

# Reproducible research and R authoring with markdown and knitr

*Laurent Gatto*

*14 June 2015*

## Contents

---

- [1 Introduction](#)
- [2 R Markdown](#)
  - [2.1 Installation](#)
  - [2.2 Markdown basics](#)
  - [2.3 File header](#)
  - [2.4 From Rmd to html \(or pdf\)](#)
  - [2.5 More markdown syntax](#)
  - [2.6 R code chunks](#)
- [3 Next steps](#)
- [4 References](#)
- [5 Session information](#)

## 1 Introduction

---

### Objectives

- Understand the concept of dynamic documents are reproducible research
- Learn R markdown basics
- Produce a simple vignette

This content is adapted from the [RStudio R Markdown - Dynamic Documents of R, Markdown basics](#) and [R code chunks](#) tutorials.

This session introduces tools to author documents that include dynamically generated analysis results (tables, figures, ...) produced with R. Bringing data, results and their interpretation together in a single, coherent document is invaluable to keep track of long and complex analyses, assure reproducibility of the pipeline and the final report (any updates at the data or analysis level are propagated at the report level) and to comprehensively communicate these results to collaborators. A popular solution for this is *literate programming*, a technique and set of tools that permit to

1. Write text and code within a single document. Here we will use the simple *markdown* syntax and include R code chunks; such documents are denoted *R markdown* documents and have the Rmd extension. More on this in the next section.

2. Extract and execute the code: this is called *tangling*.
3. Replace the code chunks with their output into the original document: this is called *weaving*.
4. Render the document into a final, easily read format such as pdf or html.

Steps 2 to 4 can be executed individually or automated into a single command such as `knitr::knit2html` (i.e. function `knit2html` from the package `knitr`) or `rmarkdown::render`, or using the RStudio editor.

Other types of document and frameworks that combine a programming and authoring languages are Sweave files (with Rnw extension, that combine LaTeX and R), [Jupyter/IPython](#) for python, R and other languages, [orgmode](#) ...

## 2 R Markdown

---

R Markdown is an authoring format that enables easy creation of dynamic documents, presentations, and reports from R. It combines the core syntax of [markdown](#) (an easy-to-write plain text format) with embedded R code chunks. R Markdown documents are fully *reproducible* (they can be automatically regenerated whenever underlying R code or data changes).

This document describes R Markdown v2 based on [knitr](#) and [pandoc](#), the workhorse that converts markdown to html and many other formats. We will focus the generation of reports such this document in html and pdf, although other formats and type of documents are available.

Note that PDF output requires a full installation of TeX and that pandoc is a third party application that needs to be installed outside of R unless you use RStudio, which bundles all necessary R packages and pandoc.

### Tip

We would also like to warn against using MS Word as output document, as this breaks the support for reproducibility. The final, compiled document should be used for rendering only (which is implicit for html or pdf files); editing should be performed on the original documents, i.e the Rmd file.

### 2.1 Installation

You can install the the required package from CRAN as follows:

