WorldPop

Workshop on Advanced Sampling Methodologies

Advanced R Programming

Dr. Gianluca Boo, WorldPop, University of Southampton

2025-09-17

Previously covered

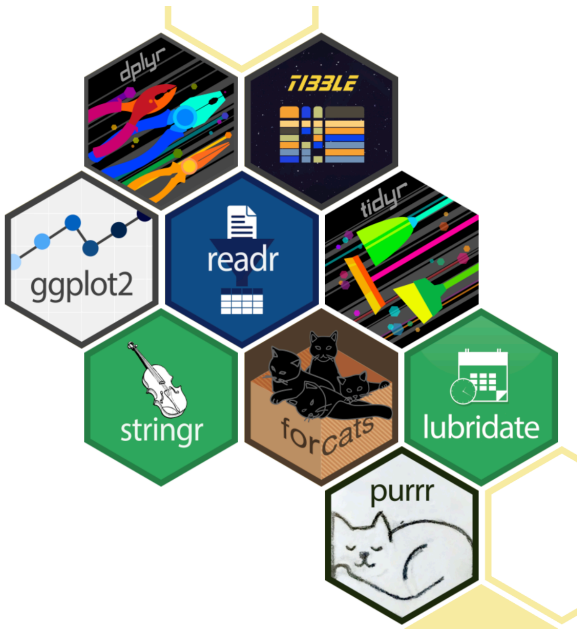
- Introduction to R and RStudio
- Data types and data structures
- Working with scripts and the console
- Importing data
- Saving and closing sessions

Agenda

- The `tydyverse`
- Pipes (`|>` and `%>%`)
- Data manipulation with `dplyr`
- Data visualization with `ggplot2`
- Practice exercises

The `tidyverse`

- A collection of R packages for **data science**
- Common design philosophy and consistent syntax



Core packages:

- **`dplyr`**: data manipulation
- **`ggplot2`**: data visualization
- **`tidyr`**: data cleaning
- **`readr`**: data import
- **`stringr`**, **`forcats`**, **`purrr`**, etc.

Install once:

```
1  install.packages("tidyverse")
```

Load into session:

```
1  library(tidyverse)
```

Pipes

- Pipes make code **read left-to-right**
- **Combine functions** in a readable manner
- Avoids deeply **nested functions**
- Ctrl+Shift+M (Windows) or Cmd+Shift+M (Mac)

Two different **types of pipes** exists:

- Base R pipe: `|>`
- Magrittr pipe: `%>%`

Example: selecting the top 2 cars from mtcars with mpg over 20, sorted from highest to lowest mpg.

Without pipes:

```
1 library(tidyverse)
2 head(arrange(filter(mtcars, mpg > 2
```

With pipes:

```
1 library(tidyverse)
2 mtcars |>
3   filter(mpg > 20) |>
4   arrange(desc(mpg)) |>
5   head(2)
```

ℹ️ Try to run the code in your Script Editor.

Data Manipulation with `dplyr`

- Part of the **tidyverse**
- Makes data manipulation **easy and readable**
- Works primarily with **data frames/tibbles**

Function	Purpose	Example
`filter()`	Subset rows based on conditions	`filter(mtcars, mpg > 20)`
`arrange()`	Sort rows	`arrange(mtcars, desc(mpg))`
`select()`	Pick columns	`select(mtcars, mpg, cyl)`
`mutate()`	Create or modify columns	`mutate(mtcars, kpl = mpg * 0.425)`
`summarise()`	Summarize data	`summarise(mtcars, avg_mpg = mean(mpg))`
`group_by()`	Group data for aggregation	`group_by(mtcars, cyl)`

dplyr::filter()

- **Purpose:** subset rows based on conditions
- **Syntax:** `filter(data, condition)`
- **Example:** filter cars above 20 mpg

```
1 library(tidyverse)
2 mtcars |>
3 filter(mpg > 20)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2

dplyr::arrange()

- **Purpose**: sort rows by column values
- **Syntax**: `arrange(data, column1, desc(column2))`
- **Example**: sort cars from highest to lowest mpg

```
1 library(tidyverse)
2 mtcars |>
3 arrange(desc(mpg))
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4

dplyr::select()

