

# Getting Started with R

3 September 2021

Molly Lewis

Lab

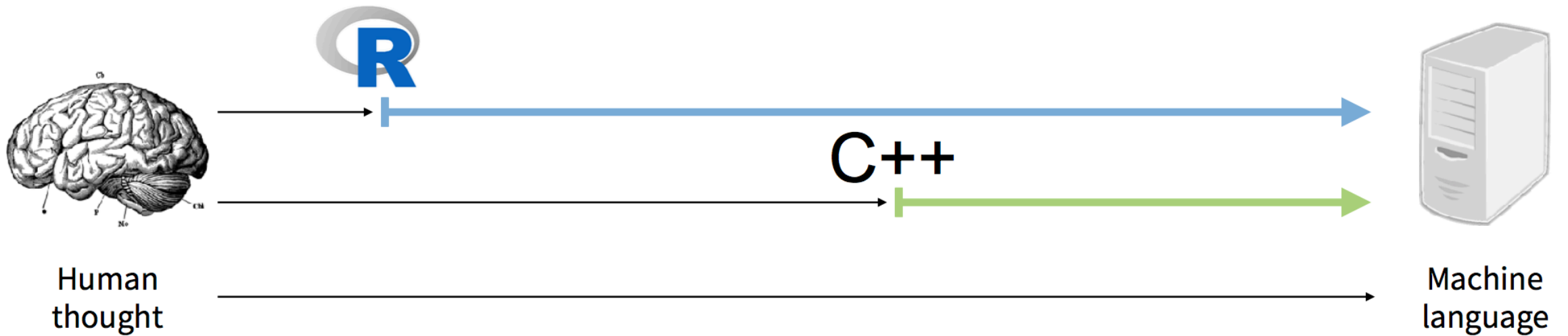


Artwork by @allison\_horst

# Why R for working with data

- Free and open-source
- Programming language (not point-and-click)
- Excellent graphics
- Easy to generate with reproducible reports
- Easy to integrate with other tools

# R - A computer language for scientists



# Rstudio Integrated Development Environment (IDE)

Source: edit file that you can run again later.

Console: type/paste commands to get output from R

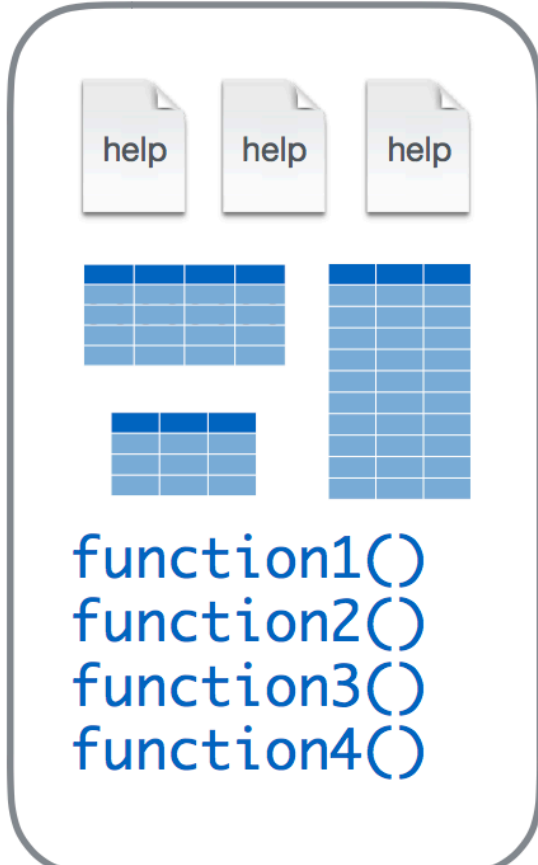
The screenshot displays the RStudio IDE interface with four main panels:

- Source (red border):** Shows the R script file `diamondPricing.R` with the following code:

