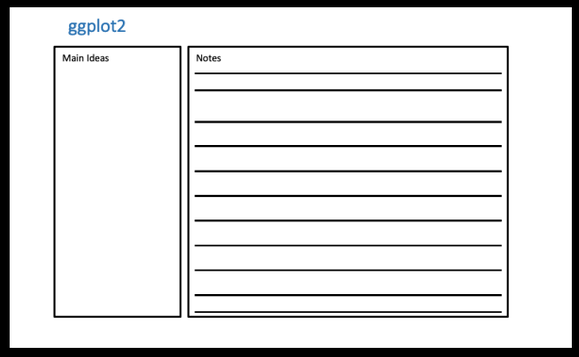
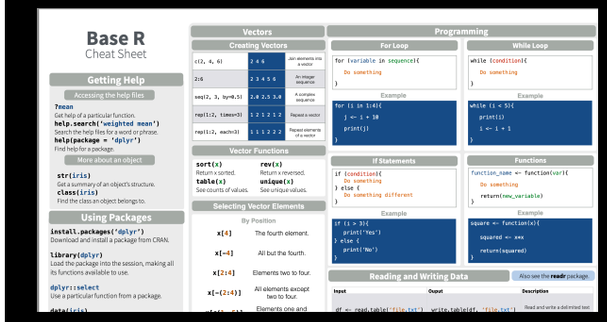
Exercise: What do you already know?

- Split into Groups of 2
- Ask you partner:
  - *What do you already know about vectors in R?*
  - *What do you already know about Data Frames in R?*





Notes form & Cheat Sheet

Please take out

There are two documents that I'd like you to take out before we start. The first is the notes form I gave you.

Data Structures in Base R

Vectors and Data Frames

There's a lot of confusion right now about the "Best" way to teach R. So far we've focused on the Tidyverse. This is great for getting people to do cool things (like create graphs) right off the bat. The downside is that people often feel confused about the "basics". Hopefully this workshop will help with that.

Your Turn

Click on **data-structures.Rmd**

Agenda

- Variables & Environments
- Vectors
- Data Frames

Quiz - Variables

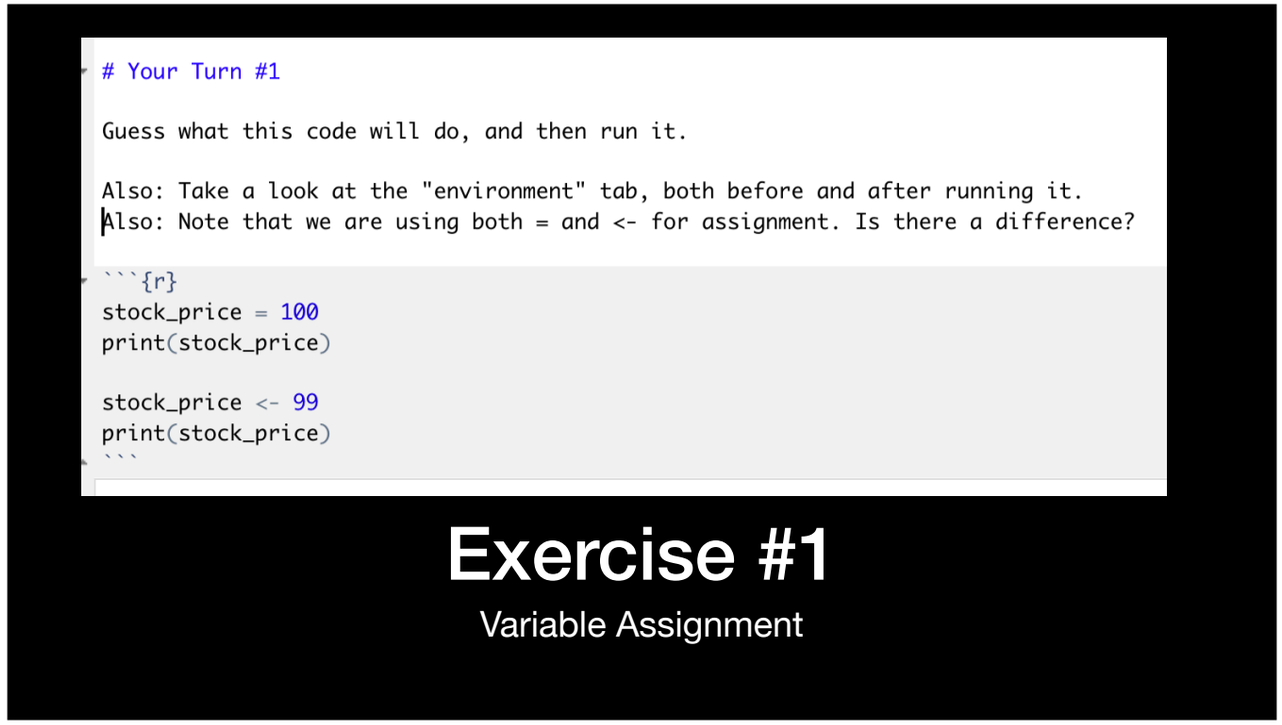
```
stock_price = 100
print(stock_price)
```

