

# Welcome!

1. Log in using Windows
2. Go to [github.com/sjfox/R\\_intro](https://github.com/sjfox/R_intro), and download the materials
3. Open up RStudio

# Introduction to R

Spencer Fox  
31 March 2015

Email: spncrfx@gmail.com  
Twitter: @foxandtheflu

# Workshop goals

# Workshop goals

- Learn to run code in R

# Workshop goals

- Learn to run code in R
- Play with R without getting bogged down with programming

# Workshop goals

- Learn to run code in R
- Play with R without getting bogged down with programming
- Become familiar enough to google

# Workshop goals

- Learn to run code in R
- Play with R without getting bogged down with programming
- Become familiar enough to google
- Make a few nice looking graphics

# Workshop outline

# Workshop outline

- Introductory information (Slides)

# Workshop outline

- Introductory information (Slides)
- First coding session
  - Using R

# Workshop outline

- Introductory information (Slides)
- First coding session
  - Using R
- Basics of R programming (Slides)

# Workshop outline

- Introductory information (Slides)
- First coding session
  - Using R
- Basics of R programming (Slides)
- Second coding session
  - R data structures and coding

# Workshop outline

- Introductory information (Slides)
- First coding session
  - Using R
- Basics of R programming (Slides)
- Second coding session
  - R data structures and coding
- Basics of plotting (Slides)

# Workshop outline

- Introductory information (Slides)
- First coding session
  - Using R
- Basics of R programming (Slides)
- Second coding session
  - R data structures and coding
- Basics of plotting (Slides)
- Final coding session
  - ggplot

# Why program?

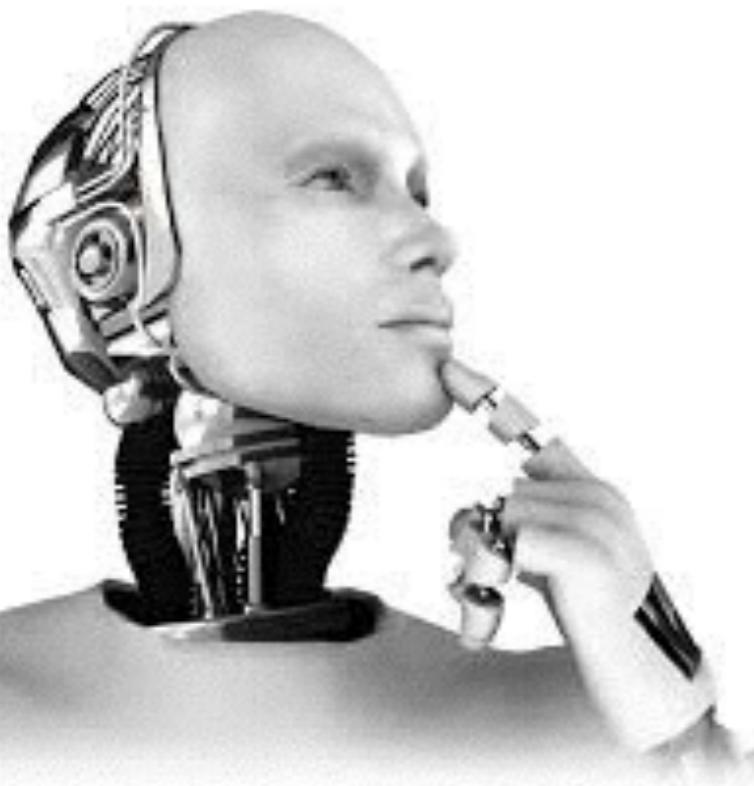
# Why program?

- Reproducibility



# Why program?

- Reproducibility
- Automation



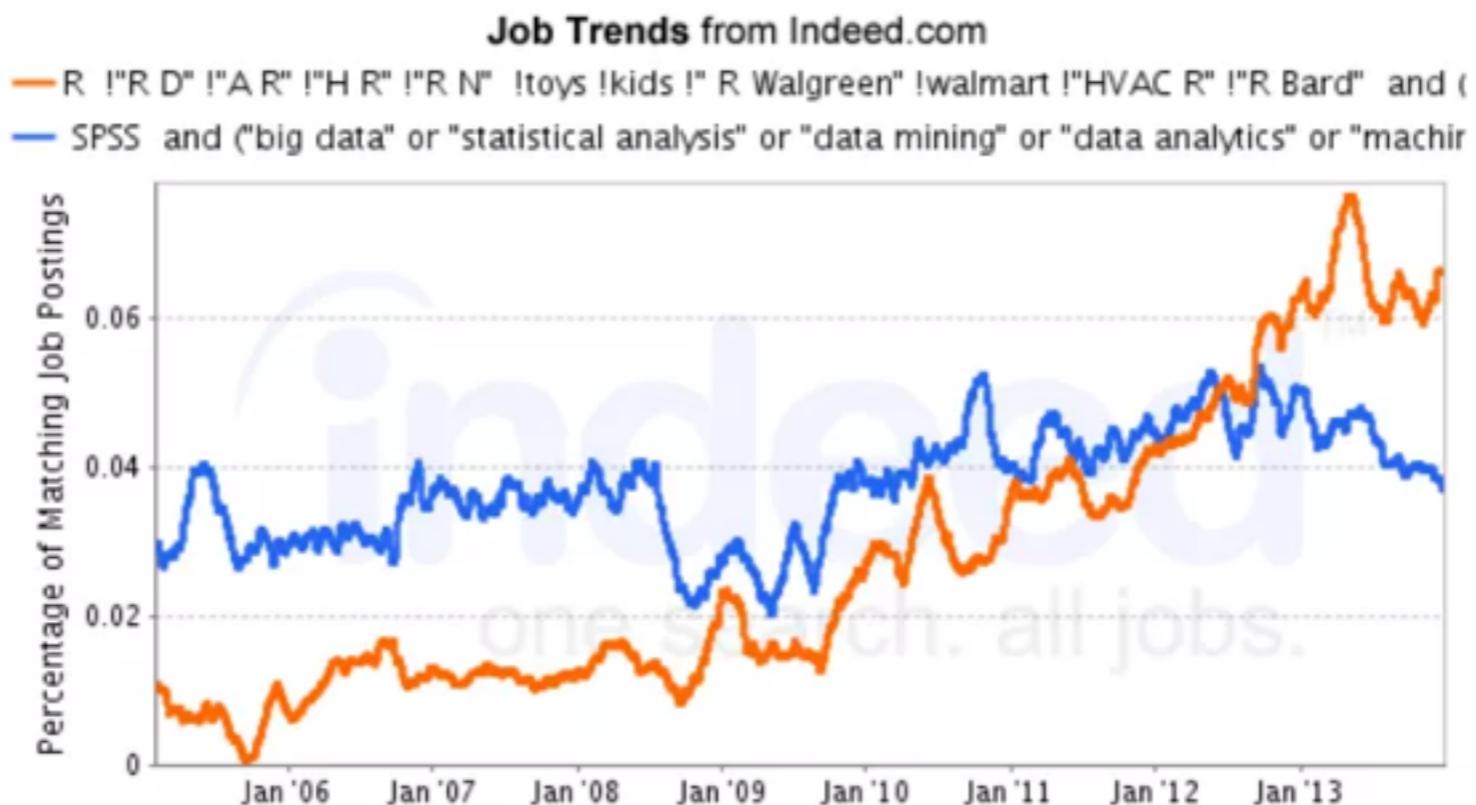
# What is R?

# What is R?

- Flexible programming language made by statisticians

# What is R?

- Flexible programming language made by statisticians
- Open source and in demand



# Why use R?

# Why use R?

- Relatively simple

# Why use R?

- Relatively simple
- Free

# Why use R?

- Relatively simple
- Free
- Powerful

# Why use R?

- Relatively simple
- Free
- Powerful
- Packages, packages, and more packages!

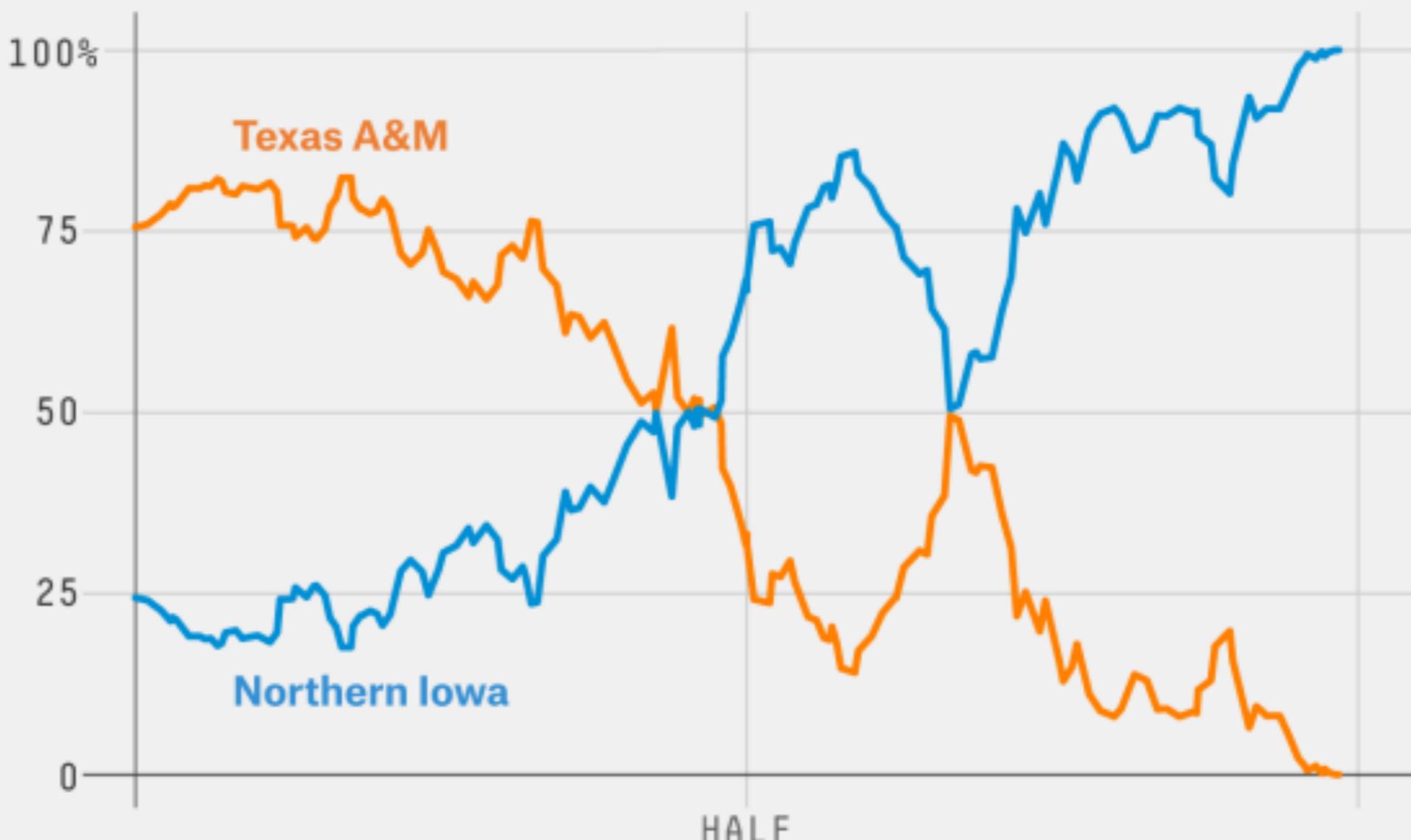
# Why use R?

- Relatively simple
- Free
- Powerful
- Packages, packages, and more packages!
- Great visualization tools

# Visualizing data

## Northern Iowa almost had it in the bag

Live win probability of the Texas A&M-Northern Iowa NCAA Tournament game before A&M's comeback, March 20, 2016



# Example R “Pipeline”

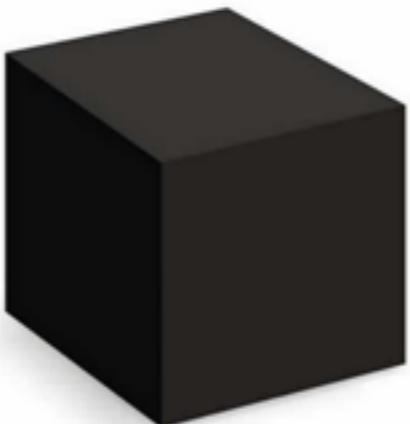
# Example R “Pipeline”

1. Generate data



# Example R “Pipeline”

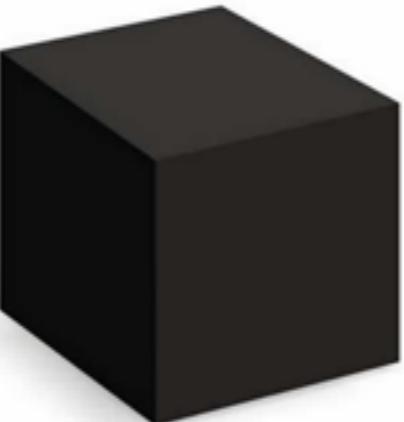
1. Generate data



2. Analyze data

# Example R “Pipeline”

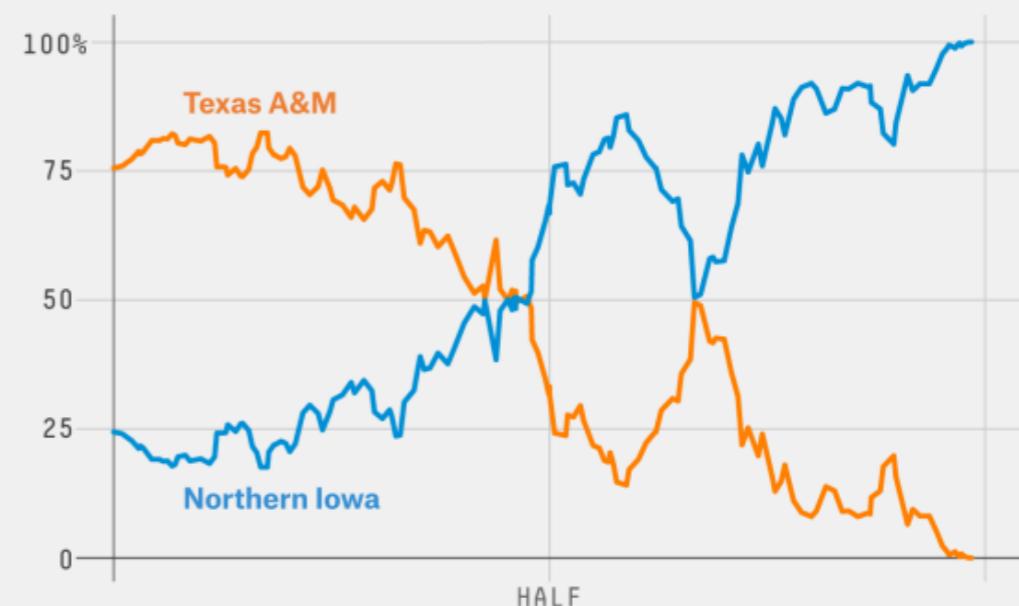
1. Generate data



2. Analyze data

3. Show analysis

**Northern Iowa almost had it in the bag**  
Live win probability of the Texas A&M-Northern Iowa NCAA Tournament game before A&M's comeback, March 20, 2016

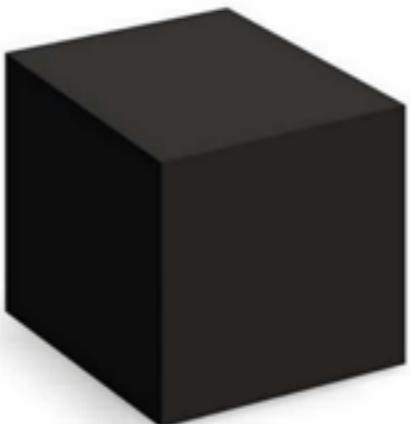


# Example R “Pipeline”

1. Generate data

in R

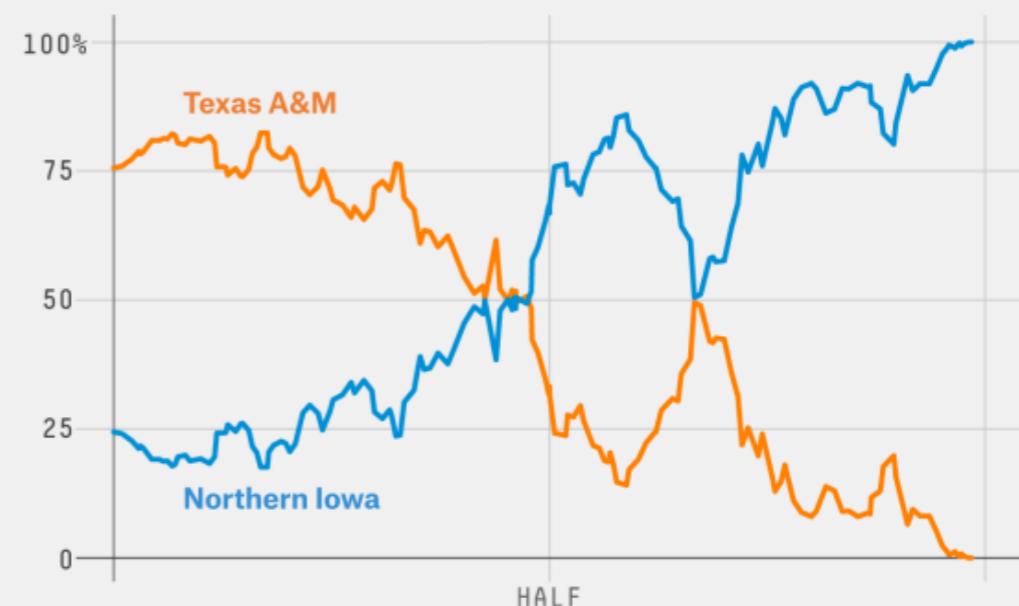
2. Analyze data



3. Show analysis



**Northern Iowa almost had it in the bag**  
Live win probability of the Texas A&M-Northern Iowa NCAA Tournament game before A&M's comeback, March 20, 2016



# Cooking a meal with R

# Cooking a meal with R

What is the desired end product?



# Cooking a meal with R

What is the desired end product?

What do we need to get it?

- Turkey
- Stuffing
- Green Beans
- Mashed Potatoes
- Gravy



# Every dish has a function



# Every dish has a function

Convert data to usable format



# Every dish has a function

Convert data to usable format



Run a statistical analysis

# Every dish has a function

Convert data to usable format



Run a statistical analysis

Run a 2nd stat analysis

# Every dish has a function

Convert data to usable format

Make a summary plot



Run a statistical analysis

Run a 2nd stat analysis

# Every function has a recipe



# Every function has a recipe

1. Ingredients
  1. Green beans
  2. Creamy mushroom soup
  3. French fried onions



# Every function has a recipe

1. Ingredients
  1. Green beans
  2. Creamy mushroom soup
  3. French fried onions
2. Instructions
  1. Heat oven to 350
  2. etc.



# Every function has a recipe

1. Ingredients
  1. Green beans
  2. Creamy mushroom soup
  3. French fried onions
2. Instructions
  1. Heat oven to 350
  2. etc.
3. Output
  1. Green bean casserole!



