Principles and Practice of Data Analysis

for Reproducible Research in R

Publication Quality Graphics and Tables

Heather Turner

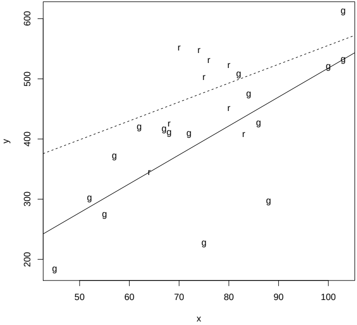
Department of Statistics, University of Warwick

2017–09–14

Publication Quality Graphics

Basic day-to-day plots may need tidying before publication

- Meaningful, readable annotations
- Sensible axis limits and tick intervals
- Uncluttered backgrounds
- Avoiding over-plotting
- Good choice of colours/adapting to greyscale



Text annotation

Axis titles are added in base R plots using `xlab` and `ylab` arguments. Special symbols can be added using ‘plotmath‘ expressions, e.g.

Subscript

```
expression(x[i])
```

Greek letters

```
expression(alpha)
```

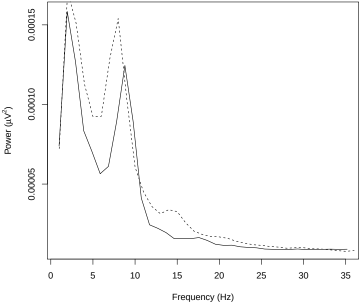
Plain text and math annotation

```
expression(paste0("Temp, ", degree, "C"))
```

Plot margins

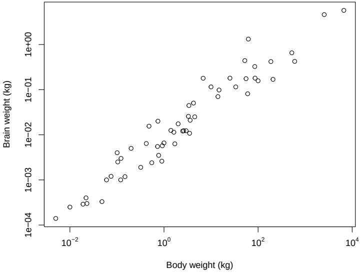
Add space on left to avoid cutting off superscript; reduce top and right margins

```
par(mar = c(5, 5, 1, 1) + 0.1) # bottom, left, top, right
plot(freq, spectra[[1]], type = "l", lty = 1,
     ylab = expression(paste("Power (", mu, V^2, ")")),
     xlab = "Frequency (Hz)")
lines(1:36, spectra[[2]], lty = 2)
```



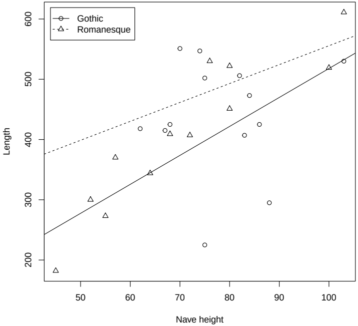
Custom Axes

```
library(magrittr) # for piping
library(purrr) # for mapping
msleep %$%
    plot(bodywt, brainwt, log = "xy", xaxt = "n",
         xlab = "Body weight (kg)", ylab = "Brain weight (kg)")
ticks <- seq(-2, 4, by = 2)
labels <- map(ticks, function(i) substitute(10^e, list(e = i)))
axis(1, at = 10^ticks, labels = as.expression(labels))
```



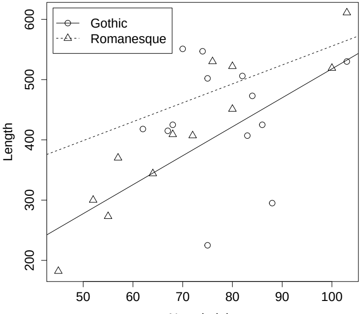
Legends

```r
cathedral %$%
    plot(y ~ x, pch = 1:2, xlab = "Nave height", ylab = "Length")
mod <- lm(y ~ x, subset = style == "g", data = cathedral)
abline(mod)
abline(update(mod, subset = style == "r"), lty = 2)
legend("topleft", inset = 0.02, lty = 1:2, pch = 1:2,
       legend = c("Gothic", "Romanesque"))
```



