Introduction to programming for data analysis

Session 3 — Data Visualization

Ulrich Aiounou
dimitri.aiounou@univ-amu.fr
AMSE, Aix Marseille University

Ewen Gallic
ewen.gallic@univ-amu.fr
AMSE, Aix Marseille University

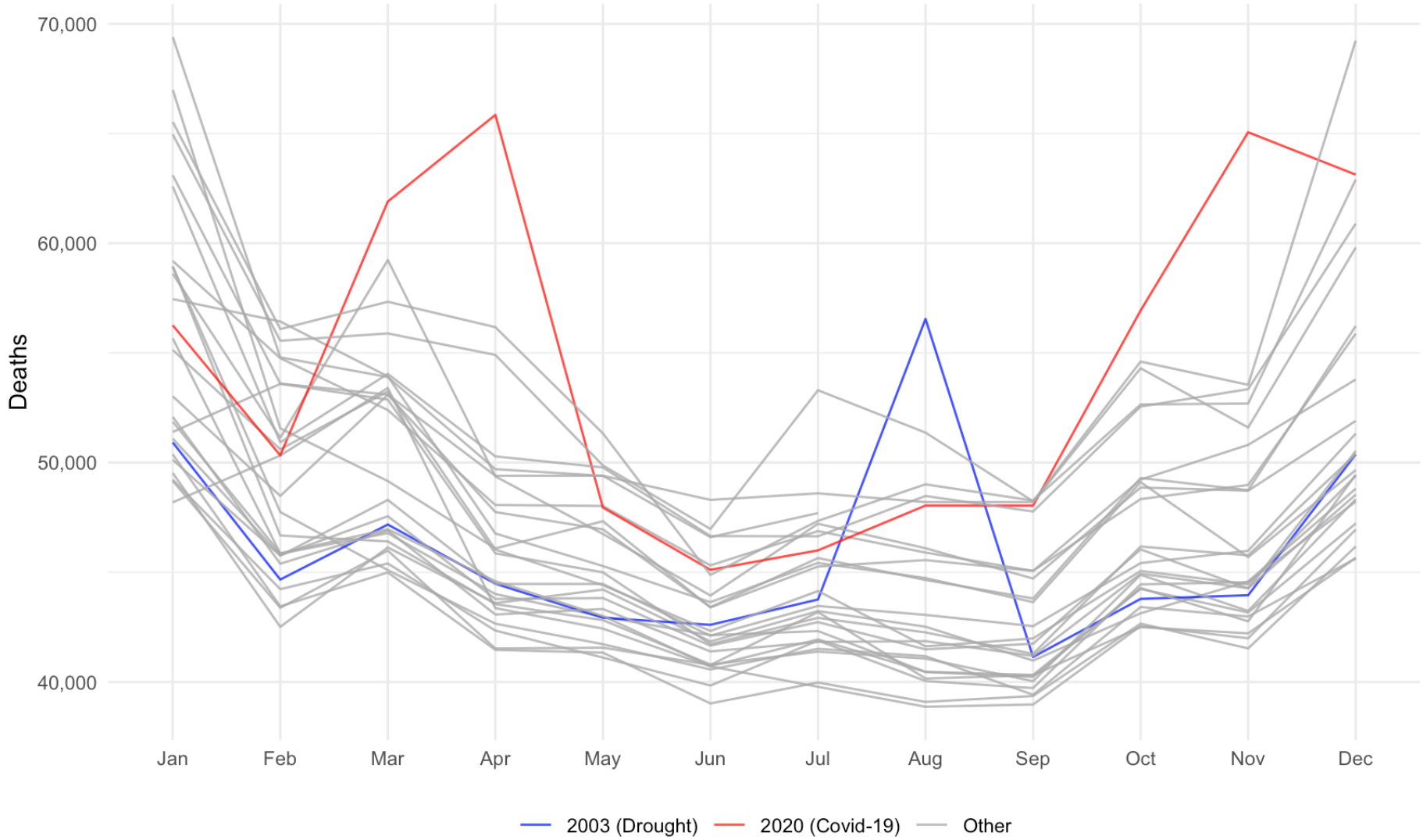
Disclaimer

- The slides were adapted from those of Pierre Michel and Morgan Raux, both researchers at AMSE, who kindly shared their work.
- This slide deck was made with Quarto and reveal.js, it is translated by Claude Sonnet 5 from a LaTex presentation previously made with beamer.

The 4 steps in any data analysis

1. Finding data
2. Cleaning data to combine several data sources
3. **Producing knowledge out from data (summary statistics, regression analysis summarised via tables and graphs)**
4. Interpreting the results (writing)

Example of a graph made with R



Monthly number of deaths in France (2020–2025). Source: Insee (ref.: 000436394). (R Codes)