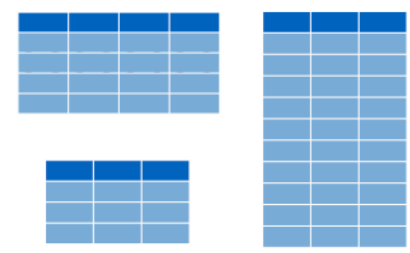
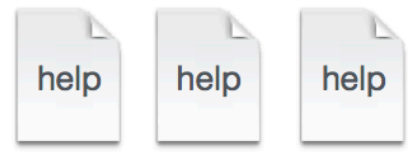
```
1 library(ggplot2)
2 source("plots/formatPlot.R")
3
4 view(diamonds)
5 summary(diamonds)
6
7 summary(diamonds$price)
8 aveSize <- round(mean(diamonds$carat), 4)
9 clarity <- levels(diamonds$clarity)
10
11 p <- qplot(carat, price,
12            data=diamonds, color=clarity,
13            xlab="Carat", ylab="Price",
14            main="Diamond Pricing")
15
```
- Workspace/History (yellow border):** Shows the current workspace with 53940 observations of 10 variables. The variables listed are `aveSize` (0.7979), `clarity` (character [8]), and `ggplot` [8]. The functions listed are `format.plot(plot, size)`.
- Console (green border):** Shows the output of the R commands, including summary statistics for `x`, `y`, and `z`, and the execution of `summary(diamonds$price)`, `aveSize`, `clarity`, `p`, and `format.plot`.
- Plots (purple border):** Displays a scatter plot titled "Diamond Pricing" showing Price (Y-axis, 0 to 15000) versus Carat (X-axis, 0.0 to 3.5). The plot is faceted by Clarity, with a legend on the right showing categories: I1, SI2, SI1, VS2, VS1, VVS2, VVS1, and IF.

Workspace/History: see list of variables and previous commands

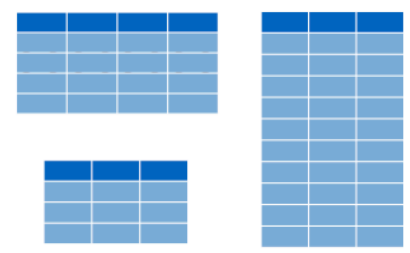
Files/Plots/Packages/Help



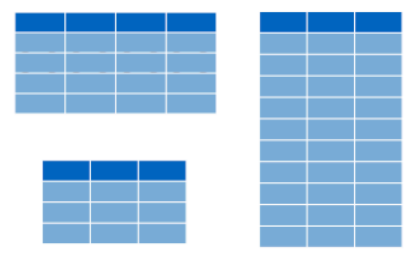
Base R



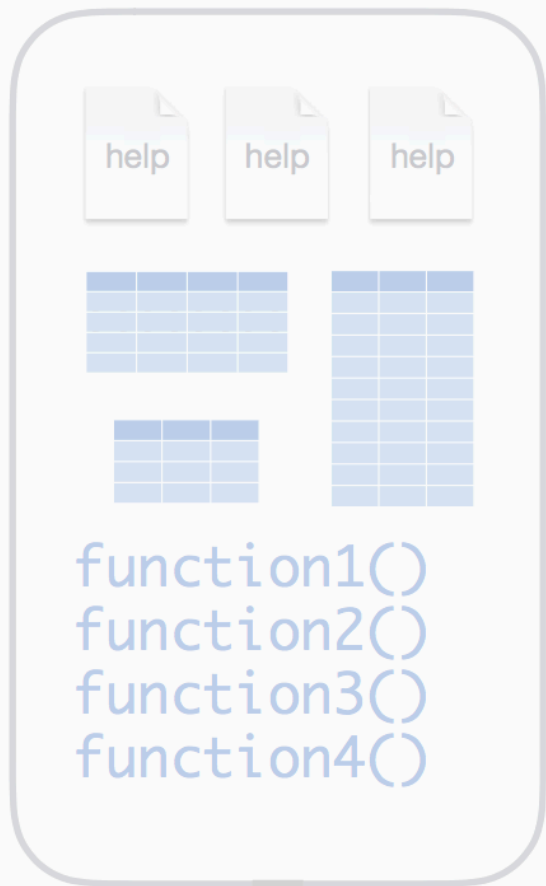
function5()  
function6()  
function7()  
function8()



