TEACHING R TO COWORKERS

Erin Grand

Uncommon Schools | Change History.

@astroeringrand

WHY?

David Robinson

Chief Data Scientist at DataCamp, works in R and Python.

Email
Twitter
Github

Teach the tidyverse to beginners

A few years ago, I wrote a post Don't teach built-in plotting to beginners (teach ggplot2). I argued that ggplot2 was not an advanced approach meant for experts, bu rather a suitable introduction to data visualization.

> *Many teachers suggest I'm overestimating their students: "No, see, my students are beginners...". If I push the point, they might insist I'm not understanding just how much of a beginner these students are, and emphasize they're looking to keep it simple and teach the basics, and that that students can get to the advanced methods later....*

> ***My claim is that this is precisely backwards.*** *ggplot2 is easier to teach beginners, not harder, and makes constructing plots simpler, not more complicated.*

@astroeringrand

Previously…

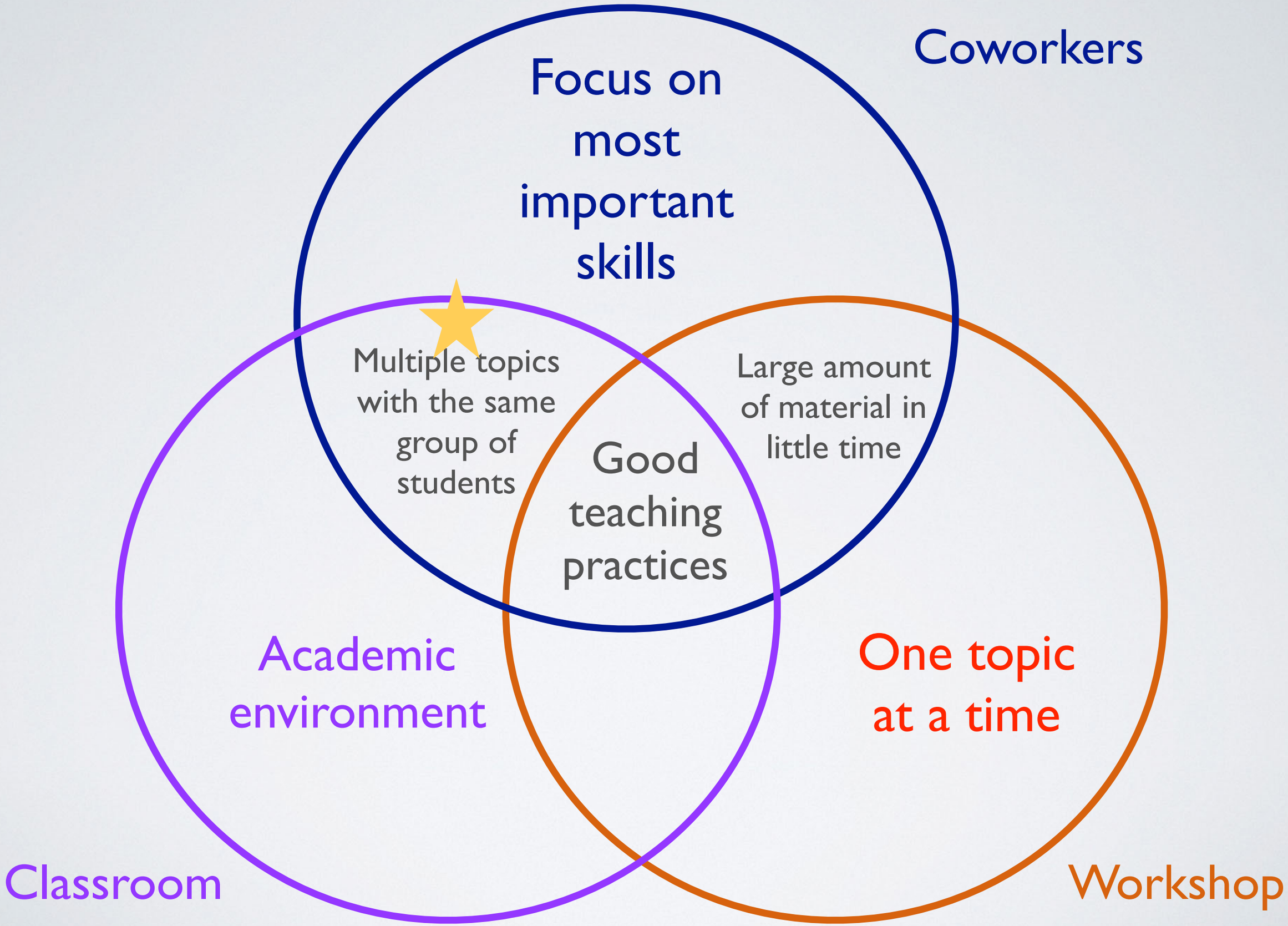
- Manual systems - take lots of time
- Using different tools to process the same data
- Manual edits cause human errors in the data processing

