Hey there, data pro.

You've come a long way

since the beginning of your learning journey.

Congratulations on your accomplishment.

Just think of all the skills

you've learned along the way.

You now know how to use

structured thinking to define

a problem and ask the right questions;

work with spreadsheets, databases,

and tools like SQL to organize and transform data;

clean your data to make sure it

has integrity before you analyze it;

create impactful visuals to illustrate key points; and

craft a compelling story to

communicate insights to stakeholders.

That's an impressive list of skills,

but we're not done yet.

Your skills set's about to get even bigger.

In this course, you'll learn about a new concept called

programming and how you can use

the R programming language to analyze your data.

By now, you know that

the data analysis process includes six phases: ask,

prepare, process, analyze, share, and act.

Now, we'll learn all about

the R programming language and how it can

help you in each phase of the process.

When you're done, you'll be

presented with an optional case study.

The case study will give you the chance to solve

a data analysis problem

using all the skills you've learned in this program.

You'll find out more about this project later on.

Let's talk about computer programming.

Computer programming refers to giving instructions to

a computer to perform an action or set of actions.

You can use different programming languages

to write these instructions.

You might choose a specific language based on

the project you want to pursue

or the problem you want to solve.

The R programming language is

super useful for organizing,

cleaning, and analyzing data.

If this is your first experience

with computer programming, welcome.

When I first started learning about data analysis,

I didn't have a background in programming either.

In fact, before I fell in love with data,

I was trained as an opera singer.

I also have a lot of friends

that came into this field from

the arts and learned about

programming later in their career.

R is a great place to start.

Learning R for the first time can be

challenging and even more empowering.

A lot of the skills you've learned in this program

will help you learn basic programming concepts.

Take it one step at a time and go at your own pace.

Just like in previous videos,

you'll start with the basics and move forward from there.

You've tackled tough challenges before and you

always come out on top. You've got this.

Let me introduce myself.

My name's Carrie. I work as a research manager at Google.

I lead a team that researches the best way to

improve the performance of people in organizations.

In other words, I help people work better and work

smarter and help organizations

function in a healthy and productive way.

I first learned R as a junior data analyst,

while I was working on a multi-year project

about virtual work.

We were looking at data on

people's virtual work experiences

and trying to understand how

working remotely impacts performance.

It was a complex project

with a lot of data to sift through.

I kept encountering problems and searching

for better and faster ways to do things.

This is when I became aware of the power of

R. Whenever I got stuck,

I'd learn a little more about

R and discover a solution to my problem.

I soon realized that R could help me

do almost anything involving

data even better and faster than I thought possible.

Fortunately, there are tons of

great online resources for R

and a super supportive online community.

If I had a question,

I'd go online and find the answer.

As the project progressed,

I was able to learn more and more and

become a much more effective data analyst.

My teammates even started coming to me for

advice about R. Realizing that

I could continue to learn my skills at any stage in

my career was an empowering experience.

Learning R unlocked my ability to

perform data analysis at the highest level.

In your future career as a data analyst,

you'll have the opportunity to

continually learn and grow.

To me, that might be

one of the coolest aspects of the job.

Learning R is one of

the most rewarding parts of that growth process.

I'm still learning new ways to use R all the time.

Plus, you can apply these skills to

other programming languages like

Python, Julia, or JavaScript.

There's no limit to how far you can go with programming.

It even goes beyond data analysis.

After I learned R, I found myself thinking

about all kinds of projects I could use programming for,

both at work and for fun.

It opens up a whole new world of possibilities.

Now, let's talk about what you're about to learn.

We'll start off with

an introduction to programming languages.

Then we'll take a closer look at R

itself and explore its main features and functions.

We'll also cover some basic programming concepts and

learn how to use them effectively in R. Next,

we'll learn how to work with data in R.

You'll discover how R can supercharge

your data analysis skills and let you clean,

transform, visualize, and report

data in new and more powerful ways.

Learning R will help you take

your data analysis to the next level.

It'll also look great on your resume.

R is widely recognized as

a key credential in entry- level job positions.

Knowing how to use R will give you a big boost in

your job hunt and will help you

stand out as a new analyst.

Coming up, we'll talk more about programming languages in

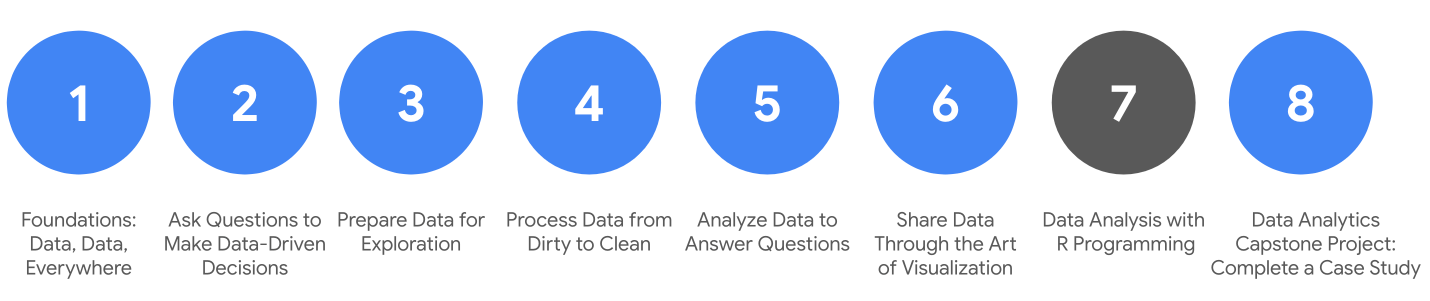
general and how they can help you analyze your data.

After that, we'll jump right into R. Before you know it,

you'll be using R to power your data analysis.

**Course syllabus**

In previous courses, you learned how to use structured thinking to solve business problems; prepare, clean, transform, and analyze data in spreadsheets and databases; and tell effective data stories. As part of your growing skillset, you also learned how to create dynamic and interactive data visualizations in Tableau. Up until now, the skills you learned were closely tied to the features and capabilities available in spreadsheets, SQL databases, and Tableau. But what if you want to work with your data in more custom ways? Or what if the standard tools don’t have the exact functionality you need? This is when the R programming language can be super helpful. Using R, you will gain additional flexibility and control over your data and analysis.



1. [Foundations: Data, Data, Everywhere](https://www.coursera.org/learn/foundations-data/home/welcome)
2. [Ask Questions to Make Data-Driven Decisions](https://www.coursera.org/learn/ask-questions-make-decisions/home/welcome)
3. [Prepare Data for Exploration](https://www.coursera.org/learn/data-preparation/home/welcome)
4. [Process Data from Dirty to Clean](https://www.coursera.org/learn/process-data/home/welcome)
5. [Analyze Data to Answer Questions](https://www.coursera.org/learn/analyze-data/home/welcome)
6. [Share Data through the Art of Visualization](https://www.coursera.org/learn/visualize-data/home/welcome)
7. **Data Analysis with R Programming** *(this course)*
8. [Google Data Analytics Capstone: Complete a Case Study](https://www.coursera.org/learn/google-data-analytics-capstone/home/welcome)

In this course, you will learn how to use the R programming language to work with your data without tool limitations. You will get plenty of practice using R for statistical analysis, and RStudio—an integrated developer environment (IDE) for R that you will use to create advanced data visualizations with lots of detail. R makes it easier to present your data with beautiful, artistic style. A few other advantages of R include its:

* **Popularity**: R is frequently used for data analysis
* **Tools:** R has a convenient library of ready-to-use tools for data cleaning and analysis
* **Focus**: R was created with statistics in mind; data analysts can conveniently use a rich library of statistical routines
* **Adaptability**: R adapts well for use in both machine learning and data analysis projects
* **Availability**: R is an open source programming language

After you get comfortable and more confident using R and RStudio, you might find that you are curious to learn and add even more programming languages to your skillset (and resume). Pretty exciting, right?

**Course content**

Course 7 – Data Analysis with R Programming

1. **Understanding the basics of R:** R is a programming language that can be used to perform tasks in every phase of the data analysis process. In this part of the course, you will learn about R and RStudio, an integrated developer environment (IDE) for R. You will explore the benefits of using RStudio to work with R. RStudio enables you to easily leverage the features and functionality of R.
2. **Programming using RStudio:** In this part of the course, you will explore the fundamental concepts associated with R. You will learn about functions and variables that you can use in your calculations and other programming. You will also learn about R packages, which are collections of R functions, code, and sample data that you can use in RStudio.
3. **Working with data in R:** The R programming language was designed to work with data at all stages of the data analysis process. In this part of the course, you will examine how R can help you structure, organize, and clean your data through functions and other processes. You will learn about data frames and how to work with them in R. You will also revisit the concept of data bias and how you can use R to address it.
4. **Visualizations, aesthetics, and annotations:** R is a great tool for creating detailed visualizations. In this part of the course, you will learn how to use R to generate and troubleshoot visualizations. You will also explore the features of R and RStudio that can help you improve the aesthetics of your visualizations. You will learn how to annotate visualizations and save the changes.
5. **Documentation and reports:** R has a number of different options to explore when you are ready to save and present your analysis. In this part of the course, you will explore R Markdown, a file format for making dynamic documents with R. You will learn how to format and export R Markdown and incorporate R code chunks in your documents.
6. **Course challenge:** At the end of the course you will apply everything you have learned in the Course Challenge. The Course Challenge will ask you questions about the key skills you have been practicing and will give you an opportunity to demonstrate those skills in three scenarios.

**Are you already familiar with R programming?**

If you have used R and RStudio before, you might find the first two modules of this course a review of basic topics that you already understand. Feel free to skip these foundational videos and readings and proceed to the module challenges for Module 1 and Module 2. The module challenges will help prepare you for the course challenge at the end of this course. To earn the certificate, you need to score 80% or higher on all graded activities in the program.

**What to expect**

You can expect to finish this course in about four to five weeks. That involves completing all the activities, including:

* **Videos** of instructors teaching new concepts and demonstrating the use of tools
* **In-video questions** that pop up during or at the end of a video to check your learning
* **Readings** to introduce new ideas and build on the concepts from the videos
* **Discussion forums** to discuss, explore, and reinforce new ideas for better learning
* **Discussion prompts** to promote thinking and engagement in the discussion forums
* **Qwiklabs** to introduce real-world, on-the-job situations, and the tools and tasks to complete assignments
* **Practice quizzes** to prepare you for graded quizzes
* **Hands-on activities** toreinforce learned skills for the graded quizzes
* **Graded quizzes** to measure your progress and give you valuable feedback

Hands-on activities promote additional opportunity to build your skills, so try to get as much out of them as possible. Assessments are based on the approach taken by the course to offer a wide variety of learning materials and activities that reinforce important skills. Graded and ungraded quizzes will help the content sink in and reinforce important skills. Ungraded practice quizzes are a chance for you to prepare for the graded quizzes, and both the graded and ungraded quizzes can be taken more than one time.

As a quick reminder, this course is designed for all types of learners, so no degree or prior experience is required. Everyone learns differently, and the Google Data Analytics Certificate has been designed with that in mind. Personalized deadlines are just a guide, so feel free to work at your own pace. There is no penalty for late assignments. If you prefer, you can extend your deadlines by returning to **Overview** in the navigation pane and clicking **Switch Sessions**. If you already missed previous deadlines, click **Reset my deadlines** instead.

If you would like to review previous content or get a sneak peek of upcoming content, you can use the navigation links at the top of this page to go to another course in the program. When you pass all the required assignments, you will be on track to earn your certificate.

**Tips**

* Try to complete all the activities in order, since new information always builds on previous lessons.
* Treat every task as if it is real-world experience. Have a mindset that you are working at a company or in an organization as a data analyst. This will help you apply what you learn in this program to the real world.
* Repeat demonstrated tasks on your own for extra practice and speed. For example, after you follow along with a video once or twice to perform the demonstrated tasks, try performing the same tasks without playing the video and receiving help from the instructor’s prompts.
* Even though they aren’t graded, be sure to participate in and complete all of the practice activities. They will help you build a strong foundation as a data analyst and prepare you for the graded assessments.
* Take advantage of all the additional resources provided, including discussion forums and links to external articles for more information.
* When you encounter useful links in the course, remember to bookmark them so you can refer to the information for study or review.
* Additional resources are free, but some sites place limits on how many articles you can access for free each month. Sometimes you can register on the site for full access, but you can always bookmark a resource and come back to view it later.
* Maximize the value of hands-on activities. Hands-on activities supplement the demonstrated tasks by encouraging additional practice with similar scenarios. A programming language’s syntax will become more natural to you the more you practice using it.
* Create a notebook or document to keep track of things to remember about the R syntax. This will become a handy and personalized reference that you can use throughout the rest of the program and anytime later.

**The R-versus-Python debate**

People often wonder which programming language they should learn first. You might be wondering about this, too. This certificate teaches the open-source programming language, R. R is a great starting point for foundational data analysis, and it has helpful packages that beginners can apply to projects. Python isn’t covered in the curriculum, but we encourage you to explore Python after completing the certificate. If you are curious about other programming languages, make every effort to continue learning.

Any language a beginner starts to learn will have some advantages and challenges. Let’s put this into context by looking at R and Python. The following table is a high-level overview based on a sampling of articles and opinions of those in the field. You can review the information without necessarily picking a side in the R vs. Python debate. In fact, if you check out RStudio’s blog article in the Additional resources section, it’s actually more about working together than winning a debate.

| **Languages** | **R** | **Python** |
| --- | --- | --- |
| **Common features** | - Open-source - Data stored in data frames - Formulas and functions readily available - Community for code development and support | - Open-source - Data stored in data frames - Formulas and functions readily available - Community for code development and support |
| **Unique advantages** | - Data manipulation, data visualization, and statistics packages - "Scalpel" approach to data: *find packages to do what you want with the data* | - Easy syntax for machine learning needs - Integrates with cloud platforms like Google  Cloud, Amazon Web Services, and Azure |
| **Unique challenges** | - Inconsistent naming conventions make it harder for beginners to select the right functions - Methods for handling variables may be a little complex for beginners to understand | - Many more decisions for beginners to make about data input/output, structure, variables, packages, and objects - "Swiss army knife" approach to data: *figure out*  *a way to do what you want with the data* |

**Additional resources**

For more information on comparing R and Python, refer to these resources:

* [R versus Python, a comprehensive guide for data professionals](https://medium.com/analytics-and-data/r-vs-python-a-comprehensive-guide-for-data-professionals-321e8dead598): This article is written by a data professional with extensive experience using both languages and provides a detailed comparison.
* [R versus Python, an objective comparison](https://www.dataquest.io/blog/python-vs-r/): This article provides a comparison of the languages using examples of code use.
* [R versus Python: What’s the best language for data science?](https://blog.rstudio.com/2019/12/17/r-vs-python-what-s-the-best-for-language-for-data-science/): This blog article provides RStudio’s perspective on the R vs. Python debate.

**Key takeaways**

Certain aspects make some programming languages easier to learn than others. But, that doesn’t make the harder languages impossible for beginners to learn. On the flip side, a programming language’s popularity doesn’t always make it the best language for beginners either.

R has been used by professionals who have a statistical or research-oriented approach to solving problems; among them are scientists, statisticians, and engineers. Python has been used by professionals looking for solutions in the data itself, those who must heavily mine data for answers; among them are data scientists, machine learning specialists, and software developers.

As you grow as a data analytics professional, you may need to learn additional programming languages. The skills and competencies you learn from your first programming experience are a good foundation. That's why this course focuses on the basics of R. You can develop the right perspective, that programming languages play an important part in the data analysis process no matter what job title you have.

The good news is that many of the concepts and coding principles that you will learn from using R in this course are transferable to other programming languages. You will also learn how to write R code in an Integrated Development Environment (IDE) called RStudio. RStudio allows you to manage projects that use R or Python, or even a combination of the two. Refer to [RStudio: A Single Home for R & Python](https://www.rstudio.com/solutions/r-and-python/) for more information. So, after you have worked with R and RStudio, learning Python or another programming language in the future will be more intuitive.

For a better idea of popular programming languages by job role, refer to [Ways to learn about programming](https://www.coursera.org/learn/data-analysis-r/supplement/y8zTf/ways-to-learn-about-programming). The programming languages most commonly used by data analysts, web designers, mobile and web application developers, and game developers are listed, along with links to resources to help you start learning more about those languages.

# Learning Log: Get ready to explore R



## Overview



So far, you’ve learned about data management, analysis, and visualization. In this upcoming course, you will learn the fundamentals of R and how it can be used to perform the same data analysis steps you already know. Now, you’ll complete an entry in your learning log considering R as a tool for data analysis. By the time you complete this entry, you will be ready to start learning this new and exciting tool!

## R you ready?



Before you start writing your learning log entry in the template linked below, let’s discuss what exactly R is. **R** is a programming language used for statistical analysis, visualization, and other data analysis. As a data analyst, you will use R to complete many of the tasks associated with the data analysis process. Understanding how it works and why you use it is crucial to developing a mastery of data analytics.

Like the other tools you have already learned in this program, R will be an important part of your data analysis toolkit. You don’t need any previous experience with R for this course; you’ll get a chance to learn the basics and practice writing R code yourself. Then, you can even try using R for your capstone project later!

In the learning log template below, you’ll get a chance to reflect on learning R and what you’re most excited about for this course.



### Access your learning log

To use the template for this course item, click the link below and select “Use Template.”

Link to learning log template: [Get ready to explore R](https://docs.google.com/document/d/1r4L9UMxdsxdc24SrFws7dNNmM_7XDDbQ8m0wpwuGIt4/template/preview)

OR

If you don’t have a Google account, you can download the template directly from the attachment below.

[Learning Log Template\_ Get ready to explore R](https://d3c33hcgiwev3.cloudfront.net/tGrlU5-XTx6q5VOfly8eYw_fceedbab923c44b7889850a8e25412f1_Learning-Log-Template_-Get-ready-to-explore-R.docx?Expires=1706918400&Signature=UJNaXnPdXEQHu~v5NwQbYxUSpmL3mo54FU2~tnXlyqIlNtM5I4nZPW9quIWwKrF7BEcczRcHNknXyCh0gc0DZNqfmfwezDvar1utVxzvrR8qWQPwVuc7~0c4I8XQfbsmzt3b9mJi08NvGkRMOQO14wrD~dTi2Dnjq58sIKO5Lqc_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[DOCX File](https://d3c33hcgiwev3.cloudfront.net/tGrlU5-XTx6q5VOfly8eYw_fceedbab923c44b7889850a8e25412f1_Learning-Log-Template_-Get-ready-to-explore-R.docx?Expires=1706918400&Signature=UJNaXnPdXEQHu~v5NwQbYxUSpmL3mo54FU2~tnXlyqIlNtM5I4nZPW9quIWwKrF7BEcczRcHNknXyCh0gc0DZNqfmfwezDvar1utVxzvrR8qWQPwVuc7~0c4I8XQfbsmzt3b9mJi08NvGkRMOQO14wrD~dTi2Dnjq58sIKO5Lqc_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)



## Reflection



Take a moment to reflect on learning about R. In your learning log template, write 2-3 sentences (40-60 words) in response to each question below:

* Which parts of R are you excited to learn about?
* Which parts seem most challenging?

When you’ve finished your entry in the learning log template, make sure to save the document so your response is somewhere accessible. This will help you continue applying data analysis to your everyday life. You will also be able to track your progress and growth as a data analyst.

Hi, great to see you again. When I first learned R, it was the visuals

that really got me hooked.

I still think it's so cool that you can write a little bit of code,

press a button and, presto, out pops an awesome data visualization.

Before we get into all the details, I thought it would be fun to give you

a quick sneak peek and show you what R can do.

What follows will be a preview of what you're going to learn.

By the end of this course, you'll not only understand all this code,

you'll be able to write and execute it as well.

For now, just sit back, relax and enjoy the show.

Let's start by loading a library and getting a dataset to work with.

We can use the palmer penguins dataset, which contains size measurements for

three penguin species that live on the Palmer Archipelago in Antarctica.

This includes data on stuff like body mass, flipper length and bill length.

The dataset has 344 rows of information sorted into eight columns.

The palmer penguins data is popular with analysts and is great for

fun exploration, visualization and teaching concepts.

We'll see more of this data set later on in the course.

Let's say we want to visualize the relationship between body mass and

flipper length.

You may guess the larger the penguin, the longer the flipper.

We can find out for sure by creating a plot. Let's make a scatter plot.

A scatter plot uses points to display the relationship between two variables.

So the two variables were going to compare our body mass and flipper length.

No need to memorize all these details right now.

You'll have time to learn more about them later on.

Play video starting at :1:54 and follow transcript1:54

Let's check out the parts of this code and how they fit together.

The first function starts the plot. If we run the code at this point,

all we get is a blank plot.

Play video starting at :2:9 and follow transcript2:09

If we add some more code,

R will put labels on each access of our plot and add lines for data.

Body mass is on the y-axis and flipper length is on the x-axis, but

the data points are not yet visible.

Play video starting at :2:26 and follow transcript2:26

To get the complete plot,

we can add some more code that tells R how to represent our data.

For example, we could use points, bars or lines.

We'll use points to create a scatter plot.

Play video starting at :2:42 and follow transcript2:42

We can go further.

For example, we can change how the plot looks.

Let's change the color of all of the points to purple.

You can hit the Up arrow to pull up the last piece of code you ran, so

we'll do that now.

And then we'll add in color equals purple inside geom point.

Now we can hit Enter to run this.

Play video starting at :3:16 and follow transcript3:16

We can also add new information to the plot and use color to highlight it.

Let's tell R to assign a different color to each species of penguin.

This way we can link data points to each group of penguins.

Play video starting at :3:34 and follow transcript3:34

Gentoos are the largest.

The legend just to the right of the plot shows

us that the blue points refer to the Gentoos.

R automatically creates a legend for

the plot to help us understand the color-coding.

R does everything you tell it to do and

even does stuff you don't ask for. It's just that helpful.

We can also use shape to highlight the different penguin species.

Play video starting at :4:7 and follow transcript4:07

Or we can use both color and shape.

Play video starting at :4:16 and follow transcript4:16

In addition to highlighting our data, we can also reorganize it.

We can break our data down into smaller groups or subsets and create a plot for

each subset.

Let's say we want to focus on the data for each species.

Facet functions let us create a separate plot for each species.

Check this out.

Facets are so great. We can even put text on our plot to point to

specific data or communicate a message.

Let's give our plot a title to clearly indicate its purpose.

Play video starting at :4:56 and follow transcript4:56

Finally, we can save our plot, so we can access or share it later on.

Play video starting at :5:6 and follow transcript5:06

Now, if we click on the Files tab, we'll find our file in the list.

Play video starting at :5:13 and follow transcript5:13

Let's open it up.

Play video starting at :5:17 and follow transcript5:17

Well, that's the end of the show.

I hope you enjoyed it as much as I did.

We were able to take a big dataset and

quickly visualize some significant patterns.

These are some of the basic functions in R. In other words,

this is just the beginning.

It's exciting to think of all the ways R can help you realize

the full power of data analysis.

As you move forward,

you learn more about each of the functions we use to create our plots.

By the end of this course, you'll be the one writing and

executing all of this code.

Coming up, we'll learn more about computer programming and

how it can help you analyze your data.

See you soon.

The advice I would give to someone just learning R is

that mistakes are part of the process.

Errors and error messages are part of the process.

When I think about the people who

are even better than I am in R,

I've come to realize they're not

necessarily smarter than I am,

but they may be a little bit more

persistent and delving a little bit deeper.

Certainly compared to when I started,

initially I'd see an error message and think,

"I did it wrong, uh-oh, game over. "

Now it's like, "That's just part of the game."

When I started to get a little bit of

exposure to what R looked like,

I was like, "That seems too sophisticated.

It seems like that probably is really hard."

But the people who used it that I had met,

we're always really enthusiastic

about it and they felt like it

had so many advantages over

other software that you can use for running analyses.

There were a lot of times before I

used R where I might use spreadsheets

or some other tool and I would be

trying to hack at what really needed to happen.

Sometimes I was using multiple tools because

an individual tool couldn't really

do all what I wanted it to do.

But it's like I knew in

my mind and yet it wasn't totally fluid,

the execution of it.

The more exposure I've gotten to R,

the more I realize a lot

of what I would try to do that way,

I can just do within one program,

and it can all interlock really fluidly.

At first, I was really unconfident.

I had a couple of

scripts where I had some friends who were better at R,

people I worked with who would sit

down and help me go through and

understand the code and so it felt

really silly to ask them the simple question of like,

"Okay, but why is a bracket here?"

Or "Why would we do this?"

But they were fortunately really patient people.

Then at some point,

our entire department said,

really everybody needs to be using this

because we need everyone on the same platform.

We need consistency in our analyses.

We need to be able to code review

each other's analyses as well.

We all took an online course

together and that helped me feel really

a lot more confident because it was walking

through each step of what you needed to know,

got an opportunity to practice,

and then it felt like, "Okay,

even if there's things I don't know,

"I've made it through introduction,

like I've made it through this next module

so I do know something."

Then once I started to apply it in my work,

there would still be points where I was like,

"Wait, I don't know how to solve this problem."

Then I would talk to a friend,

Google something and generally,

I knew a lot more than I thought

that I did and from that,

I suddenly unlocked my ability

to produce a whole lot of analyses

quickly with the big dataset and also produce

a whole lot of data visualizations

really quickly using ggplot2.

Hi, my name's Carrie and I'm

a Research Manager within People Operations at Google.

Hi. Great to have you back.

Earlier, you learned that programming means giving

instructions to a computer to

perform an action or set of actions.

Even if this is your first time programming,

you already have plenty of

experience telling a computer what to do.

For example, you've probably used

a spreadsheet function to sort

your data or perform calculations,

or you might have used SQL to tell a computer how to pull

data from a database or join two different data tables.

Programming goes even further.

It gives you the highest level of control over your data.

SQL can communicate with databases,

but a general-purpose

programming language lets you create

your own applications and

build your own functions from scratch.

To program, you first

need to know a programming language.

In this video, we'll learn about the basics of

programming languages and how

they can help you work with your data.

Programming languages are the words and symbols we

use to write instructions for computers to follow.

You can think of a programming language as

a bridge that connects humans and computers,

and allows them to communicate.

Programming languages have their own set of rules for how

these words and symbols should be used, called syntax.

Syntax shows you how to arrange the words and

symbols you enter so they make sense to a computer.

Coding is writing instructions to

the computer in the syntax

of a specific programming language.

Just like the variety of

human languages around the world,

there's lots of different programming languages

available to communicate with computers.

There's a language for almost anything you want to do,

from designing websites, to developing video games,

to working with data.

For example, Python is

a general-purpose language that

can be used for all sorts of things,

from working with artificial intelligence

to creating virtual reality experiences.

Javascript works well for developing

online apps and is an essential part of web browsers.

Some other popular programming languages for

data analysis include SAS, Scala, and Julia.

Personally speaking, R is

my favorite language for data analysis,

but you might want to explore other languages as well.

While programming languages can

look different on the surface,

they all share similar structures and coding concepts.

Once you learn your first language,

you'll find it easier to learn others.

Coming up, we'll explore R's many capabilities.

Before that, let's talk about some benefits of

using any programming language to work with your data.

I'll highlight three.

Programming helps you clarify the steps of your analysis,

saves time, and lets you

easily reproduce and share your work.

Let's start with clarity.

Programming languages have specific rules and

guidelines for giving instructions to the computer.

When you're telling a computer what to do,

your instructions have to be very clear.

There can't be any inconsistency

in the way you write code.

If there is, the code won't work.

Translating your thoughts into code

forces you to figure out exactly how to write each step

of your analysis and how all the steps fit together.

It gives your analysis a level of

precision that makes it really powerful.

Using a programming language for

data analysis also saves you lots of time.

For example, take the process of

cleaning and transforming your data.

With one line of code,

you can create a separate dataset

without any missing values.

With another line, you can

apply multiple filters on your data.

This lets you spend less time preparing

your data and more time on the analysis itself.

Finally, programming languages make

it easy to reproduce your analysis.

Data analysis is most useful when you

can reproduce your work and share it with other people.

They can double-check it and help you solve problems.

Code automatically stores all of the steps of

your analysis so you can reproduce,

and share your work at anytime in the future,

weeks, months, or even years later.

Here's an example. Let's say you're working on a project.

You've collected and cleaned

your data and started your analysis,

but the results don't add up.

You suspect a mistake was made in the process.

You'd like to discuss the issue with

a teammate and get their feedback.

If you used a spreadsheet,

you both might have to redo

the entire analysis to discover the error.

There's no easy way to

record and reproduce your steps in a spreadsheet,

but if you use a programming language,

all your work can be reproduced and shared in a moment,

from loading the data,

to creating visualizations, to reporting the results.

Plus, you can easily update your analysis and

fix any errors simply by changing the code.

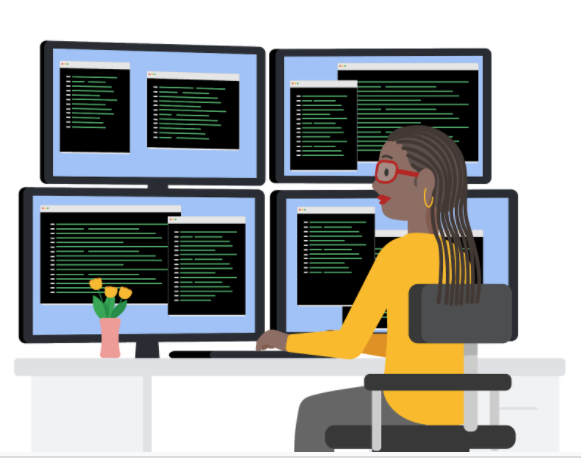
I hope that gives you a better understanding

of what programming languages are all about.

Next up, we'll check out R in more detail. See you soon.

# Ways to learn about programming

Writing programming language code can be an exciting and rewarding experience. The programming field has a long history of people helping each other improve their skills and develop best practices. You will focus on the R programming language in this course, but in the future you might choose to pursue additional programming languages based on your interests and professional goals. This reading is a general guide to help you decide which programming languages are best suited for you.



## Popular programming languages by profession

Let’s go through some potential job titles you might encounter and the most popular programming languages used in those professions. Also included is a list of additional resources for you to explore and learn more about each of the programming languages introduced.

### **Data analyst**

A data analyst collects, transforms, and organizes data to draw conclusions, make predictions, and drive informed decision-making. The most popular programming languages used by data analysts are R and Python.

**R** offers convenient statistical features for data analysis and is useful for creating advanced data visualizations. Check out these resources to learn more about R:

* [The R Project for Statistical Computing](https://www.r-project.org/): a website for downloading R, documentation, and help
* [R Manuals](https://cran.r-project.org/manuals.html): links to manuals from the R core team, including introduction, administration, and help
* [Coding Club R Tutorials](https://ourcodingclub.github.io/tutorials.html): a collection of coding tutorials for R
* [R for Beginners](https://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf): a starting guide to help you work with data, graphics, and statistics in R

**Python** is a general-purpose language that you can use to create what you need for data analysis. Here are a few resources to begin learning Python:

* [The Python Software Foundation (PSF)](https://www.python.org/about/gettingstarted/): a website with guides to help you get started as a beginner
* [Python Tutorial](https://docs.python.org/3/tutorial/): a Python 3 tutorial from the PSF site
* [Coding Club Python Tutorials](https://ourcodingclub.github.io/tutorials.html): a collection of coding tutorials for Python

**Kaggle** is an online repository of various datasets that can be used in both R and Python. It's a robust platform that regularly hosts solution-based competitions using data sets in high-interest industries. Learners may also explore a vast trove of data modeling discussions, trending plug-in models, and useful code snippets. Here are some great resources to get started in Kaggle:

* [Datasets](https://www.kaggle.com/datasets): explore and download a vast collection of data sets while up-voting your favorite collection.
* [Competitions](https://www.kaggle.com/competitions): commit individually or collaborate in a team towards data competitions for the possibility of financial rewards. Even without winning the competitions, this is a great way to network with other analysts.
* [Learn](https://www.kaggle.com/learn): use this resource for an additional perspective on data visualization, linear regression techniques, or time series charting code.

### **Web designer**

A web designer is responsible for the styling and layout of web pages containing text, graphics, and video. Web designers generally use Hypertext Markup Language v5 (HTML5) and Cascading Style Sheets (CSS) to create web pages.

**HTML5** provides structure for web pages and is used to connect to hosting platforms. Learn more about HTML5 and CSS using these resources:

* [HTML Tutorial](https://www.tutorialrepublic.com/html-tutorial/): an introduction to HTML with links to HTML5 features, examples, and references
* [HTML5 Cheat Sheet](https://www.wpkube.com/html5-cheat-sheet/): a handy summary of HTML5 tags, attributes, and compatibility with HTML4
* [HTML5 and CSS Fundamentals course](https://www.edx.org/course/html5-and-css-fundamentals): a free W3C course on edX; a verified course certificate can be issued for $199

**CSS** is used for web page design and controls graphic elements (color, layout, and font) and page presentation on multiple devices (large screens, mobile screens, and printers). Check out these cheat sheets for CSS:

* [Interactive CSS Cheat Sheet](https://htmlcheatsheet.com/css/): includes the most common CSS snippets for gradient, background, font-family, border, and much more
* [50 Best HTML & CSS Cheat Sheets](https://sharethis.com/best-practices/2020/02/best-html-and-css-cheat-sheets/): a list of 50 cheat sheets–choose a few that are useful to you

### **Mobile application developer**

A mobile application developer uses programming to create applications used on laptops, mobile phones, and tablets. The most popular programming languages for mobile application developers are Swift, Java, and C#.

**Swift** (for Apple platforms) is an open source scripting language for macOS, iOS, watchOS, and tvOS. Its main goal is to make applications run faster. Browse these resources for more information about Swift:

* [Swift.org](https://swift.org/about/): an open source community with resources to learn how to use Swift, including videos and sample code
* [Swift developer site](https://developer.apple.com/swift/): an Apple developer website with information for developers who want to use Swift
* [Swift development resources](https://developer.apple.com/swift/resources/): Apple’s collection of documentation, sample code, videos, and recommended books

**Java** (for Android devices) is the official language for Android development. The article [I want to develop Android apps - which languages should I learn?](https://www.androidauthority.com/develop-android-apps-languages-learn-391008/) explores some other languages used for Android development. Check out these resources for Java:

* [Android Studio](https://developer.android.com/studio): a downloadable integrated development environment (IDE) with tools to build apps for Android devices
* [Build your first Android app in Java](https://developer.android.com/codelabs/build-your-first-android-app#1): instructions for installing Android Studio and creating your first app
* [Java tutorial for beginners: write a simple app with no previous experience](https://www.androidauthority.com/java-tutorial-for-beginners-write-a-simple-app-with-no-previous-experience-1121975/): an overview of how to learn Java, with examples

**C#** (pronounced C-sharp) is an object-oriented programming language that is widely used to create mobile apps in the .NET open source developer platform. Xamarin extends the .NET platform with a framework for developers to create cross-platform mobile apps for both iOS and Android. Here are a few resources to help you learn C#:

* [Microsoft .NET learning materials for C#](https://dotnet.microsoft.com/learn/csharp): includes free courses, tutorials, and videos to learn the programming language C#
* [Microsoft Xamarin learning materials](https://dotnet.microsoft.com/learn/xamarin): includes free courses, tutorials, and videos to learn about mobile development with Xamarin
* [Xamarin Tutorial - build your first iOS or Android app in C#](https://dotnet.microsoft.com/learn/xamarin/hello-world-tutorial/intro): instructions for building a mobile app that displays the text “Hello World”
* [Learn C# from Codecademy](https://www.codecademy.com/learn/learn-c-sharp): a website with free basic interactive lessons, and additional activities that can be accessed with a monthly subscription

### **Web application developer**

A web application developer designs and develops network applications used across the web. The most popular programming languages used by web application developers are Java, Python, Ruby, and PHP.

**Java** is widely used to create enterprise web applications that can run on multiple clients. Java’s main strength is its “Write Once, Run Anywhere” (WORA) approach.Browse these resources to learn more about Java:

* [Oracle Java Tutorials](https://docs.oracle.com/javase/tutorial/): Java tutorials from Oracle documentation
* [Java for Beginners](https://www.homeandlearn.co.uk/java/java.html): a free Java course for beginners from the website “Home and Learn”

**Python** is a general-purpose programming language. Check out the Python resources listed in the data analyst section.

**Ruby** is a general-purpose, object-oriented programming language used for web application development. Ruby isn't the same as Ruby on Rails, which is an open source web application framework that runs using Ruby. Browse these resources to learn more about Ruby:

* [Ruby news](http://ruby-doc.org/): information about the latest Ruby releases and links to other resources
* [Ruby documentation](http://www.ruby-lang.org/en/documentation/): includes guides, tutorials, and reference material to help you learn more about Ruby
* [Ruby programmer’s guide](http://ruby-doc.com/docs/ProgrammingRuby/): a tutorial and reference guide for Ruby
* [Learn Ruby from Codecademy](https://www.codecademy.com/learn/learn-ruby): a website with free basic interactive lessons, and additional activities that can be accessed with a monthly subscription

**PHP** is a scripting language particularly suited for web application development. It was based on Perl, another programming language. PHP is simple, flexible, and relatively easy to learn. Check out these resources to learn more about PHP:

* [PHP downloads and documentation](https://www.php.net/): information about the latest PHP releases and links to other resources
* [PHP the Right Way](https://phptherightway.com/): a quick reference for popular PHP coding standards
* [Interactive PHP tutorial](https://www.learn-php.org/): a free tutorial that runs PHP code in exercises

### **Game developer**

A game developer is an application developer who specializes in video game creation. Game developers most commonly use the programming languages C# and C++.

**C#** is an object-oriented programming language that is widely used to create games. Check out the C# resources listed in the mobile application developer section.

**C++** is an extension of the C programming language that is also used to create console games, like those for Xbox. Browse more information about C++:

* [Microsoft resources for C++](https://docs.microsoft.com/en-us/cpp/?view=msvc-160): learn how to install the Visual Studio IDE and write C++ code
* [Microsoft C++ and C# code samples for gaming](https://docs.microsoft.com/en-us/samples/browse/?languages=cpp&terms=gaming): a resource with over 40 C++ and C# code samples for gaming
* [Interactive C++ tutorial](https://www.learn-cpp.org/): a free tutorial that runs C++ code in exercises

## Tips for learning programming languages

Here are a few tips to follow when you start learning a new programming language:

* Define a practice project and use the language to help you complete it. This makes the learning process more practical and engaging.
* Keep previous concepts and coding principles in mind. Many of these are transferable between programming languages. So, after you have learned one language, learning a second or third programming language tends to be much easier.
* Create and keep good notes and cheat sheets in whatever format (handwritten or typed) that works best for you.
* Create an online filing system for information that you can easily access while you work in various programming environments.

**From spreadsheets to SQL to R**

Although the programming language R might be new to you, it actually has a lot of similarities to the other tools you have explored in this program. In this reading, you will compare spreadsheet programs, SQL, and R to have a better sense of how to use each moving forward.



**Spreadsheets, SQL, and R: a comparison**

As a data analyst, there is a good chance you will work with SQL, R, and spreadsheets at some point in your career. Each tool has its own strengths and weaknesses, but they all make the data analysis process smoother and more efficient. There are two main things that all three have in common:

* **They all use filters:** for example, you can easily filter a dataset using any of these tools. In R, you can use the filter function. This performs the same task as a basic SELECT-FROM-WHERE SQL query. In a spreadsheet, you can create a filter using the menu options.
* **They all use functions:** In spreadsheets, you use functions in formulas, and in SQL, you include them in queries. In R, you will use functions in the code that is part of your analysis.

