

VISUALIZATION WITH GGPLOT2

LECTURE: UNSUPERVISED LEARNING AND EVOLUTIONARY COMPUTATION USING R

Jakob Bossek

MALEO Group, Department of Computer Science, Paderborn University, Germany

28th Oct, 2024

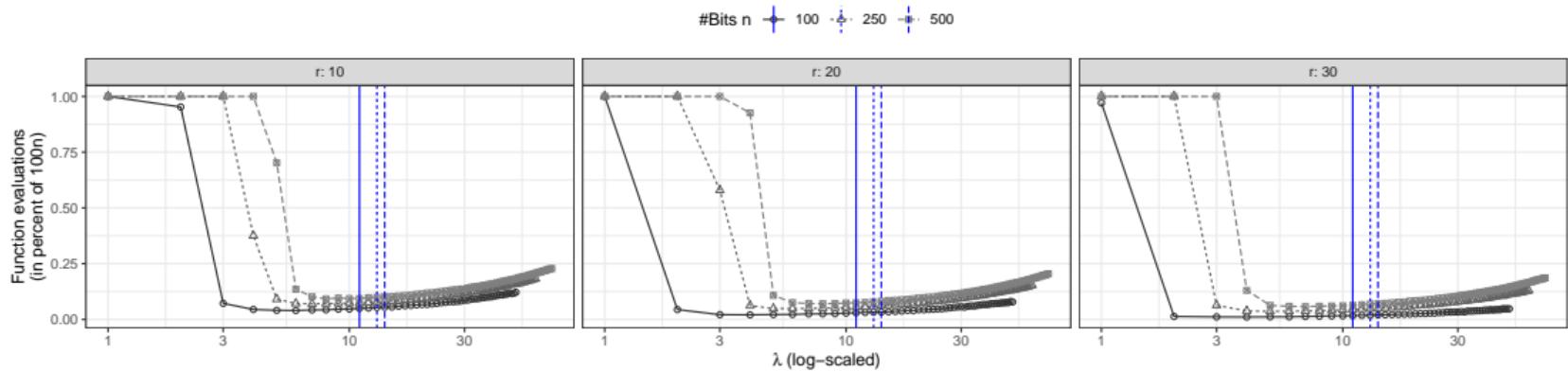
Table of Contents

1. Gallery
2. Introduction
3. Basic concepts
4. Facetting
5. Basic plots
6. Themes
7. Case study
8. Quick plots
9. Clean visualizations
10. Wrap-Up

Gallery

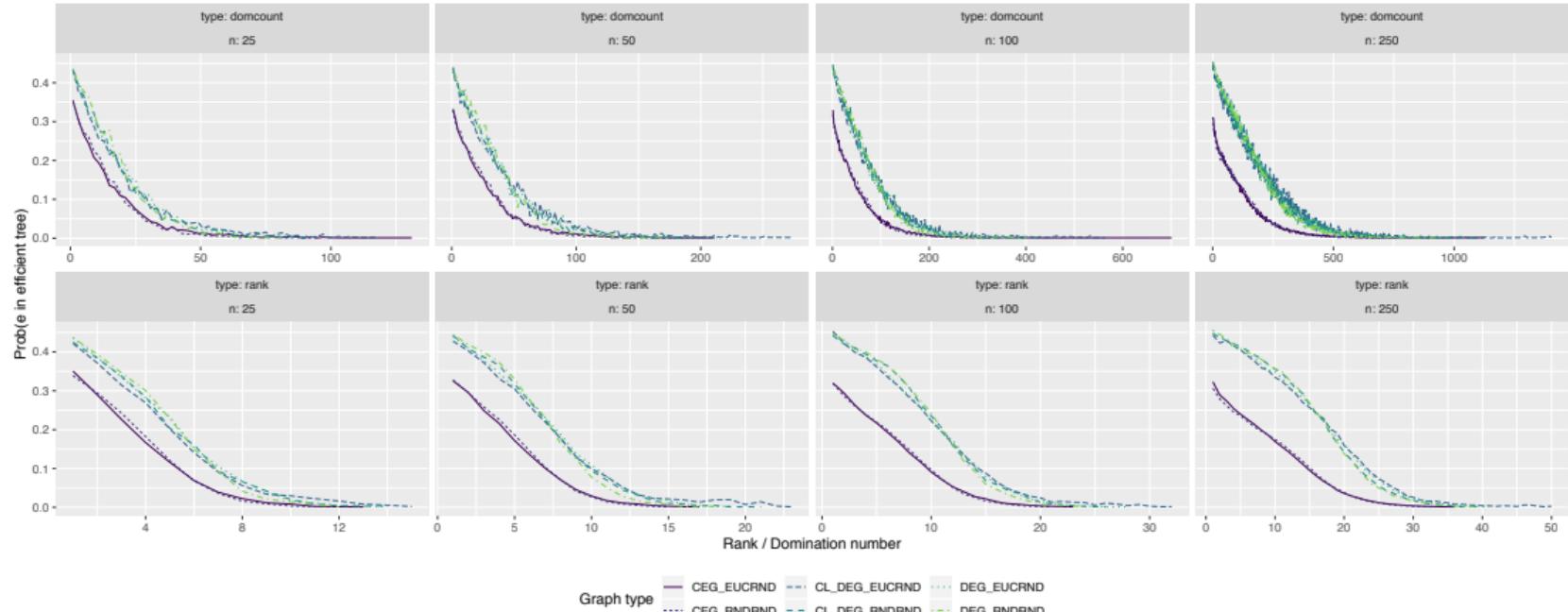
Gallery

Examples of `ggplot2` visualizations from my research:



Gallery

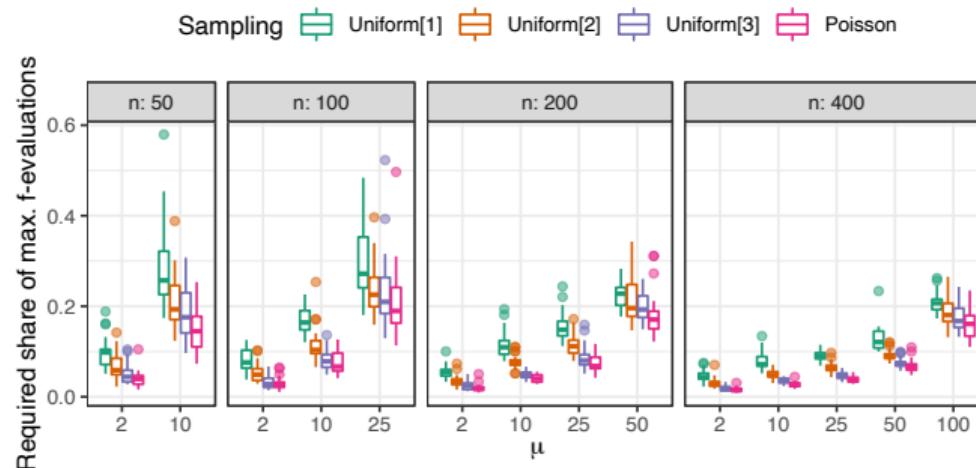
Examples of ggplot2 visualizations from my research:



CL: clustered
(C,D)EG: (complete,delauney) edge generator
RND: random, EUC: Euclidean

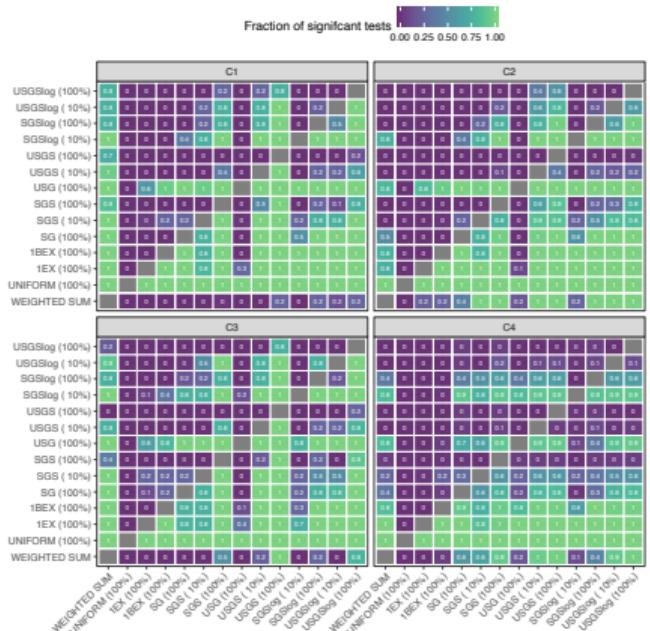
Gallery

Examples of ggplot2 visualizations from my research:



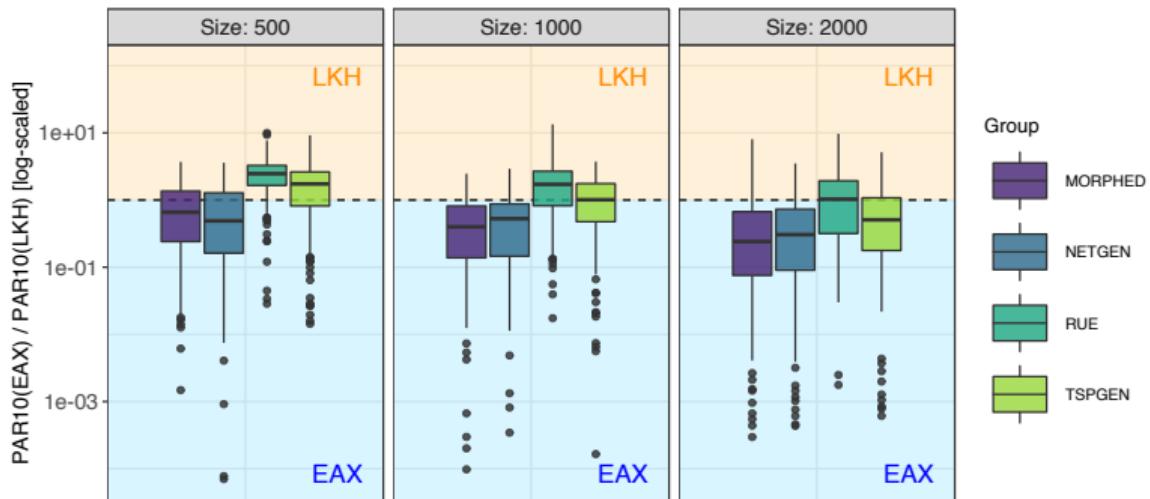
Gallery

Examples of ggplot2 visualizations from my research:



Gallery

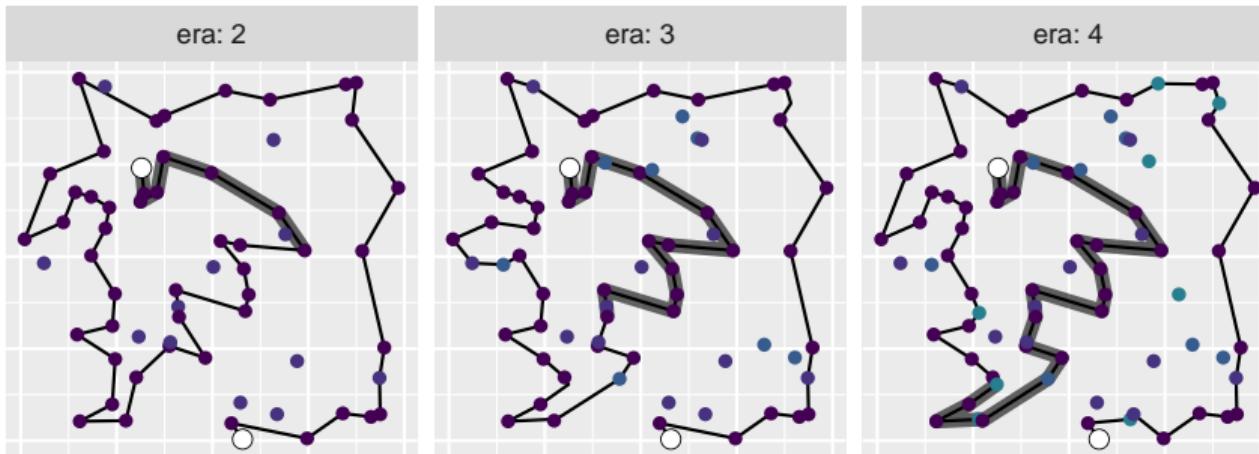
Examples of ggplot2 visualizations from my research:



Gallery

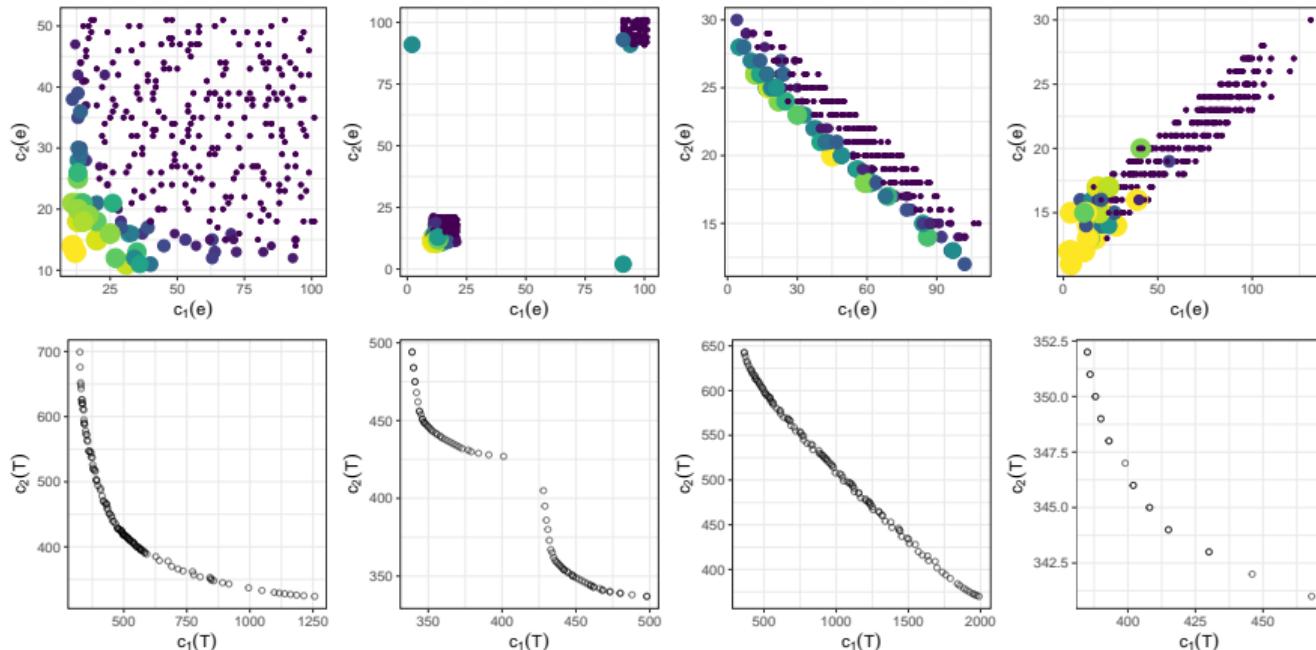
Examples of ggplot2 visualizations from my research:

