Lesson 4 of 5
Visualizing and Writing

Intro to R workshop, LU Skills School

Instructor:

Christopher Swader
LU Sociology (Assoc. Prof) and LMU (Munich, Researcher)

Teaching Assistant: Maximilian Hornung (MS programme in Social Scientific Data Analysis, LU)

Slides adapted from and inspired by:
Irina Vartanova, Institute for Future Studies, Stockholm

Today’s agenda

- Visualizing with ggplot2, plotly, gganimate
- Rmarkdown

- Use the link to download the files we will be using today: https://github.com/ChristopherSwader/R_introduction
- Download the Day 4 folder.

Basics of Visualisation in R’s ggplot2 package

ggplot2: the grammar of graphics

A unified system for variety of graphs

RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help

R course

Go to file/function | Addins | R course

02_Visualise-Exercises.Rmd

Preview | Insert

```
ggplot(data = flfp) +
  geom_point(mapping = aes(x = patr_mean, y = wvs_flfp))

```

Your turn 3

In the chunk below, add colour, size, alpha, and shape aesthetics to your graph. Experiment.

```{r}

ggplot(flfp) +
  geom_point(aes(x = patr_mean, y = wvs_flfp))

```

Your turn 4

Reproduce the plot on the slide by setting colour = "darkblue", shape = 23, fill = "lightblue", size = 3.

```{r}

```

31:1 | Your turn 2 | R Markdown

Console

Search

R Help
Search R Help ^⌥F1

About RStudio

Accessibility

RStudio Docs
RStudio Community Forum

Cheatsheets

Keyboard Shortcuts Help ⌥⇧K
Markdown Quick Reference
Roxygen Quick Reference

Diagnostics

RStudio IDE Cheat Sheet
Data Transformation with dplyr
Data Visualization with ggplot2
List manipulation with purrr
Package Development with devtools
Web Applications with shiny
Interfacing Spark with sparklyr
R Markdown Cheat Sheet
R Markdown Reference Guide

Browse Cheatsheets...

Tutorial | List

nums	num [1:4]
strs	chr [1:4]

Files | Plots | Packages | Help | Viewer

New Folder | Delete | Rename | More

Home > Dropbox > projects > R course > prep > Day2_Visualise

Name	Size	Modified
..		
02_Visualise-Exercises.Rmd	2.6 KB	Oct 3, 2020, 2:5
data-visualization-2.1.pdf	1.9 MB	Oct 2, 2020, 6:5
flfp-aggregated.rds	6.6 KB	Oct 5, 2020, 3:1

geom_functions

Different bases for visualization

(Each requires a mapping argument.)

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

ggplot2

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=1)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = group))
x, y, alpha, color, fill, group, linetype, size

b + geom_rect(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept=0, slope=1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend=lat+1, xend=long+1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5) x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)) x, y, alpha, color, fill, linetype, size, weight

discrete

d <- ggplot(mpg, aes(fl))

d + geom_bar()
x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

continuous x , continuous y

e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size

e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"), x, y, alpha, color, linetype, size

e + geom_smooth(method = lm), x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

f <- ggplot(mpg, aes(class, hwy))

f + geom_col(), x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group

f + geom_violin(scale = "area"), x, y, alpha, color, fill, group, linetype, size, weight

discrete x , discrete y

g <- ggplot(diamonds, aes(cut, color))

g + geom_count(), x, y, alpha, color, fill, shape, size, stroke

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, fill, size

continuous function

i <- ggplot(economics, aes(date, unemploy))

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

j + geom_crossbar(fatten = 2)
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))

k + geom_map(aes(map_id = state), map = map) **+ expand_limits**(x = map$long, y = map$lat), map_id, alpha, color, fill, linetype, size

THREE VARIABLES

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))l <- ggplot(seals, aes(long, lat))

l + geom_contour(aes(z = z))
x, y, z, alpha, colour, group, linetype, size, weight

l + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)
x, y, alpha, fill

l + geom_tile(aes(fill = z)), x, y, alpha, color, fill, linetype, size, width

Graph Gallery!

https://r-graph-gallery.com/

To make a graph

1. Pick a **data** set

[template]