What does this code do?

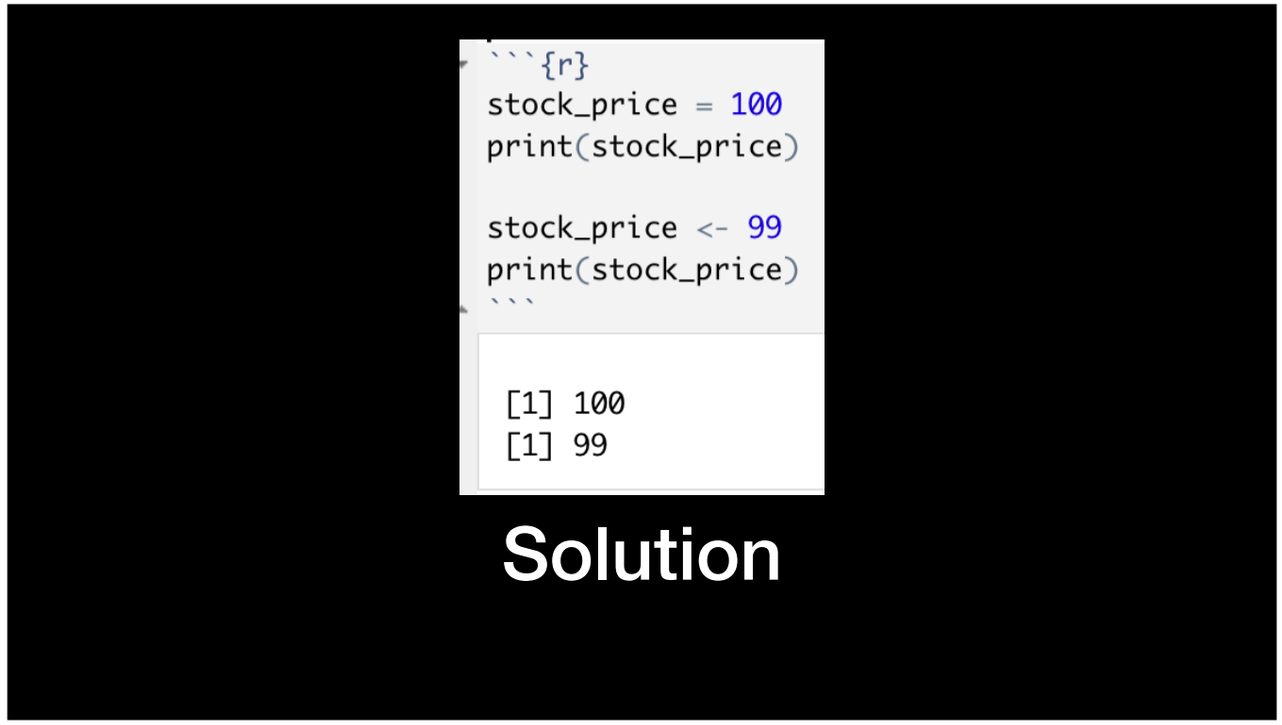
```
> stock_price = 100
> print(stock_price)
[1] 100
```

