Interactive R tutorials with `learnr`

Mine Çetinkaya-Rundel
Duke University & RStudio

@minebocek

mine-cetinkaya-rundel

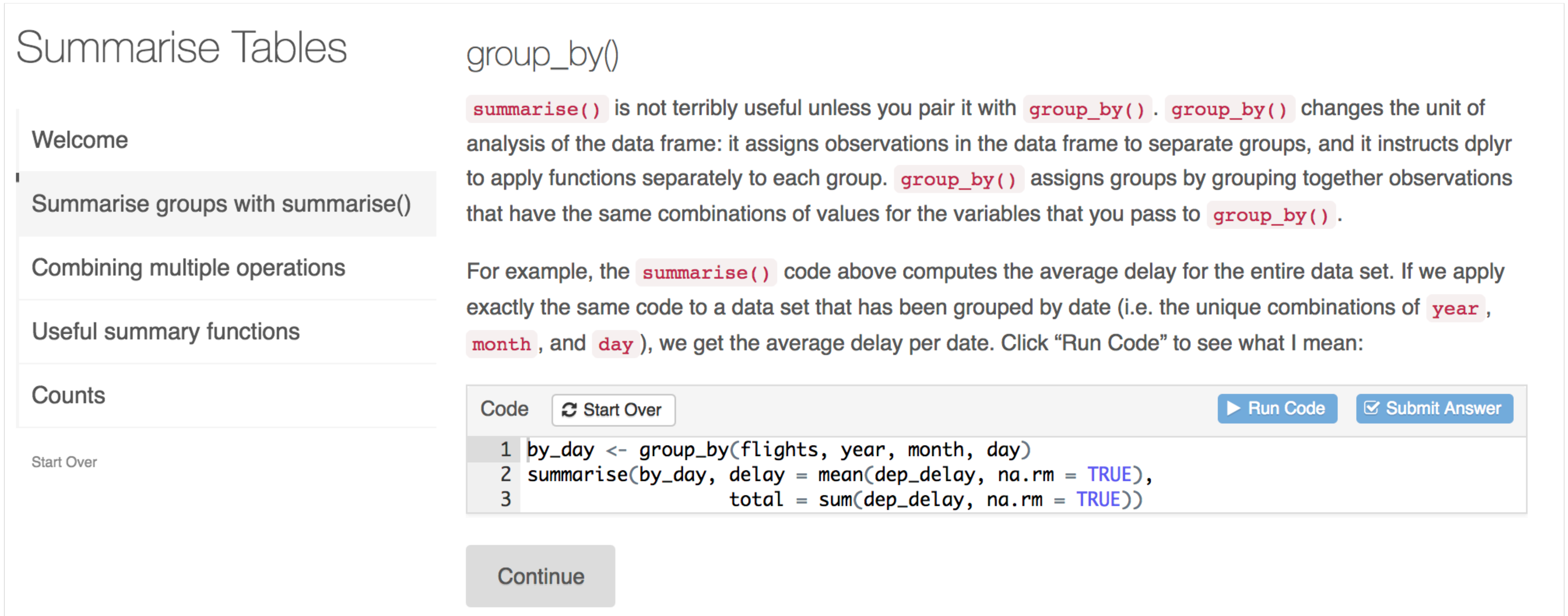
mine@stat.duke.edu

CAUSE **Webinar**
May 8, 2018

Assumed background

- Assuming you know some R
- Assuming you teach R to others (though not necessarily…)

At a glance



group_by()