Element Sizing

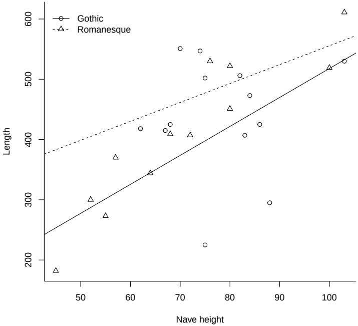
```
cathedral %$% {
    plot(y ~ x, type = "n", xlab = "Nave height", ylab = "Length",
         cex.axis = 1.4, cex.lab = 1.4)
    points(y ~ x, pch = 1:2, cex = 1.4)
}
abline(mod)
abline(update(mod, subset = style == "r"), lty = 2)
legend("topleft", inset = 0.02, lty = 1:2, pch = 1:2,
       legend = c("Gothic", "Romanesque"), cex = 1.4)
```



Remove Unnecessary Plot Elements

Boxes in base plot can be removed using `bty`

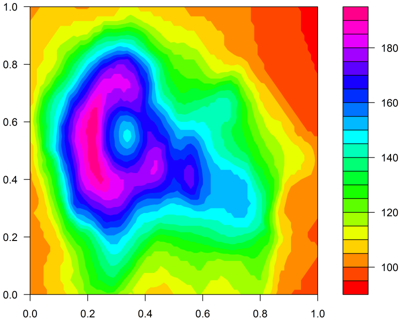
```
cathedral %$%
    plot(y ~ x, pch = 1:2, xlab = "Nave height", ylab = "Length", bty = "l
abline(mod)
abline(update(mod, subset = style == "r"), lty = 2)
legend("topleft", inset = 0.02, lty = 1:2, pch = 1:2,
       legend = c("Gothic", "Romanesque"), bty = "n")
```

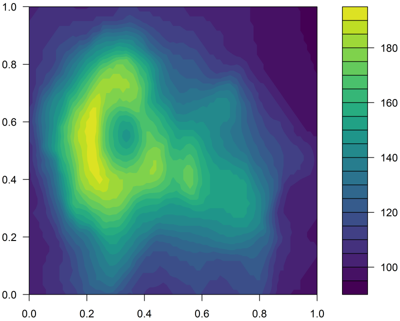


Continuous Colour Palettes

A good continous colour palette is

- perceptually uniform
- colour-blind friendly
- print-friendly



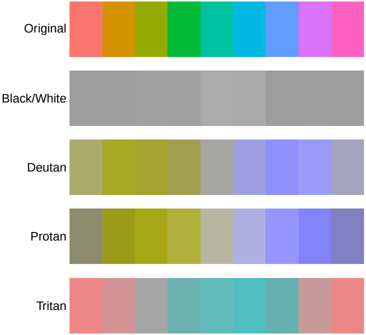


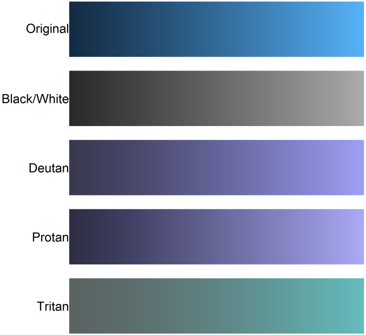
Testing Palettes

The **pals** package provides a large number of palettes, along with functions for testing propeties. E.g. for default ggplot2 scales

```
library(pals)
library(scales)
par(mar = c(0.1, 2.3, 0.1, 0.1))
pal.safe(hue_pal()(9))
```

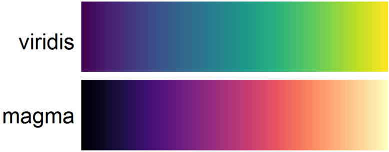
```
par(mar = c(0.1, 2.3, 0.1, 0.1))
x <- seq(0, 1, length.out = 100)
pal.safe(seq_gradient_pal(
    "#132B43", "#56B1F7", "Lab")(x))
```