Variable Assignment

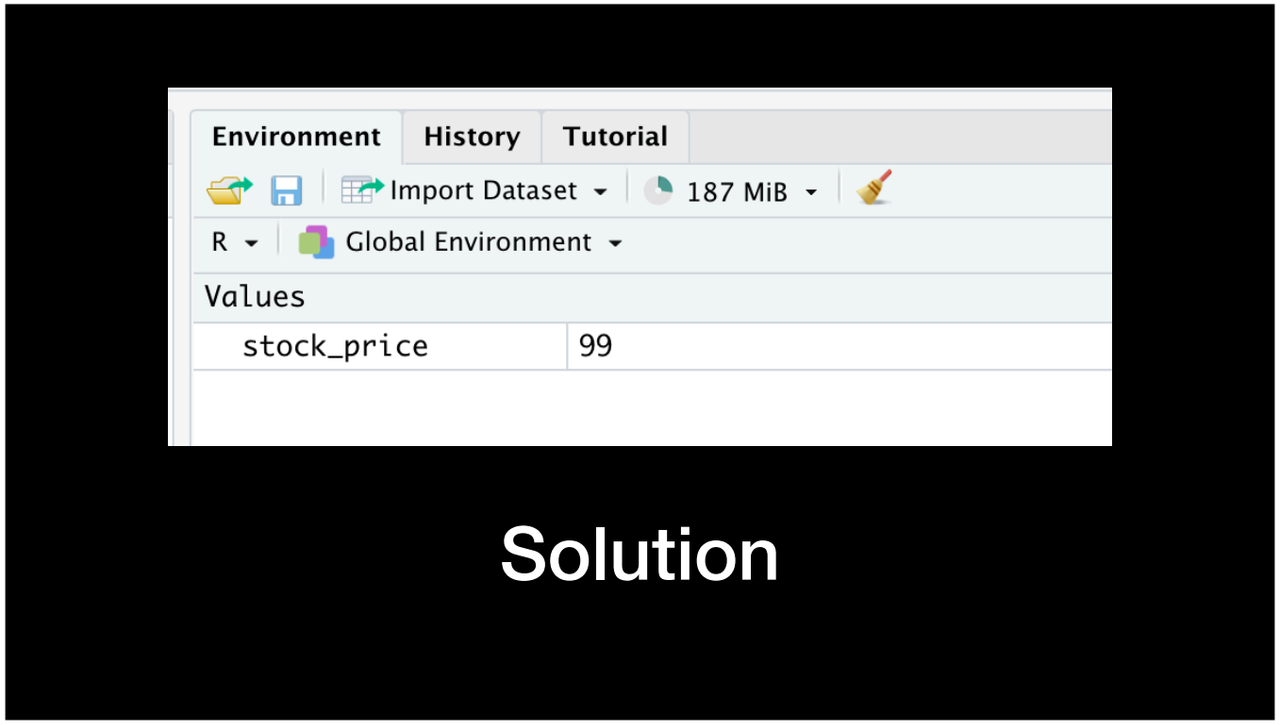
We assign the value 100 to the name stock_price. And then when we print stock_price, R prints the value that we assigned to it.



This is an exercise about variable assignment.
Tell your partner your answers to these questions, and then run the code

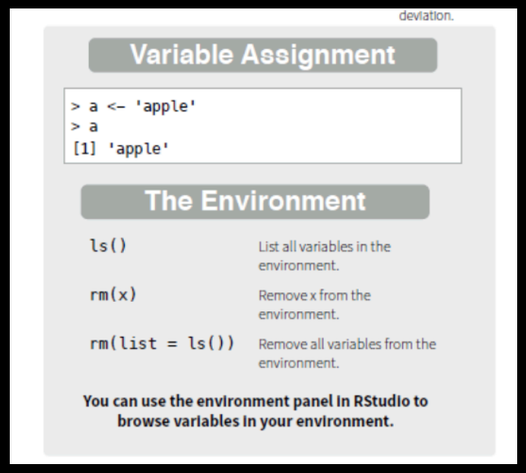


A variable is just a name that refers to a value.
The value can change (vary) over time, which is why we call it a "variable". You can assign a value to a variable with both = and <-. They do the same thing, and it's really just a matter of style.



Solution

The "Environment" tab lists all the variables in your current R session. You will often use variables to store values in your analyses. It is useful to see what is currently defined in one place. Note the "broom" icon, which deletes all variables.



Cheat Sheet

Please take a moment to look at your cheat sheet. We just covered two sections of it: Variable Assignment and The Environment

Agenda

- Variables & Environments
- Vectors
- Data Frames

As a reminder, I'm not going to cover everything there is to know about vectors - it's a surprisingly big topic. But I will be giving an overview - with plenty of exercises - over what I consider to be a few of the most important topics.