- **Purpose**: choose specific columns
- **Syntax**: `select(data, col1, col2, ...)`
- **Example**: returns only mpg, cyl, and hp columns

```
1 library(tidyverse)
2 mtcars |>
3 select(mpg, cyl, hp)
```

```
                   mpg cyl  hp
Mazda RX4         21.0   6 110
Mazda RX4 Wag     21.0   6 110
Datsun 710        22.8   4  93
Hornet 4 Drive    21.4   6 110
Hornet Sportabout 18.7   8 175
Valiant           18.1   6 105
Duster 360        14.3   8 245
Merc 240D         24.4   4  62
Merc 230          22.8   4  95
Merc 280          19.2   6 123
Merc 280C         17.8   6 123
Merc 450SE        16.4   8 180
Merc 450SL        17.3   8 180
```

dplyr::mutate()

- **Purpose**: add or modify columns
- **Syntax**: `mutate(data, new_col = expression)`
- **Example**: add a new column converting mpg to km/l

```
1 library(tidyverse)
2 mtcars |>
3 mutate(kpl = mpg * 0.425)
```

```
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
kpl
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
8.9250
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
8.9250
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
9.6900
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
9.0950
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
7.9475
Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
7.6925
```

`summarise()` and `group_by()`

- **Purpose**: aggregate data
- **Syntax**: `data %>% group_by(group_col) %>% summarise(summary_col = mean(value))`
- **Example**: show average mpg for each cylinder group

```
1 library(tidyverse)
2 mtcars |>
3   group_by(cyl) |>
4   summarise(avg_mpg = mean(mpg))
```

```
# A tibble: 3 × 2
    cyl avg_mpg
  <dbl>   <dbl>
1     4    26.7
2     6    19.7
3     8    15.1
```

ℹ️ Try to run the code in your Script Editor.

Data visualization with `ggplot2`

- Part of the **tidyverse**
- Used for **data visualization**
- Follows the **Grammar of Graphics**: layers of data, aesthetics, and geometries
- The `ggplot` function has three main components: **data**, aestics (**aes**), and geometries (**geoms**).

Scatter plot example:

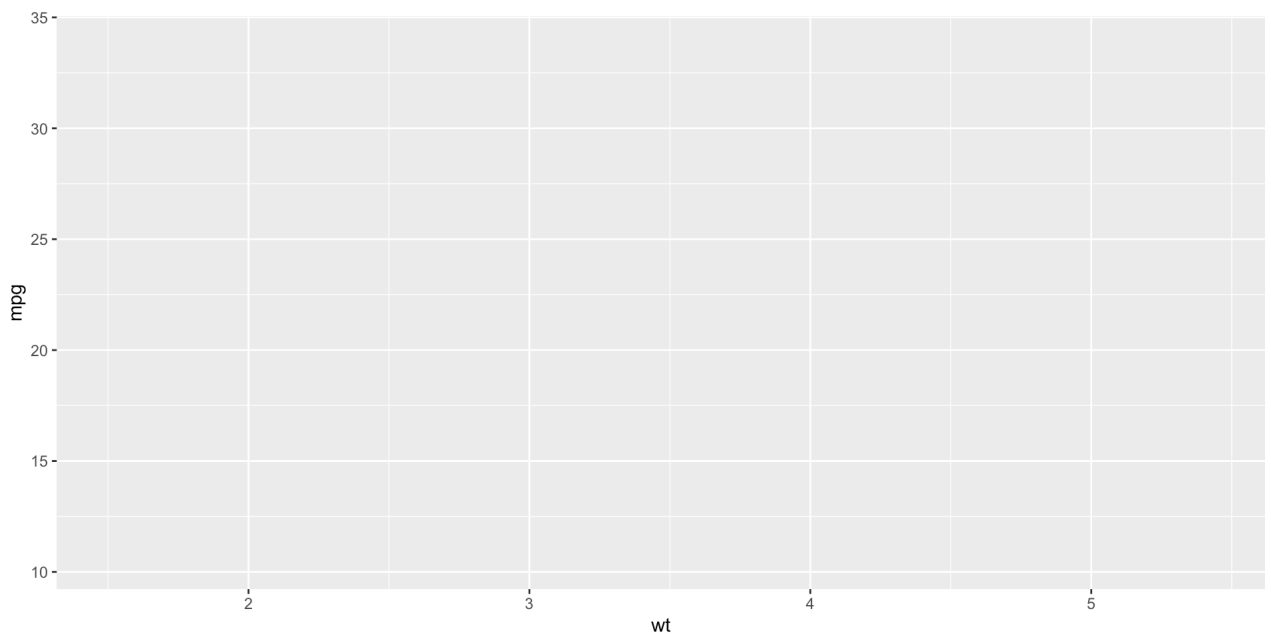
```
1  library(ggplot2)
2  ggplot(mtcars, aes(x = hp, y = mpg)) +
3    geom_point()
```

