Before we get started, please visit the link to the right to answer 3 quick questions on visualization.

Thanks!

https://shorturl.at/jCX01
Outline

1. Introduction

2. Improving your R visualization workflow
   2a. Design (ggplot2::theme)
   2b. Sizing (camcorder)
   2c. Layouts (various libraries)

3. More than ggplot2 graphs - time permitting -
   3D Plots / Interactive Plots / Animated Videos

4. Visualization Resources at Princeton
Outline

1. Introduction

2. Improving your R visualization workflow
   2a. Design (ggplot2::theme)
   2b. Sizing (camcorder)
   2c. Layouts (various libraries)

3. More than ggplot2 graphs - time permitting -
   3D Plots / Interactive Plots / Animated Videos

4. Visualization Resources at Princeton
Introductions and Logistics

Quick Introductions
• Name
• Department
• Year or Position
• Favorite or most frustrating thing about visualizing data with R?

Ask questions anytime

When you see the R logo in the corner, go to RStudio session (on Adroit)
Setup Tools for Interactive Work

1. **Navigate to myadroit.princeton.edu**

2. **Click on Clusters menu.**
   i. Select > _Adroit Cluster Shell Access_
   ii. Type the following commands:

   ```
   wget -O - https://github.com/carolinarr/workshop_datavisR2/archive/main.tar.gz | tar xz
   mv workshop_datavisR2-main/ workshop_datavisR2/
   ```

3. **Go back to myadroit.princeton.edu tab.**

4. **Click on Interactive Apps menu, select RStudio Server option.** Enter the following:
   i. Number of hours: 2
   ii. R_LIBS_USER: `/home/<your-netid>/workshop_datavisR2-main/libs`
   iii. Extra slurm options: `-p class`

5. **Click Launch button.**

6. **Click Connect to RStudio Server button.**

7. **In RStudio, find the code folder, and open**
   `data-visualization-R-intermediate_examples.qmd`
Outline

1. Introduction

2. Improving your R visualization workflow
   2a. Design (ggplot2::theme)
   2b. Sizing (camcorder)
   2c. Layouts (various libraries)

3. More than ggplot2 graphs - time permitting -
   3D Plots / Interactive Plots / Animated Videos

4. Visualization Resources at Princeton
2a. Design (ggplot2::theme)

Part 1 (of 3) of Improving your R visualization workflow
Introducing today’s dataset

Groups of guinea pigs were given Vitamin C supplements at different dosages (mg / day) and had their teeth growth measured.

- **VC**: ascorbic acid (natural)
- **OJ**: orange juice (artificial)
- Dosages: 0.5, 1.0, 2.0

Introducing today’s dataset

Our dataset:

ToothGrowth

Our research question:

Do natural supplements of vitamin C (orange juice) produce more teeth growth than artificial supplements of vitamin C (ascorbic acid)?
## How do we control each part of a ggplot2 plot?

<table>
<thead>
<tr>
<th>ggplot2 ‘System’</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom_<em>() , coord_</em>()</td>
<td>control how data is rendered</td>
</tr>
<tr>
<td>stat_<em>() , scale_</em>()</td>
<td>control how data is transformed</td>
</tr>
<tr>
<td>labs()</td>
<td>control labels (e.g., title, axis labels)</td>
</tr>
</tbody>
</table>
Data visualization with ggplot2 :: CHEATSHEET

**Basics**

ggplot2 is based on the grammar of graphics, the idea that you can build every graph from the same few kinds of elements—data, aesthetics, and geoms—visual marks that represent data points.

To display values, map variables to the data to visual properties of the geom (aesthetics) like size, color, and x and y locations.

**Geoms**

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

**GRAPHICAL PRIMITIVES**

- `a <- ggplot2::aes(x=age, y=long, group=sex)`
- `a + geom_blank() + geom_point()`
- `a + geom_curve(y=1, x=1, n=100)`
- `a + geom_path(y=1, x=1, n=5)`
- `a + geom_polygon(y=1, x=1, n=5)`
- `a + geom_rect(ymin=0, xmin=0, xmax=1, ymax=1)`
- `a + geom_spline(y=1, x=1, n=5)`
- `a + geom_abline(intercept=1, slope=2)`
- `a + geom_shape(shape=5, size=5, fill= "red")`

