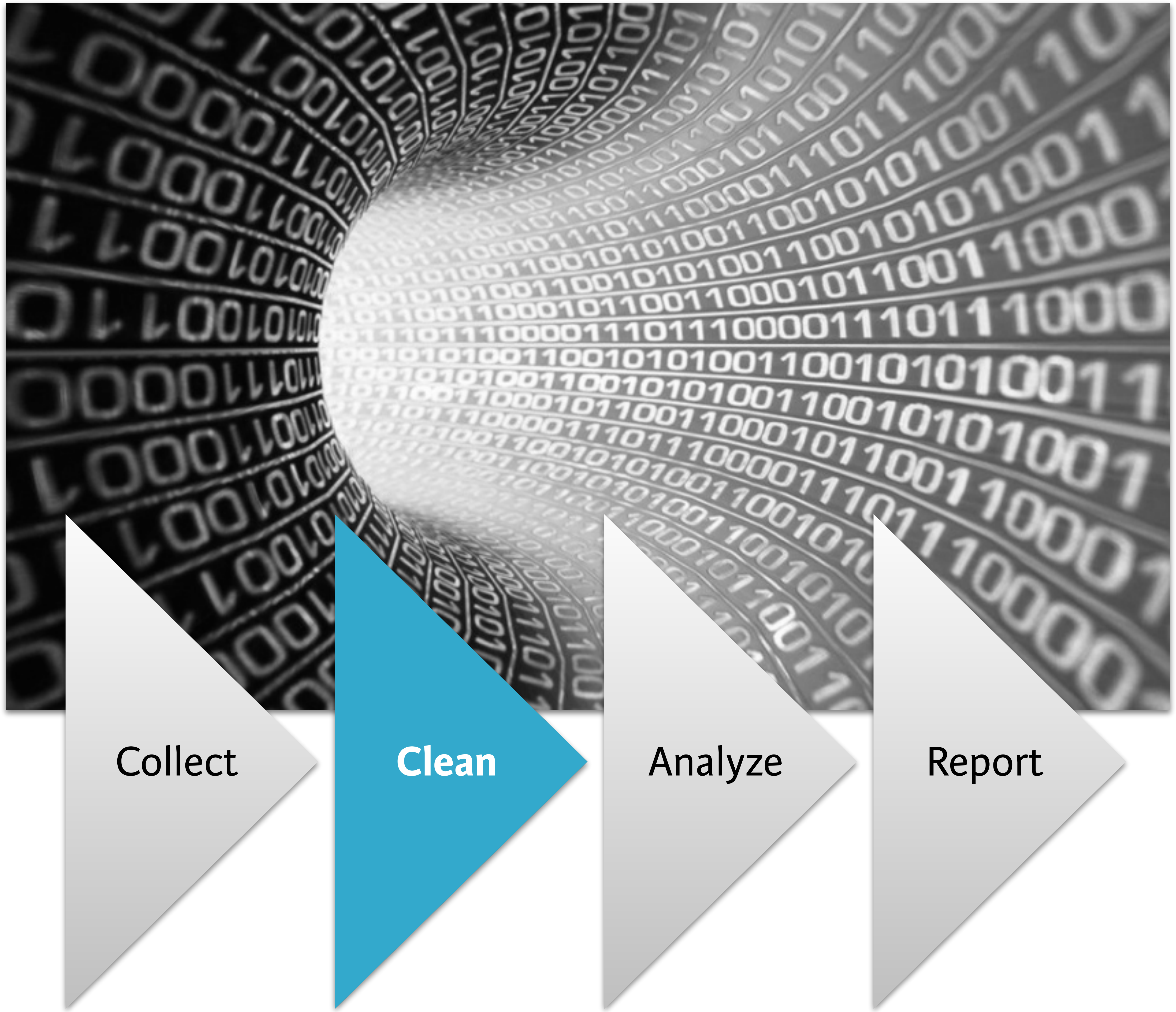
Introduction to Cleaning Data in R

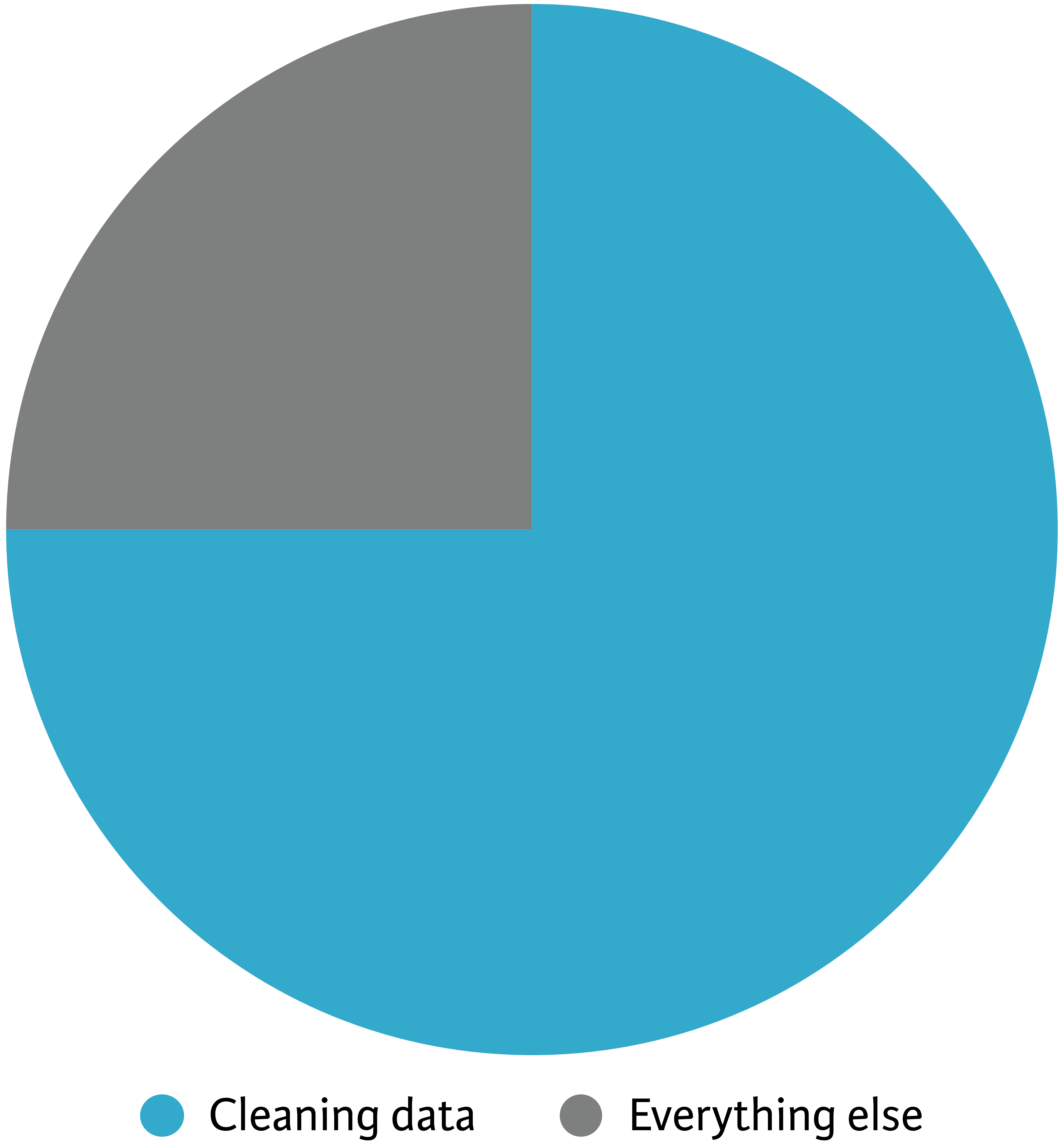
A look at some dirty data

```
> head(weather)
  X year month           measure X1 X2 X3 X4 X5 X6 X7 X8 X9 ...
1 1 2014    12  Max.TemperatureF 64 42 51 43 42 45 38 29 49 ...
2 2 2014    12 Mean.TemperatureF 52 38 44 37 34 42 30 24 39 ...
3 3 2014    12  Min.TemperatureF 39 33 37 30 26 38 21 18 29 ...
4 4 2014    12    Max.Dew.PointF 46 40 49 24 37 45 36 28 49 ...
5 5 2014    12    MeanDew.PointF 40 27 42 21 25 40 20 16 41 ...
6 6 2014    12     Min.DewpointF 26 17 24 13 12 36 -3  3 28 ...

> tail(weather)
      X year month           measure   X1   X2   X3   X4 ...
281 281 2015    12 Mean.Wind.SpeedMPH    6 <NA> <NA> <NA> ...
282 282 2015    12  Max.Gust.SpeedMPH   17 <NA> <NA> <NA> ...
283 283 2015    12    PrecipitationIn 0.14 <NA> <NA> <NA> ...
284 284 2015    12         CloudCover    7 <NA> <NA> <NA> ...
285 285 2015    12             Events Rain <NA> <NA> <NA> ...
286 286 2015    12     WindDirDegrees  109 <NA> <NA> <NA> ...
```