ℹ️ Try to run the code in your Script Editor.

aes()

- `aes()` stands for **aesthetics**
- Maps data **columns** to **visual properties** like: x, y positions, color, fill, size, shape, and alpha
- **Example**: map `wt` to x-axis, `mpg` to y-axis, and `cyl` to color

```
1 library(ggplot2)
2 ggplot(mtcars, aes(x = wt, y = mpg, color = cyl))
```

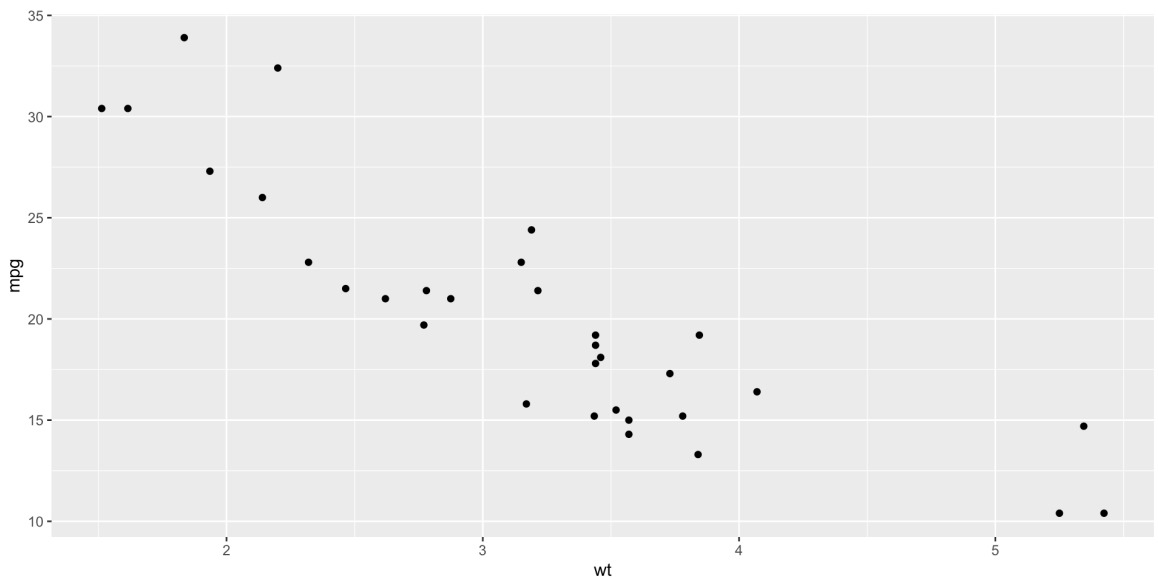


ℹ️ Try to run the code in your Script Editor.

geom_point()

- **Purpose:** scatter plots
- **Plots points** at x and y positions
- **Example**: visualize the relationship between weight and mpg

```
1 library(ggplot2)
2 ggplot(mtcars, aes(x = wt, y = mpg)) +
3   geom_point()
```