A grammar of graphics

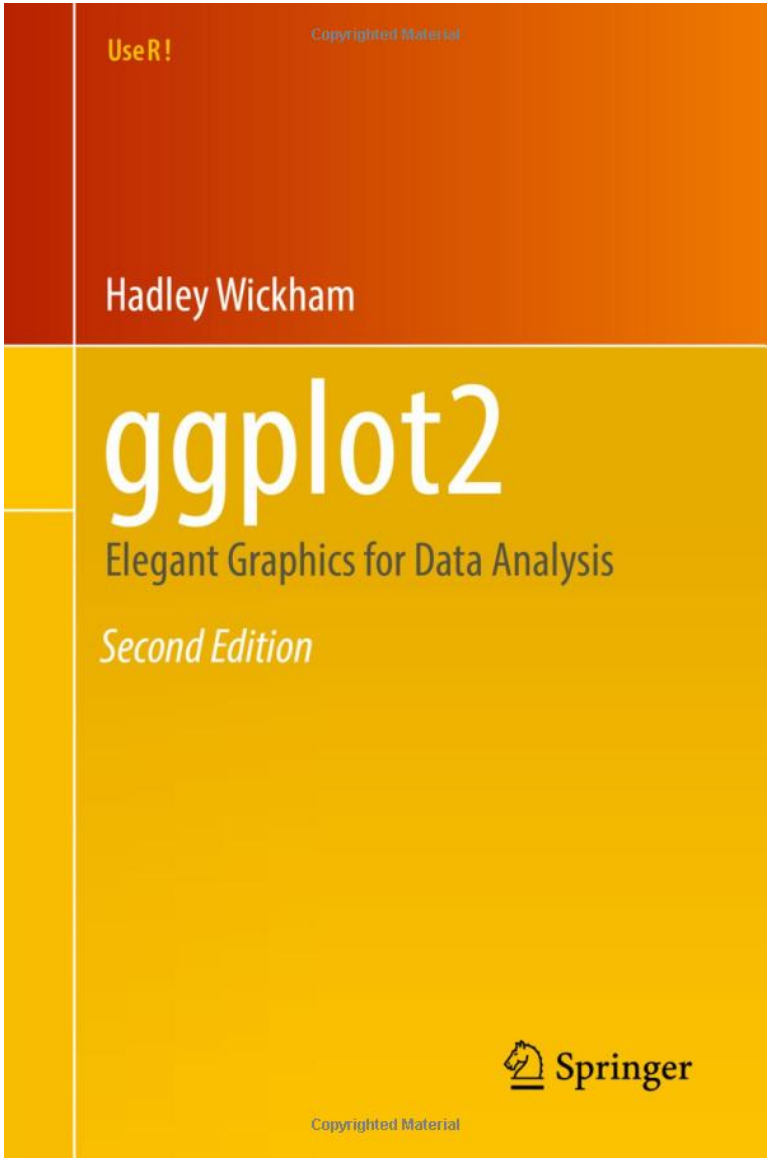
Visualization with {ggplot2}

- There are multiple packages to produce graphs in R.
  - The {base} and {ggplot2} packages offer plot functions I rely on the most.
- Here, we will introduce only functions from {ggplot2}.
- With this package, plots are created following a "**g**rammar of **g**raphics" and are constructed by adding **layers** to one another.
- The (awesome) official documentation: https://ggplot2.tidyverse.org/reference/

```
1 library(ggplot2)
```

> **Disclaimer**
>
> The slides are strongly inspired by the 🌐introduction provided by de Bruin (2024).

What is a grammar of graphics?



- A grammar of a **language** defines the rules of structuring words and phrases into meaningful expressions.
- Similarly, a grammar of **graphics** defines the rules of structuring mathematical and aesthetic elements into a meaningful graph.
- Wilkinson (2005) designed the grammar upon which `{ggplot2}` is based.
- To learn with a book: Wickham (2016).

Introduction to programming for data analysis — Session 3

The main elements in a `ggplot2` graph

- **Data**: variables mapped to aesthetic features of the graph.
- **Geoms**: objects/shapes on the graph.
- **Stats**: statistical transformations that summarize data (e.g. mean, confidence intervals).
- **Scales**: mappings of aesthetic values to data values. Legends and axes visualize scales.
- **Coordinate systems**: the plane on which data are mapped on the graphic.
- **Faceting**: splitting the data into subsets to create multiple variations of the same graph.

