,3.9,6.91
5,1,3.63,5.93
6,1,3.16,5.59
7,1,3.18,5.32
8,1,3.42,5.52
9,1,3.97,6.03
10,1,3.4,5.66
11,1,3.39,5.61
12,1,4.43,7.07
13,1,4.52,7.1
14,1,4.46,7.16
15,1,3.46,8.85
16,1,5.13,8.37
17,1,4.23,6.89
18,1,4.38,6.72
19,1,3.85,6.59
20,1,3.61,6.2
1,2,4.15,6.85
2,2,4.21,7.29
3,2,4.29,7.71
4,2,4.64,8.23
5,2,4.27,7.73
6,2,3.55,6.45
7,2,3.5,5.87
8,2,3.35,5.71
9,2,3.61,6.01
10,2,3.71,6.29
11,2,3.64,6.3
12,2,3.7,6.17
13,2,3.79,6.33
14,2,4.09,7.22
15,2,4.42,5.2
16,2,3.89,7.05
17,2,3.87,6.82
18,2,4.12,7.38
19,2,4.28,7.03
20,2,4.22,7.65
1,3,4.06,7.19
2,3,4.15,7.41
3,3,4.4,7.35
4,3,4.05,7.89
5,3,4.92,8.58
6,3,4.08,7.04
7,3,4.23,7.02
8,3,4.07,7.05
9,3,4.67,7.64
10,3,4.27,7.17
11,3,3.84,6.6
12,3,3.82,6.87
13,3,4.41,7.03
```

```
14,3,4.39,7.73
15,3,4.29,7.52
16,3,4.26,6.99
17,3,4.23,7.14
18,3,4.39,7.55
19,3,4.69,8.06
20,3,4.42,8.45
1,4,5.13,7.99
2,4,4.64,7.8
3,4,4.69,7.5
4,4,4.04,6.66
5,4,4.64,7.86
6,4,4.73,7.98
7,4,4.39,6.98
8,4,4.66,7.28
9,4,4.49,6.95
10,4,4.42,6.95
11,4,4.51,7.86
12,4,4.45,7.17
13,4,4.57,7.93
14,4,4.31,7.31
15,4,4.08,6.67
16,4,4.32,6.93
17,4,4.58,7.73
18,4,3.92,6.7
19,4,5.16,8.78
20,4,5.09,8.72
1,5,3.04,4.71
2,5,4.03,6.34
3,5,3.77,6.17
4,5,3.49,5.7
5,5,3.76,6.05
6,5,3.61,5.89
7,5,3.28,4.97
8,5,3.72,5.78
9,5,3.75,5.94
10,5,4.13,7.31
11,5,4.01,7.18
12,5,3.59,6.53
13,5,3.94,7.06
14,5,4.29,7.08
15,5,3.96,6.54
16,5,3.78,6.72
17,5,3.19,6.06
18,5,4.84,8.85
19,5,4.46,7.54
20,5,3.66,7.09
1,6,4.48,6.08
2,6,3.74,6.63
3,6,4.46,6.98
4,6,3.91,6.46
5,6,4.1,6.77
6,6,3.66,6.15
7,6,3.56,6.06
8,6,3.84,6.1
9,6,4.11,6.83
10,6,4.2,6.86
11,6,4.21,8.23
12,6,4.37,8.75
13,6,4.47,8.53
14,6,4.47,8.15
15,6,3.96,7.1
16,6,3.54,6.46
17,6,3.49,6.63
18,6,3.94,6.75
19,6,4.41,8.15
20,6,4.22,7.72
1,7,4.75,7.31
2,7,4.56,7.88
3,7,4.76,8.18
4,7,4.52,7.6
5,7,4.4,7.91
6,7,4.39,7.36
7,7,4.94,8.06
8,7,4.44,7.5
9,7,4.64,7.92
10,7,4.66,7.59
11,7,4.77,8.23
12,7,4.45,8.74
13,7,4.42,8.02
14,7,4.37,7.69
15,7,3.89,6.86
16,7,4.27,7.79
17,7,3.91,7.34
18,7,4.38,7.43
19,7,4.68,7.51
20,7,4.06,7.06
1,8,4.04,6.08
2,8,4.27,6.35
3,8,3.76,5.93
4,8,4.52,7.29
5,8,4.17,7.33
6,8,3.84,6.28
7,8,4.06,6.81
8,8,3.4,5.97
9,8,2.99,5.07
10,8,3.61,6.33
11,8,3.95,7.11
12,8,4.08,7.17
13,8,3.92,6.7
14,8,3.44,6.62
15,8,4.11,7.58
16,8,4.12,7.32
17,8,4.41,7.53
18,8,4.24,7.32
19,8,4.37,7.19
20,8,3.97,7.53
1,9,4.14,6.98
2,9,4.03,6.91
3,9,3.3,5.95
4,9,3.05,5.82
5,9,3.67,7.33
6,9,4.26,7.61
7,9,4.32,7.37
8,9,4.07,6.99
9,9,4.37,7.25
10,9,3.99,7.26
11,9,4.17,7.52
12,9,3.72,7.28
13,9,3.86,7.2
14,9,3.82,7.05
15,9,3.73,6.89
16,9,4.13,7.24
17,9,4.21,7.41
18,9,3.96,7.04
19,9,4.15,7.47
20,9,3.89,7.36
1,10,4,5.87
2,10,4.5,6.5
3,10,3.67,6.2
4,10,4.59,5.41
5,10,5.07,8.05
6,10,4.36,5.58
7,10,4.86,7.51
8,10,4.93,7.57
9,10,5.02,8.23
10,10,4.44,7.75
11,10,4.39,7.73
12,10,4.56,7.73
13,10,4.77,7.67
14,10,4.63,7.87
15,10,4.03,7.16
16,10,4.47,7.84
17,10,4.61,7.51
18,10,4.29,6.96
19,10,4.91,7.96
20,10,4.46,6.91
1,11,4.37,6.75
2,11,3.97,6.09
3,11,3.94,6.18
4,11,4.01,5.99
5,11,3.83,6.36
6,11,3.79,5.46
7,11,3.96,6.23
8,11,3.93,6.13
9,11,3.56,5.75
10,11,3.86,6.14
11,11,4.17,7.2
12,11,4.1,6.9
13,11,4.99,7.82
14,11,4.36,7.39
15,11,4.09,7.03
16,11,3.41,5.96
17,11,4.27,7.17
18,11,4.52,7.73
19,11,4.68,8.07
20,11,4.44,6.87
1,12,4.02,6.1
2,12,4.19,6.43
3,12,4.07,6.37
4,12,3.34,5.6
5,12,3.63,6.43
6,12,4.09,6.1
7,12,3.74,6.38
8,12,3.04,4.96
9,12,3.59,6.03
10,12,3.99,6.26
11,12,4.17,7.08
12,12,3.07,6.12
13,12,3.91,7.34
```