Why R?

- Free: R and RStudio are free and open sourced
- Reproducible: Easy to run scripts again with updated data.
- Collaboration: Results are easy to share as a PDF.
- Manipulation: Data manipulation is much easier due to the tidyverse.
- Understandable: You can easily follow along in someone else's code.
- Spot Checking: You can QC yourself along the way, instead of all at the end.
- Help: Finding answers to your programming questions is a quick google away.

WHAT MAKES TRAINING COWORKERS DIFFERENT?

What makes training coworkers different?



What makes training coworkers different?

- Face to face time with the same group of people for multiple sessions
- Ability to design sessions with the skills that are most needed for a specific group of people
- Pivot when needed based on feedback
- Practice with company data

HOW?

Preparing for Good Training

1. Tools to measure current skill level 🛠️
2. Structures to produce PD sessions that align with skill level growth 📖
3. Create training materials and pre-work 💡
4. Plan for practice / homework 📚
5. Feedback 💬
6. Repeat steps 3 - 5 for all sessions defined in step 2

@astroeringrand

Tools to measure current skill level 🛠️

	New		Beginner			Proficient	
	1	2	3	4	5	6	7
Package List		readr stringr (very basics) datapasta (addin)	dplyr	dplyr (continued) janitor databases: DBI, ODBC, RODC (connection basics) WriteXL (requires preparing of computer)	dplyr tidyr	dplyr tidyr purrr (map_dfr) assertr here	stringr purrr
Functionand Skills List	- Syntax (i.e. use of <- vs =) - Data Types - Operators & Comparative operators - Loading R packages - Calling functions - Naming variables - Basic indexing - Rstudio workflow - Commenting	- Reading in files as data frames - Factors vs strings - Dates and Times (basics) - Changing between data types - Creation of lookup table - Check for missing values	- select - filter - mutate - group by & summarize - *_join - pipe	- get_dupes - clean_names - arrange - group by & mutate - Connect to a SQL database and query the database in R - case_when - WriteXL	- spread & gather - unite & separate - mutate_* statements - Start writing functions	- map_dfr - assert & verify - Rnotebook parameters	- string functions - basic regular expressions - Work with lists and lists of lists
Workflow and Style	Introduction to the Rstudio IDE. Install and update packages inside Rstudio and be able to identify where new objects are shown and where to search for help. Inside a Rnotebook, write both R code and real text in the same file, knowing how and when to separate R chunks.	- Basic understanding of difference between Rmarkdown and .R script. - Continue to include comments in text and code to explain business rules or decisions made along the way. - Best practices for Project structure. (Input, Output, Scripts...etc) - Writing cleaned data files to an output directory - Computer literacy around file paths and basic coding principles.	Write R script to produce a full analysis without errors. Style of code should follow the Tidyverse style guide. http://style.tidyverse.org/	Script should be using pipes all the way through, and obey the general rules of pipes and tidyverse styling.	Understand tidy data principles, and be able to visualize the form the data needs to take to proceed (long or wide). Don't repeat code more than twice, use functions to replace this repetition. Organize functions and code with comments explaining choices made.	Use best practices around SQL queries in Rstudio. Use R projects to house data work	Write unit tests into script.