The table below presents key questions to explore a few more ways that these tools compare to each other. You can use this as a general guide as you begin to navigate R.

| **Key question** | **Spreadsheets** | **SQL** | **R** |
| --- | --- | --- | --- |
| **What is it?** | A program that uses rows and columns to organize data and allows for analysis and manipulation through formulas, functions, and built-in features | A database programming language used to communicate with databases to conduct an analysis of data | A general purpose programming language used for statistical analysis, visualization, and other data analysis |
| **What is a primary advantage?** | Includes a variety of visualization tools and features | Allows users to manipulate and reorganize data as needed to aid analysis | Provides an accessible language to organize, modify, and clean data frames, and create insightful data visualizations |
| **Which datasets does it work best with?** | Smaller datasets | Larger datasets | Larger datasets |
| **What is the source of the data?** | Entered manually or imported from an external source | Accessed from an external database | Loaded with R when installed, imported from your computer, or loaded from external sources |
| **Where is the data from my analysis usually stored?** | In a spreadsheet file on your computer | Inside tables in the accessed database | In an R file on your computer |
| **Do I use formulas and functions?** | Yes | Yes | Yes |
| **Can I create visualizations?** | Yes | Yes, by using an additional tool like a database management system (DBMS) or a business intelligence (BI) tool | Yes |

Hello again.

Now that we've talked about programming languages in general, let's get to know R.

So what is R?

R is a programming language frequently used for

statistical analysis, visualization and other data analysis.

Later on, you'll take a tour of Rstudio, which is a popular software

environment for the R language. In this video, we'll discuss R's

main features and functions and

its advantages for data analysis. R

is super cool. I'm excited for you to learn about it.

R is based on another programming language

named S. In the 1970s, John Chambers created S for

internal use at Bell Labs, a famous scientific research facility.

In the 1990s, Ross Oaxaca and

Robert Gentleman developed R at the University of Auckland,

New Zealand. The title R refers to the first names of

its two authors and plays on a single- letter title of its predecessor S.

Since then,

R has become a preferred programming language of scientists, statisticians and

data analysts around the world. There's lots of reasons why

people who work with data love R. I want to share four with you.

R is accessible, data-centric, open-source

and has an active community of users. First R is an accessible language for

beginners. Lots of people without a traditional

programming language learn R. I should know. I'm one of them.

R really appeals to anyone who wants

to solve problems that involve data. And that's one of the things that's so

great about R. It's all about data.

R is what's known as a data-centric

programming language. It's specifically designed to make

data analysis easier, more efficient and more powerful.

Another awesome thing about R is that it's

open source. Open source means that the code is

freely available and may be modified and shared by the people who use it.

Let's pause for a moment and

unpack how amazing this is. First anyone can use R for free.

Second, anyone can modify the code,

fix bugs and improve it. In fact, over the years,

lots of excellent programmers have made improvements and fixes to the R code.

For example,

anyone who knows the R language can create what's called an add-on package.

We'll talk more about R packages later.

For now, just know that literally

thousands of R packages exist, and they were all built by people who

wanted to solve specific problems. A lot of these packages are super

useful for data analysts. As an R user, you now enjoy the benefit

of the shared knowledge. And let me just add,

the R community is the best. This vibrant, diverse and

accessible community is so supportive of new learners.

You can go online anytime to find

answers to all your R questions. Check out websites like R for

Data Science Online Learning Community and RStudio Community.

On top of that, R users are all over

Twitter and other social media. You'll discover tons of resources for

professional networking, mentoring and learning.

Now that we know more about the general

benefits of R, let's talk about some specific situations when you might use it

for data analysis. Here's three scenarios:

reproducing your analysis, processing lots

of data, and creating data visualizations. First R can save and

reproduce every step of your analysis. Earlier, we discussed

how data analysis is most useful when you can easily reproduce your work and

share it with others. In R, reproducing your analysis is as easy

as pressing a button on your keyboard. Your code stores it

forever. And

you can share it with anyone at any time. Processing lots of data is

also something R does really well, just like SQL.

As you learned earlier spreadsheets

organize projects in sheets or tabs. If you've ever had to deal

with spreadsheet files that have tons of sheets or

lots of data in each sheet, you know that things can

start to move very slowly. Working with too much data in

a spreadsheet can even cause crashes. R can handle large amounts of

data much more quickly and efficiently. Finally R can create powerful visuals and

has state-of-the-art graphic capabilities. As you've seen in this program,

tools like spreadsheets and Tableau offer lots of options for

visualizing your data. R's on another level.

With only a small bit of code, you can create histograms, scatter plots,

line plots and so much more. And that's just the beginning.

If you work with more advanced packages,

you can make some seriously impressive data visualizations.

Learning R is a huge benefit to anyone

interested in becoming a data analyst. As I mentioned earlier,

knowledge of R will help you stand out as a job candidate.

And as you keep moving forward,

R will help you find solutions for more complex data problems.

You can keep learning about R

throughout your career as a data analyst. The sky's the limit when it comes to

developing your data analysis skills. That's all for now.

Coming up, we'll check out the RStudio

environment together. Before you use RStudio,

you need to download and install the basic R interface.

You'll learn how to do that

in an upcoming reading. Most analysts who work with

the R language use the RStudio environment to interact with R,

and not the basic interface.

That's why we're focusing on RStudio

in this program. Following this video,

you'll find resources for downloading R and RStudio

if you're interested in learning more. Bye for now.

### 1.

Question 1

Fill in the blank: Programming involves \_\_\_\_\_ a computer to perform an action or set of actions.

1 point

training

instructing

filtering

updating

### 2.

Question 2

What are the benefits of using a programming language to work with your data? Select all that apply.

1 point

Save time

Clarify the steps of your analysis

Choose a business task for analysis

Easily reproduce and share your work

### 3.

Question 3

The R programming language can be used for which of the following tasks? Select all that apply.

1 point

Statistical analysis

Gaming

Visualization

Data analysis

Hey there. It's time to take our tour of RStudio.

The examples we'll look at are from RStudio Cloud, but

RStudio works in a similar way across all platforms.

Feel free to use the platform that works best for you.

Later on if you want to learn more, you'll find resources on how to download and

install RStudio on your own device.

RStudio's an IDE or integrated development environment.

This means that RStudio brings together all the tools you might want to

use in a single place.

The R console which we explored earlier is one part of this environment.

RStudio also includes an editor for writing code, and tools for

managing your data and creating visuals.

RStudio is built specifically for use with R.

It'll help maximize your productivity as a data analyst.

Data analysis is like driving a car.

You can think of R and RStudio as different parts of this car.

R is like the car engine.

RStudio is kind of like the accelerator, the steering wheel,

and dashboard all-in-one.

It lets you tell the engine what to do and helps you get to where you want to go.

Just as a speedometer and navigation system make driving much easier,

RStudio's environment makes using R much easier.

In an earlier reading, you learned how to access RStudio.

So let's log into RStudio now and explore.

The RStudio environment has four main windows called panes.

Each pane helps you perform different functions.

The first time you open RStudio, you'll see three panes.

A fourth pane is hidden by default, but it's easy to open.

Just click on File in the menu, then select New File and R Script.

RStudio has lots of keyboard shortcuts.

To learn more check out Keyboard Shortcuts Help.

Play video starting at :2:11 and follow transcript2:11

You can make the panes smaller or larger by clicking on the minimize or

maximize buttons at the upper right of each pane.

Play video starting at :2:20 and follow transcript2:20

You can also click and drag the borders of the panes to adjust their size.

Play video starting at :2:26 and follow transcript2:26

Click on the Panes button for more feature options.

Play video starting at :2:33 and follow transcript2:33

Now that we've got all four panes open, let's explore each of them.

We'll start on the lower left and move clockwise from there.

You might recognize the R console from an earlier reading.

As a quick refresher, the console is the place where you give commands to R.

For example, we can tell R to show us a summary of the penguins data that

we used in an earlier video to create visuals.

You'll need to install and

load the palmer penguins dataset if you haven't done so already.

Play video starting at :3:16 and follow transcript3:16

Above the console in the upper left is the source editor pane.

You'll use the source editor when working with R Scripts.

There are two main ways of writing code in RStudio: using the console or

using the source editor.

You can type commands directly into the console, but

they'll be forgotten when you close your current session.

As we've discussed, it's important to be able to reproduce and

share all the steps of your analysis.

If you save your script in the editor,

you can access your work again at any time and show others how you did it.

Play video starting at :3:53 and follow transcript3:53

The source editor and the console also work together in RStudio.

When you execute code in the editor, the code automatically appears in the console.

If you're working on a long analysis, this makes it easy to execute

the entire code all at once or run specific sections of it as you go along.

Let's run some code in the editor and check it out.

Play video starting at :4:25 and follow transcript4:25

Pro tip: Always keep in mind that R is case-sensitive.

Here we use a capital V for the View function.

Play video starting at :4:36 and follow transcript4:36

Next, let's go to the Environment pane in the upper right.

Here you'll find all the data you currently have loaded and

can easily organize and save it.

For example if you import data from a spreadsheet,

it'll be visible in the Environment pane.

You can view each object in the Environment pane by clicking on it.

You can also toggle between a List view and a Grid view.

Play video starting at :5:7 and follow transcript5:07

To the right of the Environment tab, you'll find the History tab.

All your previous commands are saved here and they're easy to search and re-execute.

You'll find the most recent line of code at the bottom of the list.

You can copy any line to the command console by double-clicking it.

In the lower right, you'll see a pane that has tabs for Files,

Plots, Packages, and Help.

The Files tab gives you access to your file directory and

shows the contents of the current working folder.

You can easily find and manage all your files and create new project folders.

Next is the Plots tab.

If we create a plot, the result appears here.

For example,

we can create a scatter

plot with the penguins

dataset we used earlier.

Play video starting at :6:18 and follow transcript6:18

You'll learn more about creating plots in RStudio later on.

Earlier, we talked about R packages which are custom solutions to

data problems developed by R users.

RStudio gives you access to a library of R packages known as the tidyverse.

You can upgrade, install, and manage your library in the Packages pane.

Packages loaded in your current session have a check mark.

Later on, we'll explore the tidyverse in more detail.

Finally, click on the Help tab.

Here you can find helpful resources for R and RStudio.

There are tons of resources out there to help answer all your questions.

Be sure to take advantage of them.

That's our tour of RStudio.

We're just scratching the surface of what RStudio can do.

Soon you'll get to explore RStudio in more detail.

Speaking as a data professional, I love working in RStudio.

It makes my work so much easier, faster, and better.

Congratulations on finishing another step in your data analyst learning journey.

Coming up, we'll learn some basic programming concepts.

Then we'll start working with R.

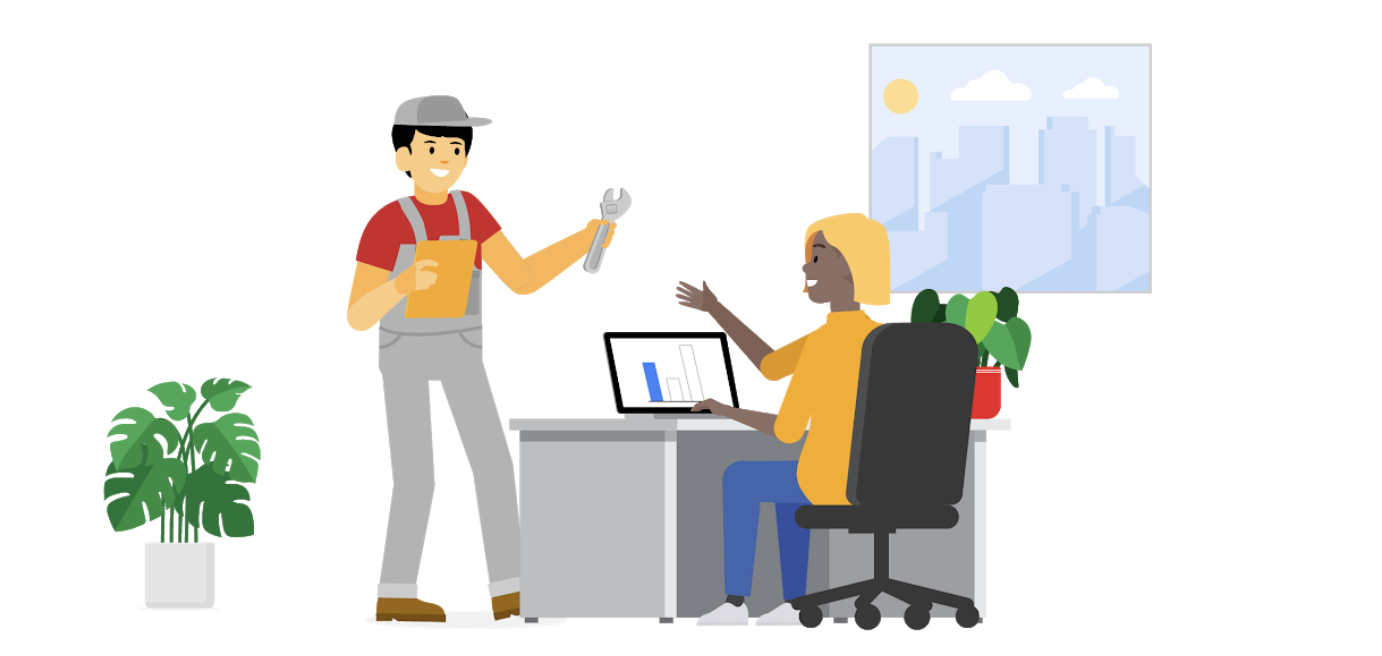
For those of you who are new to programming,

you're about to write your first lines of code.

See you then.

**When to use RStudio**

As a data analyst, you will have plenty of tools to work with in each phase of your analysis. Sometimes, you will be able to meet your objectives by working in a spreadsheet program or using SQL with a database. In this reading, you will go through some examples of when working in R and RStudio might be your better option instead.



**Why RStudio?**

One of your core tasks as an analyst will be converting raw data into insights that are accurate, useful, and interesting. That can be tricky to do when the raw data is complex. R and RStudio are designed to handle large data sets, which spreadsheets might not be able to handle as well. RStudio also makes it easy to reproduce your work on different datasets. When you input your code, it's simple to just load a new dataset and run your scripts again. You can also create more detailed visualizations using RStudio.

**When RStudio truly shines**

When the data is spread across multiple categories or groups, it can be challenging to manage your analysis, visualize trends, and build graphics. And the more groups of data that you need to work with, the harder those tasks become. That’s where RStudio comes in.

For example, imagine you are analyzing sales data for every city across an entire country. That is a lot of data from a lot of different groups–in this case, each city has its own group of data.

Here are a few ways RStudio could help in this situation:

* Using RStudio makes it easy to take a specific analysis step and perform it for each group using basic code. In this example, you could calculate the yearly average sales data for every city.
* RStudio also allows for flexible data visualization. You can visualize differences across the cities effectively using plotting features like facets–which you’ll learn more about later on.
* You can also use RStudio to automatically create an output of summary stats—or even your visualized plots—for each group.

As you learn more about R and RStudio moving forward in this program, you’ll get a better understanding of when RStudio should be your data analysis tool of choice.

**For more information**

* [**The Advantages of RStudio**](https://www.theanalysisfactor.com/the-advantages-of-rstudio/): This web page explains some of the reasons why RStudio is many analysts’ preferred choice for interfacing with R. You’ll learn about the advantages of using RStudio for data analysis, from ease of use to accessibility of graphics and more.
* [**Data analysis and R programming**](https://lgatto.github.io/2017_11_09_Rcourse_Jena/before-we-start.html): This online introduction to data analysis and R programming is a good starting point for R and RStudio users. It also includes a list of detailed explanations about the advantages of using R and RStudio. You’ll also find a helpful guide for getting set up with RStudio.

**Connecting with other analysts in the R community**

R is a powerful tool in your data analysis toolkit–and it also has a powerful community of users who are excited to share, collaborate, and connect with others. This reading will give you a few places where you can start to connect, online and in-person, with other analysts in the R community.



**Online communities**

Online communities allow you to connect with other R users no matter where you live. This list includes forums and discussion channels where you can join the conversation. It also includes social media tags you can use on your existing social media platforms to connect with other data analysts.

* [**RStudio Community:**](https://community.rstudio.com/) The RStudio Community forum is a great place to get help and find solutions to challenges you have with R–and maybe help someone else out, too!
* [**r/RLanguage**](https://www.reddit.com/r/Rlanguage/): The R language subreddit is an active online community on the social media platform Reddit, where R users go to discuss R, ask questions, and share tips.
* [**rOpenSci**](https://discuss.ropensci.org/): rOpenSci has a community forum where R users can ask questions and search for solutions. It also includes links to their Best Practices guide and support pages.
* [**R4DS Online Learning Community and Slack channel:**](https://www.rfordatasci.com/) This is a community with another Slack channel where R learners and mentors can gather and connect. This is a great place to chat about using R for data science.
* [**Twitter #rstats**](https://twitter.com/hashtag/rstats?lang=en): If you use Twitter, you can connect with other R users using the hashtag #rstats; a lot of R developers and analysts are active on Twitter.

**Meetups**

Many organizations host both in-person and online meetups for R users; you should always practice caution and be safe whenever attending meetups in-person.

* [**Local Data Analytics meetups:**](https://www.meetup.com/topics/data-analytics/) These meetups are a great way to meet other people who are interested in data analytics and build your network. These meetups are location-based, so you can connect with other data analysts in your area.
* [**R User Groups:**](https://jumpingrivers.github.io/meetingsR/r-user-groups.html)This list contains links to regional R communities, including subreddits and meetup groups. This is a useful resource if you are interested in finding R users in your area.
* [**RLadies Meetups:**](https://www.meetup.com/pro/rladies) These are in-person and virtual meetups specifically for R enthusiasts who identify as underrepresented or marginalized. These meetups are also location-based and can help you connect with other data analysts in your area.

R can be tricky to learn, but luckily there is a strong community of R users who are interested in working together and helping each other out. These resources are a good starting point if you want to begin connecting with the larger data analyst community, so take advantage of them!

Question 1

What type of software application is RStudio?

1 point

Source editor

Data visualization tool

Database

Integrated development environment

### 2.

Question 2

RStudio includes which of the following panes? Select all that apply.

1 point

R console pane

Source editor pane

Environment pane

Command pane

### 3.

Question 3

If you write code directly in the R source editor, RStudio can save your code when you close your current session.

1 point

True

False

Glossary

Data Analytics

Terms and Definitions

A

A/B testing: The process of testing two variations of the same web page to determine which

page is more successful at attracting user traffic and generating revenue

Absolute reference: A reference within a function that is locked so that rows and columns

won’t change if the function is copied

Access control: Features such as password protection, user permissions, and encryption that

are used to protect a spreadsheet

Accuracy: The degree to which data conforms to the actual entity being measured or described

Action-oriented question: A question whose answers lead to change

Administrative metadata: Metadata that indicates the technical source of a digital asset

Agenda: A list of scheduled appointments

Aggregation: The process of collecting or gathering many separate pieces into a whole

Algorithm: A process or set of rules followed for a specific task

Aliasing: Temporarily naming a table or column in a query to make it easier to read and write

Alternative text: Text that provides an alternative to non-text content, such as images and

videos

Analytical skills: Qualities and characteristics associated with using facts to solve problems

Analytical thinking: The process of identifying and defining a problem, then solving it by using

data in an organized, step-by-step manner

Annotation: Text that briefly explains data or helps focus the audience on a particular aspect of

the data in a visualization

Area chart: A data visualization that uses individual data points for a changing variable

connected by a continuous line with a filled in area underneath

Array: A collection of values in spreadsheet cells

Attribute: A characteristic or quality of data used to label a column in a table

Audio file: Digitized audio storage usually in an MP3, AAC, or other compressed format

AVERAGE: A spreadsheet function that returns an average of the values from a selected range

AVERAGEIF: A spreadsheet function that returns the average of all cell values from a given

range that meet a specified condition

B

Bad data source: A data source that is not reliable, original, comprehensive, current, and cited

(ROCCC)

Balance: The design principle of creating aesthetic appeal and clarity in a data visualization by

evenly distributing visual elements

Bar graph: A data visualization that uses size to contrast and compare two or more values

Bias: A conscious or subconscious preference in favor of or against a person, group of people,

or thing

Big data: Large, complex datasets typically involving long periods of time, which enable data

analysts to address far-reaching business problems

Boolean data: A data type with only two possible values, usually true or false

Borders: Lines that can be added around two or more cells on a spreadsheet

Box plot: A data visualization that displays the distribution of values along an x-axis

Bubble chart: A data visualization that displays individual data points as bubbles, comparing

numeric values by their relative size

Bullet graph: A data visualization that displays data as a horizontal bar chart moving toward a

desired value

Business metric: A standard of measurement used to solve a business task

Business task: The question or problem data analysis resolves for a business

C

Calculated field: A new field within a pivot table that carries out certain calculations based on

the values of other fields

Calculus: A branch of mathematics that involves the study of rates of change and the changes

between values that are related by a function

CASE: A SQL statement that returns records that meet conditions by including an if/then

statement in a query

CAST: A SQL function that converts data from one datatype to another

Causation: When an action directly leads to an outcome, such as a cause-effect relationship

Cell reference: A cell or a range of cells in a worksheet typically used in formulas and functions

Changelog: A file containing a chronologically ordered list of modifications made to a project

Channel: A visual aspect or variable that represents characteristics of the data in a visualization

Chart: A graphical representation of data from a worksheet

Circle view: A data visualization that shows comparative strength in data

Clean data: Data that is complete, correct, and relevant to the problem being solved

Cloud: A place to keep data online, rather than a computer hard drive

Cluster: A collection of data points on a data visualization with similar values

COALESCE: A SQL function that returns non-null values in a list

Coding: The process of writing instructions to a computer in the syntax of a specific

programming language

Column chart: A data visualization that uses individual data points for a changing variable,

represented as vertical columns

Combo chart: A data visualization that combines more than one visualization type

Compatibility: How well two or more datasets are able to work together

Completeness: The degree to which data contains all desired components or measures

Computer programming: The process of giving instructions to a computer in order to perform

an action or set of actions

CONCAT: A SQL function that adds strings together to create new text strings that can be used

as unique keys

CONCATENATE: A spreadsheet function that joins together two or more text strings

Conditional formatting: A spreadsheet tool that changes how cells appear when values meet

specific conditions

Confidence interval: A range of values that conveys how likely a statistical estimate reflects

the population

Confidence level: The probability that a sample size accurately reflects the greater population

Confirmation bias: The tendency to search for or interpret information in a way that confirms

pre-existing beliefs

Consent: The aspect of data ethics that presumes an individual’s right to know how and why

their personal data will be used before agreeing to provide it

Consistency: The degree to which data is repeatable from different points of entry or collection

Context: The condition in which something exists or happens

Continuous data: Data that is measured and can have almost any numeric value

CONVERT: A SQL function that changes the unit of measurement of a value in data

Cookie: A small file stored on a computer that contains information about its users

Correlation: The measure of the degree to which two variables change in relationship to each

other

COUNT: A spreadsheet function that counts the number of cells within a range that meet a

specified condition

COUNTA: A spreadsheet function that counts the total number of values within a range that

meet specified criteria

COUNTIF: A spreadsheet function that returns the number of cells in a range that match a

specified value

COUNT DISTINCT: A SQL function that only returns the distinct values in a specified range

CREATE TABLE: A SQL clause that adds a temporary table to a database that can be used by

multiple people

Cross-field validation: A process that ensures certain conditions for multiple data fields are

satisfied

CSS (Cascading Style Sheets): A programming language used for web page design that

controls graphic elements and page presentation

CSV (comma-separated values) file: A delimited text file that uses a comma to separate

values

Currency: The aspect of data ethics that presumes individuals should be aware of financial

transactions resulting from the use of their personal data and the scale of those transactions

D

Dashboard: A tool that monitors live, incoming data

Dashboard filter: A tool for showing only the data that meets a specific criteria while hiding the

rest

Data: A collection of facts

Data aggregation: The process of gathering data from multiple sources and combining it into a

single, summarized collection

Data analysis: The collection, transformation, and organization of data in order to draw

conclusions, make predictions, and drive informed decision-making

Data analysis process: The six phases of ask, prepare, process, analyze, share, and act

whose purpose is to gain insights that drive informed decision-making

Data analyst: Someone who collects, transforms, and organizes data in order to draw

conclusions, make predictions, and drive informed decision-making

Data analytics: The science of data

Data anonymization: The process of protecting people's private or sensitive data by eliminating

identifying information

Data bias: When a preference in favor of or against a person, group of people, or thing

systematically skews data analysis results in a certain direction

Data blending: A Tableau method that combines data from multiple data sources

Data composition: The process of combining the individual parts in a visualization and

displaying them together as a whole

Data constraints: The criteria that determine whether a piece of a data is clean and valid

Data design: How information is organized

Data-driven decision-making: Using facts to guide business strategy

Data ecosystem: The various elements that interact with one another in order to produce,

manage, store, organize, analyze, and share data

Data element: A piece of information in a dataset

Data engineer: A professional who transforms data into a useful format for analysis and gives it

a reliable infrastructure

Data ethics: Well-founded standards of right and wrong that dictate how data is collected,

shared, and used

Data governance: A process for ensuring the formal management of a company’s data assets

Data-inspired decision-making: Exploring different data sources to find out what they have in

common

Data integrity: The accuracy, completeness, consistency, and trustworthiness of data

throughout its life cycle

Data interoperability: The ability to integrate data from multiple sources and a key factor

leading to the successful use of open data among companies and governments

Data life cycle: The sequence of stages that data experiences, which include plan, capture,

manage, analyze, archive, and destroy

Data manipulation: The process of changing data to make it more organized and easier to

read

Data mapping: The process of matching fields from one data source to another

Data merging: The process of combining two or more datasets into a single dataset

Data model: A tool for organizing data elements and how they relate to one another

Data privacy: Preserving a data subject’s information any time a data transaction occurs

Data range: Numerical values that fall between predefined maximum and minimum values

Data replication: The process of storing data in multiple locations

Data science: A field of study that uses raw data to create new ways of modeling and

understanding the unknown

Data security: Protecting data from unauthorized access or corruption by adopting safety

measures

Data storytelling: Communicating the meaning of a dataset with visuals and a narrative that

are customized for an audience

Data strategy: The management of the people, processes, and tools used in data analysis

Data transfer: The process of copying data from a storage device to computer memory or from

one computer to another

Data type: An attribute that describes a piece of data based on its values, its programming

language, or the operations it can perform

Data validation: A tool for checking the accuracy and quality of data

Data validation process: The process of checking and rechecking the quality of data so that it

is complete, accurate, secure and consistent

Data visualization: The graphical representation of data

Data warehousing specialist: A professional who develops processes and procedures to

effectively store and organize data

Database: A collection of data stored in a computer system

Dataset: A collection of data that can be manipulated or analyzed as one unit

DATEDIF: A spreadsheet function that calculates the number of days, months, or years

between two dates

Decision tree: A tool that helps analysts make decisions about critical features of a visualization

Delimiter: A character that indicates the beginning or end of a data item

Density map: A data visualization that represents concentrations, with color representing the

number or frequency of data points in a given area on a map

Descriptive metadata: Metadata that describes a piece of data and can be used to identify it at

a later point in time

Design thinking: A process used to solve complex problems in a user-centric way

Digital photo: An electronic or computer-based image usually in BMP or JPG format

Dirty data: Data that is incomplete, incorrect, or irrelevant to the problem to be solved

Discrete data: Data that is counted and has a limited number of values

DISTINCT: A keyword that is added to a SQL SELECT statement to retrieve only non-duplicate

entries

Distribution graph: A data visualization that displays the frequency of various outcomes in a

sample

Diverging color palette: A color theme that displays two ranges of data values using two

different hues, with color intensity representing the magnitude of the values

Donut chart: A data visualization where segments of a ring represent data values adding up to

a whole

DROP TABLE: A SQL clause that removes a temporary table from a database

Duplicate data: Any record that inadvertently shares data with another record

Dynamic visualizations: Data visualizations that are interactive or change over time

E

Emphasis: The design principle of arranging visual elements to focus the audience’s attention

on important information in a data visualization

Engagement: Capturing and holding someone’s interest and attention during a data

presentation

Equation: A calculation that involves addition, subtraction, multiplication, or division (also called

a math expression)

Estimated response rate: The average number of people who typically complete a survey

Ethics: Well-founded standards of right and wrong that prescribe what humans ought to do,

usually in terms of rights, obligations, benefits to society, fairness, or specific virtues

External data: Data that lives, and is generated, outside of an organization

F

Fairness: A quality of data analysis that does not create or reinforce bias

Field: A single piece of information from a row or column of a spreadsheet; in a data table,

typically a column in the table

Field length: A tool for determining how many characters can be keyed into a spreadsheet field

Fill handle: A box in the lower-right-hand corner of a selected spreadsheet cell that can be

dragged through neighboring cells in order to continue an instruction

Filled map: A data visualization that colors areas in a map based on measurements or

dimensions

Filtering: The process of showing only the data that meets a specified criteria while hiding the

rest

Find and replace: A tool that finds a specified search term and replaces it with something else

First-party data: Data collected by an individual or group using their own resources

Float: A number that contains a decimal

Foreign key: A field within a database table that is a primary key in another table (Refer to

primary key)

Formula: A set of instructions used to perform a calculation using the data in a spreadsheet

Framework: The context a presentation needs to create logical connections that tie back to the

business task and metrics

FROM: The section of a query that indicates from which table(s) to extract the data

Function: A preset command that automatically performs a specified process or task using the

data in a spreadsheet

G

Gantt chart: A data visualization that displays the duration of events or activities on a timeline

Gap analysis: A method for examining and evaluating the current state of a process in order to

identify opportunities for improvement in the future

Gauge chart: A data visualization that shows a single result within a progressive range of

values

General Data Protection Regulation of the European Union (GDPR): Policy-making body in

the European Union created to help protect people and their data

Geolocation: The geographical location of a person or device by means of digital information

Good data source: A data source that is reliable, original, comprehensive, current, and cited

(ROCCC)

GROUP BY: A SQL clause that groups rows that have the same values from a table into

summary rows

H

HAVING: A SQL clause that adds a filter to a query instead of the underlying table that can only

be used with aggregate functions

Header: The first row in a spreadsheet that labels the type of data in each column

Headline: Text at the top of a visualization that communicates the data being presented

Heat map: A data visualization that uses color contrast to compare categories in a dataset

Highlight table: A data visualization that uses conditional formatting and color on a table

Histogram: A data visualization that shows how often data values fall into certain ranges

HTML5: A programming language that provides structure for web pages and connects to

hosting platforms

Hypothesis: A theory that one might try to prove or disprove with data

Hypothesis testing: A process to determine if a survey or experiment has meaningful results

I

IDE (Integrated Development Environment): A software application that brings together all

the tools a data analyst may want to use in a single place

Incomplete data: Data that is missing important fields

Inconsistent data: Data that uses different formats to represent the same thing

Incorrect/inaccurate data: Data that is complete but inaccurate

INNER JOIN : A SQL function that returns records with matching values in both tables

Inner query: A SQL subquery that is inside of another SQL statement

Internal data: Data that lives within a company’s own systems

Interpretation bias: The tendency to interpret ambiguous situations in a positive or negative

way

J

Java: A programming language widely used to create enterprise web applications that can run

on multiple clients

JOIN: A SQL function that is used to combine rows from two or more tables based on a related

column

K

L

Label: Text in a visualization that identifies a value or describes a scale

Leading question: A question that steers people toward a certain response

LEFT: A function that returns a set number of characters from the left side of a text string

LEFT JOIN: A SQL function that will return all the records from the left table and only the

matching records from the right table

Legend: A tool that identifies the meaning of various elements in a data visualization

LEN: A function that returns the length of a text string by counting the number of characters it

contains

Length: The number of characters in a text string

LIMIT: A SQL clause that specifies the maximum number of records returned in a query

Line graph: A data visualization that uses one or more lines to display shifts or changes in data

over time

Live data: Data that is automatically updated

Long data: A dataset in which each row is one time point per subject, so each subject has data

in multiple rows

M

Mandatory: A data value that cannot be left blank or empty

Map: A data visualization that organizes data geographically

Margin of error: The maximum amount that sample results are expected to differ from those of

the actual population

Mark: A visual object in a data visualization such as a point, line, or shape

MATCH: A spreadsheet function used to locate the position of a specific lookup value

Math expression: A calculation that involves addition, subtraction, multiplication, or division

(also called an equation)

Math function: A function that is used as part of a mathematical formula

MAX: A function that returns the largest numeric value from a range of cells

MAXIFS: A spreadsheet function that returns the maximum value from a given range that meets

a specified condition

McCandless Method: A method for presenting data visualizations that moves from general to

specific information

Measurable question: A question whose answers can be quantified and assessed

Mental model: A data analyst’s thought process and approach to a problem

Mentor: Someone who shares knowledge, skills, and experience to help another grow both

professionally and personally

Merger: An agreement that unites two organizations into a single new one

Metadata: Data about data

Metadata repository: A database created to store metadata

Metric: A single, quantifiable type of data that is used for measurement

Metric goal: A measurable goal set by a company and evaluated using metrics

MID: A function that returns a segment from the middle of a text string

MIN: A spreadsheet function that returns the smallest numeric value from a range of cells

MINIFS: A spreadsheet function that returns the minimum value from a given range that meets

a specified condition

Modulo: An operator (%) that returns the remainder when one number is divided by another

Movement: The design principle of arranging visual elements to guide the audience’s eyes from

one part of a data visualization to another

N

Naming conventions: Consistent guidelines that describe the content, creation date, and

version of a file in its name

Narrative: (Refer to story)

Networking: Building relationships by meeting people both in person and online

Nominal data: A type of qualitative data that is categorized without a set order

Normalized database: A database in which only related data is stored in each table

Notebook: An interactive, editable programming environment for creating data reports and

showcasing data skills

Null: An indication that a value does not exist in a dataset

O

Observation: The attributes that describe a piece of data contained in a row of a table

Observer bias: The tendency for different people to observe things differently (also called

experimenter bias)

Observer bias: The tendency for different people to observe things differently (also called

experimenter bias)

Open data: Data that is available to the public

Open-source: Code that is freely available and may be modified and shared by the people who

use it

Openness: The aspect of data ethics that promotes the free access, usage, and sharing of data

Operator: A symbol that names the operation or calculation to be performed

ORDER BY: A SQL clause that sorts results returned in a query

Order of operations: Using parentheses to group together spreadsheet values in order to

clarify the order in which operations should be performed

Ordinal data: Qualitative data with a set order or scale

Outdated data: Any data that has been superseded by newer and more accurate information

OUTER JOIN: A SQL function that combines RIGHT and LEFT JOIN to return all matching

records in both tables

Outer query: A SQL statement containing a subquery

Ownership: The aspect of data ethics that presumes individuals own the raw data they provide

and have primary control over its usage, processing, and sharing

P

Packed bubble chart: A data visualization that displays data in clustered circles

Pattern: The design principle of using similar visual elements to demonstrate trends and

relationships in a data visualization

PHP (Hypertext Preprocessor): A programming language for web application development

Pie chart: A data visualization that uses segments of a circle to represent the proportions of

each data category compared to the whole

Pivot chart: A chart created from the fields in a pivot table

Pivot table: A data summarization tool used to sort, reorganize, group, count, total, or average

data

Pixel: In digital imaging, a small area of illumination on a display screen that, when combined

with other adjacent areas, forms a digital image

Population: In data analytics, all possible data values in a dataset

Portfolio: A collection of materials that can be shared with potential employers

Pre-attentive attributes: The elements of a data visualization that an audience recognizes

automatically without conscious effort

Primary key: An identifier in a database that references a column in which each value is unique

(Refer to foreign key)

Problem domain: The area of analysis that encompasses every activity affecting or affected by

a problem

Problem types: The various problems that data analysts encounter, including categorizing

things, discovering connections, finding patterns, identifying themes, making predictions, and

spotting something unusual

Profit margin: A percentage that indicates how many cents of profit has been generated for

each dollar of sale

Programming language: A system of words and symbols used to write instructions that

computers follow

Proportion: The design principle of using the relative size and arrangement of visual elements

to demonstrate information in a data visualization

Python: A general-purpose programming language

Q

Qualitative data: A subjective and explanatory measure of a quality or characteristic

Quantitative data: A specific and objective measure, such as a number, quantity, or range

Query: A request for data or information from a database

Query language: A computer programming language used to communicate with a database

R

R: A programming language used for statistical analysis, visualization, and other data analysis

Random sampling: A way of selecting a sample from a population so that every possible type

of the sample has an equal chance of being chosen

Range: A collection of two or more cells in a spreadsheet

Ranking: A system to position values of a dataset within a scale of achievement or status

Record: A collection of related data in a data table, usually synonymous with row

Redundancy: When the same piece of data is stored in two or more places

Reframing: The process of restating a problem or challenge, then redirecting it toward a

potential resolution

Regular expression (RegEx): A rule that says the values in a table must match a prescribed

pattern

Relational database: A database that contains a series of tables that can be connected to form

relationships

Relativity: The process of considering observations in relation or proportion to something else

Relevant question: A question that has significance to the problem to be solved

Remove duplicates: A spreadsheet tool that automatically searches for and eliminates

duplicate entries from a spreadsheet

Repetition: The design principle of repeating visual elements to demonstrate meaning in a data

visualization

Report: A static collection of data periodically given to stakeholders

Return on investment (ROI): A formula that uses the metrics of investment and profit to

evaluate the success of an investment

Revenue: The total amount of income generated by the sale of goods or services

Rhythm: The design principle of creating movement and flow in a data visualization to engage

an audience

RIGHT: A function that returns a set number of characters from the right side of a text string

RIGHT JOIN: A SQL function that will return all records from the right table and only the

matching records from the left

Root cause: The reason why a problem occurs

ROUND: A SQL function that returns a number rounded to a certain number of decimal places.