The base layer

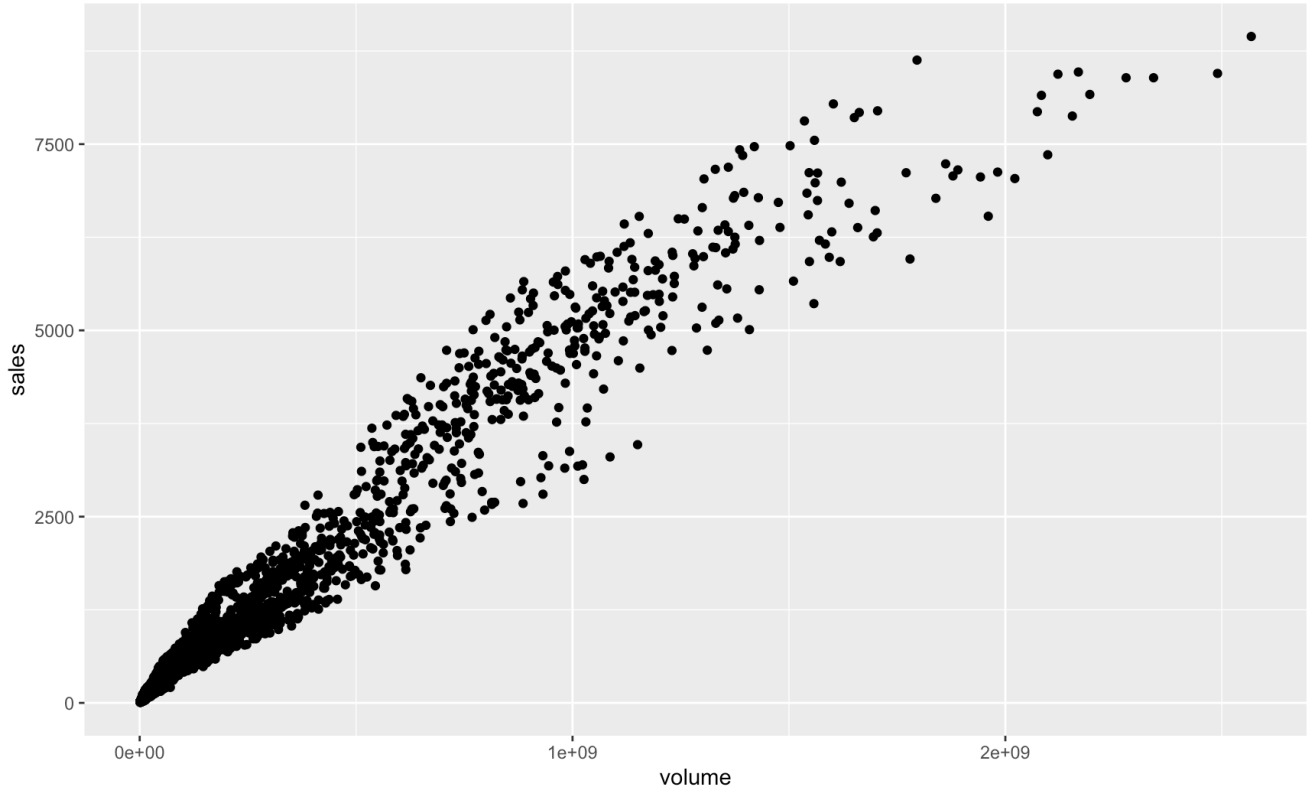
- The base layer of a `{ggplot2}` graph is created with the `ggplot()` function.
- We need to specify **two arguments**:
  - `data`: the dataset
  - `mapping`: a set of aesthetic mappings between variables in the data and visual properties
- Then, we need to provide at least **one layer which describes how to render each observation** (usually created with a `geom` function).

```
1 ggplot(
2   data = txhousing,
3   mapping = aes(x = volume, y = sales)
4 ) +
5   geom_point()
```

- Each layer is added using the `+` operator.

Our first graph with {ggplot2}

```
1 p_txhousing <- ggplot(
2   data = txhousing,
3   mapping = aes(x = volume, y = sales)
4 ) +
5   geom_point()
6
7 p_txhousing
```



Aesthetics

Geom layers and overriding aesthetics (1/2)

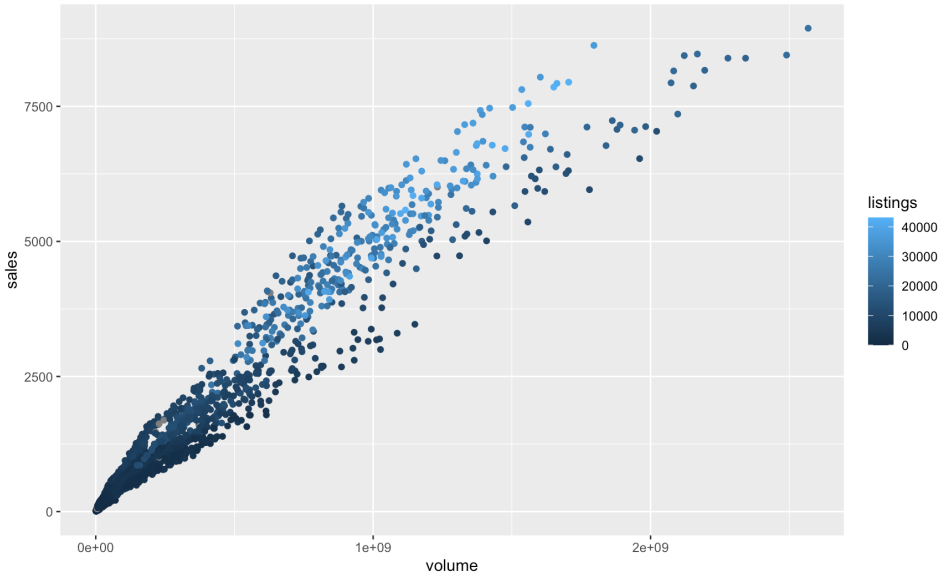
- The layers inherit aesthetics from `ggplot()`.
- If provided, new aesthetics will override those inherited from `ggplot()`.

```
ggplot(
  data = txhousing,
  mapping = aes(x = volume, y = sales)
) +
  geom_point(mapping = aes(colour = listings))
```

- `colour = listings` causes the colour of objects to vary with values of the variable `listings`.