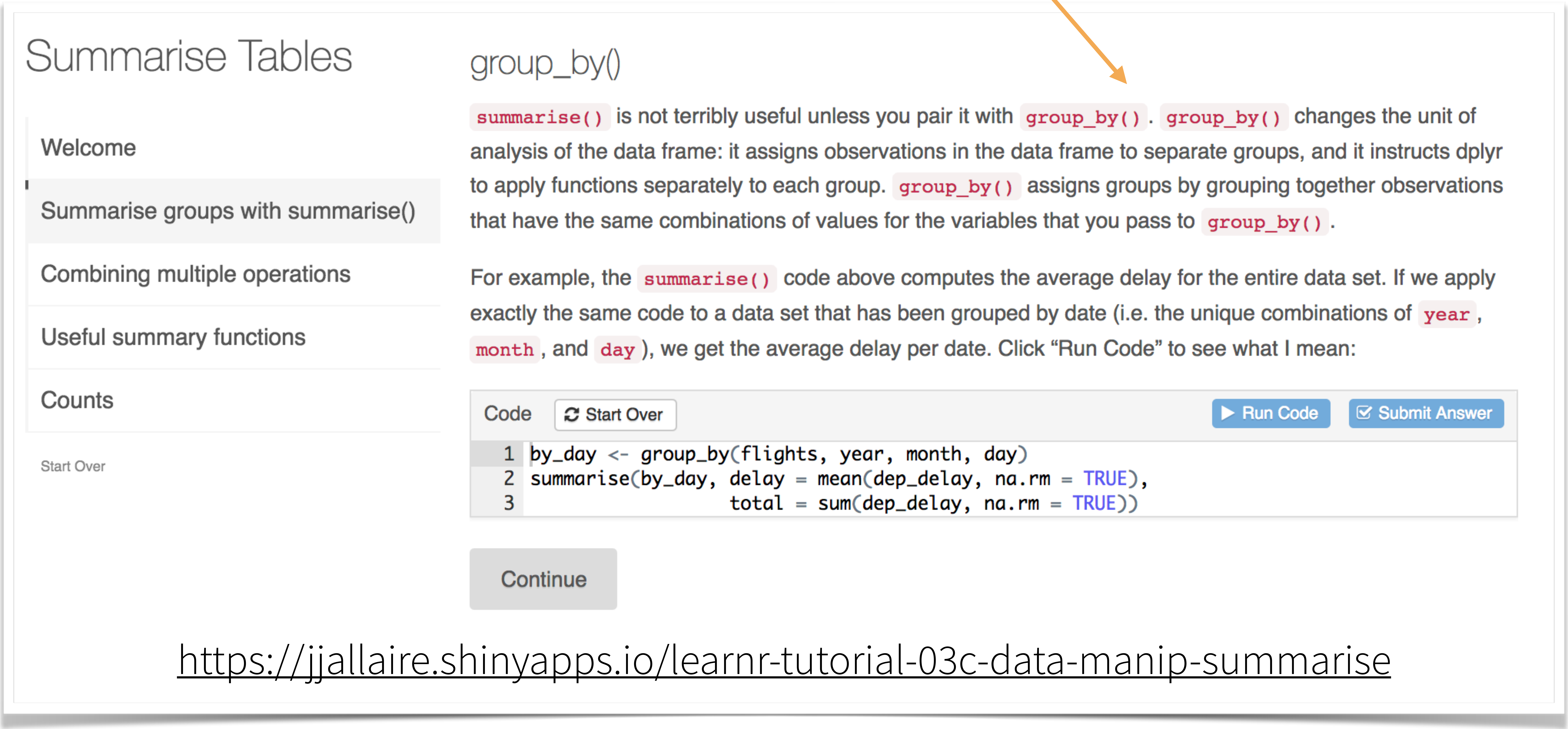
`summarise()` is not terribly useful unless you pair it with `group_by()`. `group_by()` changes the unit of analysis of the data frame: it assigns observations in the data frame to separate groups, and it instructs dplyr to apply functions separately to each group. `group_by()` assigns groups by grouping together observations that have the same combinations of values for the variables that you pass to `group_by()`.