Ruby: An object-oriented programming language for web application development

S

Sample: In data analytics, a segment of a population that is representative of the entire

population

Sampling bias: Overrepresenting or underrepresenting certain members of a population as a

result of working with a sample that is not representative of the population as a whole

Scatterplot: A data visualization that represents relationships between different variables with

individual data points without a connecting line

Schema: A way of describing how something, such as data, is organized

Scope of work (SOW): An agreed-upon outline of the tasks to be performed during a project

Second-party data: Data collected by a group directly from its audience and then sold

SELECT: The section of a query that indicates from which column(s) to extract the data

SELECT INTO: A SQL clause that copies data from one table into a temporary table without

adding the new table to the database

Small data: Small, specific data points typically involving a short period of time, which are

useful for making day-to-day decisions

SMART methodology: A tool for determining a question’s effectiveness based on whether it is

specific, measurable, action-oriented, relevant, and time-bound

Social media: Websites and applications through which users create and share content or

participate in social networking

Soft skills: Nontechnical traits and behaviors that relate to how people work

Sort range: A spreadsheet menu function that sorts a specified range and preserves the cells

outside the range

Sort sheet: A spreadsheet menu function that sorts all data by the ranking of a specific sorted

column and keeps data together across rows

Sorting: The process of arranging data into a meaningful order to make it easier to understand,

analyze, and visualize

Specific question: A question that is simple, significant, and focused on a single topic or a few

closely related ideas

SPLIT: A spreadsheet function that divides text around a specified character and puts each

fragment into a new, separate cell

Sponsor: A professional advocate who is committed to moving forward the career of another

Spotlightling: Scanning through data to quickly identify the most important insights

Spreadsheet: A digital worksheet

SQL: (Refer to Structured Query Language)

Stakeholders: People who invest time and resources into a project and are interested in its

outcome

Static data: Data that doesn’t change once it has been recorded

Static visualization: A data visualization that does not change over time unless it is edited

Statistical power: The probability that a test of significance will recognize an effect that is

present

Statistical significance: The probability that sample results are not due to random chance

Statistics: The study of how to collect, analyze, summarize, and present data

Story: The narrative of a data presentation that makes it meaningful and interesting

String data type: A sequence of characters and punctuation that contains textual information

(also called text data type)

Structural metadata: Metadata that indicates how a piece of data is organized and whether it is

part of one or more than one data collection

Structured data: Data organized in a certain format such as rows and columns

Structured Query Language: A computer programming language used to communicate with a

database

Structured thinking: The process of recognizing the current problem or situation, organizing

available information, revealing gaps and opportunities, and identifying options

Subquery: A SQL query that is nested inside a larger query

SUBSTR: A SQL function that extracts a substring from a string variable

Substring: A subset of a text string

Subtitle: Text that supports a headline by adding context and description

SUM: A spreadsheet function that adds the values of a selected range of cells

SUMIF: A spreadsheet function that adds numeric data based on one condition

Summary table: A table used to summarize statistical information about data

SUMPRODUCT: A function that multiplies arrays and returns the sum of those products

Swift: A programming language for macOS, iOS, watchOS, and tvOS

Symbol map: A data visualization that displays a mark over a given longitude and latitude

Syntax: The predetermined structure of a language that includes all required words, symbols,

and punctuation, as well as their proper placement

T

Tableau: A business intelligence and analytics platform that helps people visualize, understand,

and make decisions with data

Technical mindset: The ability to break things down into smaller steps or pieces and work with

them in an orderly and logical way

Temporary table: A database table that is created and exists temporarily on a database server

Text data type: A sequence of characters and punctuation that contains textual information

(also called string data type)

Text string: A group of characters within a cell, most often composed of letters

Third-party data: Data provided from outside sources who didn’t collect it directly

Time-bound question: A question that specifies a timeframe to be studied

Transaction transparency: The aspect of data ethics that presumes all data-processing

activities and algorithms should be explainable and understood by the individual who provides

the data

Transferable skills: Skills and qualities that can transfer from one job or industry to another

TRIM: A function that removes leading, trailing, and repeated spaces in data

Turnover rate: The rate at which employees voluntarily leave a company

Typecasting: Converting data from one type to another

U

Unbiased sampling: When the sample of the population being measured is representative of

the population as a whole

Underscores: Lines used to underline words and connect text characters

Unfair question: A question that makes assumptions or is difficult to answer honestly

Unique: A value that can’t have a duplicate

United States Census Bureau: An agency in the U.S. Department of Commerce that serves as

the nation’s leading provider of quality data about its people and economy

Unity: The design principle of using visual elements that complement each other to create

aesthetic appeal and clarity in a data visualization

Unstructured data: Data that is not organized in any easily identifiable manner

V

Validity: The degree to which data conforms to constraints when it is input, collected, or created

VALUE: A spreadsheet function that converts a text string that represents a number to a

numeric value

Variety: The design principle of using different kinds of visual elements in a data visualization to

engage an audience

Verification: A process to confirm that a data-cleaning effort was well executed and the

resulting data is accurate and reliable

Video file: A collection of images, audio files, and other data usually encoded in a compressed

format such as MP4, MV4, MOV, AVI, or FLV

Visual form: The appearance of a data visualization that gives it structure and aesthetic appeal

Visualization: (Refer to Data visualization)

VLOOKUP: A spreadsheet function that vertically searches for a certain value in a column to

return a corresponding piece of information

W

WHERE: The section of a query that specifies criteria that the requested data must meet

Wide data: A dataset in which every data subject has a single row with multiple columns to hold

the values of various attributes of the subject

WITH: A SQL clause that creates a temporary table that can be queried multiple times

World Health Organization: An organization whose primary role is to direct and coordinate

international health within the United Nations system

X

X-axis: The horizontal line of a graph usually placed at the bottom, which is often used to

represent time scales and discrete categories

Y

Y-axis: The vertical line of a graph usually placed to the left, which is often used to represent

frequencies and other numerical variables

Z

### 1.

Question 1

What tool gives data analysts the highest level of control over their data analysis?

1 point

Spreadsheet

Programming language

SQL

Tableau

### 2.

Question 2

What are the benefits of using a programming language for data analysis? Select all that apply.

1 point

It is easy to share code.

It does not require data cleaning

It does not require specific syntax.

It is faster to clean data.

### 3.

Question 3

What attribute of the R programming language makes it an open-source programming language?

1 point

The code is distributed by a company named “Open-Source.”

The code is designed to be data-centric.

The code is open to processing large amounts of data.

The code can be modified and shared by anyone who uses it.

### 4.

Question 4

Which of the following statements about the R programming language are correct? Select all that apply.

1 point

It relies on spreadsheet interfaces to clean and manipulate data

It can create world-class visualizations

It makes analysts spend more time cleaning data and less time analyzing

It can process large amounts of data

### 5.

Question 5

A data analyst is searching for a single tool that will allow them to query massive amounts of data, reproduce their analysis, and create world-class visuals. Which of the following tools is the best option for them?

1 point

A database

A dashboard

The R programming language

SQL

### 6.

Question 6

Which of the following statements about RStudio’s integrated development environment are correct? Select all that apply.

1 point

The layout of panes in R studio is fixed.

R studio is built specifically for working with R.

R studio is unable to produce visualizations.

R studio helps with file management.

### 7.

Question 7

A data analyst wants to write a large R script instead of running single R commands. Where should they write their code in RStudio?

1 point

R console

Environment tab

Source editor

Files tab

### 8.

Question 8

In RStudio, where can you find and manage all the data you currently have loaded?

1 point

Plots tab

Source editor pane

Environment pane

R console pane

Hi and welcome back.

We've given you a big-picture overview of R and RStudio.

Now we'll turn our focus to

the actual programming and

coding you'll do using RStudio.

I went pretty far in my career not knowing programming

before it became clear,

I needed to learn it.

Getting to know R was

such a valuable learning experience.

It took some time,

and I reached out to more-experienced R users

with lots of questions.

Eventually, it all came together for me.

Being open to learning new skills is

such an important part of your career.

Now I'm able to help you learn some new skills too.

I'll start by sharing the fundamentals of

programming using R in RStudio.

Earlier, we explained how R is like the engine of

a car and RStudio is like the accelerator,

steering wheel, and dashboard all in one.

Getting to know fundamentals will help you

keep your R car running smoothly.

These fundamentals are both alike and different from

the other analysis platforms you've come to

know well: spreadsheets and SQL.

Then we'll move on to coding in RStudio.

We'll discuss the syntax for performing

calculations and the standards

and naming conventions for all code.

We'll also explore the R tool known as a pipe,

which you'll use to make a sequence of

code easier to work with and read.

Then we'll check out R packages.

While these packages won't be delivered to your door,

they are delivered by the R community.

These packages contain reusable functions and more,

and are usually built by users for users like yourself.

We'll get to know a collection of

packages called the Tidyverse.

You'll learn how to install the Tidyverse

so you can start using it in RStudio.

We'll also work with some of

the more popular Tidyverse packages

like ggplot2 for visualization.

You'll be able to carry over what you've learned about

RStudio to the next part of the program,

where you'll start working with data.

As we explained earlier, for this program,

we'll use the in-browser version

of RStudio: RStudio Cloud.

But RStudio is also available to be downloaded.

So let's get going. See you soon.

Hey there.

Anytime you're learning a new skill from cooking to driving to dancing,

you should always start with the fundamentals.

Programming with R is no different.

To build this foundation, you'll get familiar with the basic concepts of R,

including functions, comments, variables, data types, vectors, and pipes.

Some of these terms might sound familiar.

For example, we've come across functions in spreadsheets and SQL.

As a quick refresher, functions are a body of

reusable code used to perform specific tasks in R.

Functions begin with function names like print or paste, and

are usually followed by one or more arguments in parentheses.

An argument is information that a function in R needs in order to run.

Here's a simple function in action.

Feel free to join in and try it yourself in RStudio using your cloud account.

Check out the reading for more details on how to get started.

Play video starting at :1:10 and follow transcript1:10

You can pause the video anytime you need to.

We'll open RStudio Cloud to get started.

We'll start our function in the console with the function name print.

This function name will return whatever we include in the values in parentheses.

We'll type an open parenthesis followed by a quotation mark.

Both the close parenthesis and

end quote automatically pop up because RStudio recognizes this syntax.

Now we just have to add the text string.

We'll type Coding in R.

Play video starting at :1:45 and follow transcript1:45

Then we'll press enter.

Play video starting at :1:48 and follow transcript1:48

Success! The code returns the words "Coding in R."

If you want to find out more about the print function or any function, all you

have to do is type a question mark, the function name, and a set of parentheses.

Play video starting at :2:5 and follow transcript2:05

This returns a page in the Help window,

which helps you learn more about the functions you're working with.

Keep in mind that functions are case-sensitive,

so typing Print with a Capital P brings back an error message.

Play video starting at :2:24 and follow transcript2:24

Functions are great, but it can be pretty time-consuming to type out lots of values.

To save time, we can use variables to represent the values.

This lets us call out the values any time we need to with just the variable.

Earlier, we learned about variables in SQL.

A variable is a representation of a value in R that can be stored for

use later during programming.

Variables can also be called objects.

As a data analyst, you'll find variables are very useful when programming.

For example, if you want to filter a dataset,

just assign a variable to the function you used to filter the data.

That way, all you have to do is use that variable to filter the data later.

When naming a variable in R, you can use a short phrase.

A variable name should start with a letter and

can also contain numbers and underscores.

So the variable 5penguin wouldn't work well because it starts with a number.

Also just like functions, variable names are case-sensitive.

Using all lower case letters is good practice whenever possible.

Now, before we get to coding a variable, let's add a comment.

Comments are helpful when you want to describe or

explain what's going on in your code.

Use them as much as possible so that you and

everyone can understand the reasoning behind it.

Comments should be used to make an R script more readable.

A comment shouldn't be treated as code, so we'll put a # in front of it.

Then we'll add our comment.

Here's an example of a variable.

Play video starting at :4:16 and follow transcript4:16

Now let's go ahead with our example.

It makes sense to use a variable name to connect to what the variable is

representing.

So we'll type the variable name first\_variable.

Play video starting at :4:30 and follow transcript4:30

Then after the variable name, we'll type a < sign, followed by a -.

Play video starting at :4:36 and follow transcript4:36

This is the assignment operator.

It assigns the value to the variable.

It looks like an arrow, which makes sense,

since it's pointing from the value to the variable.

There are other assignment operators that work too, but

it's always good to stick with just one type in your code.

Next, we'll add the value that our variable will represent.

We'll use the text, "This is my variable."

Play video starting at :5:5 and follow transcript5:05

If we type the variable and hit Run,

it will return the value that the variable represents.

This is a very basic way of using a variable.

You'll learn more ways of using variables in your code soon.

For now, let's assign a variable to a different data type, numeric.

We'll name this second\_variable, and type our assignment operator.

Play video starting at :5:30 and follow transcript5:30

We'll give it the numeric value 12.5.

Play video starting at :5:35 and follow transcript5:35

The Environment pane in the upper- right part of our work space now shows

both of our variables and their values.

There are other data types in R like logical, date, and date time.

R has a few options for dealing with these data types. We'll explore them later.

With functions, comments, variables, and data types,

you've got a good foundation for working with R.

We'll revisit these throughout this program, and

show you how they're used in different ways during analysis.

Let's finish up with two more fundamental concepts, vectors and pipes.

Simply put, a vector is a group of data elements of the same

type stored in a sequence in R.

You can make a vector using the combined function.

In R this function is just the letter c followed

by the values you want in your vector inside parentheses.

All right, let's create a vector.

Imagine this vector is for a measurement data that we need to analyze.

We'll start our code with the variable vec\_1 to assign to the vector.

Play video starting at :6:52 and follow transcript6:52

Then we'll type c and the open parenthesis.

Play video starting at :6:57 and follow transcript6:57

Then we'll type our list of numbers separated by commas.

Play video starting at :7:4 and follow transcript7:04

We'll then close our parentheses and press enter.

Play video starting at :7:11 and follow transcript7:11

This time when we type our variable and press enter, it returns our vector.

We can use this vector anywhere in our analysis with only

its variable name vec\_1.

The values in the vector will automatically be applied to our analysis.

That brings us to the last of our fundamentals, pipes.

A pipe is a tool in R for expressing a sequence of multiple operations.

A pipe is represented by a % sign, followed by a > sign, and another % sign.

It's used to apply the output of one function into another function.

Pipes can make your code easier to read and understand.

For example, this pipe filters and sorts the data.

Later, we'll learn how each part of the pipe works.

So there they are, the super six fundamentals: functions,

comments, variables, data types, vectors, and pipes.

They all work together as a foundation for using R.

It's a lot to take in, so

feel free to watch any of these videos again if you need a refresher.

When you're ready, there's so much more to know about R and RStudio.

So let's get to it.

# Vectors and lists in R

You can save this reading for future reference. Feel free to download a PDF version of this reading below:

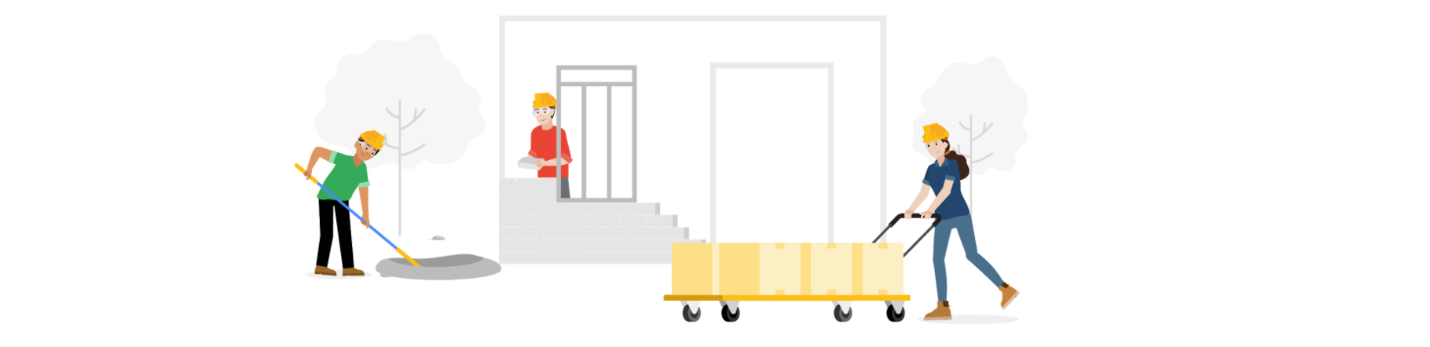
[Vectors and lists in R.pdf](https://d3c33hcgiwev3.cloudfront.net/kmBrpkVZSdWRWf-p0_1Djw_d9198f7867bf42f2acabc6750523dff1_Vectors-and-lists-in-R.pdf?Expires=1706918400&Signature=F~BBHeWPwDZPiCtY418qmJZ3oAL3lt4Oa~ex8buf6i8Aare69UaWDHt8Zi2OHpPwyplQQ-j1J5cP8BmGnpCQchnpLi1LI8iXiOiQOG5-xYVQNkdAf~cnnRYWjWF9CUx8Emp2zTb1CFjIuB0cwhrq2jekjQmPFYbFuaQPLgbrxAg_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[PDF File](https://d3c33hcgiwev3.cloudfront.net/kmBrpkVZSdWRWf-p0_1Djw_d9198f7867bf42f2acabc6750523dff1_Vectors-and-lists-in-R.pdf?Expires=1706918400&Signature=F~BBHeWPwDZPiCtY418qmJZ3oAL3lt4Oa~ex8buf6i8Aare69UaWDHt8Zi2OHpPwyplQQ-j1J5cP8BmGnpCQchnpLi1LI8iXiOiQOG5-xYVQNkdAf~cnnRYWjWF9CUx8Emp2zTb1CFjIuB0cwhrq2jekjQmPFYbFuaQPLgbrxAg_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

In programming, a **data structure** is a format for organizing and storing data. Data structures are important to understand because you will work with them frequently when you use R for data analysis. The most common data structures in the R programming language include:

* Vectors
* Data frames
* Matrices
* Arrays

Think of a data structure like a house that contains your data.



This reading will focus on vectors. Later on, you’ll learn more about data frames, matrices, and arrays.

There are two types of vectors: **atomic vectors** and **lists**. Coming up, you’ll learn about the basic properties of atomic vectors and lists, and how to use R code to create them.

## Atomic vectors

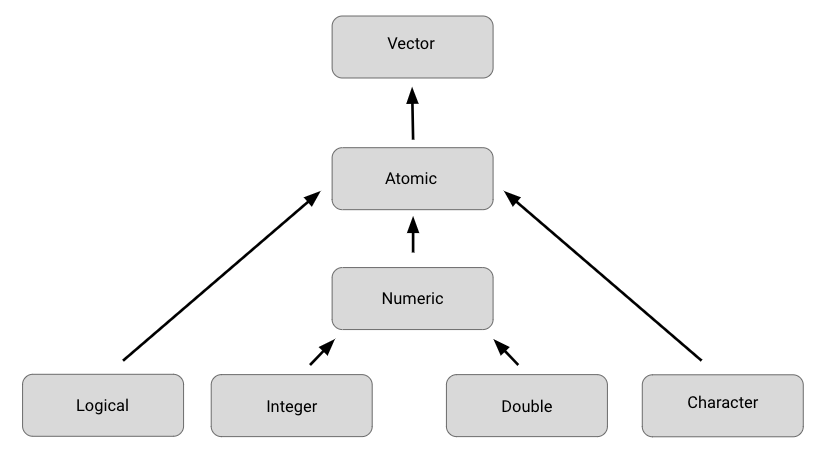
First, we will go through the different types of atomic vectors. Then, you will learn how to use R code to create, identify, and name the vectors.

Earlier, you learned that a **vector** is a group of data elements of the same type, stored in a sequence in R. You cannot have a vector that contains both logicals and numerics.

There are six primary types of atomic vectors: logical, integer, double, character (which contains strings), complex, and raw. The last two–complex and raw–aren’t as common in data analysis, so we will focus on the first four. Together, integer and double vectors are known as numeric vectors because they both contain numbers. This table summarizes the four primary types:

| **Type** | **Description** | **Example** |
| --- | --- | --- |
| Logical | True/False | **TRUE** |
| Integer | Positive and negative whole values | **3** |
| Double | Decimal values | **101.175** |
| Character | String/character values | **“Coding”** |

This diagram illustrates the hierarchy of relationships among these four main types of vectors:

Bottom: logical (arrow points to atomic), integer (arrow points to numeric), double (arrow points to numeric), character (arrow points to atomic) Second to bottom: numeric (arrow points to atomic) second level: atomic (arrow points to vector) top: vector

### **Creating vectors**

One way to create a vector is by using the **c()** function (called the “combine” function). The c() function in R combines multiple values into a vector. In R, this function is just the letter “c” followed by the values you want in your vector inside the parentheses, separated by a comma: c(x, y, z, …).

For example, you can use the c() function to store numeric data in a vector.

**c(2.5, 48.5, 101.5)**

To create a vector of integers using the c() function, you must place the letter "L" directly after each number.

**c(1L, 5L, 15L)**

You can also create a vector containing characters or logicals.

**c(“Sara” , “Lisa” , “Anna”)**

**c(TRUE, FALSE, TRUE)**

### **Determining the properties of vectors**

Every vector you create will have two key properties: type and length.

 You can determine what type of vector you are working with by using the **typeof()** function. Place the code for the vector inside the parentheses of the function. When you run the function, R will tell you the type. For example:

**typeof(c(“a” , “b”))**

**#> [1] "character"**

 Notice that the output of the typeof function in this example is **“character”**. Similarly, if you use the typeof function on a vector with integer values, then the output will include **“integer”** instead:

**typeof(c(1L , 3L))**

**#> [1] "integer"**

You can determine the length of an existing vector–meaning the number of elements it contains–by using the **length()** function. In this example, we use an assignment operator to assign the vector to the variable x. Then, we apply the length() function to the variable. When we run the function, R tells us the length is **3**.

**x <- c(33.5, 57.75, 120.05)**

**length(x)**

**#> [1] 3**

You can also check if a vector is a specific type by using an **is** function: **is.logical(), is.double(), is.integer(), is.character()**. In this example, R returns a value of **TRUE** because the vector contains integers.

**x <- c(2L, 5L, 11L)**

**is.integer(x)**

**#> [1] TRUE**

In this example, R returns a value of **FALSE** because the vector does not contain characters, rather it contains logicals.

**y <- c(TRUE, TRUE, FALSE)**

**is.character(y)**

**#> [1] FALSE**

### **Naming vectors**

All types of vectors can be named. Names are useful for writing readable code and describing objects in R. You can name the elements of a vector with the **names()** function. As an example, let’s assign the variable x to a new vector with three elements.

**x <- c(1, 3, 5)**

You can use the names() function to assign a different name to each element of the vector.

**names(x) <- c("a", "b", "c")**

Now, when you run the code, R shows that the first element of the vector is named **a**, the second **b**, and the third **c**.

**x**

**#> a b c**

**#> 1 3 5**

Remember that an atomic vector can only contain elements of the same type. If you want to store elements of different types in the same data structure, you can use a list.

## Creating lists

**Lists** are different from atomic vectors because their elements can be of any type—like dates, data frames, vectors, matrices, and more. Lists can even contain other lists.

You can create a list with the **list()** function. Similar to the c() function, the list() function is just **list** followed by the values you want in your list inside parentheses: **list(x, y, z, …)**. In this example, we create a list that contains four different kinds of elements: character (**"a"**), integer (**1L**), double (**1.5**), and logical (**TRUE**).

**list("a", 1L, 1.5, TRUE)**

Like we already mentioned, lists can contain other lists. If you want, you can even store a list inside a list inside a list—and so on.

**list(list(list(1 , 3, 5)))**

### **Determining the structure of lists**

If you want to find out what types of elements a list contains, you can use the **str()** function. To do so, place the code for the list inside the parentheses of the function. When you run the function, R will display the data structure of the list by describing its elements and their types.

Let’s apply the str() function to our first example of a list.

**str(list("a", 1L, 1.5, TRUE))**

We run the function, then R tells us that the list contains four elements, and that the elements consist of four different types: character (**chr**), integer (**int**), number (**num**), and logical (**logi**).

**#> List of 4**

**#>  $ : chr "a"**

**#>  $ : int 1**

**#>  $ : num 1.5**

**#>  $ : logi TRUE**

Let’s use the str() function to discover the structure of our second example.  First, let’s assign the list to the variable z to make it easier to input in the str() function.

**z <- list(list(list(1 , 3, 5)))**

Let’s run the function.

**str(z)**

**#> List of 1**

**#>  $ :List of 1**

**#>   ..$ :List of 3**

**#>   .. ..$ : num 1**

**#>   .. ..$ : num 3**

**#>   .. ..$ : num 5**

The indentation of the **$** symbols reflect the nested structure of this list. Here, there are three levels (so there is a list within a list within a list).

### **Naming lists**

Lists, like vectors, can be named. You can name the elements of a list when you first create it with the list() function:

**list('Chicago' = 1, 'New York' = 2, 'Los Angeles' = 3)**

**$`Chicago`**

**[1] 1**

**$`New York`**

**[1] 2**

**$`Los Angeles`**

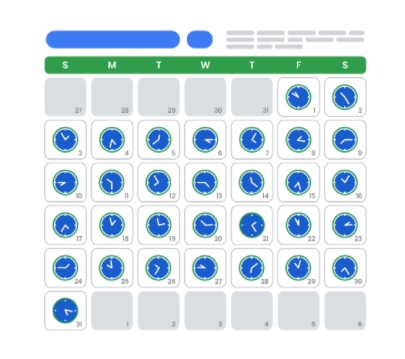
**[1] 3**

## Additional resource

To learn more about vectors and lists, check out [R for Data Science, Chapter 20: Vectors](https://r4ds.had.co.nz/vectors.html#vectors). R for Data Science is a classic resource for learning how to use R for data science and data analysis. It covers everything from cleaning to visualizing to communicating your data. If you want to get more details about the topic of vectors and lists, this chapter is a great place to start.

# Dates and times in R

In this reading, you will learn how to work with dates and times in R using the **lubridate** package. Coming up, you will use tools in the lubridate package to convert different types of data in R into date and date-time formats.



## Loading tidyverse and lubridate packages

Before you get started working with dates and times, you should load both **tidyverse** and **lubridate**. Lubridate is part of tidyverse.

First, open RStudio.

If you haven't already installed tidyverse, you can use the **install.packages()** function to do so:

* **install.packages("tidyverse")**

Next, load the tidyverse and lubridate packages using the **library()** function. First, load the core tidyverse to make it available in your current R session:

* **library(tidyverse)**

Then, load the lubridate package:

* **library(lubridate)**

Now you’re ready to be introduced to the tools in the lubridate package.

## Working with dates and times

This section covers the data types for dates and times in R and how to convert strings to date-time formats.

### **Types**

In R, there are three types of data that refer to an instant in time:

* A date **("2016-08-16")**
* A time within a day **(“20:11:59 UTC")**
* And a date-time. This is a date plus a time **("2018-03-31 18:15:48 UTC")**

The time is given in UTC, which stands for Universal Time Coordinated, more commonly called Universal Coordinated Time. This is the primary standard by which the world regulates clocks and time.

For example, to get the current date you can run the **today()** function. The date appears as year, month, and day.

**today()**

**#> [1] "2021-01-20"**

To get the current date-time you can run the **now()** function. Note that the time appears to the nearest second.

**now()**

**#> [1] "2021-01-20 16:25:05 UTC"**

When working with R, there are three ways you are likely to create date-time formats:

* From a string
* From an individual date
* From an existing date/time object

R creates dates in the standard yyyy-mm-dd format by default.

Let's go over each.

### **Converting from strings**

Date/time data often comes as strings. You can convert strings into dates and date-times using the tools provided by lubridate. These tools automatically work out the date/time format. First, identify the order in which the year, month, and day appear in your dates. Then, arrange the letters y, m, and d in the same order. That gives you the name of the lubridate function that will parse your date. For example, for the date 2021-01-20, you use the order ymd:

**ymd("2021-01-20")**

When you run the function, R returns the date in yyyy-mm-dd format.

**#> [1] "2021-01-20"**

It works the same way for any order. For example, month, day, and year. R still returns the date in yyyy-mm-dd format.

**mdy("January 20th, 2021")**

**#> [1] "2021-01-20"**

Or, day, month, and year. R still returns the date in yyyy-mm-dd format.

**dmy("20-Jan-2021")**

**#> [1] "2021-01-20"**

These functions also take unquoted numbers and convert them into the yyyy-mm-dd format.

**ymd(20210120)**

**#> [1] "2021-01-20"**

### **Creating date-time components**

The ymd() function and its variations create dates. To create a date-time from a date, add an underscore and one or more of the letters h, m, and s (hours, minutes, seconds) to the name of the function:

**ymd\_hms("2021-01-20 20:11:59")**

**#> [1] "2021-01-20 20:11:59 UTC"**

**mdy\_hm("01/20/2021 08:01")**

**#> [1] "2021-01-20 08:01:00 UTC"**

### **Optional: Switching between existing date-time objects**

Finally, you might want to switch between a date-time and a date.

You can use the function **as\_date()** to convert a date-time to a date. For example, put the current date-time—now()—in the parentheses of the function.

**as\_date(now())**

**#> [1] "2021-01-20"**

## Additional resources

To learn more about working with dates and times in R, check out the following resources:

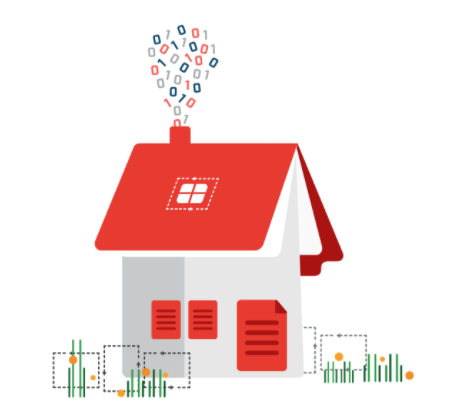
* [lubridate.tidyverse](https://lubridate.tidyverse.org/index.html): This is the “lubridate” entry from the official tidyverse documentation, which offers a comprehensive reference guide to the various tidyverse packages. Check out this link for an overview of key concepts and functions.
* [Dates and times with lubridate: Cheat Sheet](https://rawgit.com/rstudio/cheatsheets/master/lubridate.pdf): This “cheat sheet” gives you a detailed map of all the different things you can do with the lubridate package. You don’t need to know all of this information, but the cheat sheet is a useful reference for any questions you might have about working with dates and times in R.

# Other common data structures

In this reading, you will continue on the topic of data structures with an introduction to data frames and matrices. You will learn about the basic properties of each structure, and simple ways to make use of them using R code. You will also briefly explore **files**, which are often used to access and store data and related information.

## Data structures

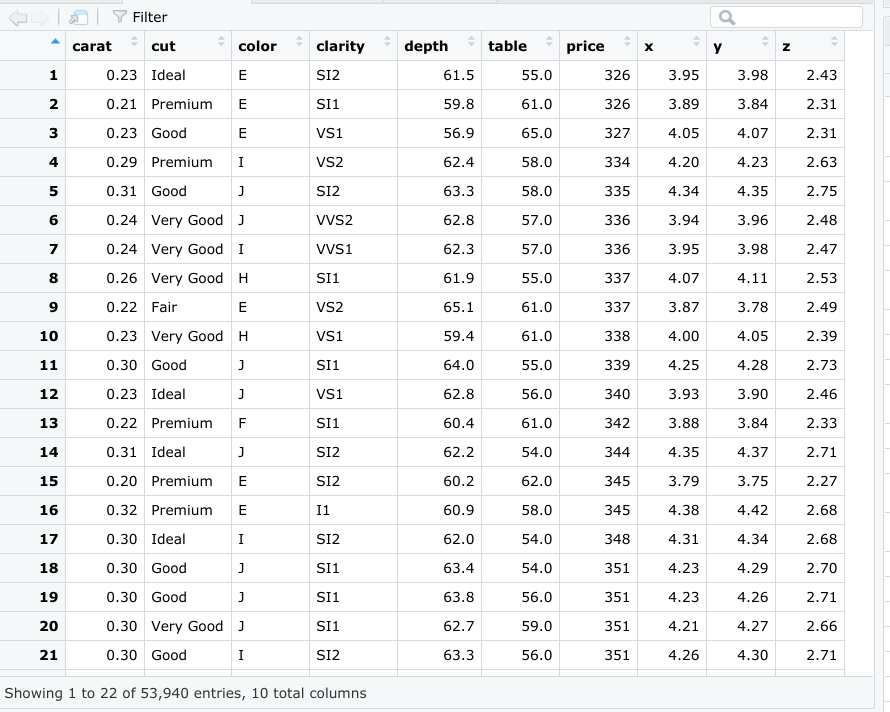
Recall that a data structure is like a house that contains your data.



### **Data frames**

Data frames are the most common way of storing and analyzing data in R, so it’s important to understand what they are and how to create them. A **data frame** is a collection of columns–similar to a spreadsheet or SQL table. Each column has a name at the top that represents a variable, and includes one observation per row. Data frames help summarize data and organize it into a format that is easy to read and use.

For example, the data frame below shows the “diamonds” dataset, which is one of the preloaded datasets in R. Each column contains a single variable that is related to diamonds: carat, cut, color, clarity, depth, and so on. Each row represents a single observation.



There are a few key things to keep in mind when you are working with data frames:

* First, columns should be named.
* Second, data frames can include many different types of data, like numeric, logical, or character.
* Finally, elements in the same column should be of the same type.

You will learn more about data frames later on in the program, but this is a great starting point.

If you need to manually create a data frame in R, you can use the **data.frame()** function. The data.frame() function takes vectors as input. In the parentheses, enter the name of the column, followed by an equals sign, and then the vector you want to input for that column. In this example, the x column is a vector with elements 1, 2, 3, and the y column is a vector with elements 1.5, 5.5, 7.5.

**data.frame(x = c(1, 2, 3) , y = c(1.5, 5.5, 7.5))**

If you run the function, R displays the data frame in ordered rows and columns.

**x y**

**1  1 1.5**

**2  2 5.5**

**3  3 7.5**

In most cases, you won’t need to manually create a data frame yourself, as you will typically import data from another source, such as a .csv file, a relational database, or a software program.

## Files

Let’s go over how to create, copy, and delete files in R. For more information on working with files in R, check out [R documentation: files](https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/files). **R documentation** is a tool that helps you easily find and browse the documentation of almost all R packages on CRAN. It’s a useful reference guide for functions in R code. Let’s go through a few of the most useful functions for working with files.

Use the **dir.create** function to create a new folder, or directory, to hold your files. Place the name of the folder in the parentheses of the function.

**dir.create ("destination\_folder")**

Use the **file.create()** function to create a blank file. Place the name and the type of the file in the parentheses of the function. Your file types will usually be something like .txt, .docx, or .csv.

**file.create (“new\_text\_file.txt”)**

**file.create (“new\_word\_file.docx”)**

**file.create (“new\_csv\_file.csv”)**

If the file is successfully created when you run the function, R will return a value of **TRUE** (if not, R will return **FALSE**).

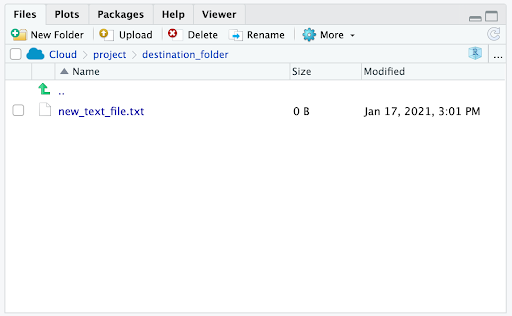
**file.create (“new\_csv\_file.csv”)**

**[1] TRUE**

Copying a file can be done using the **file.copy()** function. In the parentheses, add the name of the file to be copied. Then, type a comma, and add the name of the destination folder that you want to copy the file to.

**file.copy (“new\_text\_file.txt” , “destination\_folder”)**

If you check the Files pane in RStudio, a copy of the file appears in the relevant folder:



You can delete R files using the **unlink()** function. Enter the file’s name in the parentheses of the function.

**unlink (“some\_.file.csv”)**

## Additional resource

If you want to learn more about working with data frames, matrices, and arrays in R, check out the [Data Wrangling](http://statseducation.com/Introduction-to-R/modules/getting%20data/data-wrangling/) section of Stat Education's Introduction to R course. The section includes modules on data frames, matrices, and arrays (and more), and each module contains helpful examples of key coding concepts.

--------------------------------------------------------------------------------------------------------------------------------------

## Optional: Matrices

A **matrix** is a two-dimensional collection of data elements. This means it has both rows and columns. By contrast, a vector is a one-dimensional sequence of data elements. But like vectors, matrices can only contain a single data type. For example, you can’t have both logicals and numerics in a matrix.

To create a matrix in R, you can use the **matrix()** function. The matrix() function has two main arguments that you enter in the parentheses. First, add a vector. The vector contains the values you want to place in the matrix. Next, add at least one matrix dimension. You can choose to specify the number of rows or the number of columns by using the code **nrow =** or **ncol =**.

For example, imagine you want to create a 2x3 (two rows by three columns) matrix containing the values 3-8. First, enter a vector containing that series of numbers: **c(3:8)**. Then, enter a comma. Finally, enter **nrow = 2** to specify the number of rows.

**matrix(c(3:8), nrow = 2)**

If you run the function, R displays a matrix with three columns and two rows (typically referred to as a “2x3”) that contain the numeric values 3, 4, 5, 6, 7, 8. R places the first value (3) of the vector in the uppermost row, and the leftmost column of the matrix, and continues the sequence from left to right.

**[,1] [,2] [,3]**

**[1,]    3    5    7**

**[2,]    4    6    8**

You can also choose to specify the number of columns (**ncol =** ) instead of the number of rows (**nrow =** ).

**matrix(c(3:8), ncol = 2)**

When you run the function, R infers the number of rows automatically.

**[,1] [,2]**

**[1,]    3    6**

**[2,]    4    7**

**[3,]    5    8**

### 1.

Question 1

Why do analysts use comments In R programming? Select all that apply.

1 point

To explain their code

To make an R Script more readable

To act as functions

To provide names for variables

### 2.

Question 2

What should you use to assign a value to a variable in R?

1 point

A vector

A comment

An argument

An operator

### 3.

Question 3

Which of the following examples is the proper syntax for a function in R?

1 point

#first

print()

<- 20

data\_1

### 4.

Question 4

Which of the following examples can you use in R for date/time data? Select all that apply.

1 point

**06:11:13 UTC**

**2019-04-16**

**seven-24-2018**

**2018-12-21 16:35:28 UTC**

# Logical operators and conditional statements

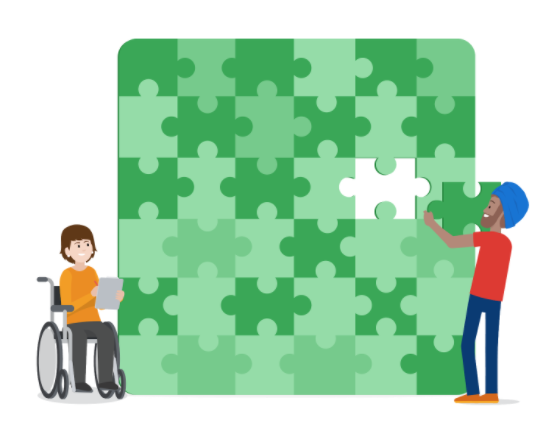
**Tip:** You may refresh on the concepts presented in [Understanding Boolean logic](https://www.coursera.org/learn/data-preparation/supplement/GgZMN/understanding-boolean-logic) to help you understand how logical operators work.

You can save this reading for future reference. Feel free to download a PDF version of this reading below:

[Logical operators and conditional statements.pdf](https://d3c33hcgiwev3.cloudfront.net/IH3jvjscStK94747HNrSCg_9681cba255d44707b891ea5e0eb0e2f1_Logical-operators-and-conditional-statements.pdf?Expires=1706918400&Signature=HN08KgdhI7IiTlovylpsPzhtdas8UctbCiQkDk8vAAlg24ithPi3KMvibOGwiIFFIMXjZNPggPpGAi0K7gdgG~cWnfA7fmuuzEywalwo4UdthdmPOtRzbf~pxpV51ByM8ZXcI0TJBl2FBi9jjepmhmWJkF59x6gWYkciX7pwx7c_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[PDF File](https://d3c33hcgiwev3.cloudfront.net/IH3jvjscStK94747HNrSCg_9681cba255d44707b891ea5e0eb0e2f1_Logical-operators-and-conditional-statements.pdf?Expires=1706918400&Signature=HN08KgdhI7IiTlovylpsPzhtdas8UctbCiQkDk8vAAlg24ithPi3KMvibOGwiIFFIMXjZNPggPpGAi0K7gdgG~cWnfA7fmuuzEywalwo4UdthdmPOtRzbf~pxpV51ByM8ZXcI0TJBl2FBi9jjepmhmWJkF59x6gWYkciX7pwx7c_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

Earlier, you learned that an **operator** is a symbol that identifies the type of operation or calculation to be performed in a formula. In this reading, you will learn about the main types of logical operators and how they can be used to create conditional statements in R code.



## Logical operators

**Logical operators** return a logical data type such as TRUE or FALSE.

There are three primary types of logical operators:

* AND (sometimes represented as & or && in R)
* OR (sometimes represented as | or || in R)
* NOT (!)