Geom layers and overriding aesthetics (2/2)

```
ggplot(
  data = txhousing,
  mapping = aes(
    x = volume,
    y = sales
  )
) +
  geom_point(
    mapping = aes(colour = listings)
  )
```



Aesthetic attributes

- Multiple attributes are available, depending on the geom used, the most common are:
  - `x`, `y`: positioning along x-axis and y-axis, resp.,
  - `colour`: colour of objects,
  - `fill`: fill colour of objects (name or hexadecimal code),
  - `linetype`: how lines should be drawn (solid, dashed, dotted, ...),
  - `shape`: shape of markers,
  - `size`: area of markers (for points), font size (for text),
  - `linesize`: thickness of the lines,
  - `alpha`: transparency (between 0 (transparent) and 1 (opaque)).

Geoms and aesthetics

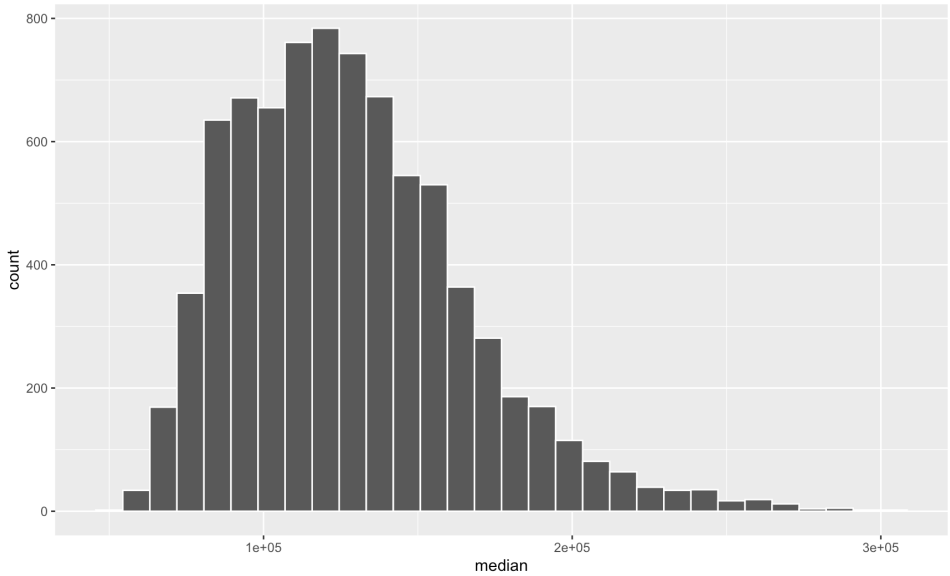
- Each geom has its required aesthetics.
  - E.g., `geom_point()` requires both `x` and `y`.
- Some aesthetics are not possible for some geoms.
  - E.g., while the `shape` aesthetic is used by `geom_point()`, it is not accepted by the `geom_bar()` function.
- Check the help pages or the online documentation, or even the 🎨gallery.

Usual visualizations

Histogram

```
1 ggplot(
2   data = txhousing,
3   mapping = aes(x = median)
4 ) +
5   geom_histogram(colour = "white")
```

A continuous variable (here: `median`) is cut into bins (default to 30). The number of observations (count) in each bin is reported.

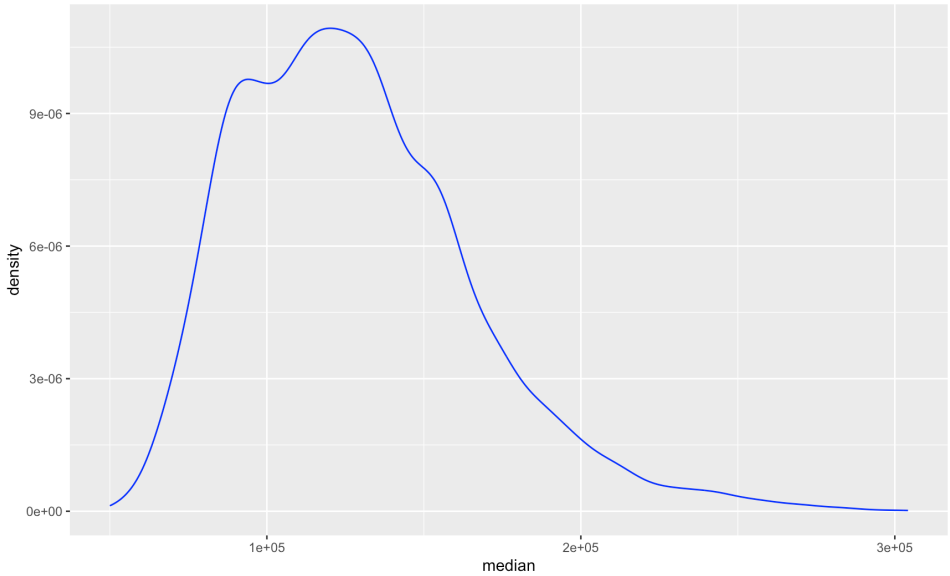


Distribution of the median housing sales in Texas between 2000 and 2015

Density plot

```
ggplot(
  data = txhousing,
  mapping = aes(x = median)
) +
  geom_density(colour = "blue")
```

This type of representation can be viewed as a smoothed version of histograms.