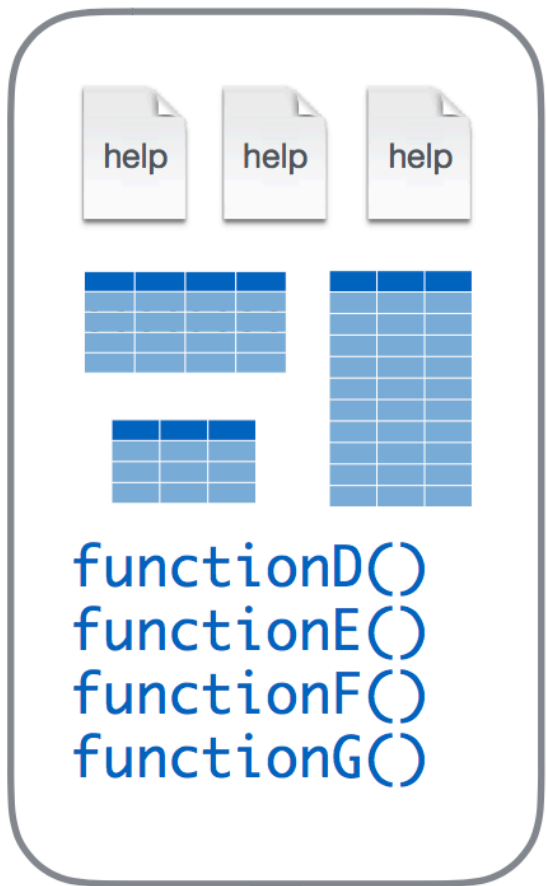
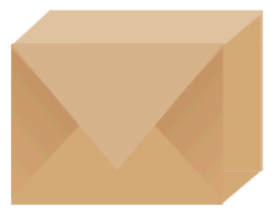
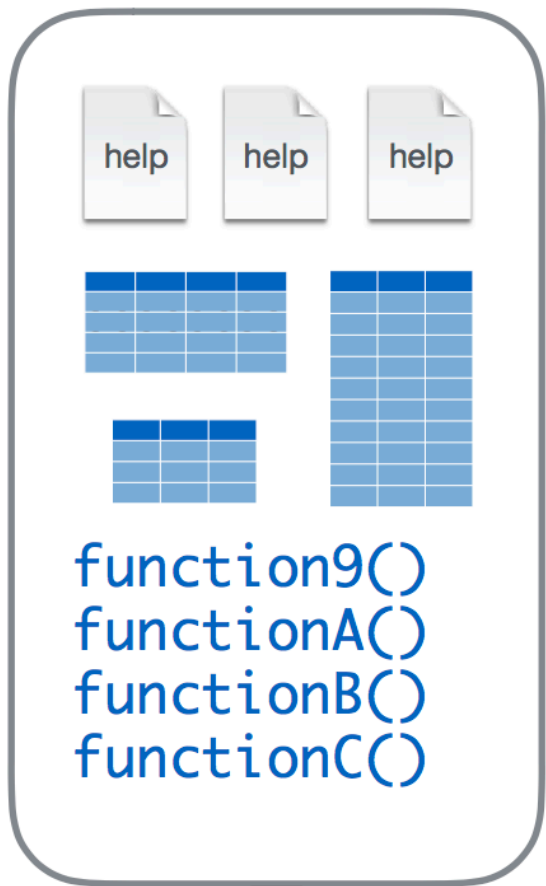
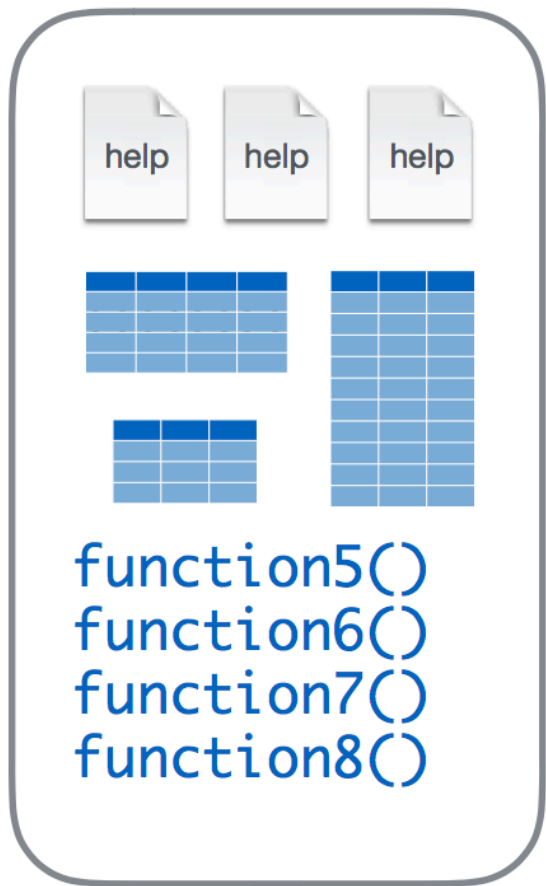
function9()  
functionA()  
functionB()  
functionC()



functionD()  
functionE()  
functionF()  
functionG()



Base R



R Packages



*CRAN*

[Mirrors](#)

[What's new?](#)

[Task Views](#)

[Search](#)

*About R*

[R Homepage](#)

[The R Journal](#)

*Software*

[R Sources](#)

[R Binaries](#)

[Packages](#)

[Other](#)

*Documentation*

[Manuals](#)

[FAQs](#)

[Contributed](#)

[A3](#)

[abbyyR](#)

[abc](#)

