

# Rmarkdown

by Sarah Trausner, Ali Usman & Timo Vornberger

---

Statistics, Visualization and More Using "R"

SS24



# Agenda

- Introduction
- Basics and syntax
- Code examples
- Other output formats (besides documents)
  - Presentations
  - Dashboards
- List of sources

# Why would you want to use Rmarkdown?

Rmarkdown is particularly well-suited for presenting results and analyses, visualizations, reports, ...

# What is Rmarkdown?

- R package
- R markdown files -> source code for rich, reproducible documents
- transform an R markdown file in two ways:
  - Knit
  - Convert
- In practice, authors almost always knit and convert their documents at the same time. (button in the RStudio IDE)

# Knit and Convert

Knit:

Rmarkdown package calls the knitr package

knitr will run each chunk of R code in the document and append the results of the code to the document next to the code chunk

saves time and facilitates reproducible reports (No copy and pastes into the report. Good when the data changes often.)

report contains the code it needs to make its own graphs, tables, etc. (author can automatically update the report by re-knitting)

Convert:

You can convert the file.

Rmarkdown package uses the pandoc program to transform the file into a new format (convert .Rmd file into a HTML, PDF, ... file)

Conversion lets you do your original work in markdown, which is very easy to use. You can include R code to knit, and you can share your document in a variety of formats.

# Rmarkdown reports rely on three frameworks

- markdown for formatting text
- knitr for embedded R code
- YAML for render parameters

# Markdown

- Markdown is a set of conventions for formatting plain text.
- Markdown files are easy to read: conventions of markdown are very simple
- Ex.: we can indicate bold/italic, headers, lists, tables,...

# knitr

- knitr package extends the basic markdown syntax to include chunks of executable R code
- knitr will run the code and add the results to the output file (display just the code, just the results, or both)
- knitr will run the code and append the results to the code chunk
- knitr will provide formatting and syntax highlighting

# YAML (yet another markup language)

- YAML header controls how Rmarkdown renders your .Rmd file
- section of key value pairs surrounded by --- marks
- To create a slideshow:

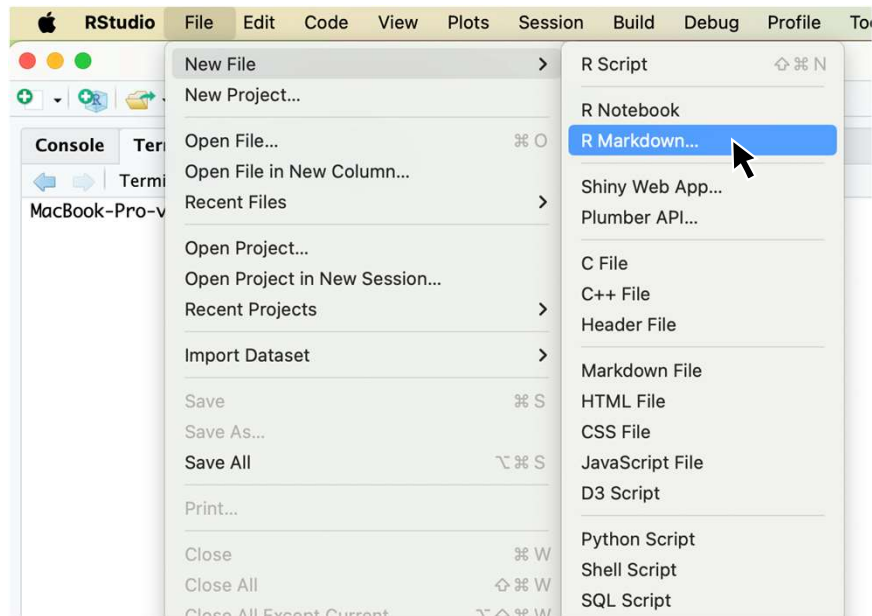
```
output: beamer_presentation
```

- ?outputfile results in all possible YAML options

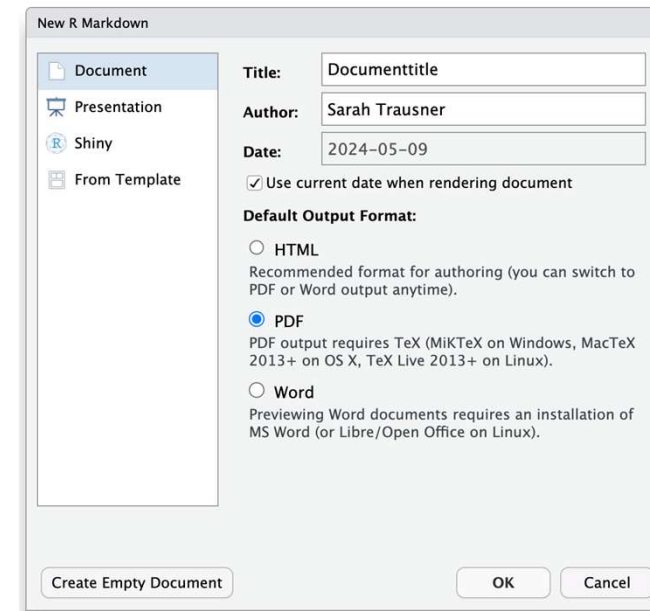
```
1 ---  
2 title: "Documenttitle"  
3 author: "Sarah Trausner"  
4 date: "`r Sys.Date()`"  
5 output: pdf_document  
6 ^ ---
```