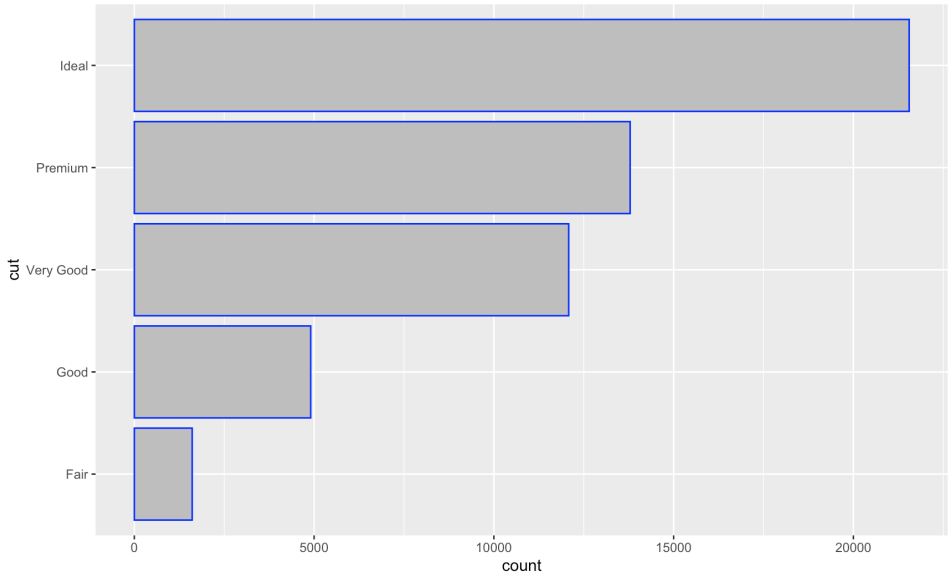


Density of the median housing sales in Texas between 2000 and 2015 (kernel estimation)

Bar plot

```
ggplot(
  data = diamonds,
  mapping = aes(y = cut)
) +
  geom_bar(
    colour = "blue", fill = "grey"
  )
```

The width of the bar plot reflects the counts of each y-value.

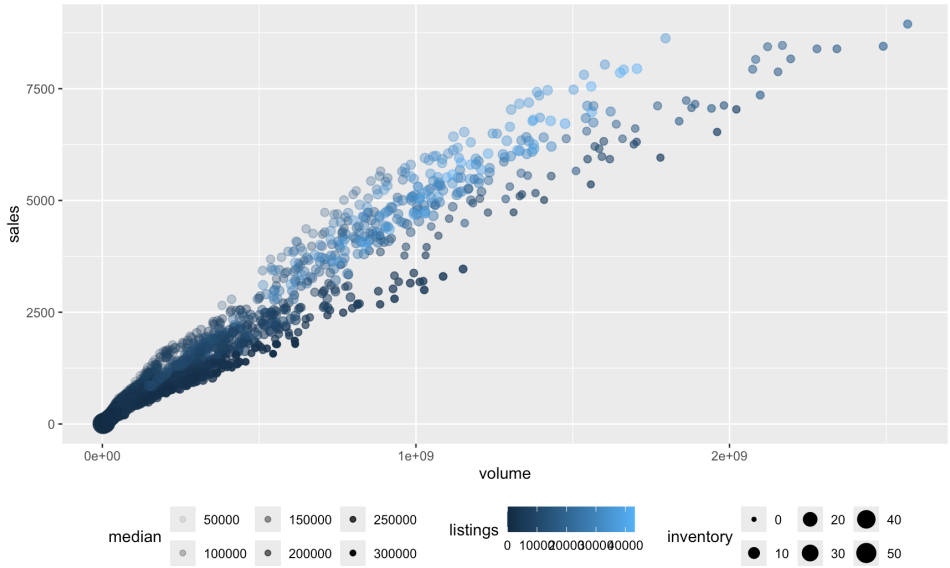


Number of observation per quality of the cut in the `diamonds` dataset.

Scatter plot

```
ggplot(
  data = txhousing,
  mapping = aes(
    x = volume, y = sales,
    colour = listings,
    alpha = median,
    size = inventory
    )
) +
  geom_point() +
  theme(legend.position = "bottom")
```

Useful to represent covariation between two continuous variables.