Rubric measure skills in 3 categories (packages, functions and skills, and workflow)
and
covers Beginner to Proficient levels

@astroeringrand

Example Scope and Sequence

- Session 1: Data Transformation
- Session 2: Cleaning Data
- Session 3: Tidy Data
- Session 4: String Manipulation
- Session 5: SQL in RStudio
- Session 6: Unit Testing

Structures to produce training sessions that align with skill level growth 📖

Use a scope and sequence to allow for detailed planning of a topic.

- Level
- Topic
- Skill
- Outcome goals
- Practice actives
- Required materials
- Time

Organize skills into topics, and topics into individual PD sessions.

Topic: Cleaning Data

Skill: janitor::get_dupes()

Goal: Students will be able to find and fix duplicates.

Practice: Students will clean a student roster with duplicates caused by different grade, dates and schools.

Required Materials: Student roster set up with duplicates

Create PD Materials 💡

1. Slides
2. R notebook examples and activities
3. Cheat Sheet
4. Homework

Note: All of the RStudio Tidyverse workshop materials are available on Github - go forth and use them!

@astroeringrand

Data Wrangling
with dplyr and tidyr
Cheat Sheet

RStudio

Syntax - Helpful conventions for wrangling

dplyr::**tbl_df(iris)**
Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

```
Source: local data frame [150 x 5]

   Sepal.Length Sepal.Width Petal.Length
1           5.1         3.5          1.4
2           4.9         3.0          1.4
3           4.7         3.2          1.3
4           4.6         3.1          1.5
5           5.0         3.6          1.4
..          ...         ...          ...
Variables not shown: Petal.Width (dbl),
  Species (fctr)
```

dplyr::**glimpse(iris)**
Information dense summary of tbl data.
utils::**View(iris)**
View data set in spreadsheet-like display (note capital V).

dplyr::**%>%**
Passes object on left hand side as first argument (or . argument) of function on righthand side.

```
x %>% f(y)  is the same as  f(x, y)
y %>% f(x, .., z)  is the same as  f(x, y, z)
```

"Piping" with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

Tidy Data - A foundation for wrangling in R

In a tidy data set:

Each **variable** is saved in its own **column**

Each **observation** is saved in its own **row**

Tidy data complements R's vectorized operations. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.

Reshaping Data - Change the layout of a data set

tidy::**gather(cases, "year", "n", 2:4)**
Gather columns into rows.

tidy::**spread(pollution, size, amount)**
Spread rows into columns.

tidy::**separate(storms, date, c("y", "m", "d"))**
Separate one column into several.

tidy::**unite(data, col, ..., sep)**
Unite several columns into one.

dplyr::**data_frame(a = 1:3, b = 4:6)**
Combine vectors into data frame (optimized).
dplyr::**arrange(mtcars, mpg)**
Order rows by values of a column (low to high).
dplyr::**arrange(mtcars, desc(mpg))**
Order rows by values of a column (high to low).
dplyr::**rename(tb, y = year)**
Rename the columns of a data frame.

Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**
Extract rows that meet logical criteria.
dplyr::**distinct(iris)**
Remove duplicate rows.
dplyr::**sample_frac(iris, 0.5, replace = TRUE)**
Randomly select fraction of rows.
dplyr::**sample_n(iris, 10, replace = TRUE)**
Randomly select n rows.
dplyr::**slice(iris, 10:15)**
Select rows by position.
dplyr::**top_n(storms, 2, date)**
Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&, |, !, xor, any, all	Boolean operators

Subset Variables (Columns)

dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**
Select columns by name or helper function.