# How to get started: (download the Rmarkdown package)

## Open a new Markdown-file



## Select Name, ... and Format



# How to get started:

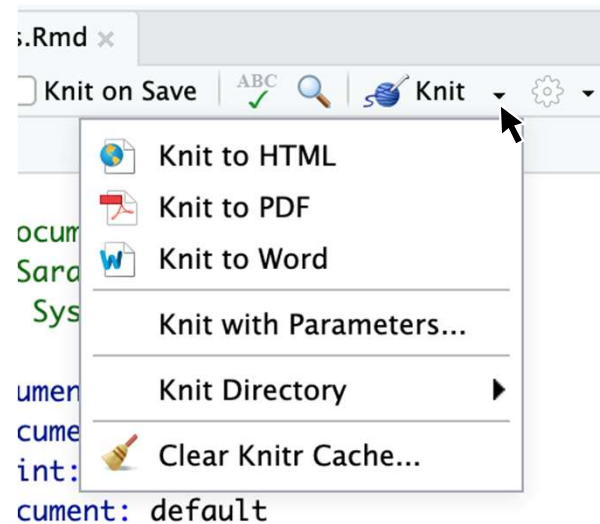
## Standard

```

1 ---
2 title: "Documenttitle"
3 author: "Sarah Trausner"
4 date: "`r Sys.Date()`"
5 output: pdf_document
6 ---
7
8 ```{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 ```
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word
15 documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
16
17 When you click the Knit button a document will be generated that includes both content as well as the output of
18 any embedded R code chunks within the document. You can embed an R code chunk like this:
19
20 ```{r cars}
21 summary(cars)
22 ```
23
24 ## Including Plots
25
26 You can also embed plots, for example:
27
28 ```{r pressure, echo=FALSE}
29 plot(pressure)
30 ```
31
32 Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated
33 the plot.

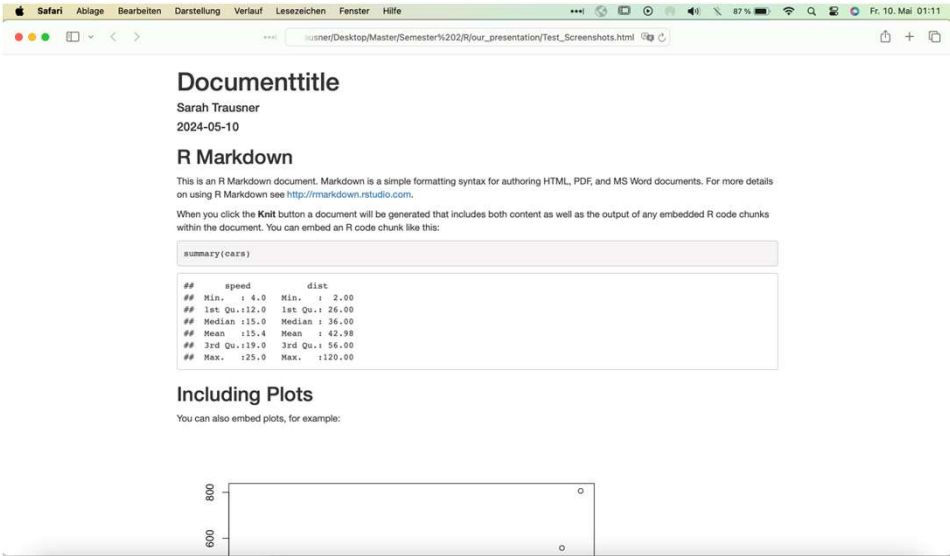
```

## Create Output



# Output

## HTML



The screenshot shows a Safari browser window displaying an HTML document. The document content is as follows:

**Documenttitle**  
Sarah Trausner  
2024-05-10


**R Markdown**  
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:

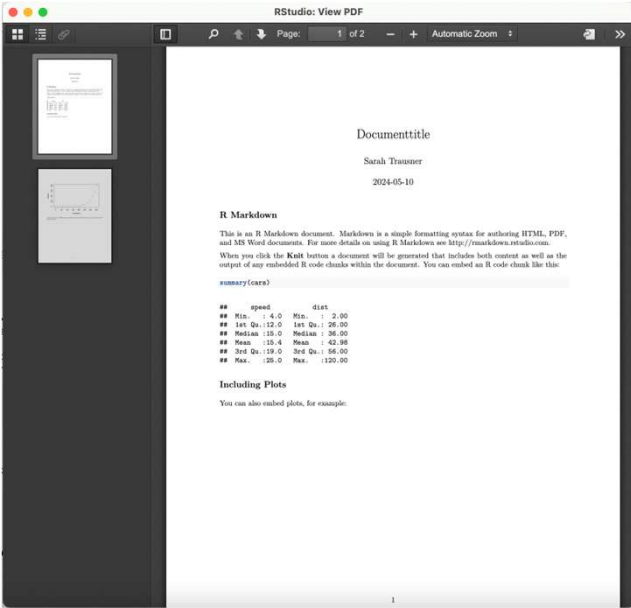
```
summary(cars)
```

```
##      speed      dist
##  Min.   : 4.0   Min.   : 2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

**Including Plots**  
You can also embed plots, for example:



## PDF



The screenshot shows the RStudio interface with a PDF document open. The document content is as follows:

**Documenttitle**  
Sarah Trausner  
2024-05-10

**R Markdown**  
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:

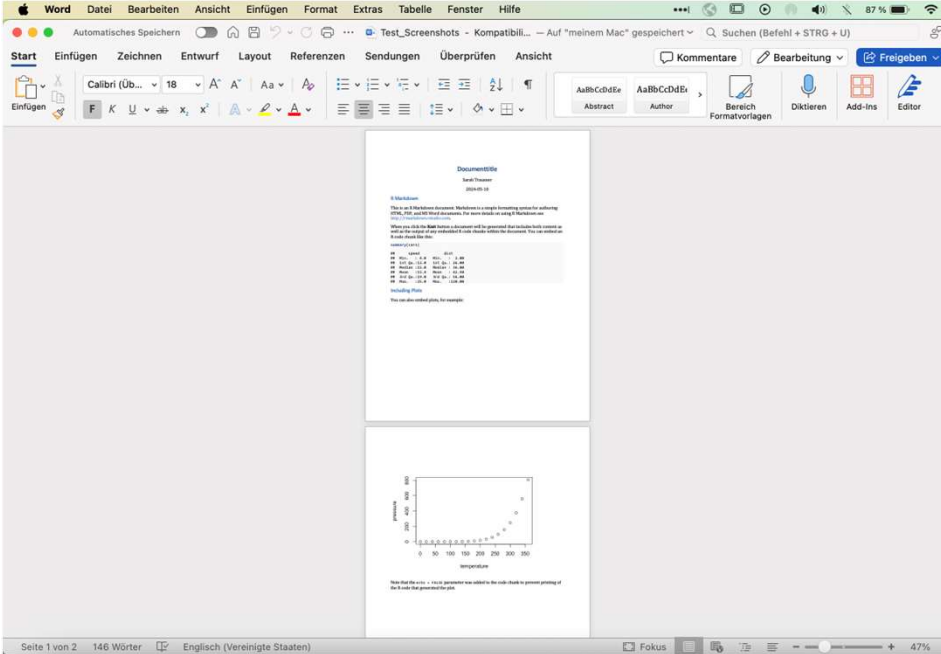
```
summary(cars)
```

```
##      speed      dist
##  Min.   : 4.0   Min.   : 2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

**Including Plots**  
You can also embed plots, for example:

# Output

## Word



The screenshot shows the Microsoft Word interface on a Mac. The document is titled "Test\_Screenshots - Kompatibil..." and is saved on the Mac. The ribbon is set to the "Start" tab, showing the "Font" group with "Calibri (Üb...)" font and size "18". The "Paragraph" group shows "Aa" and "Aa" options. The "Styles" group shows "Abstract" and "Author" styles. The "Tools" group shows "Bereich Formatvorlagen", "Diktieren", "Add-Ins", and "Editor".

The document content includes a "DocumentTitle" section with "Sarah Prosser" and "2020-05-12". Below this is a "Kurzbeschreibung" section with a paragraph of text and a table. The table has 4 columns: "Year", "Sales", "Profit", and "Growth". The data points are:

Year	Sales	Profit	Growth
2015	100	20	10%
2016	120	25	15%
2017	150	35	20%
2018	180	45	25%
2019	220	60	30%
2020	250	75	35%

Below the table is a chart titled "Including Profit" showing a scatter plot of profit over time. The x-axis is labeled "Temperature" and ranges from 0 to 350. The y-axis is labeled "Profit" and ranges from 0 to 800. The data points show a clear upward trend, indicating that profit increases as temperature increases.

The status bar at the bottom shows "Seite 1 von 2", "146 Wörter", "Englisch (Vereinigte Staaten)", "Fokus", and "47%" zoom.

# Introduction to the Syntax

12 ▾	# Header	Header
13 ▾	## Subheader	Subheader
14 ▾	### Subsubheader	Subsubheader

- Header: create headers by including # before the text
  - Sub headers are created by using multiple #
- Italic and bold text: one \* before and after a text-segment produces italic text, with two \* the text becomes bold, if we use three \* the text will be italic and bold

25 With one \* before and after a text-segment we can produce *italic* text. And with two \* the text becomes **bold**. If we use three \* the text will be ***italic and bold***.