Review the summarized logical operators below.

### **AND operator “&”**

* The AND operator takes two logical values. It returns **TRUE** only if both individual values are TRUE. This means that TRUE & TRUE evaluates to **TRUE**. However, FALSE & TRUE, TRUE & FALSE, and FALSE & FALSE all evaluate to **FALSE**.
* If you run the corresponding code in R, you get the following results: **> TRUE & TRUE [1] TRUE > TRUE & FALSE [1] FALSE > FALSE & TRUE [1] FALSE > FALSE & FALSE [1] FALSE** You can illustrate this using the results of our comparisons. Imagine you create a variable x that is equal to 10.  **x <- 10** To check if x is greater than 3 but less than 12, you can use x > 3 and x < 12 as the values of an “AND” expression.  **x > 3 & x < 12** When you run the function, R returns the result TRUE. **[1] TRUE** The first part, **x > 3** will evaluate to **TRUE** since 10 is greater than 3. The second part, **x < 12** will also evaluate to **TRUE** since 10 is less than 12. So, since both values are TRUE, the result of the AND expression is **TRUE**. The number 10 lies between the numbers 3 and 12.  However, if you make x equal to 20, the expression **x > 3 & x < 12** will return a different result.  **x <- 20 x > 3 & x < 12 [1] FALSE** Although **x > 3** is **TRUE** (20 > 3), **x < 12** is **FALSE** (20 < 12). If one part of an AND expression is FALSE, the entire expression is FALSE (TRUE & FALSE = FALSE). So, R returns the result **FALSE**.

### **OR operator “|”**

* The OR operator (|) works in a similar way to the AND operator (&). The main difference is that at least one of the values of the OR operation must be TRUE for the entire OR operation to evaluate to **TRUE**. This means that TRUE | TRUE, TRUE | FALSE, and FALSE | TRUE all evaluate to **TRUE**. When both values are FALSE, the result is **FALSE**.
* If you write out the code, you get the following results:  **> TRUE | TRUE [1] TRUE > TRUE | FALSE [1] TRUE > FALSE | TRUE [1] TRUE > FALSE | FALSE [1] FALSE** For example, suppose you create a variable y equal to 7. To check if y is less than 8 or greater than 16, you can use the following expression: **y <- 7 y < 8 | y > 16** The comparison result is TRUE (7 is less than 8) | FALSE (7 is not greater than 16). Since only one value of an OR expression needs to be TRUE for the entire expression to be TRUE, R returns a result of TRUE.  **[1] TRUE** Now, suppose y is 12. The expression y < 8 | y > 16 now evaluates to FALSE (12 < 8) | FALSE (12 > 16). Both comparisons are FALSE, so the result is **FALSE**. **y <- 12 y < 8 | y > 16 [1] FALSE**

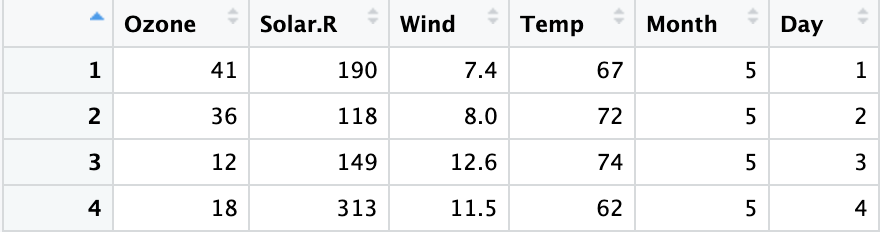
### **NOT operator “!”**

* The NOT operator (!) simply negates the logical value it applies to. In other words, !TRUE evaluates to **FALSE**, and !FALSE evaluates to **TRUE**.
* When you run the code, you get the following results:  **> !TRUE  [1] FALSE > !FALSE [1] TRUE** Just like the OR and AND operators, you can use the NOT operator in combination with logical operators. Zero is considered FALSE and non-zero numbers are taken as TRUE. The NOT operator evaluates to the opposite logical value.  Let’s imagine you have a variable x that equals 2:  **x <- 2** The NOT operation evaluates to FALSE because it takes the opposite logical value of a non-zero number (TRUE).  **> !x [1] FALSE**

-----------------

Let’s check out an example of how you might use logical operators to analyze data. Imagine you are working with the airquality dataset that is preloaded in RStudio. It contains data on daily air quality measurements in New York from May to September of 1973.

The data frame has six columns: Ozone (the ozone measurement), Solar.R (the solar measurement), Wind (the wind measurement), Temp (the temperature in Fahrenheit), and the Month and Day of these measurements (each row represents a specific month and day combination).



Let’s go through how the AND, OR, and NOT operators might be helpful in this situation.

### **AND** **example**

Imagine you want to specify rows that are extremely sunny and windy, which you define as having a Solar measurement of over 150 and a Wind measurement of over 10.

In R, you can express this logical statement as **Solar.R > 150 & Wind > 10**.

Only the rows where both of these conditions are true fulfill the criteria:

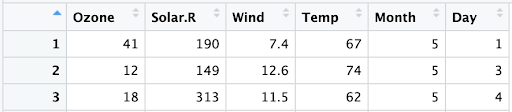
Image of a single row of the “airquality” dataset in the RStudio data viewer.

### **OR example**

Next, imagine you want to specify rows where it’s extremely sunny or it’s extremely windy, which you define as having a Solar measurement of over 150 or a Wind measurement of over 10.

In R, you can express this logical statement as **Solar.R > 150 | Wind > 10**.

All the rows where either of these conditions are true fulfill the criteria:

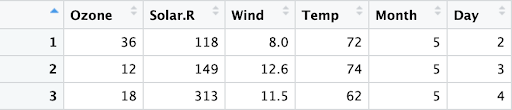


### **NOT example**

Now, imagine you just want to focus on the weather measurements for days that aren't the first day of the month.

In R, you can express this logical statement as **Day != 1**.

The rows where this condition is true fulfill the criteria:



Finally, imagine you want to focus on scenarios that aren't extremely sunny and not extremely windy, based on your previous definitions of extremely sunny and extremely windy. In other words, the following statement should not be true: either a Solar measurement greater than 150 or a Wind measurement greater than 10.

Notice that this statement is the opposite of the OR statement used above. To express this statement in R, you can put an exclamation point (!) in front of the previous OR statement: **!(Solar.R > 150 | Wind > 10)**. R will apply the NOT operator to everything within the parentheses.

In this case, only one row fulfills the criteria:



----------------------------------------------------------------------------------------------------------------------------------------

## Optional: Conditional statements

A **conditional statement** is a declaration that if a certain condition holds, then a certain event must take place. For example, “If the temperature is above freezing, then I will go outside for a walk.” If the first condition is true (the temperature is above freezing), then the second condition will occur (I will go for a walk). Conditional statements in R code have a similar logic.

Let’s discuss how to create conditional statements in R using three related statements:

* **if()**
* **else()**
* **else if()**

### **if statement**

The **if** statement sets a condition, and if the condition evaluates to **TRUE**, the R code associated with the if statement is executed.

In R, you place the code for the condition inside the parentheses of the if statement. The code that has to be executed if the condition is TRUE follows in curly braces (**expr**). Note that in this case, the second curly brace is placed on its own line of code and identifies the end of the code that you want to execute.

**if (condition) {**

**expr**

**}**

For example, let’s create a variable x equal to 4.

**x <- 4**

Next, let’s create a conditional statement: if x is greater than 0, then R will print out the string **“x is a positive number".**

**if (x > 0) {**

**print("x is a positive number")**

**}**

Since x = 4, the condition is true (4 > 0). Therefore, when you run the code, R prints out the string **“x is a positive number"**.

**[1] "x is a positive number"**

But if you change x to a negative number, like -4, then the condition will be FALSE (-4 > 0). If you run the code, R will not execute the print statement. Instead, a blank line will appear as the result.

### **else statement**

The **else** statement is used in combination with an if statement. This is how the code is structured in R:

**if (condition) {**

**expr1**

**} else {**

**expr2**

**}**

The code associated with the else statement gets executed whenever the condition of the if statement is not TRUE. In other words, if the condition is TRUE, then R will execute the code in the if statement (expr1); if the condition is not TRUE, then R will execute the code in the else statement (expr2).

Let’s try an example. First, create a variable x equal to 7.

**x <- 7**

Next, let’s set up the following conditions:

* If x is greater than 0, R will print **“x is a positive number”**.
* If x is less than or equal to 0, R will print **“x is either a negative number or zero”**.

In our code, the first condition (x > 0) will be part of the if statement. The second condition of x less than or equal to 0 is implied in the else statement. If x > 0, then R will print **“x is a positive number”**. Otherwise, R will print **“x is either a negative number or zero”**.

**x <- 7**

**if (x > 0) {**

**print ("x is a positive number")**

**} else {**

**print ("x is either a negative number or zero")**

**}**

Since 7 is greater than 0, the condition of the if statement is true. So, when you run the code, R prints out **“x is a positive number”**.

**[1] "x is a positive number"**

But if you make x equal to -7, the condition of the if statement is not true (-7 is not greater than 0). Therefore, R will execute the code in the else statement. When you run the code, R prints out **“x is either a negative number or zero”**.

**x <- -7**

**if (x > 0) {**

**print("x is a positive number")**

**} else {**

**print ("x is either a negative number or zero")**

**}**

**[1] "x is either a negative number or zero"**

### **else if statement**

In some cases, you might want to customize your conditional statement even further by adding the **else if** statement. The else if statement comes in between the if statement and the else statement. This is the code structure:

**if (condition1) {**

**expr1**

**} else if (condition2) {**

**expr2**

**} else {**

**expr3**

**}**

If the if condition (condition1) is met, then R executes the code in the first expression (expr1). If the if condition is not met, and the else if condition (condition2) is met, then R executes the code in the second expression (expr2). If neither of the two conditions are met, R executes the code in the third expression (expr3).

In our previous example, using only the if and else statements, R can only print **“x is either a negative number or zero”** if x equals 0 or x is less than zero. Imagine you want R to print the string **“x is zero”** if x equals 0. You need to add another condition using the else if statement.

Let’s try an example. First, create a variable x equal to negative 1 (“-1”), and run the code to save the variable to memory.

**x <- -1**

Now, you want to set up the following conditions:

* If x is less than 0, print **“x is a negative number”**
* If x equals 0, print **“x is zero”**
* Otherwise, print **“x is a positive number”**

In the code, the first condition will be part of the if statement, the second condition will be part of the else if statement, and the third condition will be part of the else statement. If x < 0, then R will print **“x is a negative number”.** If x = 0, then R will print **“x is zero”**. Otherwise, R will print **“x is a positive number”**.

**x <- -1**

**# run the code**

**if (x < 0) {**

**print("x is a negative number")**

**} else if (x == 0) {**

**print("x is zero")**

**} else {**

**print("x is a positive number")**

**}**

Run the code. Since -1 is less than 0,  the condition for the if statement evaluates to **TRUE**, and R prints **“x is a negative number”**.

**[1] "x is a negative number"**

If you make x equal to 0, R will first check the if condition **(x < 0)**, and determine that it is FALSE. Then, R will evaluate the else if condition. This condition, **x==0**, is TRUE. So, in this case, R prints **“x is zero”**.

If you make x equal to 1, both the if condition and the else if condition evaluate to **FALSE**. So, R will execute the else statement and print **“x is a positive number”**.

As soon as R discovers a condition that evaluates to TRUE, R executes the corresponding code and ignores the rest.

## Additional resource

To learn more about logical operators and conditional statements, check out DataCamp's tutorial [Conditionals and Control Flow in R](https://www.datacamp.com/community/tutorials/conditionals-and-control-flow-in-r). DataCamp is a popular resource for people learning about computer programming. The tutorial is filled with useful examples of coding applications for logical operators and conditional statements (and relational operators), and offers a helpful overview of each topic and the connections between them.

### 1.

Question 1

An analyst includes the following calculation in their R programming: **midyear\_sales <- (quarter\_1\_sales + quarter\_2\_sales) - overhead\_costs** Which variable will the total from this calculation be assigned to?

1 point

**midyear\_sales**

**quarter\_1\_sales**

**quarter\_2\_sales**

**overhead\_costs**

### 2.

Question 2

An analyst is checking the value of the variable x using a logical operator, so they run the following code: **x > 35 & x < 65** Which values of x would return **TRUE** when the analyst runs the code? Select all that apply.

1 point

35

50

60

70

### 3.

Question 3

A data analyst inputs the following code in RStudio: **sales\_1 <- 100 \* sales\_2** Which of the following types of operators does the analyst use in the code? Select all the apply.

1 point

arithmetic

logical

assignment

relational

Hello, there. I have to say getting

a package delivered to you

is one of life's simple pleasures.

It doesn't matter if it's a surprise package

or something you ordered yourself.

It's exciting to open

your package to discover what's inside.

No wonder those unboxing

videos on Youtube are so popular.

Well, R has a different kind

of package that our R users can open.

These packages are units of

reproducible R code and

they make it easier to keep track of code.

They're created by members of the R community to

keep track of the R functions that they write and reuse.

These community members might then

make the packages available to other users.

It's one of the great things

about being part of this community.

Packages in R include

reusable R functions and

documentation about the functions

including how to use them.

They also contain sample datasets and

tests for checking your code

to make sure it does what you want it to do.

By default, R includes a set of packages called

base R that are available to use in

RStudio when you start your first programming session.

There's also recommended packages

that are loaded but not installed.

Before using functions from one of these packages,

you'd have to load it with

a library command like library boot, for example.

Let's find out which packages we already have in RStudio.

We'll work in our console instead of a script for

now because we're practicing

and don't need to save this code for later.

To check out our packages,

we'll just run the command

installed.packages and there's our list.

Let's focus on the package and priority columns.

The package column gives the name of the package,

like cluster or graphics.

The priority column tells us what's

needed to use functions from the package.

If you come across the word base in the priority column,

then the package is already installed and loaded.

You can use all of the functions of

that package as soon as you open RStudio.

If you find the word recommended,

then the package is installed but not loaded.

You'll also notice a list of packages in

the bottom right part of our workspace.

This list includes a brief description of each package.

To load class and other uninstalled packages,

we'll need to use the library function

followed by the name of the package.

Now, the class package has a check next to it,

so it's been successfully loaded for use.

If you want to learn even

more about your loaded packages,

you can click on their names in the Packages tab.

This opens the Help tab and

shows topics related to the package you selected.

You can also use the Help function in

your programming to call up the Help tab.

While the pre-installed packages

give you tons of useful functions,

there's even more packages that

will further expand your programming abilities.

You can find thousands of

R packages just by doing an online search.

One of the most commonly used sources

of packages is CRAN.

CRAN stands for comprehensive R archive network.

It's an online archive with R packages,

source code, manuals, and documentation.

When you start working with R,

you'll be able to do your own searches to

find packages in CRAN or elsewhere.

It's almost always easier to

just search with your favorite search engine though.

So packages are a pretty big part

of using R. They give you

most of what you need to complete

your programming throughout the data analysis process.

Who knows? You might even turn

your own code into packages for others to use.

Up next, we'll keep unpacking R packages. See you soon.

**Available R packages**

To make the most of R for your data analysis, you will need to install packages. **Packages** are units of reproducible R code that you can use to add more functionality to R. The best part is that the R community creates and shares packages so that other users can access them! In this reading, you will learn more about widely used packages and where to find them.



Packages can be found in repositories, which are collections of useful packages that are ready to install. You can find repositories on [**Bioconductor**](http://bioconductor.org/), [**R-Forge**](https://r-forge.r-project.org/), [**rOpenSci**](https://ropensci.org/), or [**GitHub**](https://github.com/), but the most commonly used repository is the Comprehensive R Archive Network or [**CRAN**](https://cran.r-project.org/). CRAN stores code and documentation so that you can install packages into your own RStudio space.

**Package documentation**

Packages will not only include the code itself, but also documentation that explains the package’s author, function, and any other packages that you will need to download. When you are using CRAN, you can find the package documentation in the DESCRIPTION file.

Check out Karl Broman's [**R Package Primer**](https://kbroman.org/pkg_primer/)to learn more.

**Choosing the right packages**

With so many packages out there, it can be hard to know which ones will be the most useful for your library or directory of installed packages. Luckily, there are some great resources out there:

* [**Tidyverse**](https://www.tidyverse.org/): the tidyverse is a collection of R packages specifically designed for working with data. It’s a standard library for most data analysts, but you can also download the packages individually.
* [**Quick list of useful R packages**](https://support.rstudio.com/hc/en-us/articles/201057987-Quick-list-of-useful-R-packages): this is RStudio Support’s list of useful packages with installation instructions and functionality descriptions.
* [**CRAN Task Views**](https://cran.r-project.org/web/views/): this is an index of CRAN packages sorted by task. You can search for the type of task you need to perform and it will pull up a page with packages related to that task for you to explore.

You will discover more packages throughout this course and as you use R more often, but this is a great starting point for building your own library.

Welcome back.

As we discussed earlier, packages are a big part of what makes R so great.

Packages offer a helpful combination of code, reusable R functions, descriptive

documentation, tests for checking operability, and sample data sets.

And for lots of data analysts,

at the top of the list of useful packages is tidyverse.

Tidyverse is actually a collection of packages in R with a common design

philosophy for data manipulation, exploration, and visualization.

Using tidyverse can help you work your way through pretty much

the entire data analysis process.

The packages in tidyverse work together naturally.

I started learning about tidyverse when I was working on a survey project.

It felt like I was stepping into a more advanced zone of R.

I understood the basics, but

now I was finding out how the tidyverse improves on the basics.

That's when I got even more excited about working in R.

I realized that the more I put into learning about the tidyverse,

the more I get out of it.

On top of that, the community support for tidyverse is strong too.

It's one of the reasons why tidyverse is considered a key part of programming for

most R users.

The principles associated with tidyverse, which you'll learn both here and

at your job, have been widely adopted by the R community.

You'll find lots of tutorials and examples related to the tidyverse online

that show you these principles and how they're applied to data analytics.

Okay, let's install the tidyverse.

You can follow along on your own, using your RStudio cloud account.

Check out the reading for more details.

Earlier, you learned how to find

Base R packages using the function install packages.

To install packages like the tidyverse that aren't in Base R,

we'll use the install packages function.

As we discussed earlier, this function calls the tidyverse and

other packages from CRAN.

Let's talk about why CRAN was created.

Since packages not in Base R are mostly made by R users,

people need a reliable way to check and validate submitted code.

CRAN makes sure any R content open to the public meets

the required quality standards.

So, if it's sourced through CRAN,

you can feel good that the package is authentic and valid.

Another major source of packages and other R content is GitHub.

Now, we'll get back to installing the tidyverse.

We'll first type install.packages.

Then, between the parentheses, we'll type tidyverse in quotes.

The quotes aren't always necessary, but

best practice is to use quotes to make sure that we are accurate.

We'll press Enter and wait for RStudio to install tidyverse.

Play video starting at :3:9 and follow transcript3:09

When we click on our packages tab, we come across a lot of new packages on the list.

That's tidyverse.

You might have noticed that none of the packages are checked off.

We need to load them first before we can use them.

But that's a mighty long list.

So, let's just load the package named tidyverse for

now, using the library function.

Play video starting at :3:37 and follow transcript3:37

The return shows that not only was tidyverse loaded,

but eight other packages were too.

It also shows a list of conflicts.

Conflicts happen when packages have functions with the same names as other

functions.

Basically, the last package loaded is the one whose functions will be used,

so we'll stick with the tidyverse functions.

But it's important to note that these messages only appear once.

So, as you get more used to R,

you'll be able to figure out if you want to use certain functions over others.

The loaded packages are ggplot2,

tibble, tidyr, readr, purrr,

dplyr, stringr, and forcats.

These packages are the core of the tidyverse

because you'll use them in almost every analysis.

All of them work together to make your data analysis smooth and efficient.

With these packages, tidyverse helps you do everything from importing and

transforming data to exploring and visualizing it.

We'll check out this core of packages soon, and

we'll use them even more as we continue working in RStudio.

If you're working on your own in R,

you can check out some of the other packages too.

The packages available in tidyverse change a lot, but you can always check for

updates by running tidyverse\_update() in your console.

You can then update the packages in a couple of ways.

If you use the update packages function, it'll update all of your packages.

That might take a while.

So, if you just want to update one package, you can use the install packages

function again with the package name as your argument in parentheses.

You should update packages regularly to make sure you've got the latest version in

your code.

Conflict notifications are just one type of message that can show up in

the console.

You might find warnings and error messages as well.

A quick search using the help tab will usually tell you what the message means

and what, if anything, you'll need to do to address it.

Coming up, we'll keep moving through the tidyverse.

You'll find out more about why tidyverse is such an integral part of R.

See you.

### 1.

Question 1

When using RStudio, what does the installed.packages() function do?

1 point

Selects the best packages to use based on an analyst’s current needs

Installs all available packages for use in an RStudio session

Creates code for analysts to use to edit their packages

Presents a list of packages currently installed in an RStudio session

### 2.

Question 2

In data analytics, what is CRAN?

1 point

An R interface that has many of the same functions as RStudio

A commonly used online archive with R packages and other R resources

A collection of packages that function together to make analysis in R more efficient

A function for finding packages to use for analysis in RStudio

### 3.

Question 3

What are ggplot2, tidyr, dplyr, and forcats all a part of?

1 point

A list of variables for use in programming in RStudio

A collection of commonly used, CRAN-based data sets

A collection of core tidyverse packages

A list of functions that clean data efficiently

Great, you're back.

Have you ever taken a tour of a famous landmark or an unfamiliar city?

It can be pretty exciting.

You get to learn all about the features of the landmark or city.

Eventually, you get to know them pretty well, and

you can share what you learned with others.

Well we're here to take a different kind of tour: a tour of the tidyverse.

For this tour, we won't be traveling anywhere special, but

we will help you learn about the exciting tidyverse features.

And once you know them a little better,

you can most definitely share what you learned with others.

For this tour we'll focus on the core packages of tidyverse

we discussed earlier: ggplot2, tidyr, readr,

dplyr, tibble, purrr, stringr and forcats.

We also learned how to install and load them in RStudio.

Once they're loaded, you won't need to do anything else with their actual packages.

They'll do their thing as you program. So what is their thing?

Well, it depends, but there's four packages that

are an essential part of the workflow for

data analysts: ggplot2, dplyr, tidyr and readr.

You'll most likely use these more often than the others.

Ggplot2 is used for data visualization, specifically plots.

With ggplot2, you can create a variety of data viz by applying

different visual properties to the data variables.

Here's an example of ggplot2 in action.

You'll have your own chance to use ggplot2 later.

Tidyr is a package used for data cleaning to make tidy data.

We covered tidy or clean data earlier, but as a quick reminder, it's data where

every part of a data table or data frame is the right type in the right place.

Tidyr works with wide and long data to make sure this happens.

Next, we have readr, which is used for importing data.

The most common function from readr is read\_csv.

This will import a CSV file into R.

A CSV file contains data separated by commas in a table format.

To accurately read a dataset with readr,

you combine the function with a column specification.

The column specification describes how each column should

be converted to the most appropriate data type.

It's good to keep in mind this isn't usually necessary because readr will

figure it out for you automatically.

We'll come across readr functions as we continue to explore R.

Now on to dplyr.

Dplyr offers a consistent set of functions that help you

complete some common data manipulation tasks.

For example, the select function picks variables based on their names,

and the filter function finds cases where certain conditions are true.

And, yes, dplyr is another package we'll get to later.

There's plenty to look forward to, so that's the fab four of the tidyverse.

They'll all make your programming in R more straightforward and efficient.

The other four packages are definitely useful, too, but

you might not use them as often.

Tibble works with data frames.

Purrr works with functions and

vectors helping make your code easier to write and more expressive.

Stringr includes functions that make it easier to work with strings.

Play video starting at :4:3 and follow transcript4:03

Forcats provides tools that solve common problems with factors.

As a quick reminder, factors store categorical data in R

where the data values are limited and

usually based on a finite group like country or year.

Using the tidyverse and its packages will help you fine-tune your analysis.

And besides tidyverse,

you also learned the fundamentals of R from variables to vectors and more.

Play video starting at :4:34 and follow transcript4:34

You explored the different operators in R and

saw how they can help you complete calculations.

You had the chance to check out pipes and

how they can make your programming more efficient.

And you unpacked packages to find out how they're a big part of what

you can do in R.

Play video starting at :4:54 and follow transcript4:54

We've covered a lot of ground in just a few videos, so

this might be a good time for you to do a little review.

You can rewatch videos and revisit any other resources that can help you get

an even better grasp of all the terms, concepts and processes that are part of R.

Looking ahead, you'll start working with data in R including a more

thorough exploration of how tidyverse impacts your process.

You'll see tibble, readr and other tidyverse packages in action.

And you'll find out how to clean and organize your data in R.

All this and more coming up. I'll see you soon.

**R resources for more help**



The R community is full of dedicated users helping each other find solutions to problems and new ways of using R. There are also a lot of great blogs where you can find tutorials and other resources.  Here are a few of them:

**Note:** due to the corporate change from R Studio to Posit, references in the following resources may have changed.

* [**Posit (RStudio)**](https://posit.co/): The best place to find help with R is in R itself! You can input ‘?’ or the help() command to search in R. You can also open the Help pane to find more R resources.
* [**Posit Blog:**](https://posit.co/blog/) Posit's blog is a great place to find information about RStudio, including company news.  You can read the most recent [**featured posts**](https://blog.rstudio.com/categories/featured/) or use the search bar and the list of categories on the left side of the page to explore specific topics you might find interesting or to search for a specific post.
* [**Stack Overflow:**](https://stackoverflow.blog/)The Stack Overflow blog posts opinions and advice from other coders. This is a great place to stay in touch with conversations happening in the community.
* [**R-Bloggers:**](https://www.r-bloggers.com/)The R-Bloggers blog has useful tutorials and news articles posted by other R users in the community.
* [**R-Bloggers' tutorials for learning R:**](https://www.r-bloggers.com/2015/12/how-to-learn-r-2/#h.y5b98o9o2h1r) This blog post from R-Bloggers compiles some basic R tutorials and also links to more advanced guides.

Hi, I'm Connor and I'm

a Marketing Analytics Manager at Google Cloud.

I was running into barriers of not being able to do

certain analysis because it was too

time-consuming with my limited technical knowledge.

So I started to teach myself things

like SQL to help me access data

through the current company's database

that I had so that I

could manipulate that data to better understand it.

I can tell you at first it is

an incredibly frustrating thing

to move through because it takes a lot

of time and effort to do something that seems

very simple or something that would be

very easy to do in spreadsheets,

but may be very difficult to do at

first as you're learning how to code.

But also one of the most fulfilling things that I've ever

done because once you're able to understand something,

it opens up an entire new realm.

Learning coding was revolutionary for my job.

I remember when I first started as an analyst,

all of the data that I used was

in spreadsheets and I had to

run analysis and create formulas to manipulate the data,

understand the data, and even analyze the data.

Now, when we started to get more and more data,

the formulas that I would have run would take hours,

and I remember at one point I spent

a few hours creating a formula and then executed it,

and it took over ten hours to run.

So I left my computer

open and let it run throughout the night,

woke up and it was still running.

Fast forward, a year later,

after I'd learned SQL and Python,

I was able to run

the same type of analysis in milliseconds.

So really understanding what

it is that you're trying to do.

Coding helps you manipulate and analyze data at a rate

that previously or without

a coding knowledge would be very difficult to do.

An important aspect of any type of script,

or when you are coding is to

structure it for overall readability.

More often than not,

you're going to be working on a team.

Now it's important that when you're writing a script,

that you understand how it works,

but also that somebody else who

you work with can also come and

understand what it is that

you are trying to do within that script.

Now, it's very important

that it not only works and is efficient,

but it is also not too verbose,

meaning that it is not overly complex.

So an important aspect of

readability is if you are looking through

your code and you realize that I've

written the same thing multiple times,

or I'm using the same logic or algorithm multiple times,

that is a point in time where you can really

consolidate your code and make it a lot more concise,

which vastly helps with readability,

vastly helps anybody who

comes and is trying to read your code,

and that includes you two weeks from now.

Because I can promise you when you start coding,

you will realize that what makes sense to you right

now may not make sense to you three weeks from now.

An important aspect for readability and

overall understanding of your code is using comments.

Comments are a way to write

something out in a standardized language like English,

and a way that somebody can understand it,

but the computer doesn't pick up as actual code.

So for explaining every line that you write

or explaining an entire section of your code in

a comment allows somebody to walk

themselves through your code and

read exactly what it is that you are

trying to accomplish with the code that you've written.

Now without comments, you are leaving it up to the person

to really follow your code and understand it themselves,

which may not be an easy task for

somebody because they may have a different way of coding,

the same thing that you are doing.

Documenting your work is an important aspect.

Documentation will explain in

depth exactly what your code is doing,

why it was built,

what is the purpose for it, and any limitations.

The last one is a rather difficult concept to

understand as you are first diving

into learning a coding language,

and that is building it for scalability as well as making it dynamic.

Now when I say building something for

scalability, what I mean is,

if you are building a specific script

of code to solve a task that you're running now,

what you want to be sure of and

answer is, will this code,

or could this code be

used in the future for something else?

Now if it is, it's important that you

make your code available to be scalable.

That means that it is efficiently run

so that if the size of the data that it is running

any manipulations on increases,

it doesn't bog down your code too much and

that it can handle large data loads as well as small.

Another aspect to that is making your code dynamic.

What that means is not hard coding

any values within your code

that don't change when they need to.

So these are just a few of the best practices and

as you continue down your path as a data analyst,

you'll pick up many, many more.

There's always more to learn,

there's always more to understand,

but this should help you in beginning

down your path to understanding coding.

Question 1

When working in R, for which part of the data analysis process do analysts use the tidyr package?

1 point

Data calculations

Data cleaning

Data security

Data visualization

### 2.

Question 2

Which tidyverse package contains a set of functions, such as select(), that help with data manipulation?

1 point

forcats

dplyr

ggplot2

readr

### 3.

Question 3

An analyst is organizing a dataset in RStudio using the following code: **arrange(filter(Storage\_1, inventory >= 40), count)** Which of the following examples is a nested function in the code?

1 point

count

arrange

filter

inventory

Glossary

Data Analytics

Terms and Definitions

A

A/B testing: The process of testing two variations of the same web page to determine which

page is more successful at attracting user traffic and generating revenue

Absolute reference: A reference within a function that is locked so that rows and columns

won’t change if the function is copied

Access control: Features such as password protection, user permissions, and encryption that

are used to protect a spreadsheet

Accuracy: The degree to which data conforms to the actual entity being measured or described

Action-oriented question: A question whose answers lead to change

Administrative metadata: Metadata that indicates the technical source of a digital asset

Agenda: A list of scheduled appointments

Aggregation: The process of collecting or gathering many separate pieces into a whole

Algorithm: A process or set of rules followed for a specific task

Aliasing: Temporarily naming a table or column in a query to make it easier to read and write

Alternative text: Text that provides an alternative to non-text content, such as images and

videos

Analytical skills: Qualities and characteristics associated with using facts to solve problems

Analytical thinking: The process of identifying and defining a problem, then solving it by using

data in an organized, step-by-step manner

Annotation: Text that briefly explains data or helps focus the audience on a particular aspect of

the data in a visualization

Area chart: A data visualization that uses individual data points for a changing variable

connected by a continuous line with a filled in area underneath

Argument (R): Information needed by a function in R in order to run

Array: A collection of values in spreadsheet cells

Assignment operator (R): An operator used to assign values to variables and vectors

Attribute: A characteristic or quality of data used to label a column in a table

Audio file: Digitized audio storage usually in an MP3, AAC, or other compressed format

AVERAGE: A spreadsheet function that returns an average of the values from a selected range

AVERAGEIF: A spreadsheet function that returns the average of all cell values from a given

range that meet a specified condition

B

Bad data source: A data source that is not reliable, original, comprehensive, current, and cited

(ROCCC)

Balance: The design principle of creating aesthetic appeal and clarity in a data visualization by

evenly distributing visual elements

Bar graph: A data visualization that uses size to contrast and compare two or more values

Bias: A conscious or subconscious preference in favor of or against a person, group of people,

or thing

Big data: Large, complex datasets typically involving long periods of time, which enable data

analysts to address far-reaching business problems

Boolean data: A data type with only two possible values, usually true or false

Borders: Lines that can be added around two or more cells on a spreadsheet

Box plot: A data visualization that displays the distribution of values along an x-axis

Bubble chart: A data visualization that displays individual data points as bubbles, comparing

numeric values by their relative size

Bullet graph: A data visualization that displays data as a horizontal bar chart moving toward a

desired value

Business metric: A standard of measurement used to solve a business task

Business task: The question or problem data analysis resolves for a business

C

C#: An object-oriented programming language used to create games and mobile apps in the

.NET open source developer platform

C++: An extension of the C programming language that is used to create console games, such

as those for Xbox

Calculated field: A new field within a pivot table that carries out certain calculations based on

the values of other fields

Calculus: A branch of mathematics that involves the study of rates of change and the changes

between values that are related by a function

CASE: A SQL statement that returns records that meet conditions by including an if/then

statement in a query

Case study: A common way for employers to assess job skills and gain insight into how a

candidate approaches common data-related challenges

CAST: A SQL function that converts data from one datatype to another

Causation: When an action directly leads to an outcome, such as a cause-effect relationship

Cell reference: A cell or a range of cells in a worksheet typically used in formulas and functions

Changelog: A file containing a chronologically ordered list of modifications made to a project

Channel: A visual aspect or variable that represents characteristics of the data in a visualization

Chart: A graphical representation of data from a worksheet

Circle view: A data visualization that shows comparative strength in data

Clean data: Data that is complete, correct, and relevant to the problem being solved

Cloud: A place to keep data online, rather than a computer hard drive

Cluster: A collection of data points on a data visualization with similar values

COALESCE: A SQL function that returns non-null values in a list

Coding: The process of writing instructions to a computer in the syntax of a specific

programming language

Column chart: A data visualization that uses individual data points for a changing variable,

represented as vertical columns

Combo chart: A data visualization that combines more than one visualization type

Compatibility: How well two or more datasets are able to work together

Completeness: The degree to which data contains all desired components or measures

Computer programming: The process of giving instructions to a computer in order to perform

an action or set of actions

CONCAT: A SQL function that adds strings together to create new text strings that can be used

as unique keys

CONCATENATE: A spreadsheet function that joins together two or more text strings

Conditional formatting: A spreadsheet tool that changes how cells appear when values meet

specific conditions

Conditional statement: A declaration that if a certain condition holds, then a certain event must

take place

Confidence interval: A range of values that conveys how likely a statistical estimate reflects

the population

Confidence level: The probability that a sample size accurately reflects the greater population

Confirmation bias: The tendency to search for or interpret information in a way that confirms

pre-existing beliefs

Consent: The aspect of data ethics that presumes an individual’s right to know how and why

their personal data will be used before agreeing to provide it

Consistency: The degree to which data is repeatable from different points of entry or collection

Context: The condition in which something exists or happens

Continuous data: Data that is measured and can have almost any numeric value

CONVERT: A SQL function that changes the unit of measurement of a value in data

Cookie: A small file stored on a computer that contains information about its users

Correlation: The measure of the degree to which two variables change in relationship to each

other

COUNT: A spreadsheet function that counts the number of cells within a range that meet a

specified condition

COUNTA: A spreadsheet function that counts the total number of values within a range that

meet specified criteria

COUNTIF: A spreadsheet function that returns the number of cells within a range that match a

specified value

COUNT DISTINCT: A SQL function that only returns the distinct values in a specified range

CRAN (Comprehensive R Archive Network) (R): An online archive with R packages, source

code, manuals, and documentation

CREATE TABLE: A SQL clause that adds a temporary table to a database that can be used by

multiple people

Cross-field validation: A process that ensures certain conditions for multiple data fields are

satisfied

CSS (Cascading Style Sheets): A programming language used for web page design that

controls graphic elements and page presentation

CSV (comma-separated values) file: A delimited text file that uses a comma to separate

values

Currency: The aspect of data ethics that presumes individuals should be aware of financial

transactions resulting from the use of their personal data and the scale of those transactions

D

Dashboard: A tool that monitors live, incoming data

Data: A collection of facts

Data aggregation: The process of gathering data from multiple sources and combining it into a

single, summarized collection

Data analysis: The collection, transformation, and organization of data in order to draw

conclusions, make predictions, and drive informed decision-making

Data analysis process: The six phases of ask, prepare, process, analyze, share, and act

whose purpose is to gain insights that drive informed decision-making

Data analyst: Someone who collects, transforms, and organizes data in order to draw

conclusions, make predictions, and drive informed decision-making

Data analytics: The science of data

Data anonymization: The process of protecting people's private or sensitive data by eliminating

identifying information

Data bias: When a preference in favor of or against a person, group of people, or thing

systematically skews data analysis results in a certain direction

Data blending: A Tableau method that combines data from multiple data sources

Data composition: The process of combining the individual parts in a visualization and

displaying them together as a whole

Data constraints: The criteria that determine whether a piece of a data is clean and valid

Data design: How information is organized

Data-driven decision-making: Using facts to guide business strategy

Data ecosystem: The various elements that interact with one another in order to produce,

manage, store, organize, analyze, and share data

Data element: A piece of information in a dataset

Data engineer: A professional who transforms data into a useful format for analysis and gives it

a reliable infrastructure

Data ethics: Well-founded standards of right and wrong that dictate how data is collected,

shared, and used

Data frame: A collection of columns containing data, similar to a spreadsheet or SQL table

Data governance: A process for ensuring the formal management of a company’s data assets

Data-inspired decision-making: Exploring different data sources to find out what they have in

common

Data integrity: The accuracy, completeness, consistency, and trustworthiness of data

throughout its life cycle

Data interoperability: The ability to integrate data from multiple sources and a key factor

leading to the successful use of open data among companies and governments

Data life cycle: The sequence of stages that data experiences, which include plan, capture,

manage, analyze, archive, and destroy

Data manipulation: The process of changing data to make it more organized and easier to

read

Data mapping: The process of matching fields from one data source to another

Data merging: The process of combining two or more datasets into a single dataset

Data model: A tool for organizing data elements and how they relate to one another

Data privacy: Preserving a data subject’s information any time a data transaction occurs

Data range: Numerical values that fall between predefined maximum and minimum values

Data replication: The process of storing data in multiple locations

Data science: A field of study that uses raw data to create new ways of modeling and

understanding the unknown

Data security: Protecting data from unauthorized access or corruption by adopting safety

measures

Data storytelling: Communicating the meaning of a dataset with visuals and a narrative that

are customized for an audience

Data strategy: The management of the people, processes, and tools used in data analysis

Data structure: A format for organizing and storing data

Data transfer: The process of copying data from a storage device to computer memory or from

one computer to another

Data type: An attribute that describes a piece of data based on its values, its programming

language, or the operations it can perform

Data validation: A tool for checking the accuracy and quality of data

Data validation process: The process of checking and rechecking the quality of data so that it

is complete, accurate, secure and consistent

Data visualization: The graphical representation of data

Data warehousing specialist: A professional who develops processes and procedures to

effectively store and organize data

Database: A collection of data stored in a computer system

Dataset: A collection of data that can be manipulated or analyzed as one unit

DATEDIF: A spreadsheet function that calculates the number of days, months, or years

between two dates

Decision tree: A tool that helps analysts make decisions about critical features of a visualization

