

Update to newest tidyverse

- You will need to use the conda-forge version of the R implementation for Jupyter Notebook
 - More up to date (has tidyverse 1.3)
 - Not official implementation from R

• You will have to use the command line to perform this update

1. Open command line interface (Terminal for Mac, Command Prompt for Windows)
2. Activate your R environment, for example"

conda activate "base_R"

1. Run the following two lines to enable conda to download packages from conda-forge.

```
conda config --add channels conda-forge

conda config --set channel_priority strict
```

1. Now run the following line to update tidyverse. Answer y when it asks yes or no to proceed.

```
conda update r-tidyverse

1. Launch Jupyter notebook
```

```
In [1]: # colors!
library(tidyverse)

# Attaching packages
# tidyverse 1.3.0
# ggplot2 3.3.3      # purrr  0.3.4
# tibble 3.0.6       # dplyr  1.0.4
# tidyr  1.1.2       # stringr 1.4.0
# readr  1.4.0       # forcats 0.5.1

# Conflicts:
# dplyr::filter() masks state::filter()
# dplyr::lag()    masks state::lag()
```

```
In [2]: # More informative tibble display
diamonds

# A tibble: 53940 x 10
  carat    cut    color clarity depth  price     x     y     z
<dbl> <ord> <ord> <ord> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 0.23 Ideal E SI2 I15 59.8 61 326 3.89 3.84 2.31
2 0.21 Premium E SI1 59.8 61 326 3.89 3.84 2.31
3 0.23 Good E VS1 56.9 65 327 4.05 4.07 2.33
4 0.29 Premium I VS2 62.4 58 334 4.20 4.23 2.63
5 0.31 Good J SI2 63.3 58 336 4.34 4.36 2.75
6 0.24 Very Good J VVS2 62.8 57 336 3.94 3.96 2.48
7 0.24 Very Good I VVS1 62.3 57 336 3.95 3.98 2.47
8 0.26 Very Good H SI1 61.9 55 337 3.87 4.07 2.53
9 0.22 Fair E VS2 65.1 61 337 3.87 3.78 2.49
10 0.23 Very Good H VS1 59.4 61 338 4.00 4.05 2.39
11 0.30 Good J SI1 64.0 55 339 4.25 4.28 2.73
12 0.23 Ideal J VS1 62.8 56 340 3.93 3.90 2.46
13 0.22 Premium F SI1 60.4 61 342 3.88 3.84 2.33
14 0.31 Ideal J SI2 62.2 54 344 4.35 4.37 2.71
15 0.20 Premium E SI2 60.2 62 345 3.79 3.75 2.27
16 0.32 Premium E I1 60.9 58 345 4.38 4.42 2.68
17 0.30 Ideal I SI2 62.0 54 348 4.31 4.34 2.68
18 0.30 Good J SI1 63.4 54 351 4.23 4.29 2.70
19 0.30 Good J SI1 63.8 56 351 4.23 4.26 2.71