DM: (0.25,0.25,0.25,0.25,0.75,0.75,0.75)



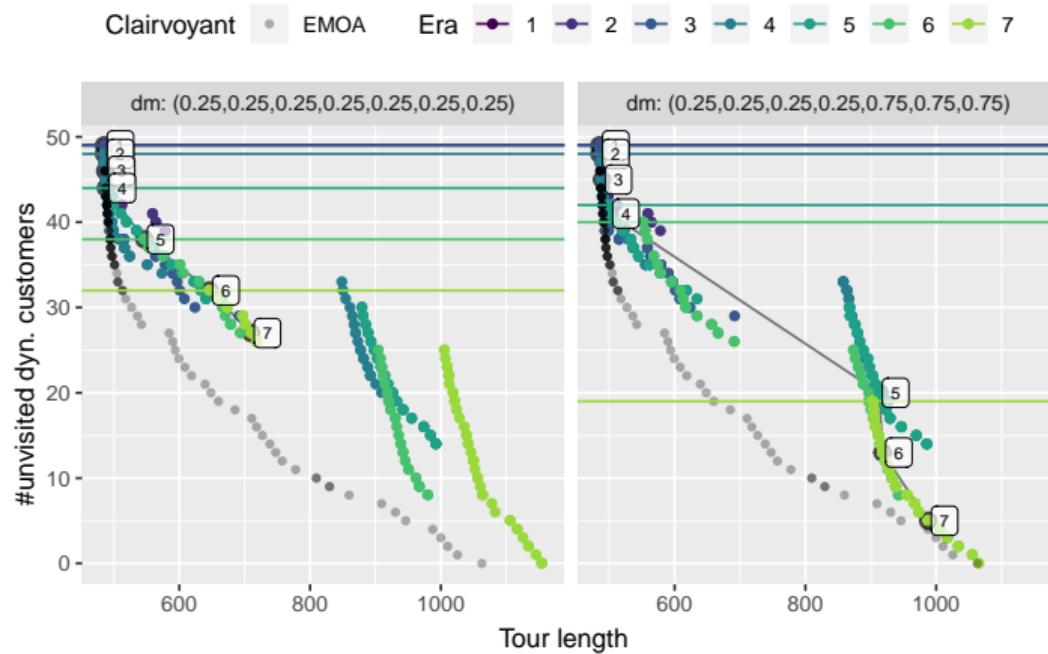
Gallery

Examples of ggplot2 visualizations from my research:



Gallery

Examples of ggplot2 visualizations from my research:



Introduction

Introduction

What is ggplot2 and why should I use it?

- ▶ Powerful R package for data visualization



Introduction

What is ggplot2 and why should I use it?

- ▶ Powerful R package for data visualization
- ▶ Based on the *Grammar of Graphics*¹ by Leland Wilkinson (Wilkinson 2005)



¹ Extended to the *layered Grammar of Graphics* by Hadley Wickham.

Introduction

What is ggplot2 and why should I use it?

- ▶ Powerful R package for data visualization
- ▶ Based on the *Grammar of Graphics*¹ by Leland Wilkinson (Wilkinson 2005)
- ▶ Easy to learn and full of features



¹ Extended to the *layered Grammar of Graphics* by Hadley Wickham.

Introduction

What is ggplot2 and why should I use it?

- ▶ Powerful R package for data visualization
- ▶ Based on the *Grammar of Graphics*¹ by Leland Wilkinson (Wilkinson 2005)
- ▶ Easy to learn and full of features
- ▶ Active development (steady implementation of new features, bugfixes)



¹ Extended to the *layered Grammar of Graphics* by Hadley Wickham.

Introduction

What is ggplot2 and why should I use it?



- ▶ Powerful R package for data visualization
- ▶ Based on the *Grammar of Graphics*¹ by Leland Wilkinson (Wilkinson 2005)
- ▶ Easy to learn and full of features
- ▶ Active development (steady implementation of new features, bugfixes)
- ▶ Active and helpful community (mailing list, stack overflow)

¹ Extended to the *layered Grammar of Graphics* by Hadley Wickham.

Introduction

What is ggplot2 and why should I use it?



- ▶ Powerful R package for data visualization
- ▶ Based on the *Grammar of Graphics*¹ by Leland Wilkinson (Wilkinson 2005)
- ▶ Easy to learn and full of features
- ▶ Active development (steady implementation of new features, bugfixes)
- ▶ Active and helpful community (mailing list, stack overflow)
- ▶ In the meanwhile de-facto standard for visualization in R

¹ Extended to the *layered Grammar of Graphics* by Hadley Wickham.

Introduction

What is ggplot2 and why should I use it?



- ▶ Powerful R package for data visualization
- ▶ Based on the *Grammar of Graphics*¹ by Leland Wilkinson (Wilkinson 2005)
- ▶ Easy to learn and full of features
- ▶ Active development (steady implementation of new features, bugfixes)
- ▶ Active and helpful community (mailing list, stack overflow)
- ▶ In the meanwhile de-facto standard for visualization in R
- ▶ Using ggplot2 is fun!

¹ Extended to the *layered Grammar of Graphics* by Hadley Wickham.

Sorry, I hate R!

No worries! You can translate what you learn today (grammer of graphics) to other languages:

Libraries inspired by ggplot2

- ▶ gadfly for the Julia language²

Sorry, I hate R!

No worries! You can translate what you learn today (grammer of graphics) to other languages:

Libraries inspired by ggplot2

- ▶ gadfly for the Julia language²

² Very young (2009), high-performance general-purpose language with focus on numerical calculations and computational science.

Sorry, I hate R!

No worries! You can translate what you learn today (grammer of graphics) to other languages:

Libraries inspired by ggplot2

- ▶ gadfly for the Julia language²
- ▶ Python implementation ggpy³

² Very young (2009), high-performance general-purpose language with focus on numerical calculations and computational science.

Sorry, I hate R!

No worries! You can translate what you learn today (grammer of graphics) to other languages:

Libraries inspired by ggplot2

- ▶ gadfly for the Julia language²
- ▶ Python implementation ggpy³

² Very young (2009), high-performance general-purpose language with focus on numerical calculations and computational science.

³ Alternative: call R functions directly from Python, e.g., with rpy2.

Sorry, I hate R!

No worries! You can translate what you learn today (grammer of graphics) to other languages:

Libraries inspired by ggplot2

- ▶ gadfly for the Julia language²
- ▶ Python implementation ggpy³
- ▶ ggramm for Matlab

² Very young (2009), high-performance general-purpose language with focus on numerical calculations and computational science.

³ Alternative: call R functions directly from Python, e.g., with rpy2.

Sorry, I hate R!

No worries! You can translate what you learn today (grammer of graphics) to other languages:

Libraries inspired by ggplot2

- ▶ gadfly for the Julia language²
- ▶ Python implementation ggpy³
- ▶ ggramm for Matlab
- ▶ GGPlot for Perl

² Very young (2009), high-performance general-purpose language with focus on numerical calculations and computational science.

³ Alternative: call R functions directly from Python, e.g., with rpy2.

Sorry, I hate R!

No worries! You can translate what you learn today (grammer of graphics) to other languages:

Libraries inspired by ggplot2

- ▶ gadfly for the Julia language²
- ▶ Python implementation gppy³
- ▶ ggramm for Matlab
- ▶ GGPlot for Perl
- ▶ etc.

² Very young (2009), high-performance general-purpose language with focus on numerical calculations and computational science.

³ Alternative: call R functions directly from Python, e.g., with rpy2.

Advantages vs disadvantages

Advantages 😊

- ▶ High level of abstraction
- ▶ Produces high-quality images
- ▶ Sticks to essential graphics design rules
- ▶ Easy to generate "new", outstanding visualizations
- ▶ Basics not that difficult to learn
- ▶ Build complex visualizations (given some effort)

Disadvantages 😞

- ▶ Data in data frame
- ▶ No support for 3D data
- ▶ No graph theory stuff

Advantages vs disadvantages

Advantages 😊

- ▶ High level of abstraction
- ▶ Produces high-quality images
- ▶ Sticks to essential graphics design rules
- ▶ Easy to generate "new", outstanding visualizations
- ▶ Basics not that difficult to learn
- ▶ Build complex visualizations (given some effort)

Disadvantages 😞

- ▶ ~~Data in data frame~~
Not a disadvantage in my eyes!
- ▶ No support for 3D data
Unfortunately no support planned, but there are similar alternatives: `rayshader` or `plotly`!
- ▶ ~~No graph theory stuff~~
Actually not true anymore! See packages `ggnetwork`,
`geomnet` or `ggnetwork` Tyner, Briatte, and Hofmann 2017

Prerequisite: tidy data

Unsurprisingly, as ggplot2's creator and main developer is Hadley Wickham, ggplot2 operates on tidy data.

Recall: Rectangular data is tidy if ...

- ▶ Each column is a variable
- ▶ Each row is an observation
- ▶ Each observational unit has its own table

Use tidyverse functionality to tidy data!

Basic concepts

Grammar of Graphics

Hadley (Wickham 2009) describes the idea as follows:

... the grammar tells us that a statistical graphic is a mapping from data to aesthetic attributes (colour, shape, size) of geometric objects (points, lines, bars). The plot may also contain statistical transformations of the data and is drawn on a specific coordinate system ...



Grammer of Graphics

Hadley (Wickham 2009) describes the idea as follows:

... the grammar tells us that a statistical graphic is a mapping from data to aesthetic attributes (colour, shape, size) of geometric objects (points, lines, bars). The plot may also contain statistical transformations of the data and is drawn on a specific coordinate system ...

- ▶ Abstraction of graphics ideas and objects



Grammar of Graphics

Hadley (Wickham 2009) describes the idea as follows:

... the grammar tells us that a statistical graphic is a mapping from data to aesthetic attributes (colour, shape, size) of geometric objects (points, lines, bars). The plot may also contain statistical transformations of the data and is drawn on a specific coordinate system ...

- ▶ Abstraction of graphics ideas and objects
- ▶ Allow "theoretical" discussion of how (new) graphics are composed



Grammer of Graphics

Hadley (Wickham 2009) describes the idea as follows:

... the grammar tells us that a statistical graphic is a mapping from data to aesthetic attributes (colour, shape, size) of geometric objects (points, lines, bars). The plot may also contain statistical transformations of the data and is drawn on a specific coordinate system ...

- ▶ Abstraction of graphics ideas and objects
- ▶ Allow "theoretical" discussion of how (new) graphics are composed
- ▶ Much like in spoken language it allows for extension and adaption to new (previously unseen) situations



Grammer of Graphics

- ▶ Introduced by Leland Wilkinson (2005) to give an answer to the question:
What is a statistical graphic?

Grammer of Graphics

- ▶ Introduced by Leland Wilkinson (2005) to give an answer to the question:
What is a statistical graphic?
- ▶ Core idea: graphic is composed of different layers

Grammer of Graphics

- ▶ Introduced by Leland Wilkinson (2005) to give an answer to the question:
What is a statistical graphic?
- ▶ Core idea: graphic is composed of different layers
- ▶ In each layer data is mapped to different aesthetics attributes (e.g., colour) of geometric objects (e.g., points)

Grammer of Graphics

- ▶ Introduced by Leland Wilkinson (2005) to give an answer to the question:
What is a statistical graphic?
- ▶ Core idea: graphic is composed of different layers
- ▶ In each layer data is mapped to different aesthetics attributes (e.g., colour) of geometric objects (e.g., points)
- ▶ Abstraction of low-level plots

Grammer of Graphics

- ▶ Introduced by Leland Wilkinson (2005) to give an answer to the question:
What is a statistical graphic?
- ▶ Core idea: graphic is composed of different layers
- ▶ In each layer data is mapped to different aesthetics attributes (e.g., colour) of geometric objects (e.g., points)
- ▶ Abstraction of low-level plots
- ▶ ~ no named methods like plot, hist or scatterplot

Grammer of Graphics

- ▶ Introduced by Leland Wilkinson (2005) to give an answer to the question:
What is a statistical graphic?
- ▶ Core idea: graphic is composed of different layers
- ▶ In each layer data is mapped to different aesthetics attributes (e.g., colour) of geometric objects (e.g., points)
- ▶ Abstraction of low-level plots
- ▶ ~ no named methods like plot, hist or scatterplot

Are you confused?