[ABCanalysis](#)

[abc.data](#)

[abcdeFBA](#)

[ABCOptim](#)

[ABCp2](#)

[ABC.RAP](#)

[abcrf](#)

[abctools](#)

[abd](#)

[abf2](#)

[ABHgenotypeR](#)

[abind](#)

[abjutils](#)

[abn](#)

[abodOutlier](#)

## Available CRAN Packages By Name

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#) [Y](#) [Z](#)

Accurate, Adaptable, and Accessible Error Metrics for Predictive Models

Access to Abbyy Optical Character Recognition (OCR) API

Tools for Approximate Bayesian Computation (ABC)

Computed ABC Analysis

Data Only: Tools for Approximate Bayesian Computation (ABC)

ABCDE\_FBA: A-Biologist-Can-Do-Everything of Flux Balance Analysis with this package

Implementation of Artificial Bee Colony (ABC) Optimization

Approximate Bayesian Computational Model for Estimating P2

Array Based CpG Region Analysis Pipeline

Approximate Bayesian Computation via Random Forests

Tools for ABC Analyses

The Analysis of Biological Data

Load Gap-Free Axon ABF2 Files

Easy Visualization of ABH Genotypes

Combine Multidimensional Arrays

Useful Tools for Jurimetrical Analysis Used by the Brazilian Jurimetrics Association