For example, the `summarise()` code above computes the average delay for the entire data set. If we apply exactly the same code to a data set that has been grouped by date (i.e. the unique combinations of `year`, `month`, and `day`), we get the average delay per date. Click “Run Code” to see what I mean:

```r
by_day <- group_by(flights, year, month, day)
summarise(by_day, delay = mean(dep_delay, na.rm = TRUE),
                  total = sum(dep_delay, na.rm = TRUE))
```

https://jjallaire.shinyapps.io/learnr-tutorial-03c-data-manip-summarise

narrative



group_by()

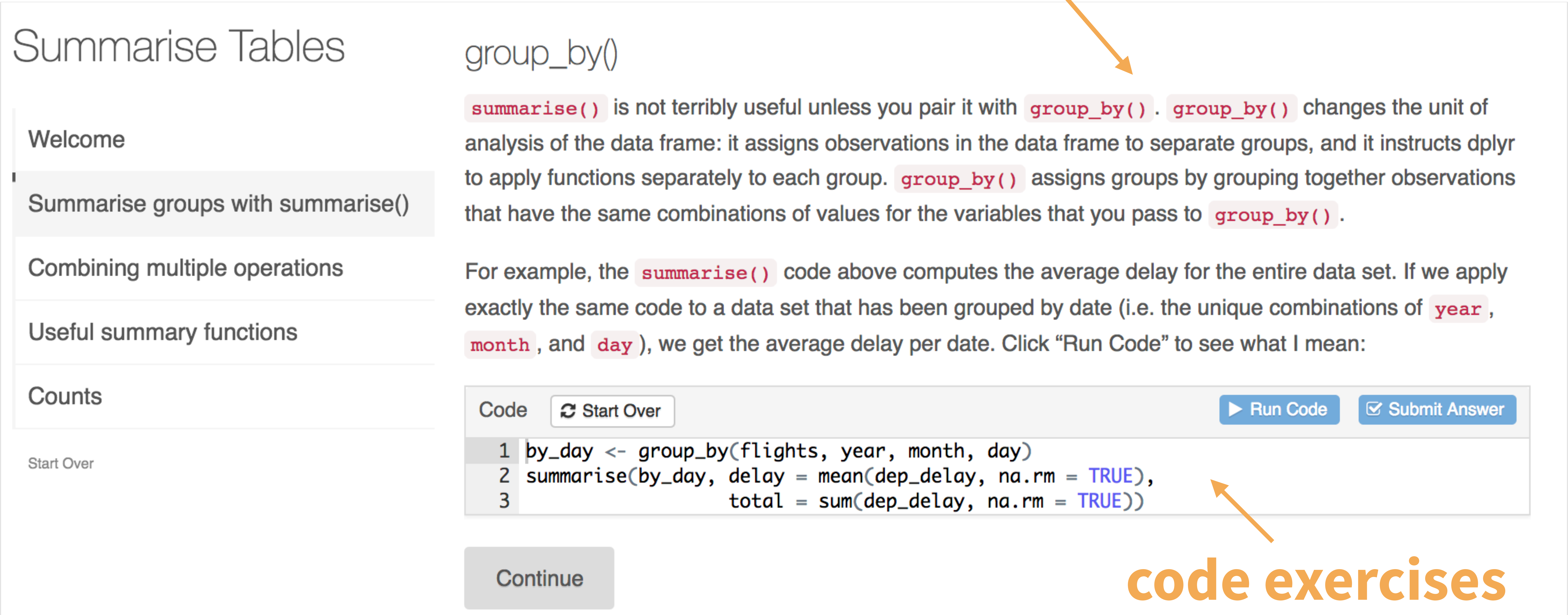
`summarise()` is not terribly useful unless you pair it with `group_by()`. `group_by()` changes the unit of analysis of the data frame: it assigns observations in the data frame to separate groups, and it instructs dplyr to apply functions separately to each group. `group_by()` assigns groups by grouping together observations that have the same combinations of values for the variables that you pass to `group_by()`.