# I thought I was learning R!!!



[www.shutterstock.com](http://www.shutterstock.com) • 116613292

Every function has a “recipe”:  
Calculating a mean

# Every function has a “recipe”: Calculating a mean

## 1. Ingredients

# Every function has a “recipe”: Calculating a mean

1. Ingredients
  1. List of numbers

# Every function has a “recipe”: Calculating a mean

1. Ingredients
  1. List of numbers
2. Instructions

# Every function has a “recipe”: Calculating a mean

1. Ingredients
  1. List of numbers
2. Instructions
  1. Count numbers

# Every function has a “recipe”: Calculating a mean

1. Ingredients
  1. List of numbers
2. Instructions
  1. Count numbers
  2. Sum all of the numbers

# Every function has a “recipe”: Calculating a mean

1. Ingredients
  1. List of numbers
2. Instructions
  1. Count numbers
  2. Sum all of the numbers
3. Output

# Every function has a “recipe”: Calculating a mean

1. Ingredients
  1. List of numbers
2. Instructions
  1. Count numbers
  2. Sum all of the numbers
3. Output
  1. Return Sum/Count

# Every function has a “recipe”: Calculating a mean

## 1. Ingredients

1. List of numbers

## 2. Instructions

1. Count numbers

2. Sum all of the numbers

## 3. Output

1. Return Sum/Count

```
get_mean = function(numbers){...}
```

# Every function has a “recipe”: Calculating a mean

1. Ingredients
  1. List of numbers
2. Instructions
  1. Count numbers
  2. Sum all of the numbers
3. Output
  1. Return Sum/Count

```
get_mean = function(numbers){...}
```

```
> get_mean(numbers=c(2,4,6,8,10))  
[1] 6
```

# Why are packages so great?



# Why are packages so great?



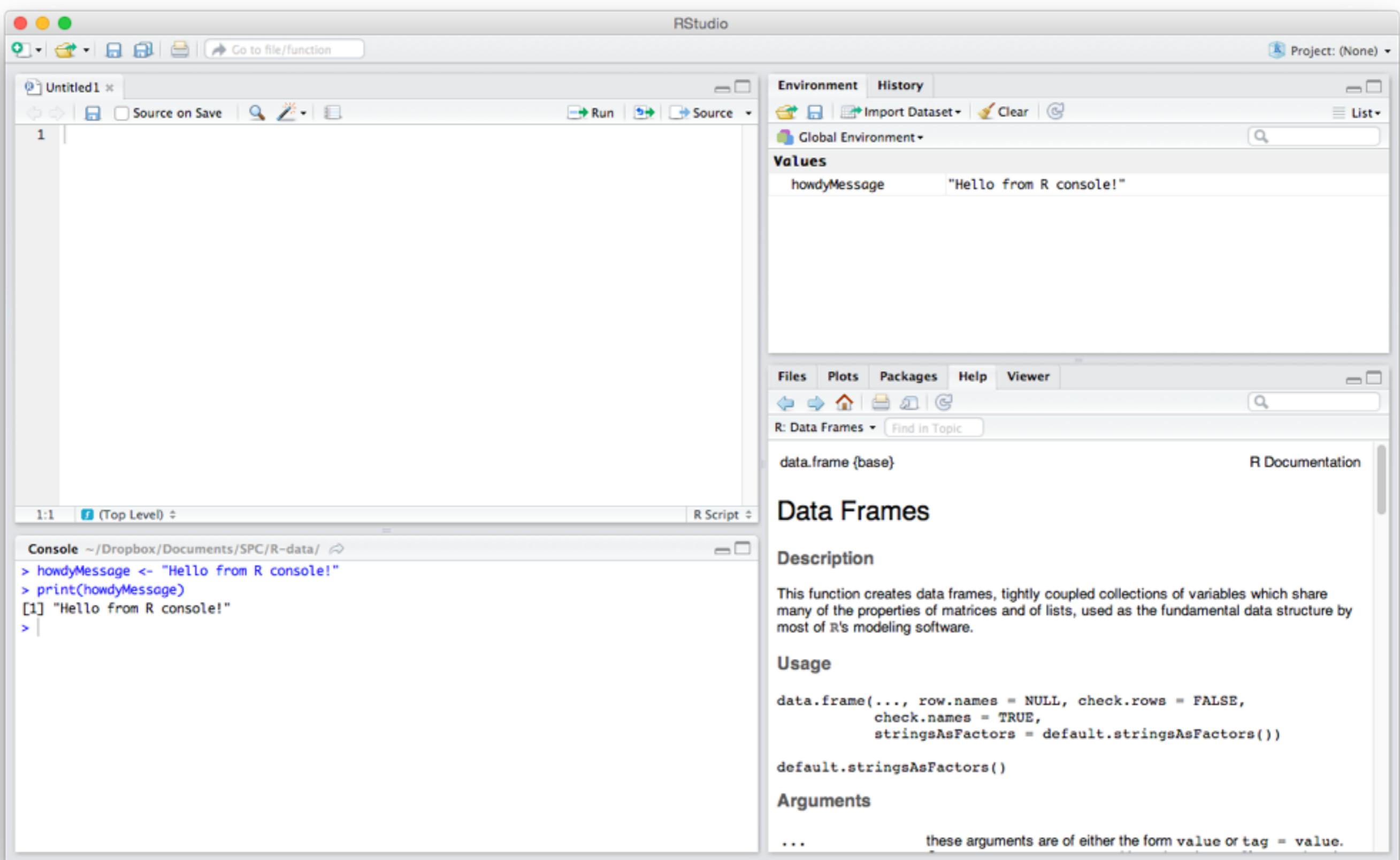
# Why are packages so great?



Just need to understand inputs and outputs, and the rest is done for you.



# Using R (RStudio)



# Using R (RStudio)

The screenshot shows the RStudio interface with the following components:

- Environment pane:** Shows the variable `howdyMessage` with the value `"Hello from R console!"`.
- Global Environment pane:** Shows the same variable `howdyMessage` with the same value.
- Data Frames documentation pane:** Displays the `data.frame` function documentation, including the description, usage, and arguments.
- Console pane:** Shows the R session history with the following commands:

```
Console ~/Dropbox/Documents/SPC/R-data/
> howdyMessage <- "Hello from R console!"
> print(howdyMessage)
[1] "Hello from R console!"
```
- Code editor pane:** Shows an untitled R script with the line `1:1` and `R Script` selected.

A large rectangular box highlights the word **Console** in the bottom-left corner of the interface.

**Console**

**Environment**   **History**

Global Environment

Values

howdyMessage   "Hello from R console!"

Files Plots Packages Help Viewer

R: Data Frames Find in Topic

data.frame {base}   R Documentation

## Data Frames

### Description

This function creates data frames, tightly coupled collections of variables which share many of the properties of matrices and of lists, used as the fundamental data structure by most of R's modeling software.

### Usage

```
data.frame(..., row.names = NULL, check.rows = FALSE,
           check.names = TRUE,
           stringsAsFactors = default.stringsAsFactors())

default.stringsAsFactors()
```

### Arguments

... these arguments are of either the form `value` or `tag = value`.

# Using R (RStudio)

The screenshot displays the RStudio IDE interface. On the left, a large rectangular box highlights the **Editor** panel, which contains a blank workspace. Below it, another large rectangular box highlights the **Console** panel, which shows the following R session:

```
1:1  (Top Level)  R Script
Console ~/Dropbox/Documents/SPC/R-data/
> howdyMessage <- "Hello from R console!"
> print(howdyMessage)
[1] "Hello from R console!"
```

On the right, a large rectangular box highlights the **Data Frames** documentation panel. It includes sections for **Description**, **Usage**, and **Arguments**. The **Description** section states: "This function creates data frames, tightly coupled collections of variables which share many of the properties of matrices and of lists, used as the fundamental data structure by most of R's modeling software." The **Usage** section shows the function signature: `data.frame(..., row.names = NULL, check.rows = FALSE, check.names = TRUE, stringsAsFactors = default.stringsAsFactors())`. The **Arguments** section notes: "... these arguments are of either the form value or tag = value."

# Using R (RStudio)

The screenshot displays the RStudio interface with three main components highlighted by large black boxes:

- Editor**: The top-left pane, which is a code editor for writing R scripts. It shows a single file named "Untitled1" with the following content:

```
1:1  (Top Level)  R Script
Console ~/Dropbox/Documents/SPC/R-data/
> howdyMessage <- "Hello from R console!"
> print(howdyMessage)
[1] "Hello from R console!"
```
- Console**: The bottom-left pane, which is the R terminal window. It shows the command-line history and the output of the "print" command.
- Environment**: The right-hand pane, which contains the R environment viewer. It shows the global environment with a variable "howdyMessage" set to the value "Hello from R console!". Below this, there is a detailed documentation view for the "data.frame" function.

**Environment** Documentation View:

**Data Frames**

**Description**

This function creates data frames, tightly coupled collections of variables which share many of the properties of matrices and of lists, used as the fundamental data structure by most of R's modeling software.

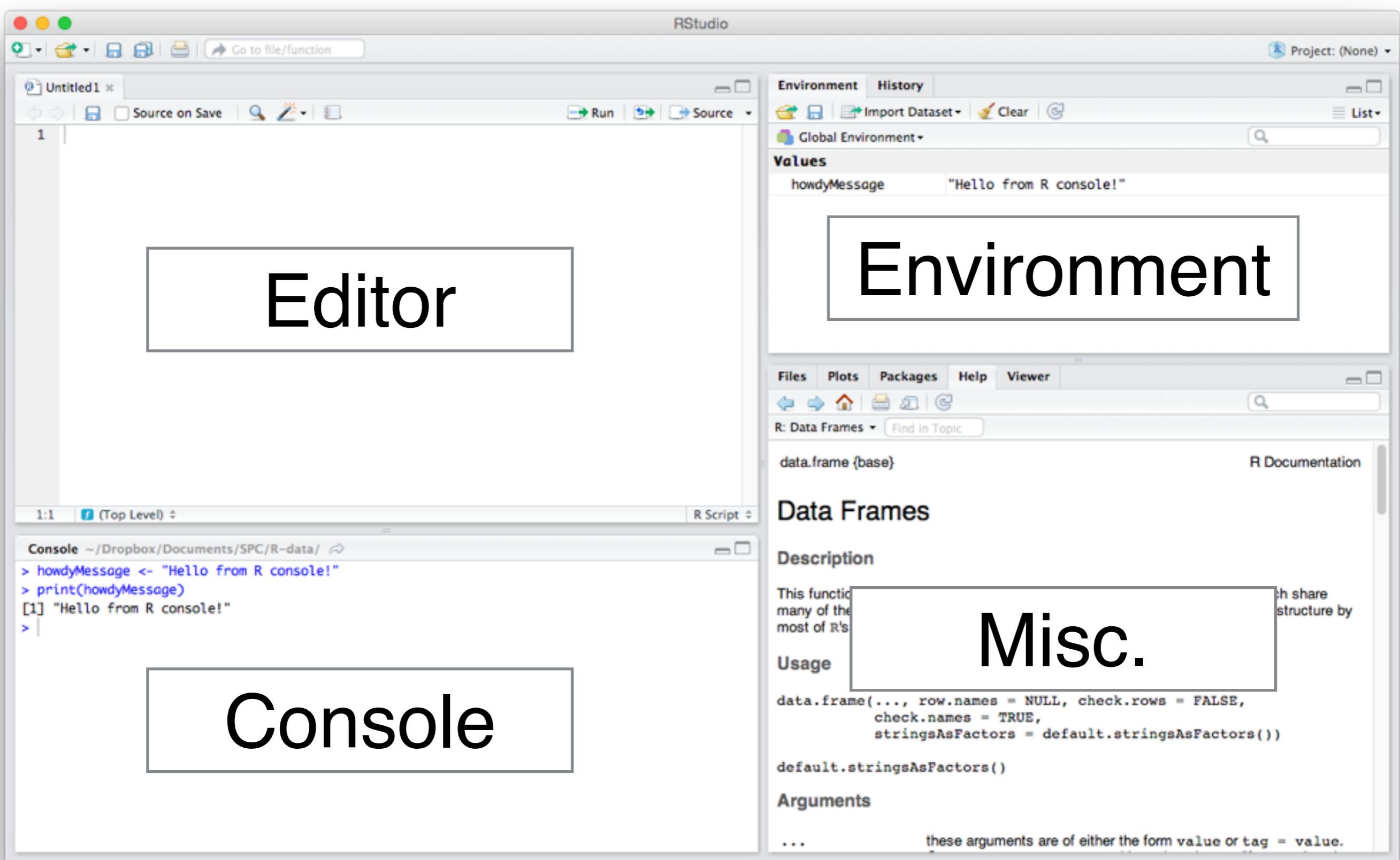
**Usage**

```
data.frame(..., row.names = NULL, check.rows = FALSE,
           check.names = TRUE,
           stringsAsFactors = default.stringsAsFactors())
default.stringsAsFactors()
```

**Arguments**

... these arguments are of either the form `value` or `tag = value`.

# Using R (RStudio)



# 1st Programming Exercise

1. open up 1\_package\_intro.R in Rstudio
2. Start playing with code
  1. Do: run code, change things and see what happens
  2. Don't: change code until it works and move on without understanding why

How to run code:

1. Highlight line(s) of code in editor
2. Type, ctrl+enter (windows) or cmd+enter (mac)
3. See code running in console
4. View output/figures

# R building blocks

At a basic level, R is a calculator

```
> 5+5  
[1] 10  
> 7*3  
[1] 21
```

# R building blocks

At a basic level, R is a calculator

```
> 5+5  
[1] 10  
> 7*3  
[1] 21
```

But it's a smart calculator

```
> x = 5  
> y = 3  
> x*y  
[1] 15  
> x+y  
[1] 8
```

# R building blocks

A very smart calculator

```
> x = 1:10  
> x  
[1] 1 2 3 4 5 6 7 8 9 10  
> z = x*3  
> z
```

# R building blocks

A very smart calculator

```
> x = 1:10  
> x  
[1] 1 2 3 4 5 6 7 8 9 10  
> z = x^3  
> z  
[1] 1 8 27 64 125 216 343 512 729 1000
```

# R data structures

Four basic atomic types

1. Character
2. Integer
3. Numeric
4. Logical

```
> x = "xyz"
> class(x)
[1] "character"
> x = 5L
> class(x)
[1] "integer"
> x = 5
> class(x)
[1] "numeric"
> x = TRUE
> class(x)
[1] "logical"
```

# Four main data structures to hold atomic types

1. Vector
2. Matrix
3. Data frame
4. List

# Four main data structures to hold atomic types

1. Vector

```
> x = vector("numeric", 10)
```

2. Matrix

```
> x
```

```
[1] 0 0 0 0 0 0 0 0 0 0
```

3. Data frame

4. List

# Four main data structures to hold atomic types

1. Vector

```
> x = vector("numeric", 10)
```

2. Matrix

```
> x
```

```
[1] 0 0 0 0 0 0 0 0 0 0
```

3. Data frame

```
> x = matrix(2, nrow= 3, ncol = 8)
```

```
> x
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
[1,]	2	2	2	2	2	2	2	2
[2,]	2	2	2	2	2	2	2	2
[3,]	2	2	2	2	2	2	2	2

4. List

# Four main data structures to hold atomic types

```
1. Vector > x = vector("numeric", 10)
2. Matrix > x
3. Data frame [1] 0 0 0 0 0 0 0 0 0 0
4. List > x = matrix(2, nrow= 3, ncol = 8)
> x
 [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,] 2 2 2 2 2 2 2 2
[2,] 2 2 2 2 2 2 2 2
[3,]