With one \* before and after a text-segment we can produce *italic* text. And with two \* the text becomes **bold**. If we use three \* the text will be ***italic and bold***.

# Introduction to the Syntax

- Subscript: use one tilde symbol before and after the characters
- Superscript: use one caret symbol before and after the characters
- We include a Link by putting the Text in [ ] and the Link afterwards in ( ).

```
41 We include a [Link to the markdown guide](https://www.markdownguide.org/getting-started/) by  
   putting the Text in [ ] and the Link afterwards in ( ).
```

```
We include a Link to the markdown guide by putting the Text in [ ] and the Link afterwards in ( ).
```

```
https://www.markdownguide.org/getting-started/
```

```
H~2~O results in H2O
```

```
X^2^ results in X2
```

# Introduction to the Syntax

- Lists: create different points with \* or use 1. , 2. , ... for an ordered list (we can write an unordered List with - , \* or +, to indicate an item)

```
53 This is an ordered List:  
54  
55 1. We create the fist item by writing 1. and then the Text.  
56 2. The second item is created in the same way, with 2. in front.
```

This is an ordered List:

1. We create the fist item by writing 1. and then the Text.
2. The second item is created in the same way, with 2. in front.

```
43 ▾ ### How to create a List  
44 * text  
45 * text  
46 * text
```

**How to create a List**

- text
- text
- text

# Introduction to the Syntax

- Lists: a subpoint is created by an indent
- Where is the mistake?

```
68 List Title:  
69 1. first item  
70 2. second item
```



```
List Title: 1. first item 2. second item
```

```
60 This is an unordered List:  
61  
62 - The fist item uses a -  
63 - just like the second one  
64 - A subpoint is created by an indent
```

This is an unordered List:

- The fist item uses a -
- just like the second one
  - A subpoint is created by an indent

# Introduction to the Syntax

```
72 Now we want to write a statements but it starts with a number followed by a period:  
73  
74 1. generation did A  
75  
76 We can fix this by using a backslash to escape the period:  
77  
78 1\. generation did A
```

Now we want to write a statements but it starts with a number followed by a period:

1. generation did A

We can fix this by using a backslash to escape the period:

1. generation did A

to force a line- / pagebrake latex-code can be used

ex.: `\pagebreak`

# Introduction to the Syntax

- What if we want to include some more text in a new paragraph in a list but want it to be indented like the list:

```
82 - The first item of the list.  
83 - The second item of the list.  
84  
85     Some more Text. Don't forget to indent it so it appears in harmony with the bullet points.  
86  
87 - The third item of the list.
```

- The first item of the list.
- The second item of the list.  
 Some more Text. Don't forget to indent it so it appears in harmony with the bullet points.
- The third item of the list.

# Introduction to the Syntax

- We can create a task list, by creating a list with added [ ]
- To check a box put a x between the [ ]

```
91 - [ ] Create a list with the \- symbol.  
92 - [ ] Then add [ ].  
93 - [x] To check a box put a x between the [ ].
```

- Create a list with the - symbol.
- Then add [ ].
- To check a box put a x between the [ ].

# Introduction to the Syntax

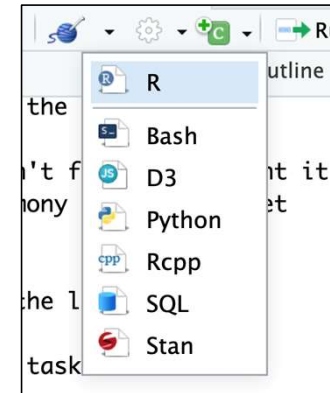
- Tables: (cell widths can vary)

128	Name	Points		-----	-----
129	-----	-----		Name	Points
130	Anna	99		Anna	99
131	Paul	86		Paul	86

135	Name	Points		-----	-----
136	:-----:	:-----:		Name	Points
137	Anna	99		Anna	99
138	Paul	86		Paul	86

- Horizontal Line: with 3 -

12	▼	---
----	---	-----



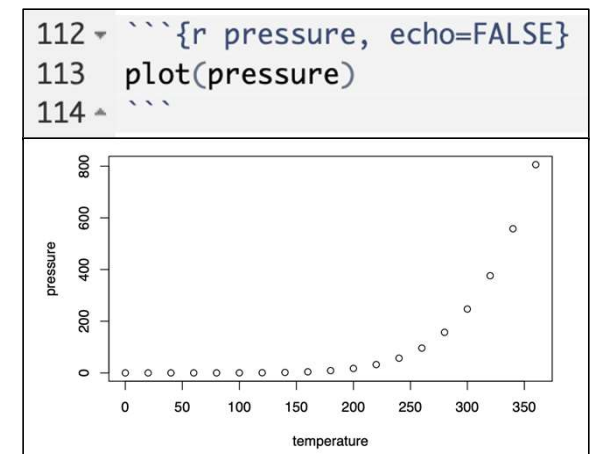
# Introduction to the Syntax

- With the +C button, code (and the output) can be implemented. We start and end a code-section with ```. In the {} we declare that we want to use R code (and the data).