For example, the `summarise()` code above computes the average delay for the entire data set. If we apply exactly the same code to a data set that has been grouped by date (i.e. the unique combinations of `year`, `month`, and `day`), we get the average delay per date. Click “Run Code” to see what I mean:

```
by_day <- group_by(flights, year, month, day)
summarise(by_day, delay = mean(dep_delay, na.rm = TRUE),
                  total = sum(dep_delay, na.rm = TRUE))
```

https://jjallaire.shinyapps.io/learnr-tutorial-03c-data-manip-summarise

narrative



group_by()

`summarise()` is not terribly useful unless you pair it with `group_by()`. `group_by()` changes the unit of analysis of the data frame: it assigns observations in the data frame to separate groups, and it instructs dplyr to apply functions separately to each group. `group_by()` assigns groups by grouping together observations that have the same combinations of values for the variables that you pass to `group_by()`.

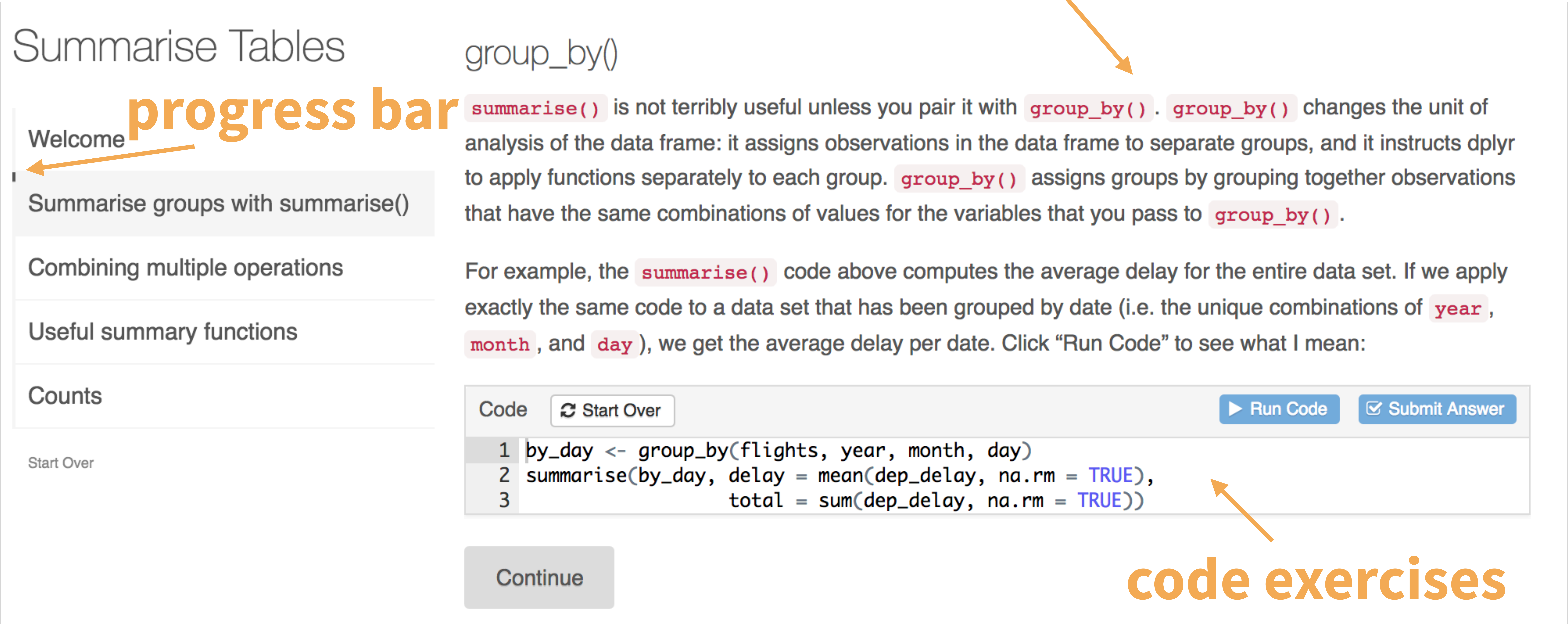
For example, the `summarise()` code above computes the average delay for the entire data set. If we apply exactly the same code to a data set that has been grouped by date (i.e. the unique combinations of `year`, `month`, and `day`), we get the average delay per date. Click “Run Code” to see what I mean:

```r
by_day <- group_by(flights, year, month, day)
summarise(by_day, delay = mean(dep_delay, na.rm = TRUE),
                  total = sum(dep_delay, na.rm = TRUE))
```