Why care about cleaning data?





What we'll cover in this course

1. Exploring raw data
2. Tidying data
3. Preparing data for analysis
4. Putting it all together

Cleaning data process

Let's practice!

Exploring raw data

Exploring raw data

- Understand the structure of your data
- Look at your data
- Visualize your data

Understanding the structure of your data

```
# Load the lunch data
> lunch <- read.csv("datasets/lunch_clean.csv")

# View its class
> class(lunch)
[1] "data.frame"

# View its dimensions
> dim(lunch)
[1] 46  7
```

Rows Columns

```
# Look at column names
> names(lunch)
[1] "year"          "avg_free"      "avg_reduced"   "avg_full"
[5] "avg_total"     "total_served"  "perc_free_red"
```

Understanding the structure of your data

```
> str(lunch)
'data.frame':   46 obs. of  7 variables:
 $ year         : int  1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 ...
 $ avg_free     : num  2.9 4.6 5.8 7.3 8.1 8.6 9.4 10.2 10.5 10.3 ...
 $ avg_reduced  : num  0 0 0.5 0.5 0.5 0.5 0.6 0.8 1.3 1.5 ...
 $ avg_full     : num  16.5 17.8 17.8 16.6 16.1 15.5 14.9 14.6 14.5 14.9 ...
 $ avg_total    : num  19.4 22.4 24.1 24.4 24.7 24.6 24.9 25.6 26.2 26.7 ...
 $ total_served : num  3368 3565 3848 3972 4009 ...
 $ perc_free_red: num  15.1 20.7 26.1 32.4 35 37.1 40.3 43.1 44.8 44.4 ...
```

Understanding the structure of your data

```
# Load dplyr
> library(dplyr)

# View structure of lunch, the dplyr way
> glimpse(lunch)
Observations: 46
Variables: 7
$ year          (int) 1969, 1970, 1971, 1972, 1973, 1974...
$ avg_free      (dbl) 2.9, 4.6, 5.8, 7.3, 8.1, 8.6, 9.4,...
$ avg_reduced   (dbl) 0.0, 0.0, 0.5, 0.5, 0.5, 0.5, 0.6,...
$ avg_full      (dbl) 16.5, 17.8, 17.8, 16.6, 16.1, 15.5...
$ avg_total     (dbl) 19.4, 22.4, 24.1, 24.4, 24.7, 24.6...
$ total_served  (dbl) 3368, 3565, 3848, 3972, 4009, 3982...
$ perc_free_red (dbl) 15.1, 20.7, 26.1, 32.4, 35.0, 37.1...
```

Understanding the structure of your data

```
# View a summary
> summary(lunch)
      year          avg_free      avg_reduced  
 Min.   :1969   Min.   : 2.90   Min.   :0.00  
 1st Qu.:1980   1st Qu.: 9.93   1st Qu.:1.52  
 Median :1992   Median :10.90   Median :1.80  
 Mean   :1992   Mean   :11.81   Mean   :1.86  
 3rd Qu.:2003   3rd Qu.:13.60   3rd Qu.:2.60  
 Max.   :2014   Max.   :19.20   Max.   :3.20  
    avg_full      avg_total     total_served  perc_free_red
 Min.   : 8.8   Min.   :19.4   Min.   :3368   Min.   :15.1  
 1st Qu.:11.4   1st Qu.:24.2   1st Qu.:4006   1st Qu.:45.6  
 Median :12.2   Median :25.9   Median :4252   Median :52.4  
 Mean   :12.8   Mean   :26.4   Mean   :4367   Mean   :51.1  
 3rd Qu.:14.2   3rd Qu.:28.3   3rd Qu.:4751   3rd Qu.:58.3  
 Max.   :17.8   Max.   :31.8   Max.   :5278   Max.   :71.6  
```

Understanding the structure of your data

- `class()` - Class of data object
- `dim()` - Dimensions of data
- `names()` - Column names
- `str()` - Preview of data with helpful details
- `glimpse()` - Better version of `str()` from dplyr
- `summary()` - Summary of data

Let's practice!

Exploring raw data

Looking at your data

```
# View the top
> head(lunch)
  year avg_free avg_reduced avg_full avg_total total_served
1 1969      2.9         0.0     16.5      19.4         3368
2 1970      4.6         0.0     17.8      22.4         3565
3 1971      5.8         0.5     17.8      24.1         3848
4 1972      7.3         0.5     16.6      24.4         3972
5 1973      8.1         0.5     16.1      24.7         4009
6 1974      8.6         0.5     15.5      24.6         3982
  perc_free_red
1          15.1
2          20.7
3          26.1
4          32.4
5          35.0
6          37.1
```