Does not matter! You can generate all types of basic plots without a deep understanding of the underlying principles ☺

Basic vocabulary

data The actual data (always a R data frame)

Basic vocabulary

data The actual data (always a R data frame)

aesthetics Visual properties *mapped* to variables of the data (i.e., columns of the data frame)

Basic vocabulary

data The actual data (always a R data frame)

aesthetics Visual properties *mapped* to variables of the data (i.e., columns of the data frame)

- ▶ Map categorical variable sex to colour: male → red, female → blue

Basic vocabulary

data The actual data (always a R data frame)

aesthetics Visual properties *mapped* to variables of the data (i.e., columns of the data frame)

- ▶ Map categorical variable `sex` to colour: male → red, female → blue
- ▶ Also possible: map continuous variable `temperature` to colour

Basic vocabulary

data The actual data (always a R data frame)

aesthetics Visual properties *mapped* to variables of the data (i.e., columns of the data frame)

- ▶ Map categorical variable `sex` to colour: male → red, female → blue
- ▶ Also possible: map continuous variable `temperature` to colour

geoms Short version of *geometric objects*; actual type of object to display (e.g., point, boxplot, . . .)

Basic vocabulary

data The actual data (always a R data frame)

aesthetics Visual properties *mapped* to variables of the data (i.e., columns of the data frame)

- ▶ Map categorical variable `sex` to colour: male → red, female → blue
- ▶ Also possible: map continuous variable `temperature` to colour

geoms Short version of *geometric objects*; actual type of object to display (e.g., point, boxplot, . . .)

facets Use multiple panels to plot subsets of the data

Starting with ggplot

CRAN version⁴

```
> install.packages("ggplot2")
> library(ggplot2)
```

Development version

Useful to check out and experiment with upcoming features:

```
> install.packages("devtools")
> devtools::install_github("tidyverse/ggplot2")
> library(ggplot2)
```

⁴

Current version number is 3.3.5

Starting with ggplot (cont.)

Data set diamonds

Should be familiar: data on $\approx 54\,000$ diamonds:

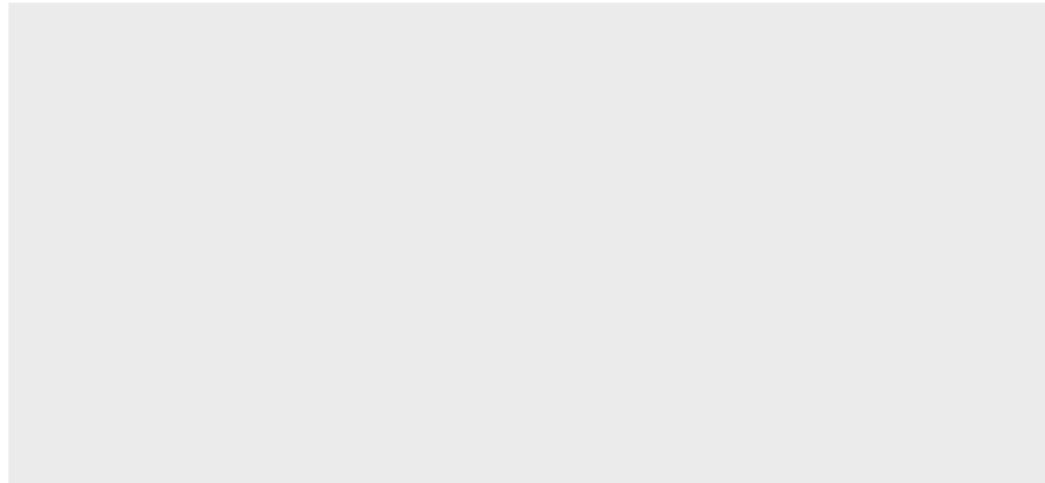
```
> require(ggplot2, quiet = TRUE)
> data(diamonds)
> ds = subset(diamonds, color %in% c("F", "G", "H", "I"))
> set.seed(123)
> ds = ds[sample(1:nrow(ds), size = 200L, replace = FALSE), ]
> head(ds, n = 6L)

## # A tibble: 6 x 10
##   carat     cut   color clarity depth table price     x     y     z
##   <dbl>    <ord> <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  1       Good    H     SI2     63.2    60  3740  6.27  6.3   3.97
## 2  0.3     Ideal   H     VS1     61.6    55   526  4.27  4.3   2.64
## 3  0.52    Ideal   H     VVS1    61      56  1748  5.17  5.21  3.17
## 4  0.9     Premium G     SI1     61.9    59  3669  6.18  6.13  3.81
## 5  0.3     Ideal   F     VS2     61.4    57   605  4.34  4.36  2.67
## 6  0.98    Premium I     I1     59.6    60  2064  6.43  6.3   3.79
```

Starting with ggplot2 (cont.)

We want a scatterplot of carat vs price

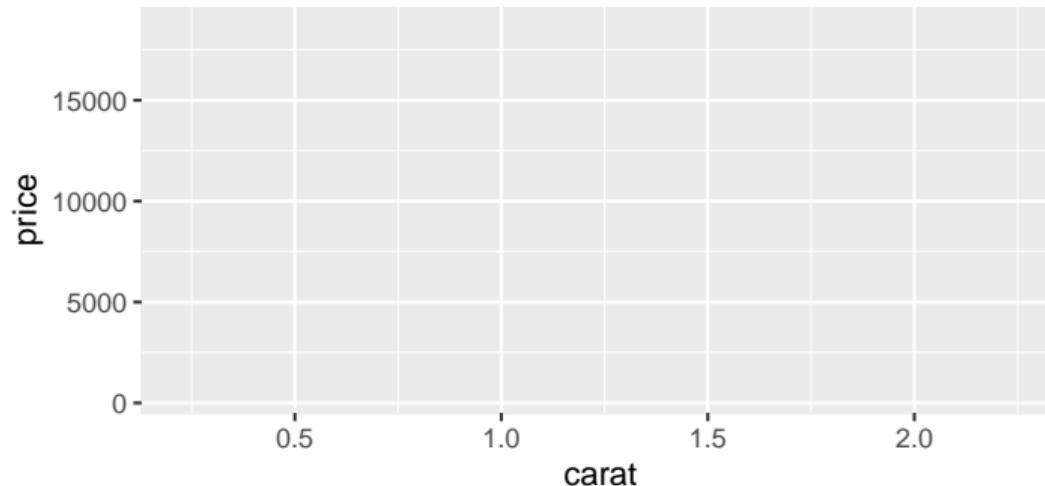
```
> ggplot(ds) # empty plot
```



Starting with ggplot2 (cont.)

We want a scatterplot of carat vs price

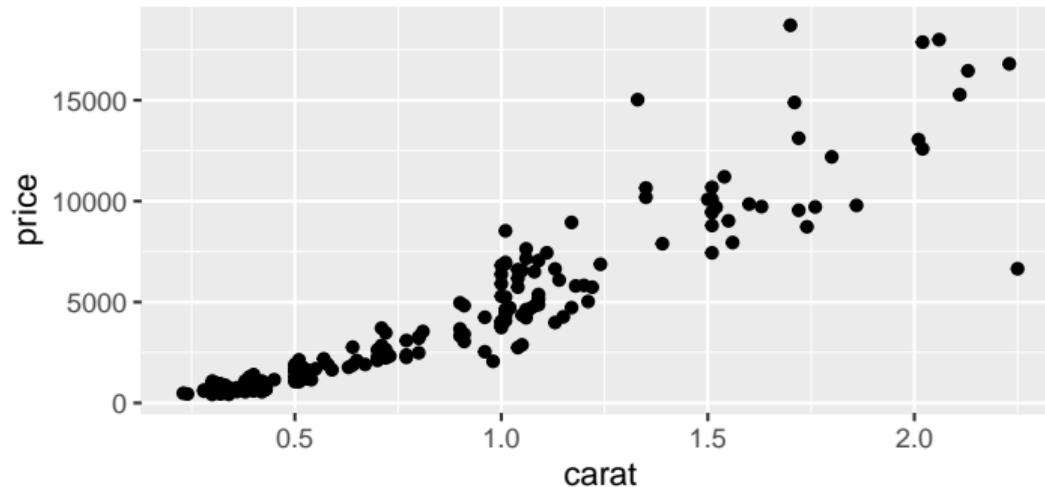
```
> ggplot(ds, aes(x = carat, y = price)) # map aesthetics
```



Starting with ggplot2 (cont.)

We want a scatterplot of carat vs price

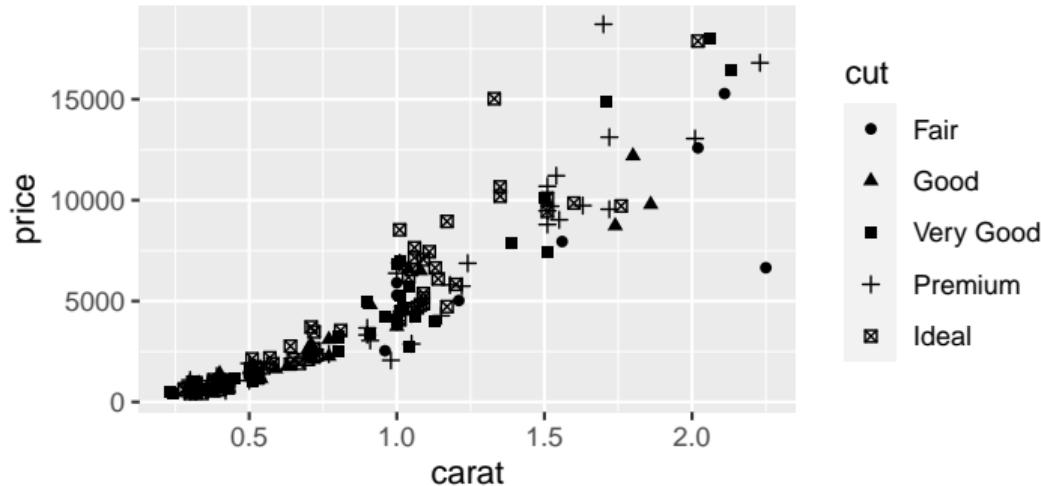
```
> ggplot(ds, aes(x = carat, y = price)) + geom_point() # add objects
```



Starting with ggplot2 (cont.)

Now we want the points to be shaped by cut:

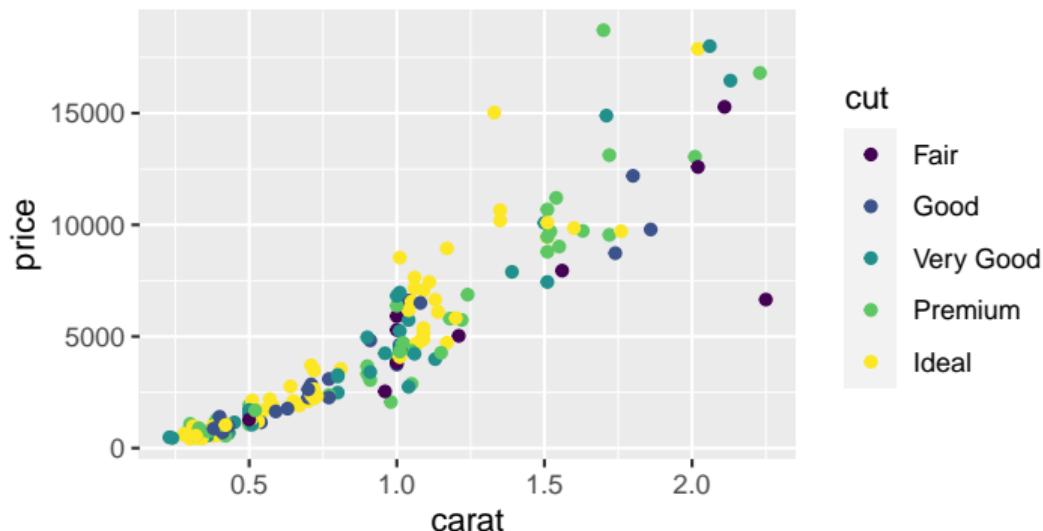
```
> g = ggplot(ds, aes(x = carat, y = price, shape = cut))  
> g + geom_point()
```



Starting with ggplot2 (cont.)

Alternatively let's map cut to the colour aesthetic:

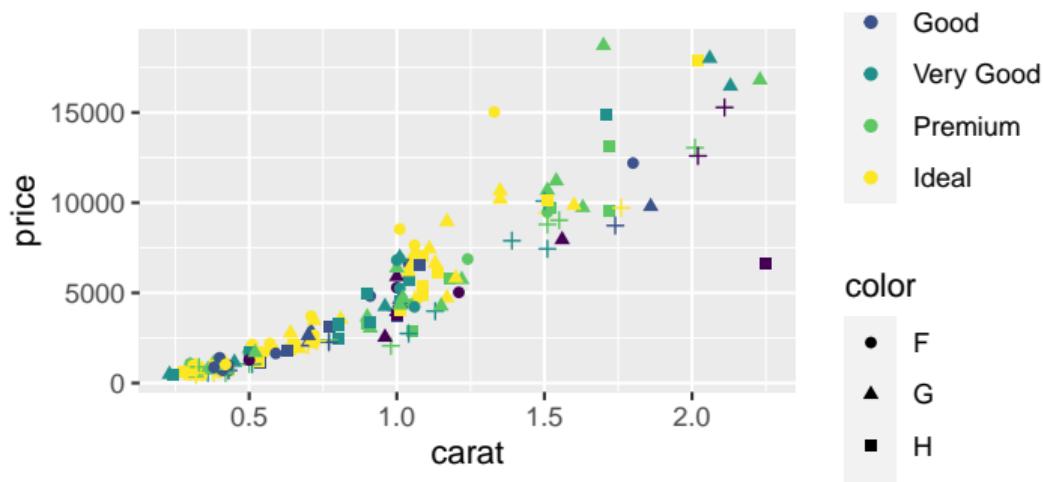
```
> g = ggplot(ds, aes(x = carat, y = price, colour = cut))  
> g + geom_point()
```



Starting with ggplot2 (cont.)

Now we want to distinguish between cut and color.

```
> g = ggplot(ds, aes(x = carat, y = price, colour = cut, shape = color))
> g + geom_point()
```



Starting with ggplot2 (cont.)