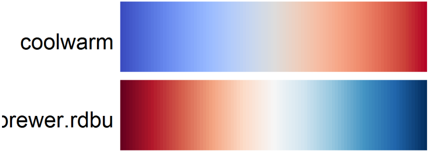


Continuous Colour Palettes

The *viridis* palettes in **pals** or **viridis** satisfy all requirements



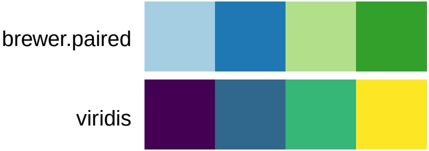
Diverging palettes cannot be print-friendly but the ‘pals‘ offers the following perceptually uniform and colour-blind friendly palettes



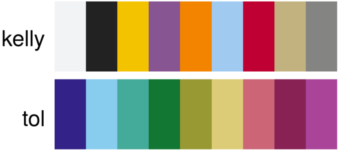
`brewer` palettes can be selected via http://colorbrewer2.org.

Discrete Colour Palettes

Discrete colour palettes can only print-friendly up to about 4 colours



Colour-blind friendly palettes are available up to about 9 colours



Too avoid problems distinguishing colours, consider using

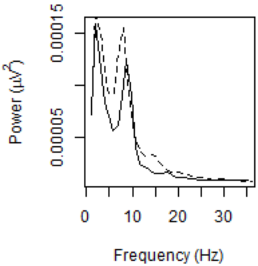
- shapes, line types or transparency
- small multiples (facets)

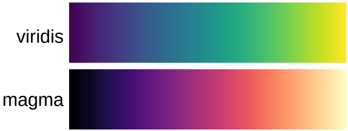
Graphic Formats

Graphics can be saved in two types of format

- *raster* composed of coloured pixels
- *vector* composed of coloured shapes

Using a format that does not suit the graphic can result in poor quality figures





Vector graphics with lots of "ink" produce large files and take a long time to load.

Choosing a format

Use vector graphics except for

- graphs with 1000s of elements, e.g. points
- graphs with large blocks of colour

Choose the file format according to document file type

Document file type	Graphics file type	
	Raster	Vector
.pdf	.png	**.pdf**
.html	**.png**	.svg
.docx	**.png**, .tiff	.eps, .svg

- bold are the defaults used by R markdown.
- .tiff, .pdf and .eps are most commonly required by journals
- .svg adapts to the fonts used in the document - useful for presentations and web publishing.

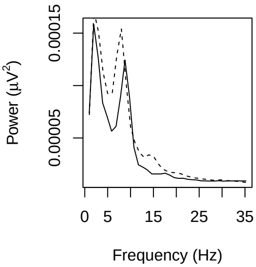
Saving Base Graphics in R

There is a corresponding function to open a graphics device of each format, e.g.

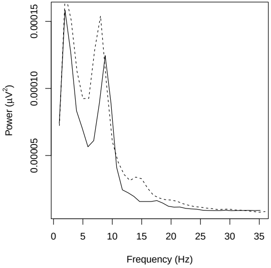
```r
pdf("plot.pdf", width = 5, height = 5)
plot(y ~ x)
dev.off()
```

Setting the width and height controls the aspect ratio - text has fixed size

3in x 3in



5in x 5in



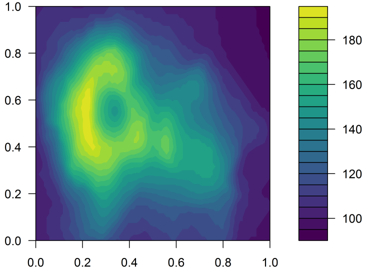
Use ‘Export‘ function in RStudio to find out size of current plot window.

Saving Raster Graphics in R

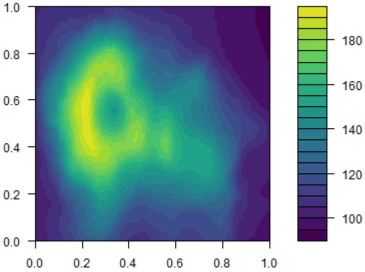
For raster graphics, set the resolution to 600 ppi (pixels per inch), e.g.

```r
png("plot.png", width = 5, height = 5, units = "in", res = 600)
plot(y ~ x)
dev.off()
```

600dpi



72dpi



300 ppi is generally fine for self-printing. The default is 72 ppi, which is suitable for on-screen viewing.