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

2. Choose a **geom** to display cases

3. **Map** aesthetic properties to variables

Histograms and Bar Charts

Histogram

data

Aesthetic mappings

+ before new line

```
ggplot(flfp, aes(patr_values)) +
  geom_histogram()
```

type of layer

The geom (style, shape) of the graph

Bar chart - frequencies

```
ggplot(flfp, aes(denom)) +
  geom_bar()
```

Scatterplots

```
ggplot(data = flfp) +
  geom_point(mapping = aes(x = patr_mean,
                           y = wvs_flfp))
```

data

+ before new line

x variable

type of layer, geom

aesthetic mapping

y variable

Basic Scatterplot

```
ggplot(flfp_agg, aes(x=patr_mean,
y=wvs_flfp)) +

  geom_point()
```

Fix axis labels

```
ggplot(flfp_agg, aes(x=patr_mean,
y=wvs_flfp)) +
  geom_point()+
  labs(y= "Female Lab.Force Part.
(WVS)", x = "Patriarchical Values")
```

Add a regression line
(with confidence interval)

```
ggplot(flfp_agg, aes(x=patr_mean,
y=wvs_flfp)) +

  geom_point()+

    labs(y= "Female Lab.Force Part.
(WVS)", x = "Patriarchical Values")
+

  geom_smooth(method = "lm", se
= T)
```

Add a curvier regression line

```
ggplot(flfp_agg, aes(x=patr_mean,
y=wvs_flfp)) +

  geom_point()+

  labs(y= "Female Lab.Force Part.
(WVS)", x = "Patriarchical Values")
+

  geom_smooth(method = "lm", se
= T, formula = y ~ poly(x, 3))
```

Highlight Region Differences

```
ggplot(flfp_agg, aes(x=patr_mean,
y=wvs_flfp)) +

  geom_point(aes(colour = region))
+

  labs(y= "Female Lab.Force Part.
(WVS)", x = "Patriarchical Values")
+

  geom_smooth(method = "lm", se
= T, formula = y ~ poly(x, 3))
```

We had to add this aesthetic to geom_point! If we would add it to ggplot() on top, it would impact EVERYTHING, including the regression line, thus splitting it.

Change stylistic *theme*

```
ggplot(flfp_agg, aes(x=patr_mean, y=wvs_flfp)) +
  geom_point(aes(colour = region))+
  theme_classic() +
  labs(y= "Female Lab.Force Part. (WVS)",
       x = "Patriarchical Values") +
  geom_smooth(method = "lm", se = T, formula =
y ~ poly(x, 3))
```

theme_classic()

theme_dark()

theme_bw()

Lots of others can be found online. There are also ways to specify all of these manually

Your Turn

- Use ggplot with flfp_agg to make a scatterplot with log_gdp on the x-axis and ilo_flfp on the y-axis. Add an unspecified smoothed line (no method), with a confidence interval. Split by region, and use the theme_light(). Adapt your axis labels. Add a special label to the to a country name of your choosing (*cntry*) using geom_text().

Highlight Region Differences

```
ggplot(flfp_agg, aes(x=log_gdp, y=ilo_flfp)) +
  geom_point(aes(colour = region)) +
  theme_light() +
  labs(y= "Female Lab.Force Part. (ILO)", x = "log GDP") +
  geom_smooth(se = T)+
  geom_text(data=filter(flfp_agg, cntry=="Russia"), aes(label=cntry))
```

Solution is to use a combination of geom_text, filter() to limit the data under consideration, and aes(label=cntry) to add the label(s).

Layering

Using this same method, we can adapt ANYTHING and everything about this plot. e.g. the font, where the label appears in relation to the point. We can connect the two with a line, offset them so they do not overlap. Change the legend...etc etc .etc

Mappings

```
ggplot(flfp_agg) +
   geom_point(aes(x = patr_mean, y = wvs_flfp))
```

Aesthetics

Aesthetics

aesthetic property

variable to map it to

```
ggplot(flfp_agg) + geom_point(aes(x = patr_mean, y = wvs_flfp, color = region))
ggplot(flfp_agg) + geom_point(aes(x = patr_mean, y = wvs_flfp, size = log_gdp))
ggplot(flfp_agg) + geom_point(aes(x = patr_mean, y = wvs_flfp, shape = region))
ggplot(flfp_agg) + geom_point(aes(x = patr_mean, y = wvs_flfp, alpha = log_gdp))
```

color

Legends added automatically

size

shape

alpha

***size* and *alpha* more appropriate for non-categorical variables**

Your turn

- In the next chunk, add colour, size, alpha, and shape aesthetics to your graph from your last individual exercise. Experiment.
- What happens when you map aesthetics to discrete and continuous variables?
- What happens when you use more than one aesthetic?