Helper functions for select - ?select

select(iris, **contains(".")**)
Select columns whose name contains a character string.
select(iris, **ends_with("Length")**)
Select columns whose name ends with a character string.
select(iris, **everything()**)
Select every column.
select(iris, **matches(".t.")**)
Select columns whose name matches a regular expression.
select(iris, **num_range("x", 1:5)**)
Select columns named x1, x2, x3, x4, x5.
select(iris, **one_of(c("Species", "Genus"))**)
Select columns whose names are in a group of names.
select(iris, **starts_with("Sepal")**)
Select columns whose name starts with a character string.
select(iris, **Sepal.Length:Petal.Width**)
Select all columns between Sepal.Length and Petal.Width (inclusive).
select(iris, **-Species**)
Select all columns except Species.

devtools::install_github("rstudio/EDAWR") for data sets
Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0 • tidyr 0.2.0 • Updated: 1/15

Make New Variables

dplyr::**mutate(iris, sepal = Sepal.Length + Sepal. Width)**
Compute and append one or more new columns.
dplyr::**mutate_each(iris, funs(min_rank))**
Apply window function to each column.
dplyr::**transmute(iris, sepal = Sepal.Length + Sepal. Width)**
Compute one or more new columns. Drop original columns.

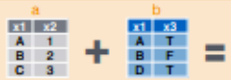
Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

dplyr::**lead**
Copy with values shifted by 1.
dplyr::**lag**
Copy with values lagged by 1.
dplyr::**dense_rank**
Ranks with no gaps.
dplyr::**min_rank**
Ranks. Ties get min rank.
dplyr::**percent_rank**
Ranks rescaled to [0, 1].
dplyr::**row_number**
Ranks. Ties got to first value.
dplyr::**ntile**
Bin vector into n buckets.
dplyr::**between**
Are values between a and b?
dplyr::**cume_dist**
Cumulative distribution.

dplyr::**cumall**
Cumulative `all`
dplyr::**cumany**
Cumulative `any`
dplyr::**cummean**
Cumulative `mean`
cumsum
Cumulative `sum`
cummax
Cumulative `max`
cummin
Cumulative `min`
cumprod
Cumulative `prod`
pmax
Element-wise `max`
pmin
Element-wise `min`

iris %>% group_by(Species) %>% **mutate(...)**
Compute new variables by group.

Combine Data Sets



Mutating Joins

dplyr::**left_join(a, b, by = "x1")**
Join matching rows from b to a.

dplyr::**right_join(a, b, by = "x1")**
Join matching rows from a to b.

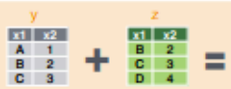
dplyr::**inner_join(a, b, by = "x1")**
Join data. Retain only rows in both sets.

dplyr::**full_join(a, b, by = "x1")**
Join data. Retain all values, all rows.

Filtering Joins

dplyr::**semi_join(a, b, by = "x1")**
All rows in a that have a match in b.

dplyr::**anti_join(a, b, by = "x1")**
All rows in a that do not have a match in b.



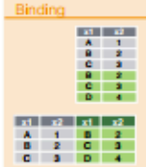
Set Operations

dplyr::**intersect(y, z)**
Rows that appear in both y and z.

dplyr::**union(y, z)**
Rows that appear in either or both y and z.

dplyr::**setdiff(y, z)**
Rows that appear in y but not z.

Binding



dplyr::**bind_rows(y, z)**
Append z to y as new rows.

dplyr::**bind_cols(y, z)**
Append z to y as new columns.
Caution: matches rows by position.

Summarize

Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

dplyr::**first**
First value of a vector.
dplyr::**last**
Last value of a vector.
dplyr::**nth**
Nth value of a vector.
dplyr::**n**
# of values in a vector.
dplyr::**n_distinct**
# of distinct values in a vector.
IQR
IQR of a vector.