Delimiter: A character that indicates the beginning or end of a data item

Density map: A data visualization that represents concentrations, with color representing the

number or frequency of data points in a given area on a map

Descriptive metadata: Metadata that describes a piece of data and can be used to identify it at

a later point in time

Design thinking: A process used to solve complex problems in a user-centric way

Digital photo: An electronic or computer-based image usually in BMP or JPG format

Dirty data: Data that is incomplete, incorrect, or irrelevant to the problem to be solved

Discrete data: Data that is counted and has a limited number of values

DISTINCT: A keyword that is added to a SQL SELECT statement to retrieve only non-duplicate

entries

Distribution graph: A data visualization that displays the frequency of various outcomes in a

sample

Diverging color palette: A color theme that displays two ranges of data values using two

different hues, with color intensity representing the magnitude of the values

Donut chart: A data visualization where segments of a ring represent data values adding up to

a whole

dplyr (R): An R package in Tidyverse that offers a consistent set of functions to complete

common data-manipulation tasks

DROP TABLE: A SQL clause that removes a temporary table from a database

Duplicate data: Any record that inadvertently shares data with another record

Dynamic visualizations: Data visualizations that are interactive or change over time

E

Emphasis: The design principle of arranging visual elements to focus the audience’s attention

on important information in a data visualization

Engagement: Capturing and holding someone’s interest and attention during a data

presentation

Equation: A calculation that involves addition, subtraction, multiplication, or division (also called

a math expression)

Estimated response rate: The average number of people who typically complete a survey

Ethics: Well-founded standards of right and wrong that prescribe what humans ought to do,

usually in terms of rights, obligations, benefits to society, fairness, or specific virtues

Experimenter bias: The tendency for different people to observe things differently (Refer to

Observer bias)

External data: Data that lives, and is generated, outside of an organization

F

Factor (R): An object that stores categorical data where the data values are limited and usually

based on a finite group, such as country or year

Fairness: A quality of data analysis that does not create or reinforce bias

Field: A single piece of information from a row or column of a spreadsheet; in a data table,

typically a column in the table

Field length: A tool for determining how many characters can be keyed into a spreadsheet field

Fill handle: A box in the lower-right-hand corner of a selected spreadsheet cell that can be

dragged through neighboring cells in order to continue an instruction

Filled map: A data visualization that colors areas in a map based on measurements or

dimensions

Filtering: The process of showing only the data that meets a specified criteria while hiding the

rest

Find and replace: A tool that finds a specified search term and replaces it with something else

First-party data: Data collected by an individual or group using their own resources

Float: A number that contains a decimal

Foreign key: A field within a database table that is a primary key in another table (Refer to

primary key)

Formula: A set of instructions used to perform a calculation using the data in a spreadsheet

Framework: The context a presentation needs to create logical connections that tie back to the

business task and metrics

FROM: The section of a query that indicates from which table(s) to extract the data

Function: A preset command that automatically performs a specified process or task using the

data in a spreadsheet

Function (R): A body of reusable code for performing specific tasks in R

G

Gantt chart: A data visualization that displays the duration of events or activities on a timeline

Gap analysis: A method for examining and evaluating the current state of a process in order to

identify opportunities for improvement in the future

Gauge chart: A data visualization that shows a single result within a progressive range of

values

General Data Protection Regulation of the European Union (GDPR): Policy-making body in

the European Union created to help protect people and their data

Geolocation: The geographical location of a person or device by means of digital information

ggplot2 (R): An R package in Tidyverse that creates a variety of data visualizations by applying

different visual properties to the data variables in R

Good data source: A data source that is reliable, original, comprehensive, current, and cited

(ROCCC)

GROUP BY: A SQL clause that groups rows that have the same values from a table into

summary rows

H

HAVING: A SQL clause that adds a filter to a query instead of the underlying table that can only

be used with aggregate functions

Header: The first row in a spreadsheet that labels the type of data in each column

Headline: Text at the top of a visualization that communicates the data being presented

Heat map: A data visualization that uses color contrast to compare categories in a dataset

Highlight table: A data visualization that uses conditional formatting and color on a table

Histogram: A data visualization that shows how often data values fall into certain ranges

HTML5: A programming language that provides structure for web pages and connects to

hosting platforms

Hypothesis: A theory that one might try to prove or disprove with data

Hypothesis testing: A process to determine if a survey or experiment has meaningful results

I

IDE (Integrated Development Environment): A software application that brings together all

the tools a data analyst may want to use in a single place

Incomplete data: Data that is missing important fields

Inconsistent data: Data that uses different formats to represent the same thing

Incorrect/inaccurate data: Data that is complete but inaccurate

INNER JOIN : A SQL function that returns records with matching values in both tables

Inner query: A SQL subquery that is inside of another SQL statement

Internal data: Data that lives within a company’s own systems

Interpretation bias: The tendency to interpret ambiguous situations in a positive or negative

way

J

Java: A programming language widely used to create enterprise web applications that can run

on multiple clients

JOIN: A SQL function that is used to combine rows from two or more tables based on a related

column

K

L

Label: Text in a visualization that identifies a value or describes a scale

Leading question: A question that steers people toward a certain response

LEFT: A function that returns a set number of characters from the left side of a text string

LEFT JOIN: A SQL function that will return all the records from the left table and only the

matching records from the right table

Legend: A tool that identifies the meaning of various elements in a data visualization

LEN: A function that returns the length of a text string by counting the number of characters it

contains

Length: The number of characters in a text string

Library: A directory containing all of a data analyst’s installed packages

LIMIT: A SQL clause that specifies the maximum number of records returned in a query

Line graph: A data visualization that uses one or more lines to display shifts or changes in data

over time

List: A vector whose elements can be of any type

Live data: Data that is automatically updated

Logical operator: An operator that returns a logical data type

Long data: A dataset in which each row is one time point per subject, so each subject has data

in multiple rows

M

Mandatory: A data value that cannot be left blank or empty

Map: A data visualization that organizes data geographically

Margin of error: The maximum amount that sample results are expected to differ from those of

the actual population

Mark: A visual object in a data visualization such as a point, line, or shape

MATCH: A spreadsheet function used to locate the position of a specific lookup value

Math expression: A calculation that involves addition, subtraction, multiplication, or division

(also called an equation)

Math function: A function that is used as part of a mathematical formula

Matrix: A two-dimensional collection of data elements with rows and columns

MAX: A function that returns the largest numeric value from a range of cells

MAXIFS: A spreadsheet function that returns the maximum value from a given range that meets

a specified condition

McCandless Method: A method for presenting data visualizations that moves from general to

specific information

Measurable question: A question whose answers can be quantified and assessed

Mental model: A data analyst’s thought process and approach to a problem

Mentor: Someone who shares knowledge, skills, and experience to help another grow both

professionally and personally

Merger: An agreement that unites two organizations into a single new one

Metadata: Data about data

Metadata repository: A database created to store metadata

Metric: A single, quantifiable type of data that is used for measurement

Metric goal: A measurable goal set by a company and evaluated using metrics

MID: A function that returns a segment from the middle of a text string

MIN: A spreadsheet function that returns the smallest numeric value from a range of cells

MINIFS: A spreadsheet function that returns the minimum value from a given range that meets

a specified condition

Modulo: An operator (%) that returns the remainder when one number is divided by another

Movement: The design principle of arranging visual elements to guide the audience’s eyes from

one part of a data visualization to another

N

Naming conventions: Consistent guidelines that describe the content, creation date, and

version of a file in its name

Narrative: (Refer to story)

Nested: Code that performs a particular function and is contained within code that performs a

broader function

Nested function: A function that is completely contained within another function

Networking: Building relationships by meeting people both in person and online

Nominal data: A type of qualitative data that is categorized without a set order

Normalized database: A database in which only related data is stored in each table

Notebook: An interactive, editable programming environment for creating data reports and

showcasing data skills

Null: An indication that a value does not exist in a dataset

O

Observation: The attributes that describe a piece of data contained in a row of a table

Observer bias: The tendency for different people to observe things differently (also called

experimenter bias)

Open data: Data that is available to the public

Open-source: Code that is freely available and may be modified and shared by the people who

use it

Openness: The aspect of data ethics that promotes the free access, usage, and sharing of data

Operator: A symbol that names the operation or calculation to be performed

ORDER BY: A SQL clause that sorts results returned in a query

Order of operations: Using parentheses to group together spreadsheet values in order to

clarify the order in which operations should be performed

Ordinal data: Qualitative data with a set order or scale

Outdated data: Any data that has been superseded by newer and more accurate information

OUTER JOIN: A SQL function that combines RIGHT and LEFT JOIN to return all matching

records in both tables

Outer query: A SQL statement containing a subquery

Ownership: The aspect of data ethics that presumes individuals own the raw data they provide

and have primary control over its usage, processing, and sharing

P

Package (R): A unit of reproducible R code

Packed bubble chart: A data visualization that displays data in clustered circles

Pattern: The design principle of using similar visual elements to demonstrate trends and

relationships in a data visualization

PHP (Hypertext Preprocessor): A programming language for web application development

Pie chart: A data visualization that uses segments of a circle to represent the proportions of

each data category compared to the whole

Pipe (R): A tool in R for expressing a sequence of multiple operations, represented with “%>%”

Pivot chart: A chart created from the fields in a pivot table

Pivot table: A data summarization tool used to sort, reorganize, group, count, total, or average

data

Pixel: In digital imaging, a small area of illumination on a display screen that, when combined

with other adjacent areas, forms a digital image

Population: In data analytics, all possible data values in a dataset

Portfolio: A collection of materials that can be shared with potential employers

Pre-attentive attributes: The elements of a data visualization that an audience recognizes

automatically without conscious effort

Primary key: An identifier in a database that references a column in which each value is unique

(Refer to foreign key)

Problem domain: The area of analysis that encompasses every activity affecting or affected by

a problem

Problem types: The various problems that data analysts encounter, including categorizing

things, discovering connections, finding patterns, identifying themes, making predictions, and

spotting something unusual

Profit margin: A percentage that indicates how many cents of profit has been generated for

each dollar of sale

Programming language: A system of words and symbols used to write instructions that

computers follow

Proportion: The design principle of using the relative size and arrangement of visual elements

to demonstrate information in a data visualization

Python: A general-purpose programming language

Q

Qualitative data: A subjective and explanatory measure of a quality or characteristic

Quantitative data: A specific and objective measure, such as a number, quantity, or range

Query: A request for data or information from a database

Query language: A computer programming language used to communicate with a database

R

R: A programming language used for statistical analysis, visualization, and other data analysis

Random sampling: A way of selecting a sample from a population so that every possible type

of the sample has an equal chance of being chosen

Range: A collection of two or more cells in a spreadsheet

Ranking: A system to position values of a dataset within a scale of achievement or status

readr (R): An R package in Tidyverse used for importing data

Record: A collection of related data in a data table, usually synonymous with row

Redundancy: When the same piece of data is stored in two or more places

Reframing: The process of restating a problem or challenge, then redirecting it toward a

potential resolution

Regular expression (RegEx): A rule that says the values in a table must match a prescribed

pattern

Relational database: A database that contains a series of tables that can be connected to form

relationships

Relativity: The process of considering observations in relation or proportion to something else

Relevant question: A question that has significance to the problem to be solved

Remove duplicates: A spreadsheet tool that automatically searches for and eliminates

duplicate entries from a spreadsheet

Repetition: The design principle of repeating visual elements to demonstrate meaning in a data

visualization

Report: A static collection of data periodically given to stakeholders

Return on investment (ROI): A formula that uses the metrics of investment and profit to

evaluate the success of an investment

Revenue: The total amount of income generated by the sale of goods or services

Rhythm: The design principle of creating movement and flow in a data visualization to engage

an audience

RIGHT: A function that returns a set number of characters from the right side of a text string

RIGHT JOIN: A SQL function that will return all records from the right table and only the

matching records from the left

Root cause: The reason why a problem occurs

ROUND: A SQL function that returns a number rounded to a certain number of decimal places.

Ruby: An object-oriented programming language for web application development

S

Sample: In data analytics, a segment of a population that is representative of the entire

population

Sampling bias: Overrepresenting or underrepresenting certain members of a population as a

result of working with a sample that is not representative of the population as a whole

Scatterplot: A data visualization that represents relationships between different variables with

individual data points without a connecting line

Schema: A way of describing how something, such as data, is organized

Scope of work (SOW): An agreed-upon outline of the tasks to be performed during a project

Second-party data: Data collected by a group directly from its audience and then sold

SELECT: The section of a query that indicates from which column(s) to extract the data

SELECT INTO: A SQL clause that copies data from one table into a temporary table without

adding the new table to the database

Small data: Small, specific data points typically involving a short period of time, which are

useful for making day-to-day decisions

SMART methodology: A tool for determining a question’s effectiveness based on whether it is

specific, measurable, action-oriented, relevant, and time-bound

Social media: Websites and applications through which users create and share content or

participate in social networking

Soft skills: Nontechnical traits and behaviors that relate to how people work

Sort range: A spreadsheet menu function that sorts a specified range and preserves the cells

outside the range

Sort sheet: A spreadsheet menu function that sorts all data by the ranking of a specific sorted

column and keeps data together across rows

Sorting: The process of arranging data into a meaningful order to make it easier to understand,

analyze, and visualize

Specific question: A question that is simple, significant, and focused on a single topic or a few

closely related ideas

SPLIT: A spreadsheet function that divides text around a specified character and puts each

fragment into a new, separate cell

Sponsor: A professional advocate who is committed to moving forward the career of another

Spotlightling: Scanning through data to quickly identify the most important insights

Spreadsheet: A digital worksheet

SQL: (Refer to Structured Query Language)

Stakeholders: People who invest time and resources into a project and are interested in its

outcome

Static data: Data that doesn’t change once it has been recorded

Static visualization: A data visualization that does not change over time unless it is edited

Statistical power: The probability that a test of significance will recognize an effect that is

present

Statistical significance: The probability that sample results are not due to random chance

Statistics: The study of how to collect, analyze, summarize, and present data

Story: The narrative of a data presentation that makes it meaningful and interesting

String data type: A sequence of characters and punctuation that contains textual information

(also called text data type)

Structural metadata: Metadata that indicates how a piece of data is organized and whether it is

part of one or more than one data collection

Structured data: Data organized in a certain format such as rows and columns

Structured Query Language: A computer programming language used to communicate with a

database

Structured thinking: The process of recognizing the current problem or situation, organizing

available information, revealing gaps and opportunities, and identifying options

Subquery: A SQL query that is nested inside a larger query

SUBSTR: A SQL function that extracts a substring from a string variable

Substring: A subset of a text string

Subtitle: Text that supports a headline by adding context and description

SUM: A spreadsheet function that adds the values of a selected range of cells

SUMIF: A spreadsheet function that adds numeric data based on one condition

Summary table: A table used to summarize statistical information about data

SUMPRODUCT: A function that multiplies arrays and returns the sum of those products

Swift: A programming language for macOS, iOS, watchOS, and tvOS

Symbol map: A data visualization that displays a mark over a given longitude and latitude

Syntax: The predetermined structure of a language that includes all required words, symbols,

and punctuation, as well as their proper placement

T

Tableau: A business intelligence and analytics platform that helps people visualize, understand,

and make decisions with data

Technical mindset: The ability to break things down into smaller steps or pieces and work with

them in an orderly and logical way

Temporary table: A database table that is created and exists temporarily on a database server

Text data type: A sequence of characters and punctuation that contains textual information

(also called string data type)

Text string: A group of characters within a cell, most often composed of letters

Third-party data: Data provided from outside sources who didn’t collect it directly

tidyr (R): An R package in Tidyverse used for data cleaning to make tidy data

Tidyverse (R): A system of packages in R with a common design philosophy for data

manipulation, exploration, and visualization

Time-bound question: A question that specifies a timeframe to be studied

Transaction transparency: The aspect of data ethics that presumes all data-processing

activities and algorithms should be explainable and understood by the individual who provides

the data

Transferable skills: Skills and qualities that can transfer from one job or industry to another

TRIM: A function that removes leading, trailing, and repeated spaces in data

Turnover rate: The rate at which employees voluntarily leave a company

Typecasting: Converting data from one type to another

U

Unbiased sampling: When the sample of the population being measured is representative of

the population as a whole

Underscores: Lines used to underline words and connect text characters

Unfair question: A question that makes assumptions or is difficult to answer honestly

Unique: A value that can’t have a duplicate

United States Census Bureau: An agency in the U.S. Department of Commerce that serves as

the nation’s leading provider of quality data about its people and economy

Unity: The design principle of using visual elements that complement each other to create

aesthetic appeal and clarity in a data visualization

Unstructured data: Data that is not organized in any easily identifiable manner

V

Validity: The degree to which data conforms to constraints when it is input, collected, or created

VALUE: A spreadsheet function that converts a text string that represents a number to a

numeric value

Variable (R): A representation of a value in R that can be stored for later use

Variety: The design principle of using different kinds of visual elements in a data visualization to

engage an audience

Vector (R): A group of data elements of the same type stored in a one-dimensional sequence in

R

Verification: A process to confirm that a data-cleaning effort was well executed and the

resulting data is accurate and reliable

Video file: A collection of images, audio files, and other data usually encoded in a compressed

format such as MP4, MV4, MOV, AVI, or FLV

Vignette (R): Documentation for an R package that describes the problem the package is

designed to solve, explains how its functions can be used, and lists any dependencies on other

packages

Visual form: The appearance of a data visualization that gives it structure and aesthetic appeal

Visualization: (Refer to Data visualization)

VLOOKUP: A spreadsheet function that vertically searches for a certain value in a column to

return a corresponding piece of information

W

WHERE: The section of a query that specifies criteria that the requested data must meet

Wide data: A dataset in which every data subject has a single row with multiple columns to hold

the values of various attributes of the subject

WITH: A SQL clause that creates a temporary table that can be queried multiple times

World Health Organization: An organization whose primary role is to direct and coordinate

international health within the United Nations system

X

X-axis: The horizontal line of a graph usually placed at the bottom, which is often used to

represent time scales and discrete categories

Y

Y-axis: The vertical line of a graph usually placed to the left, which is often used to represent

frequencies and other numerical variables

Z

### 1.

Question 1

Fill in the blank: When creating a variable for use in R, your variable name should begin with \_\_\_\_\_.

1 point

a letter

a number

an underscore

an operator

### 2.

Question 2

You want to create a vector with the values 43, 56, 12 in that exact order. After specifying the variable, what R code chunk lets you create the vector?

1 point

**v(43, 56, 12)**

**c(43, 56, 12)**

**c(12, 56, 43)**

**v(12, 56, 43)**

### 3.

Question 3

An analyst runs code to convert string data into a date/time data type that results in the following: **“2020-07-10”**. Which of the following are examples of code that would lead to this return? Select all that apply.

1 point

**dmy(“7-10-2020”)**

**ymd(20200710)**

**myd(2020, July 10)**

**mdy(“July 10th, 2020”)**

### 4.

Question 4

A data analyst inputs the following code in RStudio: **sales\_1 <- (3500.00 \* 12)** Which of the following types of operators does the analyst use in the code? Select all that apply.

1 point

Logical

Assignment

Arithmetic

Relational

### 5.

Question 5

Which of the following variables have names that follow widely accepted naming convention rules? Select all that apply.

1 point

**\*totalplums\***

**total\_plums**

**plum\_total\_1**

**1\_plum\_total**

### 6.

Question 6

Which of the following are included in R packages? Select all that apply.

1 point

Tests for checking your code

Naming conventions for R variable names

Sample datasets

Reusable R functions

### 7.

Question 7

Packages installed in RStudio are called from CRAN. CRAN is an online archive with R packages and other R-related resources.

1 point

True

False

### 8.

Question 8

A data analyst wants to take a data frame named people and filter the data where age is 10, arranged by height, and grouped by gender. Which code snippet would perform those operations in the specified order?

1 point

**people %>%**

**filter(age == 10) +**

**arrange(height) +**

**group\_by(gender)**

**group\_by( arrange( filter( people, age == 10 ), height ), gender )**

**people %>%**

**group\_by(gender) %>%**

**arrange(height) %>%**

**filter(age == 10)**

**filter( arrange( group\_by( people, gender ), height ) , age == 10 )**

# More about tibbles

In this reading, you will learn about tibbles, which are a super useful tool for organizing data in R. You will get a review of what tibbles are, how they differ from standard data frames, and how to create them in R.

## Tibbles



Tibbles are a little different from standard data frames. A data frame is a collection of columns, like a spreadsheet or a SQL table. Tibbles are like streamlined data frames that are automatically set to pull up only the first 10 rows of a dataset, and only as many columns as can fit on the screen. This is really useful when you’re working with large sets of data. Unlike data frames, tibbles never change the names of your variables, or the data types of your inputs. Overall, you can make more changes to data frames, but tibbles are easier to use. The tibble package is part of the core tidyverse. So, if you’ve already installed the tidyverse, you have what you need to start working with tibbles.

### **Creating tibbles**

Now, let’s go through an example of how to create a tibble in R. You can use the pre-loaded diamonds dataset that you’re familiar with from earlier videos. As a reminder, the diamonds dataset includes information about different diamond qualities, like carat, cut, color, clarity, and more.

You can load the dataset with the **data()** function using the following code:

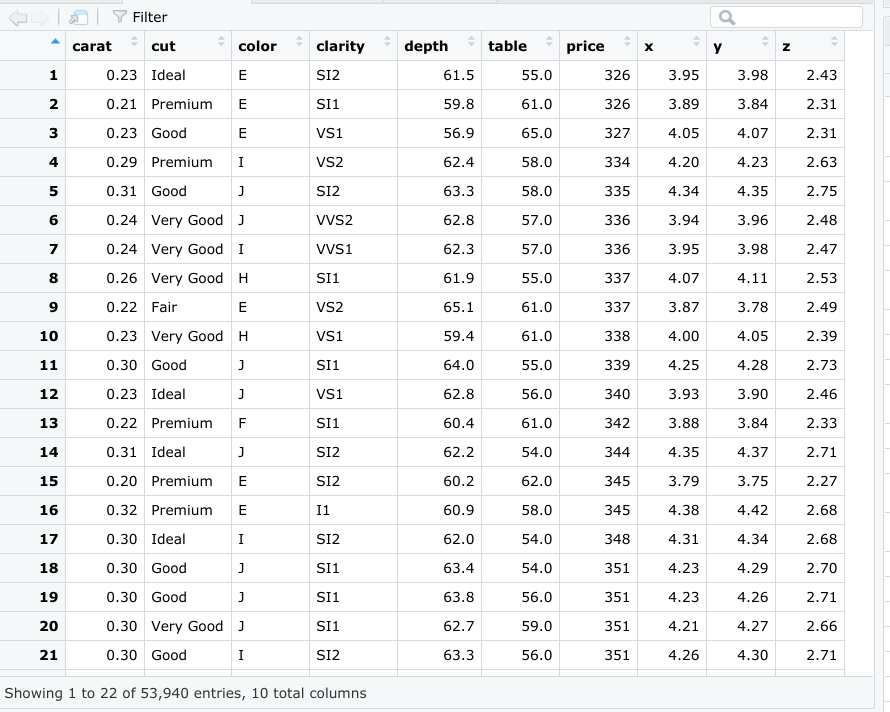
**library(tidyverse)**

**data(diamonds)**

Then, let’s add the data frame to our data viewer in RStudio with the **View()** function.

**View(diamonds)**

The dataset has 10 columns and thousands of rows. This image displays part of the data frame:

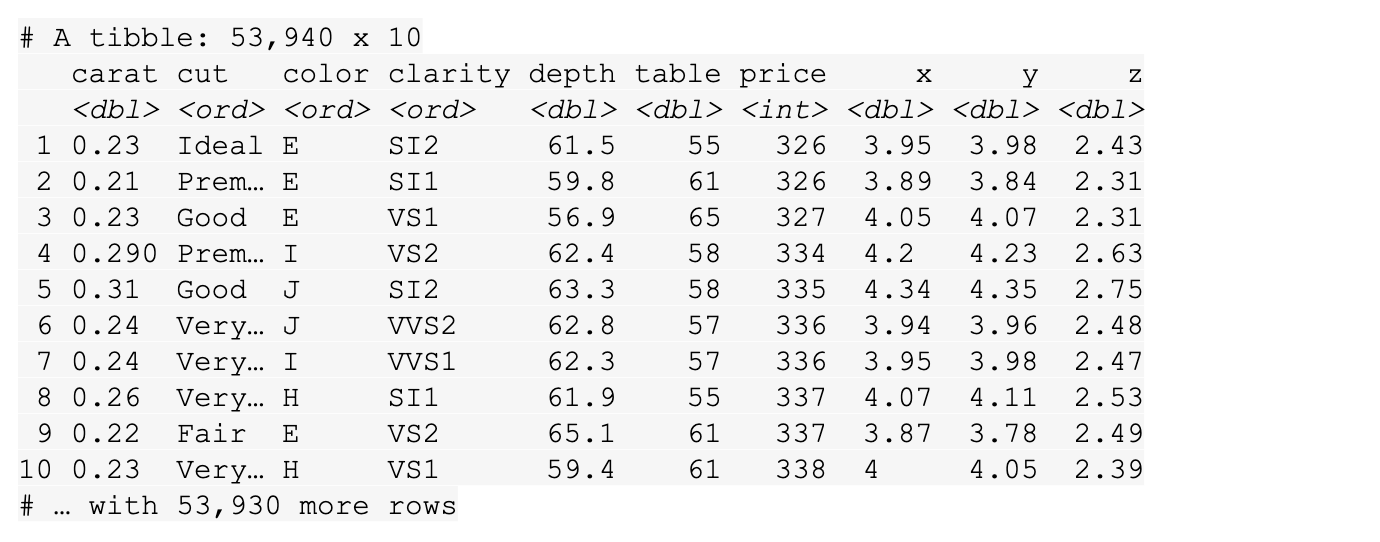


Now let’s create a tibble from the same dataset. You can create a tibble from existing data with the **as\_tibble()** function. Indicate what data you’d like to use in the parentheses of the function. In this case, you will put the word “diamonds."

**as\_tibble(diamonds)**

### **Results**

When you run the function, you get a tibble of the diamonds dataset.



While RStudio’s built-in data frame tool returns thousands of rows in the diamonds dataset, the tibble only returns the first 10 rows in a neatly organized table. That makes it easier to view and print.

## Additional resources

For more information on tibbles, check out the following resources:

* The entry for [Tibble](https://tibble.tidyverse.org/) in the tidyverse documentation summarizes what a tibble is and how it works in R code. If you want a quick overview of the essentials, this is the place to go.
* The [Tidy chapter](https://rstudio-education.github.io/tidyverse-cookbook/tidy.html) in "A Tidyverse Cookbook" is a great resource if you want to learn more about how to work with tibbles using R code. The chapter explores a variety of R functions that can help you create and transform tibbles to organize and tidy your data.

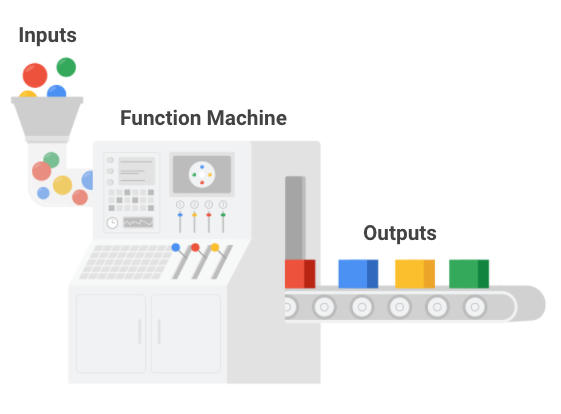
# Data-import basics

You can save this reading for future reference. Feel free to download a PDF version of this reading below:

[Data import basics.pdf](https://d3c33hcgiwev3.cloudfront.net/sVF77vMISbONi27VPuNHsA_dbe37e131fce45c39c00445461b93df1_Data-import-basics.pdf?Expires=1706918400&Signature=VkwZOapnngWz24Kk4nFzC8erDQ6X8yKnV-HdGMOi9kULOh1JgtajCpCEaaa2G76YaHanWMK~5FXtxpGKfScIGqoVtiZ6v3qYsOetOuZh2sXv2Nxg20cVs~6N0IohNpSzzXt-~XpCttDcsmsOQeSgJKSlklVoqiLIg3X4VSVIJDw_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[PDF File](https://d3c33hcgiwev3.cloudfront.net/sVF77vMISbONi27VPuNHsA_dbe37e131fce45c39c00445461b93df1_Data-import-basics.pdf?Expires=1706918400&Signature=VkwZOapnngWz24Kk4nFzC8erDQ6X8yKnV-HdGMOi9kULOh1JgtajCpCEaaa2G76YaHanWMK~5FXtxpGKfScIGqoVtiZ6v3qYsOetOuZh2sXv2Nxg20cVs~6N0IohNpSzzXt-~XpCttDcsmsOQeSgJKSlklVoqiLIg3X4VSVIJDw_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

## ****The data() function****

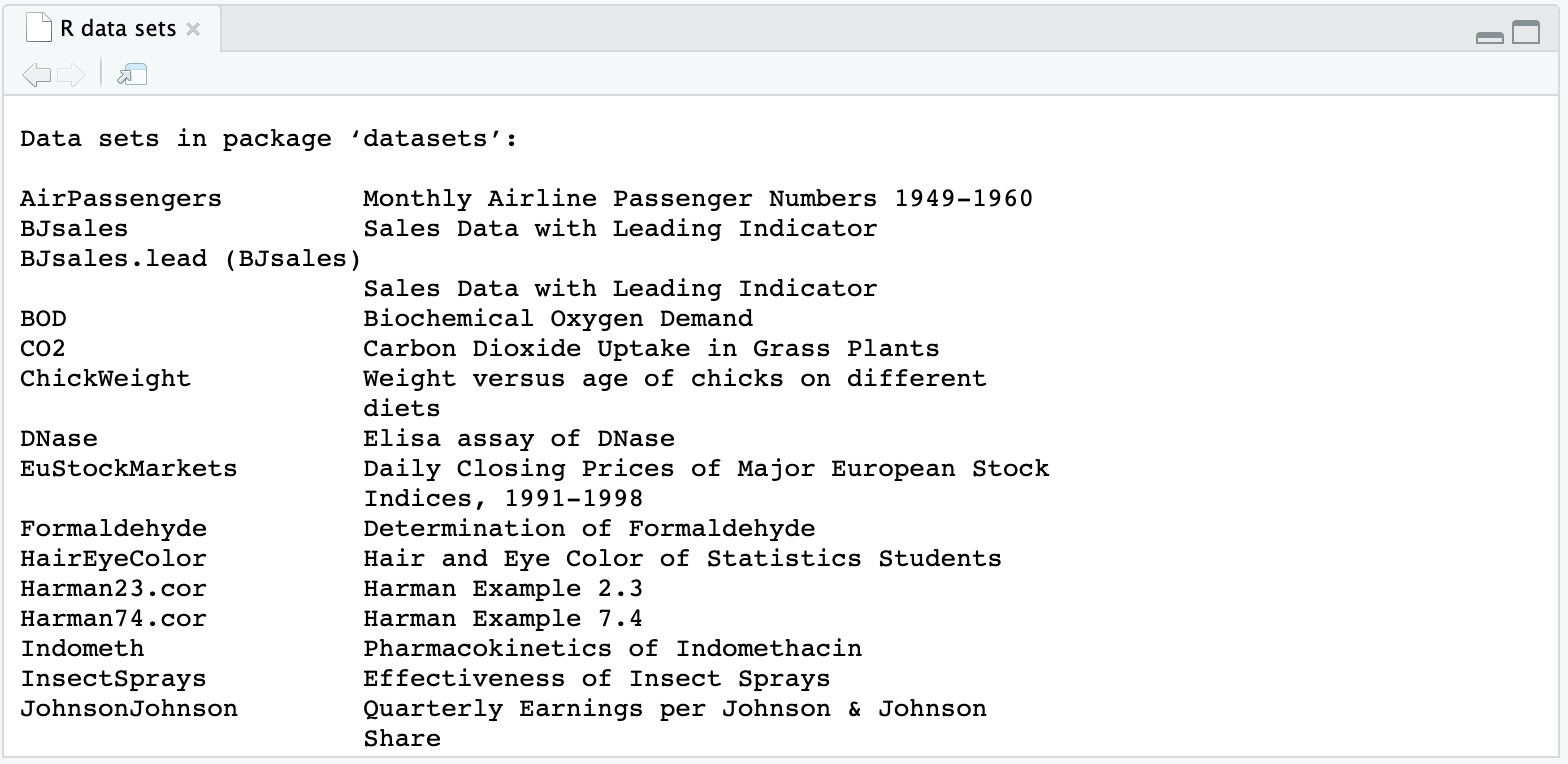


The default installation of R comes with a number of preloaded datasets that you can practice with. This is a great way to develop your R skills and learn about some important data analysis functions. Plus, many online resources and tutorials use these sample datasets to teach coding concepts in R.

You can use the **data()** function to load these datasets in R. If you run the data function without an argument, R will display a list of the available datasets.

**data()**

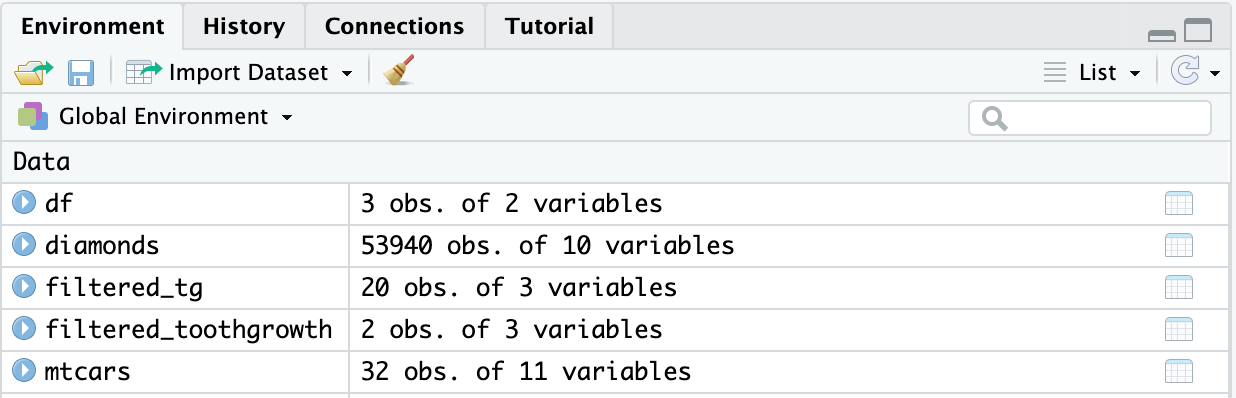
This includes the list of preloaded datasets from the datasets package.



If you want to load a specific dataset, just enter its name in the parentheses of the data() function. For example, let’s load the mtcars dataset, which has information about cars that have been featured in past issues of Motor Trend magazine.

**data(mtcars)**

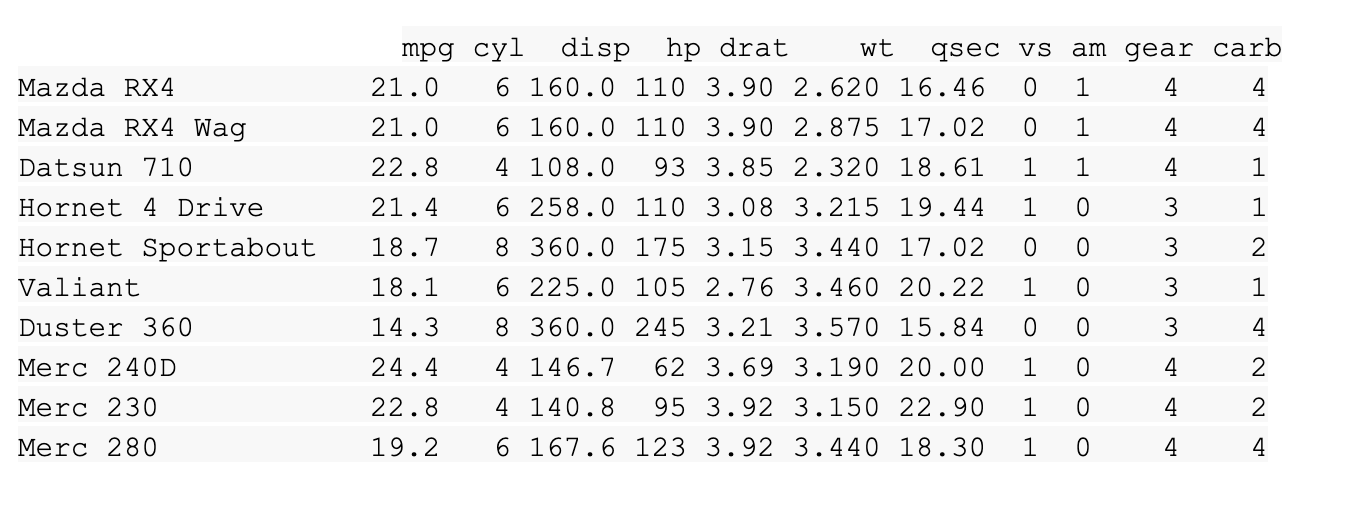
When you run the function, R will load the dataset. The dataset will also appear in the Environment pane of your RStudio. The Environment pane displays the names of the data objects, such as data frames and variables, that you have in your current workspace. In this image, mtcars appears in the fifth row of the pane. R tells us that it contains 32 observations and 11 variables.



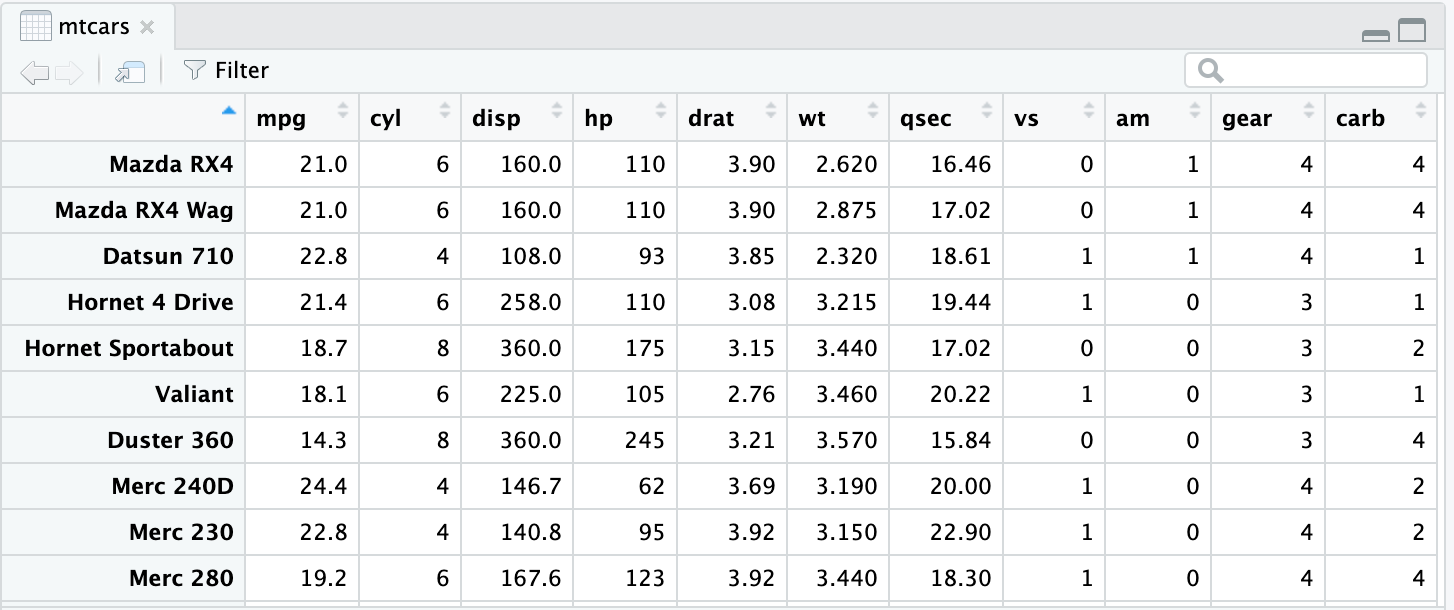
Now that the dataset is loaded, you can get a preview of it in the R console pane. Just type its name...