Vectors

- Creating Vectors
- Selecting Vector Elements
- Vectorization

We're going to start by learning how to create a vector, and why you would want to create one at all.

Quiz: What's wrong with this picture?

```
stock_price_today = 100
stock_price_yesterday = 99
stock_price_2_days_ago = 98
...
```

Let's start by talking about the motivation for vectors. What problem do they solve?
What are the pros and cons of writing code like this?
(My answer is on the next slide)

Quiz: What's wrong with this picture?

```
stock_price_today = 100
stock_price_yesterday = 99
stock_price_2_days_ago = 98
...
```

1. What happens if I get new data tomorrow? I need to rename all the variables.
2. What if I'm analyzing years of stock prices? Potentially thousands of variables.

Vectors

```
> stock_price_history = c(98, 99, 100)
> stock_price_history
[1]  98  99 100
```

- **Vectors let you assign multiple values to single name**
- Create a vector with c: c(item1, item2, ...)

*All values in a vector must be of the same type. I.e. can't mix numbers and text).
*A vector can also have just one value.
?c

Your Turn #2

Create a vector called years that contains 3 numbers:

1. The year Christopher Columbus sailed the ocean blue
2. The year America became an independent country
3. The year you started working at MarketBridge

Then print the vector to the console

```
> years = c(1492, 1776, 2021)
> print(years)
[1] 1492 1776 2021
```

My Answer

```
> stock_price_history = c(
+   "Jan" = 98,
+   "Feb" = 99,
+   "Mar" = 100)
> print(stock_price_history)
Jan Feb Mar
 98  99 100
```

"Named" Vector

Frequently used in tidyverse

R also lets you created a "named" vector. Named vectors are identical to normal vector. The only difference is that each value in the vector has a "name" that appears above the value. You create a named vector by using the = when you create the vector.

I believe that quotes are the names is options, but I like to use them to clarify that they are not the name of a variable.

Your Turn #3

Create a vector called years_named that:

1. Contains the exact same data as `years`.
2. Each element should have a name that describes what happened that year.

Then print out the vector.

```
> years_named = c("columbus" = 1492, "usa" = 1776, "mb" = 2001)
> print(years_named)
columbus      usa       mb
    1492     1776     2001
```

My Solution

Vectors

- Creating Vectors
- Selecting Vector Elements
- Vectorization

```
words =      c("my", "name", "is", "Ari")
# index        1       2     3     4
```

"Index"

Starts at 1

You sometimes need to access or modify a single element of a vector. To do this, you need to be able to tell R which element of the vector you are referring to.

Each element of a vector has a unique index. The index of the first element is always 1. The index of the second element is 2, and so on.

```
words = c("My", "name", "is", "Ari")
print(words[1])
```

words[1]

"Return the element with index 1"

In R, you use brackets to select an element of a vector by its index. Here I'm printing out the element of the vector words with index 1.
What does this code return?

```
> words = c("My", "name", "is", "Ari")
> print(words[1])
[1] "My"
```

It returns "My", because the element of vector "words" with index 1 is "My".

Changing elements by index

```
words = c("my", "name", "is", "Ari")
words[1] = "Your"
print(words)
```

You can also change the value of a single element in a vector. To do this, refer to the element by its index using the bracket notation. Put that on the left hand side of the equation, and put the new value you want on the right hand side of of the equation.

What does this program print out?

Changing elements by index

```
> words = c("my", "name", "is", "Ari")
> words[1] = "Your"
> print(words)
[1] "Your" "name" "is"   "Ari"
```

We changed the first word of the vector to be "Your".

Your Turn #4 - Vector Indexing

- Given a vector of words:
  1. Change the first word
  2. Create a new vector that rearranges the words into a question

```
> print(words)
[1] "Your" "name" "is"   "Ari"
> print(words_question)
[1] "is"   "Your" "name" "Andy"
```