Let's dive in the syntax of ggplot2

- ▶ Use the `ggplot` command to pass *data* and set *aesthetics*

```
> g = ggplot(data = ds, mapping = aes(x = carat, y = price, colour = cut))
```

- ▶ Produces **plot object**, i.e., special R object with class attribute `ggplot`
- ▶ First parameter is the source data to display
- ▶ Second parameter is the aesthetics mapping, i.e., binding variables to visual properties
- ▶ Cannot be displayed without adding at least one **layer**

Starting with ggplot2 (cont.)

Let's dive in the syntax of ggplot2 (cont.)

- ▶ Add (*geometric* or *statistical*) layers to the base, e.g.,

```
> g = g + geom_point()  
> g = g + geom_point(colour = "tomato", size = 2.1)
```

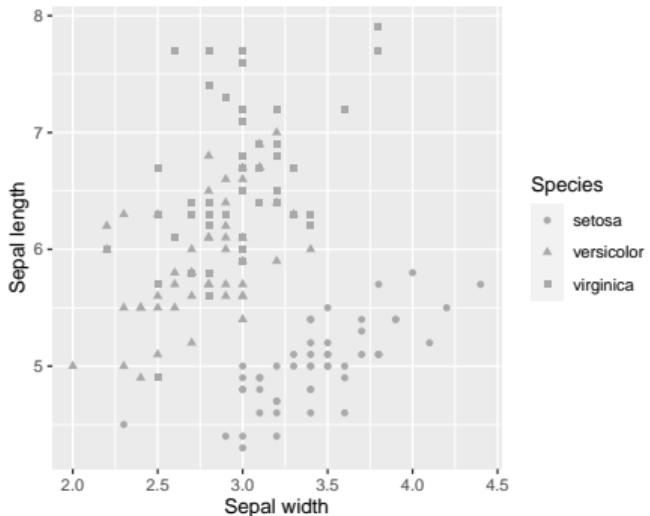
- ▶ The `+` is here of course an overloaded version of the plus operation for objects of type *ggplot*
- ▶ There is a vast number of possible geoms, e.g., lines, bars, polygons, text, ...
- ▶ Each geometric layer can take its own aesthetics mapping (valid just for this layer)

Exercises

1. Install the `ggplot2` package on your system
2. Have a look at the `iris` dataset (`data(iris)`) and check its documentation.
Generate a scatterplot of `Sepal.Width` vs. `Sepal.Length`
3. Change the colour of the points to gray
4. Find out how to change the axis labels
5. Assign distinct shapes to the points by `Species`

Sample solutions

```
> data(iris) # access docs with ?iris  
> g = ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length))  
> g = g + geom_point(aes(shape = Species), color = "darkgray")  
> g + labs(x = "Sepal width", y = "Sepal length")
```



Facetting

Starting with ggplot2 - Facets (cont.)

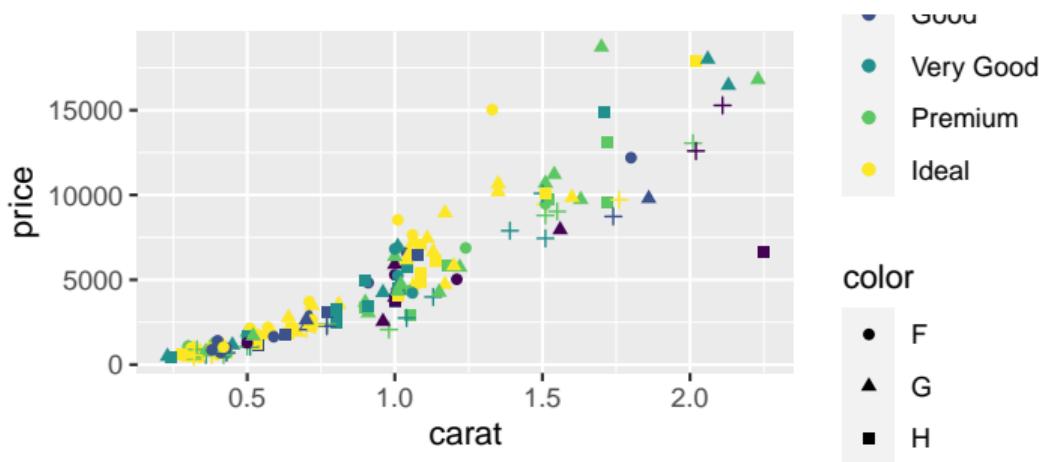
Facets

- ▶ Split data into subsets by one or two categorical variable(s)
- ▶ Render a separate plot for each subset
- ▶ Function `facet_grid(formula, ...)` for crossing of one grouping variable
- ▶ Function `facet_wrap(formula, ...)` for crossing two grouping variables
- ▶ **Keep in mind:** for categorical variables with n and m respectively factors levels you get a $m \times n$ matrix of panels

Starting with ggplot2 (cont.)

Remember that plot?

```
> g = ggplot(ds, aes(x = carat, y = price, colour = cut, shape = color))
> g + geom_point()
```

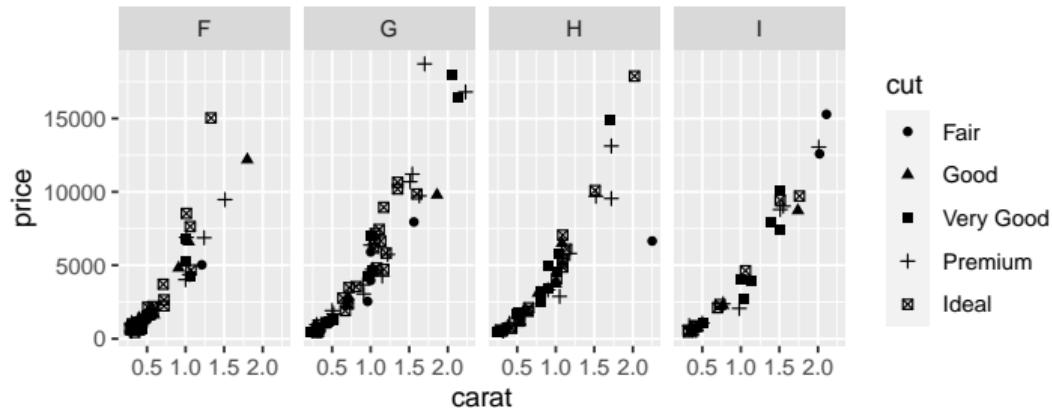


Confusing to recognize some structure here: too many colors and symbols.

Starting with ggplot2 - Facets

So let's try **facetting** and split our data according to color:

```
> g = ggplot(ds, aes(x = carat, y = price, shape = cut))  
> g + geom_point() + facet_grid(. ~ color)
```

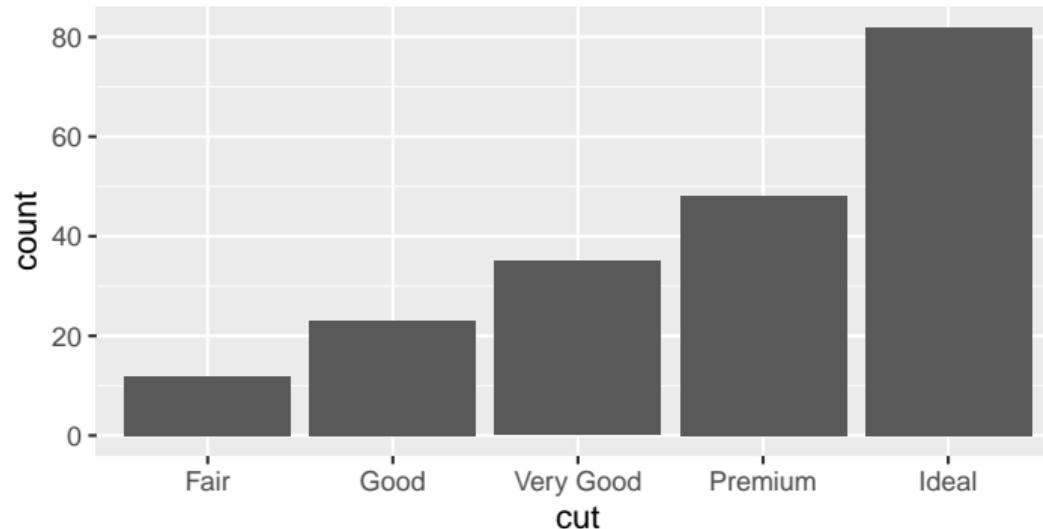


Basic plots

Basic plots - barplot

Check out the number of different *cuts* in the diamonds dataset:

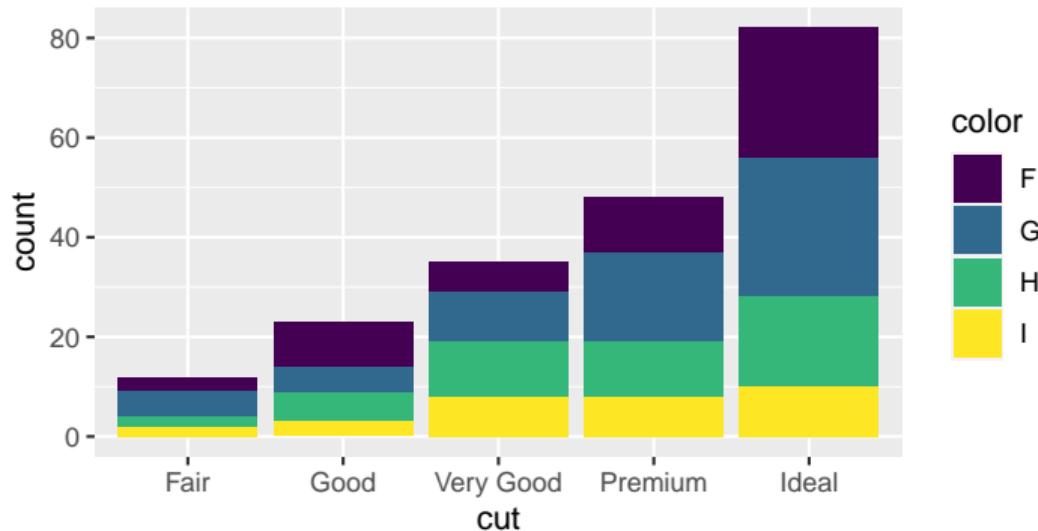
```
> ggplot(ds, aes(x = cut)) + geom_bar()
```



Starting with ggplot (cont.)

What about *cuts* and *color*?

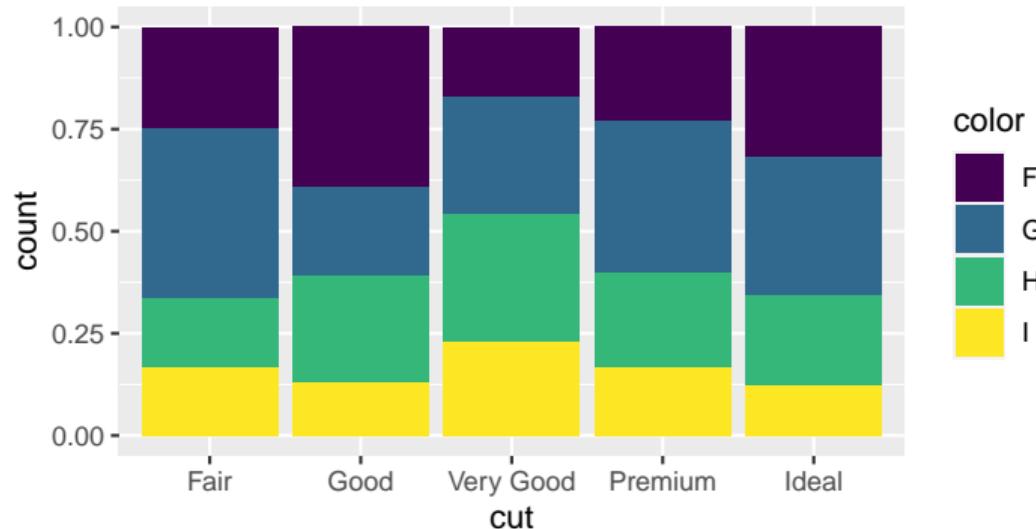
```
> ggplot(ds, aes(x = cut, fill = color)) + geom_bar()
```



Starting with ggplot (cont.)

Want the proportions instead of the absolute values?

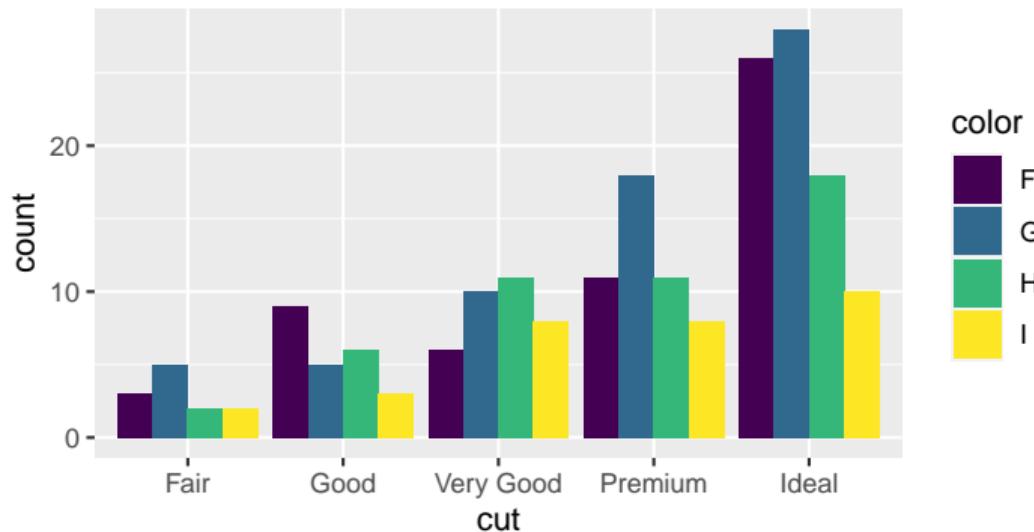
```
> ggplot(ds, aes(x = cut, fill = color)) + geom_bar(position = "fill")
```



Starting with ggplot (cont.)