```
14,12,3.79,6.33
15,12,3.82,7.3
16,12,3.55,6.7
17,12,4.06,7
18,12,4.19,7.3
19,12,5.13,8.31
20,12,4.52,8.17
1,13,4.58,7.23
2,13,4.05,6.57
3,13,3.73,6.02
4,13,4.06,6.19
5,13,3.74,6.13
6,13,3.72,6.03
7,13,4.33,6.79
8,13,3.72,5.97
9,13,4.05,6.82
10,13,3.37,6.25
11,13,4.09,7.28
12,13,3.99,7.13
13,13,4.09,7.72
14,13,3.56,6.69
15,13,3.57,6.55
16,13,3.16,5.84
17,13,3.75,6.31
18,13,4.49,7.57
19,13,4.19,6.93
20,13,3.7,6.8
1,14,3.92,6.33
2,14,3.97,6.03
3,14,4.58,7.23
4,14,3.19,6.56
5,14,4.14,5.98
6,14,3.76,5.49
7,14,3.77,5.48
8,14,3.93,6.07
9,14,3.96,6.35
10,14,3.47,5.78
11,14,3.29,5.71
12,14,3.14,5.05
13,14,3.05,5.7
14,14,3.29,5.71
15,14,3.43,5.38
16,14,3.47,5.84
17,14,3.91,6.21
18,14,3.82,6.37
19,14,4.41,6.78
20,14,4.28,6.97
1,15,3.64,5.11
2,15,3.61,5.58
3,15,3.64,5.86
4,15,3.75,4.62
5,15,3.7,7.67
6,15,3.37,5
7,15,3.71,5.66
8,15,3.71,5.79
9,15,3.75,5.12
10,15,3.09,5.47
11,15,3.37,6.44
12,15,4.86,6.39
13,15,3.39,5.86
14,15,3.64,6.36
15,15,3.73,8.58
16,15,3.3,5.7
17,15,3.51,5.99
18,15,3.6,6.34
19,15,3.54,5.58
20,15,3.24,5.95
1,16,3.66,5.96
2,16,3.82,5.8
3,16,4.07,6.74
4,16,4.54,7.08
5,16,3.92,6.14
6,16,4.01,5.99
7,16,4.59,7.28
8,16,4.76,6.49
9,16,4.73,8.64
10,16,4.2,6.49
11,16,3.74,8.63
12,16,4.36,7.26
13,16,3.6,6.27
14,16,3.6,5.84
15,16,3.39,6.42
16,16,3.39,5.8
17,16,3.45,6.05
18,16,3.14,5.48
19,16,3.01,5.68
20,16,3.29,5.58
1,17,3.57,5.12
2,17,3.44,5
3,17,3.44,5.56
4,17,3.97,6.03
5,17,3.79,5.33
6,17,3.87,5.57
7,17,3.97,6.03
8,17,3.83,6.29
9,17,4.24,6.45
10,17,4.09,6.16
11,17,3.41,5.78
12,17,3.51,6.11
13,17,4.13,6.87
14,17,3.19,5.87
15,17,3.08,5.42
16,17,2.92,4.95
17,17,3.05,7.64
18,17,2.73,4.77
19,17,2.85,4.96
20,17,3.48,5.52
1,18,3.51,5.05
2,18,3.92,5.83
3,18,3.53,4.91
4,18,3.77,5.79
5,18,4.29,5.58
6,18,4.35,6.09
7,18,4.38,6.24
8,18,3.71,5.91
9,18,4.21,6.29
10,18,4.07,6.18
11,18,3.86,6.14
12,18,3.47,5.9
13,18,3.89,6.23
14,18,3.8,6.14
15,18,3.48,5.52
16,18,3.23,5.33
17,18,3.68,5.82
18,18,3.09,5.41
19,18,3.36,6.14
20,18,3.49,5.82
1,19,4.27,6.54
2,19,4.26,8.61
3,19,4.2,6.55
4,19,4.3,5.95
5,19,4.22,6.15
6,19,4.24,5.88
7,19,3.81,5.69
8,19,3.54,5.21
9,19,3.85,6.15
10,19,4.09,5.47
11,19,4.36,7.39
12,19,3.94,6.68
13,19,3.67,6.2
14,19,3.72,6.34
15,19,3.05,5.2
16,19,3.25,5.25
17,19,3.52,5.85
18,19,3.66,5.84
19,19,3.85,6.15
20,19,3.68,6.76
1,20,3.72,5.47
2,20,4.36,6.14
3,20,4.31,6.44
4,20,4.1,5.96
5,20,3.74,5.76
6,20,3.58,5.61
7,20,4.06,6.25
8,20,3.66,5.78
9,20,4.41,6.15
10,20,3.95,6.11
11,20,4.54,7.46
12,20,4.47,7.84
13,20,4.54,7.33
14,20,3.91,6.96
15,20,3.65,6.6
16,20,3.86,6.64
17,20,3.91,6.71
18,20,3.77,6.98
19,20,4.15,6.85
20,20,3.36,6.08
1,21,3.36,4.76
2,21,3.69,5.56
3,21,4.33,6.17
4,21,3.81,6.13
5,21,3.55,5.89
6,21,4.2,5.92
7,21,3.42,5.45
8,21,3.95,5.92
9,21,4.21,6.04
10,21,4.08,7
11,21,4.24,7.2
12,21,4.11,6.95
13,21,4.11,6.64
```