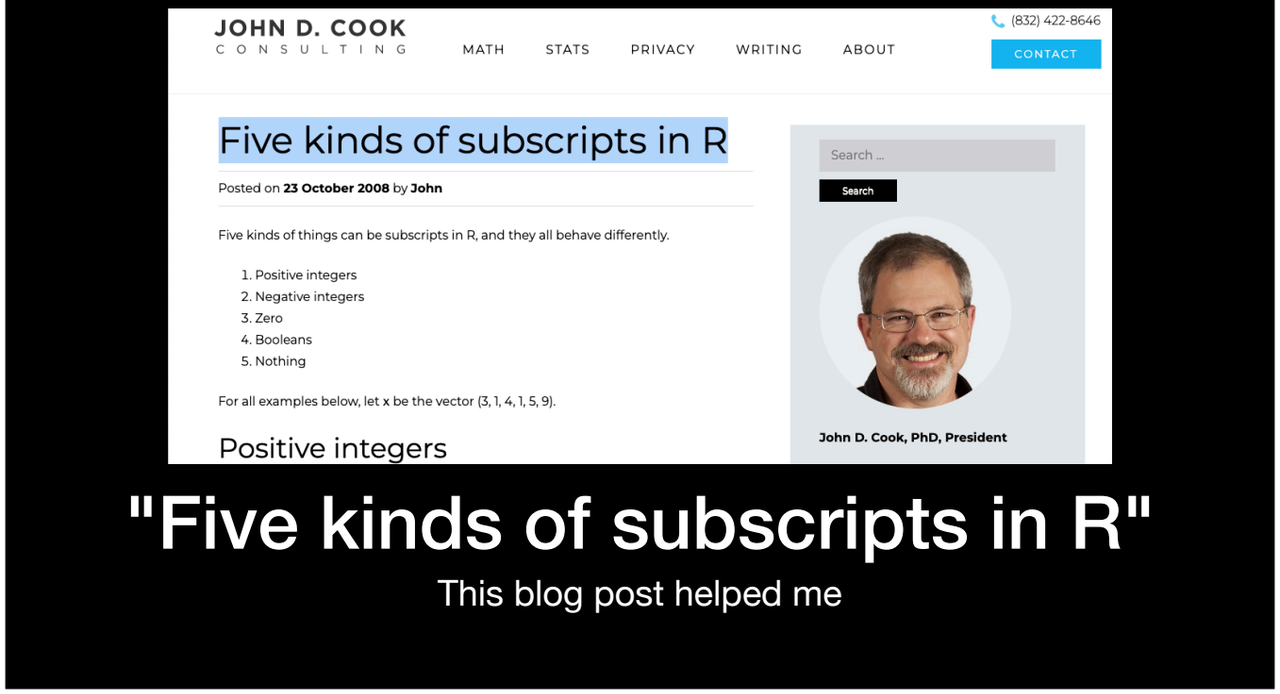
I gave you a vector of words to start with. Your goal is to create a new vector, words_question, that reads "is Your name <your name>".

The goal here is to practice working with indexes. There are lots of ways to solve this problem. To clarify how I want you to solve it (in the way which maximizes your practice with indexes), I've already written portions of the program.

```
> words = c("Your", "name", "is", "Ari")
> words[4] = "Andy"
> print(words)
[1] "Your" "name" "is"   "Andy"
> words_question = c(words[3], words[1], words[2], words[4])
> print(words_question)
[1] "is"   "Your" "name" "Andy"
```