Maybe we prefer to draw the bars side by side?

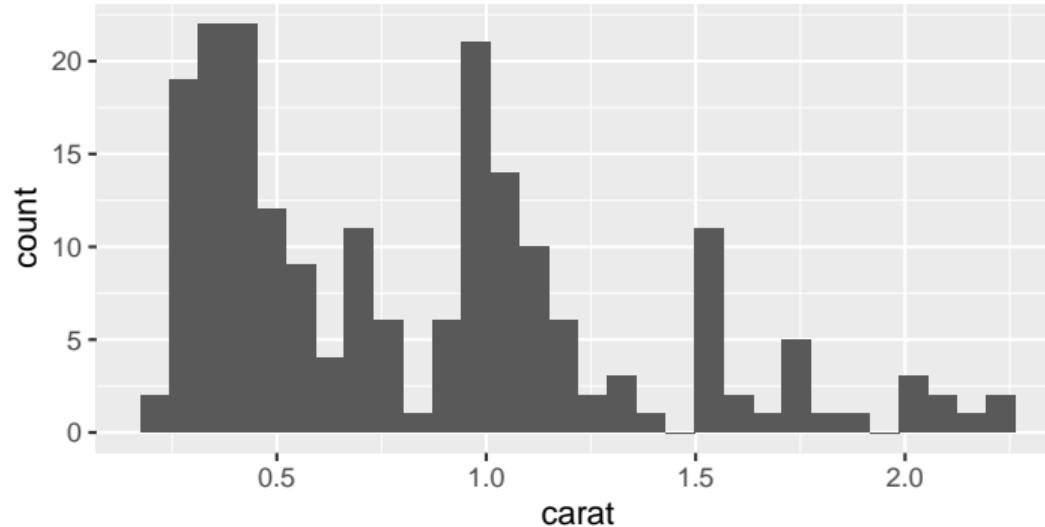
```
> ggplot(ds, aes(x = cut, fill = color)) + geom_bar(position = "dodge")
```



Basic plots - Histogram

Check out the distribution of the `carat` in the diamonds dataset:

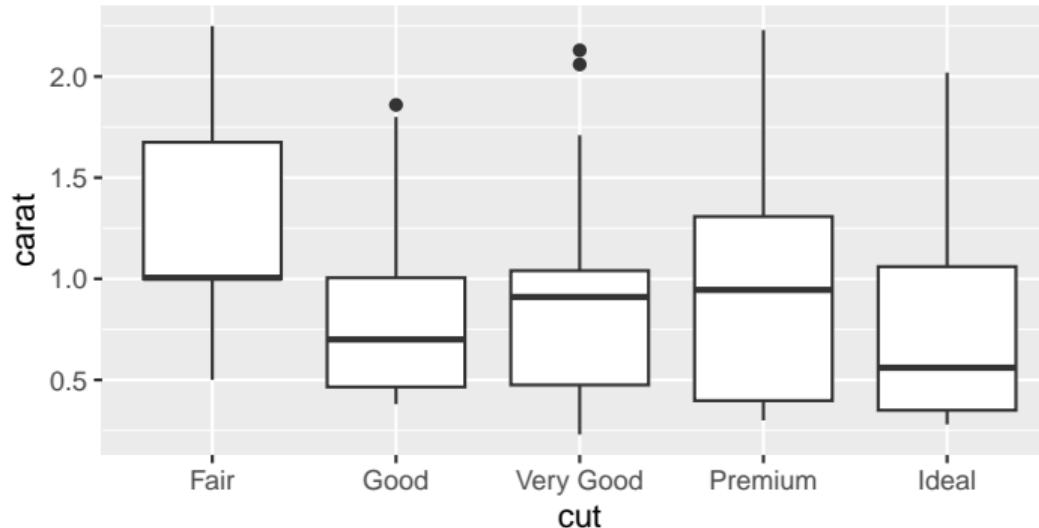
```
> ggplot(ds, aes(x = carat)) + geom_histogram()  
## 'stat_bin()' using 'bins = 30'. Pick better value with 'binwidth'.
```



Basic plots - Histogram

Check out the distribution of *carat* by *cut*:

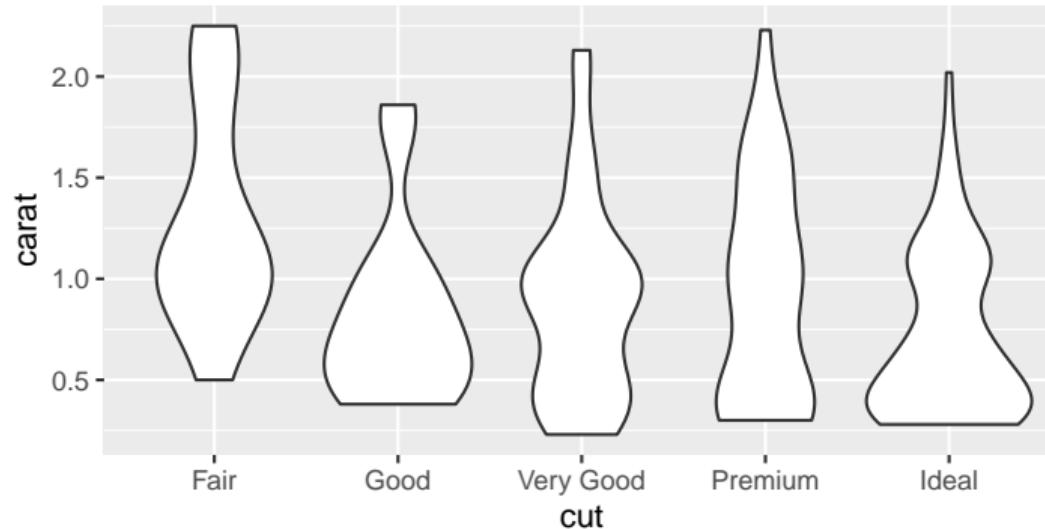
```
> ggplot(ds, aes(x = cut, y = carat)) + geom_boxplot()
```



Basic plots - Histogram

Check out the distribution of *carat* by *cut*:

```
> ggplot(ds, aes(x = cut, y = carat)) + geom_violin()
```



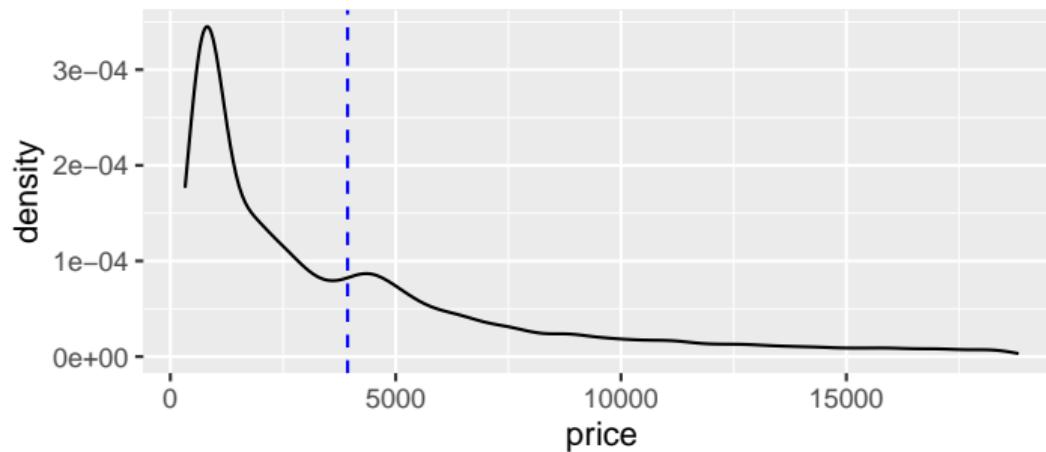
Exercises

1. Visualize the entire price distribution of diamonds with a density curve (`geom_density()`). Add a blue vertical dashed line indicating the mean value of the entire sample (`geom_vline()`)
2. Now draw individual density curves splitting the data by cut. Try to draw vertical dashed lines indicating the mean value per group
3. Next, draw individual density curves splitting the data by cut, color and/or clarity (use facetting and mapping to colors). Hint: you might want to use only a subset of the data

Sample solutions i

1. Visualize the entire price distribution of diamonds with a density curve (`geom_density()`). Add a blue vertical dashed line indicating the mean value of the entire sample (`geom_vline()`)

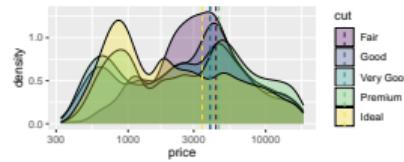
```
> g = ggplot(diamonds, aes(x = price)) + geom_density()  
> g + geom_vline(xintercept = mean(diamonds$price), linetype = "dashed", color = "blue")
```



Sample solutions ii

2. Now draw individual density curves splitting the data by cut. Try to draw vertical dashed lines indicating the mean value per group.

```
> means = diamonds %>%
+   group_by(cut) %>%
+   summarize(price = mean(price)) %>%
+   ungroup()
> g = ggplot(diamonds, aes(x = price))
> g = g + geom_density(aes(fill = cut), alpha = 0.3)
> g = g + scale_x_log10() # transform x-axis
> g + geom_vline(data = means, mapping = aes(xintercept = price, color = cut),
+   linetype = "dashed")
```



Sample solutions iii

3. Next, draw individual density curves splitting the data by cut, color and/or clarity (use facetting and mapping to colors).

```
> # plot gets really large and is thus not displayed here
> g = ggplot(
+   filter(diamonds, color %in% c("E", "I") & clarity %in% c("SI1", "SI2")))
> g = g + geom_density(aes(x = price, fill = cut), alpha = 0.3)
> g = g + facet_grid(color ~ clarity)
>
> # alternative: use interaction to "link" factors
> g = ggplot(
+   filter(diamonds, color %in% c("E", "I") & clarity %in% c("SI1", "SI2")))
> g = g + geom_density(aes(x = price, fill = cut), alpha = 0.3)
> g = g + facet_grid(. ~ interaction(color, clarity, sep = " / "))
```

Themes

Themes

ggplot2 offers reasonable defaults for as good as every aspect of a plot, but . . .

- ▶ sometimes you need to stick to a specific style-guide (e.g. black & white)

Themes

ggplot2 offers reasonable defaults for as good as every aspect of a plot, but . . .

- ▶ sometimes you need to stick to a specific style-guide (e.g. black & white)
- ▶ you want to fine-tune e.g. the axis breaks, legend position, . . .

Themes

ggplot2 offers reasonable defaults for as good as every aspect of a plot, but . . .

- ▶ sometimes you need to stick to a specific style-guide (e.g. black & white)
- ▶ you want to fine-tune e.g. the axis breaks, legend position, . . .
- ▶ you want to increase the axis label size

Themes

ggplot2 offers reasonable defaults for as good as every aspect of a plot, but . . .

- ▶ sometimes you need to stick to a specific style-guide (e.g. black & white)
- ▶ you want to fine-tune e.g. the axis breaks, legend position, . . .
- ▶ you want to increase the axis label size
- ▶ need to increase plot/panel margins

Themes

ggplot2 offers reasonable defaults for as good as every aspect of a plot, but . . .

- ▶ sometimes you need to stick to a specific style-guide (e.g. black & white)
- ▶ you want to fine-tune e.g. the axis breaks, legend position, . . .
- ▶ you want to increase the axis label size
- ▶ need to increase plot/panel margins
- ▶ you do not like the default colors and want to apply your own colour palette

Themes

ggplot2 offers reasonable defaults for as good as every aspect of a plot, but . . .

- ▶ sometimes you need to stick to a specific style-guide (e.g. black & white)
- ▶ you want to fine-tune e.g. the axis breaks, legend position, . . .
- ▶ you want to increase the axis label size
- ▶ need to increase plot/panel margins
- ▶ you do not like the default colors and want to apply your own colour palette
- ▶ etc.

Themes (cont.)

Themes

- ▶ **Themes** control the *non-data parts* of your plots

Themes (cont.)

Themes

- ▶ **Themes** control the *non-data parts* of your plots
- ▶ They are composed of multiple elements of different types, e.g `plot.title` [text], `legend.background` [rect]

Themes (cont.)

Themes

- ▶ **Themes** control the *non-data parts* of your plots
- ▶ They are composed of multiple elements of different types, e.g `plot.title` [text], `legend.background` [rect]
- ▶ Two possible approaches (which can be combined)

Themes (cont.)

Themes

- ▶ **Themes** control the *non-data parts* of your plots
- ▶ They are composed of multiple elements of different types, e.g plot.title [text], legend.background [rect]
- ▶ Two possible approaches (which can be combined)
 - ▶ **Apply predefined theme:** e.g. theme_bw() or theme_grey()

```
> # change every aspect of the rendering  
> last_plot() + theme_bw()
```

Themes (cont.)