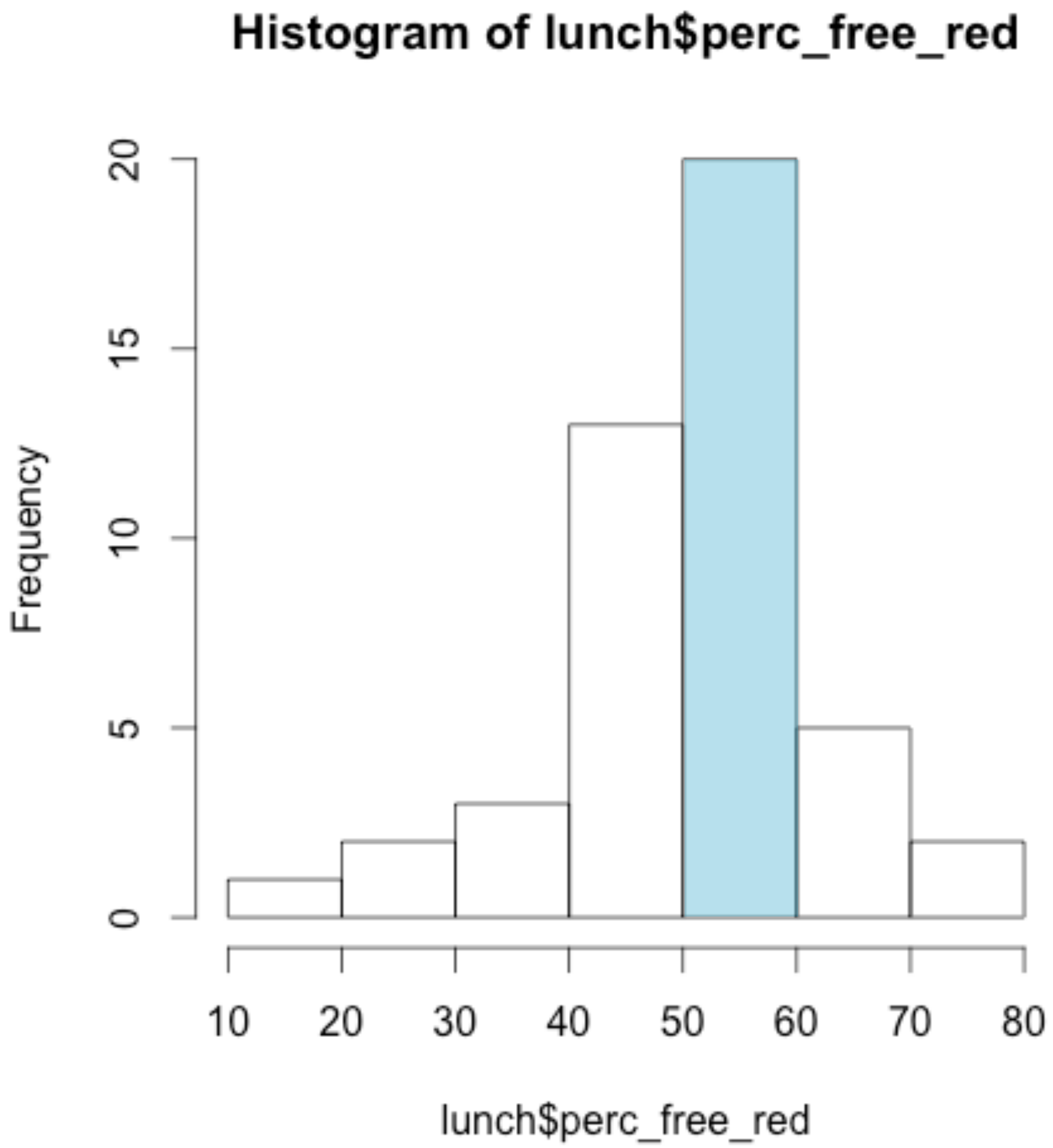
head(lunch, n = 15)

Looking at your data

```
# View the bottom
> tail(lunch)
   year avg_free avg_reduced avg_full avg_total total_served
41 2009     16.3         3.2     11.9      31.3         5186
42 2010     17.6         3.0     11.1      31.8         5278
43 2011     18.4         2.7     10.8      31.8         5274
44 2012     18.7         2.7     10.2      31.7         5215
45 2013     18.9         2.6      9.2      30.7         5098
46 2014     19.2         2.5      8.8      30.5         5020
   perc_free_red
41          62.6
42          65.3
43          66.6
44          68.2
45          70.5
46          71.6
```

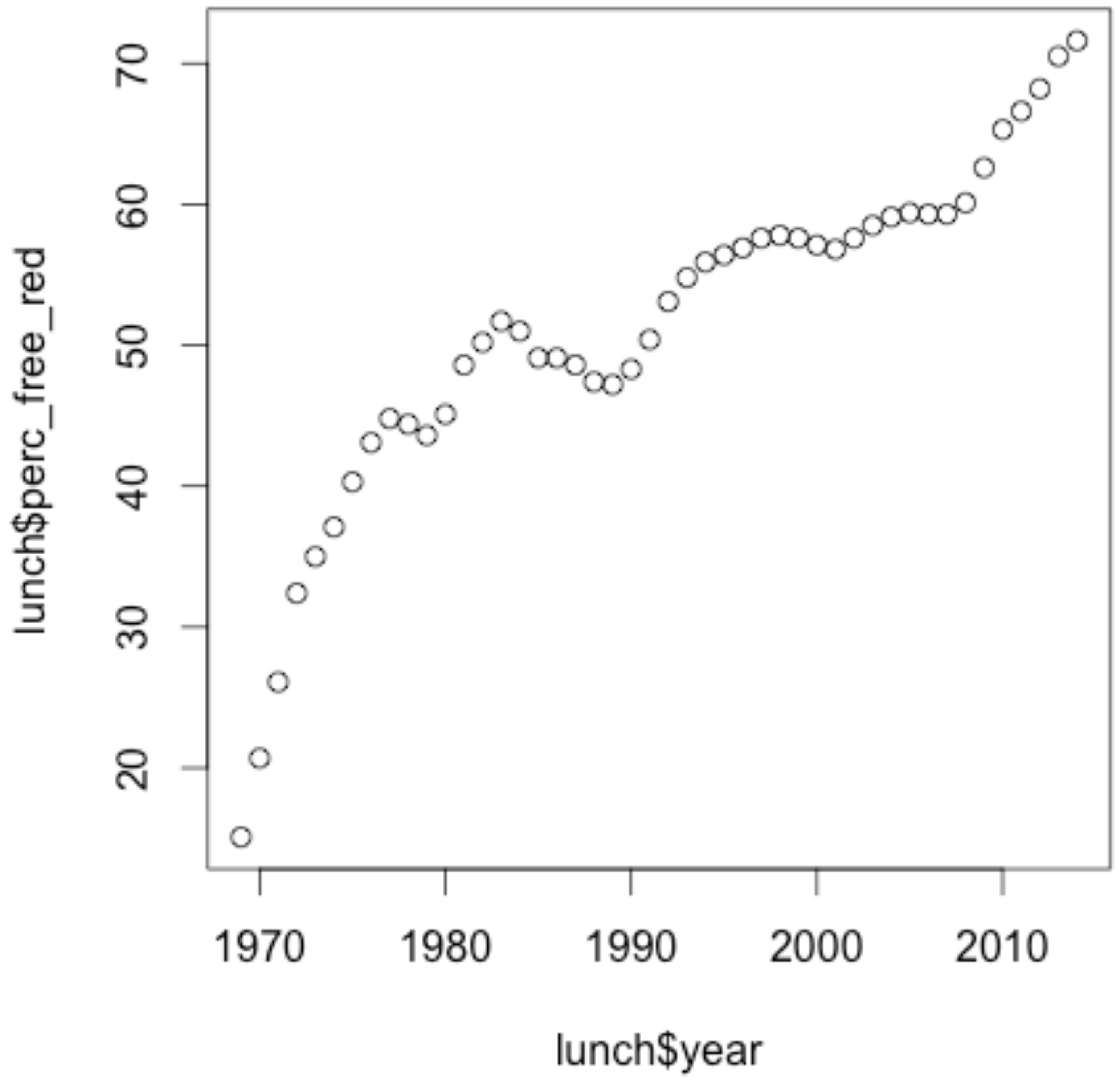
Visualizing your data

```
# View histogram
> hist(lunch$perc_free_red)
```



Visualizing your data

```
# View plot of two variables
> plot(lunch$year, lunch$perc_free_red)
```



Looking at your data

- `head()` - View top of dataset
- `tail()` - View bottom of dataset
- `print()` - View entire dataset (not recommended!)

Visualizing your data

- `hist()` - View histogram of a single variable
- `plot()` - View plot of two variables

Let's practice!