code exercises

https://jjallaire.shinyapps.io/learnr-tutorial-03c-data-manip-summarise

narrative



Summarise Tables

Welcome
Summarise groups with summarise()
Combining multiple operations
Useful summary functions
Counts
Start Over

progress bar

group_by()

`summarise()` is not terribly useful unless you pair it with `group_by()`. `group_by()` changes the unit of analysis of the data frame: it assigns observations in the data frame to separate groups, and it instructs dplyr to apply functions separately to each group. `group_by()` assigns groups by grouping together observations that have the same combinations of values for the variables that you pass to `group_by()`.

For example, the `summarise()` code above computes the average delay for the entire data set. If we apply exactly the same code to a data set that has been grouped by date (i.e. the unique combinations of `year`, `month`, and `day`), we get the average delay per date. Click “Run Code” to see what I mean:

```r
by_day <- group_by(flights, year, month, day)
summarise(by_day, delay = mean(dep_delay, na.rm = TRUE),
                  total = sum(dep_delay, na.rm = TRUE))
```

code exercises

https://jjallaire.shinyapps.io/learnr-tutorial-03c-data-manip-summarise

Uses

Within a course

- Flipped classroom:
  - Assign a `learnr` tutorial, including narrative and implementation in R that students can practice with, before introducing a concept in class
  - Cover the concept in class (quicker)
  - Allocate the time saved to hands on exercises in class
- Lecture follow-up
  - Provide the same content from the lecture as follow up exercises
- Lab exercises / assignments

Self learning

- Learn by doing
- Package tutorials
- Workshop follow ups

Demo

Roadmap

- Narrative, figures, illustrations, and equations
- Code exercises (R code chunks that students can edit and execute directly)
- Quiz questions
- Videos (supported services include YouTube and Vimeo)
- Interactive Shiny components

Getting started

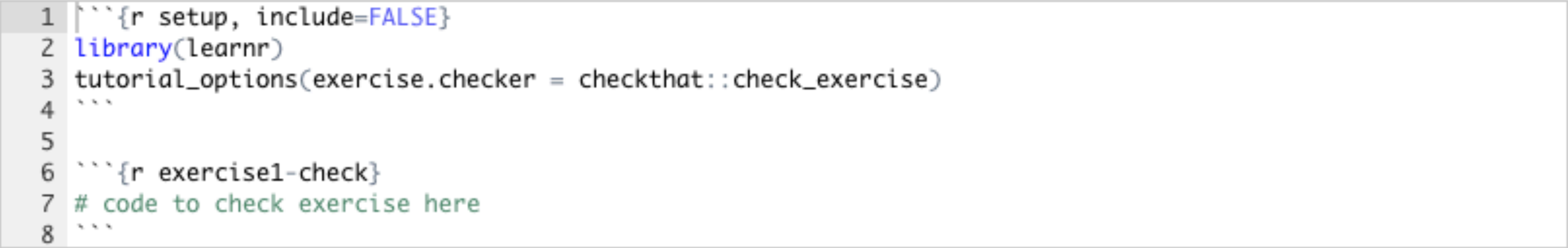
- Follow along options:
  - Local: In RStudio, install and load the `learnr` package
  - Cloud: Go to XXX

- File → New File → R Markdown... → From template → Interactive Tutorial

What’s next?