Female Lab.Force Part. (ILO)
100
75
50
25
7
8
9
10
11
log GDP
Russia
west
s/est europe
latin
east asia
ss africa
centr asia
mena
log_gdp
7
8
9
10
11

Female Lab.Force Part. (ILO)
75
50
25
7
8
9
10
11
log GDP
Russia
log_gdp
11
10
9
8
7
region
west
s/est europe
latin
east asia
ss africa
centr asia
mena

	Discrete	Continuous
Color		
Size		
Shape		

set vs. map

How do you make this plot?

Map variable

Inside of aes(): maps an aesthetic to a variable

```
ggplot(flfp_agg) +
  geom_point(aes(x = patr_mean, y = wvs_flfp, color = region))
```

Set values

```
ggplot(flfp_agg) +
  geom_point(mapping = aes(x = patr_mean, y = wvs_flfp),
                                colour ="blue")
```

Outside of aes():
sets an aesthetic to a value

```
ggplot(flfp_agg) +
geom_point(aes(x = patr_mean,
               y = wvs_flfp,
               colour = region))
```

```
ggplot(flfp_agg) +
 geom_point(aes(x = patr_mean,
                y = wvs_flfp),

                colour ="blue")
```

Your turn

Reproduce the plot by setting:

- colour = "darkblue"
- shape = 23
- fill = "lightblue"
- size = 3

Geoms

How are these plots similar?

Same: x var, y var, data

How are these plots different?

Different: geometric shape/object (geom function)

geoms

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Your turn

- Make the bar-plot of **denom** from the flfp df which you made earlier today, coloured by denom, like below.

```
ggplot(flfp, aes(denom)) +
  geom_bar(aes(fill=denom))+
  theme_classic()
```

```
ggplot(flfp, aes(denom)) +
  geom_bar(aes(color=denom))+
  theme_classic()
```

Your turn

- How to get this graph?

```
ggplot(flfp, aes(x=denom)) +
  geom_bar(aes(fill=region))+
  theme_classic()
```

boxplots

Your turn

- Use the cheatsheet/ google and decide how to replace the scatterplot on the left with a box-plot

```
ggplot(flfp_agg) +
geom_point(aes(x = region, y = wvs_flfp))+
theme_bw()
```

```
ggplot(flfp_agg) +
geom_boxplot(aes(x = region, y = wvs_flfp))+
theme_bw()
```

"Help" pages

To open the documentation for a function, type

```
?geom_histogram
```

?

function name (no parentheses)

geom_freqpoly {ggplot2} R Documentation

Histograms and frequency polygons

Description

Visualise the distribution of a single continuous variable by dividing the x axis into bins and counting the number of observations in each bin. Histograms (`geom_histogram()`) display the counts with bars; frequency polygons (`geom_freqpoly()`) display the counts with lines. Frequency polygons are more suitable when you want to compare the distribution across the levels of a categorical variable.

Usage

```
geom_freqpoly(mapping = NULL, data = NULL, stat = "bin",
  position = "identity", ..., na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE)

geom_histogram(mapping = NULL, data = NULL, stat = "bin",
  position = "stack", ..., binwidth = NULL, bins = NULL,
```

Tips

- **scan** page for relevant info, especially the arguments
- **ignore** things that don't make sense
- **try** out the examples

geom_freqpoly {ggplot2} R Documentation

Histograms and frequency polygons

Description

Visualise the distribution of a single continuous variable by dividing the x axis into bins and counting the number of observations in each bin. Histograms (`geom_histogram()`) display the counts with bars; frequency polygons (`geom_freqpoly()`) display the counts with lines. Frequency polygons are more suitable when you want to compare the distribution across the levels of a categorical variable.