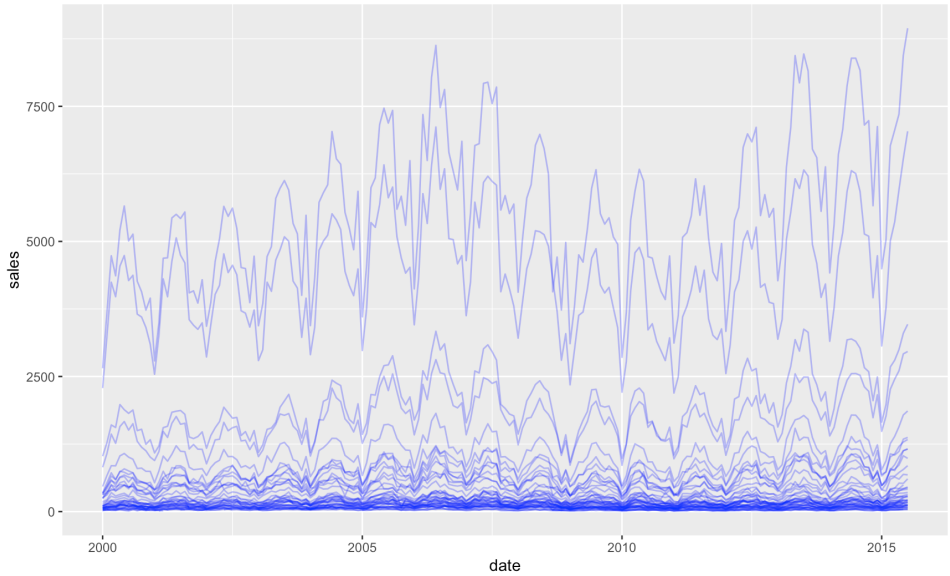


Number of housing sales vs. total values of sales in Texas

Line plot

```
ggplot(
  data = txhousing,
  mapping = aes(
    x = date, y = sales,
    group = city
    )
) +
  geom_line(
    colour = "blue", alpha = .3
  )
```

Useful to represent covariation between a continuous variable (y-axis) and an ordered variable (x-axis).



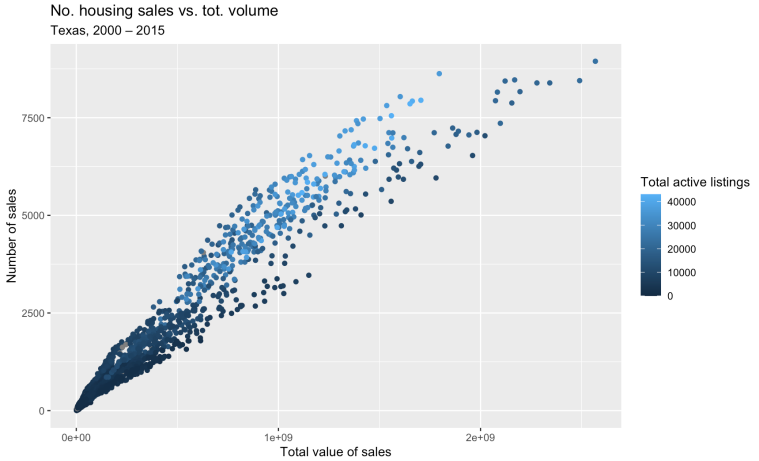
Evolution of the number of housing sales in Texas. Each line represents a city.

Labels

Labels

The `labs()` function allows to change axis, legend, and plot labels.

```
ggplot(
  data = txhousing,
  mapping = aes(
    x = volume, y = sales,
    colour = listings
  )) +
  geom_point() +
  labs(
    title = "No. housing sales vs. tot. volume",
    subtitle = "Texas, 2000 – 2015",
    x = "Total value of sales",
    y = "Number of sales",
    colour = "Total active listings"
  )
```



Number of housing sales vs. total values of sales in Texas.

Exporting Graphs

The `ggsave()` function

- The `ggsave()` function saves the last displayed plot (or a specific ggplot object if provided as argument).
- It supports many formats: **PNG** (good for presentations and web), **PDF** (for print and publications, i.e., your reports), JPEG, TIFF, SVG, EPS.
- Unless specified, in RStudio, the plot size and resolution of the 'Plots' tab is used (in inches).

```
1 # Assuming p contains a ggplot2 graph:
2 ggsave(file = "figs/plot.pdf", p, width = 5, height = 3)
```

Output quality

- When exporting a scatterplot, if the number of points is large, producing a PDF might not be a good idea: the file will be huge.
- To export a plot in PNG with a good resolution, the number of dots per inch (DPI) can be specified.

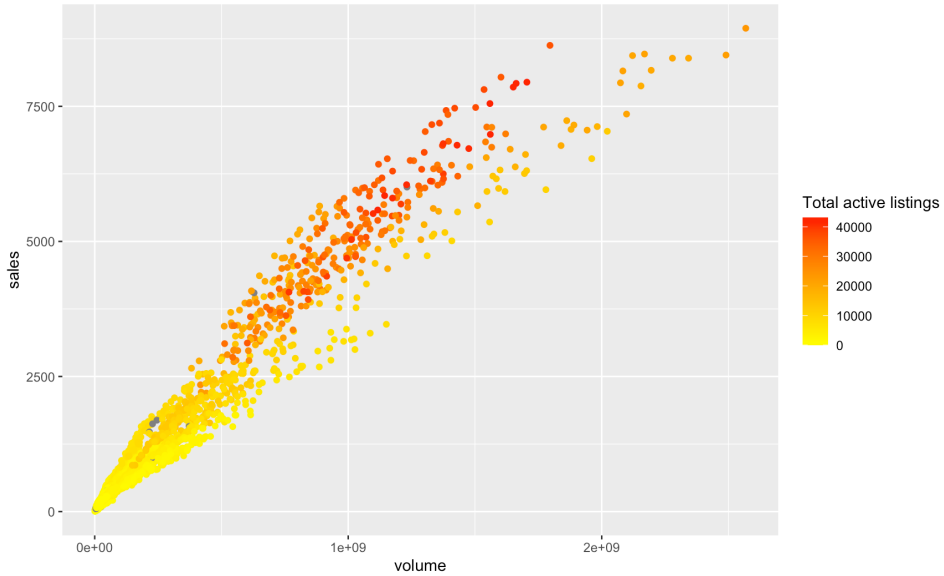
```r
# Assuming p contains a ggplot2 graph:
ggsave(
   file = "figs/plot.pdf", p, width = 5, height = 3,
   dpi = 300
)
```

Scales

- Every aesthetic (colour, size, shape, x/y position, etc.) has a corresponding **scale function**.
- Default scales are chosen automatically, but can be customized with a `scale_*` function.
- The type of scale depends on the data type of the variable:
  - **Continuous variables**: gradients, continuous axes, size scales,
  - **Discrete variables**: categorical palettes, shapes, breaks.