20 0.30 Very Good J SI1 62.7 59 351 4.21 4.27 2.66
21 0.30 Good I SI2 63.3 56 351 4.26 4.30 2.71
22 0.23 Very Good E VS2 63.8 55 352 3.85 3.92 2.48
23 0.23 Very Good H VS1 61.0 57 353 3.94 3.96 2.41
24 0.31 Very Good J SI1 59.4 62 353 4.39 4.43 2.62
25 0.31 Very Good J SI1 58.1 62 353 4.44 4.47 2.59
26 0.23 Very Good G VVS2 60.4 58 354 3.97 4.01 3.41
27 0.24 Premium I VS1 62.5 57 355 3.97 3.94 2.47
28 0.30 Very Good J VS2 62.2 57 357 4.28 4.30 2.67
29 0.23 Very Good D VS2 60.5 61 357 3.96 3.97 2.40
30 0.23 Very Good F VS1 60.9 57 357 3.96 3.99 2.42
31 0.70 Premium E SI1 60.5 58 2753 5.74 5.77 3.48
32 0.57 Premium E IF 59.8 60 2753 5.43 5.38 3.23
33 0.61 Premium F VVS1 61.8 59 2753 5.48 5.40 3.61
34 0.80 Good G VS2 64.2 58 2753 5.84 5.81 3.74
35 0.84 Good I VS1 63.7 59 2753 5.94 5.90 3.77
36 0.77 Ideal E SI2 62.1 56 2753 5.84 5.86 3.63
37 0.74 Good D SI1 63.1 59 2753 5.71 5.74 3.61
38 0.90 Very Good J SI1 63.2 60 2753 6.12 6.09 3.86
39 0.76 Premium I VS1 59.3 62 2753 5.93 5.85 3.49
40 0.76 Ideal I VVS1 62.2 55 2753 5.89 5.87 3.66
41 0.70 Very Good E VS2 62.4 60 2755 5.57 5.61 3.49
42 0.70 Very Good E VS2 62.8 60 2755 5.59 5.65 3.53
43 0.70 Very Good D VS1 63.1 59 2755 5.67 5.58 3.55
44 0.73 Ideal I VS2 61.3 56 2756 5.80 5.84 3.57
45 0.73 Ideal I VS2 61.6 55 2756 5.82 5.84 3.59
46 0.79 Ideal I SI1 61.6 56 2756 5.95 5.97 3.67
47 0.71 Ideal E SI1 61.9 56 2756 5.71 5.73 3.53
48 0.79 Good F SI1 58.1 59 2756 6.06 6.13 3.54
49 0.79 Premium E SI2 61.4 58 2756 6.03 5.96 3.68
50 0.71 Ideal G VS1 61.4 56 2756 5.76 5.73 3.53
51 0.71 Premium E SI1 60.5 55 2756 5.79 5.74 3.49
52 0.71 Premium F SI1 59.8 62 2756 5.74 5.73 3.43
53 0.70 Very Good E VS2 60.5 59 2757 5.71 5.76 3.47
54 0.70 Very Good E VS2 61.2 59 2757 5.69 5.72 3.49
55 0.72 Premium D SI1 62.7 59 2757 5.69 5.73 3.58
56 0.72 Ideal D SI1 60.8 57 2757 5.75 5.76 3.50
57 0.72 Good D SI1 63.1 55 2757 5.69 5.75 3.61
58 0.70 Very Good D SI1 62.8 60 2757 5.66 5.68 3.56
59 0.86 Premium H SI2 61.0 58 2757 6.15 6.12 3.74
60 0.75 Ideal D SI2 62.2 55 2757 5.83 5.87 3.64
```

```
In [ ]: # gather() function replaced by pivot_longer() for data tidying (we will discuss this soon)
#pivot_longer
```