> x = data.frame(vec=vec, vec1=vec+1)
> x
  vec vec1
1   1    2
2   2    3
3   3    4
4   4    5
5   5    6
```

# Four main data structures to hold atomic types

1. Vector	> x = vector("numeric", 10)
2. Matrix	> x [1] 0 0 0 0 0 0 0 0 0 0
3. Data frame	
4. List	> x = matrix(2, nrow= 3, ncol = 8) > x [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [1,] 2 2 2 2 2 2 2 2 [2,] 2 2 2 2 2 2 2 2 [3,] 2 2 2 2 2 2 2 2

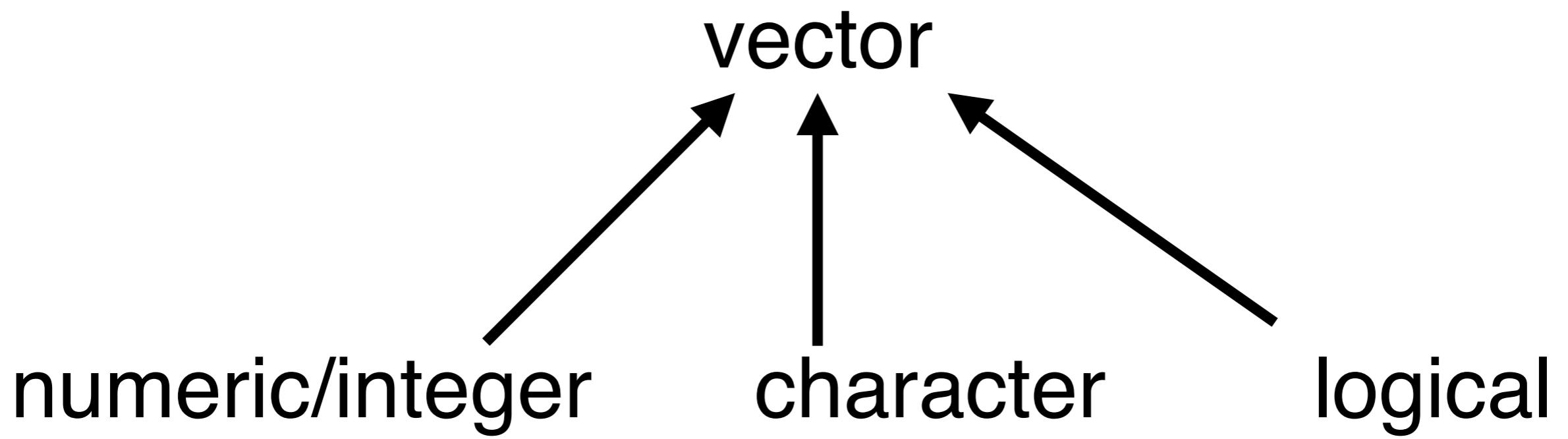
  

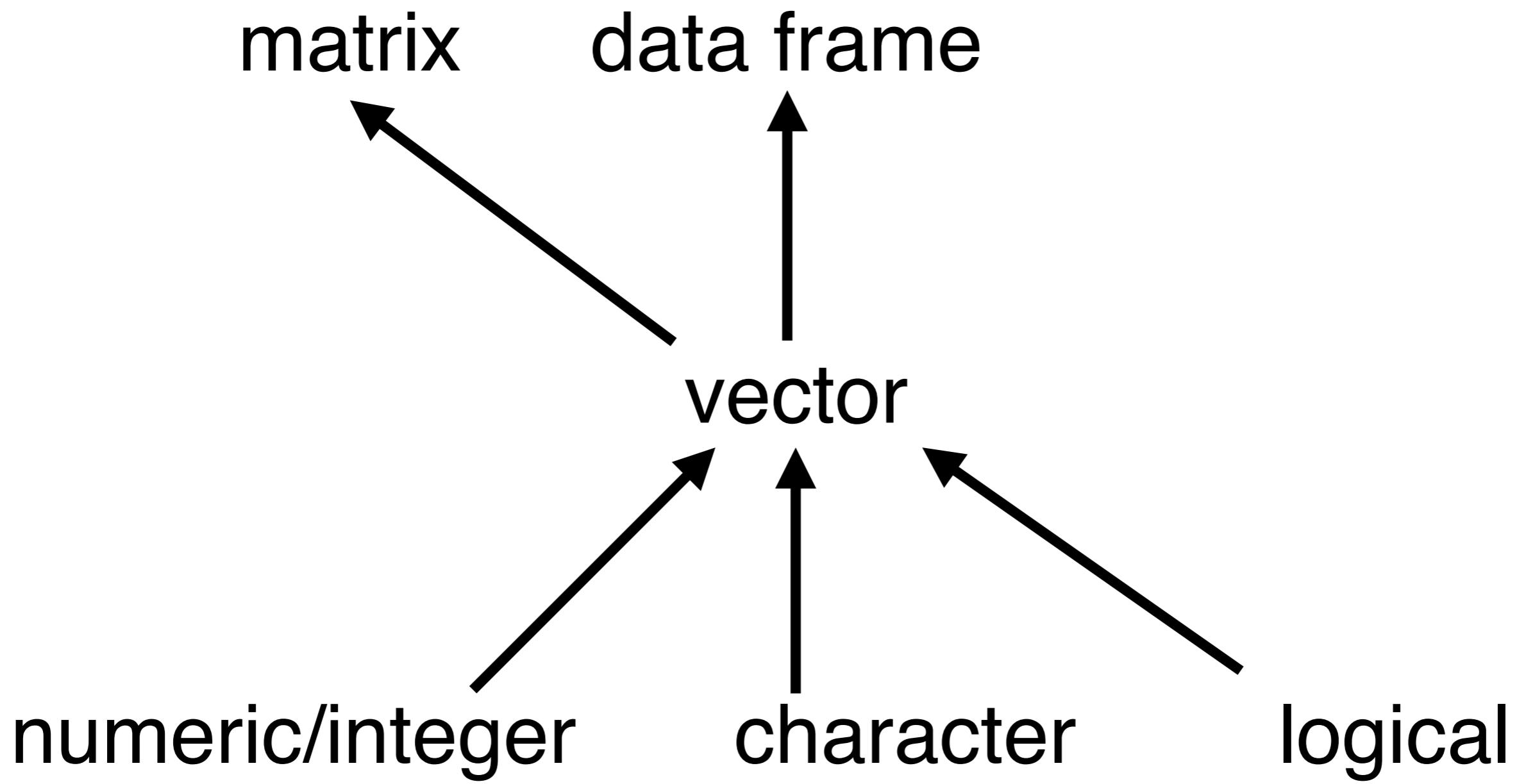
> x = data.frame(vec=vec, vec1=vec+1)	> list(1:5, 1:7, 1:3)
> x	[[1]] [1] 1 2 3 4 5
vec vec1	
1 1 2	
2 2 3	
3 3 4	[[2]] [1] 1 2 3 4 5 6 7
4 4 5	
5 5 6	[[3]] [1] 1 2 3

numeric/integer

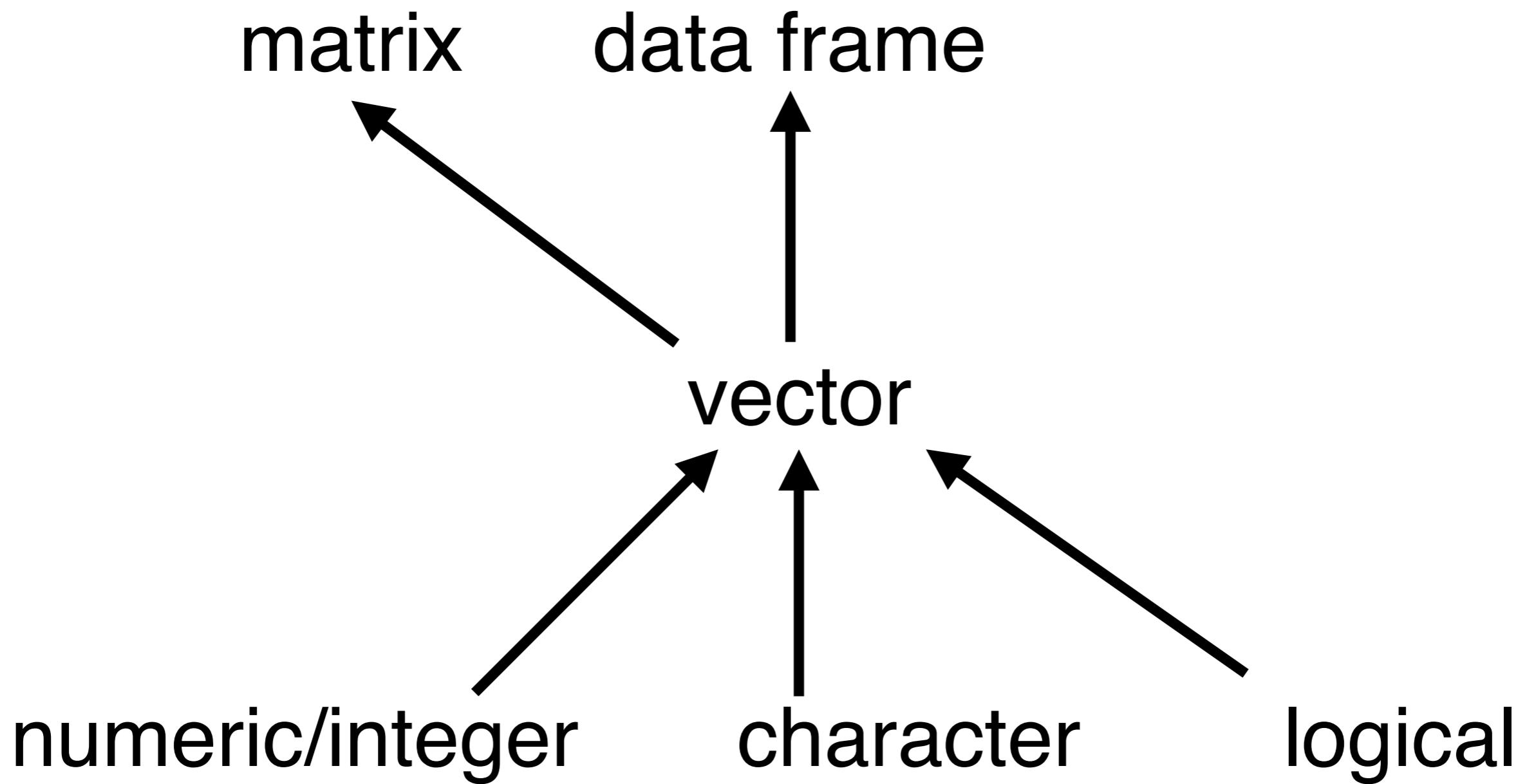
character

logical

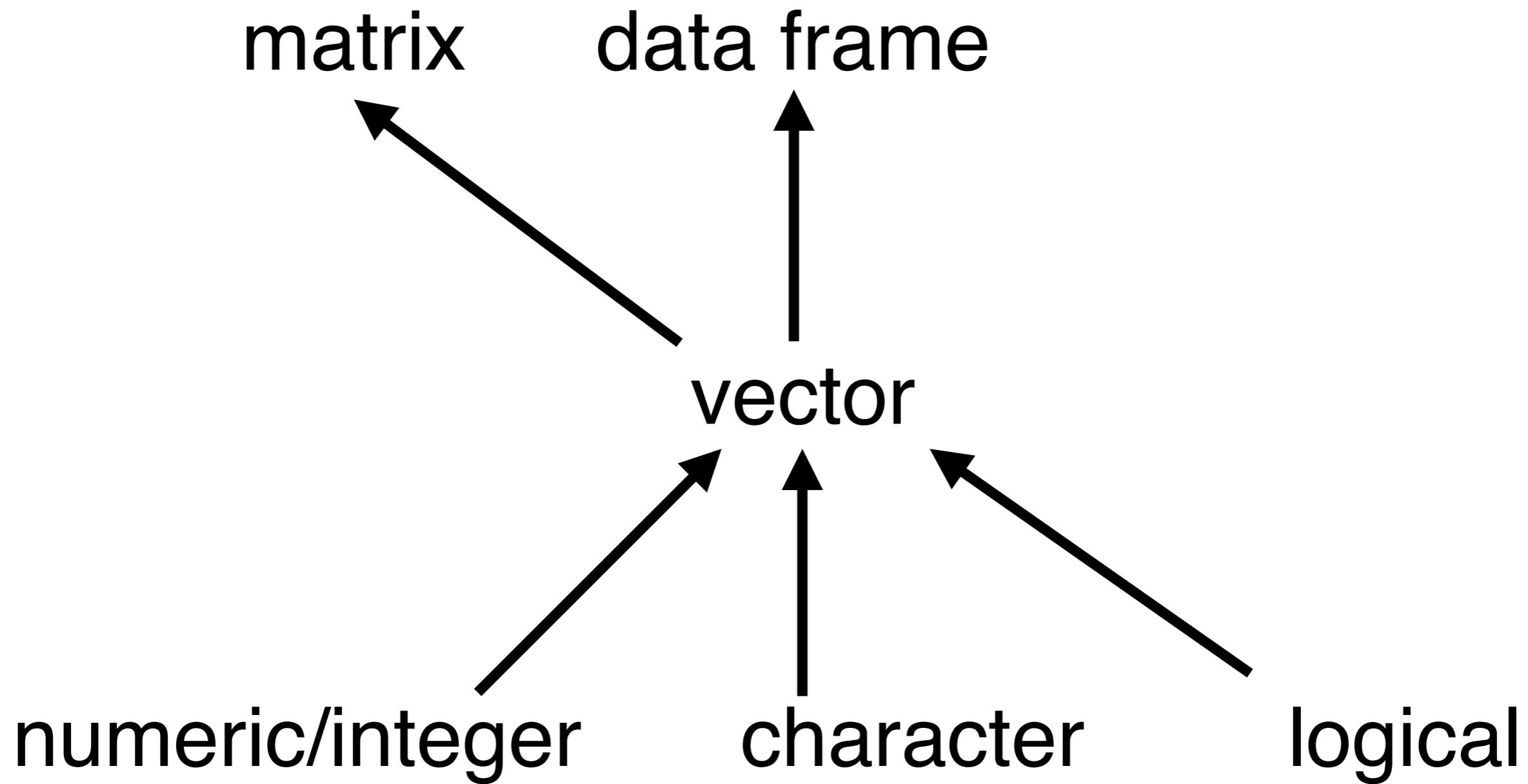




list



list



# You now have all of the main ingredients for your recipes

1. Numeric/character/logical
2. Vector
3. Matrix
4. Data frame
5. List



# R cooking techniques (some of them)

1. Manipulating data
2. Flow control
3. Calling functions



Warning: A lot of  
information is coming.  
Please ask questions as  
they come!

# manipulating data: vectors

```
> vec = c("a", "b", "c", "d", "b", "f")
> vec
[1] "a" "b" "c" "d" "b" "f"
> vec[-1]
[1] "b" "c" "d" "b" "f"
> vec[3:6]
[1] "c" "d" "b" "f"
```

# manipulating data: vectors

```
> vec = c("a", "b", "c", "d", "b", "f")
> vec
[1] "a" "b" "c" "d" "b" "f"
> vec[-1]
[1] "b" "c" "d" "b" "f"
> vec[3:6]
[1] "c" "d" "b" "f"
> vec=="b"
[1] FALSE  TRUE FALSE FALSE  TRUE FALSE
> indices = vec=="b"
> vec[indices]
[1] "b" "b"
```

# manipulating data: vectors

```
> vec = c("a", "b", "c", "d", "b", "f")
> vec
[1] "a" "b" "c" "d" "b" "f"
> vec[-1]
[1] "b" "c" "d" "b" "f"
> vec[3:6]
[1] "c" "d" "b" "f"
> vec=="b"
[1] FALSE TRUE FALSE FALSE TRUE FALSE
> indices = vec=="b"
> vec[indices]
[1] "b" "b"
> which(vec=="b")
[1] 2 5
> vec[which(vec=="b")]
[1] "b" "b"
```

# manipulating data: matrices

```
> matrix(1:25, ncol=5)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11   16   21
[2,]    2    7   12   17   22
[3,]    3    8   13   18   23
[4,]    4    9   14   19   24
[5,]    5   10   15   20   25
> mat = matrix(1:25, ncol=5)
> mat[2,3]
[1] 12
> mat[2:5,3]
[1] 12 13 14 15
> mat[1,]
[1] 1 6 11 16 21
> mat[,1]
[1] 1 2 3 4 5
```

# manipulating data: matrices (cont)

```
> mat
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11   16   21
[2,]    2    7   12   17   22
[3,]    3    8   13   18   23
[4,]    4    9   14   19   24
[5,]    5   10   15   20   25
> mat[,2]
[1] 6 7 8 9 10
> mat[which(mat[,2]==6), ]
[1] 1 6 11 16 21
```

# manipulating data: data frames

```
> df  
  temps water  
1    70   1500  
2    85   1495  
3    79   1490  
4    83   1450  
5    90   1400  
  
> df[,1]  
[1] 70 85 79 83 90  
  
> df$temp  
[1] 70 85 79 83 90
```

# manipulating data: data frames

```
> df
```

	temps	water
1	70	1500
2	85	1495
3	79	1490
4	83	1450
5	90	1400

# manipulating data: data frames

```
> df  
    temps water  
1      70   1500  
2      85   1495  
3      79   1490  
4      83   1450  
5      90   1400  
  
> which(df$water<1490)  
[1] 4 5  
> df$temp[which(df$water<1490)]  
[1] 83 90
```

# manipulating data: data frames

```
> df  
    temps water  
1      70   1500  
2      85   1495  
3      79   1490  
4      83   1450  
5      90   1400  
  
> which(df$water<1490)  
[1] 4 5  
  
> df$temp[which(df$water<1490)]  
[1] 83 90  
  
> df[which(df$water<1490), "temp"]  
[1] 83 90
```

# manipulating data: lists

```
> l  
[[1]]  
[1] 1 2 3 4 5  
  
[[2]]  
[1] 3 4 5 6 7 8 9 10  
  
[[3]]  
[1] 5 4 3  
  
> l[1]
```

# manipulating data: lists

```
> l  
[[1]]  
[1] 1 2 3 4 5
```

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

```
[[3]]  
[1] 5 4 3
```

```
> l[1]  
[[1]]  
[1] 1 2 3 4 5
```

```
> l[[1]]
```

# manipulating data: lists

```
> l  
[[1]]  
[1] 1 2 3 4 5
```

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

```
[[3]]  
[1] 5 4 3
```

```
> l[1]  
[[1]]  
[1] 1 2 3 4 5
```

```
> l[[1]]  
[1] 1 2 3 4 5
```

# manipulating data: lists

```
> l
[[1]]
[1] 1 2 3 4 5

[[2]]
[1] 3 4 5 6 7 8 9 10

[[3]]
[1] 5 4 3

> l[1]
[[1]]
[1] 1 2 3 4 5

> l[1:2]
[[1]]
[1] 1 2 3 4 5

> l[[1]]
[[2]]
[1] 3 4 5 6 7 8 9 10
```

# flow control: if/else

```
x = 10
if(x==10){
    print("x equals 10")
}
```

```
x = 7
if(x==10){
    print("x equals 10")
} else if(x==7){
    print("x equals 7")
} else {
    print("x doesn't equal either 7 or 10")
}
```

# flow control: for loop

```
vector = c(15, 30, 23, 23, 12, 24, 54, 64, 23)
count = 0
for(ii in vector){
  count = count + ii
}
count
[1] 268
```

# flow control: for loop

```
vector = c(15, 30, 23, 23, 12, 24, 54, 64, 23)
count = 0
for(ii in vector){
  count = count + ii
}
count
[1] 268
```

```
count = 0
for(ii in 1:length(vector)){
  count = count + vector[ii]
}
count
[1] 268
```

# functions

```
vector = c(15, 30, 23, 23, 12, 24, 54, 64, 23)
> sum(vector)
[1] 268
> mean(vector)
[1] 29.77778
> max(vector)
[1] 64
> min(vector)
[1] 12
> summary(vector)
   Min. 1st Qu. Median     Mean 3rd Qu. Max.
12.00    23.00   23.00    29.78   30.00   64.00
```

# functions

```
vector = c(15, 30, 23, 23, 12, 24, 54, 64, 23)
> sum(vector)
[1] 268
> mean(vector)
[1] 29.77778
> max(vector)
[1] 64
> min(vector)
[1] 12
> summary(vector)
   Min. 1st Qu. Median     Mean 3rd Qu.    Max.
12.00   23.00  23.00   29.78  30.00   64.00
```

There are a million useful functions, Google is your friend: “How to find number of rows in data frame in R?”

# functions

```
vector = c(15, 30, 23, 23, 12, 24, 54, 64, 23)
> sum(vector)
[1] 268
> mean(vector)
[1] 29.77778
> max(vector)
[1] 64
> min(vector)
[1] 12
> summary(vector)
   Min. 1st Qu. Median     Mean 3rd Qu. Max.
12.00    23.00   23.00    29.78   30.00   64.00
```

There are a million useful functions, Google is your friend: “How to find number of rows in data frame in R?”

`nrow()`

# functions

## Input and output

```
> ?sum
```

# functions

## Input and output

> ?sum

sum {base}

### Sum of Vector Elements

#### Description

sum returns the sum of all the values present in its arguments.

#### Usage

sum( . . . , na.rm = FALSE)

#### Arguments

... numeric or complex or logical vectors.

na.rm logical. Should missing values (including NaN) be removed?

# functions: creating your own

```
my_sum = function(vec){  
  count = 0  
  for(ii in vec){  
    count = count + ii  
  }  
  count  
}
```

# functions: creating your own

```
my_sum = function(vec){  
  count = 0  
  for(ii in vec){  
    count = count + ii  
  }  
  count  
}
```

```
> my_sum(vector)  
[1] 268
```

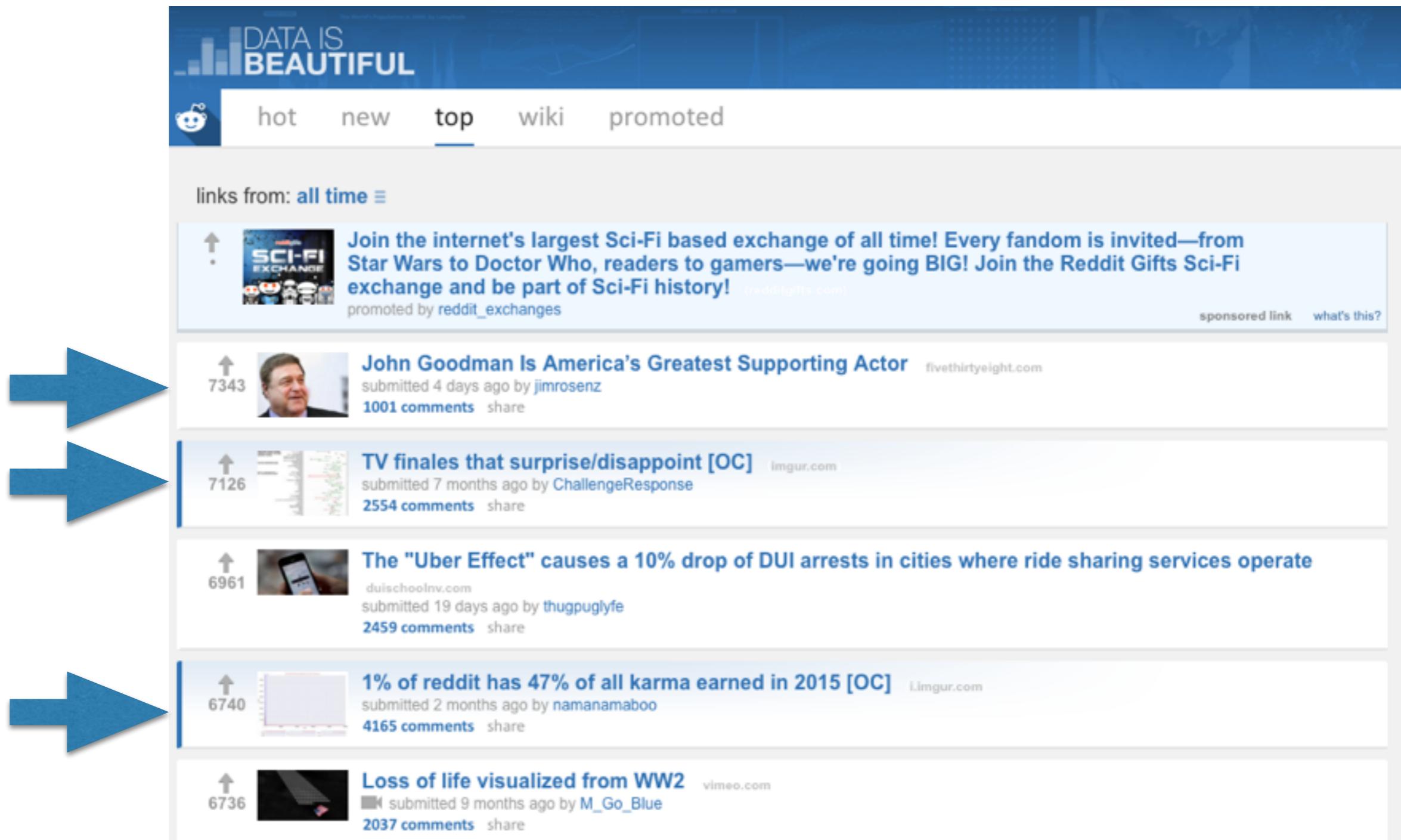
# functions: creating your own