Scales with a numeric variable

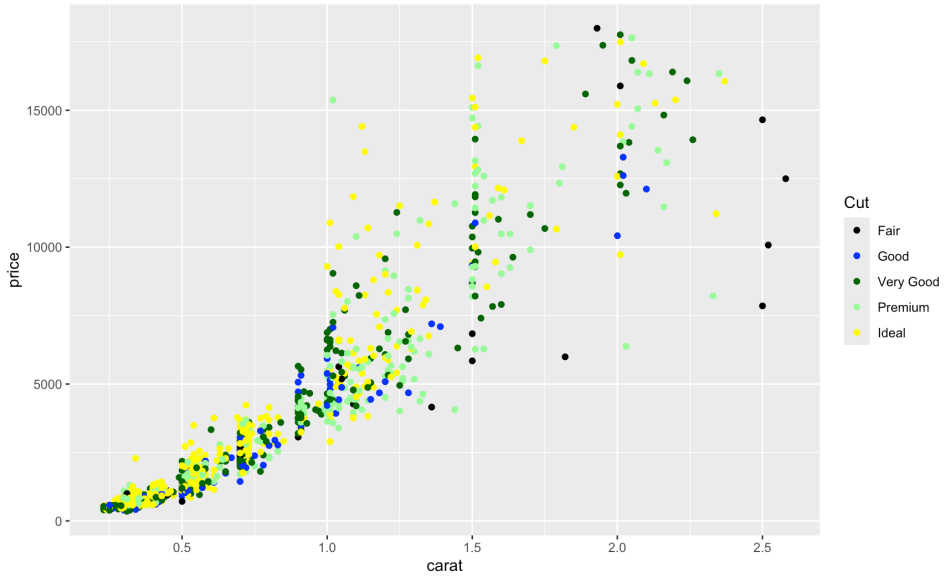
```
ggplot(
  data = txhousing,
  mapping = aes(
    x = volume, y = sales
  )
) +
  geom_point(
    mapping = aes(colour = listings)
  ) +
  scale_colour_gradient(
    "Total active listings",
    low = "yellow", high = "red"
  )
```



The `listing` variable is numeric.

Scales with a discrete variable

```
ggplot(
  data = sample_n(diamonds, size = 1000),
  mapping = aes(
    x = carat, y = price, colour = cut
  )
) +
  geom_point() +
  scale_colour_manual(
    "Cut",
    values = c(
      "Fair" = "black",
      "Good" = "blue",
      "Very Good" = "darkgreen",
      "Premium" = "palegreen",
      "Ideal" = "yellow"
    )
  )
```



The `cut` variable is discrete.

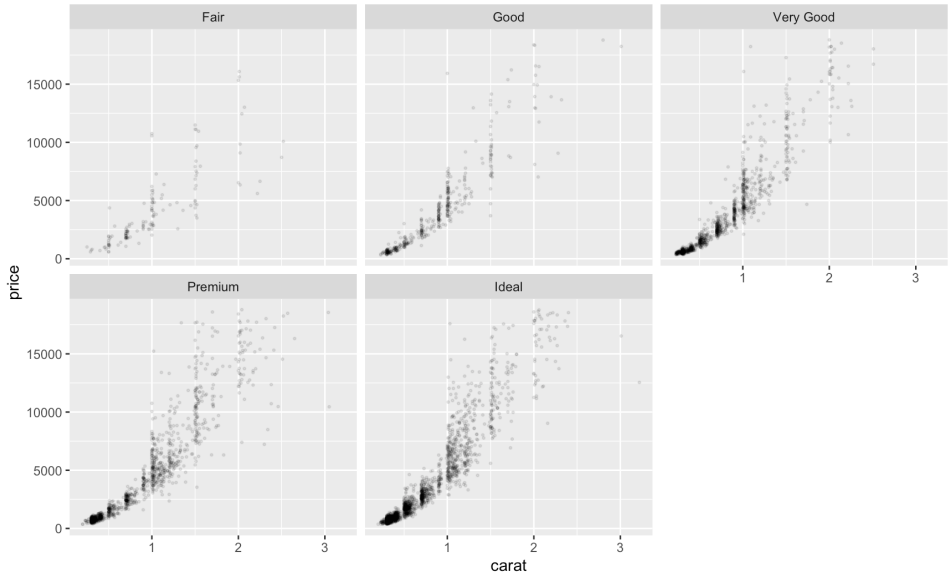
Facetting

Facetting

- Facetting corresponds to creating small subplots for subsets of data.
- Each panel shows the same plot for a different level of a variable.
- It makes it easier to compare patterns across groups.
- There are two main functions:
  - `facet_wrap()`: 1D wrapping of panels,
  - `facet_grid()`: 2D grid of panels.

