Data Visualization

Grace J. Goodwin
(she/ her/ hers)
@gracejgoodwin

ggplot

ggplot

- Open source data visualization package in R
- Breaks up graphs into semantic components (scales and layers)
- Creates nice looking, high quality graphs
- In contrast to base R, ggplot allows user to add/remove/change graph components

ggplot structure

ggplots are built from:

- a **data set**
- a **coordinate system**
- **aes** positions/maps the variables on the plot
- a **geom** draws the graph (e.g, scatterplot, histogram, etc.)

ggplot template

Think of your ggplot like a cake with layers

ggplot template

Think of your ggplot like a cake with layers

Each layer adds something to the graph

Layers:

ggplot template

Think of your ggplot like a cake with layers

Each layer adds something to the graph

Layers:

`+ geom_histogram` (specifies that graph should be a histogram, etc.)

ggplot template

Think of your ggplot like a cake with layers

Each layer adds something to the graph

Layers:

`+ geom_histogram` (specifies that graph should be a histogram, etc.)

`+ ggtitle` (adds a title)

ggplot template

Think of your ggplot like a cake with layers

Each layer adds something to the graph

Layers:

+ `geom_histogram` (specifies that graph should be a histogram, etc.)

+ `ggtitle` (adds a title)

+ `theme` (adds a theme)

ggplot template

Think of your ggplot like a cake with layers

Each layer adds something to the graph

Layers:

+ `geom_histogram` (specifies that graph should be a histogram, etc.)

+ `ggtitle` (adds a title)

+ `theme` (adds a theme)

+ `facet_wrap(~var)` (split plots by specific variable)

Some examples

Some examples

Basic Scatterplot

Markdown

```
ggplot(data = data,
       mapping = aes(x = age,
                     y = ht)) +
  geom_point()
```

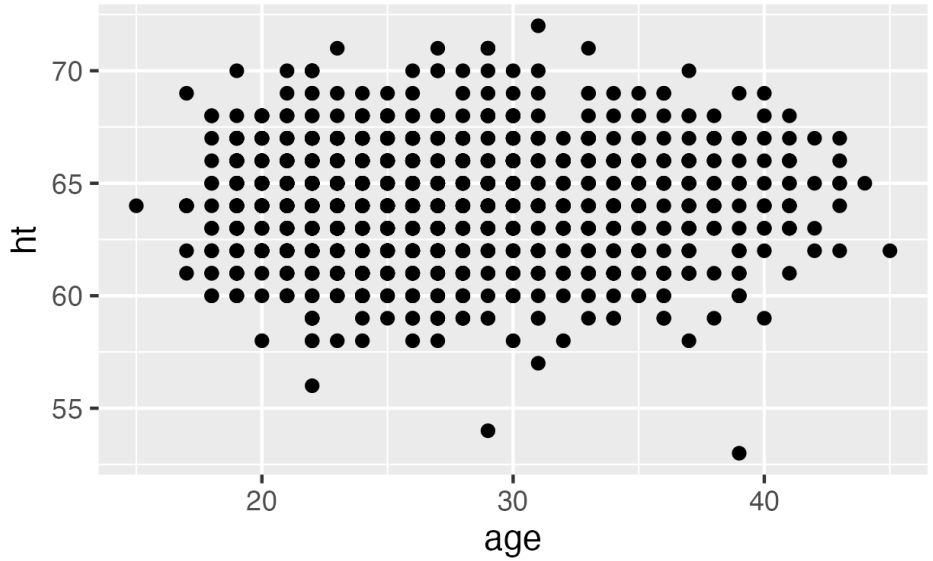
Some examples

Basic Scatterplot

Markdown

```
ggplot(data = data,
       mapping = aes(x = age,
                     y = ht)) +
  geom_point()
```