CLEANING DATA IN R

Introduction to tidy data

Principles of tidy data

name	age	eye_color	height
Jake	34	Other	6'1"
Alice	55	Blue	5'9"
Tim	76	Brown	5'7"
Denise	19	Other	5'1"

Observation

Variable or Attribute

- Observations as rows
- Variables as columns
- One type of observational unit per table

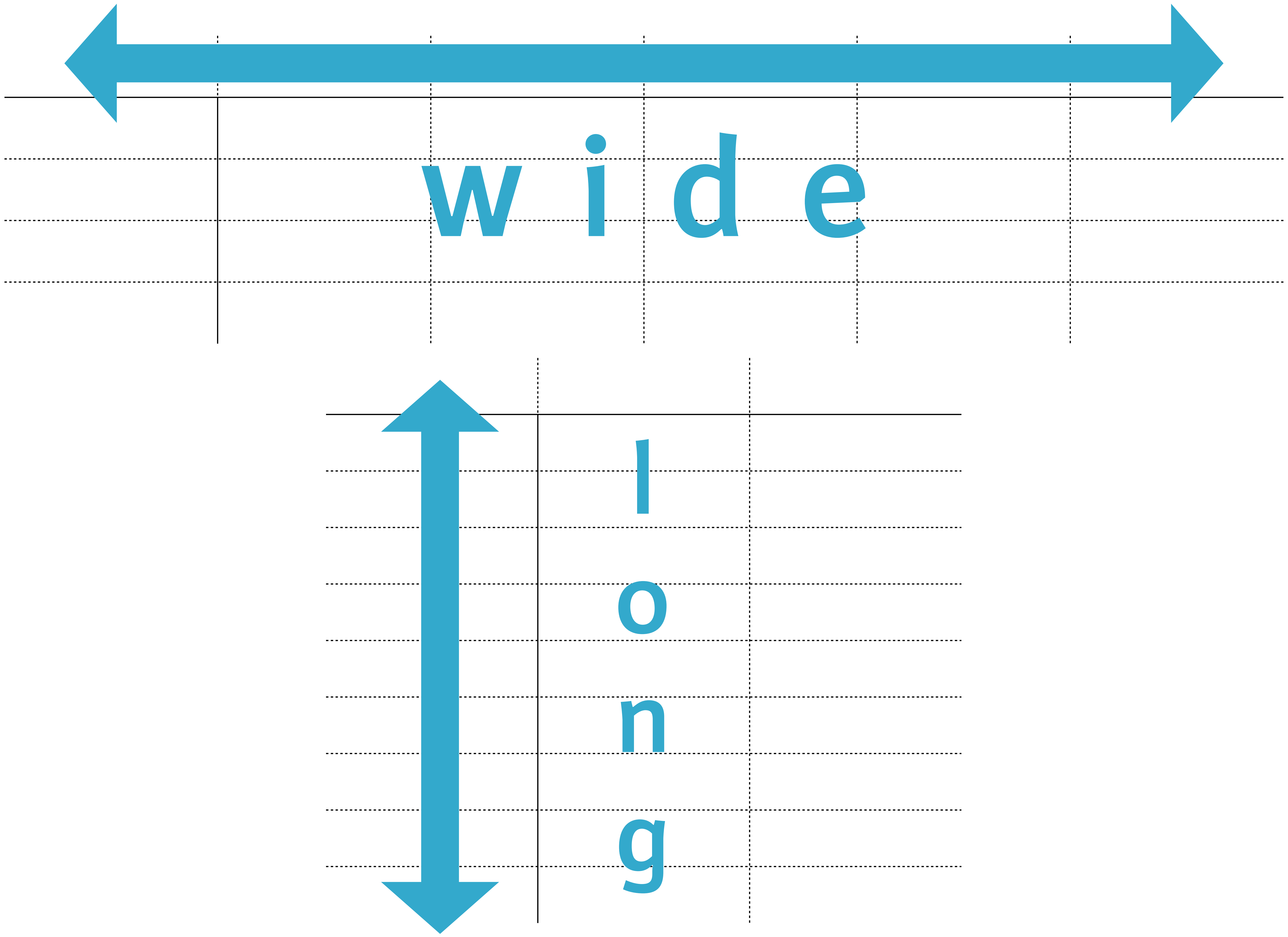
A dirty data diagnosis

name	age	brown	blue	other	height
Jake	34	0	0	1	6'1"
Alice	55	0	1	0	5'9"
Tim	76	1	0	0	5'7"
Denise	19	0	0	1	5'1"

?

Column headers are values, not variable names

Wide vs. long datasets



Let's practice!

Introduction to tidyr

Overview of tidyr

- R package by Hadley Wickham
- Apply the principles of tidy data
- Small set of simple functions

Gather columns into key-value pairs

```
# Look at wide_df
> wide_df
  col A B C
1   X 1 2 3
2   Y 4 5 6

# Gather the columns of wide_df
> gather(wide_df, my_key, my_val, -col)
  col my_key my_val
1   X      A      1
2   Y      A      4
3   X      B      2
4   Y      B      5
5   X      C      3
6   Y      C      6
```

`gather(data, key, value, …)`

`data`: a data frame

`key`: bare name of new key column

`value`: bare name of new value column

`…`: bare names of columns to gather (or not)

Spread key-value pairs into columns

```
# Look at long_df
> long_df
  col my_key my_val
1   X      A      1
2   Y      A      4
3   X      B      2
4   Y      B      5
5   X      C      3
6   Y      C      6

# Spread the key-value pairs of long_df
> spread(long_df, my_key, my_val)
  col A B C
1   X 1 2 3
2   Y 4 5 6
```

`spread(data, key, value)`

`data`: a data frame

`key`: bare name of column containing keys

`value`: bare name of column containing values

Let's practice!

Introduction to tidyr

Separate columns

```
# View the treatments data
> treatments
  patient treatment year_mo response
1       X         A 2010-10        1
2       Y         A 2010-10        4
3       X         B 2012-08        2
4       Y         B 2012-08        5
5       X         C 2014-12        3
6       Y         C 2014-12        6

# Separate year_mo into two columns
> separate(treatments, year_mo, c("year", "month"))
  patient treatment year month response
1       X         A 2010    10        1
2       Y         A 2010    10        4
3       X         B 2012    08        2
4       Y         B 2012    08        5
5       X         C 2014    12        3
6       Y         C 2014    12        6
```

`separate(data, col, into)`

`data`: a data frame

sep = "-"

`col`: bare name of column to separate

`into`: character vector of new column names

Unite columns

```
# View treatments data
> treatments
  patient treatment year month response
1       X         A 2010    10        1
2       Y         A 2010    10        4
3       X         B 2012    08        2
4       Y         B 2012    08        5
5       X         C 2014    12        3
6       Y         C 2014    12        6

# Unite year and month to form year_mo column
> unite(treatments, year_mo, year, month)
  patient treatment year_mo response
1       X         A 2010_10        1
2       Y         A 2010_10        4
3       X         B 2012_08        2
4       Y         B 2012_08        5
5       X         C 2014_12        3
6       Y         C 2014_12        6
```

`unite(data, col, ...)`

`data`: a data frame

sep = "-"

`col`: bare name of new column

`...`: bare names of columns to unite

Summary of key tidyr functions

- `gather()` - Gather columns into key-value pairs
- `spread()` - Spread key-value pairs into columns
- `separate()` - Separate one column into multiple
- `unite()` - Unite multiple columns into one

Let's practice!