<pre>103 ▾ ```{r} 104   x&lt;-rnorm(20, 0, 1) 105   print(x) 106 ▸ ```</pre>	<pre>x&lt;-rnorm(20, 0, 1) print(x)  ## [1] -0.04594090  0.20107671  0.50475946 -1.58004848  1.62629412  0.84458057 ## [7] -1.37169695 -0.30325613  0.06315377  0.63977600 -0.88752668  0.21172950 ## [13]  0.85377466 -1.09873893 -1.29244367  0.19967067 -0.05385400  0.34963154 ## [19] -1.99888373  2.31719465</pre>
--	--

# Introduction to the Syntax

- By including `echo="FALSE"` we can just see the plot/output and not the code
- Many things can be specified in the `{ }`.
  - Hide warnings: `warning=FALSE`
  - Hide text output: `results='hide'`
  - Given size of output: `fig.width = a, fig.height = b`



# Exercise 1

1. Create a pdf-document with the title: “Übung1 Rmarkdown report”, your name as the autor and the current date. Use “Overview” for the header and “Task description:” as a sub header. Write down the task description and format it like in the provided pdf.

Include a Link to the Rdocumentation-Website about the iris dataset: <https://www.rdocumentation.org/packages/datasets/versions/3.6.2/to/pics/iris> with the text: Click here for the link to the dataset description of iris.

2. Use the iris dataset and display the difference in petal-length / petal-width between the species. Don't include the code. Use ggplot2 to crate a scatterplot (Petal.Length compared to Petal.Width) and two boxplots (Petal.Length / Petal.Width per species). [Look at the provided pdf.]

3. Resize the 3 plots (fig.width = 10, fig.height = 4) and display the first few lines of the dataset on a second page ( use: head() ). Include the code. [Look at the provided pdf.]

## Rcode chunks in RMarkdown



- Rcode Chunks is very powerful code structure.
- You can produce text outputs , graphs ,charts ,Dashboards etc...
- By using code chunks you have control over these outputs
- As Example you can hide ,show something or can set height width etc..
- These options are Separated by the comma in R chunks Curly braces

# Rcode chunks in RMarkdown

- Rcode Chunks is very powerful code structure.
- You can produce text outputs , graphs ,charts ,Dashboards etc...
- By using code chunks you have control over these outputs
- As Example you can hide ,show something or can set height width etc..
- These options are Separated by the comma in R chunks Curly braces

# Markdown Code Chunks Syntax

- As example for Code

```
```{r, chunk-label, results='hide', fig.height=4}
```

```
```{r}  
# execute code if the date is later than a specified day  
do_it = Sys.Date() > '2018-02-14'  
```  
  
```{r, eval=do_it}  
x = rnorm(100)  
```
```

# Use variables in chunk options

- Usually chunk options take constant values
- But it is also possible to take value from arbitrary R expressions
- For example, you can define a variable like `width` in the beginning of a document, and use it later in other code chunks

```
```{r}
my_width <- 7
...

```{r, fig.width=my_width}
plot(cars)
...`
```

# Use of if-else statement in a chunk option

```
```{r}
fig_small <- FALSE # change to TRUE for larger figures
width_small <- 4
width_large <- 8
...

```{r, fig.width=if (fig_small) width_small else width_large}
plot(cars)
...
```
```

# Lazy-loaded OR Cache large objects

- When the chunk option `cache = true`, cached objects will be lazy-loaded into the R session
- It is use full when you want to use large subset of data from the database but wanna to use subset of this data

```
```{r, read-data, cache=TRUE}  
full <- read.csv("HUGE.csv")  
rows <- subset(full, price > 100)  
# next we only use `rows`  
...  
  
```{r}  
plot(rows)  
...  
```
```

# Hide code, text output, messages, or plots

- By default it display all the possible outputs, by doing customization you can hide them individually one by one like so

*Hide source code:*

```
```{r, echo=FALSE}  
1 + 1  
```
```

# Similarly for other options

Hide text output (you can also use `results = FALSE`):

```
```{r, results='hide'}  
print("You will not see the text output.")  
```
```

Hide messages:

```
```{r, message=FALSE}  
message("You will not see the message.")  
```
```

Hide warning messages:

```
```{r, warning=FALSE}  
# this will generate a warning but it will be suppressed  
1:2 + 1:3  
```
```

Hide plots:

```
```{r, fig.show='hide'}  
plot(cars)  
```
```

# Source Code Reformat

- If you set the tidy option true then it source code will be reformatted accordingly
- For this function tidy\_source() is mostly used
- If you wants a list of arguments then you can use tidy.opts

```
tidy({r, tidy=TRUE, tidy.opts=list(arrow=TRUE, indent=2)})  
  
1+           1  
x=1:10  
if(TRUE){  
print('Hello world!')  
}  
...
```

## R Markdown Inline code

- It allows you to insert code into your documents to dynamically update portion of your text