```
my_sum = function(vec){  
  count = 0  
  for(ii in vec){  
    count = count + ii  
  }  
  count  
}  
  
> my_sum(vector)  
[1] 268  
  
> new_vec = c(vector, 12, 15, 23, 13, 16, 23, 43)  
> new_vec  
[1] 15 30 23 23 12 24 54 64 23 12 15 23 13 16 23 43  
> my_sum(new_vec)  
[1] 413
```

# 2nd Programming Exercise

1. open up 2\_data\_structures.R in Rstudio
2. Start playing with code

# Visualizing data

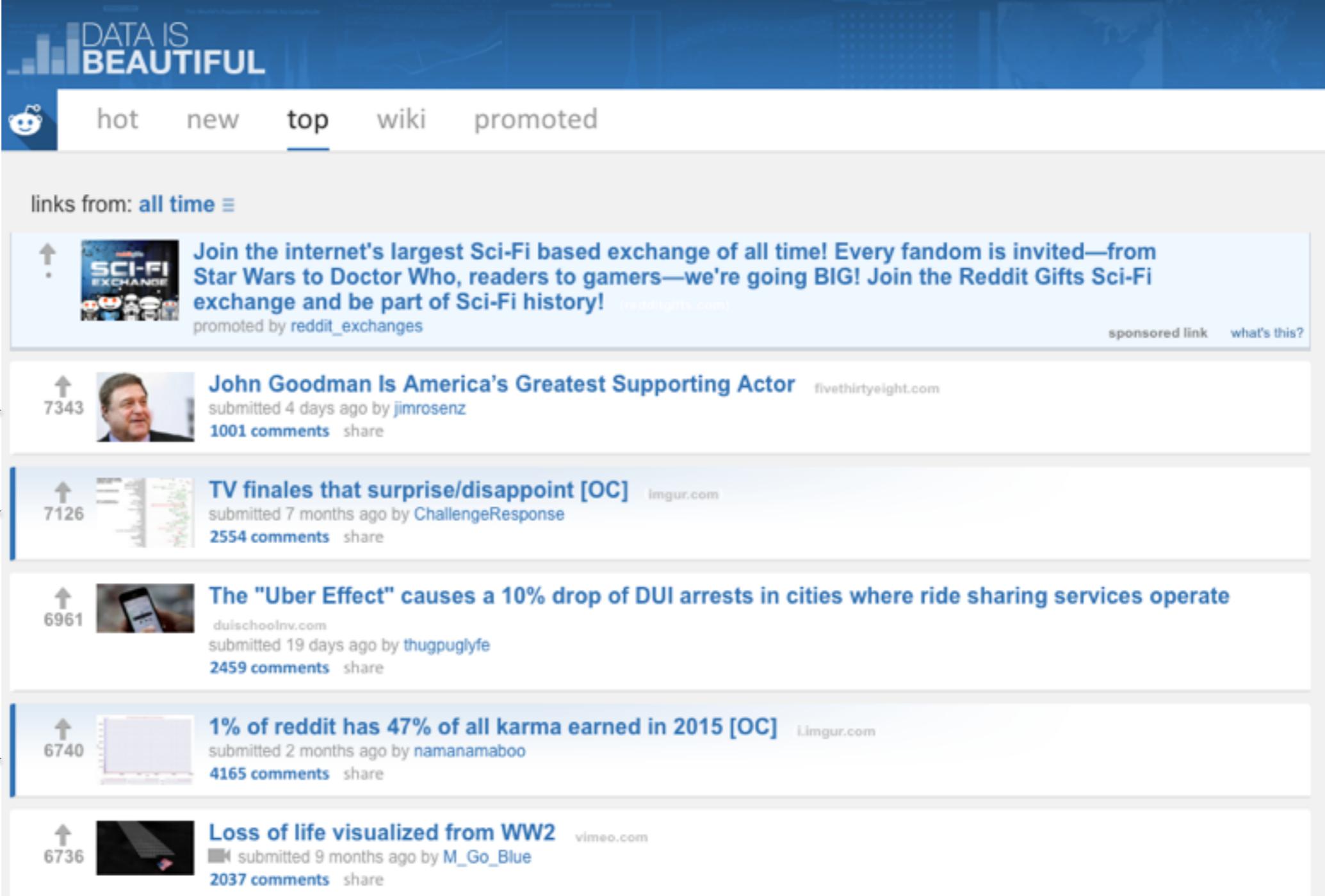


The screenshot shows the homepage of the [/r/dataisbeautiful](#) subreddit on Reddit. The page features a banner at the top with the text "DATA IS BEAUTIFUL". Below the banner, there is a navigation bar with links for "hot", "new", "top", "wiki", and "promoted". The "top" link is underlined, indicating the current view. A dropdown menu for "links from: all time" is visible. The main content area displays several posts:

- SCI-FI EXCHANGE**: Join the internet's largest Sci-Fi based exchange of all time! Every fandom is invited—from Star Wars to Doctor Who, readers to gamers—we're going BIG! Join the Reddit Gifts Sci-Fi exchange and be part of Sci-Fi history! (reddigifts.com) sponsored link what's this?
- John Goodman Is America's Greatest Supporting Actor** submitted 4 days ago by jimrosenz | 1001 comments | share
- TV finales that surprise/disappoint [OC]** submitted 7 months ago by ChallengeResponse | 2554 comments | share
- The "Uber Effect" causes a 10% drop of DUI arrests in cities where ride sharing services operate** duischoolinv.com submitted 19 days ago by thugpuglyfe | 2459 comments | share
- 1% of reddit has 47% of all karma earned in 2015 [OC]** submitted 2 months ago by namanamaboo | 4165 comments | share
- Loss of life visualized from WW2** submitted 9 months ago by M\_Go\_Blue | 2037 comments | share

[www.reddit.com/r/dataisbeautiful](http://www.reddit.com/r/dataisbeautiful)

# Visualizing data



The screenshot shows the homepage of the [/r/dataisbeautiful](#) subreddit on Reddit. The page features a blue header with the subreddit name and a navigation bar with links for "hot", "new", "top", "wiki", and "promoted". The "top" link is underlined, indicating the current view. Below the header, a banner for "DATA IS BEAUTIFUL" is visible. The main content area displays several data visualization posts:

- SCI-FI EXCHANGE**: Join the internet's largest Sci-Fi based exchange of all time! Every fandom is invited—from Star Wars to Doctor Who, readers to gamers—we're going BIG! Join the Reddit Gifts Sci-Fi exchange and be part of Sci-Fi history! [\(reddigifts.com\)](#) sponsored link what's this?
- John Goodman Is America's Greatest Supporting Actor** [\(fivethirtyeight.com\)](#)  
submitted 4 days ago by [jimrosenz](#) [1001 comments](#) [share](#)
- TV finales that surprise/disappoint [OC]** [\(imgur.com\)](#)  
submitted 7 months ago by [ChallengeResponse](#) [2554 comments](#) [share](#)
- The "Uber Effect" causes a 10% drop of DUI arrests in cities where ride sharing services operate** [\(duischoolinv.com\)](#)  
submitted 19 days ago by [thugpuglyfe](#) [2459 comments](#) [share](#)
- 1% of reddit has 47% of all karma earned in 2015 [OC]** [\(imgur.com\)](#)  
submitted 2 months ago by [namanamaboo](#) [4165 comments](#) [share](#)
- Loss of life visualized from WW2** [\(vimeo.com\)](#)  
submitted 9 months ago by [M\\_Go\\_Blue](#) [2037 comments](#) [share](#)

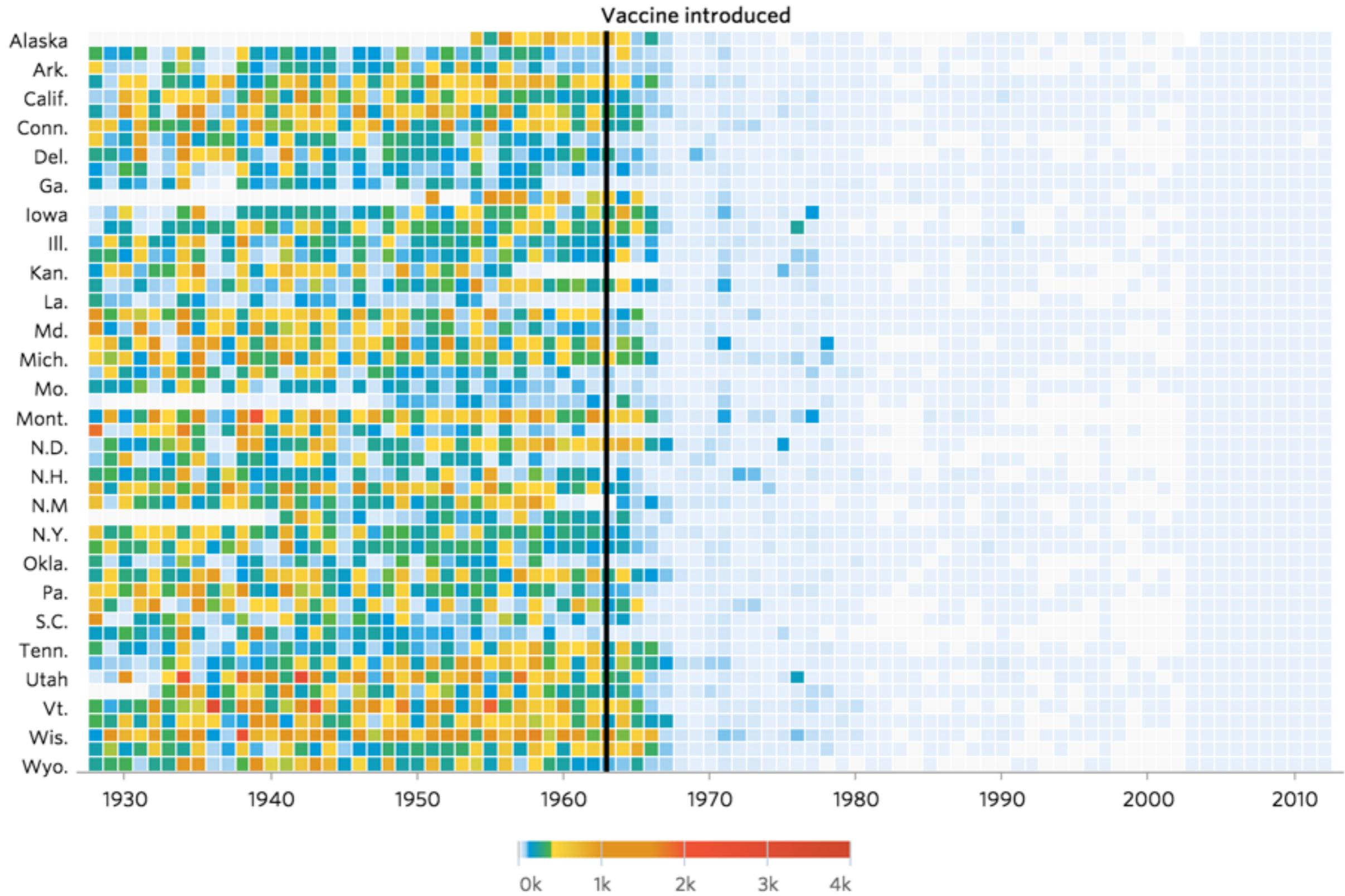
Three large blue arrows point from the left towards the first three posts on the page.

ggplot2

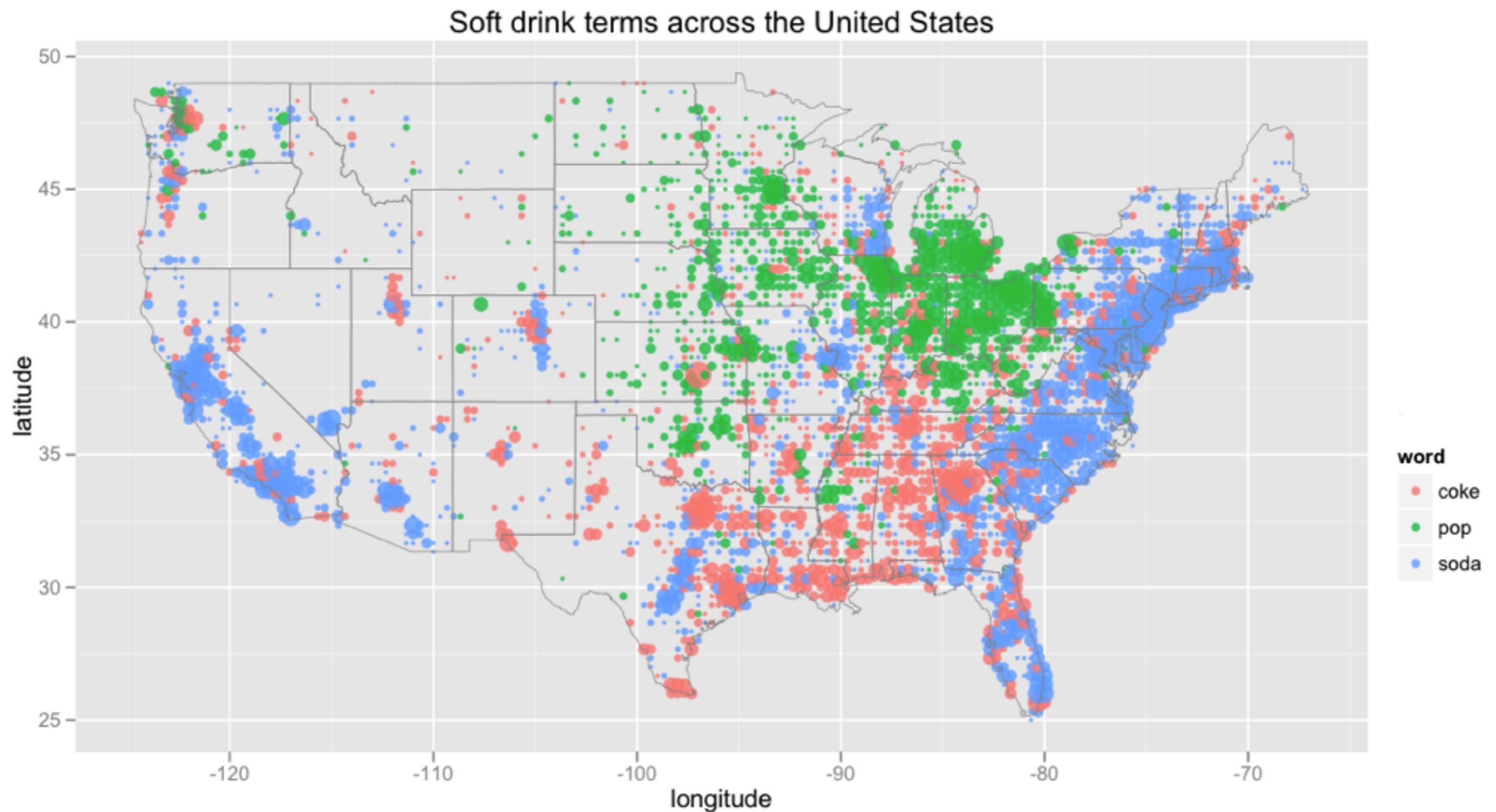
[www.reddit.com/r/dataisbeautiful](http://www.reddit.com/r/dataisbeautiful)

# ggplot2 visualizations

## Measles



# ggplot2 visualizations



# The grammar of graphics (ggplot)

# The grammar of graphics (ggplot)

## 1. Data

- Raw data for plotting

# The grammar of graphics (ggplot)

## 1. Data

- Raw data for plotting

## 2. Geometries

- The shape that will represent the data
  - point, line, bar, etc.

# The grammar of graphics (ggplot)

## 1. Data

- Raw data for plotting

## 2. Geometries

- The shape that will represent the data
  - point, line, bar, etc.

## 3. Aesthetics

- Color, size, shape, etc.

# The grammar of graphics (ggplot)

1. Data
  - Raw data for plotting
2. Geometries
  - The shape that will represent the data
    - point, line, bar, etc.
3. Aesthetics
  - Color, size, shape, etc.
4. Scales
  - Mapping data to aesthetic (how to color geoms, data range to plot, etc)

# A simple example

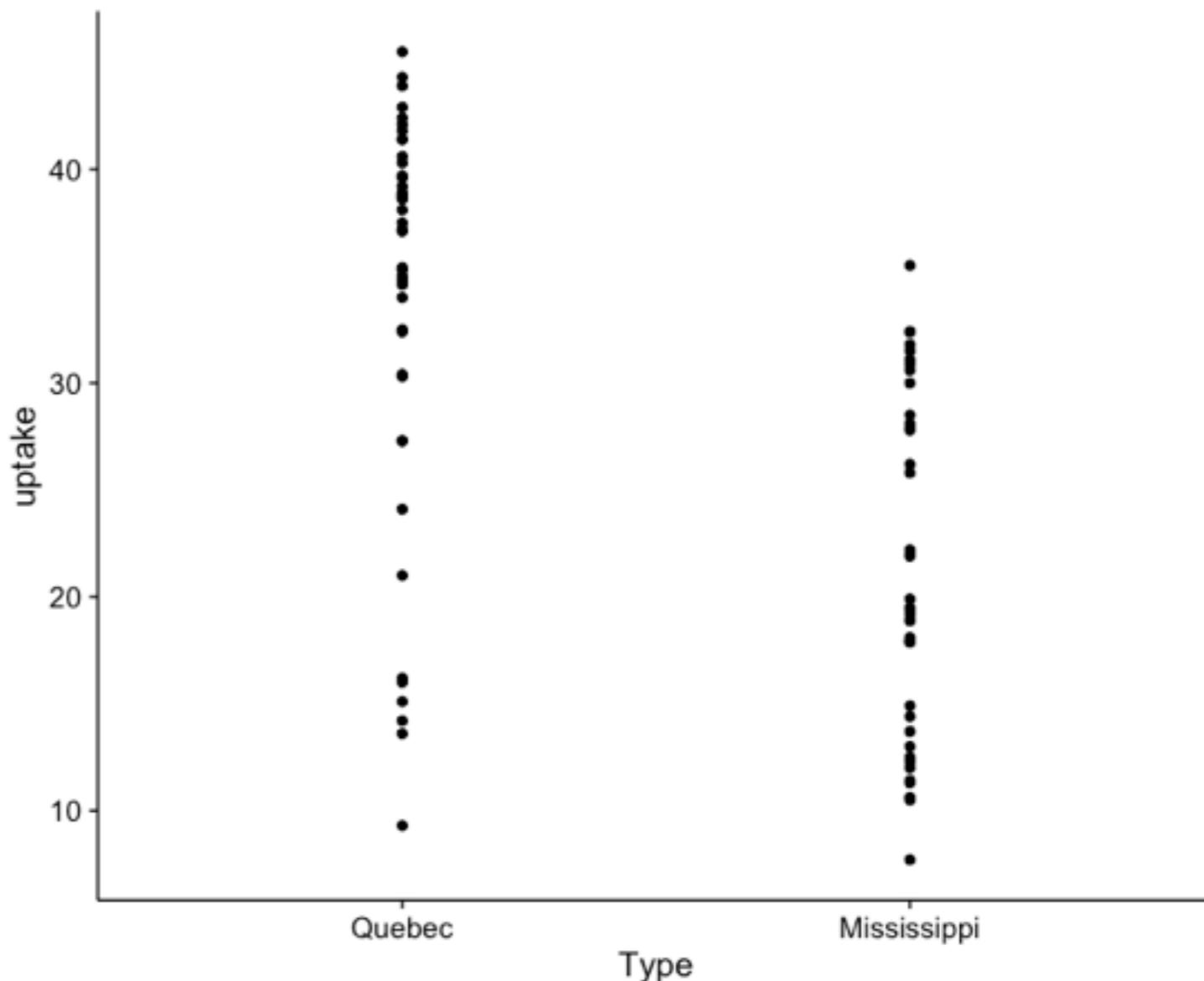
```
> head(CO2)
```

	Plant	Type	Treatment	conc	uptake
1	Qn1	Quebec	nonchilled	95	16.0
2	Qn1	Quebec	nonchilled	175	30.4
3	Qn1	Quebec	nonchilled	250	34.8
4	Qn1	Quebec	nonchilled	350	37.2
5	Qn1	Quebec	nonchilled	500	35.3
6	Qn1	Quebec	nonchilled	675	39.2

# A simple example