Code checking

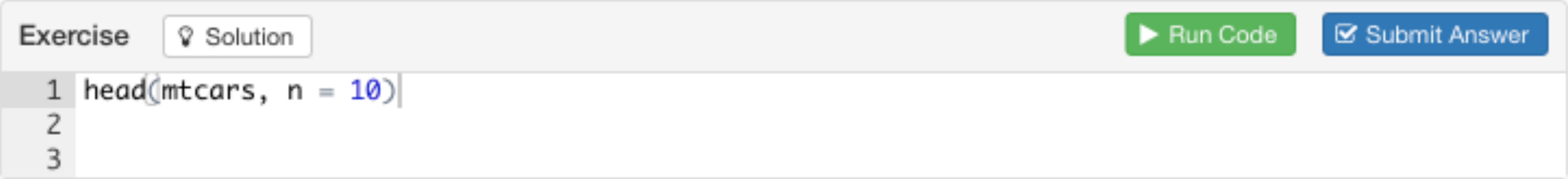
- No built in code checking feature, but hooks for using other packages for code checking
  - `checkr` by Danny Kaplan: github.com/dtkaplan/checkr
  - `grader` by Garrett Grolemund: github.com/rstudio-education/grader
- In the `setup` chunk of the tutorial: set the `exercise.checker` option to , and then add a "-check" chunk for any exercise you want to check



~~~
```{r setup, include=FALSE}
library(learnr)
tutorial_options(exercise.checker = checkthat::check_exercise)
```


```{r exercise1-check}
# code to check exercise here
```
~~~

When you provide a “-check” chunk for an exercise, an additional “Submit Answer” button is added to the exercise editor:



```
head(mtcars, n = 10)
```

This is provided so that users can experiment with various solutions before formally submitting an answer they believe is correct.

Recording events

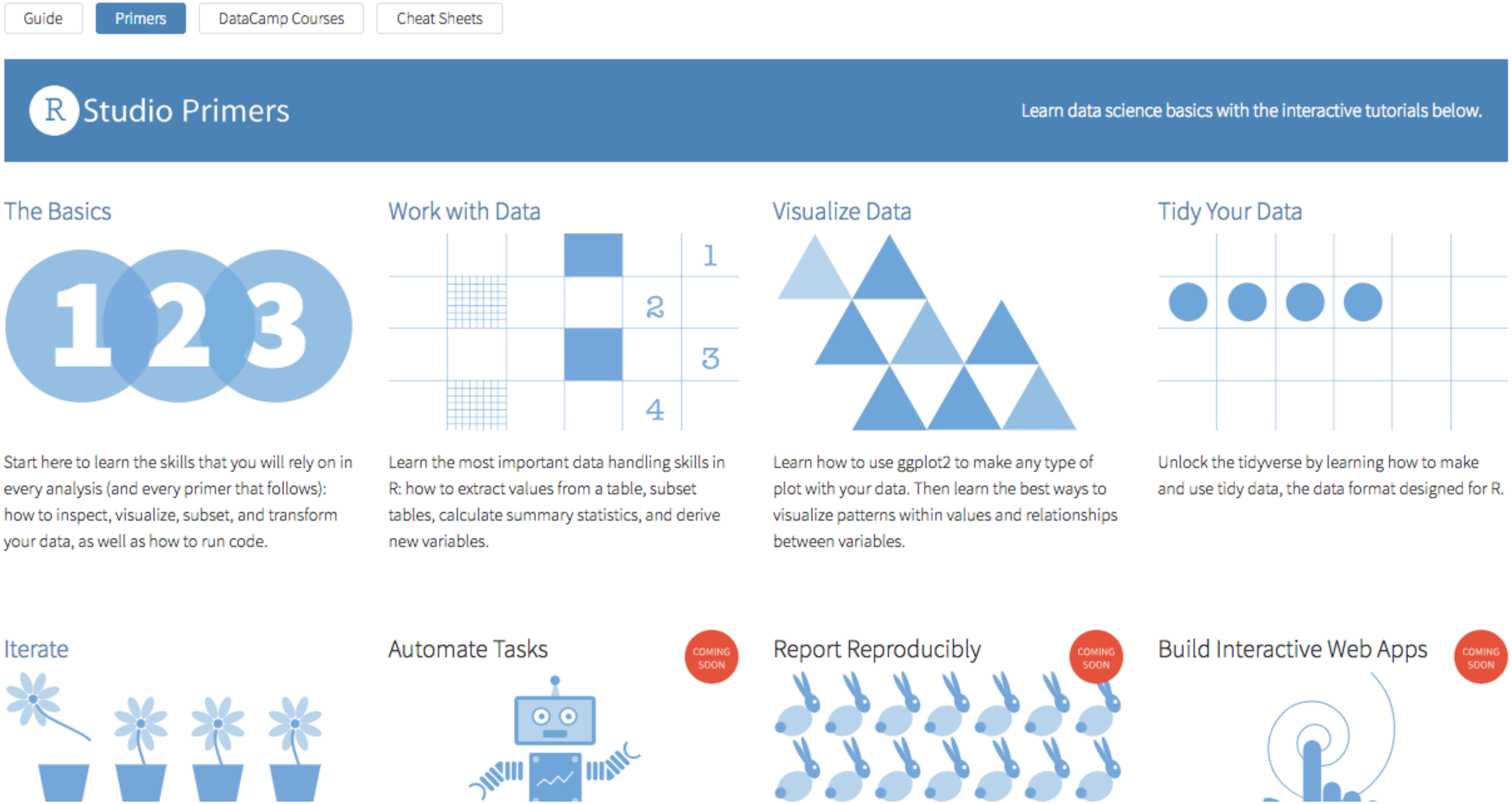
- Recording events like exercise and question submissions, requests for hints/solutions, etc.