```
```${r}  
x <- 5 radius of a circle  
```
```

```
r pi * x^2
```

- You just give the reference of r and use its value to make dynamic Calculation

# Code Languages

Kintar can execute code in many different available languages

- Python
- SQL
- Bash
- Rcpp
- More...

If you want to execute the code in other languages what you need is to simply replace `r` with the language name for example `python`

```
## Bash
```

```
```{bash}  
ls *.Rmd  
```
```

```
## Python
```

```
```{python}  
x = 'hello, python world!'  
print(x.split(' '))  
```
```

But the chunks options like `echo` results etc will be remain valid

# Rmarkdown Cheat Sheet

- Rmarkdown Cheat Sheet is already uploded you can take help from there

## rmarkdown :: CHEAT SHEET

### What is rmarkdown?



**.Rmd files** - Develop your code and ideas side-by-side in a single document. Run code as individual chunks or as an entire document.

**Dynamic Documents** - Knit together plots, tables, and results with narrative text. Render to a variety of formats like HTML, PDF, MS Word, or MS Powerpoint.

**Reproducible Research** - Upload, link to, or attach your report to share. Anyone can read or run your code to reproduce your work.

### Workflow

1. Open a new **.Rmd file** in the RStudio IDE by going to **File > New File > R Markdown**.
2. **Embed code** in chunks. Run code by line, by chunk, or all at once.
3. **Write text** and add tables, figures, images, and

### SOURCE EDITOR

The screenshot shows the RStudio Source Editor with the following callouts:

- 1. New File**: Points to the 'File' menu.
- 2. Embed Code**: Points to the 'Run' button in the toolbar.
- 3. Write Text**: Points to the text area of an R Markdown chunk.
- 4. Set Output Format(s) and Options**: Points to the 'Output Format' dropdown menu.
- 5. Save and Render**: Points to the 'Knit' button in the toolbar.
- 6. Share**: Points to the 'Share' button in the toolbar.

The code in the editor includes:

```

1 ---
2 title: "Document Title"
3 author: "Author Name"
4 output:
5   html_document:
6     toc: TRUE
7 ---
8
9 [r setup, inc
10 knitr::opts_chunk$set(echo = TRUE)
11
12 ## R Markdown
13 This is an R Markdown document.
14 Markdown is a simple formatting
15 syntax for authoring HTML, PDF,
16 and MS Word documents.
17
18 ```{r cars}
19 summary(cars)
20 ```
21
22 ---
  
```

### RENDERED OUTPUT

The screenshot shows the rendered HTML output with the following content:

Document Title

Author Name

- R Markdown
- Including Plots

R Markdown

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>.

```
summary(cars)
```

| ## | speed        | dist           |
|----|--------------|----------------|
| ## | Min. : 4.0   | Min. : 2.00    |
| ## | 1st Qu.:12.0 | 1st Qu.: 26.00 |
| ## | Median :15.0 | Median : 36.00 |
| ## | Mean :15.4   | Mean : 42.98   |
| ## | 3rd Qu.:19.0 | 3rd Qu.: 56.00 |
| ## | Max. :25.0   | Max. :120.00   |



# What else can be created with RMarkdown?

Sarah Trausner, Ali Usman & Timo Vornberger

14/05/2024

# Other output formats (besides basic markdown documents)

- Presentations
  - ioslides (built-in; this presentation)
  - slidy (built-in)
  - beamer (built-in)
  - powerpoint (built-in)
  - reveal.js (additional package)
  - xaringan (additional package)
- Dashboards (with packages flexdashboard and shiny)
- Books (with package bookdown)
- Handouts and other specific documents (with additional packages)

# **Presentations**

# Built-in presentation formats

## ioslides

- creates nice simple presentations
- has some unique features (incremental bullets, display modes, ...)
- using a logo, changing the transition speed or backgrounds is really easy
- for more features you need to write your own css-file and add it to the header

# Built-in presentation formats

slidy

- has more flexibility than ioslides
- the font can be edited directly in the header instead of editing a separate css-file
- interesting features like a countdown timer in the footer or a customized footer text
- you can use predefined themes

# Built-in presentation formats

beamer

- produces pdf-files
- latex needs to be installed

# YAML Options

Like on R Markdown HTML documents, the beginning has a YAML part:

```
---  
title: "What else can be created with RMarkdown?"  
author: "Timo Vornberger"  
date: "14/05/2024"  
output:  
  ioslides_presentation:  
    widescreen: true  
    logo: plus_logo.png  
    ...  
---
```

# YAML Options

A selection

- `incremental = TRUE`: slide bullets are rendered incrementally
- `transition = "slower"`: default, slower, faster, 0.5, ..
- `css = styles.css`: includes an css-file for customization
- `self-contained = TRUE`: produces a standalone HTML where file every linked dependency is included
- `slide_level = 2`: defines which header level is considered as slide separator  
...
- By typing `?ioslides_presentation` in your console you can get a list of all options for the output format

# Creating a slide

## Titles and headings

- create an intro slide by using one # sign: # Presentations
- create the a new slide by using two # signs: ## Title
- add subtitles by adding a pipe after your title: ## Title | Subtitle
- create a slide without a title using three asterisks: \*\*\*
- create subheadings by using more # signs:  
### Subheading or #### Subsubheading

# Changing the background color

..by adding `{data-background=#00B9E3}` to the slide title

# Changing the background to an image

..by adding `{data-background="wallpaper.png" data-background-size=cover}`  
to the slide title

# Format the text

- use - to create new bullet points
  - use four spaces and + to create new sub points
- use > - to create incremental bullet points (if not already set in YAML options)
- use *\*text\** to create *italicized* text
- use **\*\*text\*\*** to create **bold** text
- use [link](http://www.google.de) to create a [link](http://www.google.de)

Notes, in case you forget what to say

# Including R: Highlighting Code

```
### <b>
```

```
code
```

```
### </b>
```

```
print_factors <- function(x) {  
  print(paste("The factors of",x,"are:"))  
  for(i in 1:x) {  
    if((x %% i) == 0) {  
      print(i)  
    }  
  }  
}
```

# Including R: R Setup Chunk

After the YAML part you can include a setup chunk, which gets executed at the very beginning. This can be used to load packages which are needed later:

```
{r setup, include = FALSE}  
library(flexdashboard)  
library(ggplot2)
```

# Including R: Code Chunks

No options

When specifying no options for the code chunk, the code and the output are presented on the slide:

```
head(mtcars)
```

```
##           mpg cyl  disp  hp  drat    wt  qsec vs  am  gear  carb
## Mazda RX4      21.0   6  160 110 3.90 2.620 16.46 0  1   4    4
## Mazda RX4 Wag  21.0   6  160 110 3.90 2.875 17.02 0  1   4    4
## Datsun 710     22.8   4  108  93 3.85 2.320 18.61 1  1   4    1
## Hornet 4 Drive  21.4   6  258 110 3.08 3.215 19.44 1  0   3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02 0  0   3    2
## Valiant        18.1   6  225 105 2.76 3.460 20.22 1  0   3    1
```

# Including R: Code Chunks

echo = FALSE

With echo = FALSE the code is not presented, only the output:

```
##           mpg cyl  disp  hp  drat    wt  qsec vs  am  gear  carb
## Mazda RX4      21.0   6  160 110  3.90  2.620 16.46  0  1   4    4
## Mazda RX4 Wag  21.0   6  160 110  3.90  2.875 17.02  0  1   4    4
## Datsun 710     22.8   4  108  93  3.85  2.320 18.61  1  1   4    1
## Hornet 4 Drive  21.4   6  258 110  3.08  3.215 19.44  1  0   3    1
## Hornet Sportabout 18.7   8  360 175  3.15  3.440 17.02  0  0   3    2
## Valiant        18.1   6  225 105  2.76  3.460 20.22  1  0   3    1
```

# Including R: Code Chunks

```
eval = FALSE
```

With `eval = FALSE` we can only show the code without the output:

```
head(mtcars)
```

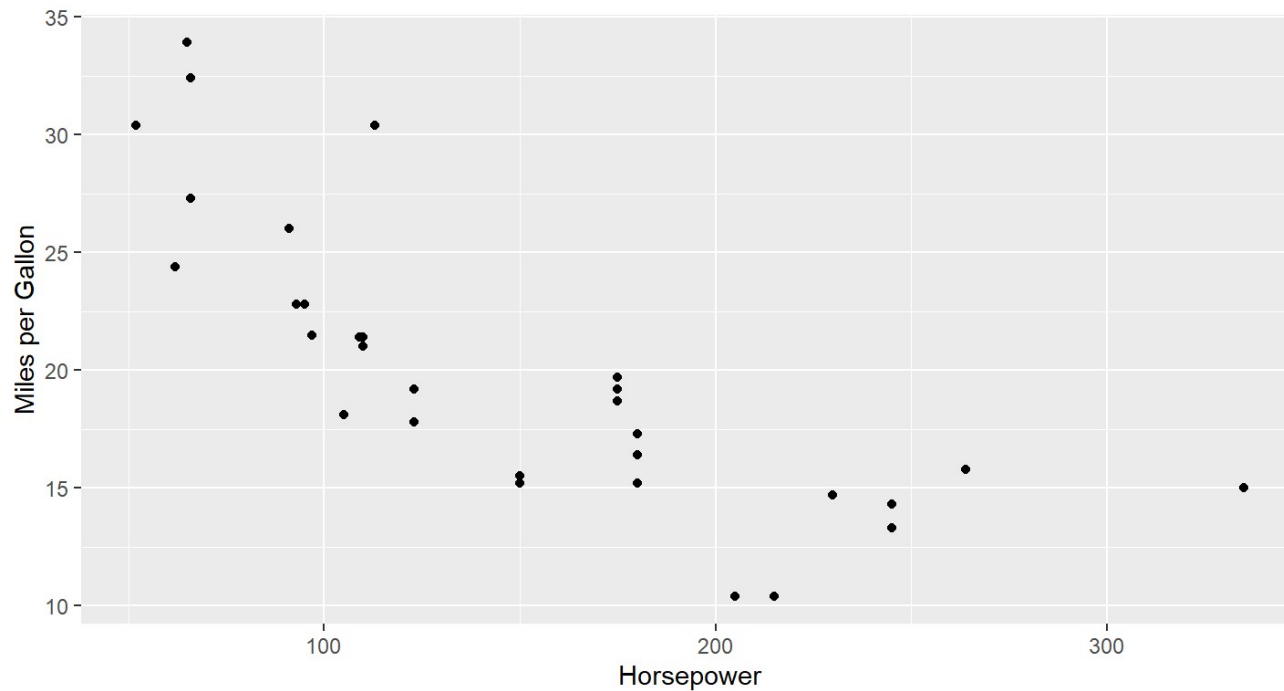
# Including R: Code Chunks

`include = FALSE`

With `include = FALSE` we can write code in our rmarkdown file without including it in the slide:

# Including R: Plots