```
> head(CO2)
```

	Plant	Type	Treatment	conc	uptake
1	Qn1	Quebec	nonchilled	95	16.0
2	Qn1	Quebec	nonchilled	175	30.4
3	Qn1	Quebec	nonchilled	250	34.8
4	Qn1	Quebec	nonchilled	350	37.2
5	Qn1	Quebec	nonchilled	500	35.3
6	Qn1	Quebec	nonchilled	675	39.2

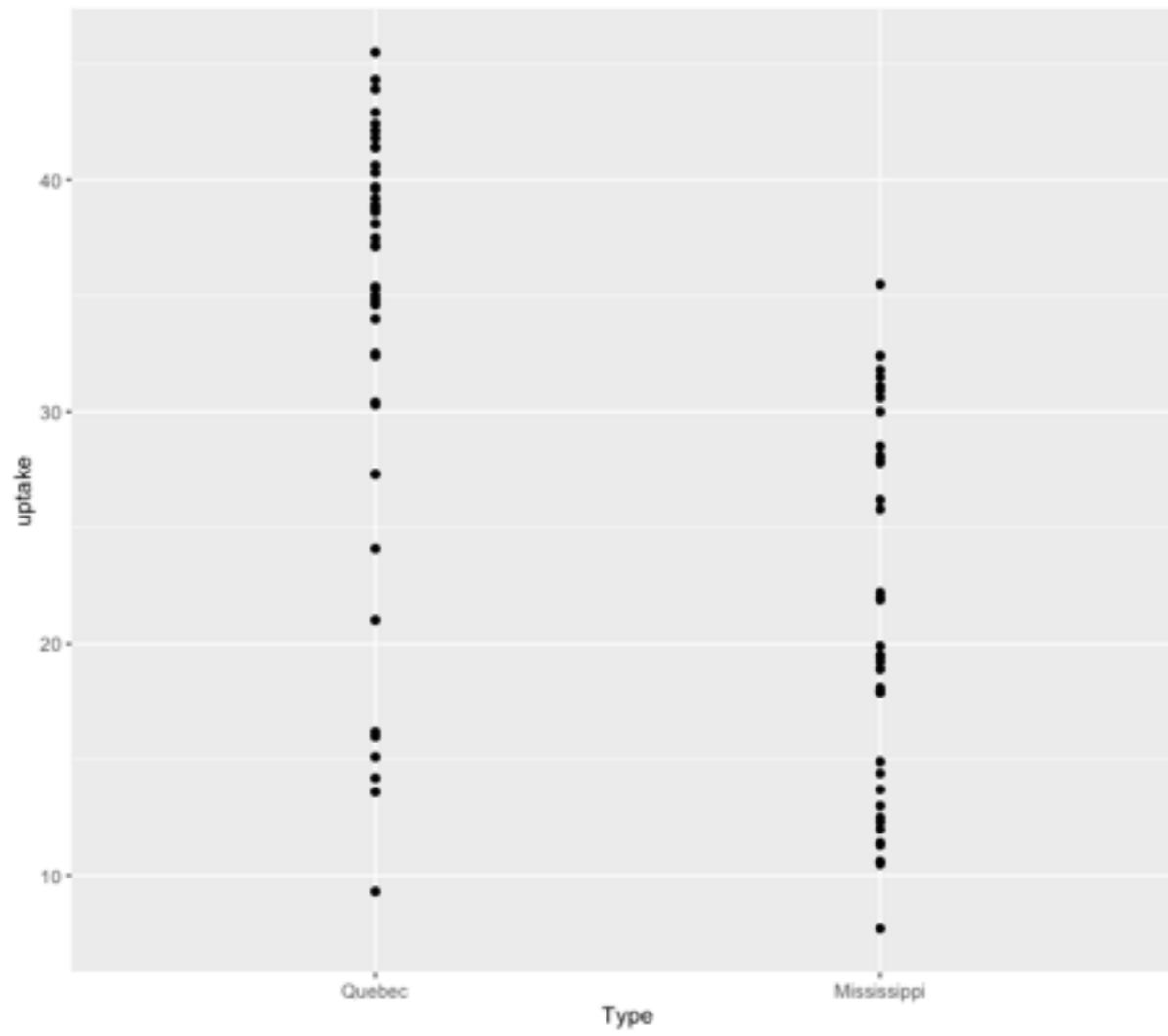


```
ggplot(CO2, aes(x=Type, y=uptake))+geom_point()
```

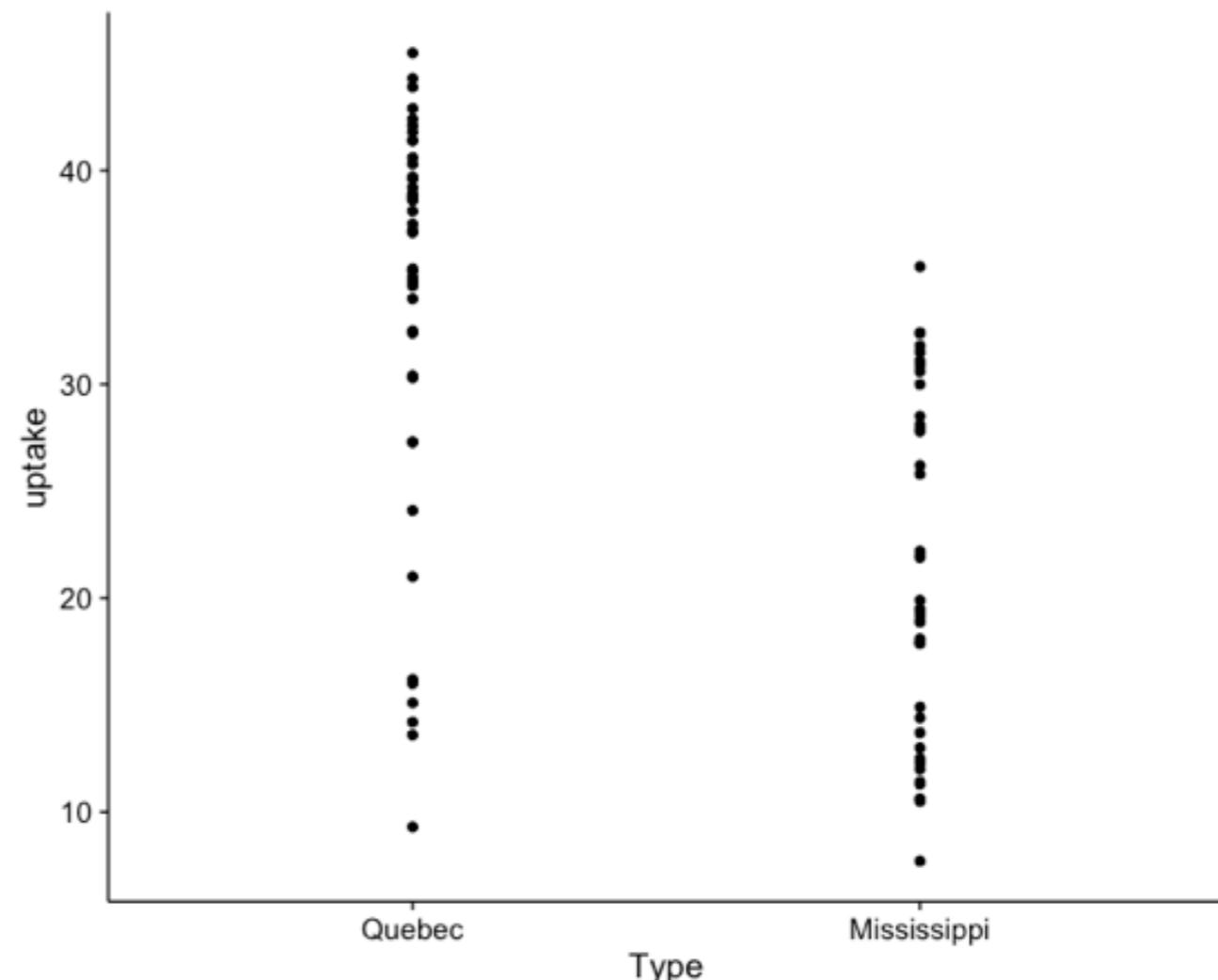
# A simple example

note that this uses “cowplot,” because I can’t stand ggplot2 default themes

ggplot2 default



cowplot default



```
ggplot(CO2, aes(x=Type, y=uptake))+geom_point()
```

# A simple example

```
> head(CO2)
```

	Plant	Type	Treatment	conc	uptake
1	Qn1	Quebec	nonchilled	95	16.0
2	Qn1	Quebec	nonchilled	175	30.4
3	Qn1	Quebec	nonchilled	250	34.8
4	Qn1	Quebec	nonchilled	350	37.2
5	Qn1	Quebec	nonchilled	500	35.3
6	Qn1	Quebec	nonchilled	675	39.2

```
ggplot(CO2, aes(x=Type, y=uptake))+geom_point()
```

# A simple example

```
> head(CO2)
```

	Plant	Type	Treatment	conc	uptake
1	Qn1	Quebec	nonchilled	95	16.0
2	Qn1	Quebec	nonchilled	175	30.4
3	Qn1	Quebec	nonchilled	250	34.8
4	Qn1	Quebec	nonchilled	350	37.2
5	Qn1	Quebec	nonchilled	500	35.3
6	Qn1	Quebec	nonchilled	675	39.2

```
ggplot(CO2, aes(x=Type, y=uptake))+geom_point()
```



Data frame

# A simple example

```
> head(C02)
```

	Plant	Type	Treatment	conc	uptake
1	Qn1	Quebec	nonchilled	95	16.0
2	Qn1	Quebec	nonchilled	175	30.4
3	Qn1	Quebec	nonchilled	250	34.8
4	Qn1	Quebec	nonchilled	350	37.2
5	Qn1	Quebec	nonchilled	500	35.3
6	Qn1	Quebec	nonchilled	675	39.2

```
ggplot(C02, aes(x=Type, y=uptake))+geom_point()
```



Data frame Aesthetics

# A simple example

```
> head(CO2)
```

	Plant	Type	Treatment	conc	uptake
1	Qn1	Quebec	nonchilled	95	16.0
2	Qn1	Quebec	nonchilled	175	30.4
3	Qn1	Quebec	nonchilled	250	34.8
4	Qn1	Quebec	nonchilled	350	37.2
5	Qn1	Quebec	nonchilled	500	35.3
6	Qn1	Quebec	nonchilled	675	39.2

```
ggplot(CO2, aes(x=Type, y=uptake))+geom_point()
```

↑  
Data frame    Aesthetics

↑  
Geometry

# A simple example

```
> head(CO2)
```

	Plant	Type	Treatment	conc	uptake
1	Qn1	Quebec	nonchilled	95	16.0
2	Qn1	Quebec	nonchilled	175	30.4
3	Qn1	Quebec	nonchilled	250	34.8
4	Qn1	Quebec	nonchilled	350	37.2
5	Qn1	Quebec	nonchilled	500	35.3
6	Qn1	Quebec	nonchilled	675	39.2

```
ggplot(CO2, aes(x=Type, y=uptake))+geom_point()
```

Data frame    Aesthetics

Geometry  
Link with +

# A simple example

```
> head(CO2)
```

	Plant	Type	Treatment	conc	uptake
1	Qn1	Quebec	nonchilled	95	16.0
2	Qn1	Quebec	nonchilled	175	30.4
3	Qn1	Quebec	nonchilled	250	34.8
4	Qn1	Quebec	nonchilled	350	37.2
5	Qn1	Quebec	nonchilled	500	35.3
6	Qn1	Quebec	nonchilled	675	39.2