Usage

```
geom_freqpoly(mapping = NULL, data = NULL, stat = "bin",
  position = "identity", ..., na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE)

geom_histogram(mapping = NULL, data = NULL, stat = "bin",
  position = "stack", ..., binwidth = NULL, bins = NULL,
```

Your turn

- Try to predict what this code will do. Then run it.

```
ggplot(flfp_agg) +
  geom_point(aes(x = patr_mean, y = wvs_flfp))+
  geom_smooth(aes(x = patr_mean, y = wvs_flfp))+
  theme_bw()
```

Multiple geoms on the same plot

```
ggplot(flfp_agg) +
geom_point(aes(x = patr_mean, y = wvs_flfp)) +
geom_smooth(aes(x = patr_mean, y = wvs_flfp))
```

Global mapping

Mappings (and data) that appear in ggplot() will apply globally to every layer

```
ggplot(flfp, aes(x = patr_mean, y = wvs_flfp)) +
      geom_point() +
      geom_smooth()
```

Local mapping

Mappings (and data) that appear in a geom_ function will add to or override the global mappings for that layer only

```
ggplot(flfp_agg, aes(x = patr_mean, y = wvs_flfp)) +
  geom_point(aes(colour = region)) +
  geom_smooth()+
  theme_bw()
```

Local mapping

Data can also be set locally by geom

```
ggplot(flfp_agg, aes(x = patr_mean, y = wvs_flfp)) +
  geom_point(aes(colour = region)) +
  geom_smooth(data = filter(flfp_agg, region == "mena"))+
  theme_bw()
```

Different geoms use different aesthetics

```
ggplot(flfp_agg, aes(x = patr_mean, y = wvs_flfp)) +
  geom_point(aes(colour = region)) +
  geom_smooth(aes(linetype = region),se = FALSE, method = "lm")+
  theme_bw()
```

Different geoms use different aesthetics

TWO VARIABLES

continuous x , continuous y

e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)** x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_jitter(height = 2, width = 2**)** x, y, alpha, color, fill, shape, size

e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"**)**, x, y, alpha, color, linetype, size

e + geom_smooth(method = lm**)**, x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)**, x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Your turn

- Make the scatterplot below with **labels** representing country codes. Use the cheatsheet.

Of course, one can also adjust the labels so they do not cover the points.

```
ggplot(flfp, aes(x = patr_mean, y = wvs_flfp)) +
geom_point(aes(colour = region)) +
geom_text(aes(label = code))
```

What else?

Facets

```
ggplot(flfp_agg) +
  geom_point(aes(x = patr_mean, y = wvs_flfp)) +
  facet_wrap(vars(region))+
  theme_bw()
```

Position adjustments

How overlapping objects are arranged

```
ggplot(flfp_agg) +
  geom_bar(aes(x = region,
               fill = oil_gr),
           position = "stack") +
  theme_bw()
```

Default position

```
ggplot(flfp) +
   geom_bar(aes(x = region,
                fill = oil_gr),
                position = "dodge")
```

Dodged position

Scales

Customise colour scales, other mappings

```
ggplot(flfp_agg) +
  geom_point(aes(x = patr_mean, y = wvs_flfp, colour = region)) +
  scale_color_viridis_d()+
  theme_bw()
```

Use the template to make any plot of your choice

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
 stat = <STAT> , position = <POSITION> ) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

required

Not required, sensible defaults supplied

geom_sf()
for maps

James Cheshire,

https://ggplot2.tidyverse.org/reference/index.html

ggplot2 part of the tidyverse

Reference Articles News

Reference

Plot basics

All ggplot2 plots with a call to `ggplot()`, supplying default data and aesthethic mappings, specified by `aes()`. You then add layers, scales, coords and facets with `+`. To save a plot to disk, use `ggsave()`.