Tables

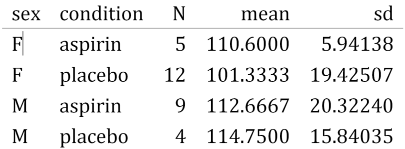
Tables produced using ‘kable‘ already follow good design principles

- Text columns are left-aligned
- Numeric columns are right-aligned
- Column headers are aligned with the data
- Columns are the width of the data
- The style is uncluttered: no gridlines, vertical lines (rules), border, coloured backgrounds, or excessive text formatting.

.Rmd → PDF

sex	condition	N	Mean	SD
F	aspirin	5	110.6000	5.94138
F	placebo	12	101.3333	19.42507
M	aspirin	9	112.6667	20.32240
M	placebo	4	114.7500	15.84035

.Rmd → Word



sex	condition	N	mean	sd
F	aspirin	5	110.6000	5.94138
F	placebo	12	101.3333	19.42507
M	aspirin	9	112.6667	20.32240
M	placebo	4	114.7500	15.84035

Simple Summary Tables

Just a little more tidying is required in the case of simple summary tables

```r
aspirin <- aspirin %>%
    mutate(sex = factor(sex, c("F", "M"), c("Female", "Male")),
           condition = factor(condition, c("aspirin", "placebo"),
                              c("Aspirin", "Placebo")))
aspirin %>%
    group_by(sex, condition) %>%
    summarise(N = n(), Mean = mean(value), SD = sd(value)) %>%
    kable(digits = c(rep(0, 3), 1, 2),
          col.names = c("Sex", "Condition", "N", "Mean", "SD"))
```

Sex	Condition	N	Mean	SD
Female	Aspirin	5	110.6	5.94
Female	Placebo	12	101.3	19.43
Male	Aspirin	9	112.7	20.32
Male	Placebo	4	114.8	15.84

Row Groups

A simple way to obtain row groups is to set the duplicated values to empty characters

```r
x <- aspirin %>%
    group_by(sex, condition) %>%
    summarise(N = n(), Mean = mean(value), SD = sd(value))
x2 <- x %>%
    ungroup() %>%
    mutate(sex = replace(as.character(sex), duplicated(sex), ""))
kable(x2, digits = c(rep(0, 3), 1, 2),
      col.names = c("Sex", "Condition", "N", "Mean", "SD"))
```

Sex	Condition	N	Mean	SD
Female	Aspirin	5	110.6	5.94
	Placebo	12	101.3	19.43
Male	Aspirin	9	112.7	20.32
	Placebo	4	114.8	15.84

Complex Tables

For more complex tables there are two main tasks

- producing and collating the summaries in a structured form
- styling the table

Unfortunately the second task is dependent on the output format - markdown will only take us so far.

There are *many* R packages to help create tables (this summary lists 27: `https://github.com/ropenscilabs/packagemetrics`). We will focus on two packages suited to publication-quality output

kableExtra producing LaTeX (for PDF) or HTML

htmlTable producing HTML suitable for copy-paste to Word

Alternative Row Groups

```
library(kableExtra)
kable(select(x, -sex), format = "latex", booktabs = TRUE,
      col.names = c("", "N", "Mean", "SD")) %>%
    kable_styling() %>%
    group_rows("Female", 1, 2) %>%
    group_rows("Male", 3, 4)
```

	N	Mean	SD
Female			
Aspirin	5	110.6000	5.94138
Placebo	12	101.3333	19.42507
Male			
Aspirin	9	112.6667	20.32240
Placebo	4	114.7500	15.84035

Using booktabs

booktabs is a LaTeXpackage required to create nicely laid out tables.

When using `kable` in a markdown document to produce a simple table in PDF, `knitr` (or *Knit*) takes care of loading the **booktabs**.