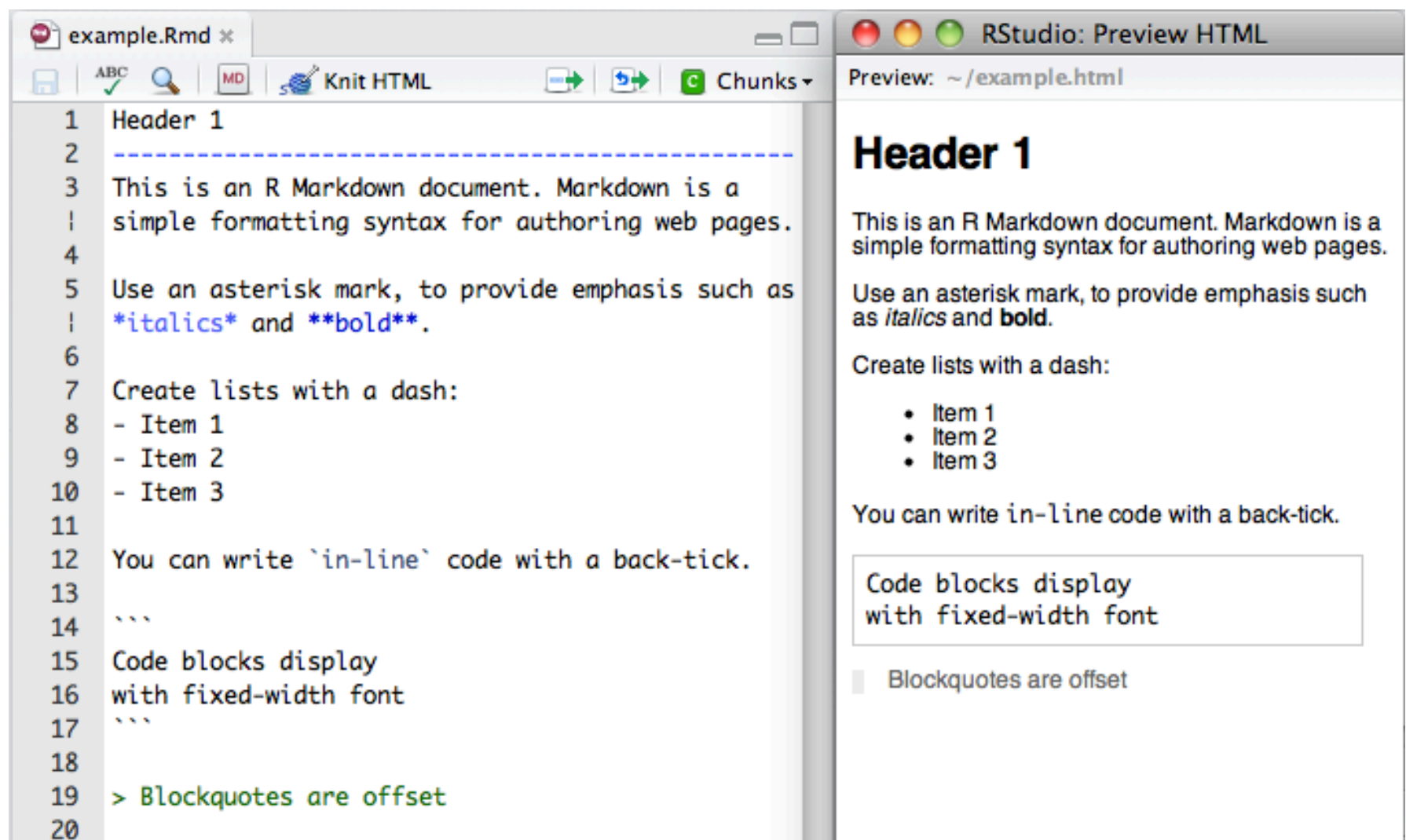
```
install.packages("knitr")
install.packages("rmarkdown")
```

These packages are pre-installed with RStudio.

## 2.2 Markdown basics

The figure below, taken from [the RStudio markdown \(v2\) tutorial](#) illustrates basic markdown syntax and its output using RStudio.

- Section headers can be defined using ===== or ----- (level 1 and 2 respectively) or one or multiple # (for level 1, 2, ... respectively).
- Italic and bold fonts are defined using one to two \* around the text.
- Bullet lists items start with a -.
- In-line code and verbatim expression are surrounded by back ticks `.
- Code blocks start and end with three back ticks.
- Starting a line with > offsets the text.



## 2.3 File header

R Markdown version 2 uses an *optional* header to define, among other things, the title, author and output formats of the R Markdown document. Below, we want to use html as final format; replace with pdf\_document to produce a pdf report.

```
---
title: "Title comes here"
author: "Your name"
date: "12 June 2015"
output: html_document
---
```

## 2.4 From Rmd to html (or pdf)

If you are using RStudio, the simplest way to generate your final output is to open your Rmd file and click the Knit HTML (or Knit PDF, ...) button.

From R, you can use the `knitr::knit2html` or `rmarkdown::render` functions and give the Rmd source file as input.

1. Both options will first use the `knitr::knit` function to *weave* the document and generate the markdown md file that includes the code outputs.
2. The rendering of the final output document will be done using `markdown::markdownToHTML` (in case of `knitr::knit2html`), or the more recent `rmarkdown::render`.

```
library("knitr")
knit2html("my_rr_document.Rmd")
```

```
library("rmarkdown")
render("my_rr_document.Rmd") ## default output is html
render("my_rr_document.Rmd", output_format = "html_document")
```

For pdf outputs using `knitr`

```
knit2pdf("my_rr_document.Rmd")
```

or `rmarkdown`

```
library("rmarkdown")
render("my_rr_document.Rmd", output_format = "pdf_document")
```

And, to render all output formats defined in the header

```
render("my_rr_document.Rmd", output_format = "all")
```