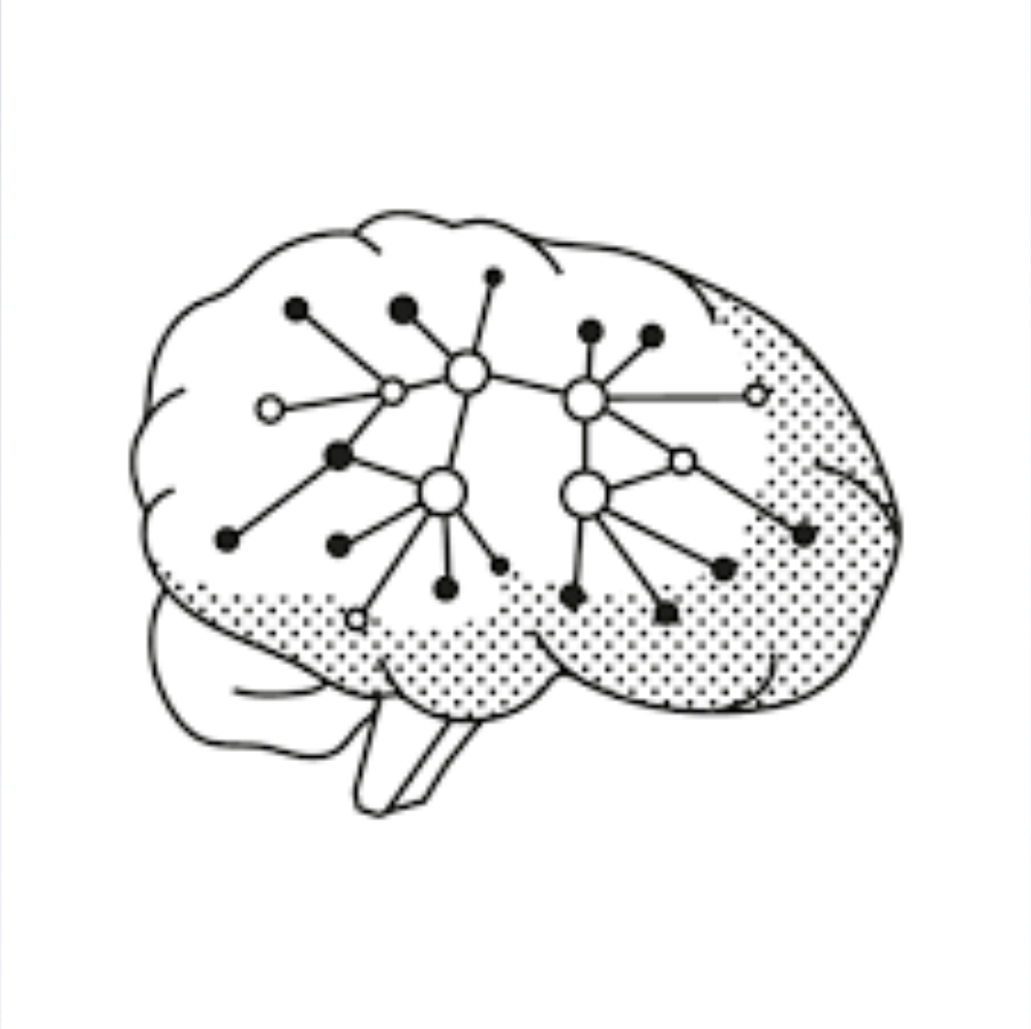
min
Minimum value in a vector.
max
Maximum value in a vector.
mean
Mean value of a vector.
median
Median value of a vector.
var
Variance of a vector.
sd
Standard deviation of a vector.

Group Data

dplyr::**group_by(iris, Species)**
Group data into rows with the same value of Species.
dplyr::**ungroup(iris)**
Remove grouping information from data frame.

iris %>% group_by(Species) %>% **summarise(...)**
Compute separate summary row for each group.

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com
devtools::install_github("rstudio/EDAWR") for data sets
Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0 • tidyr 0.2.0 • Updated: 1/15



Use best practices for teaching. Keep an ideal mental model in mind to guide your teaching.

@astroeringrand

Feedback 💬

1. Check in with your coworkers to assess their skills and progress

2. Create new goals together

3. Have action steps for how to reach those goals, including the completion of "homework"

@astroeringrand

WHAT HAVE I LEARNED?

Best Practices

- Use data sets that coworkers are used to seeing and working with in examples
- Record sessions, if possible
- Before the first session, send instructions on how to set up the technology. Hold office hours to help anyone that needs it
- Introduce a style guide early on to align everyone on the same page
- Be involved with the R community in order to stay up to date on breaking changes and cool new packages

@astroeringrand