Themes

- ▶ **Themes** control the *non-data parts* of your plots
- ▶ They are composed of multiple elements of different types, e.g `plot.title` [text], `legend.background` [rect]
- ▶ Two possible approaches (which can be combined)
 - ▶ **Apply predefined theme:** e.g. `theme_bw()` or `theme_grey()`

```
> # change every aspect of the rendering
> last_plot() + theme_bw()
```

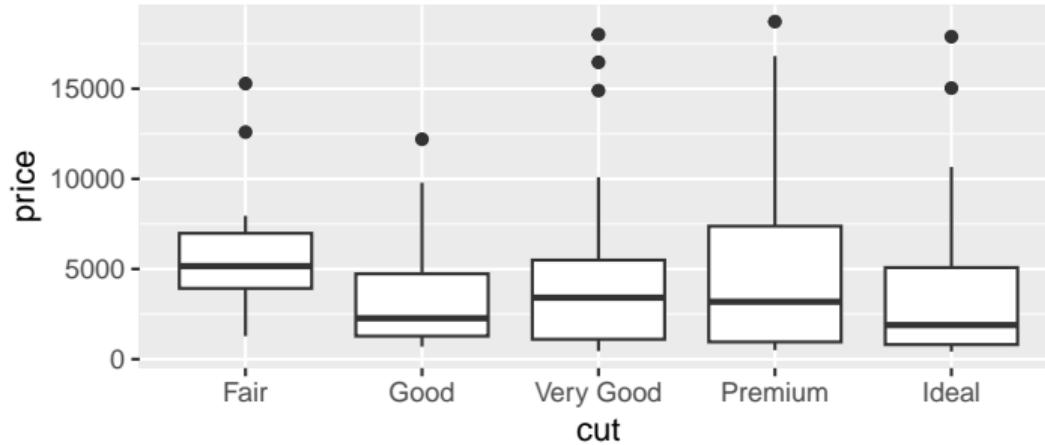
- ▶ **Change only specific parts:** e.g.

```
> # change position of the legend and the colour of the plot title
> last_plot() + theme(
+   legend.position = "top",
+   plot.title = element_text(colour = "tomato"))
```

Themes (cont.)

Basic theme

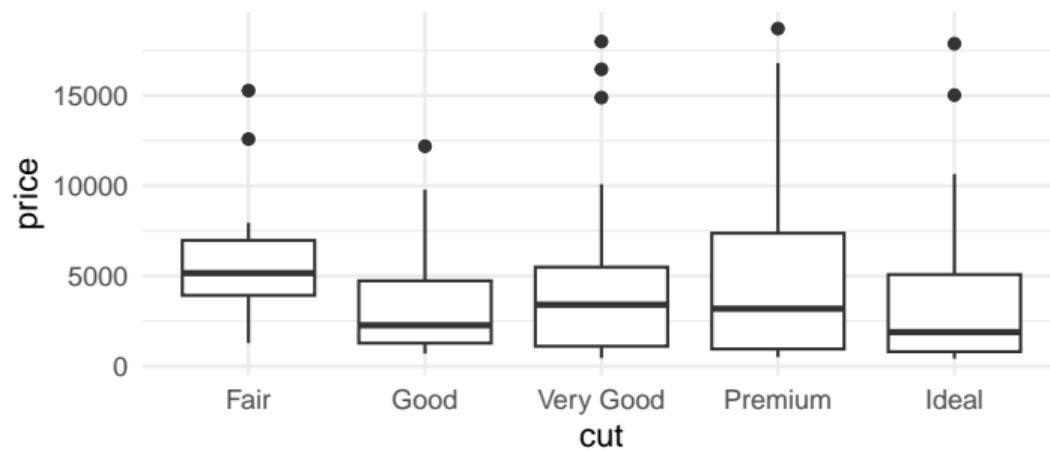
```
> g = ggplot(ds, aes(x = cut, y = price)) + geom_boxplot()  
> g
```



Themes (cont.)

Predefined minimalistic theme

```
> last_plot() + theme_minimal()
```



Themes (cont.)

Specific theming

```
> last_plot() +  
+   labs(title = "Distribution of prices") +  
+   theme(plot.title = element_text(colour = "gray", size = 13))
```



Themes (cont.)

Creating a custom theme

If there is no theme that fits your imagination (or style-guide), build your own:

```
> my_theme = function() {  
+   theme(  
+     legend.position = "top",  
+     panel.background = element_rect(fill = NA, colour = NA),  
+     plot.background = element_rect(colour = NA, fill = NA)  
+   #, ...  
+ }  
+
```

Set it globally for the entire session:

```
> theme_set(my_theme())
```

How to save a plot

Here comes ggsave(...)

- ▶ Convenient function for saving a plot (wrapper for graphic device driver)
- ▶ Only mandatory argument to ggsave is the **file name**⁵
- ▶ By default saves the last generated plot (determined via `last_plot()`)

Example

Save plot myplot in the current working directory

```
> ggsave("figure.pdf", plot = myplot)
```

⁵ Other stuff like image size is guessed in a reasonable way.

How to save a plot

More examples

- ▶ Save the last plot generated (maybe the one on your screen)

```
> ggsave("figure.pdf")
```

- ▶ Save arbitrary ggplot object myplot in memory

```
> ggsave("figure.pdf", plot = myplot)
```

- ▶ Select device driver which should be used

```
> ggsave("figure.png")
```

```
> ggsave("figure.eps")
```

- ▶ Device is determined by the file extension `figure.eps`
- ▶ build-in support for all common graphic types, i. e., pdf, eps, png, jpeg, svg, ...)
- ▶ see `?ggsave` for more information

Case study

Case study 1

Q: What is the number of diamonds per color for the two best cut values?

Case study 1

Q: What is the number of diamonds per color for the two best cut values?
First, let's do the aggregation:

```
> data(diamonds)
> X = diamonds %>%
+   filter(cut %in% c("Ideal", "Premium")) %>%
+   group_by(cut, color) %>%
+   summarize(n = n()) %>%
+   ungroup() %>%
+   mutate(perc = n / sum(n))

## 'summarise()' has grouped output by 'cut'. You can override using the '.groups'
## argument.

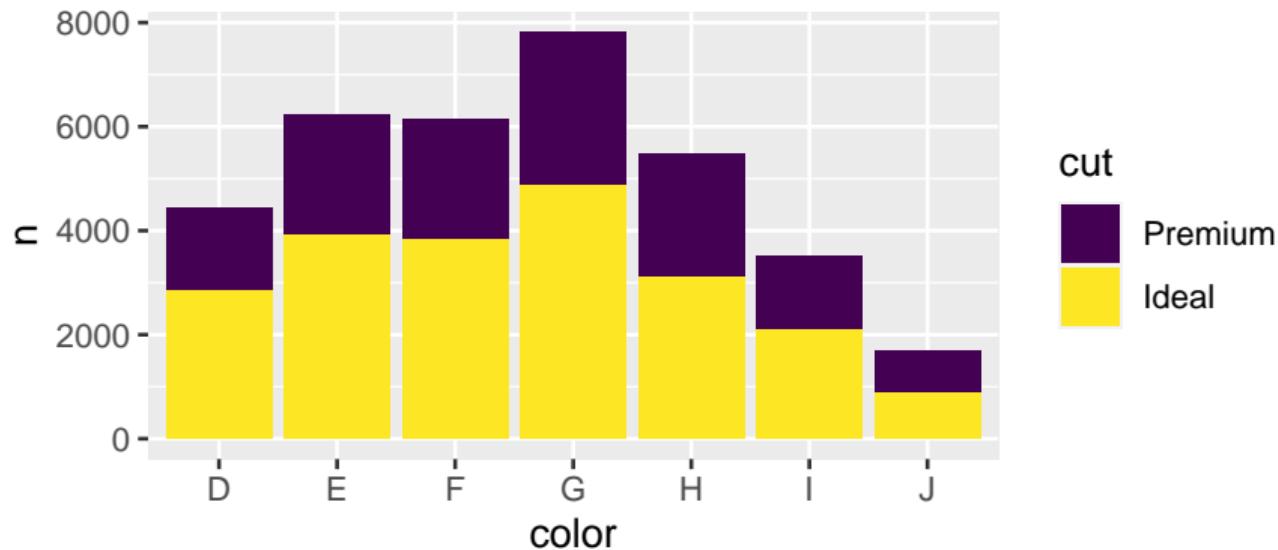
> print(X, n = 4)

## # A tibble: 14 x 4
##   cut     color     n     perc
##   <ord>    <ord> <int>  <dbl>
## 1 Premium D      1603 0.0454
## 2 Premium E      2337 0.0661
```

Case study 1 (cont.)

Now let's generate a basic bar plot and refine it iteratively:

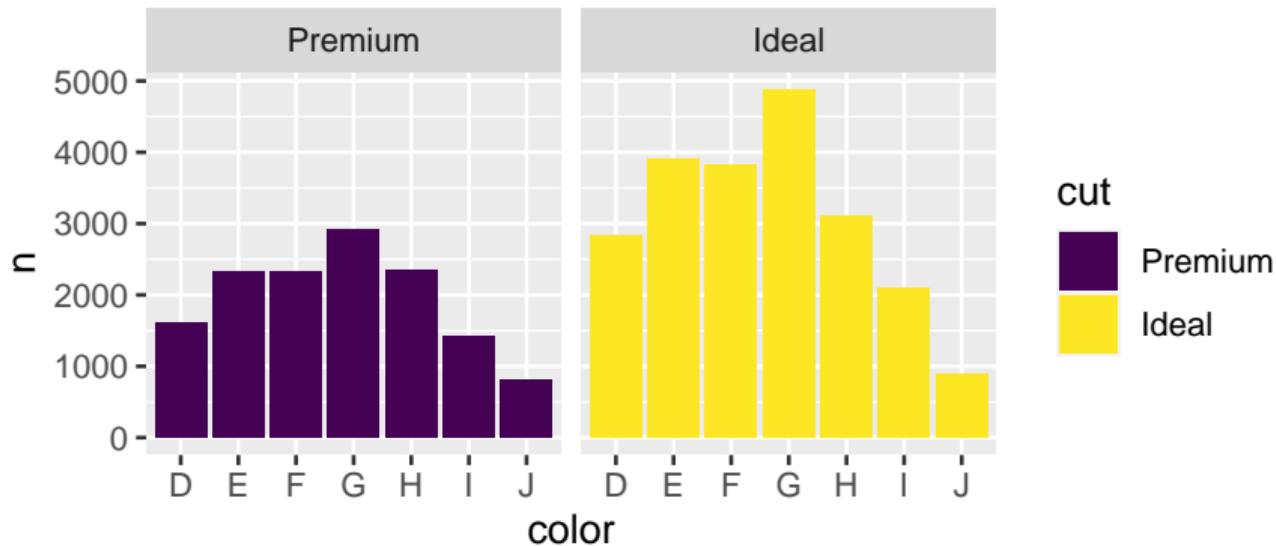
```
> g = ggplot(X) + aes(x = color, y = n, fill = cut) + geom_col()  
> g
```



Case study 1 (cont.)

Now let's generate a basic bar plot and refine it iteratively:

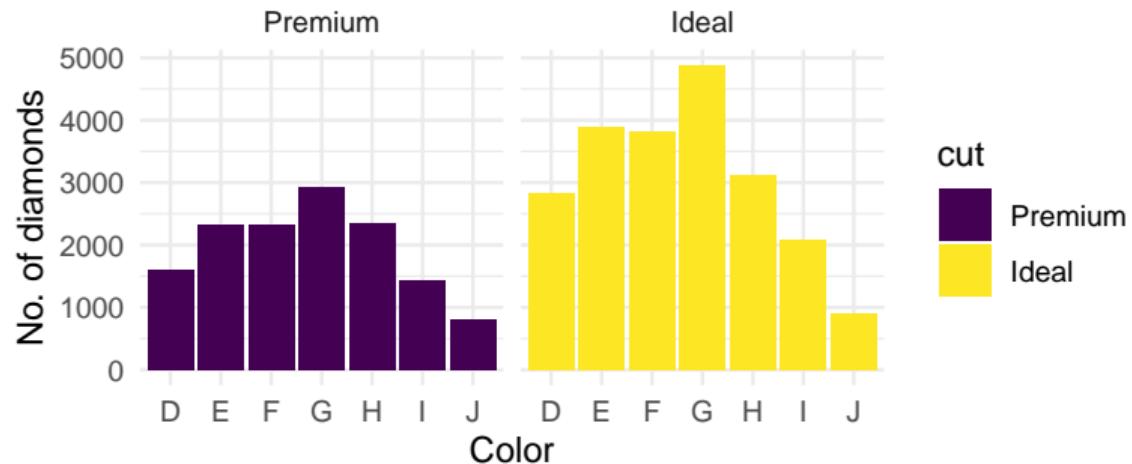
```
> g = g + facet_grid(. ~ cut)  
> g
```



Case study 1 (cont.)

Now let's generate a basic bar plot and refine it iteratively:

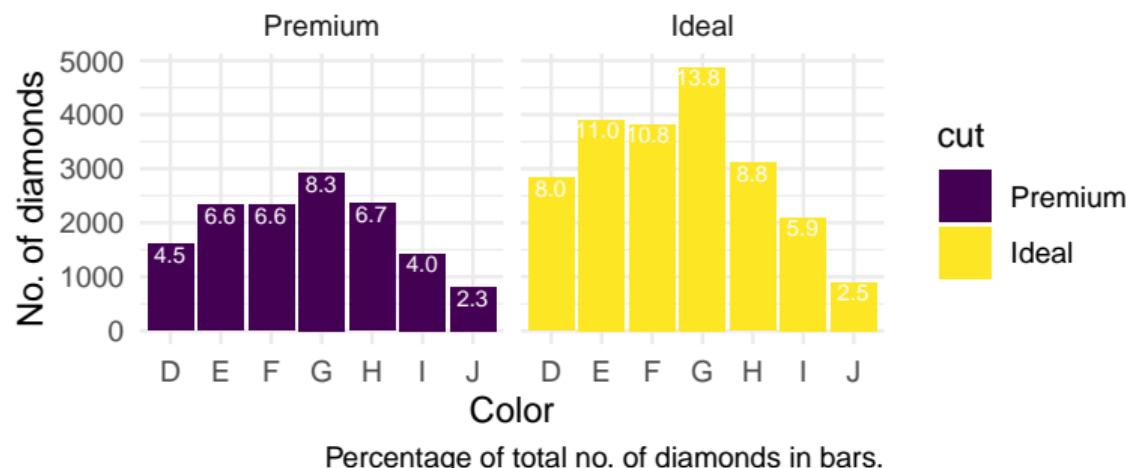
```
> g = g + theme_minimal() + labs(x = "Color", y = "No. of diamonds")  
> g
```



Case study 1 (cont.)