Common symptoms of messy data

Column headers are values, not variable names

name	age	brown	blue	other	height
Jake	34	0	0	1	6'1"
Alice	55	0	1	0	5'9"
Tim	76	1	0	0	5'7"
Denise	19	0	0	1	5'1"

name	age	eye_color	height
Jake	34	Other	6'1"
Alice	55	Blue	5'9"
Tim	76	Brown	5'7"
Denise	19	Other	5'1"

Variables are stored in both rows and columns

name	measurement	value
Jake	n_dogs	1
Jake	n_cats	0
Jake	n_birds	1
Alice	n_dogs	1
Alice	n_cats	2
Alice	n_birds	0

name	n_dogs	n_cats	n_birds
Jake	1	0	1
Alice	1	2	0

Multiple variables are stored in one column

name	sex_age	eye_color	height
Jake	M.34	Other	6'1"
Alice	F.55	Blue	5'9"
Tim	M.76	Brown	5'7"
Denise	F.19	Other	5'1"

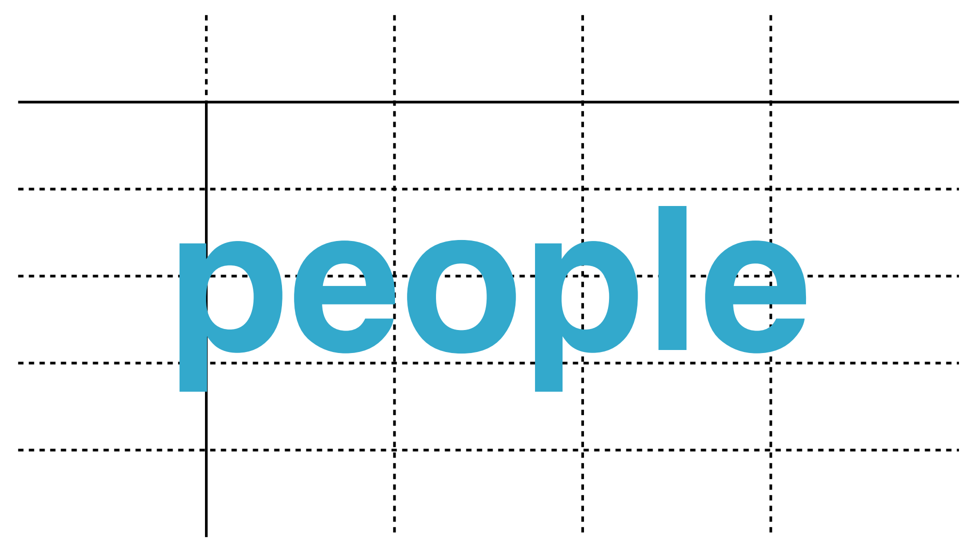
name	sex	age	eye_color	height
Jake	M	34	Other	6'1"
Alice	F	55	Blue	5'9"
Tim	M	76	Brown	5'7"
Denise	F	19	Other	5'1"

Other common symptoms

- A single observational unit is stored in multiple tables
- Multiple types of observational units are stored in the same table

name	age	height	pet_name	pet_type	pet_height
Jake	34	6'1"	Larry	Dog	25"
Jake	34	6'1"	Chirp	Bird	3"
Alice	55	5'9"	Wally	Dog	30"
Alice	55	5'9"	Sugar	Cat	10"
Alice	55	5'9"	Spice	Cat	12"

Alice's name, age, and height are duplicated 3x



Let's practice!

Type conversions

Types of variables in R

- `character: "treatment", "123", "A"`
- `numeric: 23.44, 120, NaN, Inf`
- `integer: 4L, 1123L`
- `factor: factor("Hello"), factor(8)`
- `logical: TRUE, FALSE, NA`

Types of variables in R

```
> class("hello")
[1] "character"

> class(3.844)
[1] "numeric"

> class(77L)
[1] "integer"

> class(factor("yes"))
[1] "factor"

> class(TRUE)
[1] "logical"
```

Type conversions

```
> as.character(2016)
[1] "2016"

> as.numeric(TRUE)
[1] 1

> as.integer(99)
[1] 99

> as.factor("something")
[1] something
Levels: something

> as.logical(0)
[1] FALSE
```

Overview of lubridate

- Written by Garrett Grolemund & Hadley Wickham
- Coerce strings to dates

Dates with lubridate

```
# Load the lubridate package
> library(lubridate)

# Experiment with basic lubridate functions
> ymd("2015-08-25")
[1] "2015-08-25 UTC"
```
year-month-day

```
> ymd("2015 August 25")
[1] "2015-08-25 UTC"
```
year-month-day

```
> mdy("August 25, 2015")
[1] "2015-08-25 UTC"
```
month-day-year

```
> hms("13:33:09")
[1] "13H 33M 9S"
```
hour-minute-second

```
> ymd_hms("2015/08/25 13.33.09")
[1] "2015-08-25 13:33:09 UTC"
```
year-month-day hour-minute-second

Let's practice!

String manipulation

Overview of stringr

- R package written by Hadley Wickham
- Suite of helpful functions for working with strings
- Functions share consistent interface

Key functions in stringr for cleaning data

```
# Trim leading and trailing white space
> str_trim("  this is a test     ")
[1] "this is a test"    white space removed

# Pad string with zeros
> str_pad("24493", width = 7, side = "left", pad = "0")
[1] "0024493"  7 digits

# Create character vector of names
> friends <- c("Sarah", "Tom", "Alice")

# Search for string in vector
> str_detect(friends, "Alice")
[1] FALSE FALSE  TRUE

# Replace string in vector
> str_replace(friends, "Alice", "David")
[1] "Sarah" "Tom"   "David"
```

Key functions in stringr for cleaning data

- `str_trim()` - Trim leading and trailing white space
- `str_pad()` - Pad with additional characters
- `str_detect()` - Detect a pattern
- `str_replace()` - Find and replace a pattern

Other helpful functions in base R

- `tolower()` - Make all lowercase
- `toupper()` - Make all uppercase

```
# Make all lowercase
> tolower("I AM TALKING LOUDLY!!")
[1] "i am talking loudly!!"

# Make all uppercase
> toupper("I am whispering...")
[1] "I AM WHISPERING..."
```

Let's practice!

Missing and special values

Missing values

- May be random, but dangerous to assume
- Sometimes associated with variable/outcome of interest
- In R, represented as NA
- May appear in other forms
    - #N/A (Excel)
    - Single dot (SPSS, SAS)
    - Empty string

Special values

- `Inf` - "Infinite value" (indicative of outliers?)
  - `1/0`
  - `1/0 + 1/0`
  - `33333^33333`
- `NaN` - "Not a number" (rethink a variable?)
  - `0/0`
  - `1/0 - 1/0`

Finding missing values

```
# Create small dataset
> df <- data.frame(A = c(1, NA, 8, NA),
                   B = c(3, NA, 88, 23),
                   C = c(2, 45, 3, 1))
# Check for NAs
> is.na(df)
         A     B     C
[1,] FALSE FALSE FALSE
[2,]  TRUE  TRUE FALSE
[3,] FALSE FALSE FALSE
[4,]  TRUE FALSE FALSE

# Are there any NAs?
> any(is.na(df))
[1] TRUE

# Count number of NAs
> sum(is.na(df))
[1] 3
```

4 rows, 3 columns

Same size: 4 rows, 3 columns

Finding missing values

```
# Use summary() to find NAs
> summary(df)
       A               B               C
 Min.   :1.00   Min.   : 3.0   Min.   : 1.00
 1st Qu.:2.75   1st Qu.:13.0   1st Qu.: 1.75
 Median :4.50   Median :23.0   Median : 2.50
 Mean   :4.50   Mean   :38.0   Mean   :12.75
 3rd Qu.:6.25   3rd Qu.:55.5   3rd Qu.:13.50
 Max.   :8.00   Max.   :88.0   Max.   :45.00
 NA's   :2      NA's   :1
```