```
ggplot(CO2, aes(x=Type, y=uptake))+geom_point()
```

Data frame

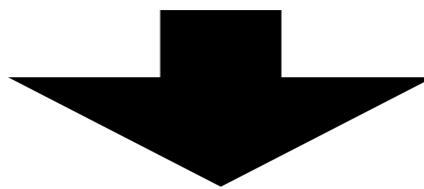
Aesthetics

data column names

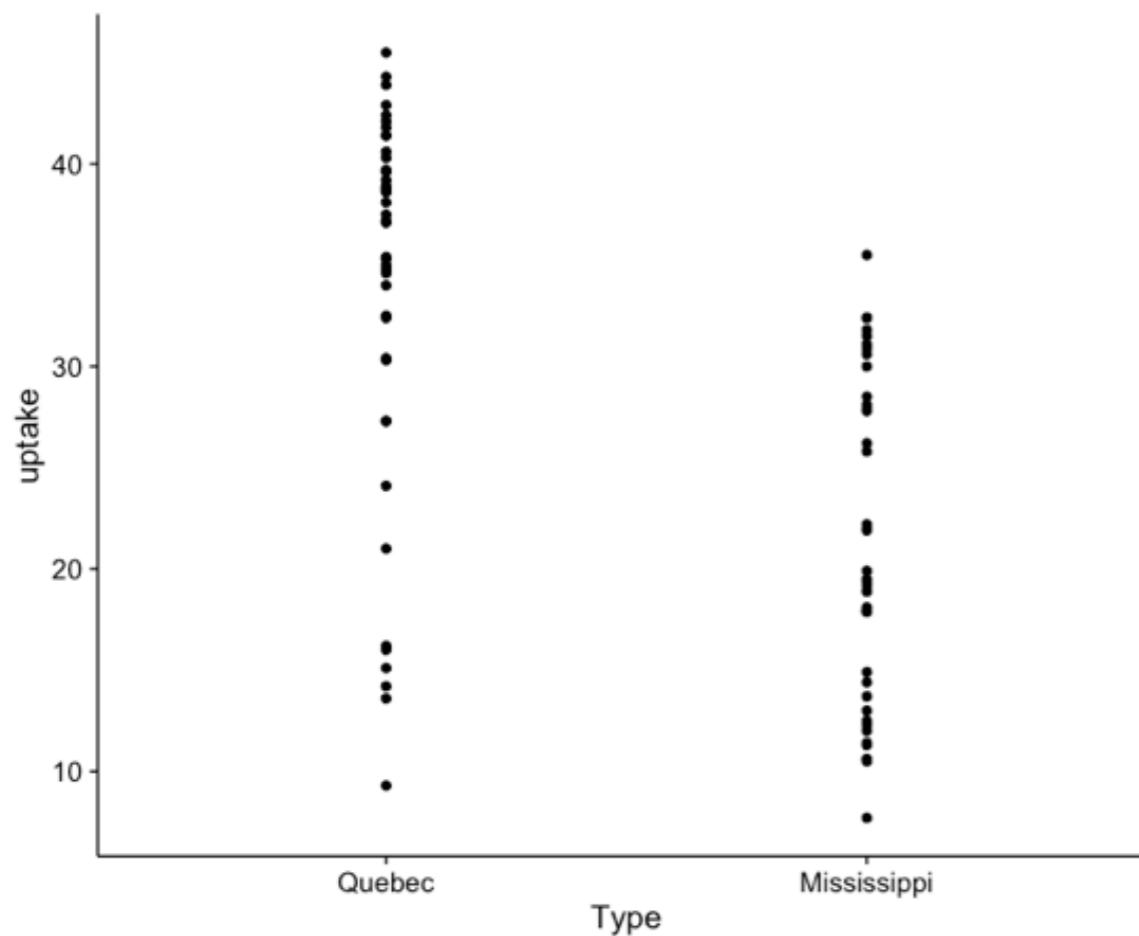
Geometry

Link with +

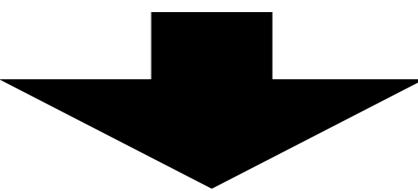
```
ggplot(CO2, aes(x=Type, y=uptake))+geom_point()
```



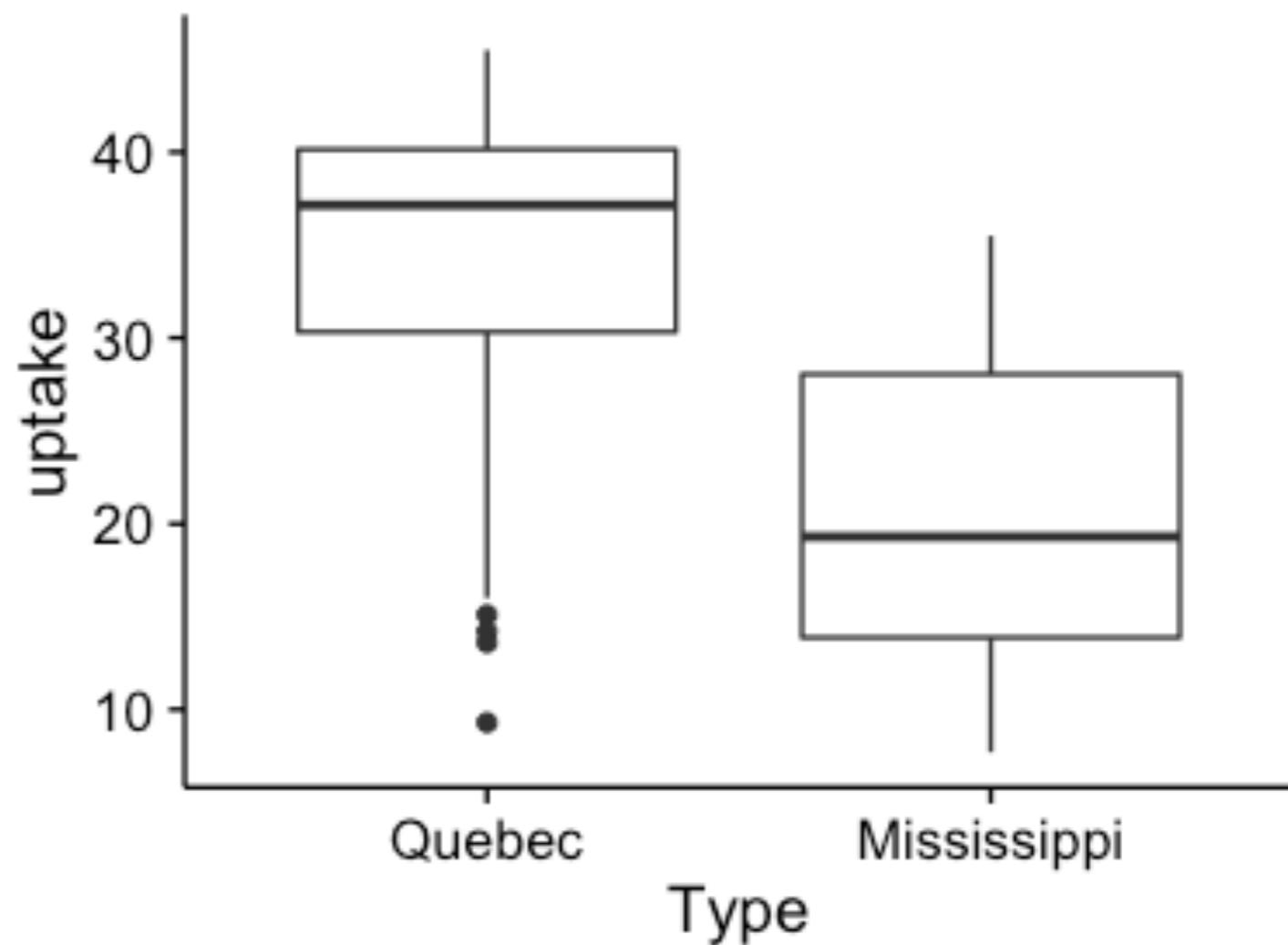
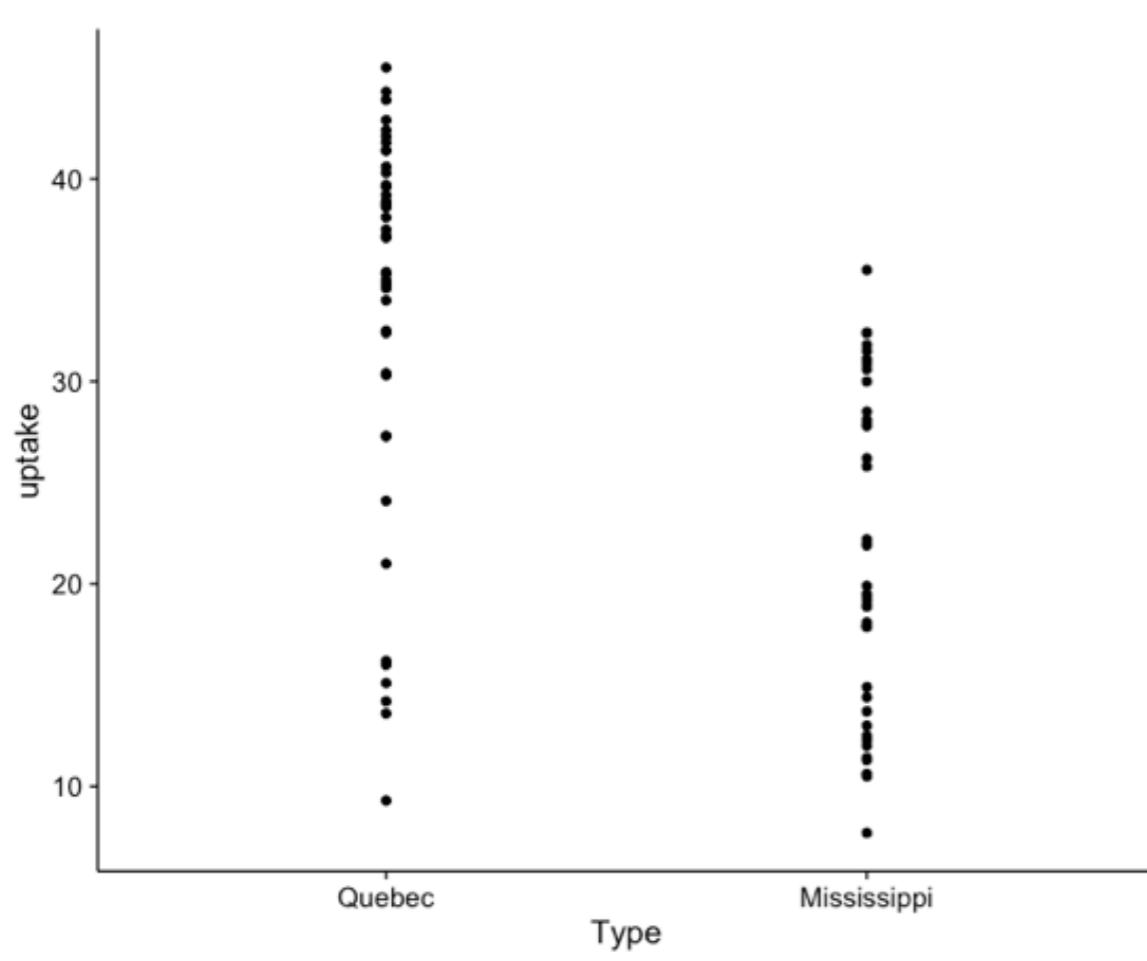
```
ggplot(CO2, aes(x=Type, y=uptake))+geom_boxplot()
```



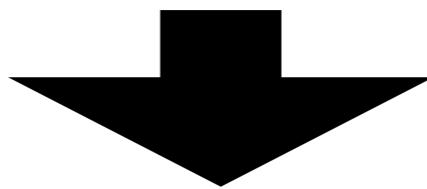
```
ggplot(CO2, aes(x=Type, y=uptake))+geom_point()
```



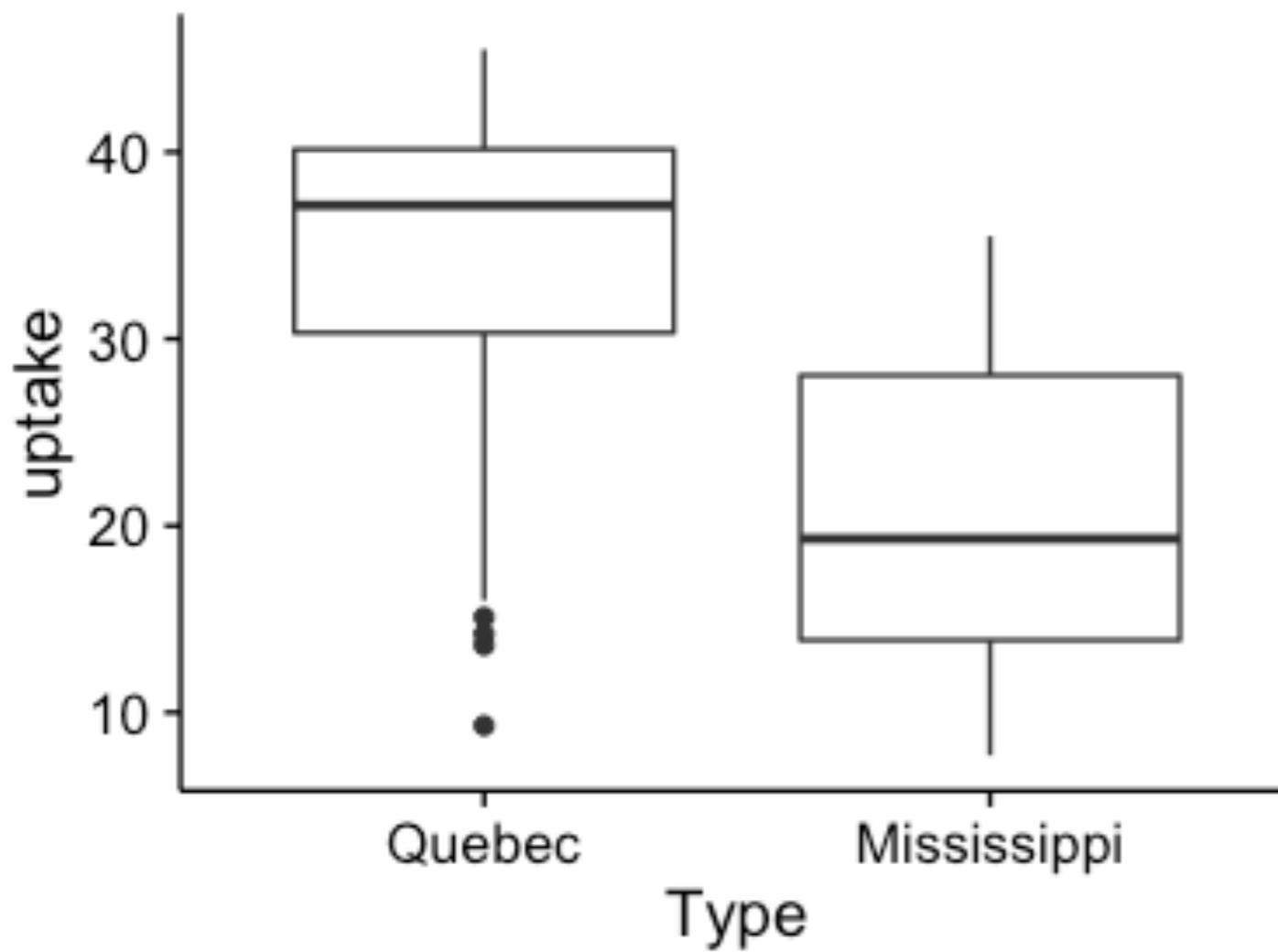
```
ggplot(CO2, aes(x=Type, y=uptake))+geom_boxplot()
```



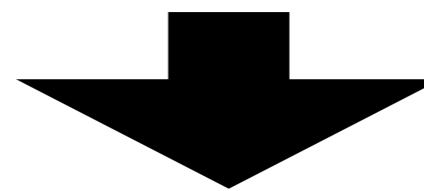
```
ggplot(CO2, aes(x=Type, y=uptake))+geom_boxplot()
```



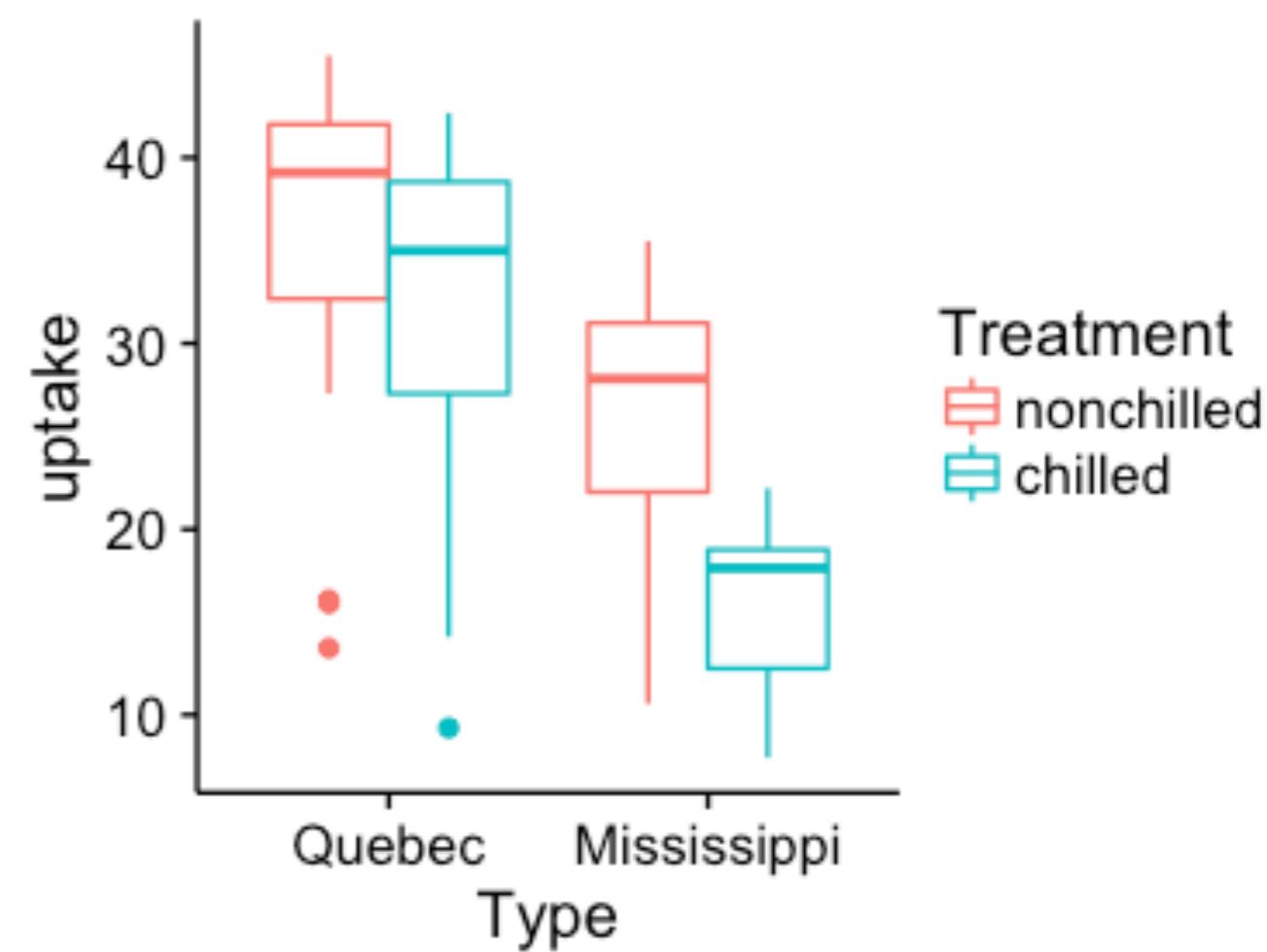
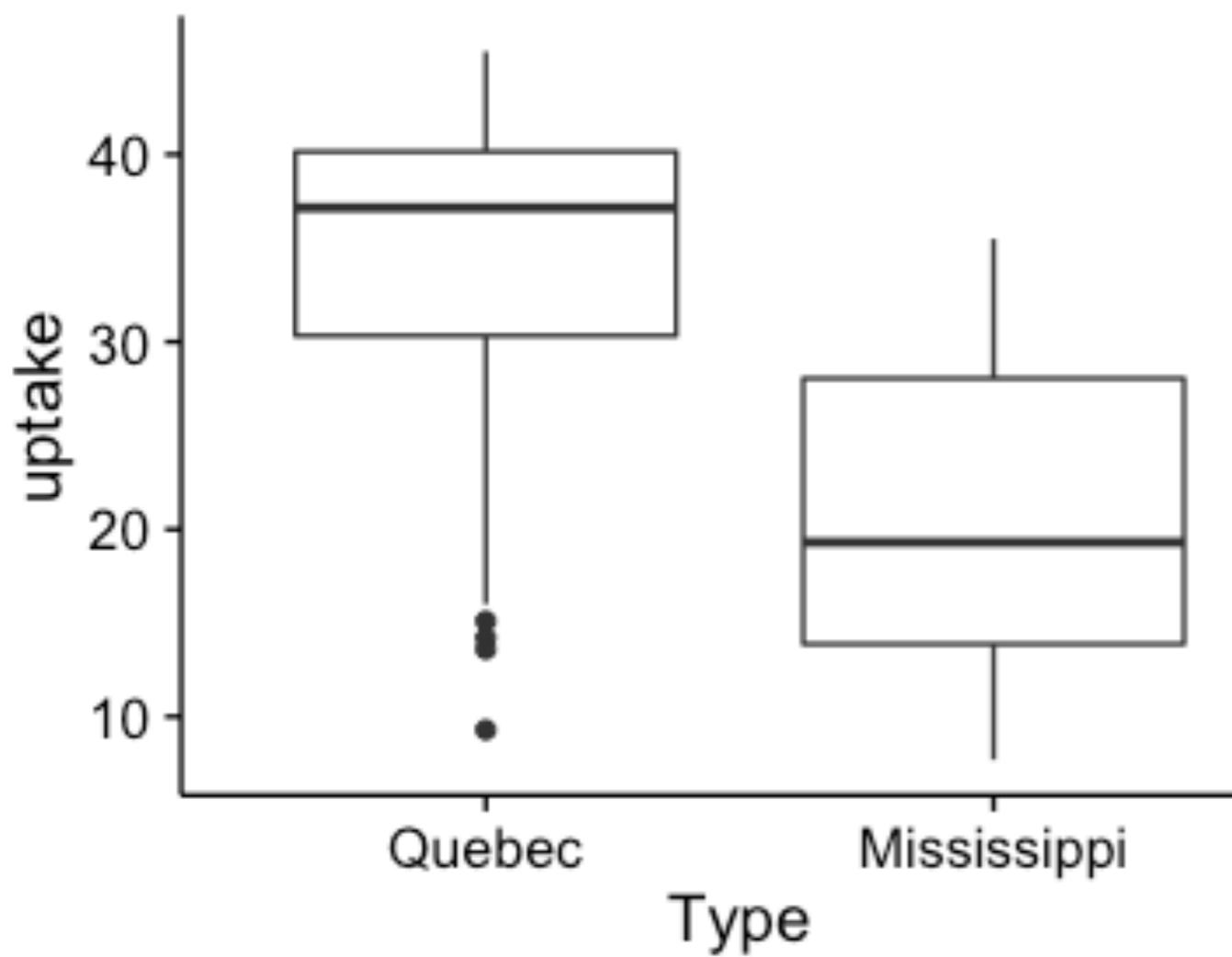
```
ggplot(CO2, aes(x=Type, y=uptake, color = Treatment))+  
  geom_boxplot()
```



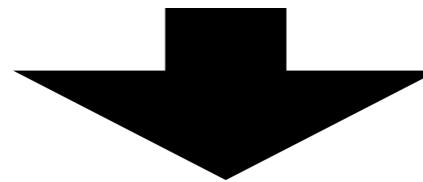
```
ggplot(CO2, aes(x=Type, y=uptake))+geom_boxplot()
```



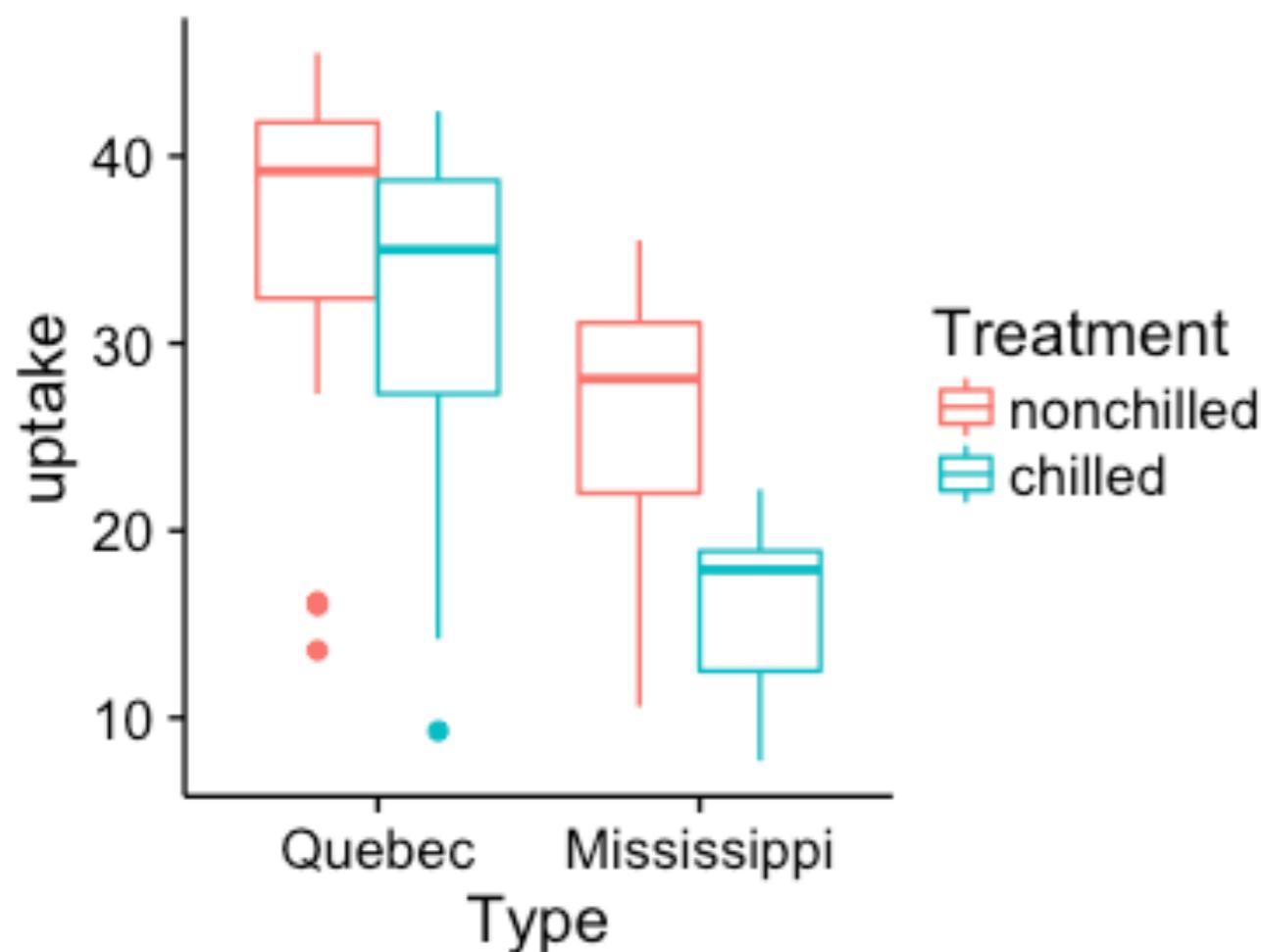
```
ggplot(CO2, aes(x=Type, y=uptake, color = Treatment))+  
  geom_boxplot()
```



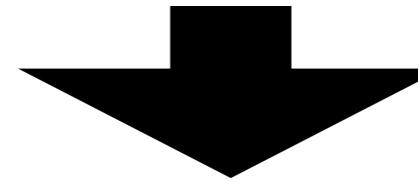
```
ggplot(CO2, aes(x=Type, y=uptake, color = Treatment))+  
  geom_boxplot()
```



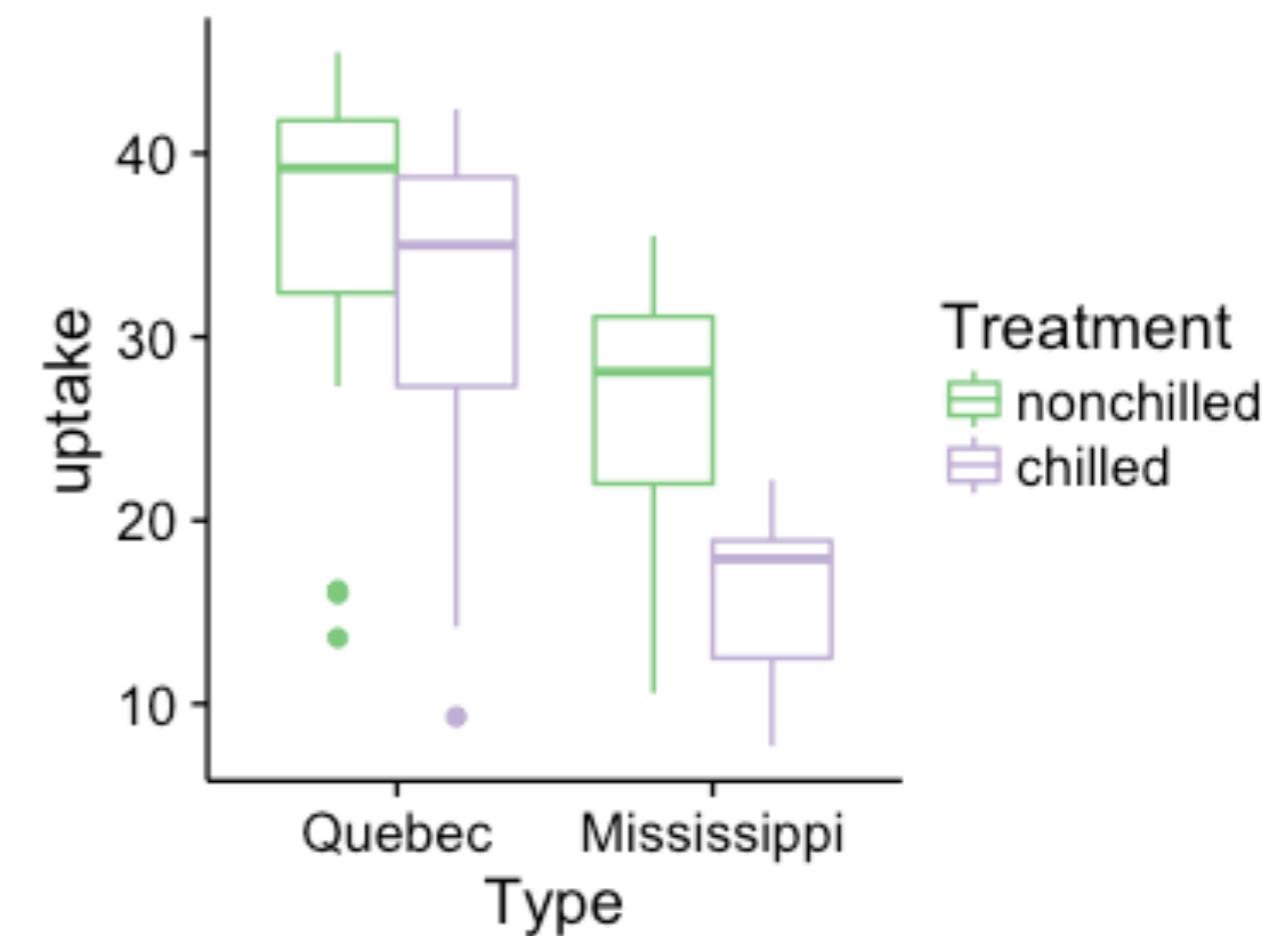
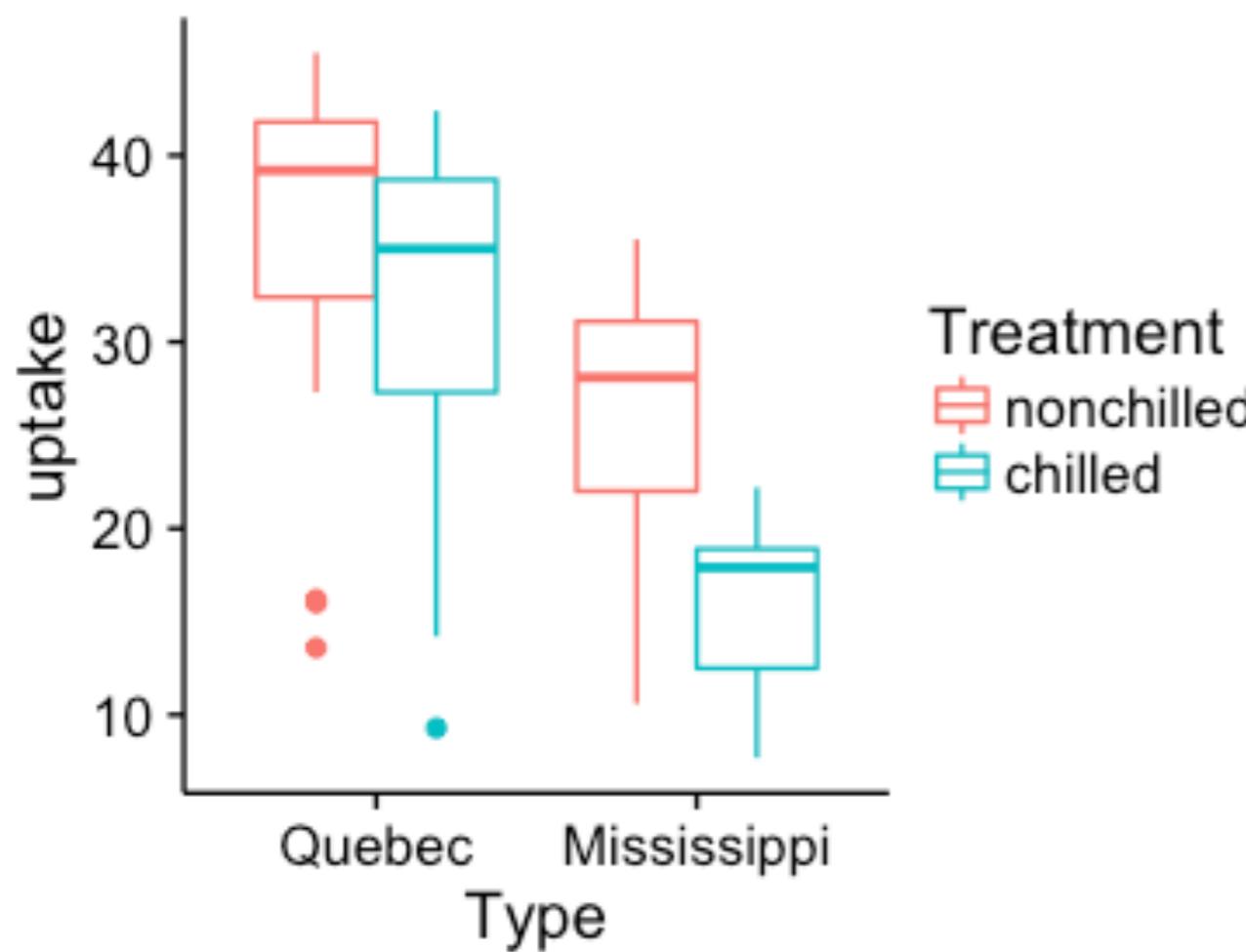
```
ggplot(CO2, aes(x=Type, y=uptake, color = Treatment))+  
  geom_boxplot() + scale_color_brewer(type = "qual")
```



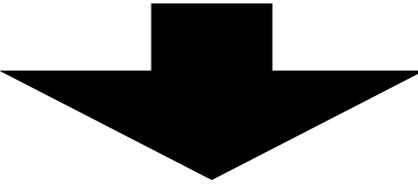
```
ggplot(CO2, aes(x=Type, y=uptake, color = Treatment))+  
  geom_boxplot()
```



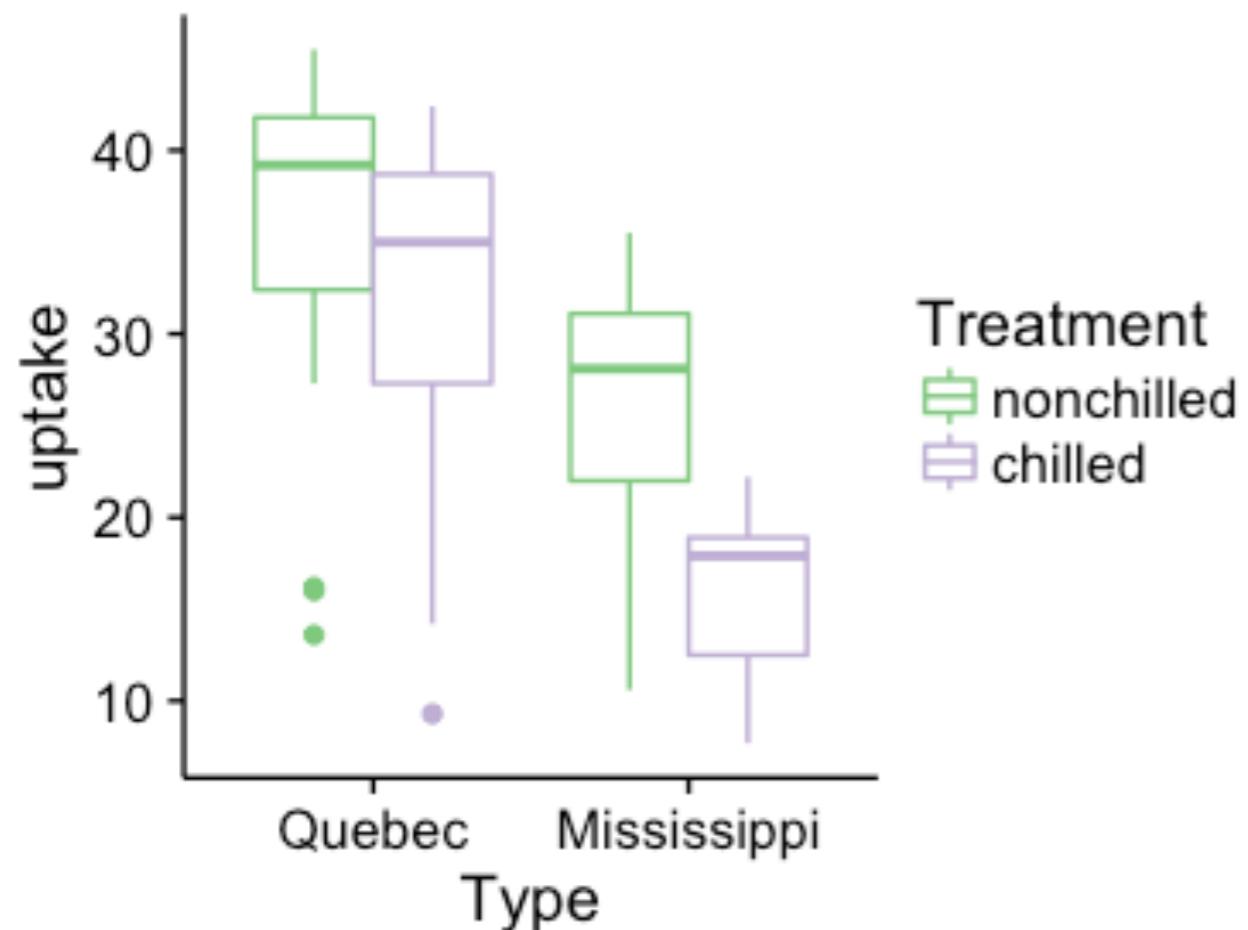
```
ggplot(CO2, aes(x=Type, y=uptake, color = Treatment))+  
  geom_boxplot() + scale_color_brewer(type = "qual")
```