Output



Basic Histogram

Markdown

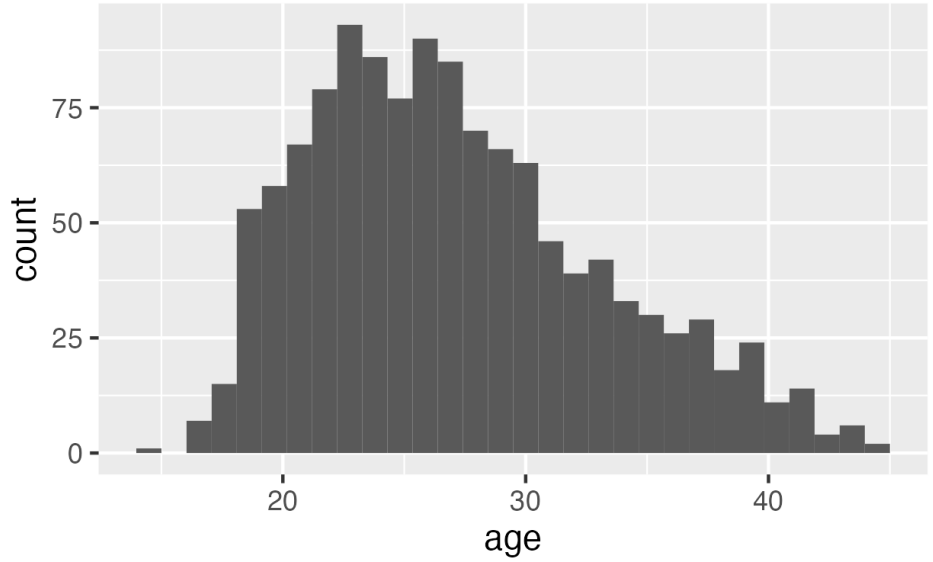
```
ggplot(data = data,
       mapping = aes(x = age)) +
  geom_histogram()
```

Basic Histogram

Markdown

```
ggplot(data = data,
       mapping = aes(x = age)) +
  geom_histogram()
```

Output



Add some color with `color =`

Markdown

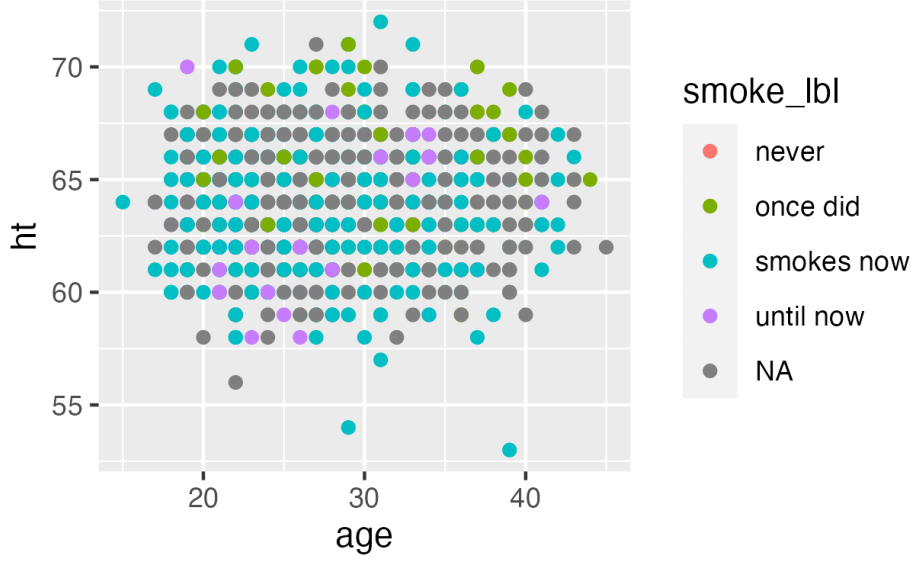
```
ggplot(data = data,
       mapping = aes(x = age,
                     y = ht,
                     color = smoke_lbl)) +
  geom_point()
```

Add some color with `color =`

Markdown

```
ggplot(data = data,
       mapping = aes(x = age,
                     y = ht,
                     color = smoke_lbl)) +
  geom_point()
```

Output



We can change which colors the data is mapped to by using a `scale_` function.