Dealing with missing values

```
# Find rows with no missing values
> complete.cases(df)
[1]  TRUE FALSE  TRUE FALSE

# Subset data, keeping only complete cases
> df[complete.cases(df), ]
  A  B C
1 1  3 2
3 8 88 3

# Another way to remove rows with NAs
> na.omit(df)
  A  B C
1 1  3 2
3 8 88 3
```

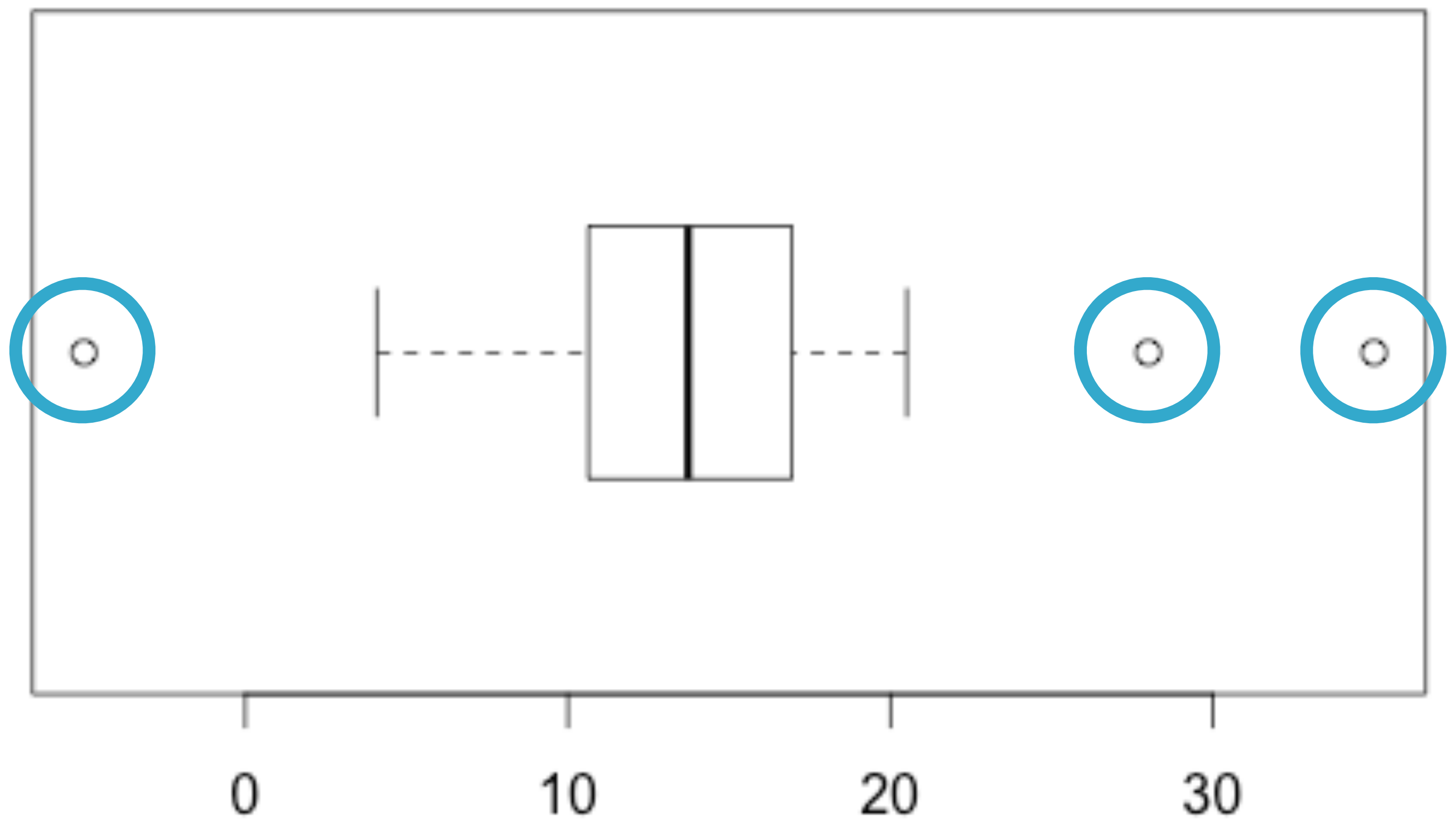
Let's practice!

Outliers and obvious errors

Outliers

```
# Simulate some data
> set.seed(10)
> x <- c(rnorm(30, mean = 15, sd = 5), -5, 28, 35)

# View a boxplot
> boxplot(x, horizontal = TRUE)
```