**LINE SEGMENTS**

- Common aesthetics: x, y, alpha, color, linetype, size
- `a + geom_line(x=intercept=1, slope=2)`
- `a + geom_line(x=intercept=1, slope=2)`
- `a + geom_polygon(y=1, x=1, n=5)`
- `a + geom_rect(ymin=0, xmin=0, xmax=1, ymax=1)`
- `a + geom_spline(y=1, x=1, n=5)`
- `a + geom_abline(intercept=1, slope=2)`
- `a + geom_shape(shape=5, size=5, fill= "red")`

**ONE VARIABLE CONTINUOUS**

- `c <- ggplot2::aes(x=age, y=long)`
- `c + geom_bar(stat="identity")`
- `c + geom_point()`
- `c + geom_density(kernel="gaussian")`
- `c + geom_histogram(binwidth=3)`
- `c + geom_freqpoly()`
- `c + geom_area(stat="identity")`
- `c + geom_density(kernel="gaussian")`

**TWO VARIABLES CONTINUOUS**

- `d <- ggplot2::aes(x=age, y=long)`
- `d + geom_point()`
- `d + geom_density2d()`
- `d + geom_contour()`

**FIVE VARIABLES**

- `e <- ggplot2::aes(x=age, y=long, weight, color, size)`
- `e + geom_point()`
- `e + geom_density2d()`
- `e + geom_contour()`

**AES**

Common aesthetic values.

- `color` and `fill` - string ("red", "blue")
- `linetype` - integer or string (0: "blank", 1: "solid", 2: "dashed", 3: "dotted", 4: "dashed", 5: "long dash", 6: "twodash")
- `shape` - integer (line width in mm)

**CHEATSHEET**

CC BY-SA 3.0 | posit.co CC BY-SA 3.0 | posit.co

https://posit.co/resources/cheatsheets/ ggplot2 3.4.2 - Updated: 2023-07
# How do we control each part of a ggplot2 plot?

<table>
<thead>
<tr>
<th>ggplot2 ‘System’</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom_<em>(), coord_</em>()</td>
<td>control how data is rendered</td>
</tr>
<tr>
<td>stat_<em>(), scale_</em>()</td>
<td>control how data is transformed</td>
</tr>
<tr>
<td>labs()</td>
<td>control labels (e.g., title, axis labels)</td>
</tr>
<tr>
<td>theme*()</td>
<td>control “non-data parts” (e.g., fonts, background, ticks, etc.)</td>
</tr>
</tbody>
</table>

“Themes don’t change the perceptual properties of the plot, but they do help you make the plot aesthetically pleasing or match an existing style guide.”
– Hadley Wickham

[https://ggplot2-book.org/themes](https://ggplot2-book.org/themes), Hadley Wickham
How do we control each part of a ggplot2 plot?

<table>
<thead>
<tr>
<th>ggplot2 ‘System’</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>geom_*()</code>, <code>coord_*()</code></td>
<td>control how data is rendered</td>
</tr>
<tr>
<td><code>stat_*()</code>, <code>scale_*()</code></td>
<td>control how data is transformed</td>
</tr>
<tr>
<td><code>labs()</code></td>
<td>control labels (e.g., title, axis labels)</td>
</tr>
<tr>
<td><code>theme*()</code></td>
<td>control “non-data parts”</td>
</tr>
<tr>
<td>(<strong>“non-data”</strong></td>
<td>(e.g., fonts, background, ticks, etc.)</td>
</tr>
</tbody>
</table>

2., 2.a.i

# How do we control each part of a ggplot2 plot?

## ggplot2 ‘System’

<table>
<thead>
<tr>
<th>Function</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom_<em>(), coord_</em>()</td>
<td>control how data is rendered</td>
</tr>
<tr>
<td>stat_<em>(), scale_</em>()</td>
<td>control how data is transformed</td>
</tr>
<tr>
<td>labs()</td>
<td>control labels (e.g., title, axis labels)</td>
</tr>
<tr>
<td>theme*()</td>
<td>control “non-data parts” (e.g., fonts, background, ticks, etc.)</td>
</tr>
</tbody>
</table>