```
14,21,3.35,6.27
15,21,3.71,6.29
16,21,3.22,5.4
17,21,3.87,6.13
18,21,3.48,6.14
19,21,3.93,6.57
20,21,3.71,6.35
1,22,3.17,4.95
2,22,3.53,5.09
3,22,3.66,6.15
4,22,3.89,5.92
5,22,3.67,5.45
6,22,3.94,5.87
7,22,3.05,4.57
8,22,3.84,5.66
9,22,3.63,5.81
10,22,4.03,5.72
11,22,4.08,6.54
12,22,3.97,6.47
13,22,4.58,6.79
14,22,4.11,6.64
15,22,3.25,6.37
16,22,3.69,5.93
17,22,3.87,7.5
18,22,3.76,6.11
19,22,3.91,6.09
20,22,3.54,6.21
1,23,2.97,4.53
2,23,3.14,5.11
3,23,3.59,5.41
4,23,3.32,4.62
5,23,3.57,5.24
6,23,4.24,5.82
7,23,3.44,4.56
8,23,3.76,5.24
9,23,4.17,5.58
10,23,3.97,5.65
11,23,3.89,5.98
12,23,4.07,5.8
13,23,4.02,6.35
14,23,4.39,6.11
15,23,3.69,5.18
16,23,3.8,5.7
17,23,4.21,5.48
18,23,3.69,5.43
19,23,4.33,6.04
20,23,3.59,4.66
1,24,4.23,6.08
2,24,4.09,5.91
3,24,3.97,6.28
4,24,3.46,5.41
5,24,3.96,5.6
6,24,3.75,5.5
7,24,2.78,4.28
8,24,3.47,5.59
9,24,3.44,4.81
10,24,2.84,4.1
11,24,3.47,5.84
12,24,3.56,6.38
13,24,3.93,5.69
14,24,3.47,5.78
15,24,3.43,5.82
16,24,3.79,6.21
17,24,3.68,6.01
18,24,3.84,6.35
19,24,4.21,6.98
20,24,3.76,6.36
1,25,4.53,6.78
2,25,3.94,5.68
3,25,4.38,7.49
4,25,3.64,6.55
5,25,4.31,6.56
6,25,4.29,6.15
7,25,3.44,5.68
8,25,4.24,7.26
9,25,4.55,6.32
10,25,3.91,5.96
11,25,3.29,5.65
12,25,3.83,6.29
13,25,4.33,7.11
14,25,3.93,6.07
15,25,3.38,5.68
16,25,3.63,5.99
17,25,4.06,6.88
18,25,3.67,6.33
19,25,4.19,6.93
20,25,3.36,6.33
```

B Colours

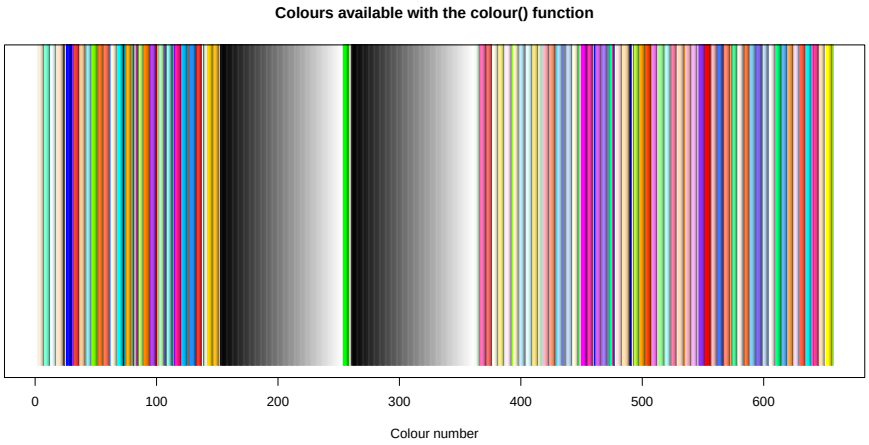
Colours may be specified in several ways; the most intuitive is by predefined **name**; these can be listed with the `colours` or `colors` methods.