- This is possible with `learnr`, though not very simple

- With other R tools that allow for writing out to spreadsheets (e.g. Google Sheets) and building dashboards (e.g. `shinydashboard`) it's possible to build a dashboard for your class where you can track their progress and learn from what they're struggling with

Resources

Try rstudio.cloud/learn/primers



Learn data science basics with the interactive tutorials below.

The Basics

Start here to learn the skills that you will rely on in every analysis (and every primer that follows): how to inspect, visualize, subset, and transform your data, as well as how to run code.

Work with Data

Learn the most important data handling skills in R: how to extract values from a table, subset tables, calculate summary statistics, and derive new variables.

Visualize Data

Learn how to use ggplot2 to make any type of plot with your data. Then learn the best ways to visualize patterns within values and relationships between variables.

Tidy Your Data

Unlock the tidyverse by learning how to make and use tidy data, the data format designed for R.

Iterate

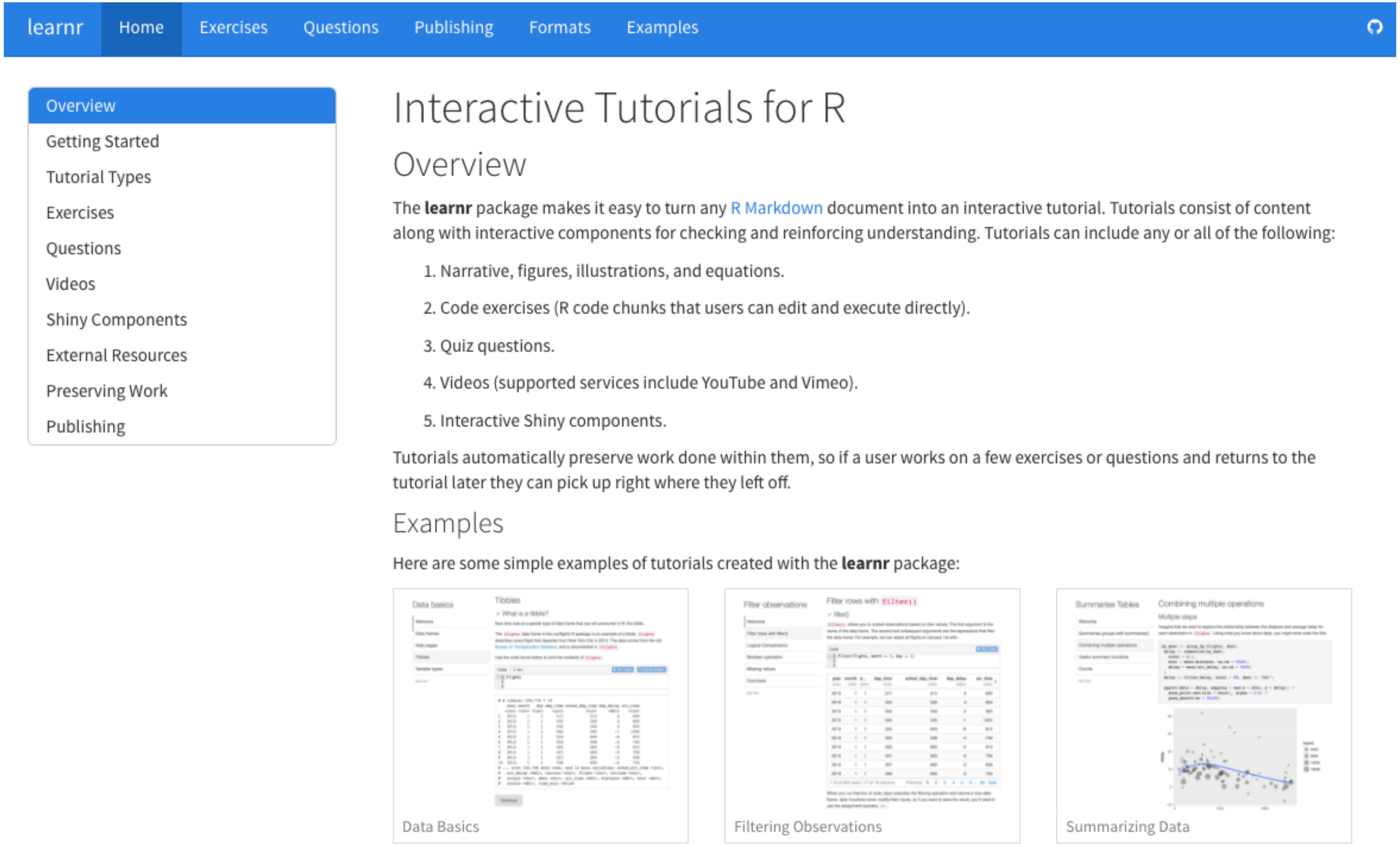
Automate Tasks

Report Reproducibly

Build Interactive Web Apps

Build

rstudio.github.io/learnr/



Interactive Tutorials for R

Overview

The **learnr** package makes it easy to turn any R Markdown document into an interactive tutorial. Tutorials consist of content along with interactive components for checking and reinforcing understanding. Tutorials can include any or all of the following:

1. Narrative, figures, illustrations, and equations.
2. Code exercises (R code chunks that users can edit and execute directly).
3. Quiz questions.
4. Videos (supported services include YouTube and Vimeo).
5. Interactive Shiny components.

Tutorials automatically preserve work done within them, so if a user works on a few exercises or questions and returns to the tutorial later they can pick up right where they left off.

Examples

Here are some simple examples of tutorials created with the **learnr** package:

Data Basics

Filtering Observations

Summarizing Data

Review

github.com/mine-cetinkaya-rundel/cause-learnr github.com/mine-cetinkaya-rundel/cause-learnr

@minebocek

mine-cetinkaya-rundel

mine@stat.duke.edu