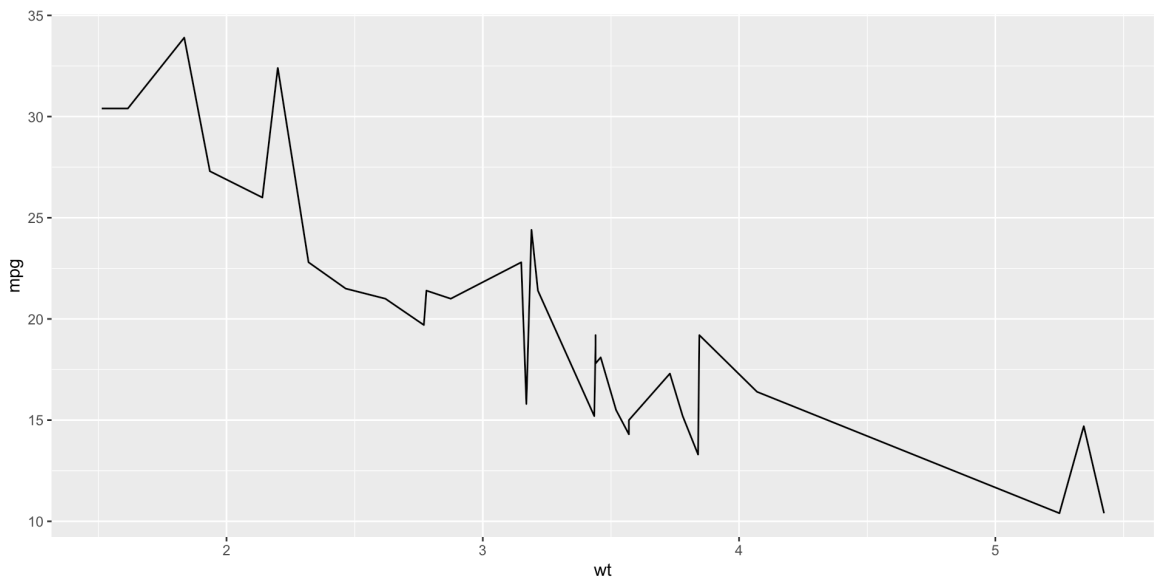


ℹ️ Try to run the code in your Script Editor.

geom_line()

- **Purpose**: line plots
- Connects data points with a **line**, often used for time series
- **Example**: show how unemployment changes over time

```
1 library(ggplot2)
2 ggplot(mtcars, aes(x = wt, y = mpg)) +
3   geom_line()
```

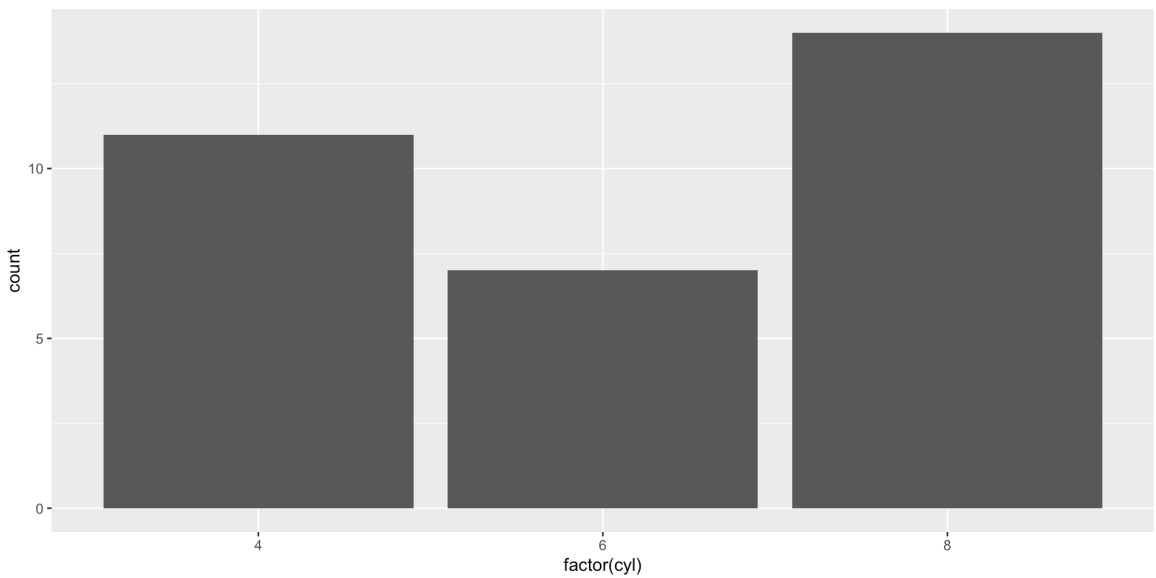


ℹ️ Try to run the code in your Script Editor.

geom_bar()