Now let's generate a basic bar plot and refine it iteratively:

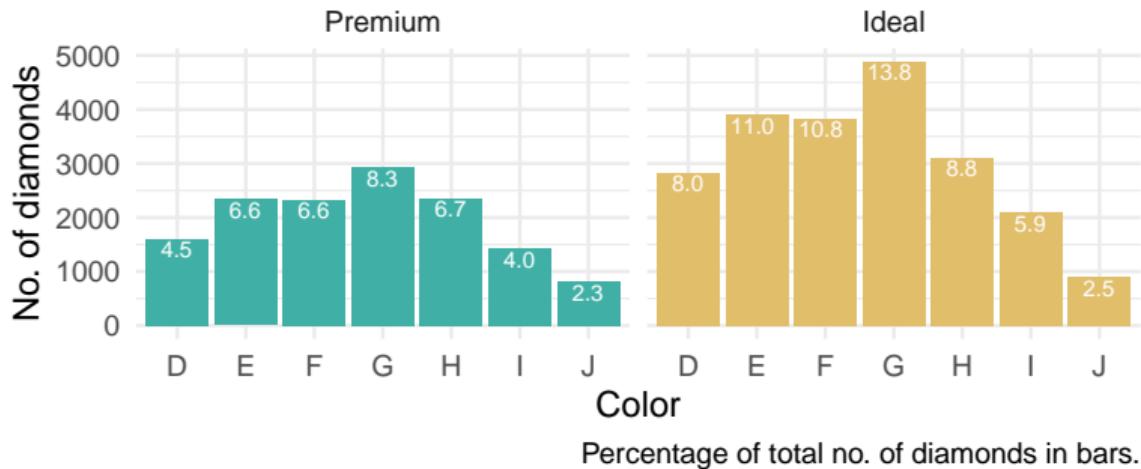
```
> g = g + geom_text(  
+   mapping = aes(label = format(100 * perc, digits = 2)),  
+   size = 2.5, vjust = 1.15, hjust = 0.6, color = 'white')  
> g = g + labs(caption = "Percentage of total no. of diamonds in bars.")  
> g
```



Case study 1 (cont.)

Now let's generate a basic bar plot and refine it iteratively:

```
> g = g + scale_fill_manual(  
+   values = c("Ideal"  = "#E1BE6A", "Premium" = "#40B0A6")) +  
+   theme(legend.position = "none")  
> g
```



Case study 2

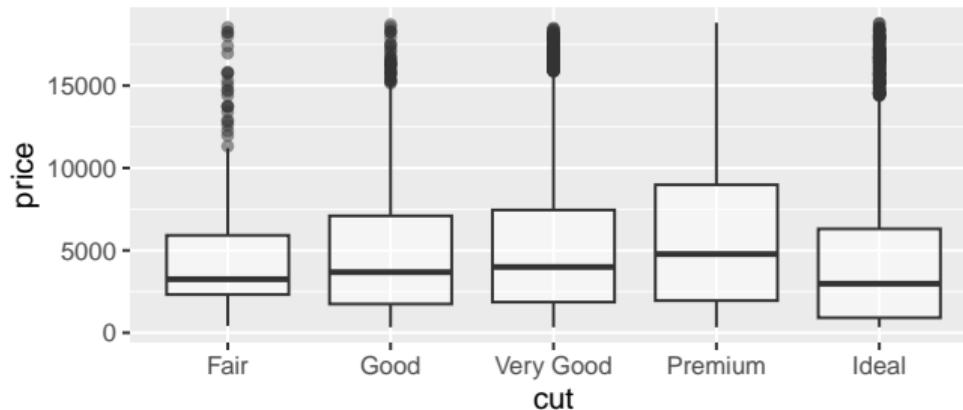
Q: What is the distribution of the price for all diamonds with color I or J separated by cut?

Case study 2

Q: What is the distribution of the price for all diamonds with color I or J separated by cut?

Again, first we generate a basic plot:

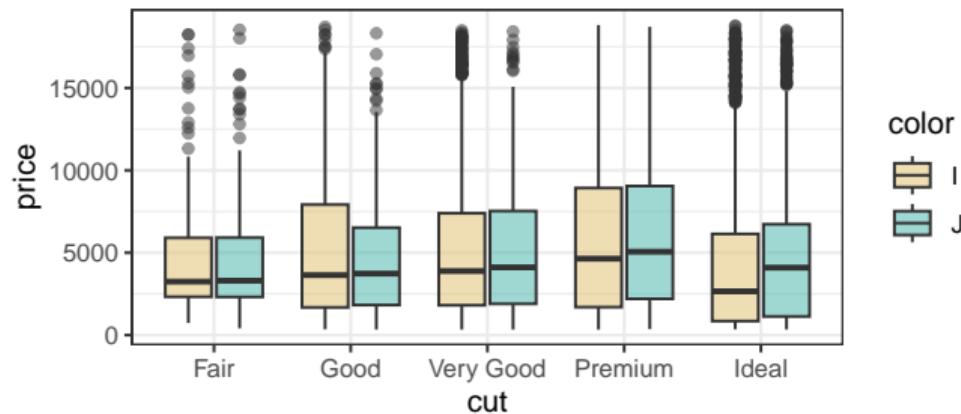
```
> g = ggplot(filter(diamonds, color %in% c("I", "J")))
> g + aes(x = cut, y = price) + geom_boxplot(alpha = 0.5)
> g
```



Case study 2 (cont.)

Now let's refine:

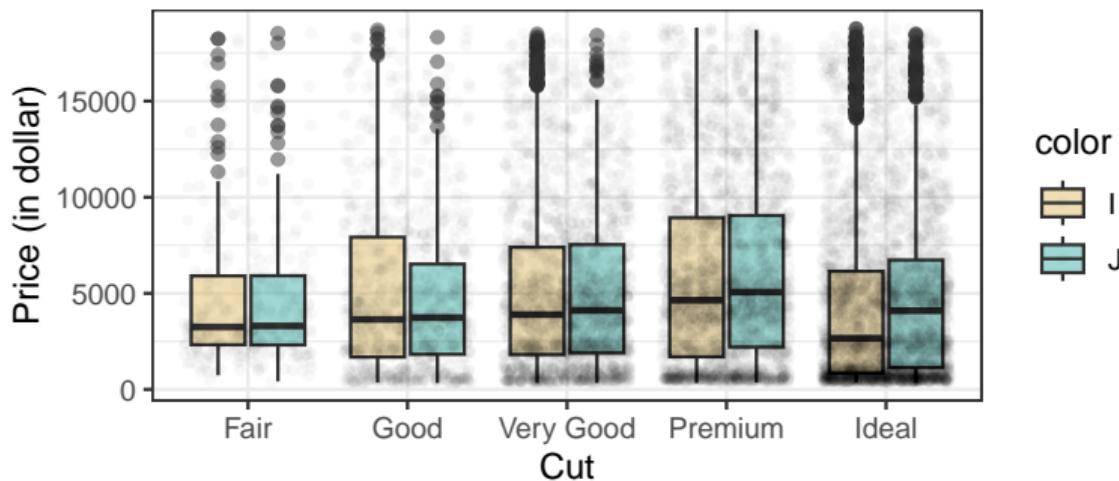
```
> g = g + aes(fill = color)
> g = g + theme(legend.position = "top") + theme_bw()
> g = g + scale_fill_manual(values = c("I" = "#E1BE6A", "J" = "#40B0A6"))
> g
```



Case study 2 (cont.)

Let's add the actual observations:

```
> g = g + geom_jitter(aes(fill = NULL), alpha = 0.03, size = 1)
> g = g + xlab("Cut") + ylab("Price (in dollar)")
> g
```



Quick plots

Quick plots

Shortcut and consistent interface for users familiar with base R's `plot()` function:

```
> # qplot has some more arguments, but these are the most important ones  
> qplot(x, y, data, facets = NULL, geom = "auto",  
+        xlim = c(NA, NA), ylim = c(NA, NA),  
+        main = NULL, xlab = NULL, ylab = NULL)
```

`x, y` Aesthetics passed into each layer

`data` Optional dataframe (if none is passed it is generated internally).

`facets` Faceting formula (passed down to `facet_grid()` or `facet_wrap()`)

`geom` Geometric as a string, e.g., "boxplot" or "density"

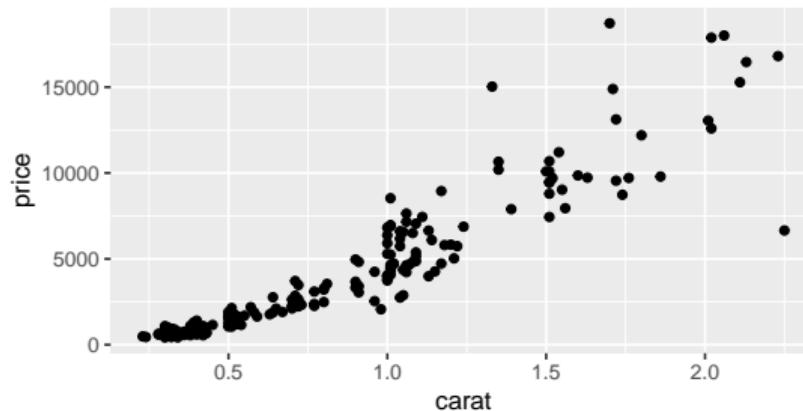
`xlim, ylim` x- and y-axis limits

`main, xlab, ylab` Title and axis labels

Quick plots

Let's look at some examples:

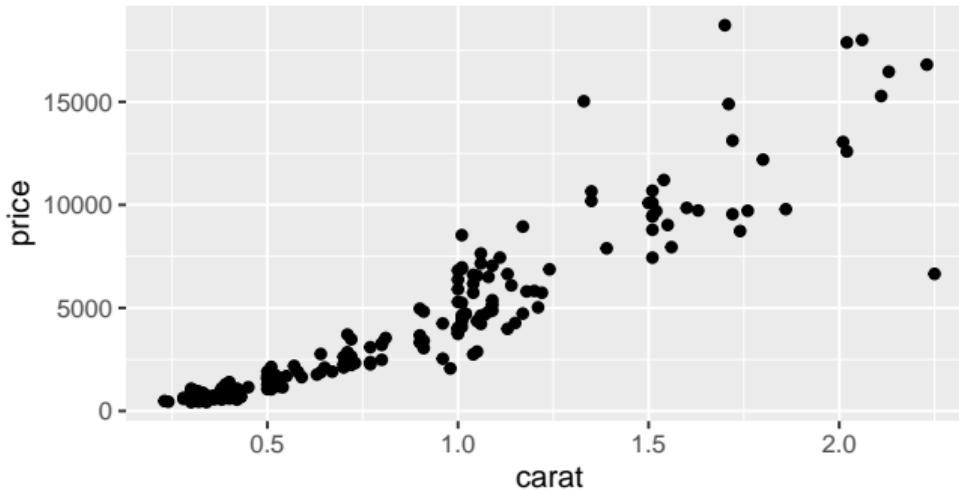
```
> qplot(carat, price, data = ds, geom = "point")  
  
## Warning: `qplot()`' was deprecated in ggplot2 3.4.0.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was  
## generated.
```



Quick plots

Quick plots makes an educated guess if no geom is specified.

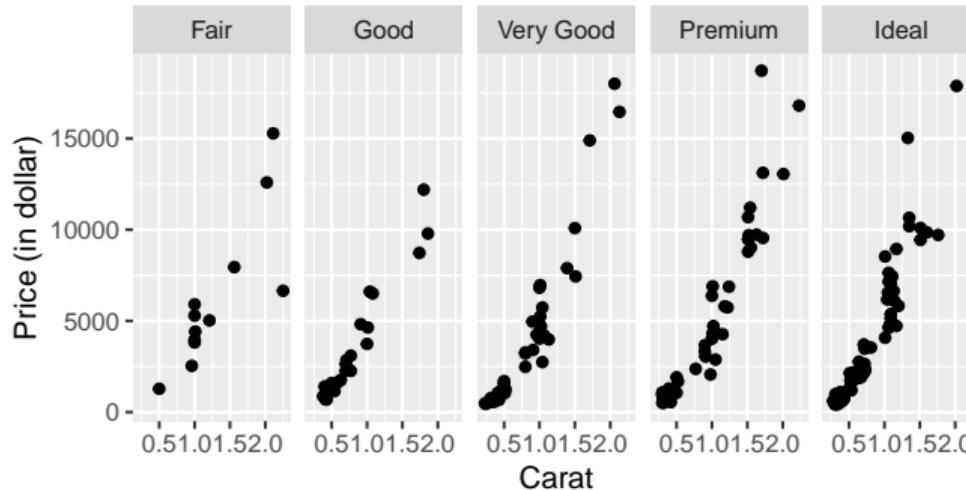
```
> qplot(carat, price, data = ds)
```



Quick plots

Facets are possible as well:

```
> qplot(carat, price, data = ds, facets = . ~ cut,  
+       xlab = "Carat", ylab = "Price (in dollar)")
```



Clean visualizations

Awesome graphics focus on data

How does a "good" (statistical) plot look like?

- ▶ Focus on actually **visualizing** data

Awesome graphics focus on data

How does a "good" (statistical) plot look like?

- ▶ Focus on actually **visualizing data**
- ▶ Get rid of **chart junk**, i. e., every non-data aspect which obscures the data

Awesome graphics focus on data

How does a "good" (statistical) plot look like?

- ▶ Focus on actually **visualizing data**
- ▶ Get rid of **chart junk**, i. e., every non-data aspect which obscures the data
- ▶ Be as minimalistic as possible (but not too minimalistic)

Awesome graphics focus on data

How does a "good" (statistical) plot look like?

- ▶ Focus on actually **visualizing data**
- ▶ Get rid of **chart junk**, i. e., every non-data aspect which obscures the data
- ▶ Be as minimalistic as possible (but not too minimalistic)
- ▶ Use color-blind colour palettes

Awesome graphics focus on data

How does a "good" (statistical) plot look like?