Modelling Multivariate Data with Additive Bayesian Networks

Angle-Based Outlier Detection

# Using packages

**1**

```
install.packages("foo")
```

Downloads files to computer

**1 x per computer**

**2**

```
library("foo")
```

Loads package

**1 x per R Session**



# Tidyverse



The [tidyverse](#) is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

data  
visualisation



reading  
data



modern  
data frames



data  
wrangling

# tidyverse

data  
tidying



dealing  
with factors



functional  
programming



string  
manipulation



# tidyverse

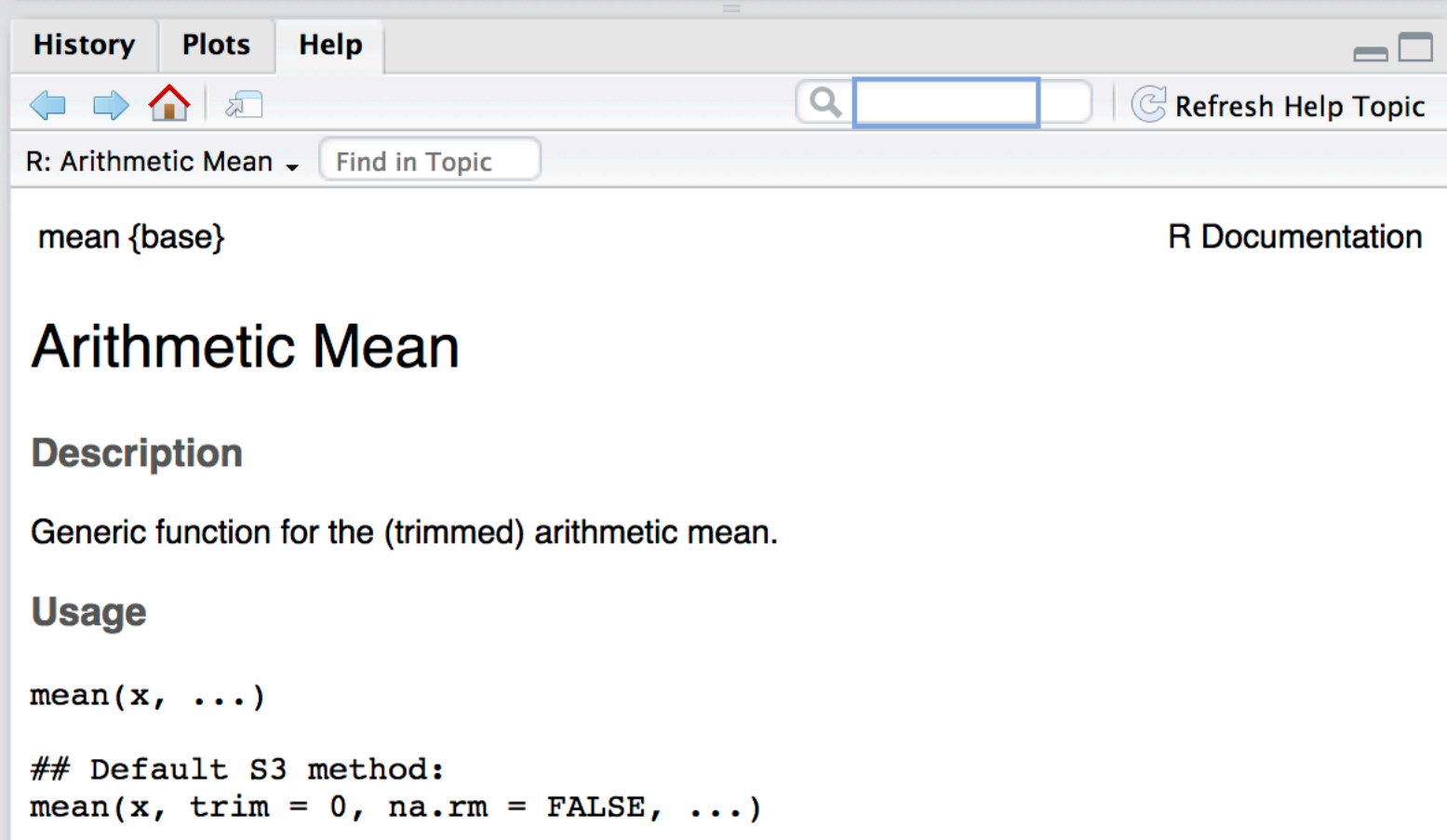


An R package that serves as a short cut for installing and loading the components of the tidyverse.

```
library("tidyverse")
```

# Getting help

## 1. R help files



The screenshot shows the R Help window with the 'History', 'Plots', and 'Help' tabs. The 'Help' tab is active, displaying the documentation for the 'mean' function. The window title is 'R: Arithmetic Mean'. The breadcrumb path is 'R: Arithmetic Mean'. The search bar is empty. The 'Refresh Help Topic' button is visible. The main content area shows the following text:

```
mean {base} R Documentation
```

## Arithmetic Mean

### Description

Generic function for the (trimmed) arithmetic mean.

### Usage

```
mean(x, ...)
```

## Default S3 method:  
mean(x, trim = 0, na.rm = FALSE, ...)

# Anatomy of an R help file

Two ways to access:

1. Peruse in Help pane
2. ?<name> in console

The name of the function, and the library it is in. `mean {base}` R Documentation Arithmetic Mean

Description

What it does. Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)
```

```
## Default S3 method:  
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

- `x` An R object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for `trim = 0`, only.
- `trim` the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of `trim` outside that range are taken as the nearest endpoint.
- `na.rm` a logical value indicating whether NA values should be stripped before the computation proceeds.
- `...` further arguments passed to or from other methods.

Value

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If `x` is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning. If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

[weighted.mean](#), [mean.POSIXct](#), [colMeans](#) for row and column means.

Examples

```
x <- c(0:10, 50)  
xm <- mean(x)  
c(xm, mean(x, trim = 0.10))
```

[Package *base* version 3.4.3 [Index](#)]

Annotations:

- The function's name, and in the parentheses the named arguments it expects, in the order it expects them. If an argument has a default value, it is shown. Arguments without default values (e.g. `x`) must be provided by you.
- More details on each named argument. This will tell you what class of thing each argument has to be—an object, a number, a data frame, a logical value, etc.
- The ellipsis allows other arguments to be passed to and from the function.
- Other related functions
- Self-contained examples that you can run at the console. These may use built-in datasets or other R functions.
- Visit the package's Index page to look for Demos and Vignettes detailing how it works.

# Getting help

1. R help files
2. Cheatsheets  
(<https://rstudio.cloud/learn/cheat-sheets>)
3. Google!

## 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>) +  
  <GEOM_FUNCTION>[mapping = aes(<MAPPINGS>)]  
  stat = <STAT> position = <POSITION> +  
  <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. More than files names in file explanation.

**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, yend, y, yend, alpha, angle, color, curvature, linetype, size)

a + geom\_path(linetype = "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 = 1:115, 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(aes(l = "spuslan") x, y, alpha, color, fill, group, linetype, size, weight)

**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

**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 = 12)  
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))

j + geom\_crossbar(latten = 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



mean function r

Google Search

I'm Feeling Lucky

# Today

1. Make sure you have R, Rstudio and tidyverse installed on your laptop
2. Work through online tutorial on R programming basics: <https://rstudio.cloud/learn/primers/1.2>
3. (complete readings for today)

# Acknowledgements

Slide 2 adapted from

<https://www.andrew.cmu.edu/user/achoulde/94842/>

by CC license

Slides 3-11, adapted from

<https://github.com/rstudio-education/remaster-the-tidyverse/>

By CC license

Slide 10 adapted from

<https://education.rstudio.com/blog/2020/07/teaching-the-tidyverse-in-2020-part-1-getting-started/> by CC license