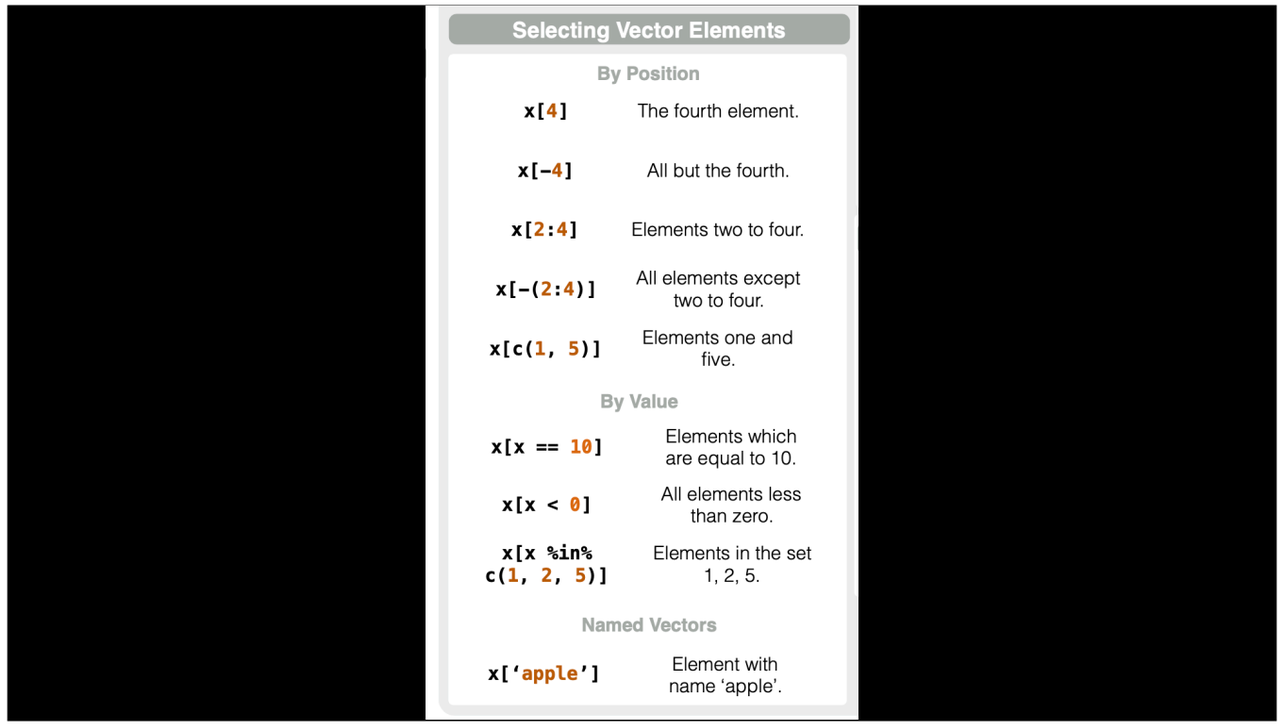
My solution

The goal here was for you to get practice changing the value of an index by using its subscript.
Also, I wanted you to see that you can create a new vector just by getting elements of another vector one by one.



This might surprise you, but subscripting vectors in R is actually a huge topic. In ths workshop we're just scractching the surface of the topic, and that's by design. I want you to get some quick wins with an important topic.

If you want to go deeper into this topic, then just google the phrase "Rive kinds of subscripts in R". This blog post, by John D. Cook, really helped me get fluent with subscripts, and I think it can help you as well.



The cheatsheet I gave you also shows you other possibilities for subscripting.

- -Named vectors (easy)
- -multiple indexes (easy)
- -"By value" (hard - see blog post)

Vectors

- Creating Vectors
- Selecting Vector Elements
- Vectorization

Vectorization

- Definition: "*A single line of code that does something to every element of a vector.*"
- Special feature of R
- Used all the time

```
> c(1, 2, 3) * 2
[1] 2 4 6
```

```r
stock_prices_usd   = c(98, 99, 100)
euro_exchange_rate = 0.95
stock_prices_euro  = stock_prices_usd * euro_exchange_rate

print(stock_prices_euro)
```

Quiz

What does this code do?

```
> stock_prices_usd   = c(98, 99, 100)
> euro_exchange_rate = 0.95
> stock_prices_euro  = stock_prices_usd * euro_exchange_rate
> print(stock_prices_euro)
[1] 93.10 94.05 95.00
```

Convert Dollars to Euros

Vectorize the conversion

Your Turn #5

- Recreate your vector "years" that contains the year Columbus sailed the ocean blue, the year the US became an independent country and the year you joined MB
- Using a vectorized operation, create a new variable "years_since" that contains the years since 1492 that each event happened

```
years       = c(1492, 1776, 2021)
years_since = ...
```

```
> years = c(1492, 1776, 2021)
> years_since = years - 1492
> print(years_since)
[1]   0 284 529
```

My Solution

The subraction is vectorized over the years variable

Vectors

- Creating Vectors
- Selecting Vector Elements
- Vectorization

Recap key points covered.
Mention that there are lots of other things to say about vectors, but we're aiming for working knowledge or the basics here.

Agenda

- Variables & Environments
- Vectors
- Data Frames

Data Frames

- Creating Data Frames
- Working with columns

Quiz: What's wrong with this picture?

```
stock_prices _usd  = c(98, 99, 100)
stock_prices_euro = c(93.1, 94.05, 95)
stock_prices_yen   = c(12544, 12672, 12800)
...
```

I'll give my answer in the next slide

Quiz: What's wrong with this picture?

```
stock_prices _usd  = c(98, 99, 100)
stock_prices_euro = c(93.1, 94.05, 95)
stock_prices_yen   = c(12544, 12672, 12800)
...
```

These vectors are all related to each other. There are potentially hundreds of different currencies. Do I really need a different variable for each currency?