Facetting with `facet_wrap()`

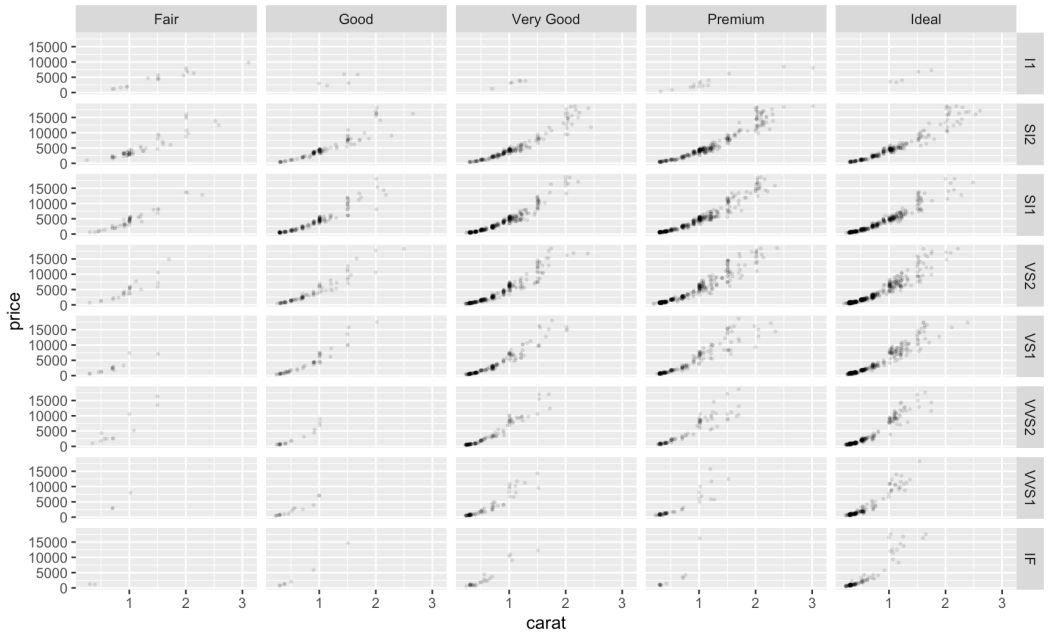
```
ggplot(
  data = sample_n(diamonds, size = 5000),
  mapping = aes(
    x = carat, y = price
  )
) +
  geom_point(size = .5, alpha = .1) +
  facet_wrap(~ cut)
```



Diamonds prices vs. their weights depending on the quality of the cut.

Facetting with `facet_grid()`

```
ggplot(
  data = sample_n(
    diamonds,
    size = 5000
  ),
  mapping = aes(
    x = carat, y = price
  )
) +
  geom_point(size = .5, alpha = .1) +
  facet_grid(clarity ~ cut)
```



Diamonds prices vs. their weights depending on the quality of the cut and the clarity of the diamond.

To go further

- This presentation only showed a glimpse of what `ggplot2` has to offer.
- To go further: 🌐https://ggplot2-book.org/

Exercises

Practice with the second tutorial!

Appendix

References

Bruin, J. 2024. “R Graphics: Introduction to Ggplot2.” May 2024. https://stats.oarc.ucla.edu/r/seminars/ggplot2_intro/.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis, Second Edition*. Springer Cham. https://doi.org/https://doi.org/10.1007/978-3-319-24277-4.

Wilkinson, Leland. 2005. *The Grammar of Graphics*. Springer New York, NY.

Codes for the graph showing the number of deaths in France (1/3)

```r
# Source: https://www.insee.fr/en/statistiques/serie/000436394
file <- "data/deces_france_insee/monthly_values.csv"
deces <- read_csv2(
    file, skip = 4,
    col_names = c("period", "deaths", "codes")
)

deces <-
  deces |>
  mutate(
    date = ym(period),
    month = month(date, label = TRUE),
    year = year(date)
  )
```

Codes for the graph showing the number of deaths in France (2/3)

```r
# Prepare data for the plot: highlight 2003 and 2021
data_plot_deaths <- deces |>
  mutate(
    year_excess = case_when(
      year == 2003 ~ "2003 (Drought)",
      year == 2020 ~ "2020 (Covid-19)",
      TRUE ~ "Other"
    ),
    year_excess = factor(
      year_excess,
      levels = c("2003 (Drought)", "2020 (Covid-19)", "Other")
    )
  )
```

Codes for the graph showing the number of deaths in France (3/3)

```
p_deaths <- ggplot(
  data = data_plot_deaths,
  mapping = aes(x = month, y = deaths, group = year, colour = year_excess)
) +
  geom_line(alpha = .8) +
  scale_colour_manual(
    NULL, values = c(
      "2003 (Drought)" = "blue",  "2020 (Covid-19)" = "red",
      "Other" = "darkgray"
  )) +
  labs(x = NULL, y = "Deaths") +
  scale_y_continuous(
    labels = scales::label_number(suffix = "", scale = 1, big.mark = ",")
  ) + theme_minimal() + theme(legend.position = "bottom")
```

⏮ Go back