**Exercise:** Experiment with R markdown and the features described so far. To create your starting document, create a new R Markdown file using the RStudio menu or copy/paste the template below.

```
---  
title: "Title comes here"  
author: "Your name"  
date: "12 June 2015"  
output: html_document  
---
```

This is an `R` Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using `R` Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded `R` code chunks within the document. You can embed an `R` code chunk like this:

## 2.5 More markdown syntax

### 2.5.1 Emphasis

You can use `*` or `_` to format *italic* and **bold** text.

```
*italic*    **bold**
```

```
_italic_    __bold__
```

### 2.5.2 Headers

```
## Header 2
```

```
### Header 3
```

### 2.5.3 Lists

Unordered List:

```
* Item 1
* Item 2
  + Item 2a
  + Item 2b
```

Ordered List:

```
1. Item 1
2. Item 2
3. Item 3
  + Item 3a
  + Item 3b
```

### 2.5.4 Links

To use links, enclose the link text in `[]` and the the actual link in `()`:  
`[my link](http://linkurl.com)` or use a plain http address:

```
http://example.com
```

```
[linked phrase](http://example.com)
```

## 2.5.5 Images

To add a static figure to the document, use the link syntax and precede it by `!`:  
`![image text](./fig/myfig.png)`.

Image source can be on-line or local files.

```
![alt text](http://example.com/logo.png)
```

```
![alt text](figures/img.png)
```

## 2.5.6 Block quotes

A friend once said:

It's always better to give than to receive.

A friend once said:

> It's always better to give than to receive.

## 2.5.7 Plain code

Plain code blocks are displayed in a fixed-width font but not evaluated (see below for evaluation of code blocks), use 3 back ticks (see figure above)

```
This text is displayed verbatim / preformatted
```

We can also define in-line code using single back ticks.

```
we can also define `in-line` code using single back ticks.
```

## 2.5.8 Horizontal Rule / Page Break

Three or more asterisks or dashes:

```
*****
```

```
-----
```

## 2.5.9 Tables

There is a simple markdown syntax to produce adequately formatted tables:

First Header	Second Header
Content Cell	Content Cell
Content Cell	Content Cell
Content Cell	Content Cell

which is produced with

First Header	Second Header
Content Cell	Content Cell
Content Cell	Content Cell
Content Cell	Content Cell

### 2.5.10 Embedding Equations

You can embed LaTeX or MathML equations in R Markdown files using the following syntax:

- `$equation$` for inline equations (note there must not be white space adjacent to the `$` delimiters)
- `$$ equation $$` for display equations
- `<math>...</math>` for MathML equations.

For example:

```
1
2 The Arithmetic mean is equal to  $\frac{1}{n} \sum_{i=1}^n x_i$ , or
3 the summation of n numbers divided by n.
```

### 2.5.11 Super- and subscripts

H<sub>2</sub>O is a liquid. 2<sup>10</sup> is 1024.