This approach clutters the namespace and makes it hard to find the right data when you need it

```
> df = data.frame(stock_prices_usd,
+                 stock_prices_euro,
+                 stock_prices_yen)
> df
  stock_prices_usd stock_prices_euro stock_prices_yen
1               98             93.10            12544
2               99             94.05            12672
3              100             95.00            12800
```

Data Frame

Collection of vectors

-A data frame is just a collection of vectors.
-You create a data frame with the function data.frame.
-Each parameter you give data.frame becomes a column of the data.frame.
-The name of the variable you give it winds up being the name of the column.
-Data frames are rectangles - each vector needs to be the same length.

```
> df = data.frame(usd  = stock_prices_usd,
+                 euro = stock_prices_euro,
+                 yen  = stock_prices_yen)
> df
  usd  euro   yen
1  98 93.10 12544
2  99 94.05 12672
3 100 95.00 12800
```

Setting Column Names

(Similar to "named vector")

By default, the name of a column is the name of the variable you give to data.frame. However, you can supply a custom name for the column with the x = y syntax. This is similar how we created a named vector. Again, the "name" can optionally appear in quotes.

Your Turn #6 - Create a df

- Create a data frame like you see here
- 1 column called "event"
- 1 column caled "year"

	event	year
1	born	1978
2	hs	1996
3	mb	2021

```
> events = c("columbus", "usa", "mb")
> years = c(1492, 1776, 2021)
> df = data.frame(event = events,
+                 year  = years)
> df
     event year
1 columbus 1492
2      usa 1776
3       mb 2021
```

My solution. Note that I liked using plural for the vector name, but chose singular for the column name. A lot of time in programming is spent on pedantic things like naming conventions on when to use singular/plural.

Data Frames

- Creating Data Frames
- Working with columns

The most important thing about working with columns of a data frame is this: remember that they are just vectors. Everything that you previously learned about vectors applies when you are working with columns of a data frame.

df$column

Whenever you want to access a column of a data frame, you'll need to use the $ operator. The syntax is what you see here: The name of the data frame, then $, then the name of the column

Data Frame Column Operations

1. Select a column
2. Create a new column
3. Update a column
4. Delete a column

	stock_prices_usd	stock_prices_euro	stock_prices_yen
1	98	93.10	12544
2	99	94.05	12672
3	100	95.00	12800

When I manipulate data frames in base R, it's often to work with a specific column. These are the 4 operations I do. I'm going to give you an example of doing each operation right now. The examples will all use this data frame. Please pay attention to the syntax. And remember: each column is just a vector.

```
> df
  stock_prices_usd stock_prices_euro stock_prices_yen
1               98              93.10             12544
2               99              94.05             12672
3              100              95.00             12800
> df$stock_prices_usd
[1]  98  99 100
```

Select a column

It is very common that you just want to access all the values in a column of a data frame

```
> df
  stock_prices_usd stock_prices_euro stock_prices_yen
1               98             93.10            12544
2               99             94.05            12672
3              100             95.00            12800
> df$stock_prices_bitcoin = c(Inf, Inf, Inf)
> df
  stock_prices_usd stock_prices_euro stock_prices_yen stock_prices_bitcoin
1               98             93.10            12544                  Inf
2               99             94.05            12672                  Inf
3              100             95.00            12800                  Inf
```

Create a new column

You create a new column just by assigning something to it. (Similar to how you create a new variable - just assign something to it)

```
> df
  stock_prices_usd stock_prices_euro stock_prices_yen stock_prices_bitcoin
1               98             93.10            12544                  Inf
2               99             94.05            12672                  Inf
3              100             95.00            12800                  Inf
> df$stock_prices_bitcoin = df$stock_prices_usd * 0.00003316
> df
  stock_prices_usd stock_prices_euro stock_prices_yen stock_prices_bitcoin
1               98             93.10            12544           0.00324968
2               99             94.05            12672           0.00328284
3              100             95.00            12800           0.00331600
```

Updating a Column

This is a very common pattern - setting a column equal to some vectorized operation on another column

```
> df
  stock_prices_usd stock_prices_euro stock_prices_yen stock_prices_bitcoin
1               98             93.10            12544           0.00324968
2               99             94.05            12672           0.00328284
3              100             95.00            12800           0.00331600
> df$stock_prices_bitcoin = NULL
> df
  stock_prices_usd stock_prices_euro stock_prices_yen
1               98             93.10            12544
2               99             94.05            12672
3              100             95.00            12800
```

Deleting a column

This is new. We haven't seen NULL before. But it just means "delete this object." In this case, we delete a column by assigning NULL to it.