Markdown

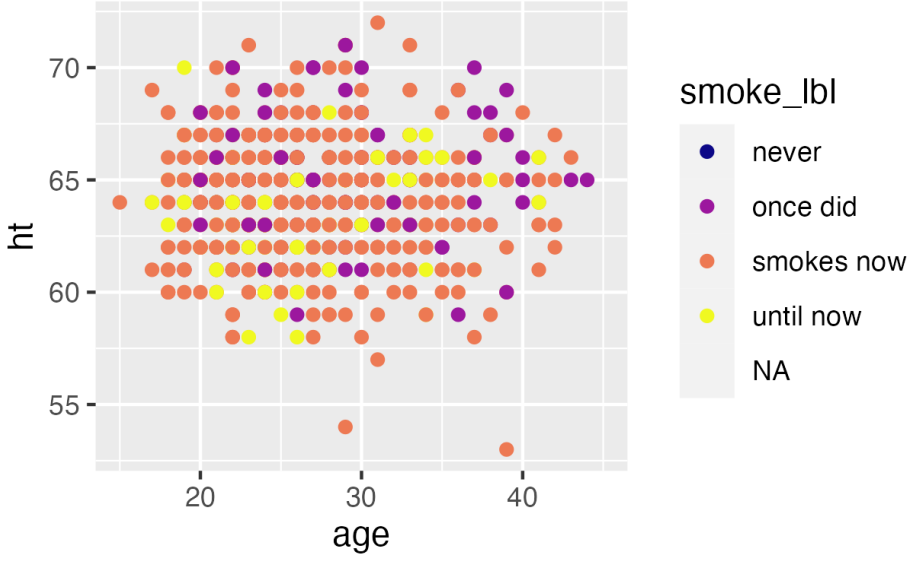
```
ggplot(data = data,
       mapping = aes(x = age,
                     y = ht,
                     color = smoke_lbl)) +
  geom_point() +
  scale_color_viridis_d(option = "plasma")
```

We can change which colors the data is mapped to by using a `scale_` function.

Markdown

```
ggplot(data = data,
       mapping = aes(x = age,
                     y = ht,
                     color = smoke_lbl)) +
  geom_point() +
  scale_color_viridis_d(option = "plasma")
```

Output



Plot titles and formatting

Markdown

```
ggplot(data,mapping =
aes(age,ht,color = smoke_lbl)) +
  geom_point() +
  scale_color_viridis_d(option = "plasma") +
  labs(title = "My Title",
       subtitle = "My Subtitle",
       x = "Age",
       y = "Height (inches)",
       color = "Smoking habits") +
  theme_apa()
```

Plot titles and formatting

Markdown

```
ggplot(data,mapping =
aes(age,ht,color = smoke_lbl)) +
  geom_point() +
  scale_color_viridis_d(option = "plasma") +
  labs(title = "My Title",
       subtitle = "My Subtitle",
       x = "Age",
       y = "Height (inches)",
       color = "Smoking habits") +
  theme_apa()
```

`facet_wrap` creates separate plot for all of the labels in the "smoking" variable