Jupyter Notebook Widgets

(optional)

- You can add fancy add-ons to Jupyter Notebook using nbextensions.

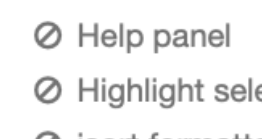
1. Install ipynbwidgets. Run and hit y when prompted:

```
conda install ipynbwidgets
```

1. Install nbextensions. Run and hit y when prompted:

```
conda install jupyter_contrib_nbextensions
```

1. Launch Jupyter notebook.

 Quit Logout

Files Running Clusters Nbextensions

Configurable nbextensions

disable configuration for nbextensions without explicit compatibility (they may break your notebook environment, but can be useful to show for nbextension development)

filter: by description, section, or tags

<input type="checkbox"/> (some) LaTeX environments for Jupyter	<input type="checkbox"/> 2to3 Converter	<input type="checkbox"/> AddBefore	<input type="checkbox"/> Autopap8
<input type="checkbox"/> AutoSaveTime	<input type="checkbox"/> Autoscroll	<input type="checkbox"/> Cell Filter	<input type="checkbox"/> Code Font Size
<input type="checkbox"/> Code pretty	<input type="checkbox"/> Codefolding	<input type="checkbox"/> Codefolding in Editor	<input type="checkbox"/> CodeMirror mode extensions
<input type="checkbox"/> Collapsible Headings	<input type="checkbox"/> Comment/Uncomment Hotkey	<input type="checkbox"/> contrib_nbextensions_help_item	<input type="checkbox"/> datasteampipe
<input type="checkbox"/> Equation Auto Numbering	<input type="checkbox"/> ExecuteTime	<input type="checkbox"/> Execution Dependencies	<input type="checkbox"/> Exercise
<input type="checkbox"/> Exercise2	<input type="checkbox"/> Export Embedded HTML	<input type="checkbox"/> Freeze	<input type="checkbox"/> Gist-it
<input type="checkbox"/> Help panel	<input type="checkbox"/> Hide Header	<input type="checkbox"/> Hide Input	<input type="checkbox"/> Hide input all
<input type="checkbox"/> Highlight selected word	<input type="checkbox"/> highlighter	<input type="checkbox"/> Hinterland	<input type="checkbox"/> Initialization cells
<input type="checkbox"/> Isort formatter	<input checked="" type="checkbox"/> jupyter-js-widgets/extension	<input type="checkbox"/> Keyboard shortcut editor	<input type="checkbox"/> Launch QTC Console
<input type="checkbox"/> Limit Output	<input type="checkbox"/> Live Markdown Preview	<input type="checkbox"/> Load TeX macros	<input type="checkbox"/> Move selected cells
<input type="checkbox"/> Navigation-Hotkeys	<input type="checkbox"/> Live Nbextensions dashboard tab	<input type="checkbox"/> Nbextensions edit menu item	<input type="checkbox"/> nbTranslate
<input type="checkbox"/> Notify	<input type="checkbox"/> Preview	<input type="checkbox"/> Python MarkDown	<input type="checkbox"/> Rubeusband
<input type="checkbox"/> Ruler	<input type="checkbox"/> Ruler in Editor	<input type="checkbox"/> Runtools	<input type="checkbox"/> Scratchpad
<input type="checkbox"/> ScrollDown	<input type="checkbox"/> Select CodeMirror Keymap	<input type="checkbox"/> SKILL Syntax	<input type="checkbox"/> Skip-Traceback
<input type="checkbox"/> Snippets	<input type="checkbox"/> Snippets Menu	<input type="checkbox"/> spellchecker	<input type="checkbox"/> Split Cells Notebook
<input type="checkbox"/> Table of Contents (2)	<input type="checkbox"/> table_beautifier	<input type="checkbox"/> Toggle all line numbers	<input type="checkbox"/> Tree Filter
<input type="checkbox"/> Variable Inspector	<input type="checkbox"/> zenmode		

2.
 - Be careful if you enable certain widgets and then try to share your .ipynb file. If the other party doesn't have the same Nbextensions enabled, it can cause issues.
 - For example, frozen cells may remain frozen, but the recipient of your .ipynb file may not be able to unfreeze.

Reminders

- There is an online textbook
- There is a GitHub page with suggested reading in this textbook (I'll try to also include in this notebook file)

Data Structures in R

(Chapters 20 and 10 in the Textbook)

	Homogeneous	Heterogeneous
1D	(Atomic) Vector	List
2D	Matrix	Dataframe/Tibble
3D	Arrays	

- Homogeneous: All elements are of the same type
- Heterogeneous: Elements can be different types