- **Purpose**: bar charts
- Visualizes **counts or summaries** of categorical data
- **Example**: counts the number of cars for each cylinder group

```
1 library(ggplot2)
2 ggplot(mtcars, aes(x = factor(cyl))) +
3   geom_bar()
```

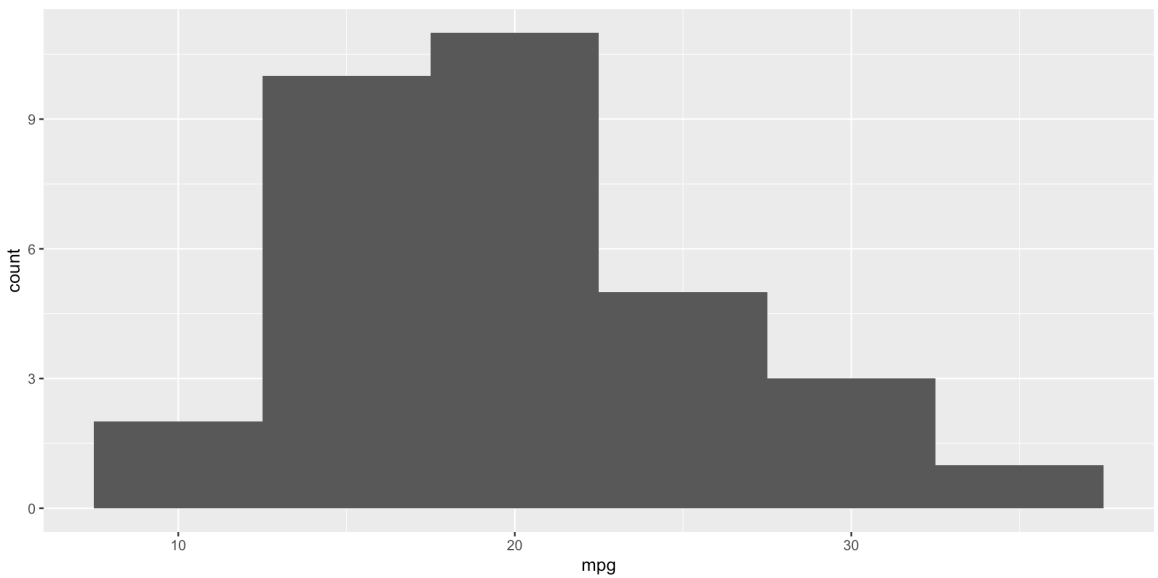


ℹ️ Try to run the code in your Script Editor.

geom_histogram()

- **Purpose**: histogram for continuous data
- Groups data into bins to show **distribution**
- **Example**: show the distribution of mpg values

```
1 library(ggplot2)
2 ggplot(mtcars, aes(x = mpg)) +
3   geom_histogram(binwidth = 5)
```