```
ggplot(CO2, aes(x=Type, y=uptake, color = Treatment))+  
  geom_boxplot() + scale_color_brewer(type = "qual")
```



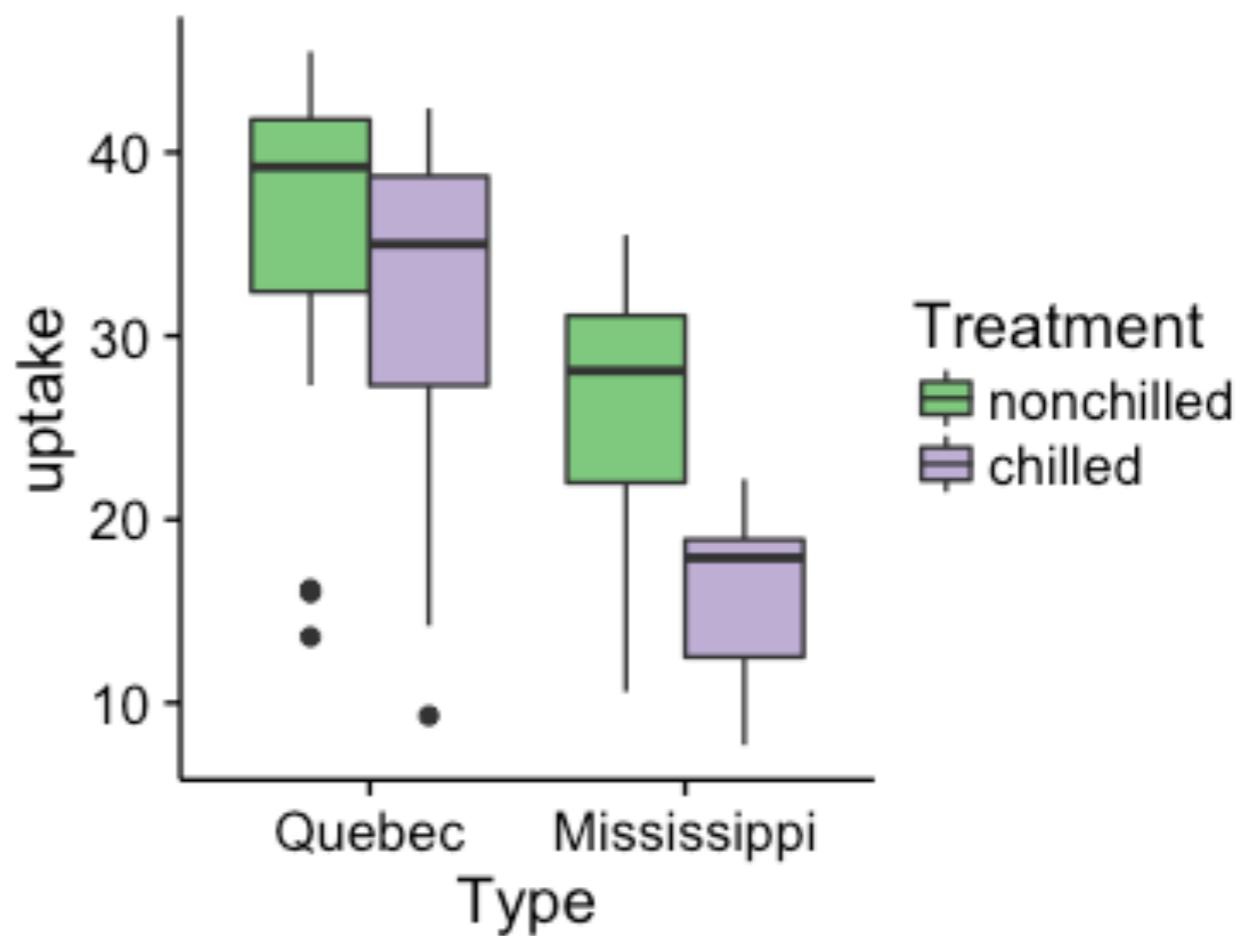
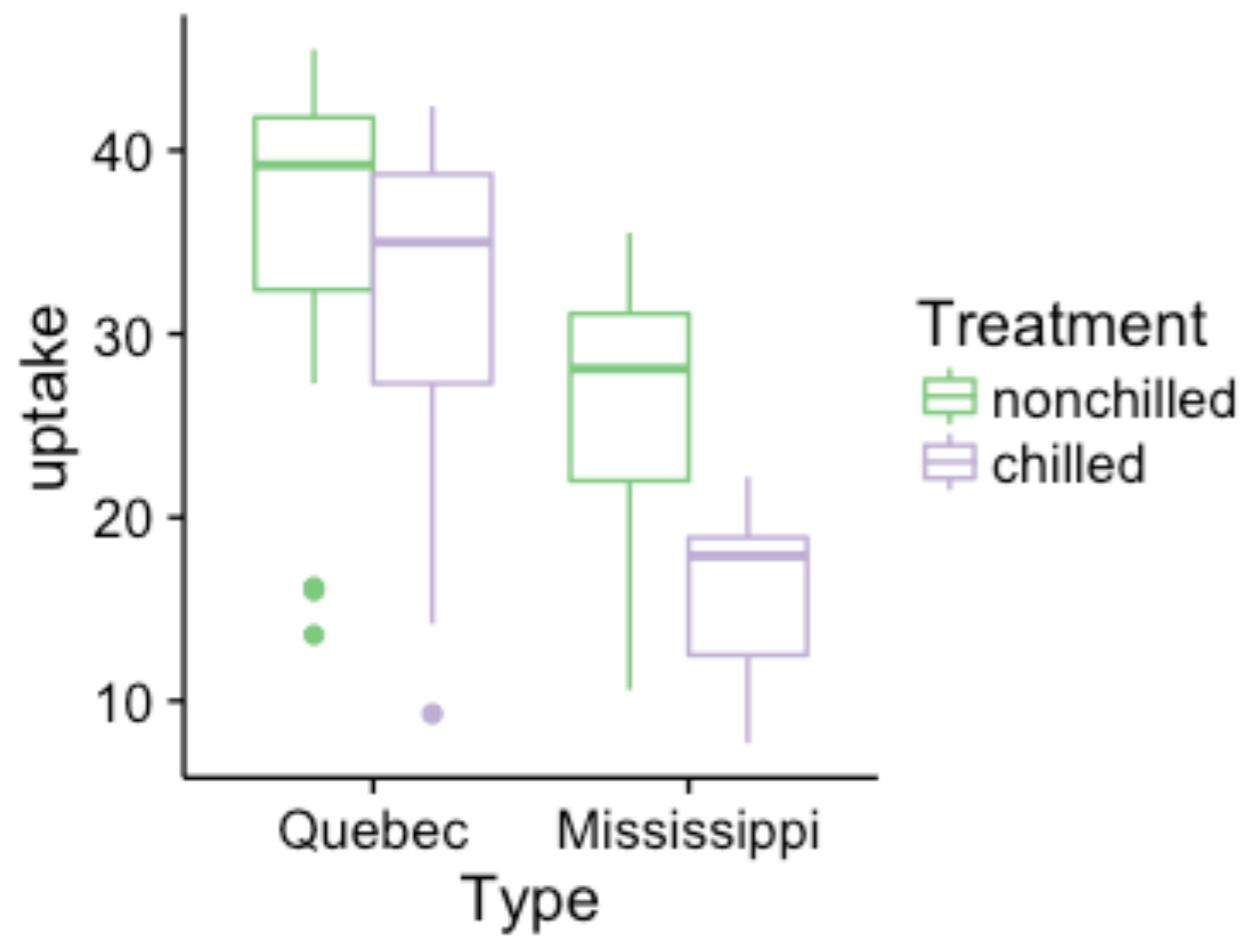
```
ggplot(CO2, aes(x=Type, y=uptake, fill = Treatment))+  
  geom_boxplot() + scale_fill_brewer(type = "qual")
```



```
ggplot(CO2, aes(x=Type, y=uptake, color = Treatment))+  
  geom_boxplot() + scale_color_brewer(type = "qual")
```



```
ggplot(CO2, aes(x=Type, y=uptake, fill = Treatment))+  
  geom_boxplot() + scale_fill_brewer(type = "qual")
```



# Frequently used geoms + aesthetics

- `geom_bar()` • `color`
- `geom_line()` • `size`
- `geom_point()` • `fill`
- `geom_histogram()` • `alpha`
- `geom_ribbon()` • `shape`
- `geom_text()` • `linetype`
- `geom_boxplot()` • `group`

<http://docs.ggplot2.org/current/>

# Second example