**mtcars**

...and then press ctrl (or cmnd) and enter.



You can also display the dataset by clicking directly on the name of the dataset in the Environment pane. So, if you click on **mtcars** in the Environment pane, R automatically runs the View() function and displays the dataset in the RStudio data viewer.



Try experimenting with other datasets in the list if you want some more practice.

## ****The readr package****

In addition to using R’s built-in datasets, it is also helpful to import data from other sources to use for practice or analysis. The readr package in R is a great tool for reading rectangular data. Rectangular data is data that fits nicely inside a rectangle of rows and columns, with each column referring to a single variable and each row referring to a single observation.

Here are some examples of file types that store rectangular data:

* **.csv** **(comma separated values)**: a .csv file is a plain text file that contains a list of data. They mostly use commas to separate (or delimit) data, but sometimes they use other characters, like semicolons.
* **.tsv (tab separated values)**: a .tsv file stores a data table in which the columns of data are separated by tabs. For example, a database table or spreadsheet data.
* **.fwf** **(fixed width files)**: a .fwf file has a specific format that allows for the saving of textual data in an organized fashion.
* **.log:** a .log file is a computer-generated file that records events from operating systems and other software programs.

Base R also has functions for reading files, but the equivalent functions in readr are typically much faster. They also produce tibbles, which are easy to use and read.

The readr package is part of the core tidyverse. So, if you’ve already installed the tidyverse, you have what you need to start working with readr. If not, you can install the tidyverse now.

### **readr functions**

The goal of readr is to provide a fast and friendly way to read rectangular data. readr supports several read\_ functions. Each function refers to a specific file format.

* **read\_csv()**: comma-separated values (.csv) files
* **read\_tsv()**: tab-separated values files
* **read\_delim()**: general delimited files
* **read\_fwf()**: fixed-width files
* **read\_table()**: tabular files where columns are separated by white-space
* **read\_log()**: web log files

These functions all have similar syntax, so once you learn how to use one of them, you can apply your knowledge to the others. This reading will focus on the read\_csv() function, since .csv files are one of the most common forms of data storage and you will work with them frequently.

In most cases, these functions will work automatically: you supply the path to a file, run the function, and you get a tibble that displays the data in the file. Behind the scenes, readr parses the overall file and specifies how each column should be converted from a character vector to the most appropriate data type.

### **Reading a .csv file with readr**

The readr package comes with some sample files from built-in datasets that you can use for example code. To list the sample files, you can run the readr\_example() function with no arguments.

**readr\_example()**

**[1] "challenge.csv"     "epa78.txt"         "example.log"**

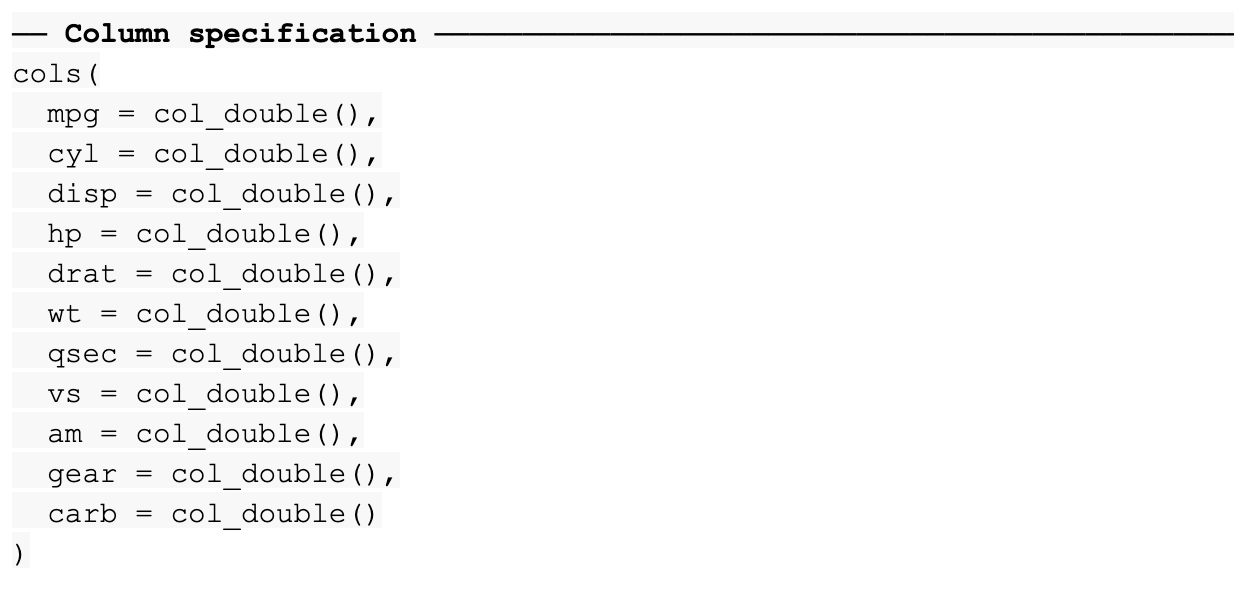
**[4] "fwf-sample.txt"    "massey-rating.txt" "mtcars.csv"**

**[7] "mtcars.csv.bz2"    "mtcars.csv.zip"**

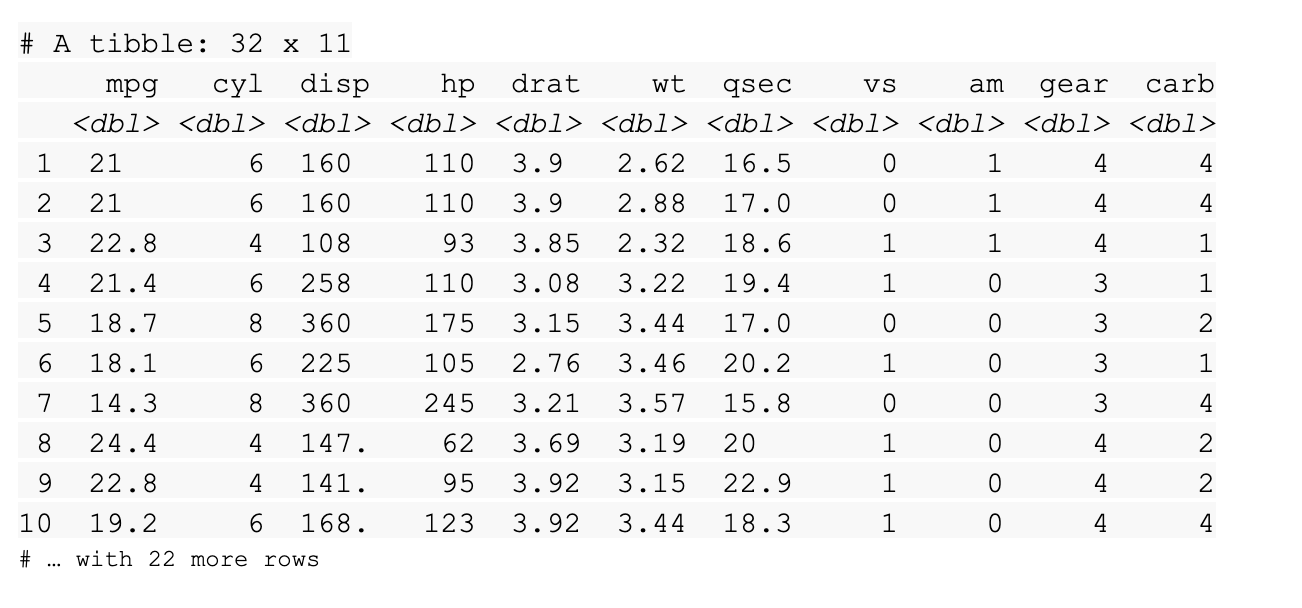
The **“mtcars.csv”** file refers to the mtcars dataset that was mentioned earlier. Let’s use the **read\_csv()** function to read the **“mtcars.csv”** file, as an example. In the parentheses, you need to supply the path to the file. In this case, it’s **“readr\_example(“mtcars.csv”)**.

**read\_csv(readr\_example("mtcars.csv"))**

When you run the function, R prints out a column specification that gives the name and type of each column.



R also prints a tibble.



------------------------------------------------------------------------------------------------------

## Optional: the readxl package

To import spreadsheet data into R, you can use the readxl package. The readxl package makes it easy to transfer data from Excel into R. Readxl supports both the legacy .xls file format and the modern xml-based .xlsx file format.

The readxl package is part of the tidyverse but is not a core tidyverse package, so you need to load readxl in R by using the library() function.

**library(readxl)**

### **Reading an .xlsx file with readxl**

Like the readr package, readxl comes with some sample files from built-in datasets that you can use for practice. You can run the code **readxl\_example()** to see the list.

You can use the **read\_excel()** function to read a spreadsheet file just like you used read\_csv() function to read a  .csv file. The code for reading the example file **“type-me.xlsx”** includes the path to the file in the parentheses of the function.

**read\_excel(readxl\_example("type-me.xlsx"))**

You can use the [excel\_sheets()](https://readxl.tidyverse.org/reference/excel_sheets.html) function to list the names of the individual sheets.

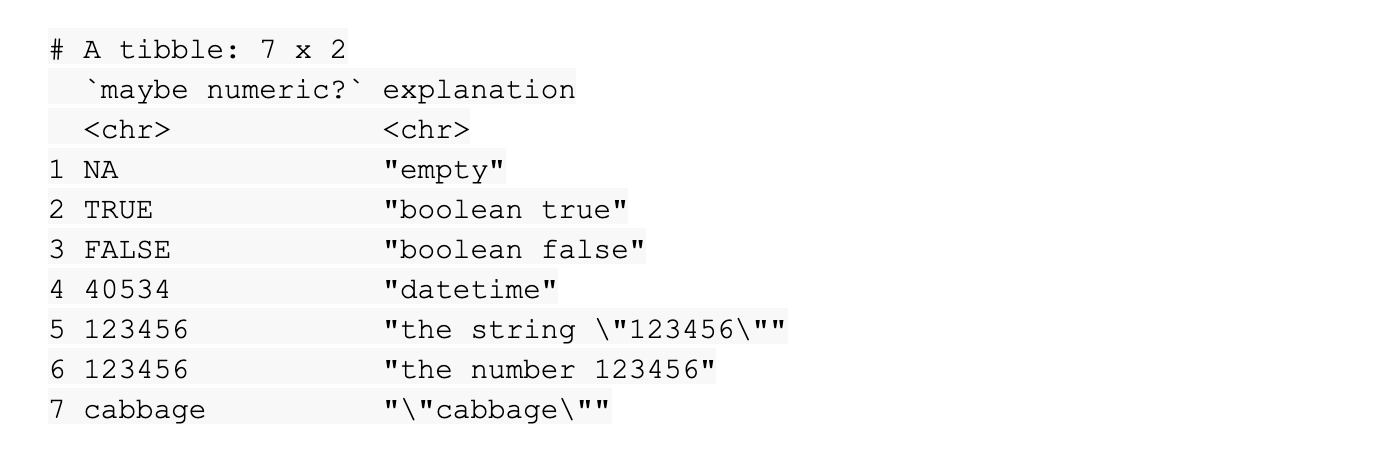
**excel\_sheets(readxl\_example("type-me.xlsx"))**

**[1] "logical\_coercion" "numeric\_coercion" "date\_coercion" "text\_coercion"**

You can also specify a sheet by name or number.  Just type **“sheet =”** followed by the name or number of the sheet. For example, you can use the sheet named **“numeric\_coercion”** from the list above.

**read\_excel(readxl\_example("type-me.xlsx"), sheet = "numeric\_coercion")**

When you run the function, R returns a tibble of the sheet.



## Additional resources

* If you want to learn how to use readr functions to work with more complex files, check out the [Data Import chapter](https://r4ds.had.co.nz/data-import.html) of the R for Data Science book. It explores some of the common issues you might encounter when reading files, and how to use readr to manage those issues.
* The [readxl](https://readxl.tidyverse.org/) entry in the tidyverse documentation gives a good overview of the basic functions in readxl, provides a detailed explanation of how the package operates and the coding concepts behind them, and offers links to other useful resources.
* The R "datasets" package contains lots of useful preloaded datasets. Check out [The R Datasets Package](https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/00Index.html) for a list. The list includes links to detailed descriptions of each dataset.

### .

Question 1

Which of the following are best practices for creating data frames? Select all that apply.

1 point

Columns should be named

Each column should contain the same number of data items

Rows should be named

All data stored should be the same type

### 2.

Question 2

Why are tibbles a useful variation of data frames?

1 point

Tibble can change the data type of inputs

Tibbles make printing easier

Tibbles can create row names

Tibbles make changing the names of variables easier.

### 3.

Question 3

Tidy data is a way of standardizing the organization of data within R.

1 point

True

False

### 4.

Question 4

Which R function can be used to make changes to a data frame?

1 point

colnames()

head()

str()

mutate()

Hi again. Now that we've got

a little more experience with the data frames,

we can start doing some interesting things like clean,

standardize, manipulate, and visualize data.

We'll go through some common tasks

that you'll perform as a data analyst.

But we're just scratching

the surface of what you might want to do in

R. We'll start with

the basics and learn how to clean up our columns.

There will be a reading with

a handy list you can refer to afterwards too.

Let's install the Here,

Skimr and Janitor packages now.

We'll go ahead and open our console.

First, we'll add the Here package.

This package makes referencing files easier.

To install it, we'll just write install.packages.

Then in the parentheses,

we'll put Here and RStudio will install it.

After we install it,

we'll also need to load it using library.

Next, we'll install Skimr and Janitor.

As a quick reminder,

these packages simplify data cleaning tasks.

They're both really useful

and do slightly different things.

The Skimr package makes summarizing data

really easy and let's you skim through it more quickly.

We'll install it now.

Play video starting at :1:32 and follow transcript1:32

The Janitor package has functions for cleaning data.

Play video starting at :1:39 and follow transcript1:39

After it's done installing,

we'll still need to load it.

Finally, we want to make sure

the dplyr package is

loaded since we are going to be

using some of its features.

Play video starting at :1:55 and follow transcript1:55

There, now we've got

all the packages we need for basic data cleaning.

Now, let's load some data in.

Later, when you're practicing with your own data,

you can use read to grab a file.

For example, if you had a CSV

you wanted to load, you could write,

read underscore CSV,

and input the file name in the parentheses.

This is where the Here package comes in handy.

Be sure to install and load

the Here package before trying to save CSV files.

For now, we'll load a really fun package to

practice with, the palmer penguin package.

This is a dataset we've used before,

but just as a quick reminder,

the palmer penguin data has lots of information

about three penguin species in the Palmer Archipelago,

including size measurements, clutch sizes,

and blood isotope ratios.

Who doesn't love penguins?

First, we'll install the package.

We'll type install.packages and input palmerpenguins.

Play video starting at :3:9 and follow transcript3:09

Then remember to load it by using the library function.

Now that we've got this data loaded into our library,

we can try some cleaning functions on our columns.

There's a few different functions that we can

use to get summaries of our data frame.

Skim without charts, glimpse, head, and select.

The skim without charts function gives us

a pretty comprehensive summary

of a dataset. Let's try it out.

When we run this,

we get a lot of info back.

First, it gives us a summary with the name

of the dataset and the number of rows and columns.

It also gives us the column types and

a summary of the different data types

contained in the data frame.

Or we could use Glimpse to get

a really quick idea of what's in this dataset.

When we run this command,

it'll show us a summary of the data.

There's 344 rows and eight columns.

We have species, island, measurements for bills,

which are basically beaks and flippers,

the penguins' body mass in grams,

the sex, and finally,

the year the data was recorded.

We can also use Head to get a preview of

the column names and the first few rows of this data set.

Having the column names summarized like

this will make it easier to clean them up.

We can use select to specify

certain columns or to

exclude columns we don't need right now.

Let's say we only need to check the species column.

We can input penguins,

then a pipe to indicate we'll add

another command, and our select.

We'll jump back into

an R script because it will be easier to see.

Play video starting at :5:3 and follow transcript5:03

Now we have just the species column,

or maybe we want everything except the species column.

We'll put minus species instead of species,

and now we have every column but species.

The select statement is useful for pulling

just a subset of variables from a large dataset.

This lets you focus on specific groups of variables.

There's a lot of other select functions that

build on this that we'll cover later on.

Now that we know our column names,

we've got a better idea of what we might want to change.

The rename function makes it easy to change column names.

Starting with the penguin data,

we'll type rename and change the name

of our island column to island underscore new.

Play video starting at :5:57 and follow transcript5:57

Now, looking at our column names,

we can see the column name has changed.

Or let's say we want to change

our columns so that they're

spelled and formatted correctly.

In spreadsheet programs, as long as

our column names are meaningful, they're fine.

But since we have to type

the column names over and over in R,

we need them to be consistent.

Similar to the rename function,

the rename\_with() function can

change column names to be more consistent.

For example, maybe we want all

of our column names to be in uppercase.

We can use the rename\_with() function to do that.

This will automatically make our column names uppercase.

But since variable names are usually lowercase,

we'll use the "To lower" option to change it back.

Play video starting at :6:54 and follow transcript6:54

The clean names function in the Janitor package will

automatically make sure that

the column names are unique and consistent.

Let's try the clean names function on our penguins data.

Play video starting at :7:15 and follow transcript7:15

This ensures that there's only characters,

numbers, and underscores in the names.

Now you know some functions for

cleaning columns in your datasets.

Try practicing them on your

own with the palmer penguins data.

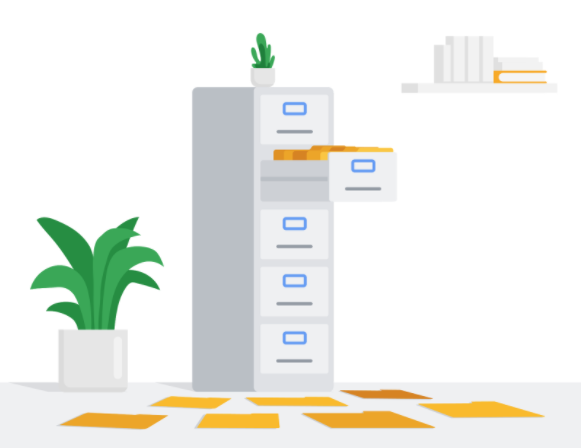
Once you're comfortable with these functions,

we'll learn even more about data

cleaning in R. See you soon.

# File-naming conventions

An important part of cleaning data is making sure that all of your files are accurately named. Although individual preferences will vary a bit, most analysts generally agree that file names should be accurate, consistent, and easy to read. This reading provides some general guidelines for you to follow when naming or renaming your data files.



## What’s in a (file)name?

When you first start working with R (or any other programming language, analysis tool, or platform, for that matter), you or your company should establish naming conventions for your files. This helps ensure that anyone reviewing your analysis–yourself included–can quickly and easily find what they need. Next are some helpful “do’s” and “don’ts” to keep in mind when naming your files.

### **Do**

* Keep your filenames to a reasonable length
* Use underscores and hyphens for readability
* Start or end your filename with a letter or number
* Use a standard date format when applicable; example: YYYY-MM-DD
* Use filenames for related files that work well with default ordering; example: in chronological order, or logical order using numbers first

| **Examples of good filenames** |
| --- |
| 2020-04-10\_march-attendance.R |
| 2021\_03\_20\_new\_customer\_ids.csv |
| 01\_data-sales.html |
| 02\_data-sales.html |

### **Don't**

* Use unnecessary additional characters in filenames
* Use spaces or “illegal” characters; examples: &, %, #, <, or >
* Start or end your filename with a symbol
* Use incomplete or inconsistent date formats; example: M-D-YY
* Use filenames for related files that do not work well with default ordering; examples: a random system of numbers or date formats, or using letters first

| **Examples of filenames to avoid** |
| --- |
| 4102020marchattendance<workinprogress>.R |
| \_20210320\*newcustomeridsforfebonly.csv |
| firstfile\_for\_datasales/1-25-2020.html |
| secondfile\_for\_datasales/2-5-2020.html |

## Additional resources

These resources include more info about some of the file naming standards discussed here, and provide additional insights into best practices.

* [**How to name files**](https://speakerdeck.com/jennybc/how-to-name-files): this resource from Speaker Deck is a playful take on file naming. It includes several slides with tips and examples for how to accurately name lots of different types of files. You will learn why filenames should be both machine readable and human readable.
* [**File naming and structure**](https://www.tikar.or.id/?q=node/205): this resource from the Princeton University Library provides an easy-to-scan list of best practices, considerations, and examples for developing file naming conventions.

# More on R operators

You might remember that an **operator** is a symbol that identifies the type of operation or calculation to be performed in a formula. In an earlier video, you learned how to use the assignment and arithmetic operators to assign variables and perform calculations. In this reading, you will review a detailed summary of the main types of operators in R, and learn how to use specific operators in R code.

## Operators

 In R, there are four main types of operators:

1. Arithmetic
2. Relational
3. Logical
4. Assignment

Review the specific operators in each category and check out some examples of how to use them in R code.

### **Arithmetic operators**

**Arithmetic operators** let you perform basic math operations like addition, subtraction, multiplication, and division.

The table below summarizes the different arithmetic operators in R. The examples used in the table are based on the creation of two variables: : x equals 2 and y equals 5. Note that you use the assignment operator to store these values:

**x <- 2**

**y <- 5**

| **Operator** | **Description** | **Example Code** | **Result/ Output** |
| --- | --- | --- | --- |
| + | Addition | x + y | [1] 7 |
| - | Subtraction | x - y | [1] -3 |
| \* | Multiplication | x \* y | [1] 10 |
| / | Division | x / y | [1] 0.4 |
| %% | Modulus (returns the remainder after division) | y %% x | [1] 1 |
| %/% | Integer division (returns an integer value after division) | y%/% x | [1] 2 |
| ^ | Exponent | y ^ x | [1]25 |

**Relational operators**

**Relational operators,** also known as comparators, allow you to compare values. Relational operators identify how one R object relates to another—like whether an object is less than, equal to, or greater than another object. The output for relational operators is either TRUE or FALSE (which is a logical data type, or boolean).

The table below summarizes the six relational operators in R. The examples used in the table are based on the creation of two variables: x equals 2 and y equals 5. Note that you use the assignment operator to store these values.

**x <- 2**

**y <- 5**

If you perform calculations with each operator, you get the following results. In this case, the output is boolean: TRUE or FALSE. Note that the [1] that appears before each output is used to represent how output is displayed in RStudio.

| **Operator** | **Description** | **Example Code** | **Result/Output** |
| --- | --- | --- | --- |
| < | Less than | x < y | [1] TRUE |
| > | Greater than | x > y | [1] FALSE |
| <= | Less than or equal to | x < = 2 | [1] TRUE |
| >= | Greater than or equal to | y >= 10 | [1] FALSE |
| == | Equal to | y == 5 | [1] TRUE |
| != | Not equal to | x != 2 | [1] FALSE |

**Logical operators**

**Logical operators** allow you to combine logical values. Logical operators return a logical data type or boolean (TRUE or FALSE)**.** You encountered logical operators in an earlier reading, [Logical operators and conditional statements](https://www.coursera.org/learn/data-analysis-r/supplement/I39VT/logical-operators-and-conditional-statements), but here is a quick refresher.

The table below summarizes the logical operators in R.

| **Operator** | **Description** |
| --- | --- |
| & | Element-wise logical AND |
| && | Logical AND |
| | | Element-wise logical OR |
| || | Logical OR |
| ! | Logical NOT |

Next, check out some examples of how logical operators work in R code.

**Element-wise logical AND (&) and OR (|)**

You can illustrate logical AND (&) and OR (|) by comparing numerical values. Create a variable x that is equal to 10.

**x <- 10**

The AND operator returns TRUE only if both individual values are TRUE.

**x > 2 & x < 12**

[1] TRUE

10 is greater than 2 and 10 is less than 12. So, the operation evaluates to **TRUE**.

The OR operator (|) works in a similar way to the AND operator (&). The main difference is that just one of the values of the OR operation needs to be TRUE for the entire OR operation to evaluate to TRUE. Only if both values are FALSE will the entire OR operation evaluate to **FALSE**.

Now try an example with the same variable **(x <- 10)**:

**x > 2 | x < 8**

**[1] TRUE**

10 is greater than 2, but 10 is not less than 8. But since at least one of the values (10>2) is TRUE, the OR operation evaluates to **TRUE**.

**Logical NOT (!)**

The NOT operator simply negates the logical value, and evaluates to its opposite. In R, zero is considered FALSE and all non-zero numbers are considered TRUE.

For example, apply the NOT operator to your variable **(x <- 10)**:

**!(x < 15)**

**[1] FALSE**

The NOT operation evaluates to **FALSE** because it takes the opposite logical value of the statement **x < 15**, which is TRUE (10 is less than 15).

### **Assignment operators**

**Assignment operators** let you assign values to variables.

In many scripting programming languages you can just use the equal sign (=) to assign a variable. For R, the best practice is to use the arrow assignment (<-). Technically, the single arrow assignment can be used in the left or right direction. But the rightward assignment is not generally used in R code.

You can also use the double arrow assignment, known as a scoping assignment. But the scoping assignment is for advanced R users, so you won’t learn about it in this reading.

The table below summarizes the assignment operators and example code in R. Notice that the output for each variable is its assigned value.

| **Operator** | **Description** | **Example Code (after the sample code below, typing x will generate the output in the next column)** | **Result/ Output** |
| --- | --- | --- | --- |
| <- | Leftwards assignment | x <- 2 | [1] 2 |
| <<- | Leftwards assignment | x <<- 7 | [1] 7 |
| = | Leftwards assignment | x = 9 | [1] 9 |
| -> | Rightwards assignment | 11 -> x | [1] 11 |
| ->> | Rightwards assignment | 21 ->> x | [1] 21 |

The operators you learned about in this reading are a great foundation for using operators in R.

## Additional resource

Check out the article about [R Operators](https://r-coder.com/operators-r/#Assignment_operators_in_R) on the R Coder website for a comprehensive guide to the different types of operators in R. The article includes lots of useful coding examples, and information about miscellaneous operators, the infix operator, and the pipe operator.

Hey, great to have you back.

We've learned how to create data frames and perform some basic cleaning functions.

Now it's time to start getting organized in R. Coming up I'll teach you some

functions that will help you organize and filter your data.

These functions look a little different in R

than in the other tools we've used so far.

But the reason we use them is still the same.

If we don't organize our data we can't turn information into knowledge. Organizing

our data and comparing different metrics from that data helps us find new insights.

In other words it makes our data useful.

To help us do this, we'll use the arrange, group by and filter functions.

Let's start by sorting our data.

We'll keep working with the palmer penguins data from earlier.

In case you don't remember, refer to the link below.

We'll also need to load the right packages.

All the packages we'll need are part of the core tidyverse.

So let's load the core tidyverse now.

Play video starting at :1:11 and follow transcript1:11

We can use the arrange function to choose

which variable we want to sort by, for

example let's say we want to sort our penguin data by bill

length. We'll type in a range and our column name.

And when we execute this command it will return a tibble with data

sorted by bill lengths.

It's currently in ascending order.

If we want to sort it in descending order we just add a minus sign before the column name.

Play video starting at :1:48 and follow transcript1:48

And now,

the longest penguin bill is first.

Now it's important to remember this data is just in our console to

save this as a data frame will start by naming it.

Then we'll input the function we used to arrange the previous version of

the penguins data.

Play video starting at :2:11 and follow transcript2:11

When we execute this it'll save a new data frame and

we can use view penguins2 to add it to our data.

This lets you save cleaned data without losing information from the original

dataset.

You can also sort by data using the group by function.

Group by is usually combined with other functions.

For example, we might want to group by a certain column and

then perform an operation on those groups. With our penguin data,

we can group by island and

then use the summarize function to get the mean bill length.

We checked out the summarize function when we introduced piping. Basically

the summarize function lets us get high level information about our penguin data.

So let's build our group by statement first.

Play video starting at :3:8 and follow transcript3:08

We're not interested in NA values so

we can leave those out using the drop underscore NA argument.

This addresses any missing values in our dataset.

It's important to be careful when using drop\_na.

It's useful doing a group level summary statistic like this but

it will remove rows from the data.

Now let's use summarize. We'll title the summary

column mean bill length millimeters.

And then we'll build the mean statement.

Play video starting at :3:49 and follow transcript3:49

And when we run this we get a data frame with the three islands and

the mean bill length of the penguins living there.

We can get other summaries too, for example, if we want to know the maximum

bill length, we can write a similar function and replace mean with max.

Play video starting at :4:24 and follow transcript4:24

And now we know that

the penguin with

the longest bill lived

on Biscoe island.

Both group by and summarize can perform multiple tasks.

For example, we could group by island and species and

then summarize to calculate both the mean and max.

To do that, we can write a similar command. We'll put species and

island in our group by and drop any missing values.

Play video starting at :5:2 and follow transcript5:02

And then we can add a summarize

statement with a max and

mean calculation.

Play video starting at :5:24 and follow transcript5:24

And when we run this,

we have both groupings and

Play video starting at :5:28 and follow transcript5:28

the max and mean.

Thanks to piping we can combine all of these cleaning and

transforming tasks into one code chunk.

Finally we can filter results using the filter function.

Let's say we only want data on Adelie penguins.

We'll start with the dataset we're using and then add the filter.

Play video starting at :5:56 and follow transcript5:56

You might notice that we're using two equal signs here; that's on purpose.

The double equal sign means exactly equal to in R. And

now we have a data frame that only contains data on Adelie penguins.

This lets us narrow down our analysis if we need to.

Being able to clean and organized data is a key step in the data analysis process

and knowing the right tool for the job is an important skill for a data analyst.

R makes wrangling data easier and gives you a lot of

functionality across the different stages of the data analysis process.

Now that we've cleaned our data, we can get ready to transform it. Coming up,

we'll learn how to use the separate, unite and mutate functions and

how to use them to transform our data in R. See you next time.

# Optional: Manually create a data frame

Coming up in the next video, you are going to learn how to transform data in R. The video will be using manually entered data instead of a data set from an R package.

If you would like to follow along with the video in your own RStudio console, you can copy and paste the following code to enter the data and create a data frame:

**id <- c(1:10)**

**name <- c("John Mendes", "Rob Stewart", "Rachel Abrahamson", "Christy Hickman", "Johnson Harper", "Candace Miller", "Carlson Landy", "Pansy Jordan", "Darius Berry", "Claudia Garcia")**

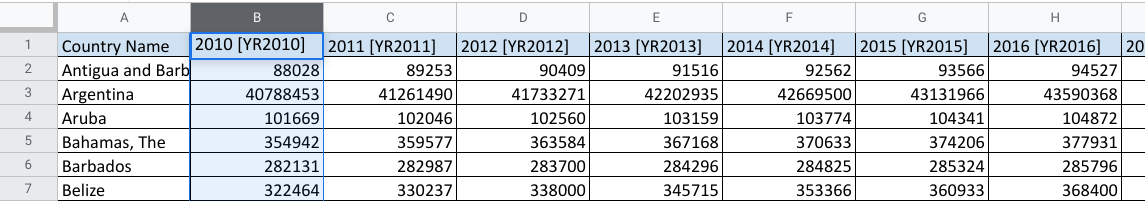
**job\_title <- c("Professional", "Programmer", "Management", "Clerical", "Developer", "Programmer", "Management", "Clerical", "Developer", "Programmer")**

**employee <- data.frame(id, name, job\_title)**

Then, you can perform the functions from the video in your own console to practice transforming and cleaning data in R! Practicing along with the video will help you explore how these functions are supposed to work while also executing them yourself. You can also use this data frame to practice more after the video.

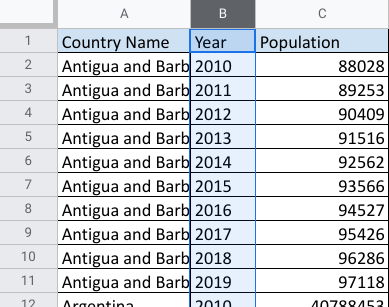
**Wide to long with tidyr**

When organizing or tidying your data using R, you might need to convert wide data to long data or long to wide. Recall that this is what data looks like in a wide format spreadsheet:



**Wide data** has observations across several columns. Each column contains data from a different condition of the variable. In this example the columns are different years.

Now check out the same data in a long format:



To review what you already learned about the difference, **long data** has all the observations in a single column, and the variable conditions are placed into separate rows.

**The pivot\_longer and pivot\_wider functions**



There are compelling reasons to use both formats. But as an analyst, it is important to know how to tidy data when you need to. In R, you may have a data frame in a wide format that has several variables and conditions for each variable. It might feel a bit messy.

That’s where pivot\_longer()comes in. As part of the tidyr package, you can use this R function to lengthen the data in a data frame by increasing the number of rows and decreasing the number of columns. Similarly, if you want to convert your data to have more columns and fewer rows, you would use the pivot\_wider() function.

**Additional resources**

To learn more about these two functions and how to apply them in your R programming, check out these resources:

* [**Pivoting**](https://tidyr.tidyverse.org/articles/pivot.html): Consider this a starting point for tidying data through wide and long conversions. This web page is taken directly from tidyr package information at [**tidyverse.org**](https://www.tidyverse.org/). It explores the components of the pivot\_longer and pivot\_wider functions using specific details, examples, and definitions.
* [**CleanItUp 5: R-Ladies Sydney: Wide to Long to Wide to…PIVOT**](https://rladiessydney.org/courses/ryouwithme/02-cleanitup-5/): This resource gives you additional details about the pivot\_longer and pivot\_wider functions. The examples provided use interesting datasets to illustrate how to convert data from wide to long and back to wide.
* [**Plotting multiple variables**](https://scc.ms.unimelb.edu.au/resources-list/simple-r-scripts-for-analysis/r-scripts)[**:**](https://www.datamentor.io/r-programming/saving-plot/) This resource explains how to visualize wide and long data, with ggplot2 to help tidy it. The focus is on using pivot\_longer to restructure data and make similar plots of a number of variables at once. You can apply what you learn from the other resources here for a broader understanding of the pivot functions.

**Working with biased data**

Every data analyst will encounter an element of bias at some point in the data analysis process. That’s why it’s so important to understand how to identify and manage biased data whenever possible. You might recall we explored bias in detail in Course 3 of this program. In this reading, you will read a real-life example of an analyst who discovered bias in their data, and learn how they used R to address it.

**Addressing biased data with R**



This scenario was shared by a quantitative analyst who collects data from people all over the world. They explain how they discovered bias in their data, and how they used R to address it:

“I work on a team that collects survey-like data. One of the tasks my team does is called a side-by-side comparison. For example, we might show users two ads side-by-side at the same time. In our survey, we ask which of the two ads they prefer. In one case, after many iterations, we were seeing consistent bias in favor of the first item. There was also a measurable decrease in the preference for an item if we swapped its position to second.

So we decided to add randomization to the position of the ads using R. We wanted to make sure that the items appeared in the first and second positions with similar frequencies. We used sample() to inject a randomization element into our R programming. In R, the sample() function allows you to take a random sample of elements from a data set. Adding this piece of code shuffled the rows in our data set randomly. So when we presented the ads to users, the positions of the ads were now random and controlled for bias. This made the survey more effective and the data more reliable.”

**Key takeaways**

The sample() function is just one of many functions and methods in R that you can use to address bias in your data. Depending on the kind of analysis you are conducting, you might need to incorporate some advanced processes in your programming. Although this program won’t cover those kinds of processes in detail, you will likely learn more about them as you get more experience in the data analytics field.

To learn more about bias and data ethics, check out these resources:

* [**Bias function:**](https://www.rdocumentation.org/packages/SimDesign/versions/2.2/topics/bias) This web page is a good starting point to learn about how the bias function in R can help you identify and manage bias in your analysis.
* [**Data Science Ethics**](https://datasciencebox.org/02-ethics.html): This online course provides slides, videos, and exercises to help you learn more about ethics in the world of data analytics. It includes information about data privacy, misrepresentation in data, and applying ethics to your visualizations.

### 1.

Question 1

Which of the following functions can a data analyst use to get a statistical summary of their dataset? Select all that apply.

1 point

ggplot2()

cor()

mean()

sd()

### 2.

Question 2

A data analyst inputs the following command:  **quartet %>% group\_by(set) %>% summarize(mean(x), sd(x), mean(y), sd(y), cor(x, y)).** Which of the functions in this command can help them determine how strongly related their variables are?

1 point

sd(y)

cor(x,y)

sd(x)

mean(y)

### 3.

Question 3

Fill in the blank: The bias function compares the actual outcome of the data with the \_\_\_\_\_ outcome to determine whether or not the model is biased.