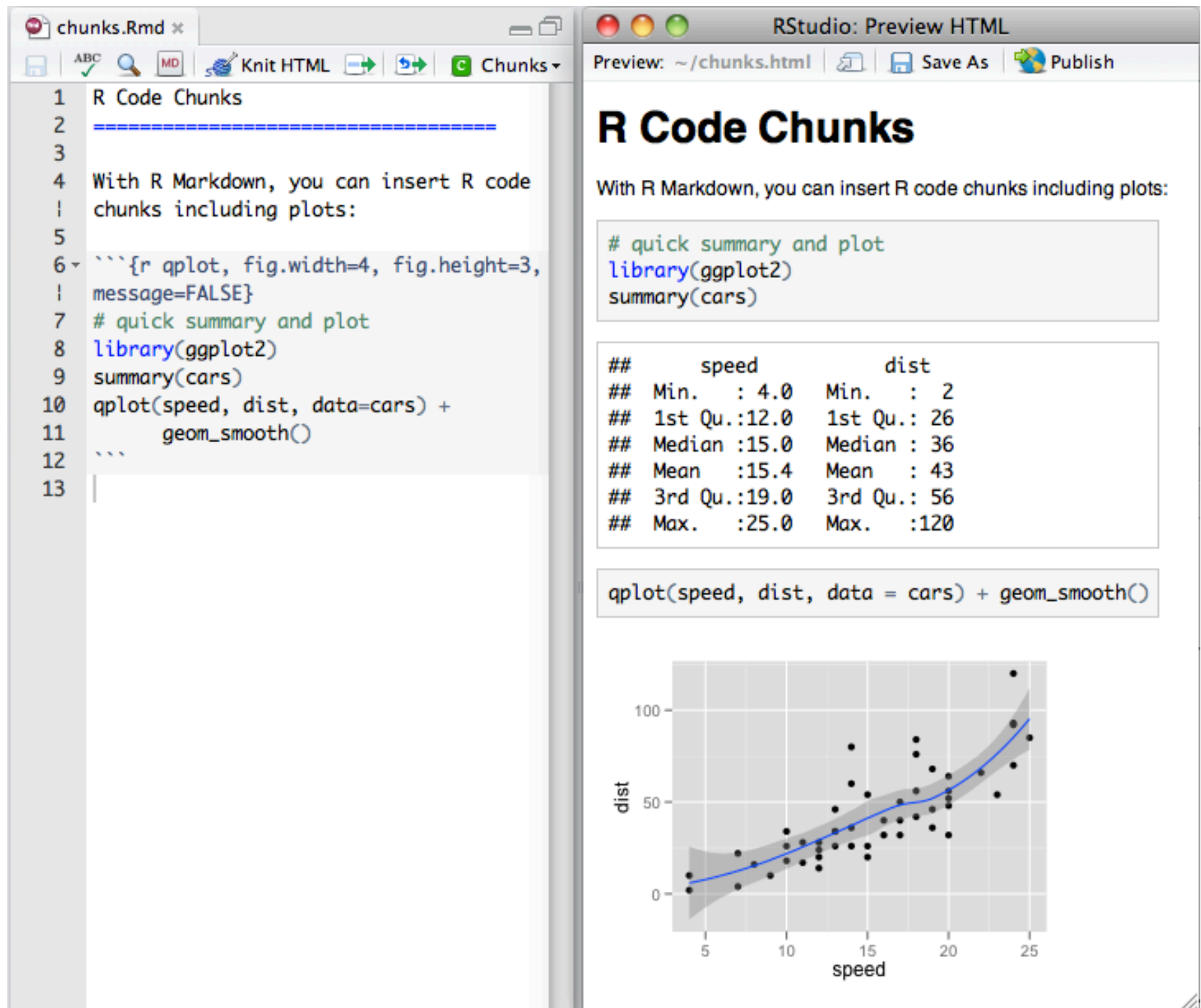
```
H~2~O is a liquid.  2^10^ is 1024.
```



**Exercise:** Complement you Rmd file with some new syntax elements.

## 2.6 R code chunks

To include R code in the R markdown file, the native code chunk syntax is augmented with code chunk tags inside `{r, ...}`, as illustrated below:



The screenshot displays the RStudio interface with two main panes. The left pane shows the source editor for a file named 'chunks.Rmd'. It contains an R code chunk defined by three backticks and the tag `{r}`. The code inside the chunk includes a comment, a `qplot` function call with specific figure dimensions, a `message=FALSE` option, and a `ggplot2` plot of 'dist' vs 'speed' with a smoothed line. The right pane shows the 'Preview HTML' window, which renders the markdown content. It includes the title 'R Code Chunks', a paragraph about inserting R code chunks, the R code from the chunk (highlighted in a light gray box), the output of the `summary(cars)` function (displayed as a table in a light gray box), the `qplot` code (also in a light gray box), and the resulting scatter plot with a smoothed line.

```
1 R Code Chunks
2 =====
3
4 With R Markdown, you can insert R code
5 chunks including plots:
6 ```{r qplot, fig.width=4, fig.height=3,
7   message=FALSE}
8 # quick summary and plot
9 library(ggplot2)
10 summary(cars)
11 qplot(speed, dist, data=cars) +
12   geom_smooth()
13 ```
```

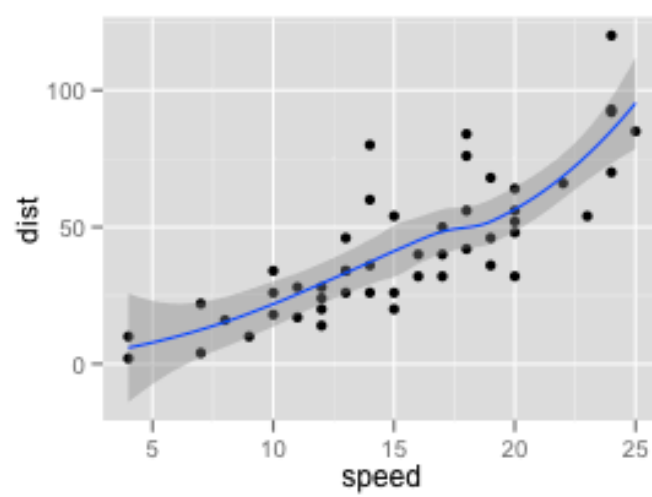
**R Code Chunks**

With R Markdown, you can insert R code chunks including plots:

```
# quick summary and plot
library(ggplot2)
summary(cars)
```