```
> head(txhousing)
```

	city	year	month	sales	volume	median	listings	inventory	date
1	Abilene	2000	1	72	5380000	71400	701	6.3	2000.000
2	Abilene	2000	2	98	6505000	58700	746	6.6	2000.083
3	Abilene	2000	3	130	9285000	58100	784	6.8	2000.167
4	Abilene	2000	4	98	9730000	68600	785	6.9	2000.250
5	Abilene	2000	5	141	10590000	67300	794	6.8	2000.333
6	Abilene	2000	6	156	13910000	66900	780	6.6	2000.417

# Second example

```
> head(txhousing)
  city year month sales volume median listings inventory date
1 Abilene 2000     1    72  5380000  71400      701       6.3 2000.000
2 Abilene 2000     2    98  6505000  58700      746       6.6 2000.083
3 Abilene 2000     3   130  9285000  58100      784       6.8 2000.167
4 Abilene 2000     4    98  9730000  68600      785       6.9 2000.250
5 Abilene 2000     5   141 10590000  67300      794       6.8 2000.333
6 Abilene 2000     6   156 13910000  66900      780       6.6 2000.417

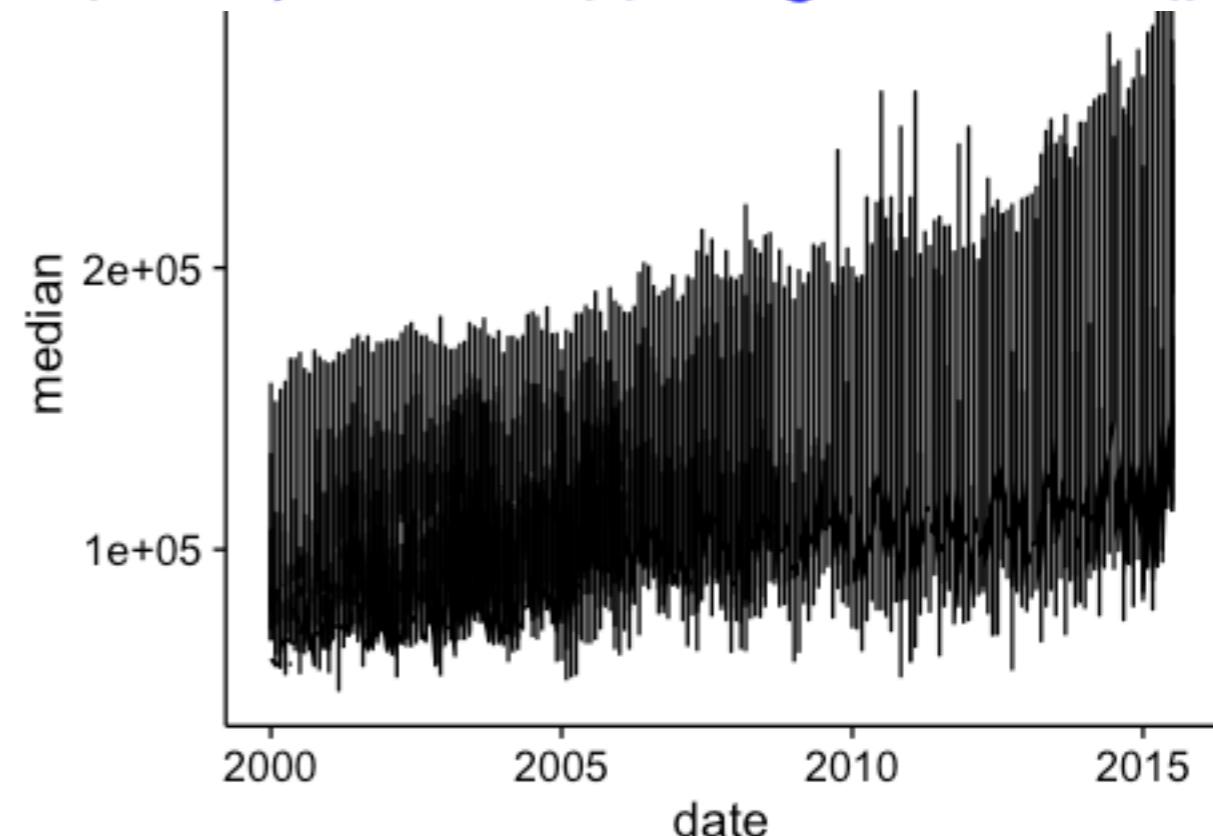
ggplot(txhousing, aes(date, median)) + geom_line()
```

# Second example

```
> head(txhousing)
```

	city	year	month	sales	volume	median	listings	inventory	date
1	Abilene	2000	1	72	5380000	71400	701	6.3	2000.000
2	Abilene	2000	2	98	6505000	58700	746	6.6	2000.083
3	Abilene	2000	3	130	9285000	58100	784	6.8	2000.167
4	Abilene	2000	4	98	9730000	68600	785	6.9	2000.250
5	Abilene	2000	5	141	10590000	67300	794	6.8	2000.333
6	Abilene	2000	6	156	13910000	66900	780	6.6	2000.417

```
ggplot(txhousing, aes(date, median)) + geom_line()
```

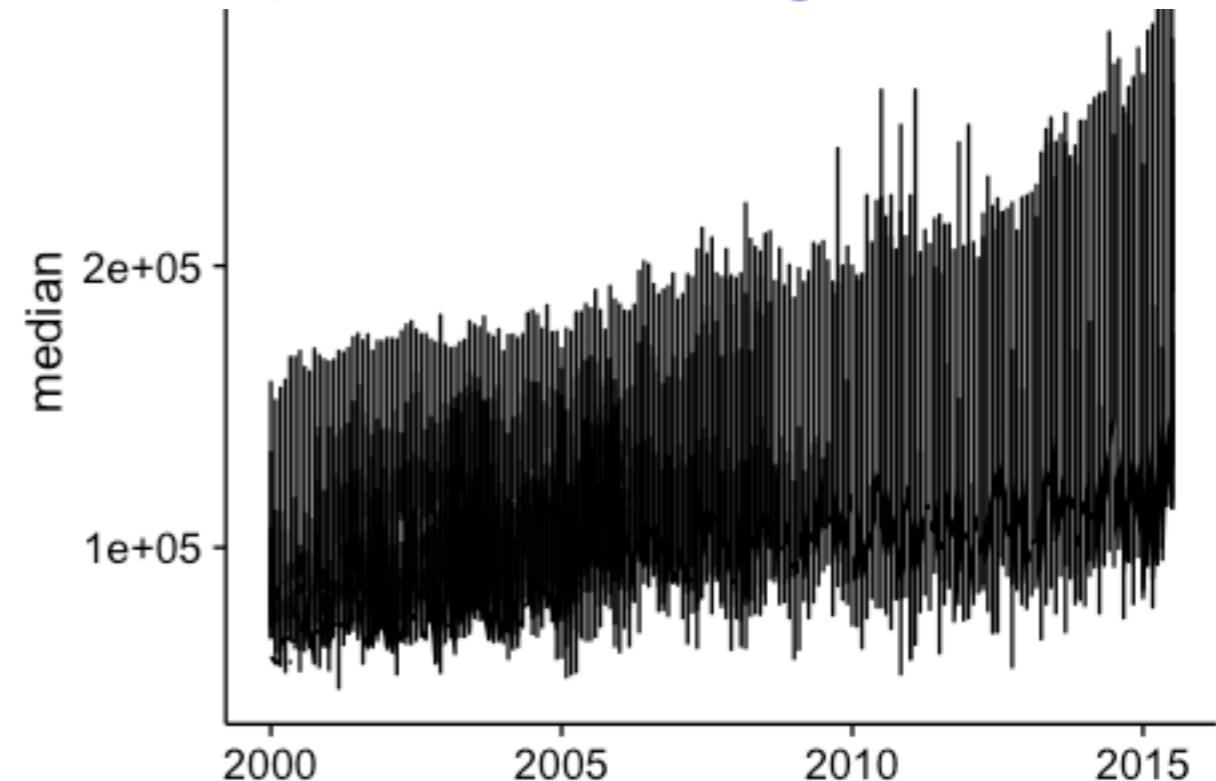
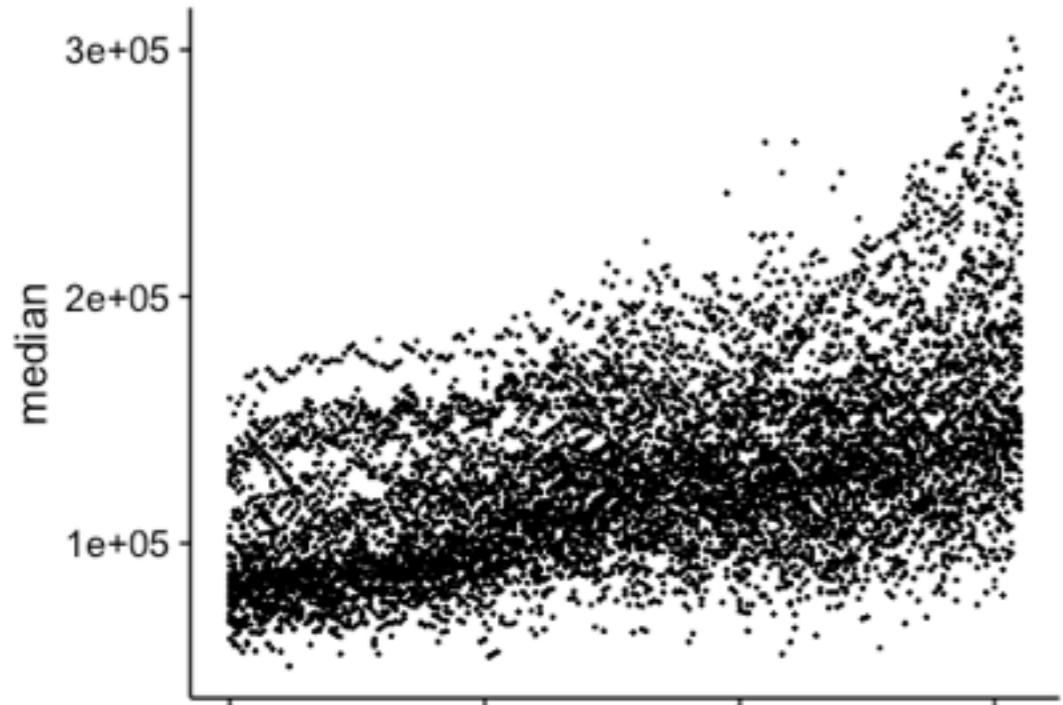


# Second example

```
> head(txhousing)
```

	city	year	month	sales	volume	median	listings	inventory	date
1	Abilene	2000	1	72	5380000	71400	701	6.3	2000.000
2	Abilene	2000	2	98	6505000	58700	746	6.6	2000.083
3	Abilene	2000	3	130	9285000	58100	784	6.8	2000.167
4	Abilene	2000	4	98	9730000	68600	785	6.9	2000.250
5	Abilene	2000	5	141	10590000	67300	794	6.8	2000.333
6	Abilene	2000	6	156	13910000	66900	780	6.6	2000.417

```
ggplot(txhousing, aes(date, median)) + geom_line()
```



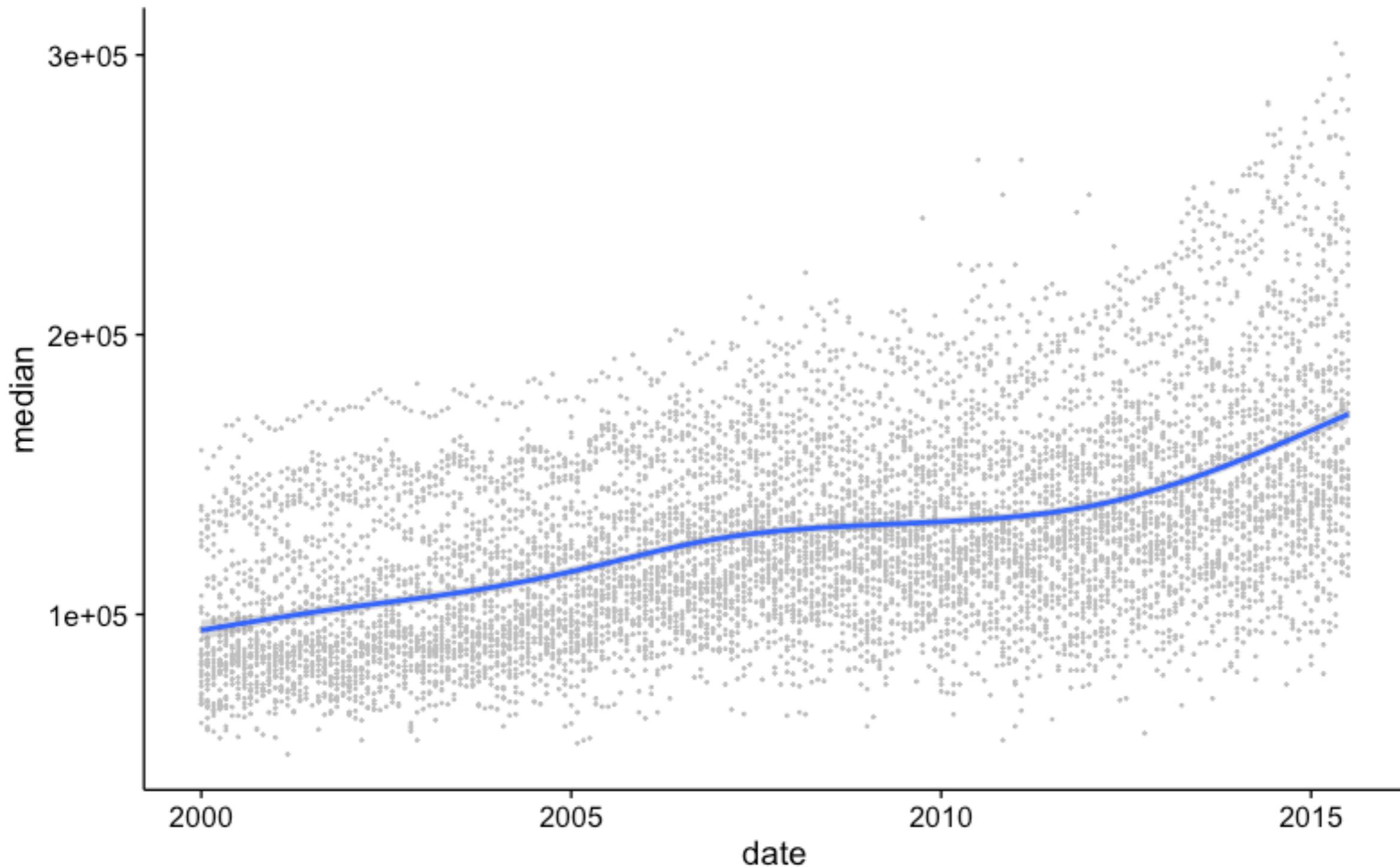
```
ggplot(txhousing, aes(date, median)) + geom_point(size=.2)
```

```
ggplot(txhousing, aes(date, median, color = city)) + geom_line()
```



What if we want to understand all Texas trends?

```
ggplot(txhousing, aes(date, median)) +  
  geom_point(size=.2, color="grey") + stat_smooth()
```



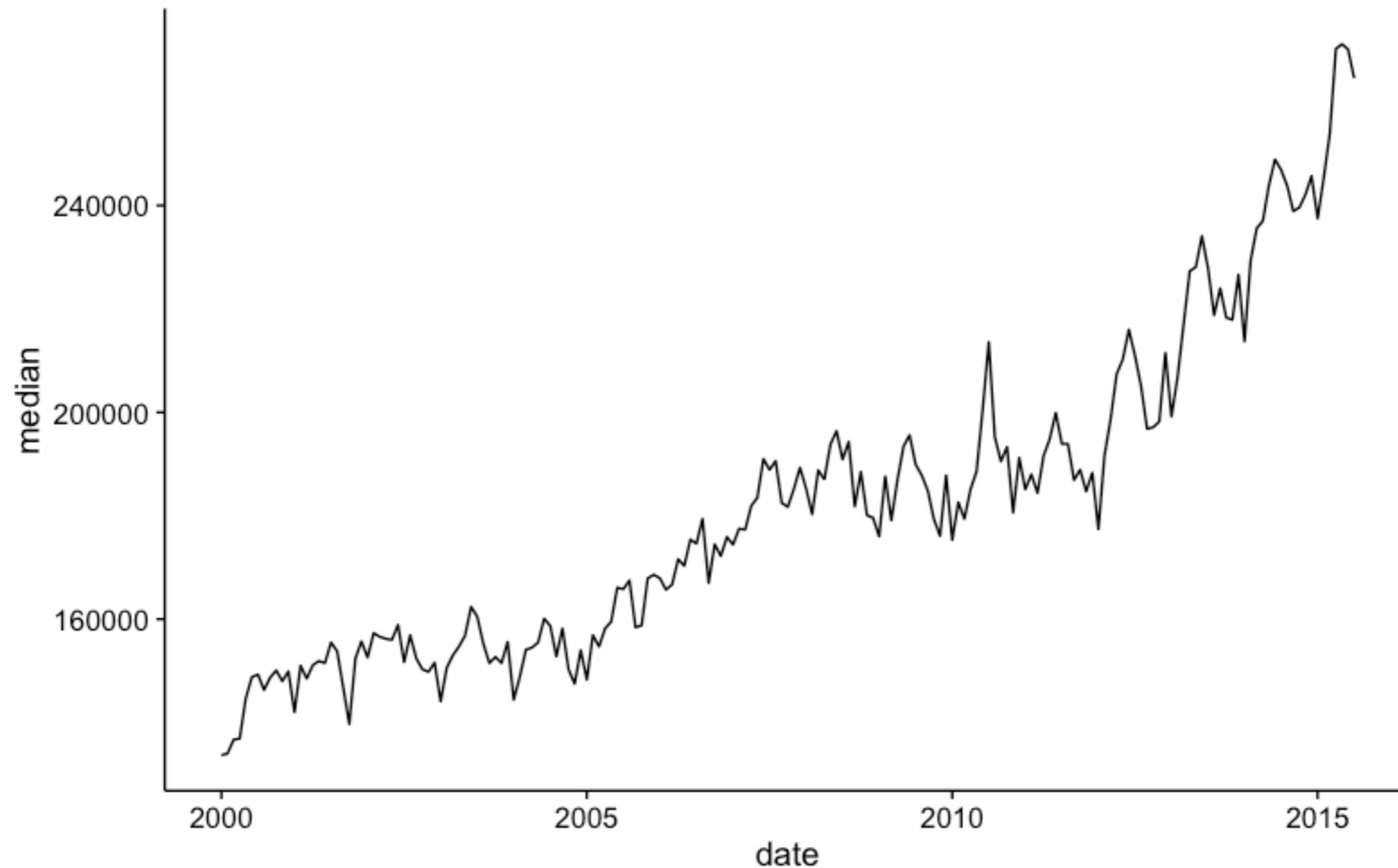
What if instead we're only interested in Austin?

# What if instead we're only interested in Austin?

```
housing = txhousing[which(txhousing$city=="Austin"), ]  
ggplot(housing, aes(date, median)) + geom_line()
```

# What if instead we're only interested in Austin?

```
housing = txhousing[which(txhousing$city=="Austin"), ]  
ggplot(housing, aes(date, median)) + geom_line()
```



# 3rd Programming Exercise

1. open up 3\_ggplot2.R in Rstudio
2. Start playing with code
3. get addicted to ggplot

# R resources

- stack overflow (google)
- Hadley Wickham's website - <http://hadley.nz/>
- <http://www.r-bloggers.com/how-to-learn-r-2/>
- A Beginner's Guide to R (Use R!) by Alain Zuur, Elena N. Ieno, and Erik Misters
- The Art of R Programming: A Tour of Statistical Software Design by Norman Matloff
- ggplot2: Elegant Graphics for Data Analysis (Use R!) by Hadley Wickham. — Maybe wait for the second edition (it's slightly outdated)