```
ggplot(mtcars, aes(x = hp, y = mpg)) +  
  geom_point() +  
  labs(x = "Horsepower", y = "Miles per Gallon")
```



# Dashboards

with flexdashboard

# Layout

- first-level section (#): generates a page
- second-level section (##): generates a column or a row
  - in the empty template the columns/rows are created with a long line of minus signs, this works as well
- third-level section (###): generates a box (containing one or more dashboard components)

# YAML options

A selection

- `orientation: columns:` placement of the second-level sections (can be set to rows or columns)
- `vertical_layout: fill:` behavior of the vertical layout, you can choose between “fill” and “scroll”
- `storyboard: TRUE:` uses a different layout scheme, with kind of “big tabs” on the top of the page
- `theme: yeti:` changes the base appearance of the dashboard
- ...

# Attributes on sections

Second-level sections may have attributes

- for columns: set the width of sections with `{data-width = 650}` after the title of the section
- for rows: set the height of sections with `{data-height = 350}` after the title of the section
- in both cases you can use `{.tabset}` after the title so that the sections on the third level will be arranged in tabs

# Value boxes

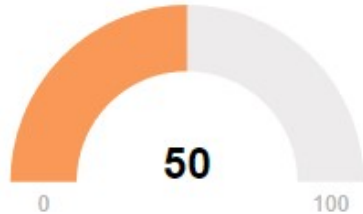
- include one or more simple values in the dashboard



- is used with function `valueBox()` (see detailed syntax in R help)
- for example: `valueBox(99, caption = "Red Balloons", icon = "fa-music")`
  - icons can e.g. be chosen from the icon set of font awesome

# Gauges

- display values on a meter within a specified range



- is used with function `gauge()` (see detailed syntax in R help)
- for example:  

```
gauge(value = 50, min = 0, max = 100, gaugeSectors(success = c(0, 33), warning = c(34, 66), danger = c(67, 100)))
```

# Exercise on flexdashboards

Create a *Fleet Dashboard* about the mtcars dataset with **two pages**, using the theme “flatly”.

The first page *Overview* shows the whole data set in a table.

(you can use package DT with function datatable(), package kableExtra with function kable() or any other nice visualization)

The second page *Details* is divided in **three rows**:

The first row contains two elements:

a valueBox with **the number of cars** including an icon (e.g. “fa-car”),

a gauge with the **average fuel consumption** of all cars (specify a useful total range and useful sectors)

The second row contains **two graphical visualizations** of the dataset

(e.g. a scatterplot of hp/mpg and a histogram of wt)

These visualizations should not be side by side, but arranged as a **tabset**, so you can decide which graph you want to see.

The third row *About the author* contains a **Text box** with your name and has a height of 50.

# Limitations of RMarkdown

## Challenges

- Common problems ([with solutions](#))
- Limitations in complexity and performance
  - large datasets can be resource-intensive
  - not as powerful as proprietary BI software
- Debugging can be challenging
  - especially when errors happen during rendering

# Limitations of RMarkdown

## Alternatives

- Shiny: as alternative for flexdashboards as part of RMarkdown
- Sweave: a tool embedding R code in LaTeX documents to create dynamic reports
- Jupyter Notebooks can also be used with R coding

# Sources

[https://rmarkdown.rstudio.com/articles\\_intro.html](https://rmarkdown.rstudio.com/articles_intro.html)

<https://bookdown.org/yihui/rmarkdown-cookbook/hide-one.html>

<https://bookdown.org/yihui/rmarkdown/language-engines.html>

<https://www.r-bloggers.com/2019/09/mastering-r-presentations/>

<https://bookdown.org/yihui/rmarkdown/presentations.html>

<https://bookdown.org/yihui/rmarkdown/dashboards.html>

<https://pkgs.rstudio.com/flexdashboard/index.html>

**Thank you for your attention!**