**“data”**

**“non-data”**

ggplot2::theme’s components

**complete themes**

`ggplot2.tidyverse.org/reference/ggtheme.html`

```r
theme_grey(), or `theme_gray()`
theme_bw()
theme_linedraw()
theme_light()
theme_dark()
theme_minimal()
theme_classic()
theme_void()
```
ggplot2::theme’s components

complete themes

**theme() function**
function that allows you to override current theme’s defaults. requires:

```
general format:
theme()
```

example:
```
theme(axis.title = element_text(hjust = c(0,0.5)),
       axis.line.x = element_line(linewidth = 3),
       legend.margin = margin(t = 0, r = 0, l = 5, r = 5),
       panel.background = element_rect(fill = "pink"))
```
**ggplot2::theme’s components**

**complete themes**

**theme() function**
function that allows you to override current theme’s defaults. requires:

**elements**
ggplot2.tidyverse.org/reference/theme.html
to identify which (non-data) parts of the plot you want to control

**general format:**
```
theme(element)
```

examples:
```
theme(
  axis.title = element_text(hjust = c(0,0.5)),
  axis.line.x = element_line(linewidth = 3),
  legend.margin = margin(t = 0, r = 0, l = 5, r = 5),
  panel.background = element_rect(fill = "pink")
)
```
**ggplot2::theme’s components**

**complete themes**

**theme() function**
function that allows you to override current theme’s defaults. requires:

**elements**

http://ggplot2.tidyverse.org/reference/theme.html
to identify which (non-data) parts of the plot you want to control

**element() function**

ggplot2.tidyverse.org/reference/element.html
function that specifies how the element you want to control is drawn (e.g. as a line? text?)
ggplot2::theme’s components

**complete themes**

**theme() function**
function that allows you to override current theme’s defaults. requires:

**elements**
http://ggplot2.tidyverse.org/reference/theme.html
to identify which (non-data) parts of the plot you want to control

**element() function**
ggplot2.tidyverse.org/reference/element.html
function that specifies how the element you want to control is drawn (e.g. as a line? text?)

general format:
```
theme(element = element())
```

examples:
```
theme(
  axis.title = element_text(hjust = c(0,0.5)) ,
  axis.line.x = element_line(linewidth = 3) ,
  legend.margin = margin(t = 0, r = 0, l = 5, r = 5),
  panel.background = element_rect(fill = “pink”)
)
```
Quick Recap

<table>
<thead>
<tr>
<th>ggplot2 ‘System’</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom_<em>(), coord_</em>()</td>
<td>control how data is rendered</td>
</tr>
<tr>
<td>stat_<em>(), scale_</em>()</td>
<td>control how data is transformed</td>
</tr>
<tr>
<td>labs()</td>
<td>control labels (e.g., title, axis labels)</td>
</tr>
<tr>
<td>theme*()</td>
<td>control “non-data parts” (e.g., fonts, background, ticks, etc.)</td>
</tr>
</tbody>
</table>

Taking the time to implement these tweaks is what makes effective plots that are easy to digest, and that conform to publishing standards.

https://ggplot2-book.org/themes, Hadley Wickham
Resources

ggplot2 documentation by Hadley Wickham et al.

ggplot2: Elegant Graphics for Data Analysis (3e)
by Hadley Wickham, Danielle Navarro, Thomas Lin Pedersen
https://ggplot2-book.org/

R for Data Science (2e)
by Hadley Wickham and Garrett Grolemund
https://r4ds.hadley.nz/
2b. Sizing (*camcorder*)

Part 2 (of 3) of Improving your R visualization workflow
Maintaining consistent plot size with *camcorder*

When plotting in R...

Two factors that influence the resulting plot:

1. Image’s dimensions (width, height)
   - *RStudio* – using digital dimensions, however your Plot pane is positioned on a certain day/time
   - *ggsave()* – using physical dimensions, specified as arguments

2. Your screen/printer resolution (ppi = pixels-per-inch, and dpi = dots-per-inch)
   - *RStudio* – exported at 72 dpi by default (not suitable for print)
   - *ggsave()* – specified as arguments, but default is 300 dpi

Problem -> RStudio operating off digital dimensions, ggsave() operating off physical

Some plot elements adjust to space, others don’t (Text!)

Potential Solution -> View plots with fixed physical dimensions and resolution with:

```r
 camcorder::gg_record()
```


[https://www.christophenicault.com/post/understand_size_dimension_ggplot2/](https://www.christophenicault.com/post/understand_size_dimension_ggplot2/)
2c. Layouts (various libraries)

Part 3 (of 3) of Improving your R visualization workflow
## Layout Options – there are many!