1 point

final

predicted

desired

probable

Glossary

Data Analytics

Terms and Definitions

A

A/B testing: The process of testing two variations of the same web page to determine which

page is more successful at attracting user traffic and generating revenue

Absolute reference: A reference within a function that is locked so that rows and columns

won’t change if the function is copied

Access control: Features such as password protection, user permissions, and encryption that

are used to protect a spreadsheet

Accuracy: The degree to which data conforms to the actual entity being measured or described

Action-oriented question: A question whose answers lead to change

Administrative metadata: Metadata that indicates the technical source of a digital asset

Agenda: A list of scheduled appointments

Aggregation: The process of collecting or gathering many separate pieces into a whole

Algorithm: A process or set of rules followed for a specific task

Aliasing: Temporarily naming a table or column in a query to make it easier to read and write

Alternative text: Text that provides an alternative to non-text content, such as images and

videos

Analytical skills: Qualities and characteristics associated with using facts to solve problems

Analytical thinking: The process of identifying and defining a problem, then solving it by using

data in an organized, step-by-step manner

Annotation: Text that briefly explains data or helps focus the audience on a particular aspect of

the data in a visualization

Anscombe’s quartet: Four datasets that have nearly identical summary statistics but contain

different plotted values

Area chart: A data visualization that uses individual data points for a changing variable

connected by a continuous line with a filled in area underneath

Argument (R): Information needed by a function in R in order to run

Arithmetic operator: An operator used to perform basic math operations such as addition,

subtraction, multiplication, and division

Array: A collection of values in spreadsheet cells

Assignment operator: An operator used to assign values to variables and vectors

Attribute: A characteristic or quality of data used to label a column in a table

Audio file: Digitized audio storage usually in an MP3, AAC, or other compressed format

AVERAGE: A spreadsheet function that returns an average of the values from a selected range

AVERAGEIF: A spreadsheet function that returns the average of all cell values from a given

range that meet a specified condition

B

Bad data source: A data source that is not reliable, original, comprehensive, current, and cited

(ROCCC)

Balance: The design principle of creating aesthetic appeal and clarity in a data visualization by

evenly distributing visual elements

Bar graph: A data visualization that uses size to contrast and compare two or more values

Bias: A conscious or subconscious preference in favor of or against a person, group of people,

or thing

Big data: Large, complex datasets typically involving long periods of time, which enable data

analysts to address far-reaching business problems

Boolean data: A data type with only two possible values, usually true or false

Borders: Lines that can be added around two or more cells on a spreadsheet

Box plot: A data visualization that displays the distribution of values along an x-axis

Bubble chart: A data visualization that displays individual data points as bubbles, comparing

numeric values by their relative size

Bullet graph: A data visualization that displays data as a horizontal bar chart moving toward a

desired value

Business metric: A standard of measurement used to solve a business task

Business task: The question or problem data analysis resolves for a business

C

C#: An object-oriented programming language used to create games and mobile apps in the

.NET open source developer platform

C++: An extension of the C programming language that is used to create console games, such

as those for Xbox

Calculated field: A new field within a pivot table that carries out certain calculations based on

the values of other fields

Calculus: A branch of mathematics that involves the study of rates of change and the changes

between values that are related by a function

CASE: A SQL statement that returns records that meet conditions by including an if/then

statement in a query

Case study: A common way for employers to assess job skills and gain insight into how a

candidate approaches common data-related challenges

CAST: A SQL function that converts data from one datatype to another

Causation: When an action directly leads to an outcome, such as a cause-effect relationship

Cell reference: A cell or a range of cells in a worksheet typically used in formulas and functions

Changelog: A file containing a chronologically ordered list of modifications made to a project

Channel: A visual aspect or variable that represents characteristics of the data in a visualization

Chart: A graphical representation of data from a worksheet

Circle view: A data visualization that shows comparative strength in data

Clean data: Data that is complete, correct, and relevant to the problem being solved

Cloud: A place to keep data online, rather than a computer hard drive

Cluster: A collection of data points on a data visualization with similar values

COALESCE: A SQL function that returns non-null values in a list

Coding: The process of writing instructions to a computer in the syntax of a specific

programming language

Column chart: A data visualization that uses individual data points for a changing variable,

represented as vertical columns

Combo chart: A data visualization that combines more than one visualization type

Compatibility: How well two or more datasets are able to work together

Completeness: The degree to which data contains all desired components or measures

Computer programming: The process of giving instructions to a computer in order to perform

an action or set of actions

CONCAT: A SQL function that adds strings together to create new text strings that can be used

as unique keys

CONCATENATE: A spreadsheet function that joins together two or more text strings

Conditional formatting: A spreadsheet tool that changes how cells appear when values meet

specific conditions

Conditional statement: A declaration that if a certain condition holds, then a certain event must

take place

Confidence interval: A range of values that conveys how likely a statistical estimate reflects

the population

Confidence level: The probability that a sample size accurately reflects the greater population

Confirmation bias: The tendency to search for or interpret information in a way that confirms

pre-existing beliefs

Consent: The aspect of data ethics that presumes an individual’s right to know how and why

their personal data will be used before agreeing to provide it

Consistency: The degree to which data is repeatable from different points of entry or collection

Context: The condition in which something exists or happens

Continuous data: Data that is measured and can have almost any numeric value

CONVERT: A SQL function that changes the unit of measurement of a value in data

Cookie: A small file stored on a computer that contains information about its users

Correlation: The measure of the degree to which two variables change in relationship to each

other

COUNT: A spreadsheet function that counts the number of cells within a range that meet a

specified condition

COUNTA: A spreadsheet function that counts the total number of values within a range that

meet specified criteria

COUNTIF: A spreadsheet function that returns the number of cells within a range that match a

specified value

COUNT DISTINCT: A SQL function that only returns the distinct values in a specified range

CRAN (Comprehensive R Archive Network) (R): An online archive with R packages, source

code, manuals, and documentation

CREATE TABLE: A SQL clause that adds a temporary table to a database that can be used by

multiple people

Cross-field validation: A process that ensures certain conditions for multiple data fields are

satisfied

CSS (Cascading Style Sheets): A programming language used for web page design that

controls graphic elements and page presentation

CSV (comma-separated values) file: A delimited text file that uses a comma to separate

values

Currency: The aspect of data ethics that presumes individuals should be aware of financial

transactions resulting from the use of their personal data and the scale of those transactions

D

Dashboard: A tool that monitors live, incoming data

Data: A collection of facts

Data aggregation: The process of gathering data from multiple sources and combining it into a

single, summarized collection

Data analysis: The collection, transformation, and organization of data in order to draw

conclusions, make predictions, and drive informed decision-making

Data analysis process: The six phases of ask, prepare, process, analyze, share, and act

whose purpose is to gain insights that drive informed decision-making

Data analyst: Someone who collects, transforms, and organizes data in order to draw

conclusions, make predictions, and drive informed decision-making

Data analytics: The science of data

Data anonymization: The process of protecting people's private or sensitive data by eliminating

identifying information

Data bias: When a preference in favor of or against a person, group of people, or thing

systematically skews data analysis results in a certain direction

Data blending: A Tableau method that combines data from multiple data sources

Data composition: The process of combining the individual parts in a visualization and

displaying them together as a whole

Data constraints: The criteria that determine whether a piece of a data is clean and valid

Data design: How information is organized

Data-driven decision-making: Using facts to guide business strategy

Data ecosystem: The various elements that interact with one another in order to produce,

manage, store, organize, analyze, and share data

Data element: A piece of information in a dataset

Data engineer: A professional who transforms data into a useful format for analysis and gives it

a reliable infrastructure

Data ethics: Well-founded standards of right and wrong that dictate how data is collected,

shared, and used

Data frame: A collection of columns containing data, similar to a spreadsheet or SQL table

Data governance: A process for ensuring the formal management of a company’s data assets

Data-inspired decision-making: Exploring different data sources to find out what they have in

common

Data integrity: The accuracy, completeness, consistency, and trustworthiness of data

throughout its life cycle

Data interoperability: The ability to integrate data from multiple sources and a key factor

leading to the successful use of open data among companies and governments

Data life cycle: The sequence of stages that data experiences, which include plan, capture,

manage, analyze, archive, and destroy

Data manipulation: The process of changing data to make it more organized and easier to

read

Data mapping: The process of matching fields from one data source to another

Data merging: The process of combining two or more datasets into a single dataset

Data model: A tool for organizing data elements and how they relate to one another

Data privacy: Preserving a data subject’s information any time a data transaction occurs

Data range: Numerical values that fall between predefined maximum and minimum values

Data replication: The process of storing data in multiple locations

Data science: A field of study that uses raw data to create new ways of modeling and

understanding the unknown

Data security: Protecting data from unauthorized access or corruption by adopting safety

measures

Data storytelling: Communicating the meaning of a dataset with visuals and a narrative that

are customized for an audience

Data strategy: The management of the people, processes, and tools used in data analysis

Data structure: A format for organizing and storing data

Data transfer: The process of copying data from a storage device to computer memory or from

one computer to another

Data type: An attribute that describes a piece of data based on its values, its programming

language, or the operations it can perform

Data validation: A tool for checking the accuracy and quality of data

Data validation process: The process of checking and rechecking the quality of data so that it

is complete, accurate, secure and consistent

Data visualization: The graphical representation of data

Data warehousing specialist: A professional who develops processes and procedures to

effectively store and organize data

Database: A collection of data stored in a computer system

Dataset: A collection of data that can be manipulated or analyzed as one unit

DATEDIF: A spreadsheet function that calculates the number of days, months, or years

between two dates

Decision tree: A tool that helps analysts make decisions about critical features of a visualization

Delimiter: A character that indicates the beginning or end of a data item

Density map: A data visualization that represents concentrations, with color representing the

number or frequency of data points in a given area on a map

Descriptive metadata: Metadata that describes a piece of data and can be used to identify it at

a later point in time

Design thinking: A process used to solve complex problems in a user-centric way

Digital photo: An electronic or computer-based image usually in BMP or JPG format

Dirty data: Data that is incomplete, incorrect, or irrelevant to the problem to be solved

Discrete data: Data that is counted and has a limited number of values

DISTINCT: A keyword that is added to a SQL SELECT statement to retrieve only non-duplicate

entries

Distribution graph: A data visualization that displays the frequency of various outcomes in a

sample

Diverging color palette: A color theme that displays two ranges of data values using two

different hues, with color intensity representing the magnitude of the values

Donut chart: A data visualization where segments of a ring represent data values adding up to

a whole

dplyr (R): An R package in Tidyverse that offers a consistent set of functions to complete

common data-manipulation tasks

DROP TABLE: A SQL clause that removes a temporary table from a database

Duplicate data: Any record that inadvertently shares data with another record

Dynamic visualizations: Data visualizations that are interactive or change over time

E

Emphasis: The design principle of arranging visual elements to focus the audience’s attention

on important information in a data visualization

Engagement: Capturing and holding someone’s interest and attention during a data

presentation

Equation: A calculation that involves addition, subtraction, multiplication, or division (also called

a math expression)

Estimated response rate: The average number of people who typically complete a survey

Ethics: Well-founded standards of right and wrong that prescribe what humans ought to do,

usually in terms of rights, obligations, benefits to society, fairness, or specific virtues

Experimenter bias: The tendency for different people to observe things differently (Refer to

Observer bias)

External data: Data that lives, and is generated, outside of an organization

F

Factor (R): An object that stores categorical data where the data values are limited and usually

based on a finite group, such as country or year

Fairness: A quality of data analysis that does not create or reinforce bias

Field: A single piece of information from a row or column of a spreadsheet; in a data table,

typically a column in the table

Field length: A tool for determining how many characters can be keyed into a spreadsheet field

Fill handle: A box in the lower-right-hand corner of a selected spreadsheet cell that can be

dragged through neighboring cells in order to continue an instruction

Filled map: A data visualization that colors areas in a map based on measurements or

dimensions

Filtering: The process of showing only the data that meets a specified criteria while hiding the

rest

Find and replace: A tool that finds a specified search term and replaces it with something else

First-party data: Data collected by an individual or group using their own resources

Float: A number that contains a decimal

Foreign key: A field within a database table that is a primary key in another table (Refer to

primary key)

Formula: A set of instructions used to perform a calculation using the data in a spreadsheet

Framework: The context a presentation needs to create logical connections that tie back to the

business task and metrics

FROM: The section of a query that indicates from which table(s) to extract the data

Function: A preset command that automatically performs a specified process or task using the

data in a spreadsheet

Function (R): A body of reusable code for performing specific tasks in R

FWF (fixed-width file): A text file with a specific format, which enables the saving of textual

data in an organized fashion

G

Gantt chart: A data visualization that displays the duration of events or activities on a timeline

Gap analysis: A method for examining and evaluating the current state of a process in order to

identify opportunities for improvement in the future

Gauge chart: A data visualization that shows a single result within a progressive range of

values

General Data Protection Regulation of the European Union (GDPR): Policy-making body in

the European Union created to help protect people and their data

Geolocation: The geographical location of a person or device by means of digital information

ggplot2 (R): An R package in Tidyverse that creates a variety of data visualizations by applying

different visual properties to the data variables in R

Good data source: A data source that is reliable, original, comprehensive, current, and cited

(ROCCC)

GROUP BY: A SQL clause that groups rows that have the same values from a table into

summary rows

H

HAVING: A SQL clause that adds a filter to a query instead of the underlying table that can only

be used with aggregate functions

head() (R): An R function that returns a preview of the column names and the first few rows of a

dataset

Header: The first row in a spreadsheet that labels the type of data in each column

Headline: Text at the top of a visualization that communicates the data being presented

Heat map: A data visualization that uses color contrast to compare categories in a dataset

Highlight table: A data visualization that uses conditional formatting and color on a table

Histogram: A data visualization that shows how often data values fall into certain ranges

HTML5: A programming language that provides structure for web pages and connects to

hosting platforms

Hypothesis: A theory that an analysis is trying to prove or disprove with data

Hypothesis testing: A process to determine if a survey or experiment has meaningful results

I

IDE (Integrated Development Environment): A software application that brings together all

the tools a data analyst may want to use in a single place

Incomplete data: Data that is missing important fields

Inconsistent data: Data that uses different formats to represent the same thing

Incorrect/inaccurate data: Data that is complete but inaccurate

INNER JOIN : A SQL function that returns records with matching values in both tables

Inner query: A SQL subquery that is inside of another SQL statement

Internal data: Data that lives within a company’s own systems

Interpretation bias: The tendency to interpret ambiguous situations in a positive or negative

way

J

Java: A programming language widely used to create enterprise web applications that can run

on multiple clients

JOIN: A SQL function that is used to combine rows from two or more tables based on a related

column

K

L

Label: Text in a visualization that identifies a value or describes a scale

Leading question: A question that steers people toward a certain response

LEFT: A function that returns a set number of characters from the left side of a text string

LEFT JOIN: A SQL function that will return all the records from the left table and only the

matching records from the right table

Legend: A tool that identifies the meaning of various elements in a data visualization

LEN: A function that returns the length of a text string by counting the number of characters it

contains

Length: The number of characters in a text string

Library: A directory containing all of a data analyst’s installed packages

LIMIT: A SQL clause that specifies the maximum number of records returned in a query

Line graph: A data visualization that uses one or more lines to display shifts or changes in data

over time

List: A vector whose elements can be of any type

Live data: Data that is automatically updated

Log file: A computer-generated file that records events from operating systems and other

software programs

Logical operator: An operator that returns a logical data type

Long data: A dataset in which each row is one time point per subject, so each subject has data

in multiple rows

M

Mandatory: A data value that cannot be left blank or empty

Map: A data visualization that organizes data geographically

Margin of error: The maximum amount that sample results are expected to differ from those of

the actual population

Mark: A visual object in a data visualization such as a point, line, or shape

MATCH: A spreadsheet function used to locate the position of a specific lookup value

Math expression: A calculation that involves addition, subtraction, multiplication, or division

(also called an equation)

Math function: A function that is used as part of a mathematical formula

Matrix: A two-dimensional collection of data elements with rows and columns

MAX: A function that returns the largest numeric value from a range of cells

MAXIFS: A spreadsheet function that returns the maximum value from a given range that meets

a specified condition

McCandless Method: A method for presenting data visualizations that moves from general to

specific information

Measurable question: A question whose answers can be quantified and assessed

Mental model: A data analyst’s thought process and approach to a problem

Mentor: Someone who shares knowledge, skills, and experience to help another grow both

professionally and personally

Merger: An agreement that unites two organizations into a single new one

Metadata: Data about data

Metadata repository: A database created to store metadata

Metric: A single, quantifiable type of data that is used for measurement

Metric goal: A measurable goal set by a company and evaluated using metrics

MID: A function that returns a segment from the middle of a text string

MIN: A spreadsheet function that returns the smallest numeric value from a range of cells

MINIFS: A spreadsheet function that returns the minimum value from a given range that meets

a specified condition

Modulo: An operator (%) that returns the remainder when one number is divided by another

Movement: The design principle of arranging visual elements to guide the audience’s eyes from

one part of a data visualization to another

mutate() (R): An R function that makes changes to a dataframe separating and merging

columns or creating new variables

N

Naming conventions: Consistent guidelines that describe the content, creation date, and

version of a file in its name

Narrative: (Refer to story)

Nested: Code that performs a particular function and is contained within code that performs a

broader function

Nested function: A function that is completely contained within another function

Networking: Building relationships by meeting people both in person and online

Nominal data: A type of qualitative data that is categorized without a set order

Normalized database: A database in which only related data is stored in each table

Notebook: An interactive, editable programming environment for creating data reports and

showcasing data skills

Null: An indication that a value does not exist in a dataset

O

Observation: The attributes that describe a piece of data contained in a row of a table

Observer bias: The tendency for different people to observe things differently (also called

experimenter bias)

Open data: Data that is available to the public

Open-source: Code that is freely available and may be modified and shared by the people who

use it

Openness: The aspect of data ethics that promotes the free access, usage, and sharing of data

Operator: A symbol that names the operation or calculation to be performed

ORDER BY: A SQL clause that sorts results returned in a query

Order of operations: Using parentheses to group together spreadsheet values in order to

clarify the order in which operations should be performed

Ordinal data: Qualitative data with a set order or scale

Outdated data: Any data that has been superseded by newer and more accurate information

OUTER JOIN: A SQL function that combines RIGHT and LEFT JOIN to return all matching

records in both tables

Outer query: A SQL statement containing a subquery

Ownership: The aspect of data ethics that presumes individuals own the raw data they provide

and have primary control over its usage, processing, and sharing

P

Package (R): A unit of reproducible R code

Packed bubble chart: A data visualization that displays data in clustered circles

Pattern: The design principle of using similar visual elements to demonstrate trends and

relationships in a data visualization

PHP (Hypertext Preprocessor): A programming language for web application development

Pie chart: A data visualization that uses segments of a circle to represent the proportions of

each data category compared to the whole

Pipe (R): A tool in R for expressing a sequence of multiple operations, represented with “%>%”

Pivot chart: A chart created from the fields in a pivot table

Pivot table: A data summarization tool used to sort, reorganize, group, count, total, or average

data

Pixel: In digital imaging, a small area of illumination on a display screen that, when combined

with other adjacent areas, forms a digital image

Population: In data analytics, all possible data values in a dataset

Portfolio: A collection of materials that can be shared with potential employers

Pre-attentive attributes: The elements of a data visualization that an audience recognizes

automatically without conscious effort

Primary key: An identifier in a database that references a column in which each value is unique

(Refer to foreign key)

Problem domain: The area of analysis that encompasses every activity affecting or affected by

a problem

Problem types: The various problems that data analysts encounter, including categorizing

things, discovering connections, finding patterns, identifying themes, making predictions, and

spotting something unusual

Profit margin: A percentage that indicates how many cents of profit has been generated for

each dollar of sale

Programming language: A system of words and symbols used to write instructions that

computers follow

Proportion: The design principle of using the relative size and arrangement of visual elements

to demonstrate information in a data visualization

Python: A general-purpose programming language

Q

Qualitative data: A subjective and explanatory measure of a quality or characteristic

Quantitative data: A specific and objective measure, such as a number, quantity, or range

Query: A request for data or information from a database

Query language: A computer programming language used to communicate with a database

R

R: A programming language used for statistical analysis, visualization, and other data analysis

Random sampling: A way of selecting a sample from a population so that every possible type

of the sample has an equal chance of being chosen

Range: A collection of two or more cells in a spreadsheet

Ranking: A system to position values of a dataset within a scale of achievement or status

readr (R): An R package in Tidyverse used for importing data

Record: A collection of related data in a data table, usually synonymous with row

Redundancy: When the same piece of data is stored in two or more places

Reframing: The process of restating a problem or challenge, then redirecting it toward a

potential resolution

Regular expression (RegEx): A rule that says the values in a table must match a prescribed

pattern

Relational database: A database that contains a series of tables that can be connected to form

relationships

Relational operator: An operator used to compare values, also known as a comparator

Relativity: The process of considering observations in relation or proportion to something else

Relevant question: A question that has significance to the problem to be solved

Remove duplicates: A spreadsheet tool that automatically searches for and eliminates

duplicate entries from a spreadsheet

Repetition: The design principle of repeating visual elements to demonstrate meaning in a data

visualization

Report: A static collection of data periodically given to stakeholders

Return on investment (ROI): A formula that uses the metrics of investment and profit to

evaluate the success of an investment

Revenue: The total amount of income generated by the sale of goods or services

Rhythm: The design principle of creating movement and flow in a data visualization to engage

an audience

RIGHT: A function that returns a set number of characters from the right side of a text string

RIGHT JOIN: A SQL function that will return all records from the right table and only the

matching records from the left

Root cause: The reason why a problem occurs

ROUND: A SQL function that returns a number rounded to a certain number of decimal places.

Ruby: An object-oriented programming language for web application development

S

Sample: In data analytics, a segment of a population that is representative of the entire

population

Sampling bias: Overrepresenting or underrepresenting certain members of a population as a

result of working with a sample that is not representative of the population as a whole

Scatterplot: A data visualization that represents relationships between different variables with

individual data points without a connecting line

Schema: A way of describing how something, such as data, is organized

Scope of work (SOW): An agreed-upon outline of the tasks to be performed during a project

Second-party data: Data collected by a group directly from its audience and then sold

SELECT: The section of a query that indicates from which column(s) to extract the data

SELECT INTO: A SQL clause that copies data from one table into a temporary table without

adding the new table to the database

Small data: Small, specific data points typically involving a short period of time, which are

useful for making day-to-day decisions

SMART methodology: A tool for determining a question’s effectiveness based on whether it is

specific, measurable, action-oriented, relevant, and time-bound

Social media: Websites and applications through which users create and share content or

participate in social networking

Soft skills: Nontechnical traits and behaviors that relate to how people work

Sort range: A spreadsheet menu function that sorts a specified range and preserves the cells

outside the range

Sort sheet: A spreadsheet menu function that sorts all data by the ranking of a specific sorted

column and keeps data together across rows

Sorting: The process of arranging data into a meaningful order to make it easier to understand,

analyze, and visualize

Specific question: A question that is simple, significant, and focused on a single topic or a few

closely related ideas

SPLIT: A spreadsheet function that divides text around a specified character and puts each

fragment into a new, separate cell

Sponsor: A professional advocate who is committed to moving forward the career of another

Spotlightling: Scanning through data to quickly identify the most important insights

Spreadsheet: A digital worksheet

SQL: (Refer to Structured Query Language)

Stakeholders: People who invest time and resources into a project and are interested in its

outcome

Static data: Data that doesn’t change once it has been recorded

Static visualization: A data visualization that does not change over time unless it is edited

Statistical power: The probability that a test of significance will recognize an effect that is

present

Statistical significance: The probability that sample results are not due to random chance

Statistics: The study of how to collect, analyze, summarize, and present data

Story: The narrative of a data presentation that makes it meaningful and interesting

String data type: A sequence of characters and punctuation that contains textual information

(also called text data type)

Structural metadata: Metadata that indicates how a piece of data is organized and whether it is

part of one or more than one data collection

Structured data: Data organized in a certain format such as rows and columns

Structured Query Language: A computer programming language used to communicate with a

database

Structured thinking: The process of recognizing the current problem or situation, organizing

available information, revealing gaps and opportunities, and identifying options

Subquery: A SQL query that is nested inside a larger query

SUBSTR: A SQL function that extracts a substring from a string variable

Substring: A subset of a text string

Subtitle: Text that supports a headline by adding context and description

SUM: A spreadsheet function that adds the values of a selected range of cells

SUMIF: A spreadsheet function that adds numeric data based on one condition

Summary table: A table used to summarize statistical information about data

SUMPRODUCT: A function that multiplies arrays and returns the sum of those products

Swift: A programming language for macOS, iOS, watchOS, and tvOS

Symbol map: A data visualization that displays a mark over a given longitude and latitude

Syntax: The predetermined structure of a language that includes all required words, symbols,

and punctuation, as well as their proper placement

T

Tableau: A business intelligence and analytics platform that helps people visualize, understand,

and make decisions with data

Technical mindset: The ability to break things down into smaller steps or pieces and work with

them in an orderly and logical way

Temporary table: A database table that is created and exists temporarily on a database server

Text data type: A sequence of characters and punctuation that contains textual information

(also called string data type)

Text string: A group of characters within a cell, most often composed of letters

Third-party data: Data provided from outside sources who didn’t collect it directly

Tibble (R): A streamlined variation of data frames

Tidy data (R): A way of standardizing the organization of data within R

tidyr (R): An R package in Tidyverse used for data cleaning to make tidy data

Tidyverse (R): A system of packages in R with a common design philosophy for data

manipulation, exploration, and visualization

Time-bound question: A question that specifies a timeframe to be studied

Transaction transparency: The aspect of data ethics that presumes all data-processing

activities and algorithms should be explainable and understood by the individual who provides

the data

Transferable skills: Skills and qualities that can transfer from one job or industry to another

TRIM: A function that removes leading, trailing, and repeated spaces in data

TSV (Tab-separated values file): A text file that stores a data table by separating columns of

data with tabs

Turnover rate: The rate at which employees voluntarily leave a company

Typecasting: Converting data from one type to another

U

Unbiased sampling: When the sample of the population being measured is representative of

the population as a whole

Underscores: Lines used to underline words and connect text characters

Unfair question: A question that makes assumptions or is difficult to answer honestly

Unique: A value that can’t have a duplicate

United States Census Bureau: An agency in the U.S. Department of Commerce that serves as

the nation’s leading provider of quality data about its people and economy

Unity: The design principle of using visual elements that complement each other to create

aesthetic appeal and clarity in a data visualization

Unstructured data: Data that is not organized in any easily identifiable manner

V

Validity: The degree to which data conforms to constraints when it is input, collected, or created

VALUE: A spreadsheet function that converts a text string that represents a number to a

numeric value

Variable (R): A representation of a value in R that can be stored for later use

Variety: The design principle of using different kinds of visual elements in a data visualization to

engage an audience

Vector (R): A group of data elements of the same type stored in a one-dimensional sequence in

R

Verification: A process to confirm that a data-cleaning effort was well executed and the

resulting data is accurate and reliable

Video file: A collection of images, audio files, and other data usually encoded in a compressed

format such as MP4, MV4, MOV, AVI, or FLV

Vignette (R): Documentation for an R package that describes the problem the package is

designed to solve, explains how its functions can be used, and lists any dependencies on other

packages

Visual form: The appearance of a data visualization that gives it structure and aesthetic appeal

Visualization: (Refer to Data visualization)

VLOOKUP: A spreadsheet function that vertically searches for a certain value in a column to

return a corresponding piece of information

W

WHERE: The section of a query that specifies criteria that the requested data must meet

Wide data: A dataset in which every data subject has a single row with multiple columns to hold

the values of various attributes of the subject

WITH: A SQL clause that creates a temporary table that can be queried multiple times

World Health Organization: An organization whose primary role is to direct and coordinate

international health within the United Nations system

X

X-axis: The horizontal line of a graph usually placed at the bottom, which is often used to

represent time scales and discrete categories

Y

Y-axis: The vertical line of a graph usually placed to the left, which is often used to represent

frequencies and other numerical variables

### 1.

Question 1

What scenarios would prevent you from being able to use a tibble?

1 point

You need to store numerical data

You need to change the data types of inputs

You need to create column names

You need to create row names

### 2.

Question 2

A data analyst is working with a large data frame. It contains so many columns that they don’t all fit on the screen at once. The analyst wants a quick list of all of the column names to get a better idea of what is in their data. What function should they use?

1 point

**mutate()**

**str()**

**colnames()**

**head()**

### 3.

Question 3

You are working with the ToothGrowth dataset. You want to use the **glimpse()** function to get a quick summary of the dataset. Write the code chunk that will give you this summary.

1

RunReset

How many variables does the ToothGrowth dataset contain?

1 point

5

4

2

3

### 4.

Question 4

You have a data frame named employees with a column named last\_name. What will the name of the employees column be in the results of the function **rename\_with(employees, toupper)**?

1 point

**last\_name**

**Last\_name**

**Last\_Name**

**LAST\_NAME**

### 5.

Question 5

A data analyst is working with the penguins dataset and wants to sort the penguins by body\_mass\_g from least to greatest. When they run the following code the penguin body mass data is not displayed in the correct order.

**penguins %>% arrange(body\_mass\_g)**

**head(penguins)**

What can the data analyst do to fix their code?

1 point

Use the **print()** function instead of the **head()** function

Add a minus sign in front of body\_mass\_g to reverse the order

Correct the capitalization of **arrange()** to **Arrange()**

Save the results of **arrange()** to a variable that gets passed to **head()**

### 6.

Question 6

You are working with the penguins dataset and want to understand the year of data collection for all combinations of species, island, and sex. At this point, the following code has already been written into your script:

**penguins %>%**

**drop\_na() %>%**

**group\_by(species, island, sex) %>%**

**summarize(min = min(year), max = max(year))**

1

RunReset

When you run the code in the code box, how many separate observational rows are returned by this code chunk?

1 point

6

2

10

3

### 7.

Question 7

A data analyst is working with a data frame called zoo\_records. They want to create a new column named is\_large\_animal that signifies if an animal has a weight of more than 199 kilograms. What code chunk lets the analyst create the is\_large\_animal column?

1 point

**zoo\_records %>% mutate(weight > 199 = is\_large\_animal)**

**zoo\_records %>% mutate(is\_large\_animal = weight > 199)**

**zoo\_records %>% mutate(is\_large\_animal == weight > 199)**

**zoo\_records %>% mutate(weight > 199 <- is\_large\_animal)**

### 8.

Question 8

A data analyst is using the **unite()** function to combine two columns into a single column. What does the **sep** parameter of the **unite()** function represent?

1 point

The vector of columns to join into the final column

The strings to place between each column

The data frame that is the target of the operation

The name of the final column formed from the original columns

### 9.

Question 9

In R, which statistical measure demonstrates how strong the relationship is between two variables?

1 point

Correlation

Standard deviation

Average

Maximum

### 10.

Question 10

A data analyst is studying weather data. They write the following code chunk:

**bias(actual\_temp, predicted\_temp)**

What will this code chunk calculate?

1 point

The total average of the values

The maximum difference between the actual and predicted values

The minimum difference between the actual and predicted values

The average difference between the actual and predicted values

# Common problems when visualizing in R

You can save this reading for future reference. Feel free to download a PDF version of this reading below:

[Common problems encountered when visualizing in R.pdf](https://d3c33hcgiwev3.cloudfront.net/pALIn_4vSECCyJ_-L4hAJg_3c83255ab5004b1a99957907331217b0_Common-problems-encountered-when-visualizing-in-R.pdf?Expires=1706918400&Signature=avrEZPGJFeBOFJplnFEvcu7wZZ6AWUJxnmJPmT8fINv64J020WJaelNa-8kO0NGA3BNzlQgdQvfETteQIAMrjwlWHyBWf8wLbVjPR2gqnjDjRskJrD9ipGqw2UiZTVCl-7dcam1j86m6Ynxzj2aLlj108BgV8ELWC5ylZKI~wdA_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[PDF File](https://d3c33hcgiwev3.cloudfront.net/pALIn_4vSECCyJ_-L4hAJg_3c83255ab5004b1a99957907331217b0_Common-problems-encountered-when-visualizing-in-R.pdf?Expires=1706918400&Signature=avrEZPGJFeBOFJplnFEvcu7wZZ6AWUJxnmJPmT8fINv64J020WJaelNa-8kO0NGA3BNzlQgdQvfETteQIAMrjwlWHyBWf8wLbVjPR2gqnjDjRskJrD9ipGqw2UiZTVCl-7dcam1j86m6Ynxzj2aLlj108BgV8ELWC5ylZKI~wdA_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

Coding errors are an inevitable part of writing code—especially when you are first beginning to learn a new programming language. In this reading, you will learn how to recognize common coding errors when creating visualizations using **ggplot2**. You will also find links to some resources that you can use to help address any coding problems you might encounter moving forward.

## Common coding errors in ggplot2

When working with R code in ggplot2, a lot of the most common coding errors involve issues with syntax, like misplaced characters. That is why paying attention to details is such an important part of writing code. When there is an error in your code that R is able to detect, it will generate an error message. Error messages can help point you in the right direction, but they won’t always help you figure out the precise problem.

Let’s explore a few of the most common coding errors you might encounter in ggplot2.

### **Case sensitivity**

R code is case sensitive. If you accidentally capitalize the first letter in a certain function, it might affect your code. Here is an example:

**Glimpse(penguins)**

The error message lets you know that R cannot find a function named “Glimpse”:

**Error in Glimpse(penguins) : could not find function "Glimpse"**

But you know that the function glimpse (lowercase “g”) does exist. Notice that the error message doesn’t explain exactly what is wrong but does point you in a general direction.

Based on that, you can figure out that this is the correct code:

**glimpse(penguins)**

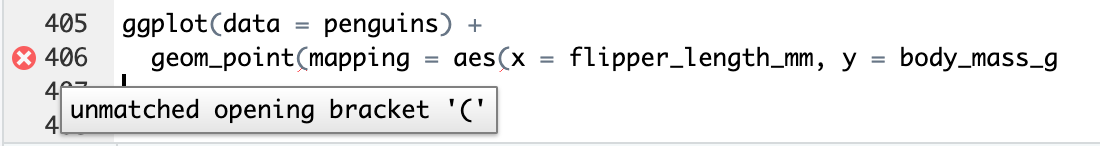
### **Balancing parentheses and quotation marks**

Another common R coding error involves parentheses and quotation marks. In R, you need to make sure that every opening parenthesis in your function has a closing parenthesis, and every opening quotation mark has a closing quotation mark. For example, if you run the following code, nothing happens. R does not create the plot. That is because the second line of code is missing two closing parentheses:

**ggplot(data = penguins) +**

**geom\_point(mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g**

RStudio does alert you to the problem. To the left of the line of code in your RStudio source  editor, you might notice a red circle with a white “X” in the center. If you hover over the circle with your cursor, this message appears:



RStudio lets you know that you have an unmatched opening bracket. So, to correct the code, you know that you need to add a closing bracket to match each opening bracket.

Here is the correct code:

**ggplot(data = penguins) +**

**geom\_point(mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g))**

### **Using the plus sign to add layers**

In ggplot2, you need to add a plus sign (“+”) to your code when you add a new layer to your plot. Putting the plus sign in the wrong place is a common mistake. The plus sign should always be placed at the end of a line of code, and not at the beginning of a line.

Here’s an example of code that includes incorrect placement of the plus sign:

**ggplot(data = penguins)**

**+ geom\_point(mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g))**

In this case, R’s error message identifies the problem, and prompts you to correct it:

**Error: Cannot use `+.gg()` with a single argument. Did you accidentally put + on a new line?**

Here is the correct code:

**ggplot(data = penguins) +**

**geom\_point(mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g))**

You also might accidentally use a pipe instead of a plus sign to add a new layer to your plot, like this:

**ggplot(data = penguins)%>%**

**geom\_point(mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g))**

You then get the following error message:

**Error: `data` must be a data frame, or other object coercible by `fortify()`, not an S3 object with class gg/ggplot**

Here is the correct code:

**ggplot(data = penguins) +**

**geom\_point(mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g))**

Keeping these issues in mind and paying attention to details when you write code will help you reduce errors and save time, so you can stay focused on your analysis.

## Help resources

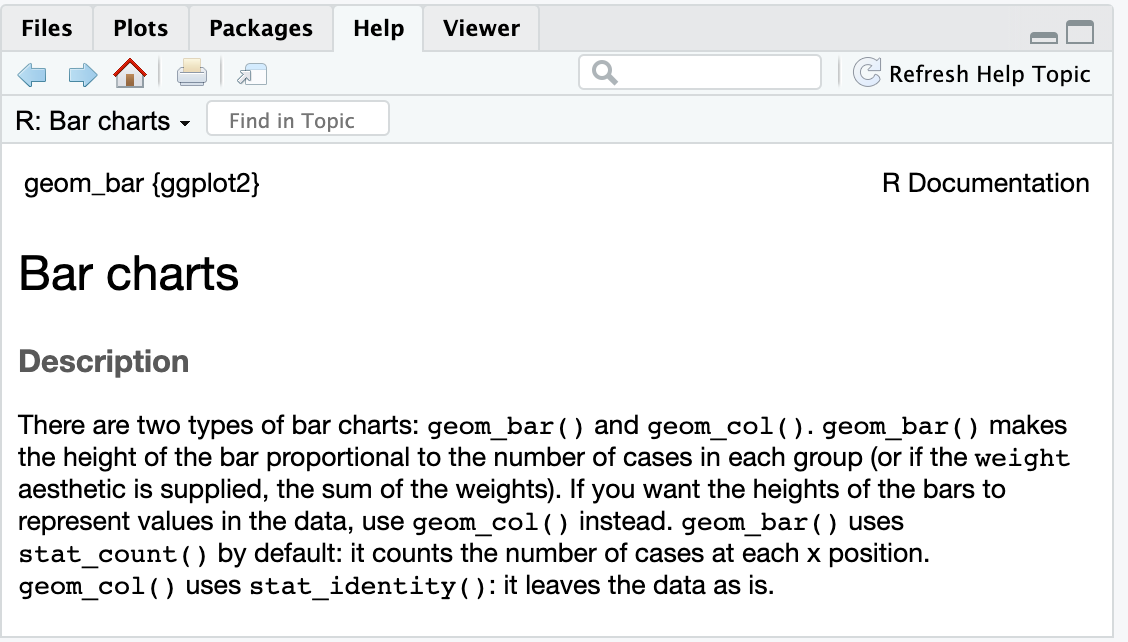
Everyone makes mistakes when writing code–it is just part of the learning process. Fortunately, there are lots of helpful resources available in RStudio and online.

### **R documentation**

R has built-in documentation for all functions and packages. To learn more about any R function, just run the code **?function\_name**. For example, if you want to learn more about the geom\_bar function, type:

?geom\_bar

When you run the code, an entry on “geom\_bar” appears in the Help viewer in the lower-right pane of your RStudio workspace. The entry begins with a “Description” section that discusses bar charts:



The [RDocumentation website](https://www.rdocumentation.org/) contains much of the same content in a slightly different format, with additional examples and links.

### **ggplot2 documentation**

The [ggplot2 page](https://ggplot2.tidyverse.org/), which is part of the official tidyverse documentation, is a great resource for all things related to ggplot2. It includes entries on key topics, useful examples of code, and links to other helpful resources.

### **Online search**

Doing an online search for the error message you are encountering (and including “R” and the function or package name in your search terms) is another option. There is a good chance someone else has already encountered the same error and posted about it online.

### **The R community**

If the other resources don’t help, you can try reaching out to the R community online. There are lots of useful online forums and websites where people ask for and receive help, including:

* [R for Data Science Online Learning Community](https://www.rfordatasci.com/)
* [RStudio Community](https://community.rstudio.com/)
* [Stackoverflow](http://stackoverflow.com/)
* [Twitter (#rstats)](https://twitter.com/hashtag/rstats?ref_src=twsrc%5Etfw%7Ctwcamp%5Etweetembed%7Ctwterm%5E1229486581620367361%7Ctwgr%5Eshare_3&ref_url=https%3A%2F%2Fwww.t4rstats.com%2F&src=hashtag_click)

### 1.

Question 1

In ggplot2, you can use the \_\_\_\_\_ function to specify the data frame to use for your plot.

1 point

aes()

geom\_point()

labs()

ggplot()

### 2.

Question 2

In ggplot2, you use the plus sign (+) to add a layer to your plot.

1 point

True

False

### 3.

Question 3

In ggplot2, what function do you use to map variables in your data to visual features of your plot?

1 point

The ggplot() function

The geom\_point() function

The geom\_bar() function

The aes() function

### 4.

Question 4

What type of plot will the following code create? **ggplot(data = penguins) + geom\_point(mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g))**

1 point

Scatterplot

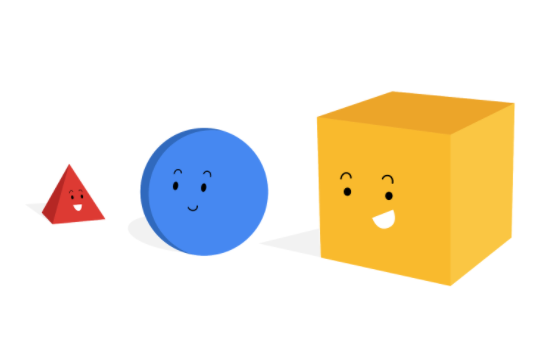
Line diagram

Boxplot

Bar chart

**Aesthetic attributes**

In this reading, you will learn about the three basic aesthetic attributes to consider when creating ggplot2 visualizations in R: **color, size,** and **shape.** These attributes are essential tools for creating data visualizations with ggplot2 and are built directly into its code.



**Aesthetics in ggplot2**

**ggplot2** is an R package that allows you to create different types of data visualizations right in your R workspace. In ggplot2, an **aesthetic**is defined as a visual property of an object in your plot.

There are three aesthetic attributes in ggplot2:

* **Color**: this allows you to change the color of all of the points on your plot, or the color of each data group
* **Size**: this allows you to change the size of the points on your plot by data group
* **Shape**: this allows you to change the shape of the points on your plot by data group

Here’s an example of how aesthetic attributes are displayed in R:

**ggplot(data, aes(x=distance, y= dep\_delay, color=carrier, size=air\_time, shape = carrier))       geom\_point()**

By applying these aesthetic attributes to your work with ggplot2, you can create data visualizations in R that clearly communicate trends in your data.