##	speed	dist
##	Min. : 4.0	Min. : 2
##	1st Qu.: 12.0	1st Qu.: 26
##	Median : 15.0	Median : 36
##	Mean : 15.4	Mean : 43
##	3rd Qu.: 19.0	3rd Qu.: 56
##	Max. : 25.0	Max. : 120

```
qplot(speed, dist, data = cars) + geom_smooth()
```



The plot shows 'dist' on the y-axis (ranging from 0 to 100) and 'speed' on the x-axis (ranging from 5 to 25). It displays a scatter of black points with a blue smoothed line and a gray shaded confidence interval.

The following code chunk options are available:

- `{r chunkname}` the first unnamed string is used to name the code chunk; useful for following the code execution and debugging.
- `{r, eval=TRUE}` by default the code in the chunk is executed. Alternatively, set `eval=FALSE`.
- `{r, echo=TRUE}` by default, the code is displayed before the output. Use `echo=FALSE` to hide the code chunk content.
- Control if messages, warnings or errors are to be displayed with `{r, message=TRUE, warning=TRUE, error=TRUE}` or `FALSE`.

- Figure dimensions can be controlled with `fig.height` and `fig.width`.
- To avoid wasting time in repeating long calculations over and over again, it is possible to cache specific code chunks specifying `cache=TRUE` in the chunk header.
- To execute in-line code, use ``r 1+1`` (no space in front of the `r`, though).

Tables can easily be printed inside an code chunk. Below, we explicitly create and use a `data.frame`.

```
dfr <- data.frame(name = c("John", "David", "Caroline", "Igor"),
                  id = c(123, 234, 321, 231),
                  gender = c("M", "M", "F", "M"))

dfr
```

```
##      name  id gender
## 1   John 123      M
## 2  David 234      M
## 3 Caroline 321      F
## 4   Igor 231      M
```

Tables produced in R as data frames or matrices can be rendered with the helper function `knitr::kable` and are then displayed accordingly.

```
library("knitr")
kable(dfr)
```

<b>name</b>	<b>id</b>	<b>gender</b>
John	123	M
David	234	M
Caroline	321	F
Igor	231	M

**Exercise:** Using the `iris` data set, create a reproducible report that documents the data (dimensions, summary statistics, ...) and provides a set of visualisations (a PCA plot, pairs, ...). To conclude your report, add a *Session information* section with the output of `sessionInfo()`.

## 3 Next steps

---

- Publishing your R markdown documents on the web with [R Pubs](#).
- Using R markdown to create package vignettes.
- Source code and R markdown documents versioning using, for example, [git](#) and [GitHub](#).
- Producing dynamic documents with [shiny](#) (there will be a dedicated lab on shiny).

## 4 References

---

- The [knitr](#) package, including excellent documentation.
- [markdown](#) and [rmarkdown](#) packages
- [R markdown](#) documentation
- [R markdown video](#)

## 5 Session information

---

```
## R version 3.2.0 Patched (2015-04-22 r68234)
## Platform: x86_64-unknown-linux-gnu (64-bit)
## Running under: Ubuntu 14.04.2 LTS
##
## locale:
##  [1] LC_CTYPE=en_GB.UTF-8      LC_NUMERIC=C
##  [3] LC_TIME=en_GB.UTF-8      LC_COLLATE=en_GB.UTF-8
##  [5] LC_MONETARY=en_GB.UTF-8  LC_MESSAGES=en_GB.UTF-8
##  [7] LC_PAPER=en_GB.UTF-8     LC_NAME=C
##  [9] LC_ADDRESS=C             LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats      graphics  grDevices  utils      datasets  base
##
## other attached packages:
## [1] knitr_1.10.5    BiocStyle_1.7.3
##
## loaded via a namespace (and not attached):
##  [1] magrittr_1.5      formatR_1.2      tools_3.2.0      htmltools_0.2.
##  [5] yaml_2.1.13       stringi_0.4-1    rmarkdown_0.6.1  highr_0.5
##  [9] stringr_1.0.0     digest_0.6.8     evaluate_0.7
```