`ggplot()`	Create a new ggplot
`aes()`	Construct aesthetic mappings
`` `+`(<gg>) `%+%` ``	Add components to a plot
`ggsave()`	Save a ggplot (or other grid object) with sensible defaults
`qplot()` `quickplot()`	Quick plot

Layer: geoms

A layer combines data, aesthetic mapping, a geom (geometric object), a stat (statistical transformation), and a position

Contents

- Plot basics
- Layer: geoms
- Layer: stats
- Layer: position adjustment
- Layer: annotations
- Aesthetics
- Scales
- Guides: axes and legends
- Facetting
- Facetting: labels
- Coordinate systems
- Themes
- Programming with ggplot2
- Extending ggplot2
- Vector helpers
- Data

Saving graphs

Manually saving plots

Save plots manually with the export menu

Saving plots

ggsave() saves the last plot.
Uses size on screen:

```
ggsave("my-plot.pdf")
ggsave("my-plot.png")
```

Specify size in inches

```
ggsave("my-plot.pdf", width = 6, height = 6)
```

Your Turn

- Use the ggsave() function to save the plot in a pdf file called "01_patr-values-vs-flfp-scatter.pdf".

```
ggsave("01_patr-values-vs-flfp-scatter.pdf")
```

I generally have way better results using ”zoom” and then saving manually compared to this function

We can also make graphics interactive

Plotly demonstration

Your turn

- Run the code below.

```
pl <- ggplot(flfp_agg, aes(x = patr_mean,
                           y = wvs_flfp,
                          colour = region,
                          label = cntry)) +
                           geom_point()

library(plotly)
ggplotly(pl)
```

gganimate demonstration

- Line 291 in Day 4 code

Your turn

- There is a geom_smooth() argument commented out in the code i have given you. What will this line do if added? Remove the pound signs (don’t forget the plus sign above) and re-run the code.
- What is revealed by the slopes here?

rmarkdown

rmarkdown

www.rstudio.com

R Markdown

Plain text file with 3 types of content:

A header surrounded by — — —

Text in markdown

Code chunks surrounded by ```

Markdown language

```
Heading
=======

## Sub-heading

Paragraphs are separated
by a blank line.

Two spaces at the end of a
line
produces a line break.

Text attributes _italic_,
**bold**, `monospace`.

Horizontal rule:

---

Bullet list:

  * apples
  * oranges
  * pears

Numbered list:

  1. lather
  2. rinse
  3. repeat
```

```
<h1>Heading</h1>

<h2>Sub-heading</h2>

<p>Paragraphs are separated
by a blank line.</p>

<p>Two spaces at the end of a line<br
/>
produces a line break.</p>

<p>Text attributes <em>italic</em>,
<strong>bold</strong>,
<code>monospace</code>.</p>

<p>Horizontal rule:</p>

<hr />

<p>Bullet list:</p>

<ul>
<li>apples</li>
<li>oranges</li>
<li>pears</li>
</ul>

<p>Numbered list:</p>

<ol>
<li>lather</li>
```

Heading

Sub-heading

Paragraphs are separated by a blank line.

Two spaces at the end of a line
produces a line break.

Text attributes *italic*, **bold**, `monospace`.

Horizontal rule:

Bullet list:

- apples
- oranges
- pears

Numbered list:

1. lather
2. rinse
3. repeat

An example.

knitr
pandoc
Rmd
md
HTML
PDF
LaTeX
ioslides
slidy, Beamer
Powerpoint
Microsoft
Word

Logistics

1 Knitr runs the document in a fresh R session, which means you need to load the libraries that the document uses *in the document*

CC BY-SA RStudio

Logistics

1 Knitr runs the document in a fresh R session, which means you need to load the libraries that the document uses *in the document*

2 Objects made in one code chunk will be available to code in later code chunks.

CC BY-SA RStudio

knitr is multi-lingual!

- ☑ SAS
- ☑ python
- ☑ more

CC BY-SA RStudio

python

To embed non R code, change the chunk label from r to the language to use.

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

Some python code:

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

```
## hello, python world!
## ['hello,', 'python', 'world!']
```

CC BY-SA RStudio

Markdown

R-Notebook.Rmd

Preview Insert

```
---
title: "R Notebook"
output: html_notebook
---

Text written in **markdown**

```{r}
# code written in R
(x <- rnorm(7))
```

[1] -1.2 1.0 -0.5 0.9 -0.6 -1.1 -1.5

Text written in _markdown_

```{r}
# code written in R
```
```

16:20 Chunk 2 R Markdown

Text in markdown

Headers

Use # to create headers.
Multiple #'s create lower level headers.

```
# Header 1
## Header 2
### Header 3
#### Header 4
##### Header 5
###### Header 6
```

Header 1
Header 2
Header 3
Header 4
Header 5
Header 6

Text

Text is rendered as plain text. Surround text with _, **, or ` to format it.