**Additional resources**

For more information about aesthetic attributes, check out these resources:

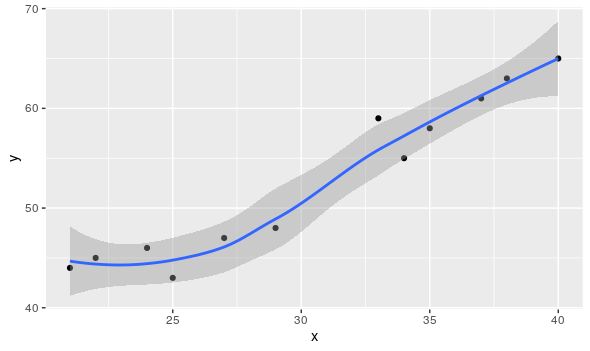
* [**Data visualization with ggplot2 cheat sheet**](https://ggplot2.tidyverse.org/): RStudio’s cheat sheet is a great reference to use while working with ggplot2. It has tons of helpful information, including explanations of how to use geoms and examples of the different visualizations that you can create.
* [**RDocumentation aes function**](https://www.rdocumentation.org/packages/ggplot2/versions/3.3.3/topics/aes): This guide describes the syntax of the **aes** function and explains what each argument does.

**Smoothing**

In this reading, you will learn about smoothing in ggplot2 and how it can be used to make your data visualizations in R clearer and easier to follow. Sometimes it can be hard to understand trends in your data from scatter plots alone. **Smoothing** enables the detection of a data trend even when you can't easily notice a trend from the plotted data points. ggplot2’s smoothing functionality is helpful because it adds a **smoothing line** as another layer to a plot; the smoothing line helps the data to make sense to a casual observer.

| **Example code** |
| --- |
| **ggplot(data, aes(x=distance, y= dep\_delay)) + geom\_point() + geom\_smooth()** |

The example code creates a plot with a trend line similar to the blue line below.

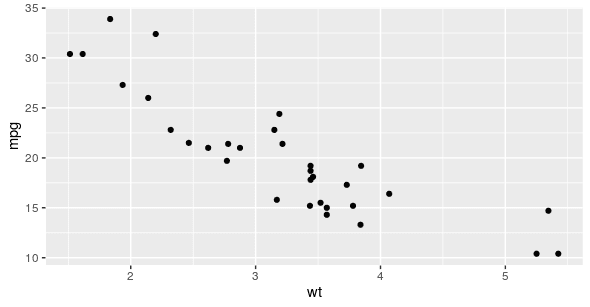


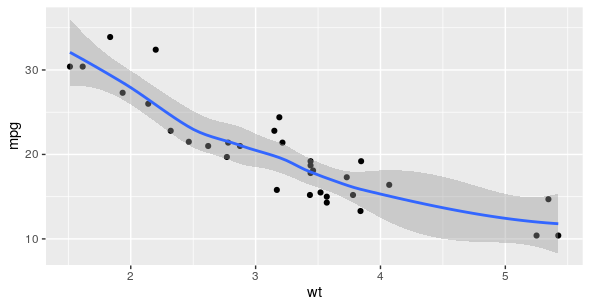
**Two types of smoothing**



| **Type of smoothing** | **Description** | **Example code** |
| --- | --- | --- |
| **Loess smoothing** | The loess smoothing process is best for smoothing plots with less than 1000 points. | **ggplot(data, aes(x=, y=))+  geom\_point() +       geom\_smooth(method="loess")** |
| **Gam smoothing** | Gam smoothing, or generalized additive model smoothing, is useful for smoothing plots with a large number of points. | **ggplot(data, aes(x=, y=)) + geom\_point() +         geom\_smooth(method="gam", formula = y ~s(x))** |

The smoothing functionality in ggplot2 helps make data plots more readable, so you are better able to recognize data trends and make key insights. The first plot below is the data before smoothing, and the second plot below is the same data after smoothing.





**Key takeaways**

Smoothing helps data professionals reveal trends. When scatterplots alone lack clarity, smoothing adds a trend line, making underlying patterns in the data easier to spot for casual observers. ggplot2 offers two smoothing methods: Loess is best for plots with fewer than 1,000 points, it creates a flexible, local smoother. Gam is ideal for larger datasets because it uses a more robust model for general trends. Smoothing enhances data communication, adding a visual cue to highlight trends so data visualizations become clearer and more impactful for audiences.

**Filtering and plots**

By this point you have likely downloaded at least a few packages into your R library. The tools in some of these packages can actually be combined and used together to become even more useful. This reading will share a few resources that will teach you how to use the filter function from **dplyr** to make the plots you create with **ggplot2** easier to read.



**Example of filtering data for plotting**

Filtering your data before you plot it allows you to focus on specific subsets of your data and gain more targeted insights. To do this, just include the dplyr filter() function in your ggplot syntax.

| **Example code** |
| --- |
| **data %>%    filter(variable1 == "DS") %>%   ggplot(aes(x = weight, y = variable2, colour = variable1)) +   geom\_point(alpha = 0.3,  position = position\_jitter()) + stat\_smooth(method = "lm")** |

**Additional resources**

To learn more details about ggplot2 and filtering with dplyr, check out these resources:

* [**Putting it all together: (dplyr+ggplot)**](https://rladiessydney.org/courses/ryouwithme/03-vizwhiz-1/#1-4-putting-it-all-together-dplyr-ggplot)**:** The RLadies of Sydney’s course on R uses real data to demonstrate R functions. This lesson focuses specifically on combining dplyr and ggplot to filter data before plotting it. The instructional video will guide you through every step in the process while you follow along with the data they have provided.
* [**Data transformation:**](https://r4ds.had.co.nz/transform.html)This resource focuses on how to use the filter() function in R, and demonstrates how to combine filter() with ggplot(). This is a useful resource if you are interested in learning more about how filter() can be used before plotting.
* [**Visualizing data with ggplot2:**](https://datacarpentry.org/dc_zurich/R-ecology/05-visualisation-ggplot2.html) This comprehensive guide includes everything from the most basic uses for ggplot2 to creating complicated visualizations. It includes the filter() function in most of the examples so you can learn how to implement it in R to create data visualizations.

### 1.

Question 1

Which of the following aesthetics attributes can you map to the data in a scatterplot? Select all that apply.

1 point

Shape

Size

Text

Color

### 2.

Question 2

Which of the following functions let you display smaller groups, or subsets, of your data?

1 point

geom\_bar()

geom\_point()

facet\_wrap()

ggplot()

### 3.

Question 3

What is the role of the x argument in the following code?

**ggplot(data = diamonds) + geom\_bar(mapping = aes(x = cut))**

1 point

A variable

A function

An aesthetic

A dataset

### 4.

Question 4

A data analyst creates a scatterplot with a lot of data points. It is difficult for the analyst to distinguish the individual points on the plot because they overlap. What function could the analyst use to make the points easier to find?

1 point

geom\_point()

geom\_line()

geom\_bar()

geom\_jitter()

**Adding annotations in R**



Annotations are a useful way to add notes to your plot. They help you explain the plot’s purpose, highlight important data points, or comment on any data trends or findings the plot illustrates. You have already learned how to add notes as labels, titles, subtitles, and captions. You can also draw arrows or add shapes to your plot to create more emphasis. Usually you add these kinds of annotations in your presentation application after you have saved the visualizations. But, you can now add lines, arrows, and shapes to your plots using **ggplot2**.

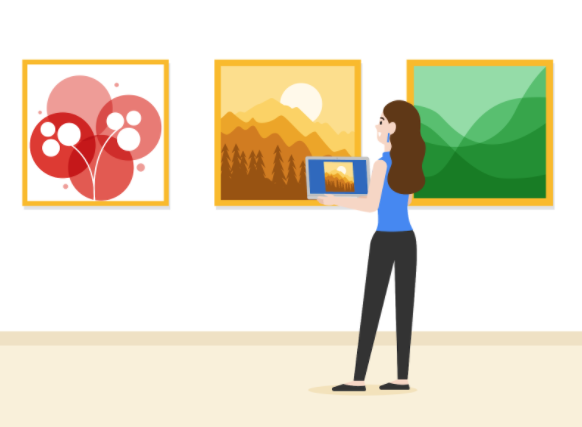
**Resources**

Check out these resources to learn more:

* [**Create an annotation layer**](https://ggplot2.tidyverse.org/reference/annotate.html): This guide explains how to add an annotation layer with ggplot2. It includes sample code and data visualizations with annotations created in ggplot2.
* [**How to annotate a plot in ggplot2**](https://www.r-graph-gallery.com/233-add-annotations-on-ggplot2-chart.html)**:** This resource includes explanations about how to add different kinds of annotations to your ggplot2 plots, and is a great reference if you need to quickly look up a specific kind of annotation.
* [**Annotations**](https://ggplot2-book.org/annotations.html)**:** Chapter eight of the online ggplot2 textbook is focused entirely on annotations. It provides in-depth explanations of the different types of annotations, how they are used, and detailed examples.
* [**How to annotate a plot**](https://www.r-bloggers.com/2017/02/how-to-annotate-a-plot-in-ggplot2/)**:** This R-Bloggers article includes explanations about how to annotate plots in ggplot2. It starts with basic concepts and covers more complicated information the further on you read.
* [**Text Annotations**](https://viz-ggplot2.rsquaredacademy.com/textann.html)**:** This resource focuses specifically on adding text annotations and labels to ggplot2 visualizations.

**Saving images without ggsave()**

In most cases, **ggsave()** is the simplest way to save your plot. But there are situations when it might be best to save your plot by writing it directly to a graphics device. This reading will cover some of the different ways you can save images and plots without ggsave(), and includes additional resources to check out if you want to learn more.



A graphics device allows a plot to appear on your computer. Examples include:

* A window on your computer (screen device)
* A PDF, PNG, or JPEG file (file device)
* An SVG, or scalable vector graphics file (file device)

When you make a plot in R, it has to be “sent” to a specific graphics device. To save images without using ggsave(), you can open an R graphics device like **png()** or **pdf()**; these will allow you to save your plot as a .png or .pdf file. You can also choose to print the plot and then close the device using **dev.off()**.

| **Example of using png()** | **Example of using pdf()** |
| --- | --- |
| **png(file = "exampleplot.png", bg = "transparent") plot(1:10) rect(1, 5, 3, 7, col = "white") dev.off()** | **pdf(file = "/Users/username/Desktop/example.pdf",     width = 4,      height = 4)  plot(x = 1:10,       y = 1:10) abline(v = 0) text(x = 0, y = 1, labels = "Random text") dev.off()** |

To learn more about the different processes for saving images, check out these resources:

* [**Saving images without ggsave()**](https://ggplot2.tidyverse.org/reference/ggsave.html#saving-images-without-ggsave-): This resource is pulled directly from the ggplot2 documentation at [**tidyverse.org**](https://www.tidyverse.org/). It explores the tools you can use to save images in R, and includes several examples to follow along with and learn how to save images in your own R workspace.
* [**How to save a ggplot**](https://www.datanovia.com/en/blog/how-to-save-a-ggplot/): This resource covers multiple different methods for saving ggplots. It also includes copyable code with explanations about how each function is being used so that you can better understand each step in the process.
* [**Saving a plot in R:**](https://www.datamentor.io/r-programming/saving-plot/) This guide covers multiple file formats that you can use to save your plots in R. Each section includes an example with an actual plot that you can copy and use for practice in your own R workspace.

### 1.

Question 1

Which of the following are benefits of adding labels and annotations to your plot? Select all that apply.

1 point

Indicating the main purpose of your plot

Helping stakeholders quickly understand your plot

Highlighting important data in your plot

Choosing a geom for your plot

### 2.

Question 2

A data analyst is creating a plot for a presentation to stakeholders. The analyst wants to add a caption to the plot to help communicate important information. What function could the analyst use?

1 point

The labs() function

The facet\_wrap() function

The geom\_bar() function

The geom\_point() function

### 3.

Question 3

What function can you use to put a text label inside the grid of your plot to call out specific data points?

1 point

The facet\_wrap() function

The labs() function

The aes() function

The annotate() function

### 4.

Question 4

You are working with the penguins dataset. You create a scatterplot with the following code:

**ggplot(data = penguins) +**

**geom\_point(mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g)) +**

You want to use the labs() function to add the title “Penguins” to your plot. Add the code chunk that lets you add the title "Penguins" to your plot.

1

RunReset

Where does your visualization display the title?

1 point

The lower right

The upper left

The upper right

The lower left

### 1.

Question 1

Which of the following statements about ggplot is true?

1 point

ggplot is designed to make cleaning data easy.

ggplot allows analysts to create different types of plots.

ggplot is the default plotting package in base R.

ggplot allows analysts to create plots using a single function.

### 2.

Question 2

Fill in the blank: In ggplot2, you use the \_\_\_\_\_ to add layers to your plot.

1 point

The plus sign (+)

The ampersand symbol (&)

The equals sign (=)

The pipe operator (%>%)

### 3.

Question 3

A data analyst creates a plot using the following code chunk:

**ggplot(data = buildings) +**

**geom\_bar(mapping = aes(x = construction\_year, color = height))**

Which of the following represents a variable in the code chunk?

1 point

mapping

construction\_year

data

ggplot

### 4.

Question 4

Which code snippet will make all of the bars in the plot purple?

1 point

**ggplot(data = buildings) +**

**geom\_bar(mapping = aes(x = construction\_year, color=”purple”))**

**ggplot(data = buildings) +**

**geom\_bar(mapping = aes(x = construction\_year, color=height))**

**ggplot(data = buildings) +**

**geom\_bar(mapping = aes(x = construction\_year)) +**

**color(“purple”)**

**ggplot(data = buildings) +**

**geom\_bar(mapping = aes(x = construction\_year), color=”purple”)**

### 5.

Question 5

The \_\_\_\_\_ aesthetic makes some points on a plot more transparent than others.

1 point

fill

linetype

alpha

color

### 6.

Question 6

You are working with the penguins dataset. You create a scatterplot with the following code chunk:

**ggplot(data = penguins) +**

**geom\_point(mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g))**

You want to highlight the different years of data collection on your plot. Add a code chunk to the second line of code to map the aesthetic alpha to the variable island.

NOTE: the three dots (...) indicate where to add the code chunk. You may need to scroll in order to find the dots.

1

geom\_point(mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g, ...))

RunReset

What islands does your visualization display?

1 point

Cebu, Borneo, Torgersen

Biscoe, Java, Buton

Biscoe, Dream, Torgersen

Cebu, Java, Hispaniola

### 7.

Question 7

Which aesthetic of the geom\_smooth function can be used to change the style of the line?

1 point

line

linetype

linelook

linestyle

### 8.

Question 8

What function can be used to facet a plot on two variables?

1 point

**facet\_grid()**

**geom\_wrap()**

**facet\_wrap()**

**facet\_layout()**

### 9.

Question 9

A data analyst uses the **annotate()** function to create a text label for a plot. Which attributes of the text can the analyst change by adding code to the argument of the **annotate()** function? Select all that apply.

1 point

Change the size of the text.

Change the color of the text.

Change the font style of the text.

Change the text into a title for the plot.

### 10.

Question 10

In R studio, what default options does the Export functionality of the Plots tab give for exporting plots?

1 point

PDF

Slideshow

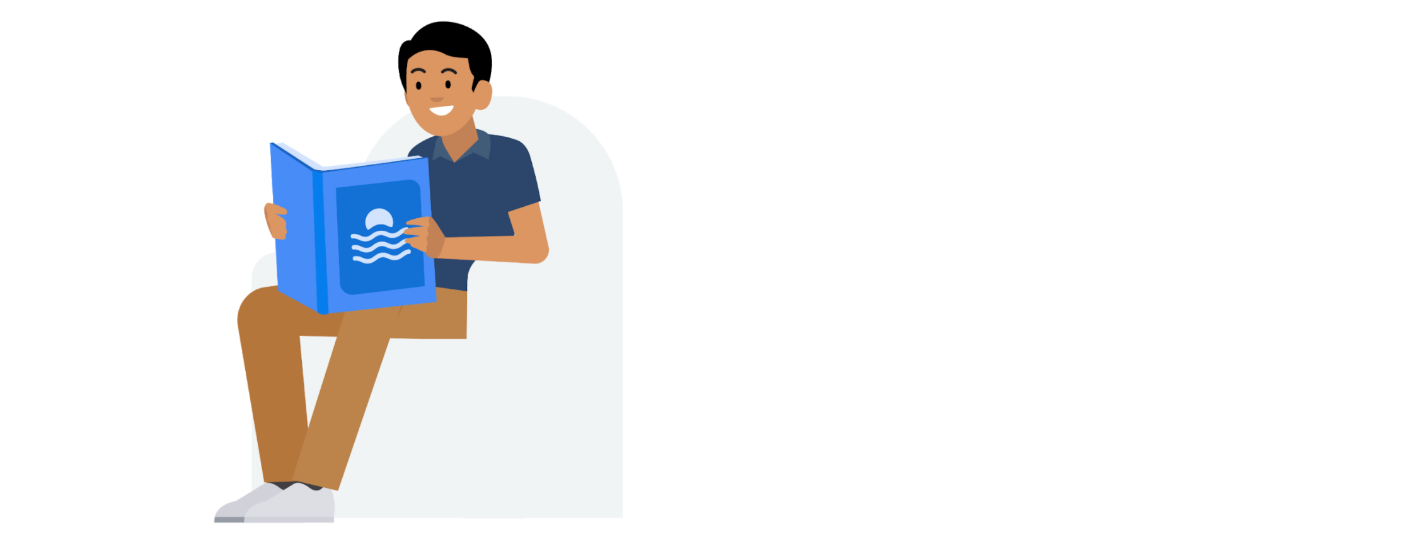
Image

HTML

**R Markdown resources**

**R Markdown** is a useful tool that allows you to save and execute code, and generate shareable reports for stakeholders. As you learn more about how to use it, it can be helpful to bookmark some resources to refer to later.

This reading explores some great online resources that will help you learn more about R Markdown and how to use it to document your analysis.



**R Markdown documentation**

RStudio's [R Markdown documentation](https://rmarkdown.rstudio.com/lesson-1.html) includes a series of tutorials that will help you learn about the main features of R Markdown, including code chunks, output formats, notebooks, interactive documents, and more. The tutorials include online lessons that you can complete directly in your RStudio Cloud workspace.

**R Markdown reference materials**

RStudio has developed a reference guide and a cheat sheet that you can bookmark and use whenever you practice writing R Markdown files.

* The [R Markdown Reference Guide](https://rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf?_ga=2.49295910.1034302809.1602760608-739985330.1601281773) contains three sections: Markdown syntax, knitr chunk options, and Pandoc options. The guide is super detailed and includes tons of examples and explanations so that you can easily find the exact information you need to customize your R Markdown documents.
* The [R Markdown Cheat Sheet](https://rmarkdown.rstudio.com/lesson-15.html) is a convenient summary of the different steps and workflow processes for R. It also includes sections with abbreviated explanations of knitr and pandoc chunk options, and other useful information to review or look up while you work.

**R for Data Science book**

For a well-organized introduction to the basics of R Markdown, check out the [Communicate](https://r4ds.had.co.nz/communicate-intro.html) section of the **R for Data Science** book. It covers the main features and functions of R Markdown, the various output formats, and the workflow for combining text and code to create an analysis notebook.

**R Markdown: The Definitive Guide**

If you want to really explore the capabilities of R Markdown in a systematic way, [R Markdown: The Definitive Guide](https://bookdown.org/yihui/rmarkdown/) provides a comprehensive guide to the R Markdown ecosystem. This book contains four main parts:

1. [Part I](https://bookdown.org/yihui/rmarkdown/installation.html) explains how to install the relevant packages and offers an overview of R Markdown, including the syntax for Markdown and code chunks.
2. [Part II](https://bookdown.org/yihui/rmarkdown/documents.html) provides detailed documentation of the built-in output formats included in R Markdown, like document formats and presentation formats.
3. [Part III](https://bookdown.org/yihui/rmarkdown/dashboards.html) shares several R Markdown extension packages that allow you to build different applications or generate output documents with different styles.
4. [Part IV](https://bookdown.org/yihui/rmarkdown/parameterized-reports.html) covers advanced topics in R Markdown.

**Optional: Jupyter notebooks**



**Jupyter notebooks** are documents that contain computer code and rich text elements – such as comments, links, or descriptions of your analysis and results. You will find them used in a variety of online tools, including Project Jupyter, Kaggle, and Google Colaboratory ("Colab" for short). These notebooks can be executable documents that you can run to perform an analysis.

Jupyter notebooks can come in handy with everything from data cleaning and transformation, to statistical modeling and visualizations. They are compatible with R, so you can consider them as an alternative to R Markdown. And just like R Markdown documents, you can easily share Jupyter notebooks with team members and stakeholders.

**Jupyter notebooks in Kaggle**

If you are working in Kaggle, there are two types of notebooks available: Jupyter notebooks and scripts (including R Markdown scripts). For more information, refer to the [How to Use Kaggle Notebooks](https://www.kaggle.com/docs/notebooks) page.

**Jupyter notebooks in Google Colab**

Google Colab is a product from Google Research. Colab is a hosted Jupyter notebook service that requires no setup to use. For more information, refer to the [Welcome to Colaboratory](https://colab.research.google.com/notebooks/intro.ipynb) page.

**Additional resources**

To learn more about Jupyter notebooks, check out these resources:

* [**Project Jupyter**](https://jupyter.org/): This is the home of Jupyter notebooks, as well as JupyterLab – the web-based interactive development environment for Jupyter notebooks, code, and data.
* [**Jupyter Notebook: An Introduction**](https://realpython.com/jupyter-notebook-introduction/): This detailed introduction of Jupyter notebooks comes from the people at Real Python, a tutorial-based site devoted to all things Python. You can take a video course or follow the written tutorial to get started with Jupyter notebooks and learn about its features and capabilities.

And, just like R Markdown, Jupyter notebooks include basic formatting tools and rules that will help you keep your work organized and user-friendly. In fact, Jupyter uses R Markdown as its language for writing and formatting text in a notebook.

To learn about basic formatting in Jupyter notebooks, check out these resources:

* [**The Jupyter** **Notebook**](https://jupyter-notebook.readthedocs.io/en/stable/notebook.html)**:** This resource provides an overview of Jupyter notebooks, including information about the structure of the user interface and notebook document. You’ll also learn about the basic workflow for using a notebook document, along with information about keyboard shortcuts and other features that will help you format your work.
* [**Using Jupyter Notebook for Writing**](https://gtribello.github.io/mathNET/assets/notebook-writing.html)**:** This resource focuses on how to use Markdown to format your writing in a Jupyter notebook. Use this as a guide to manage the syntax of your writing, including making titles and subtitles and adding links.
* [**The Jupyter Notebook Formatting Guide**](https://medium.com/analytics-vidhya/the-jupyter-notebook-formatting-guide-873ab39f765e): This resource includes a wide variety of formatting options for Jupyter notebooks. You’ll learn about the basics as well as some more advanced options, like embedding PDF documents and videos.

After you know how to apply basic formatting to your notebooks, you can start exploring more advanced options.

# Output formats in R Markdown

You can save this reading for future reference. Feel free to download a PDF version of this reading below:

[Output formats available in RMarkdown.pdf](https://d3c33hcgiwev3.cloudfront.net/HMmvfrAKSl6Jr36wCjpejg_f7ff98b532974cfca3bdea0be6731e4f_Output-formats-available-in-RMarkdown.pdf?Expires=1706918400&Signature=ccb17DocKNnUSR54KbY6pPPFeGnYm-MS-zOWpfipoZlHNFJhRUVuXcH~jT3p2iDORee5ZAoCuboHqQPXjoYvdSsCeW~F8C5CQ8Kdi4btW4eAUl0YZom9MPHxc89Tebs1We~MB0De4h5jGUPtNFT3-tKpWVhW91XJPS-EkZdbLiw_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[PDF File](https://d3c33hcgiwev3.cloudfront.net/HMmvfrAKSl6Jr36wCjpejg_f7ff98b532974cfca3bdea0be6731e4f_Output-formats-available-in-RMarkdown.pdf?Expires=1706918400&Signature=ccb17DocKNnUSR54KbY6pPPFeGnYm-MS-zOWpfipoZlHNFJhRUVuXcH~jT3p2iDORee5ZAoCuboHqQPXjoYvdSsCeW~F8C5CQ8Kdi4btW4eAUl0YZom9MPHxc89Tebs1We~MB0De4h5jGUPtNFT3-tKpWVhW91XJPS-EkZdbLiw_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

This reading will explore the different types of output formats you can produce with R Markdown.

## Setting the output of an R Markdown document

When working in RStudio, you can set the output of a document in R Markdown by changing the YAML header.

For example, the following code creates an HTML document:

---

title: "Demo"

output: html\_document

---

And the following code creates a PDF document:

---

title: "Demo"

output: pdf\_document

---

The **Knit** button in the RStudio source editor renders a file to the first format listed in its output field (HTML is the default). You can render a file to additional formats by clicking the dropdown menu next to the knit button.

A screenshot of a computer

Description automatically generated

## Available document outputs

In addition to the default HTML output (**html\_document**), you can create other types of documents in R Markdown using the following output settings:

* **pdf\_document** – This creates a PDF file with LaTeX (an open source document layout system). If you don’t already have LaTeX, RStudio will automatically prompt you to install it.
* **word\_document** – This creates a Microsoft Word document (.docx).
* **odt\_document** – This creates an OpenDocument Text document (.odt).
* **rtf\_document** – This creates a Rich Text Format document (.rtf).
* **md\_document** – This creates a Markdown document (which strictly conforms to the original Markdown specification)
* **github\_document** – This creates a GitHub document which is a customized version of a Markdown document designed for sharing on GitHub.

For a detailed guide to creating different types of R Markdown documents, check out the [Documents](https://bookdown.org/yihui/rmarkdown/documents.html) chapter in R Markdown: The Definitive Guide.

### **Notebooks**

A **notebook** (html\_notebook) is a variation on an HTML document (html\_document). Overall, the output formats are similar; the main difference between them is that the rendered output of a notebook always includes an embedded copy of the source code.

Notebooks and HTML documents also have different purposes. HTML documents are good for communicating with stakeholders. Notebooks are better for collaborating with other data analysts or data scientists.

To learn more, check out the section on [Notebooks](https://rmarkdown.rstudio.com/lesson-10.html) in the R Markdown documentation.

### **Presentations**

You can also use R Markdown to produce presentations. Automatically inserting the results of your R code into a presentation can save you lots of time.

R Markdown renders files to specific presentation formats when you use the following output settings:

* **beamer\_presentation** – for PDF presentations with beamer
* **ioslides\_presentation** – for HTML presentations with ioslides
* **slidy\_presentation** – for HTML presentations with Slidy
* **powerpoint\_presentation** – for PowerPoint presentations
* **revealjs : : revealjs\_presentation** – for HTML presentations with reveal.js (a framework for creating HTML presentations that requires the reveal.js package)

 To learn more, check out the section on [Slide Presentations](https://rmarkdown.rstudio.com/lesson-11.html) in the R Markdown documentation.

### **Dashboards**

Dashboards are a useful way to quickly communicate a lot of information. The[flexdashboard](https://github.com/rstudio/flexdashboard) package lets you publish a group of related data visualizations as a dashboard. Flexdashboard also provides tools for creating sidebars, tabsets, value boxes, and gauges.

To learn more, visit the [flexdashboard for R](https://rmarkdown.rstudio.com/flexdashboard/) page and the [Dashboards](https://rmarkdown.rstudio.com/lesson-12.html) section in the R Markdown documentation.

### **Shiny**

**Shiny** is an R package that lets you build interactive web apps using R code. You can embed your apps in R Markdown documents or host them on a webpage.

To call Shiny code from an R Markdown document, add  runtime: shiny to the YAML header:

---

title: "Shiny Web App"

output: html\_document

runtime: shiny

---

To learn more about Shiny and how to use R code to add interactive components to an R Markdown document, check out the [Shiny](https://shiny.rstudio.com/tutorial/) tutorial from RStudio.

### **Other formats**

Other packages provide even more output formats:

* The [bookdown](https://github.com/rstudio/bookdown) package is helpful for writing books and long-form articles.
* The [prettydoc](https://github.com/yixuan/prettydoc/) package provides a range of attractive themes for R Markdown documents.
* The[rticles](https://github.com/rstudio/rticles)package provides templates for various journals and publishers.

 Visit the [RStudio Formats](https://rmarkdown.rstudio.com/formats.html) page in the R Markdown documentation for a more comprehensive list of output formats and packages.

## Additional resources

For more information, check out these additional resources:

* The [R Markdown gallery](https://rmarkdown.rstudio.com/gallery.html) from RStudio has tons of examples of the outputs you can create with R Markdown.
* The [R Markdown Formats](https://r4ds.had.co.nz/r-markdown-formats.html) chapter in the R for Data Science book provides more details about the output formats introduced in this reading. This reading was compiled from information in this book.

### 1.

Question 1

R Markdown is a file format for making dynamic documents with R. What are the benefits of creating this kind of document? Select all that apply.

1 point

Save, organize, and document code

Generate a report with executable code chunks

Perform calculations for analysis more efficiently

Create a record of your cleaning process

### 2.

Question 2

A data analyst needs to create a shareable report in RStudio. They first want to change the default file format that gets exported by the Knit button to .pdf. What value should they use for the output field in the YAML header?

1 point

**document\_pdf**

**pdf\_document**

**pdf\_knit**

**knit\_pdf**

### 3.

Question 3

Which code snippet implements the correct syntax for writing a piece of hyperlinked text in markdown?

1 point

**(link)[www.coursera.com]**

**link\*\*www.coursera.com**

**[link](www.coursera.com )**

**>link–www.cousera.com<**

### 4.

Question 4

Fill in the blank: \_\_\_\_\_ code is code that can be inserted directly into a .rmd file.

1 point

Inline

YAML

Executable

Markdown

### 5.

Question 5

A data analyst wants to add a bulleted list to their R Markdown document. What symbol can they type to create this formatting?

1 point

Hashtags

Brackets

Delimiters

Asterisks

### 6.

Question 6

A data analyst is working in a .rmd file and comes across the text **```{r analysis}**. What is the purpose of the text “analysis”?

1 point

It is a label for the code chunk

It alters the output file format of Knit

It changes the way the code gets exported

It runs the code in analysis mode

### 7.

Question 7

What does the **---** delimiter (three hyphens) indicate in an R Markdown notebook?

1 point

Code chunk

Bold text

Italic text

YAML metadata

### 8.

Question 8

Why would a data analyst create a template of their .rmd file? Select all that apply.

1 point

To save time when creating the same kind of document

To customize the appearance of a final report

To create an interactive notebook

To prevent other users from editing the file

### 1.

Question 1

**Scenario 1, questions 1-7**

As part of the data science team at Gourmet Analytics, you use data analytics to advise companies in the food industry. You clean, organize, and visualize data to arrive at insights that will benefit your clients. As a member of a collaborative team, sharing your analysis with others is an important part of your job.

Your current client is Chocolate and Tea, an up-and-coming chain of cafes.



The eatery combines an extensive menu of fine teas with chocolate bars from around the world. Their diverse selection includes everything from plantain milk chocolate, to tangerine white chocolate, to dark chocolate with pistachio and fig. The encyclopedic list of chocolate bars is the basis of Chocolate and Tea’s brand appeal. Chocolate bar sales are the main driver of revenue.

Chocolate and Tea aims to serve chocolate bars that are highly rated by professional critics. They also continually adjust the menu to make sure it reflects the global diversity of chocolate production. The management team regularly updates the chocolate bar list in order to align with the latest ratings and to ensure that the list contains bars from a variety of countries.

They’ve asked you to collect and analyze data on the latest chocolate ratings. In particular, they’d like to know which countries produce the highest-rated bars of super dark chocolate (a high percentage of cocoa). This data will help them create their next chocolate bar menu.

Your team has received a dataset that features the latest ratings for thousands of chocolates from around the world. Click [here](https://www.kaggle.com/rtatman/chocolate-bar-ratings) to access the dataset. Given the data and the nature of the work you will do for your client, your team agrees to use R for this project.

**Your supervisor asks you to write a short summary of the benefits of using R for the project. Which of the following benefits would you include in your summary? Select all that apply.**

1 point

Define a problem and ask the right questions

Quickly process lots of data

Create high-quality data visualizations

Easily reproduce and share the analysis

### 2.

Question 2

**Scenario 1, continued**

Before you begin working with your data, you need to import it and save it as a data frame. To get started, you open your RStudio workspace and load the tidyverse library. You upload a .csv file containing the data to RStudio and store it in a project folder named flavors\_of\_cacao.csv.

**You use the read\_csv() function to import the data from the .csv file. Assume that the name of the data frame is chocolate\_df and the .csv file is in the working directory. What code chunk lets you create the data frame?**

1 point

**chocolate\_df <-read\_csv("flavors\_of\_cacao.csv")**

**chocolate\_df  <- "flavors\_of\_cacao.csv"(read\_csv)**

**read\_csv("flavors\_of\_cacao.csv") + chocolate\_df**

**chocolate\_df + read\_csv("flavors\_of\_cacao.csv")**

### 3.

Question 3

**Scenario 1, continued**

Now that you’ve created a data frame, you want to find out more about how the data is organized. The data frame has hundreds of rows and lots of columns.

**Assume the name of your data frame is flavors\_df. What code chunk lets you review the column names in the data frame?**

1 point

**rename(flavors\_df)**

**arrange(flavors\_df)**

**col(flavors\_df)**

**colnames(flavors\_df)**

### 4.

Question 4

**Scenario 1, continued**

Next, you begin to clean your data. When you check out the column headings in your data frame you notice that the first column is named Company...Maker.if.known. (Note: The period after known is part of the variable name.) For the sake of clarity and consistency, you decide to rename this column Company (without a period at the end).

**Assume the first part of your code chunk is:**

**flavors\_df %>%**

**What code chunk do you add to change the column name?**

1 point

**rename(Company...Maker.if.known. <- Company)**

**rename(Company...Maker.if.known. = Company)**

**rename(Company <- Company...Maker.if.known.)**

**rename(Company = Company...Maker.if.known.)**

### 5.

Question 5

After previewing and cleaning your data, you determine what variables are most relevant to your analysis. Your main focus is on Rating, Cocoa.Percent, and Company.Location. You decide to use the select() function to create a new data frame with only these three variables.

**Assume the first part of your code is:**

**trimmed\_flavors\_df <- flavors\_df %>%**

**Add the code chunk that lets you select the three variables.**

1

RunReset

What company location appears in row 1 of your tibble?

1 point

Scotland

France

Colombia

Canada

### 6.

Question 6

Next, you select the basic statistics that can help your team better understand the ratings system in your data.

**Assume the first part of your code is:**

**trimmed\_flavors\_df %>%**

**You want to use the summarize() and max() functions to find the maximum rating for your data. Add the code chunk that lets you find the maximum value for the variable *Rating.***

1

RunReset

What is the maximum rating?

1 point

6

5

4.5

5.5

### 7.

Question 7

After completing your analysis of the rating system, you determine that any rating greater than or equal to 3.75 points can be considered a high rating. You also know that Chocolate and Tea considers a bar to be super dark chocolate if the bar's cocoa percentage is greater than or equal to 80%. You decide to create a new data frame to find out which chocolate bars meet these two conditions.

**Assume the first part of your code is:**

**best\_trimmed\_flavors\_df <- trimmed\_flavors\_df %>%**

**You want to apply the filter() function to the variables *Cocoa.Percent* and *Rating*. Add the code chunk that lets you filter the new data frame for chocolate bars that contain at least 80% cocoa and have a rating of at least 3.75 points.**

1

RunReset

How many rows does your tibble include?

1 point

12

22

8

20

### 8.

Question 8

Now that you’ve cleaned and organized your data, you’re ready to create some useful data visualizations. Your team assigns you the task of creating a series of visualizations based on requests from the Chocolate and Tea management team. You decide to use ggplot2 to create your visuals.

**Assume your first line of code is:**

**ggplot(data = best\_trimmed\_flavors\_df) +**

**You want to use the geom\_bar() function to create a bar chart. Add the code chunk that lets you create a bar chart with the variable *Company* on the x-axis.**

1

RunReset

How many bars does your bar chart display?

1 point

8

6

10

4

### 9.

Question 9

Your bar chart reveals the locations that produce the highest-rated chocolate bars. To get a better idea of the specific rating for each location, you’d like to highlight each bar.

**Assume that you are working with the code chunk:**

**ggplot(data = best\_trimmed\_flavors\_df) +**

**geom\_bar(mapping = aes(x = Company.Location))**

**Add a code chunk to the second line of code to map the aesthetic *alpha* to the variable *Rating*.**

**NOTE: the three dots (...) indicate where to add the code chunk.**

1

geom\_bar(mapping = aes(x = Company.Location, ...))

RunReset

**According to your bar chart, which two company locations produce the highest rated chocolate bars?**

1 point

Canada and Amsterdam

Scotland and Amsterdam

U.S.A. and France

Canada and France

### 10.

Question 10

**Scenario 2, continued**

A teammate creates a new plot based on the chocolate bar data. The teammate asks you to make some revisions to their code.

**Assume your teammate shares the following code chunk:**

**ggplot(data = best\_trimmed\_flavors\_df) +**

**geom\_bar(mapping = aes(x = Cocoa.Percent)) +**

**What code chunk do you add to the third line to create wrap around facets of the variable *Cocoa.Percent*?**

1 point

**facet\_wrap(~Cocoa.Percent)**

**facet\_wrap(%>%Cocoa.Percent)**

**facet(=Cocoa.Percent)**

**facet\_wrap(Cocoa.Percent~)**

### 11.

Question 11

**Scenario 2, continued**

Your team has created some basic visualizations to explore different aspects of the chocolate bar data. You’ve volunteered to add titles to the plots. You begin with a scatterplot.

**Assume the first part of your code chunk is:**

**ggplot(data = trimmed\_flavors\_df) +**

**geom\_point(mapping = aes(x = Cocoa.Percent, y = Rating)) +**

**What code chunk do you add to the third line to add the title *Best Chocolates* to your plot?**

1 point

**labs(title = “Best Chocolates”)**

**labs("Best Chocolates" = title)**

**labs(title <- "Best Chocolates")**

**labs("Best Chocolates")**

### 12.

Question 12

**Scenario 2, continued**

Next, you create a new scatterplot to explore the relationship between different variables. You want to save your plot so you can access it later on. You know that the ggsave() function defaults to saving the last plot that you displayed in RStudio, so you’re ready to write the code to save your scatterplot.

**Assume your first two lines of code are:**

**ggplot(data = trimmed\_flavors\_df) +**

**geom\_point(mapping = aes(x = Cocoa.Percent, y = Rating))**

**What code chunk do you add to the third line to save your plot as a png file with *chocolate* as the file name?**

1 point

**ggsave(chocolate.png)**

**ggsave(“chocolate.png”)**

**ggsave(“chocolate”)**

**ggsave(“png.chocolate”)**

### 13.

Question 13

**Scenario 2, continued**

As a final step in the analysis process, you create a report to document and share your work. Before you share your work with the management team at Chocolate and Tea, you are going to meet with your team and get feedback. Your team wants the documentation to include all your code and display all your visualizations.

**You want to record and share every step of your analysis, let teammates run your code, and display your visualizations. What do you use to document your work?**

1 point

An R Markdown notebook

A database

A spreadsheet

A data frame