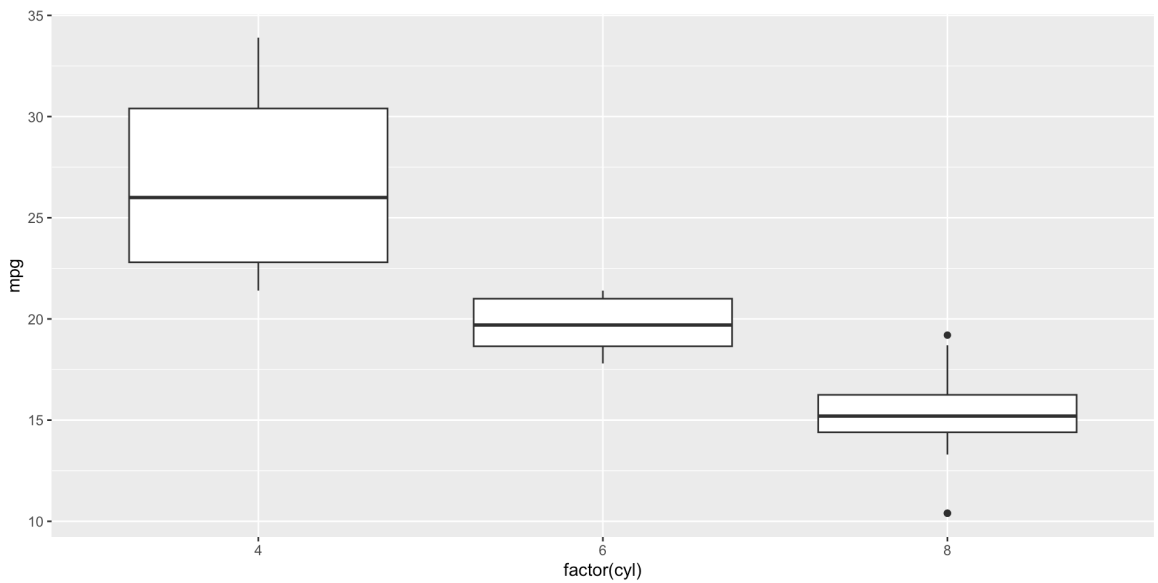


ℹ️ Try to run the code in your Script Editor.

geom_boxplot()

- **Purpose**: boxplots for distribution and outliers
- Summarizes **median, quartiles, and extremes**
- **Example**: compare mpg across cylinder groups

```
1  library(ggplot2)
2  ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
3    geom_boxplot()
```

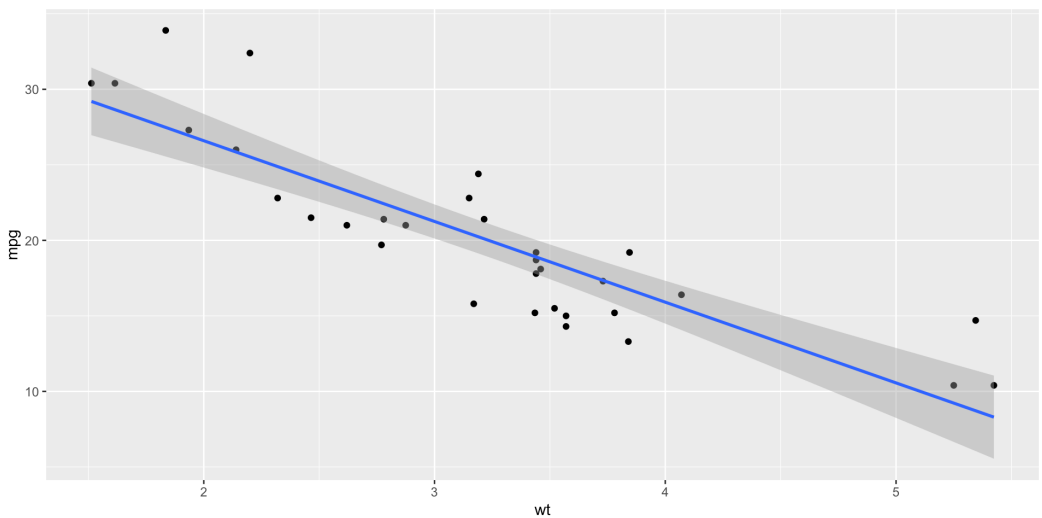


ℹ️ Try to run the code in your Script Editor.

geom_smooth()

- **Purpose**: add trend lines / regression lines
- Can show **linear models** or **smoothed trends**
- **Example**: add a linear regression line to the scatter plots

```
1 library(ggplot2)
2 ggplot(mtcars, aes(x = wt, y = mpg)) +
3   geom_point() +
4   geom_smooth(method = "lm")
```



ℹ️ Try to run the code in your Script Editor.

Data cleaning with `tidyr`

- Part of the **tidyverse**
- Focused on **reshaping and tidying data**
- Make datasets **long and tidy** so that each variable is a column, each observation a row, and each value a cell

Function	Purpose	Example
`pivot_longer()`	Converts wide data to long (columns → rows)	`pivot_longer(df, cols = c(a, b), names_to="var", values_to="val")`
`pivot_wider()`	Converts long data to wide (rows → columns)	`pivot_wider(df, names_from=var, values_from=val)`
`separate()`	Splits one column into multiple columns	`separate(df, col, into=c("x","y"), sep="-")`

Function	Purpose	Example
`unite()`	Combines multiple columns into one	`unite(df, "date", year, month, sep="-")`
`drop_na()`	Removes rows with missing values	`drop_na(df)`
`fill()`	Fills missing values with previous/next values	`fill(df, year, .direction="down")`
`replace_na()`	Replaces missing values with a specified value	`replace_na(df, list(x=0, y="unknown"))`
`nest()`	Creates nested (list-column) data frames	`nest(df, data = c(x, y))`
`unnest()`	Expands nested data frames back to flat format	`unnest(df, data)`

tidyr::pivot_longer()