If we use an alternative function to generate latex code, such as `group_rows`, we must load any required packages ourselves. This can be done in the YAML header

```
---
output: pdf_document
header-includes:
- \usepackage{booktabs}
---
```

or using `usepackage_latex` from **kableExtra**.

As with using `kable`, we must use `results = "asis"` for the code chunk in which the latex code is generated.

Column Groups

First we spread the statistics across columns by sex

```
library(tidyr)
x2 <- x %>%
    gather(statistic, value, N:SD) %>%
    unite(id, sex, statistic) %>%
    spread(id, value) %>%
    select(condition, Female_N, Female_Mean, Female_SD, Male_N,
           everything())
x2


## # A tibble: 2 x 7
##   condition Female_N Female_Mean Female_SD Male_N Male_Mean  Male_SD
## *    <fctr>    <dbl>       <dbl>     <dbl>  <dbl>     <dbl>    <dbl>
## 1   Aspirin        5    110.6000   5.94138      9  112.6667 20.32240
## 2   Placebo       12    101.3333  19.42507      4  114.7500 15.84035
```

Column Groups

Then we use `latex` to specify column groups

```r
kable(x2, format = "latex", booktabs = TRUE,
      col.names = c("", rep(c("N", "Mean", "SD"), 2))) %>%
    kable_styling() %>%
    add_header_above(c(" " = 1, "Female" = 3, "Male" = 3))
```

	Female			Male		
	N	Mean	SD	N	Mean	SD
Aspirin	5	110.6000	5.94138	9	112.6667	20.32240
Placebo	12	101.3333	19.42507	4	114.7500	15.84035

Adding a Total Line

First compute the totals

```
tot <- aspirin %>%
    group_by(sex) %>%
    summarise(N = n(), Mean = mean(value), SD = sd(value))
tot <- c(sex = "Total", tot[1, 2:4], tot[2, 2:4])
names(tot) <- colnames(x2)
x2 <- bind_rows(x2, tot)
x2
```

```
## # A tibble: 3 x 7
##   condition Female_N Female_Mean Female_SD Male_N Male_Mean  Male_SD
##       <chr>    <dbl>       <dbl>     <dbl>  <dbl>     <dbl>    <dbl>
## 1   Aspirin        5    110.6000   5.94138      9  112.6667 20.32240
## 2   Placebo       12    101.3333  19.42507      4  114.7500 15.84035
## 3     Total       17    104.0588  16.94650     13  113.3077 18.41369
```

Modifying the LaTex Code

kableExtra does not have a option to add a midrule in the table.

To adapt the LaTeX code we can capture the print output as a character string and add a line

```
out <- capture.output(
    kable(x2, format = "latex", booktabs = TRUE,
          col.names = c("", rep(c("N", "Mean", "SD"), 2))) %>%
        kable_styling() %>%
        add_header_above(c(" " = 1, "Female" = 3, "Male" = 3))
)
n <- length(out)
out <- c(out[1:(n - 4)], "\\midrule", out[(n - 3):n])
```

Modifying the LaTex Code

The following concatenates the character strings with new lines

```r
cat(out, sep = "\n")
```

	Female			Male		
	N	Mean	SD	N	Mean	SD
Aspirin	5	110.6000	5.94138	9	112.6667	20.32240
Placebo	12	101.3333	19.42507	4	114.7500	15.84035
Total	17	104.0588	16.94650	13	113.3077	18.41369

Models Example

```r
library(texreg)
texreg(list(model1, model2), booktabs = TRUE, dcolumn = TRUE,
       omit.coef = "(Intercept)", use.packages = FALSE,
       scalebox = 0.8)
```

	Model 1	Model 2
x	0.09	
	(0.20)	
groupB		-4.09^{***}
		(0.52)
groupA:x		0.83^{***}
		(0.21)
groupB:x		1.01^{***}
		(0.21)
R^2	0.00	0.59
Adj. R^2	-0.01	0.58
Num. obs.	100	100
RMSE	2.21	1.43

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$