```
> colours()

 [1] "white"          "aliceblue"      "antiquewhite"   "antiquewhite1"
 [5] "antiquewhite2" "antiquewhite3" "antiquewhite4" "aquamarine"
 ...
```

> **Note:** These colours are shown various ways on the PDF colour chart http://research.stowers-institute.org/efg/R/Color/Chart/ColorChart.pdf.

These colours can be visualised as a bar graph:

```
> plot(seq(1:length(colors())), rep(2, length(colours())), type="h",
+      lwd=2, col=colors(), ylim=c(0,1), xlab="Colour number",
+      ylab="", yaxt="n",
+      main="Colours available with the colour() function")
```



An individual colour number can be identified interactively with the `identify` function; left-click on the vertical colour bar at its midpoint; right-click anywhere in the graph when done.

```
> abline(h=0.5, lwd=3)
> (selected <- identify(seq(1:length(colors())),
+                       rep(0.5, length(colors()))))
> colors()[selected]; rm(selected)
```

For example, clicking on the light blue bar near colour 430, and then right-clicking to end the interaction, shows the colour number and name:

```
[1] 432
```

```
[1] "lightskyblue2"
```

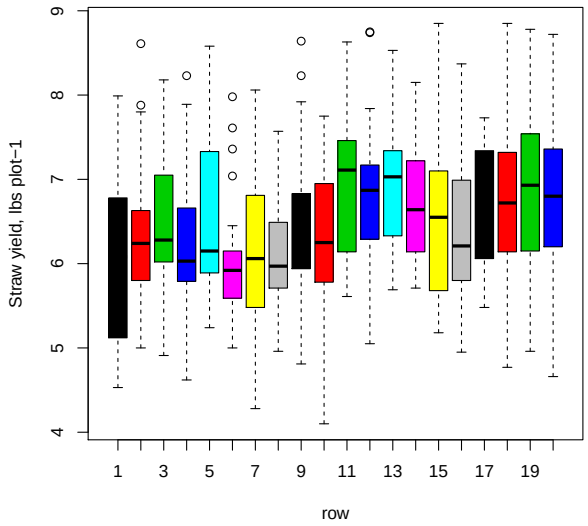
Colours can also be referred by **number**; this is their position in the active **palette**. These names are displayed or extracted with the `palette` function:

```
> palette()
```

```
[1] "black"   "red"     "green3"  "blue"    "cyan"    "magenta"
[7] "yellow"  "gray"

> palette()[2]

[1] "red"
```

Numbered colours are often used when the graphical element matches a numbered element in some data structure:

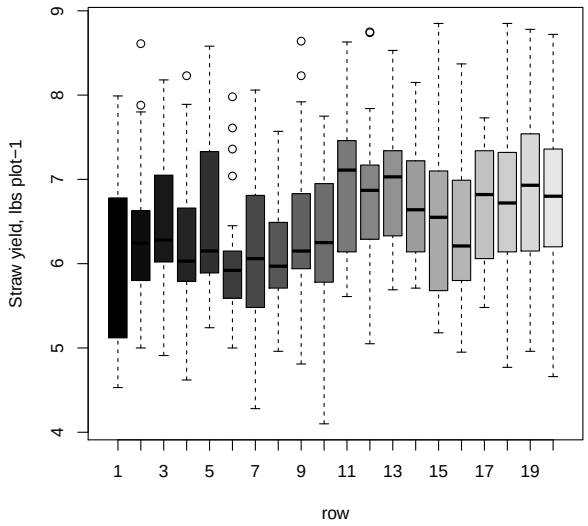
```
> boxplot(mhw$straw ~ mhw$r, col=mhw$r, xlab="row",
+         ylab="Straw yield, lbs plot-1")
```



Here the row number is directly used as the colour: row 1 black, row 2 red, row 3 green etc. Note that the colours are **recycled** if there are more plot elements than colours in the palette.

The `palette` function can also be used to set the palette. For example to make a 20-element grey-scale to match the 20 rows of wheat plots:

```
> palette(gray(seq(0,.9,len=20))); palette()

 [1] "black"    "#0C0C0C"  "#181818"  "gray14"   "gray19"   "#3C3C3C"
 [7] "#484848"  "#555555"  "gray38"   "#6D6D6D"  "#797979"  "gray52"
[13] "gray57"   "#9D9D9D"  "darkgray" "gray71"   "#C1C1C1"  "#CDCDCD"
[19] "gray85"   "#E6E6E6"

> boxplot(mhw$straw ~ mhw$r, col=mhw$r,
+         xlab="row", ylab="Straw yield, lbs plot-1")
> palette("default"); palette()

[1] "black"   "red"     "green3"  "blue"    "cyan"    "magenta"
[7] "yellow"  "gray"
```

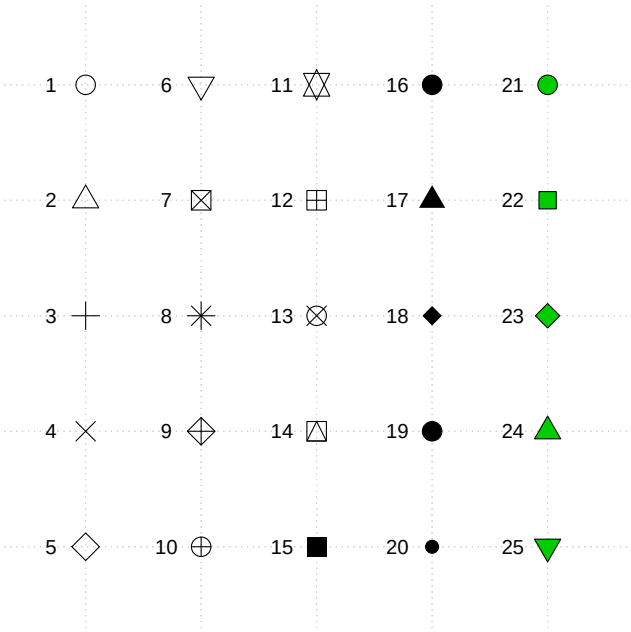


Note how some colours are given by their Red-Green-Blue saturations, as two hexidecimal digits per hue, e.g. `#E6E6E6#` is approximately 90% of white:

```
> as.numeric("0xe6")/as.numeric("0xff")

[1] 0.90196
```

This example also shows the `gray` function to set up a grey-scale colour ramp; other colour ramp functions are `rgb`, `hsv`, `rainbow`, and `heat.colors`.

Plotting symbols There are 25 pre-defined plotting symbols which can be specified with the `pch`, `fg` and `bg` graphics arguments:



In addition, ASCII characters can be used; e.g. `pch=51` prints a '1'.