- **Purpose**: convert wide data to long data
- **Example**: turn multiple stock columns into key–value pairs

```
 1 library(tidyverse)
 2 stocks <- data.frame(
 3   year = 2015:2016,
 4   stockA = c(10, 20),
 5   stockB = c(15, 25)
 6 )
 7
 8 stocks |>
 9 pivot_longer(cols = c(stockA, stockB), names_to = "stock", values_to = "pri
```

```
# A tibble: 4 × 3
   year stock  price
  <int> <chr>  <dbl>
1  2015 stockA    10
2  2015 stockB    15
3  2016 stockA    20
4  2016 stockB    25
```

ℹ️ Try to run the code in your Script Editor.

tidyr::pivot_wider()

- **Purpose**: convert long data to wide data
- **Example**: spread stock names into separate columns

```
1 library(tidyverse)
2 long_data <- data.frame(
3   year = c(2015, 2015, 2016, 2016),
4   stock = c("A", "B", "A", "B"),
5   price = c(10, 15, 20, 25)
6 )
7
8 long_data |>
9 pivot_wider(names_from = stock, values_from = price)
```

```
# A tibble: 2 × 3
   year     A     B
  <dbl> <dbl> <dbl>
1  2015    10    15
2  2016    20    25
```

ℹ️ Try to run the code in your Script Editor.

tidyr::separate()

- **Purpose**: split one column into multiple
- **Example**: split "2020-01" into year and month columns

```r
library(tidyverse)
df <- data.frame(date = c("2020-01", "2020-02"))

df |>
separate(date, into = c("year", "month"), sep = "-")
```

```
  year month
1 2020    01
2 2020    02
```

ℹ️ Try to run the code in your Script Editor.

tidyr::unite()

- **Purpose**: combine multiple columns into one
- **Example**: combine year and month into a single "date" column

```r
library(tidyverse)
df <- data.frame(year = c(2020, 2020), month = c("01", "02"))

df |>
unite("date", year, month, sep = "-")
```

```
     date
1 2020-01
2 2020-02
```

ℹ️ Try to run the code in your Script Editor.

`tidyr::drop_na()`

- **Purpose**: remove rows with missing values
- **Example**: remove rows where any column is `NA`

```
1 library(tidyverse)
2 df <- data.frame(a = c(1, NA, 3), b = c("x", "y", NA))
3
4 df |> drop_na()
```

```
  a b
1 1 x
```

ℹ️ Try to run the code in your Script Editor.

tidyr::fill()

- **Purpose**: fill missing values using previous or next values
- **Example**: fill missing years with the last known value

```r
library(tidyverse)
df <- data.frame(
  year = c(2020, NA, NA, 2021, NA),
  value = 1:5
)

df |>
fill(year, .direction = "down")
```

```
  year value
1 2020     1
2 2020     2
3 2020     3
4 2021     4
5 2021     5
```

ℹ️ Try to run the code in your Script Editor.

And many more

- This is only a **snapshot of the all the possibilities** offered by the `tidyverse` package collection
- There are excellent **on-line resources** to support you with learning how to use different packages
- **Cheat sheets** are an excellent way to have a snapshot of the main functionalities of a package

Take home

- The `tidyverse` provides a unified, consistent toolkit for modern data science in R
- Pipes (`|>` / `%>%`) make your code easier to read and write
- `dplyr` simplifies data manipulation with intuitive verbs
- `ggplot2` enables powerful and flexible data visualization
- `tidyr` helps keep your data clean and tidy for analysis

Resources

Tidyverse Website – https://www.tidyverse.org

RStudio Cheat Sheets – https://posit.co/resources/cheatsheets/

Data Visualization with ggplot2 – https://ggplot2.tidyverse.org

Posit Community Forum – https://community.rstudio.com

Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). *R for Data Science* (2nd ed.). O'Reilly Media. Retrieved from https://r4ds.hadley.nz

Exercise

Please download the R script with exercises from GitHub and try to complete it.