- ▶ Focus on actually **visualizing data**
- ▶ Get rid of **chart junk**, i. e., every non-data aspect which obscures the data
- ▶ Be as minimalistic as possible (but not too minimalistic)
- ▶ Use color-blind colour palettes
- ▶ Be understandable/readable even when printed in black & white (grayscale)

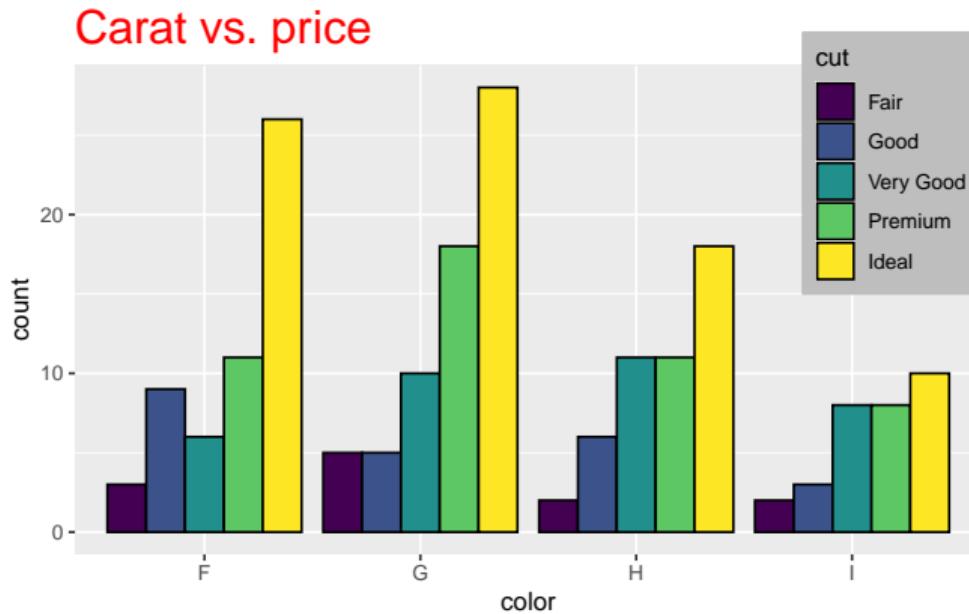
Awesome graphics focus on data

How does a "good" (statistical) plot look like?

- ▶ Focus on actually **visualizing data**
- ▶ Get rid of **chart junk**, i. e., every non-data aspect which obscures the data
- ▶ Be as minimalistic as possible (but not too minimalistic)
- ▶ Use color-blind colour palettes
- ▶ Be understandable/readable even when printed in black & white (grayscale)
- ▶ Fit in the overall picture

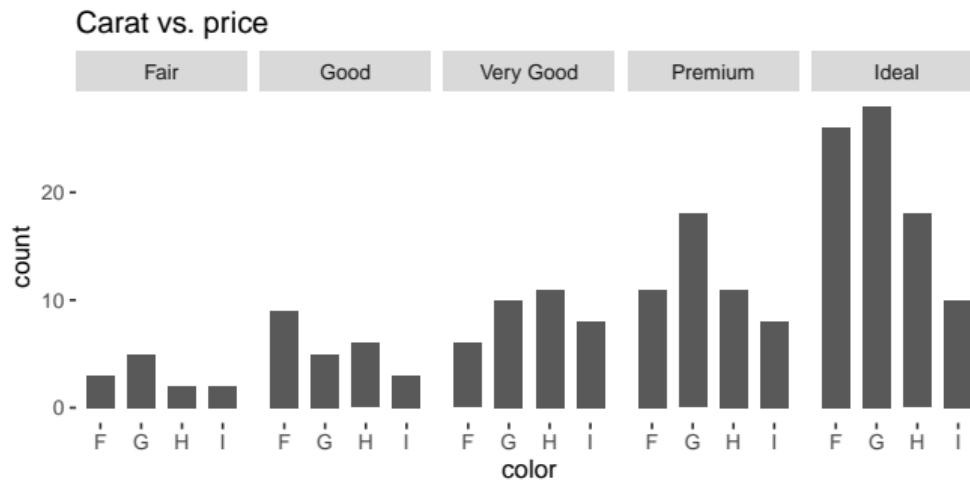
Awesome graphics focus on data (cont.)

Example of a really bad graphic



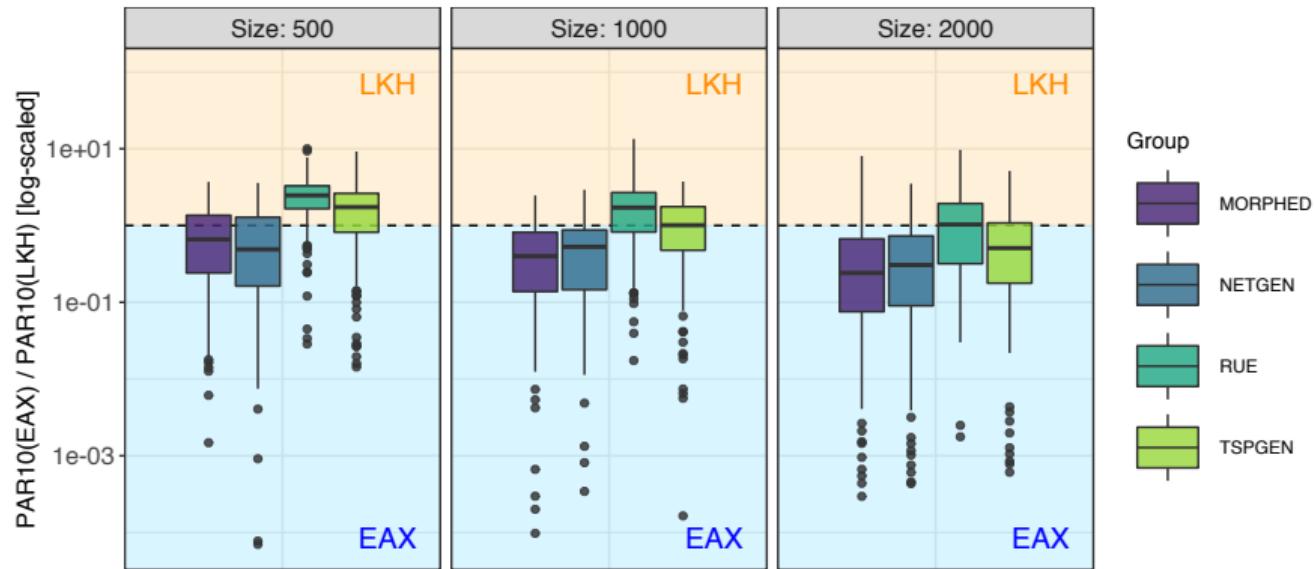
Awesome graphics focus on data (cont.)

Much cleaner with faceting



Example

Remember that one from the beginning?



Example

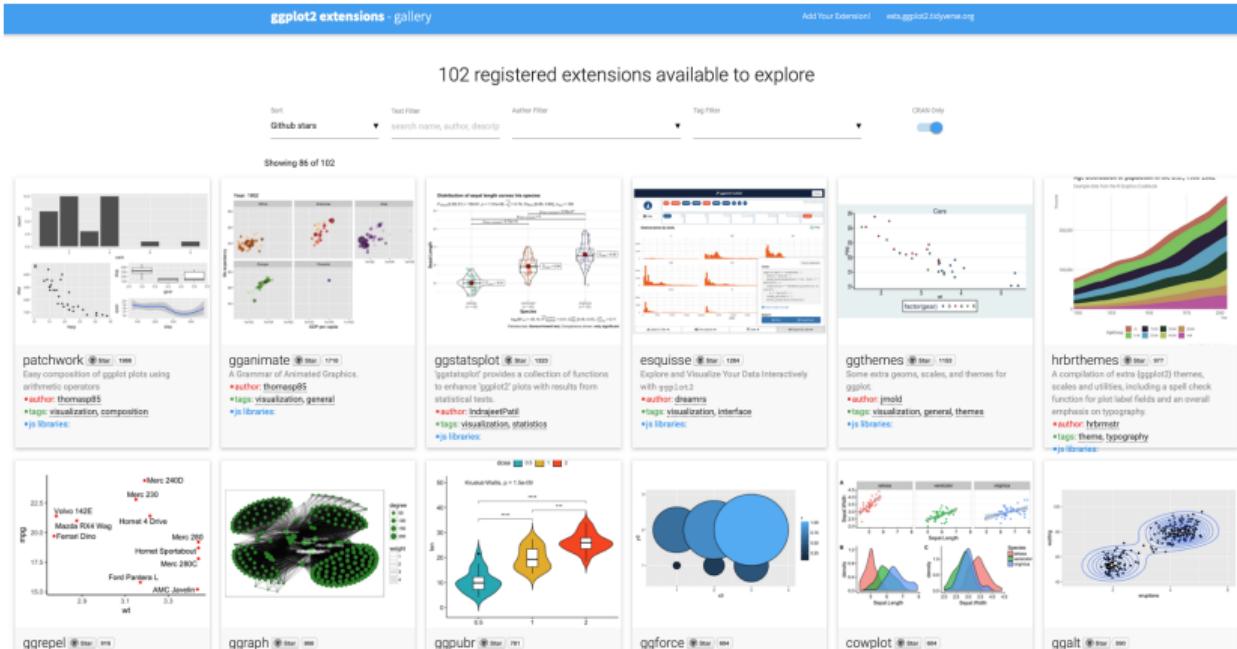
Code can get **lengthy** if you aim to fine-tune many details! 😊

```
> g = ggplot(tmp.df, aes(x = duel, y = ratio, fill = as.factor(group)))
> g = g + geom_blank()
> g = g + annotate("rect", xmin = -Inf, xmax = Inf, ymin = 1, ymax = Inf, alpha = 0.15, fill = "orange")
> g = g + annotate("rect", xmin = -Inf, xmax = Inf, ymin = 0, ymax = 1, alpha = 0.15, fill = "deepskyblue")
> g = g + geom_hline(yintercept = 1, linetype = "dashed")
> g = g + geom_boxplot(alpha = 0.825)
> g = g + scale_y_log10() #limits = c(1/15000, 100)
> g = g + theme_bw()
> g = g + geom_text(data = data.frame(x = c(1.2, 0.7), y = c(1/20000, 700), label = c("EAX", "LKH"), group = NA),
+   mapping = aes(x = x, y = y, label = label), color = c("blue", "darkorange"), size = 5)
> g = g + labs(
+   x = "Solver duel", y = "PAR10(EAX) / PAR10(LKH) [log-scaled]",
+   fill = "Group")
> g = g + viridis::scale_fill_viridis(discrete = TRUE, begin = 0.1, end = 0.85)
> g = g + facet_wrap(. ~ Size, nrow = 1L, labeller = label_both)
> g = g + theme(
+   legend.position = "right",
+   axis.title.y = element_text(size = 12), axis.text.y = element_text(size = 12),
+   axis.title.x = element_blank(), axis.text.x = element_blank(),
+   strip.text = element_text(size = 12),
+   legend.key = element_rect(color = NA, fill = NA), legend.key.size = unit(1.4, "cm"),
+   legend.title = element_text(size = 12),
+   legend.text = element_text(size = 12))
> g = g + guides(fill = guide_legend(nrow = 5))
```

Take away message

- ▶ ggplot2 is a powerful tool for applied data analysis
- ▶ Implementation of the *Grammar of Graphics*
- ▶ Build complex graphics with ease
- ▶ Highly customizable via *themes*
- ▶ Very similar implementations available for other languages
- ▶ Definitely worth the learn effort! ☺

ggplot2 is huge!



Further reading

- ▶ **Hadley's book** (Wickham 2009)⁶ (printed book is nice to have, but almost instantly outdated 😊)
- ▶ **Official documentation**⁷
- ▶ **Extension library**⁸ (my personal favourites: `ggridge`, `geomnet`, `plotly`, `patchwork`, `cowplot`)
- ▶ Chapter on *Exploratory Data Analysis* (EDA) in (Wickham and Grolemund 2017)⁹

⁶ <https://ggplot2-book.org>

⁷ <https://ggplot2.tidyverse.org>

⁸ <https://exts.ggplot2.tidyverse.org>

⁹ <https://r4ds.had.co.nz/exploratory-data-analysis.html>

What comes next?

Overview of the upcoming week:

Date	Content	Speaker
04 Nov.	ggplot2	Jakob
05 Nov.	Tutorial on tidyverse	Raphael
11 Nov.	Exploratory Data Analysis (EDA)	Raphael ¹⁰
12 Nov.	Tutorial on ggplot2	Raphael
18 Nov.	Hierarchical Clustering and k -means	Jakob
19 Nov.	Tutorial on EDA	Raphael
:	:	:

¹⁰ Jakob will be at the TU Dresden.

Wrap-Up

Wrap-Up

Todays content

Grammer of Graphics and its implementation in ggplot2

Wrap-Up

Todays content

Grammer of Graphics and its implementation in ggplot2

Your task(s)

- ▶ Again, we just scratched the surface here! Work through the data transformation chapter in (Wickham and Grolemund 2017)
- ▶ Work on exercise sheet
- ▶ Ideally search for more material online

References |

- Wilkinson, Leland (2005). *The Grammar of Graphics (Statistics and Computing)*. Berlin, Heidelberg: Springer-Verlag. ISBN: 0387245448.
- Tyner, Sam, François Briatte, and Heike Hofmann (2017). "Network Visualization with ggplot2". In: *The R Journal* 9.1, pp. 27–59. DOI: 10.32614/RJ-2017-023.
- Wickham, Hadley (2009). *ggplot2: elegant graphics for data analysis*. Springer New York. ISBN: 978-0-387-98140-6.
- Wickham, Hadley and Garrett Grolemund (Jan. 2017). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 1st ed. O'Reilly Media. ISBN: 1491910399.