<table>
<thead>
<tr>
<th>Package</th>
<th>Function(s)</th>
<th>ggsave() Compatible?</th>
<th>Alignment</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>grid</td>
<td>viewport, grid.layout</td>
<td>no</td>
<td>no</td>
<td>- good overview explaining grobs, viewports, etc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- more technical</td>
</tr>
<tr>
<td>gridExtra</td>
<td>grid.arrange</td>
<td>yes</td>
<td>no</td>
<td>- extensions to grid package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- axes not aligned</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- simple introduction</td>
</tr>
<tr>
<td>gtable</td>
<td>rbind, cbind</td>
<td>yes</td>
<td>yes</td>
<td>- built on grid, higher-level layout functions</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- ggplot2 built on this</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- alignment possible, but complicated</td>
</tr>
<tr>
<td>egg</td>
<td>ggarrange</td>
<td>yes</td>
<td>yes</td>
<td>- good overview of functions</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- alignment possible</td>
</tr>
<tr>
<td>patchwork</td>
<td>plot_layout</td>
<td>yes</td>
<td>yes</td>
<td>- most intuitive? &quot;/&quot; top/bottom, &quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- control layouts, can create common legend</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- add empty space</td>
</tr>
<tr>
<td>cowplot</td>
<td>plot_grid</td>
<td>yes?</td>
<td>yes?</td>
<td>- plot_grid vignette</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- can easily remove common y-axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- can combine base R and ggplot plots</td>
</tr>
<tr>
<td>ggpubr</td>
<td>ggarrange</td>
<td>yes</td>
<td>yes</td>
<td>- easy creation of a common legend</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- multi-page layouts</td>
</tr>
</tbody>
</table>

Table copied and modified from [https://cran.r-project.org/web/packages/egg/vignettes/Ecosystem.html](https://cran.r-project.org/web/packages/egg/vignettes/Ecosystem.html). Several additions from blog post [https://adiradaniel.netlify.app/post/ggmultipane/](https://adiradaniel.netlify.app/post/ggmultipane/).
Layout Options

The drawback here is that you may again have to deal with size/resolution issues for each multi-plot configuration.

Possible Alternative?

*Design with physical dimensions in mind.*

1. Mock-up desired overall size of layout, with dimensions of each image.
2. Use camcorder to work on and create images with specific dimensions.
3. Save image as .svg file.
5. Create layout in Adobe InDesign.*

*To get Adobe InDesign visit https://oit.princeton.edu/creativecloud*
Summary so far

1. Introduction

2. Improving your R visualization workflow
   2a. Design (ggplot2::theme)
   2b. Sizing (camcorder)
   2c. Layouts (various libraries)

3. More than ggplot2 graphs - time permitting
   3D Plots / Interactive Plots / Animated Videos

4. Visualization Resources at Princeton

**summarytable::dfSummary(data)**
- a way to explore data
**d_, s_, plot_*, etc.**
- a way to organize variables

**ggplot2::geom_*, coord_*, stat_*, scale_*, lab()**
- data components

**ggplot2::theme(element = element())**
- non-data components
- lean on relative sizes
- create your own theme, and set it for work session

**physical dimensions, dpi, ppi**
**camcorder::gg_record()**
- see plots as the way they will actually be saved

**various packages for multi-image layouts**
- all built on top of grid
- can find a favorite, varying levels of ease-of-use
- can work with .svg’s and Adobe?
Outline

1. Introduction

2. Improving your R visualization workflow
   2a. Design (ggplot2::theme)
   2b. Sizing (camcorder)
   2c. Layouts (various libraries)

3. More than ggplot2 graphs - time permitting -
   3D Plots / Interactive Plots / Animated Videos

4. Visualization Resources at Princeton
3D Plots

Quick Overview
3D Plots

Common packages:

- rgl
  - examples:
    - [https://r-graph-gallery.com/3d.html](https://r-graph-gallery.com/3d.html)

- plotly

- rayshader (3D maps)
  - [https://www.rayshader.com/](https://www.rayshader.com/)
Interactive Plots

Quick Overview
Interactive Plots

Common packages:

• plotly
  • Getting Started with Plotly in ggplot2
    https://plotly.com/ggplot2/getting-started/

• highcharter
  • https://jkunst.com/highcharter/
Animated Plots

Quick Overview
Animated Plots

• For example, create gifs, or videos of data over time
• Common packages:
  • camcorder
    • use `gg_playback()` function to create video of every image in your folder
  • gganimate
    • [https://gganimate.com/](https://gganimate.com/)
  • av
    • `av` package by ropensci
  • ffmpeg (not an R package)
Feedback

https://shorturl.at/eijqP
Outline

1. Introduction

2. Improving your R visualization workflow
   2a. Design (ggplot2::theme)
   2b. Sizing (camcorder)
   2c. Layouts (various libraries)

3. More than ggplot2 graphs - *time permitting* -
   3D Plots / Interactive Plots / Animated Videos

4. Visualization Resources at Princeton
Visualization Services at Princeton

- Princeton Institute for Computational Science and Engineering (PICSciE) & Research Computing
- Princeton Library
  - Maps and Geospatial Information Center – GIS Consultations and Workshops
  - Data and Statistical Services (DSS) – Consultations on statistical graphs
  - Stokes Library – Stokes Viz Hub – Consultations and workshops on visualizations for qualitative data
- Center for Data, Analytics, and Reporting (CeDAR) – workshops on Tableau
- VizE Lab – ethnographic data visualization
- Office of Population Research – workshops on qualitative visualizations
- Data Driven Social Sciences – workshops, R group
1. Introduction

2. Improving Vis Workflow

3. More Than ggplot2 Graphs

4. Visualization Resources

PICSciE’s and Research Computing’s Visualization Staff

Eliot Feibush, PhD
Visualization Scientist
efeibush@princeton.edu

William (Bill) Guthe
Senior Graphical Information Systems Visualization Analyst
wguthe@princeton.edu

Carolina Roe-Raymond, PhD
Visualization Analyst
c.roe-raymond@princeton.edu
Visualization Services

Consultations
One-on-one consultations, either in person or through email.

Projects
Select collaborations where our staff helps build visualizations for your research.

Training
Offer live and recorded workshop on visualization.

Software and Hardware
Access to
- high performance computing systems
- visualization lab
- computers with software such as R, Python, MATLAB, VisIt, ArcGISPro, QGIS, and others.

Example Topics and Questions
- Visualization Design
- Visualization Tools
- Troubleshooting Code

PICSciE Workshops
researchcomputing.princeton.edu/workshops

GIS (Geographical Information Systems) Workshops
library.princeton.edu/collections/pumagic/workshops
Examples

Design **Consultation** Example
focused on improving visualizations prepared for presentations and publications
Examples

Results of a Project
created by our staff member Eliot, and researchers Joakim Andén and Yoel Shkolnisky

Cryo-EM Reconstruction of Human Malaria Parasite
Examples

Results of a Project
Interactive site created by our staff members Carolina and Bill, in collaboration with Princeton’s Web Development Services

Before

After
Examples

Results of a Project
created by our staff member Eliot and Bill, with researchers Maider Llaguno-Munitxa and Elie Bou-Zeid
Examples

Find more examples in our Visualization Gallery

https://researchcomputing.princeton.edu/services/visualization/visualization-gallery
How to Get Help

Contact us, we’re happy to help!

Our visualization team can be reached at: visrc@princeton.edu
Summary

1. Introduction

2. Improving your R visualization workflow
   2a. Design (ggplot2::theme)
   2b. Sizing (camcorder)
   2c. Layouts (various libraries)

3. More than ggplot2 graphs - *time permitting*
   3D Plots / Interactive Plots / Animated Videos

4. Visualization Resources at Princeton
   visrc@princeton.edu

summarytable::dfSummary(data)
   a way to explore data
d_, s_, plot_, etc.
   a way to organize variables

ggplot2::geom_(), coord_(),
stat_(), scale_(), lab()
data components

theme(element = element())
non-data components
lean on relative sizes
create your own theme, and set it for work session

physical dimensions, dpi, ppi

various packages for multi-image layouts
all built on top of grid
can find a favorite, varying levels of ease-of-use
can work with .svg’s and Adobe?

camcorder::gg_record()
see plots as the way they will actually be saved