Four Common Types

- Integer: integer values
- Double: real numbers
- Logical: Boolean values (TRUE or FALSE)
- Character: text-strings ("price", "hello", "12")

```
In [3]: print(mpg)

# A tibble: 234 x 11
  manufacturer model      displ year   cyl trans  drv      cty   hwy fl      class
<chr>         <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr>
1 audi        a4          1.8 1999    4 auto(l f 18 29 p comp-
2 audi        a4          1.8 1999    4 manual f  21 29 p comp-
3 audi        a4          2 2008    4 manual f  20 31 p comp-
4 audi        a4          2 2008    4 auto(a f  21 30 p comp-
5 audi        a4          2.8 1999    6 auto(l f  16 26 p comp-
6 audi        a4          2.8 1999    6 manual f  18 26 p comp-
7 audi        a4          3.1 2008    6 auto(a f  18 27 p comp-
8 audi        a4 quat... 1.8 1999    4 manual f  18 26 p comp-
9 audi        a4 quat... 1.8 1999    4 auto(l f  16 25 p comp-
10 audi       a4 quat... 2 2008    4 manual f  20 28 p comp-
# ... with 224 more rows
```

```
In [4]: mpg %>%
mutate(old = (year < 2008)) %>%
print(width = Inf)

# A tibble: 234 x 12
  manufacturer model      displ year   cyl trans  drv      cty   hwy fl      class
<chr>         <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr>
1 audi        a4          1.8 1999    4 auto(l f  18 29 p comp-
2 audi        a4          1.8 1999    4 manual(m5) f  21 29 p comp-
3 audi        a4          2 2008    4 manual(m5) f  20 31 p comp-
4 audi        a4          2 2008    4 auto(av) f  21 30 p comp-
5 audi        a4          2.8 1999    6 auto(l f  16 26 p comp-
6 audi        a4          2.8 1999    6 manual(m5) f  18 26 p comp-
7 audi        a4          3.1 2008    6 auto(av) f  18 27 p comp-
8 audi        a4 quattro 1.8 1999    4 manual(m5) f  18 26 p comp-
9 audi        a4 quattro 1.8 1999    4 auto(l f  16 25 p comp-
10 audi       a4 quattro 2 2008    4 manual(m5) f  20 28 p comp-
# ... with 224 more rows

class old
<chr> <lg>
1 compact TRUE
2 compact TRUE
3 compact FALSE
4 compact FALSE
5 compact TRUE
6 compact TRUE
7 compact FALSE
8 compact TRUE
9 compact TRUE
10 compact FALSE
# ... with 224 more rows
```

```
In [6]: # Create a real-valued vector
(x <- c(1.1, 2, 3.3))

1.1 2 3.3
```

```
In [7]: # Check the type
typeof(x)

'double'
```

```
In [8]: # Check if it is a vector
is_atomic(x)

# Check if it is a real-valued atomic vector
is_double(x)

TRUE
TRUE
```

```
In [9]: # How many elements are in the vector?
length(x)

3
```

```
In [10]: # Create an integer vector
(y <- c(1L, 2L, 3L, 10L))
typeof(y)
length(y)

1 2 3 10
'integer'
4
```

```
In [13]: # Create a logical-valued vector
(z <- c(TRUE, T, F, FALSE))
typeof(z)

# This will return an error
#(z <- c(TRUE, T, F, False))

TRUE · TRUE · FALSE · FALSE
'logical'
```

```
In [15]: # Create a character vector
(w <- c("hello", "world"))
typeof(w)
length(w)

'hello' · 'world'
'character'
2
```

```
In [16]: # fancier creation methods
c(1:10) # vector of integers 1 to 10, equivalent to seq(1, 10)
seq(2, 10, 2) # vector of even integers 2 to 10

1 2 3 4 5 6 7 8 9 10
2 4 6 8 10
```

```
In [17]: ?seq
```

```
In [19]: (x <- c(1:4))
(y <- c(1, 2, 3, 4))

typeof(x)
typeof(y)

1 2 3 4
1 2 3 4
'integer'
'double'
```

Note: Differences between integer and real values are important when comparing values

```
In [20]: # Floating point operations are never exact
(x <- sqrt(2)^2)
x == 2

# use dplyr::near() to compare
near(x, 2)

2
4.44089209850063e-16
FALSE
TRUE
```

```
In [22]: x <- c(1)
is_atomic(x)

TRUE
```

```
In [23]: # there are no scalars in R, just one-element vectors
is_atomic(1)
is_double(2.2)
is_character("hello world")

TRUE
TRUE
TRUE
```

```
In [24]: length(200)
length("hello world")

1
1
```