Add two spaces
at the end of a line
to start a new line

```
Text
_italics_
**bold**
`code`
```

Text

italics

bold

`code`

Lists

Use asterisks to make bullet points.
Use numbers to make numbered lists.

```
Bullets

* bullet 1
* bullet 2

Numbered list

1. item 1
2. item 2
```

Bullets

- bullet 1
- bullet 2

Numbered list

1. item 1
2. item 2

Hyperlinks

Use brackets to denote a link.
Place the URL in parentheses.

```
This is a
[link](www.git.com).
```

This is a link.

Images

Use a link preceded by an ! to insert an image.

The link text should be a URL (if the image is hosted online), or a file path (if the image is saved as a file)

```
![](images/1.png)
The RStudio logo.
```

Markdown

Dictionary of formatting cues.

IDE Reference

Go to **Help > Markdown Quick Reference**

Inline code

Place code in a sentence with **`r <code>`**. R Markdown will replace the code with its results.

```
Today is
`r Sys.Date()`.
```

Today is 2015-04-16.

CC BY-SA RStudio

Inline code

Code whose results are inserted into the text.

```
Today is `r Sys.Date()`.
```

Surround with `r `

Code to run. Only the result will be included.

CC BY-SA RStudio

The sample includes `r nrow(flfp)` countries. The average FLFP rate is `r round(mean(flfp$wvs_flfp), 2)`.

Code chunks

Insert a chunk of R code with

```
```{r}
# some code
```
```

⌘ + Opt + i (Mac)

Ctrl + Alt + i (PC)

CC BY-SA RStudio

chunk options

By default, R Markdown includes both the code and its results

```
Here's some code
```{r}
dim(iris)
```
```

Here's some code

```
dim(iris)
```

```
## [1] 150   5
```

CC BY-SA RStudio

echo

Add options in the brackets after r.
echo = FALSE hides the code.

```
Here's some code
```{r echo=FALSE}
dim(iris)
```
```

Here's some code

```
## [1] 150   5
```

CC BY-SA RStudio

eval

eval = FALSE prevents the code from being run. As a result, no results will be displayed with the code.

```
Here's some code
```{r eval=FALSE}
dim(iris)
```
```

Here's some code

```
dim(iris)
```

CC BY-SA RStudio

include

include = FALSE runs the code, but prevents both the code and the results from appearing (e.g. to setup).

```
Here's some code
```{r include=FALSE}
dim(iris)
```
```

Here's some code

CC BY-SA RStudio

fig.height, fig.width

Specify the dimension of plots (in inches) with fig.width and fig.height. Separate multiple arguments with commas.

```
Here’s a plot
```{r echo=FALSE, fig.width=3, fig.height=5}
hist(iris[[2]])
```
```

CC BY-SA RStudio

warning

Suppresses warnings. Works similarly with messages.

```
Here's some code
```{r warning=FALSE}
warning("Something troublesome I do not want you to see")
```
```

Here's some code

CC BY-SA RStudio

global chunk options

Hides the code in every chunk unless over-rewritten within {r}.

```{r setup}
knitr::opts_chunk$set(echo = FALSE)
```

Header

A section of *key:value* pairs separated by dashed lines ———

```
---
title: "Untitled"
author: "RStudio"
date: "February 4, 2015"
output: html_document
---
Text of document
```

CC BY-SA RStudio

output

The output: field sets the format of the final report

output value	creates
html_document	html
pdf_document	pdf (requires Tex)
word_document	Microsoft Word (.docx)
odt_document	OpenDocument Text
rtf_document	Rich Text Format
md_document	Markdown
github_document	Github compatible markdown
ioslides_presentation	ioslides HTML slides
slidy_presentation	slidy HTML slides
beamer_presentation	Beamer pdf slides (requires Tex)

CC BY-SA RStudio

Now…

- Let’s demonstrate the utility of rmarkdown by using rmarkdown to generate a powerpoint presentation about… rmarkdown.
- (by the way, I tried to get bing copilot AI to help me generate such a presentation, but it got a bit confused with the meta aspects of the task and failed).
- Open up the script called ”rmarkdown introduction.R”