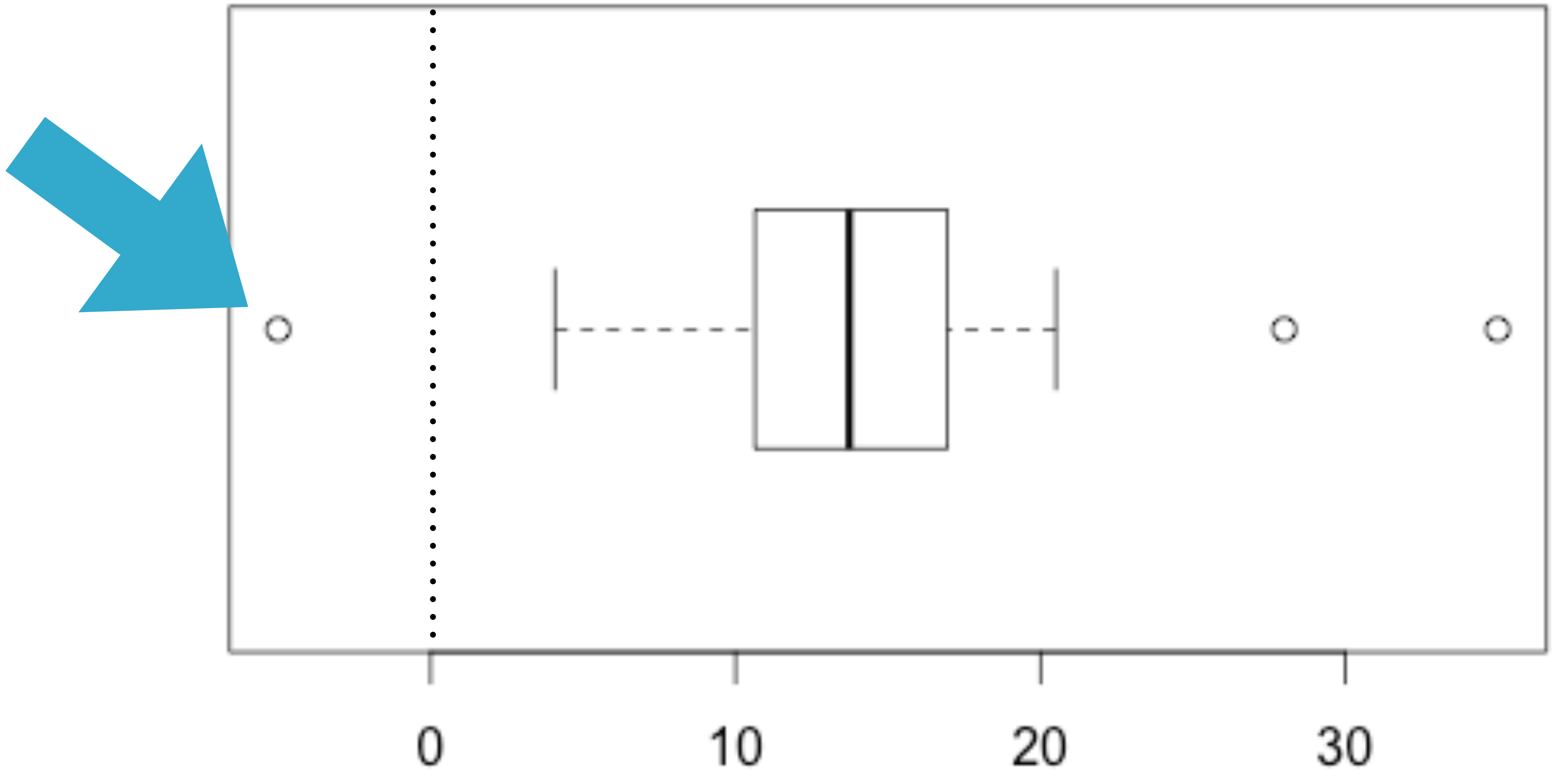
Outliers

Outliers

- Extreme values distant from other values
- Several causes
  - Valid measurements
  - Variability in measurement
  - Experimental error
  - Data entry error
- May be discarded or retained depending on cause

Obvious errors

What if these values are supposed to represent ages?



Obvious errors

- May appear in many forms
  - Values so extreme they can't be plausible (e.g. person aged 243)
  - Values that don't make sense (e.g. negative age)
- Several causes
  - Measurement error
  - Data entry error
  - Special code for missing data (e.g. -1 means missing)
- Should generally be removed or replaced

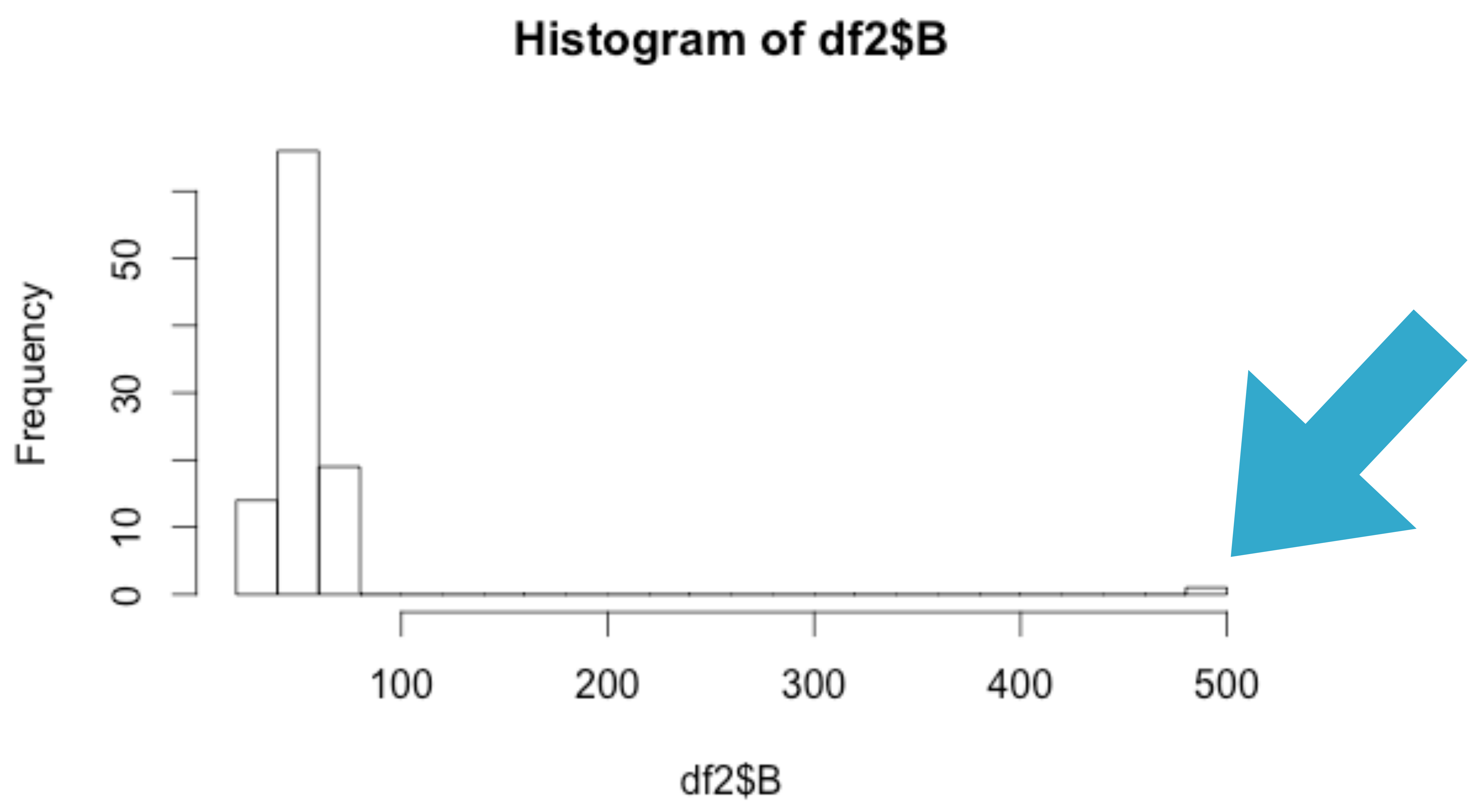
Finding outliers and errors

```
# Create another small dataset
> df2 <- data.frame(A = rnorm(100, 50, 10),
                    B = c(rnorm(99, 50, 10), 500),
                    C = c(rnorm(99, 50, 10), -1))

# View a summary
> summary(df2)
       A               B                C
 Min.   :23.7   Min.   : 26.9   Min.   :-1.0
 1st Qu.:43.7   1st Qu.: 43.7   1st Qu.:40.3
 Median :51.9   Median : 49.8   Median :48.5
 Mean   :50.4   Mean   : 54.9   Mean   :47.8
 3rd Qu.:56.9   3rd Qu.: 56.6   3rd Qu.:56.3
 Max.   :77.2   Max.   :500.0   Max.   :75.1
```