Vector Coercion

- Vectors can be coerced upstream
 - If you create a vector with mixed types, everything will be coerced to the most flexible type
 - TRUE and FALSE values are coerced to 1 and 0 respectively
- Most flexible to least flexible:
1. Character
 2. Double
 3. Integer
 4. Logical

```
In [25]: # everything is coerced to character values
(x <- c(1L, 2.2, TRUE, "hello"))
typeof(x)

# everything is coerced to real values
(y <- c(1L, FALSE, 1.1))
typeof(y)

'1' · '2.2' · 'TRUE' · 'hello'
'character'
1 0 1 1
'double'
```

```
In [26]: # We can explicitly coerce vectors using ""as""
(x <- as.integer(c(TRUE, FALSE)))
(y <- as.character(c(1L, 2L, 10L)))
typeof(x)
typeof(y)

1 0
'1' · '2' · '10'
'integer'
'character'
```

```
In [28]: x <- c(1, 2)
typeof(x)
typeof(as.integer(x))

'double'
'integer'
```

```
In [29]: # This helps with tibbles as well

print(mpg)
mpg %>%
mutate(cty = as.double(cty)) %>%
print()

# A tibble: 234 x 11
  manufacturer model      displ year   cyl trans  drv      cty   hwy fl      class
<chr>         <chr>      <dbl> <int> <int> <chr> <chr> <dbl> <int> <chr>
1 audi        a4          1.8 1999    4 auto(l f 18 29 p comp-
2 audi        a4          1.8 1999    4 manual f  21 29 p comp-
3 audi        a4          2 2008    4 manual f  20 31 p comp-
4 audi        a4          2 2008    4 auto(a f  21 30 p comp-
5 audi        a4          2.8 1999    6 auto(l f  16 26 p comp-
6 audi        a4          2.8 1999    6 manual f  18 26 p comp-
7 audi        a4          3.1 2008    6 auto(a f  18 27 p comp-
8 audi        a4 quat... 1.8 1999    4 manual f  18 26 p comp-
9 audi        a4 quat... 1.8 1999    4 auto(l f  16 25 p comp-
10 audi       a4 quat... 2 2008    4 manual f  20 28 p comp-
# ... with 224 more rows
```

1. Create the following vectors:

- A vector of your name and age
- A vector of your seven favorite numbers
- A vector of logical values

Try to guess the type and length of each, before checking it explicitly using typeof() and length().

1. Run both ?seq and ?rep to see some more fancy ways of creating vectors.

```
In [33]: x <- c("Mike", 29)
typeof(x)
length(x)

'character'
2
```

```
In [35]: y <- c(7, 2.71828, 64, 1, 2, 3, 4)
typeof(y)
length(y)

'double'
7
```

```
In [36]: z <- c(TRUE, T, F, FALSE)
typeof(z)
length(z)

'logical'
4
```

Subsetting Vectors

- You can use [...] to extract elements of vectors
- Indexing in R starts at 1

```
In [43]: x <- c('one', 'two', 'three', 'four', 'five')
x[1] # same as x[0] in Python
x[4]

'one'
'four'
```

```
In [44]: # This returns a one-element vector (not a scalar!)
is_atomic(x[1])

TRUE
```

```
In [45]: # extract elements 2 through 4 (including 2 and 4!)
x[2:4]

'two' · 'three' · 'four'
```

```
In [48]: # You can subset using other vectors
x[c(1, 3)]
x[c(1, 1, 1, 1, 2, 1)]

'one' · 'three'
'one' · 'one' · 'one' · 'one' · 'two' · 'one'
```

```
In [49]: # negative values drop elements
x[c(-1, -2)]

# order is not important
x[c(-2, -1)]

'three' · 'four' · 'five'
'three' · 'four' · 'five'
```

```
In [50]: # comparison statements involving vectors return logical vectors
y <- c(1.1, 1.2, 2, 5.5)
y > 1.5

FALSE · FALSE · TRUE · TRUE
```

```
In [54]: # Logical vectors can be used to subset vectors
x[c(T, T, F, F)]

'two' · 'three'
```

```
In [55]: # We can combine this to get statements like this:
# all elements of y, greater than 1.5
y[y > 1.5]

# this is equivalent to
y[c(F, F, T, T)]

2 5 5
```

```
In [56]: c(1, 2, 3)

1 2 3
```

```
In [58]: # you can add names to the elements of a vector
(x <- c(a = 1, b = 2, c = 3))

# this can be used to subset the vector
x[c('c', 'a')]

# get a vector of the names
names(x)

a b c
1 2 3
```

c: 3 a: 1

a' b' c'