Markdown

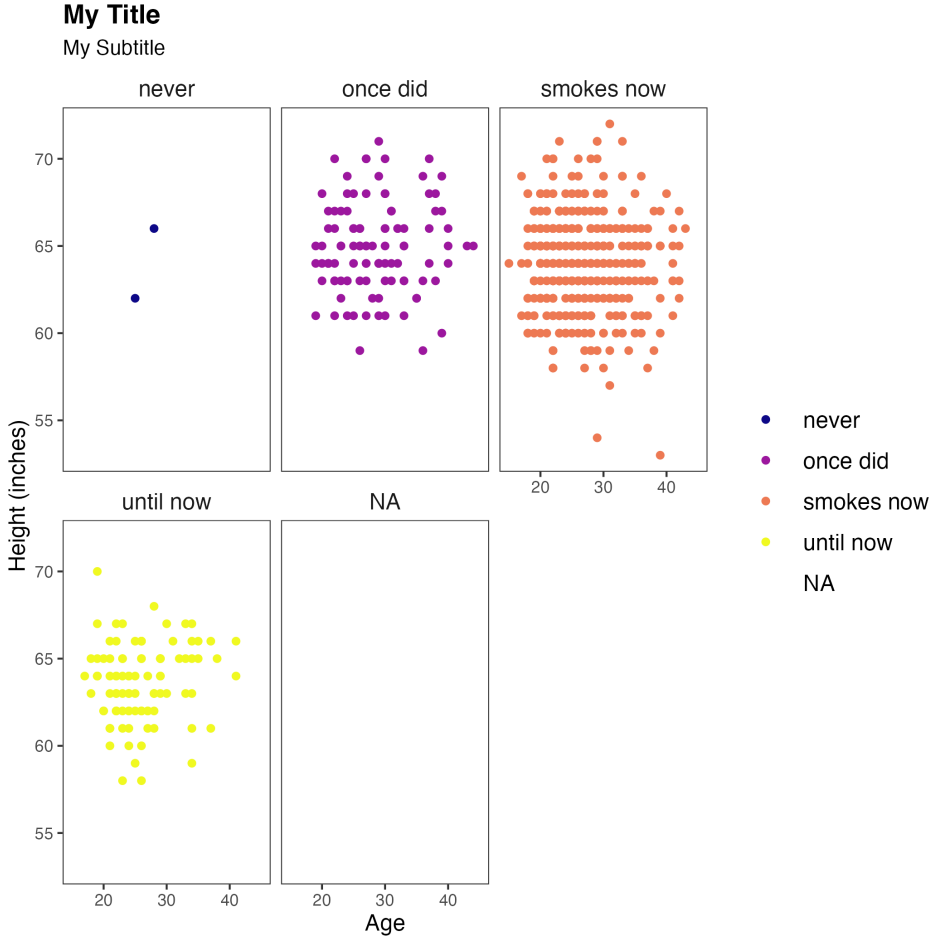
```
ggplot(data,mapping =
aes(age,ht,color = smoke_lbl)) +
  geom_point() +
  scale_color_viridis_d(option = "plasma") +
  labs(title = "My Title",
       subtitle = "My Subtitle",
       x = "Age",
       y = "Height (inches)",
       color = "Smoking habits") +
  theme_apa() +
  facet_wrap(~smoke_lbl)
```

`facet_wrap` creates separate plot for all of the labels in the "smoking" variable

Markdown

```
ggplot(data,mapping =
aes(age,ht,color = smoke_lbl)) +
  geom_point() +
  scale_color_viridis_d(option = "plasma") +
  labs(title = "My Title",
       subtitle = "My Subtitle",
       x = "Age",
       y = "Height (inches)",
       color = "Smoking habits") +
  theme_apa() +
  facet_wrap(~smoke_lbl)
```

Output



Save your plots

- RMarkdown: just knit your file and your plots will show up as part of your HTML, Word, or PDF document.

Save your plots

- RMarkdown: just knit your file and your plots will show up as part of your HTML, Word, or PDF document.
- Use the `ggsave` function.

Save your plots

- RMarkdown: just knit your file and your plots will show up as part of your HTML, Word, or PDF document.
- Use the `ggsave` function.
- By default, `ggsave` will save the last plot you made. So you can add this line of code beneath each of the graphs you want to save:

Save your plots

- RMarkdown: just knit your file and your plots will show up as part of your HTML, Word, or PDF document.
- Use the `ggsave` function.
- By default, `ggsave` will save the last plot you made. So you can add this line of code beneath each of the graphs you want to save:

```
ggsave("filename/my_plot.png")
```

Data visualization with ggplot2 : : CHEAT SHEET

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and **geoms**—visual marks that represent data points.

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

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                   required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
 stat = <STAT>, position = <POSITION>) +                   Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

ggplot(data = mpg, **aes**(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

last_plot() Returns the last plot.

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Aes

Common aesthetic values.

color and fill - string ("red", "#RRGGBB")

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

a + geom_blank() and **a + expand_limits()**
Ensure limits include values across all plots.

b + geom_curve(aes(yend = lat + 1, xend = long + 1), curvature = 1) - 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(alpha = 50)) - x, y, alpha, color, fill, group, subgroup, 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

TWO VARIABLES

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

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

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) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

one discrete, one continuous
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

both discrete
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_density_2d()
x, y, alpha, color, group, linetype, size

h + geom_hex()
x, y, alpha, color, 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))

More Resources