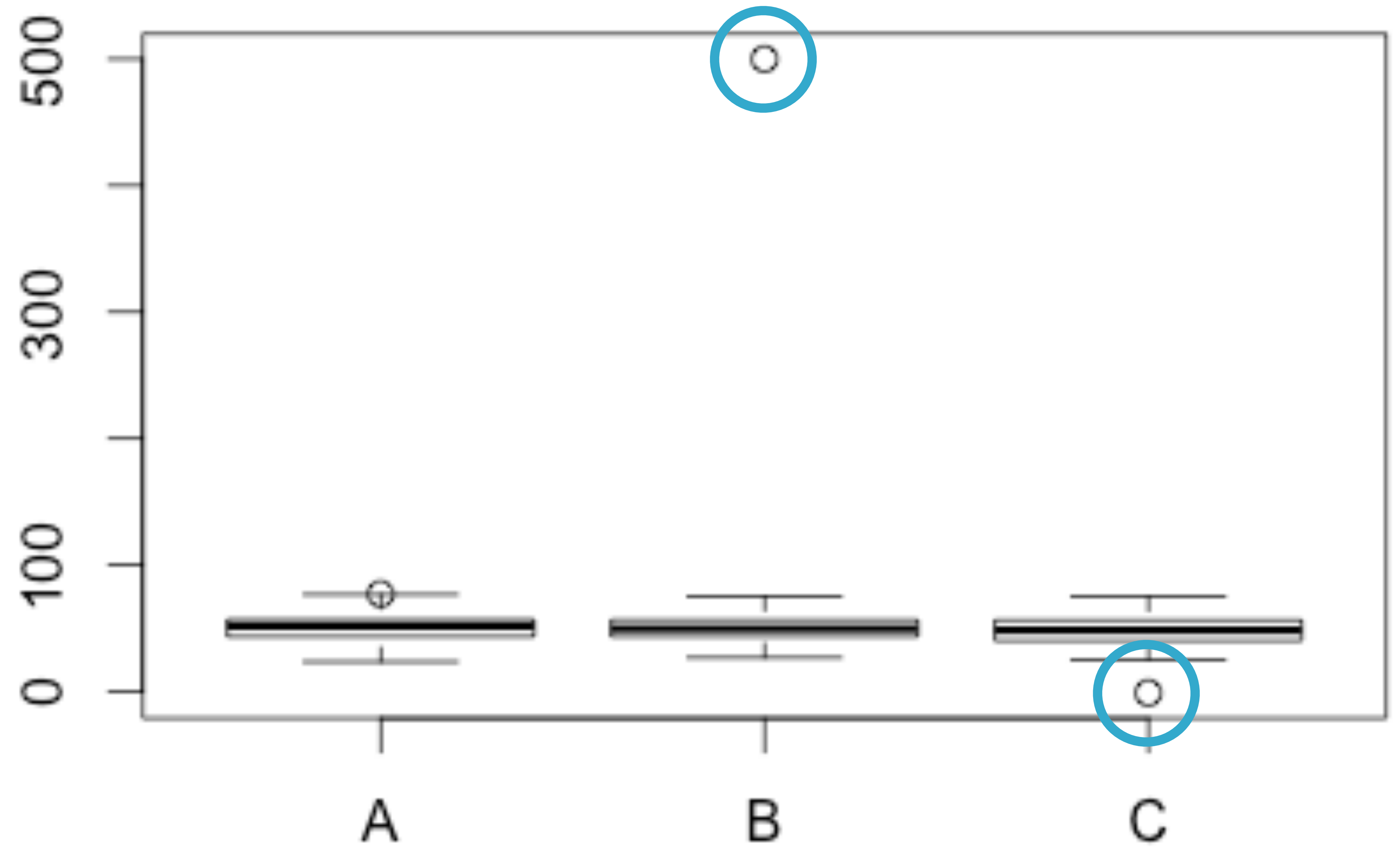
Finding outliers and errors

```
# View a histogram
> hist(df2$B, breaks = 20)
```



Finding outliers and errors

```
# View a boxplot
> boxplot(df2)
```



Let's practice!

Time to put it all together!

The challenge

- Historical weather data from Boston, USA
- 12 months beginning Dec 2014
- The data are dirty
  - Column names are values
  - Variables coded incorrectly
  - Missing and extreme values
  - ...
- **Clean the data!**

Understanding the structure of your data

- `class()` - Class of data object
- `dim()` - Dimensions of data
- `names()` - Column names
- `str()` - Preview of data with helpful details
- `glimpse()` - Better version of `str()` from dplyr
- `summary()` - Summary of data

Looking at your data

- `head()` - View top of dataset
- `tail()` - View bottom of dataset
- `print()` - View entire dataset (not recommended!)

Visualizing your data

- `hist()` - View histogram of a single variable
- `plot()` - View plot of two variables

Let's practice!

Let's tidy the data

Column names are values

```
> head(weather)
  X year month           measure X1 X2 X3 X4 X5 X6 X7 X8 X9 ...
1 1 2014    12  Max.TemperatureF 64 42 51 43 42 45 38 29 49 ...
2 2 2014    12 Mean.TemperatureF 52 38 44 37 34 42 30 24 39 ...
3 3 2014    12  Min.TemperatureF 39 33 37 30 26 38 21 18 29 ...
4 4 2014    12    Max.Dew.PointF 46 40 49 24 37 45 36 28 49 ...
5 5 2014    12    MeanDew.PointF 40 27 42 21 25 40 20 16 41 ...
6 6 2014    12     Min.DewpointF 26 17 24 13 12 36 -3  3 28 ...
```

Values are variable names

```
> head(weather2)
  X year month              measure day value
1 1 2014    12  Max.TemperatureF  X1    64
2 2 2014    12 Mean.TemperatureF  X1    52
3 3 2014    12  Min.TemperatureF  X1    39
4 4 2014    12    Max.Dew.PointF  X1    46
5 5 2014    12    MeanDew.PointF  X1    40
6 6 2014    12     Min.DewpointF  X1    26
```

Let's practice!

Prepare the data for analysis

Dates with lubridate

```
# Load the lubridate package
> library(lubridate)

# Experiment with basic lubridate functions
> ymd("2015-08-25")
[1] "2015-08-25 UTC"
```

year-month-day

```
> ymd("2015 August 25")
[1] "2015-08-25 UTC"
```

year-month-day

```
> mdy("August 25, 2015")
[1] "2015-08-25 UTC"
```

month-day-year

```
> hms("13:33:09")
[1] "13H 33M 9S"
```

hour-minute-second

```
> ymd_hms("2015/08/25 13.33.09")
[1] "2015-08-25 13:33:09 UTC"
```

year-month-day hour-minute-second

Type conversions

```
> as.character(2016)
[1] "2016"

> as.numeric(TRUE)
[1] 1

> as.integer(99)
[1] 99

> as.factor("something")
[1] something
Levels: something

> as.logical(0)
[1] FALSE
```

Let's practice!

Missing, extreme, and unexpected values

Finding missing values

```
# Create a small dataset
> x <- data.frame(a = c(2, 5, NA, 8),
                  b = c(NA, 34, 9, NA))

# Return data frame of TRUEs and FALSEs
> is.na(x)
         a     b
[1,] FALSE  TRUE
[2,] FALSE FALSE
[3,]  TRUE FALSE
[4,] FALSE  TRUE

# Count number of TRUEs
> sum(is.na(x))
[1] 3

# Find indices of missing values in column b
> which(is.na(x$b))
[1] 1 4
```

Identifying errors

- Context matters!
- Plausible ranges
- Numeric variables in weather data
  - Percentages (0-100)
  - Temperatures (Fahrenheit)
  - Wind speeds (miles per hour)
  - Pressures (inches of mercury)
  - Distances (miles)
  - Eighths (of cloud cover)

Let's practice!

Your data are clean!

Clean weather data

```
# View head of clean data
> head(weather6)
        date    events cloud_cover max_dew_point_f ...
1 2014-12-01      Rain           6              46 ...
2 2014-12-02 Rain-Snow           7              40 ...
3 2014-12-03      Rain           8              49 ...
4 2014-12-04      None           3              24 ...
5 2014-12-05      Rain           5              37 ...
6 2014-12-06      Rain           8              45 ...

# View tail of clean data
          date events cloud_cover max_dew_point_f ...
361 2015-11-26   None           6              49 ...
362 2015-11-27   None           7              52 ...
363 2015-11-28   Rain           8              50 ...
364 2015-11-29   None           4              33 ...
365 2015-11-30   None           6              26 ...
366 2015-12-01   Rain           7              43 ...
```

Summary of your accomplishments

- Inspected the data
- Tidied the data
- Improved date representations
- Dealt with incorrect variable codings
- Found and dealt with missing data
- Identified and corrected errors
- Visualized the result

Congratulations!