Your Turn #7 - Columns

- You are given this data frame
  - Modify it to have the correct values
  - Using vectorization, create a new column called "years_since" that has the years since 1492 for each event
  - Delete the "year" column
  - Print the result

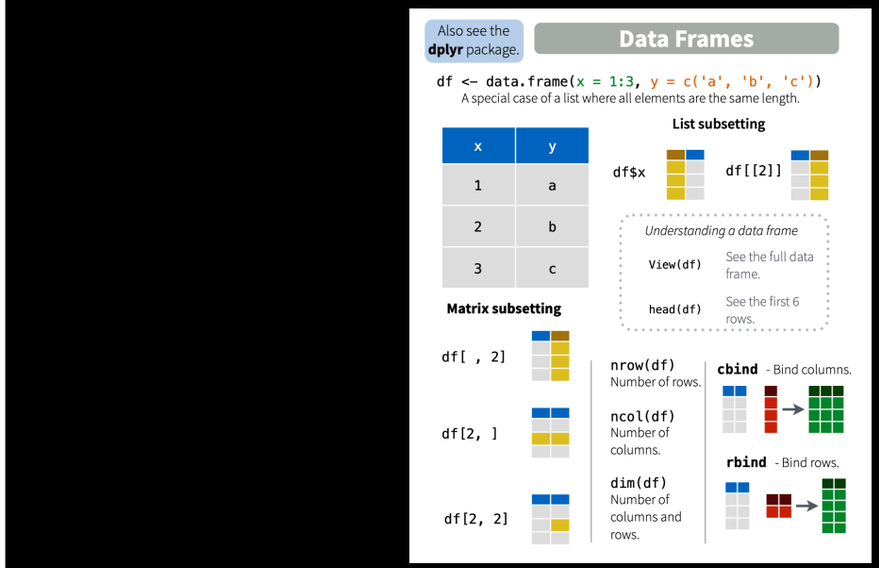
```
> df
      event year
1 columbus 1492
2       usa 1776
3        mb 2021
```

```
> events = c("columbus", "usa", "mb")
> years  = c(1492, 1776, 2021)
>
> df = data.frame(event = events,
+                 year  = years)
>
> df$years_since = df$year - 1492
> df$year = NULL
>
> print(df)
     event years_since
1 columbus           0
2      usa         284
3       mb         529
```

Data Frames

- Creating Data Frames
- Working with columns

So this concludes what I wanted to cover about working with data frames with R. Of course, there's more to cover (such as how to access specific elements of the data frame). But this goal here was just to cover a few topics, and give you hands on experience solving problems.



Cheat Sheet

Point out what we did and did not cover.

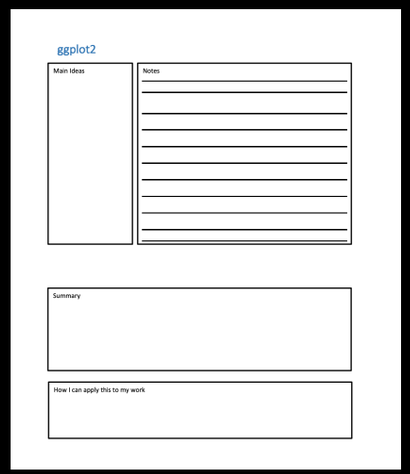
Recap

- Variables & Environments: `x = 1`
- Vectors
  - Creating Vectors: `x = c(1, 2, 3)`
  - Selecting Vector Elements: `x[1]`
  - Vectorization: `x + 1`
- Data Frames
  - Creating Data Frames: `df = data.frame(x = c('a', 'b'), y = c(1, 2))`
  - Working with columns: `df$z = df$x * 2`

And this also covers the entire workshop.

Closing Exercise

- Fill out the "Summary" section
- Fill out the "How I can apply this to my work"


ggplot2

Main Ideas

Notes

Summary

How I can apply this to my work

Any Questions?