



Data
cleaning

Stat405

1. Intro to data cleaning

2. Missing values

3. Subsetting

4. Modifying

5. Short cuts

Clean data is:

Columnar

(rectangular, observations in rows, variables in columns)

Consistent

Concise

Complete

Correct

Correct

Can't restore correct values without original data but can remove clearly incorrect values

Options:

- Remove entire row

- Mark incorrect value as missing

What is a missing value?

In R, written as NA. Has special behaviour:

`NA + 3 = ?`

`NA > 2 = ?`

`mean(c(2, 7, 10, NA)) = ?`

`NA == NA ?`

Use `is.na()` to see if a value is NA

Many functions have `na.rm` argument

Your turn

Look at histograms and scatterplots of x, y, z from the diamonds dataset

Which values are clearly incorrect? Which values might we be able to correct?

(Remember measurements are in millimetres, 1 inch = 25 mm)

Plots

```
qplot(x, data = diamonds, binwidth = 0.1)
```

```
qplot(y, data = diamonds, binwidth = 0.1)
```

```
qplot(z, data = diamonds, binwidth = 0.1)
```

```
qplot(x, y, data = diamonds)
```

```
qplot(x, z, data = diamonds)
```

```
qplot(y, z, data = diamonds)
```

Modifying data

To modify, must first know how to extract, or **subset**. Many different methods available in R. We'll start with most explicit then learn some shortcuts.

Basic structure:

```
df$varname
```

```
df[row index, column index]
```

\$

Remember `str(diamonds)` ?

That hints at how to extract individual variables:

`diamonds$carat`

`diamonds$price`

[

positive integers	select specified
negative integers	omit specified
characters	extract named items
nothing	include everything
logicals	select T, omit F

Challenge

There is an equivalency between logical (boolean) and numerical (set) indexing.

How do you change a logical index to a numeric index? And vice versa?

What are the equivalents of the boolean operations for numerical indices?

```
# Nothing
str(diamonds[, ])

# Positive integers & nothing
diamonds[1:6, ] # same as head(diamonds)
diamonds[, 1:4] # watch out!

# Positive integers * 2
diamonds[1:10, 1:4]
diamonds$carat[1:100]

# Negative integers
diamonds[-(1:53900), -1]

# Character vector
diamonds[, c("depth", "table")]
diamonds[1:100, "carat"]
```

[+ logical vectors

```
# The most complicated to understand, but  
# the most powerful. Lets you extract a  
# subset defined by some characteristic of  
# the data
```

```
x_big <- diamonds$x > 10
```

```
head(x_big)
```

```
tail(x_big)
```

```
sum(x_big)
```

```
diamonds$x[x_big]
```

```
diamonds[x_big, ]
```

Useful functions for logical vectors

TRUE = 1; FALSE = 0

```
table(zeros)
```

```
sum(zeros)
```

```
mean(zeros)
```

```
x_big <- diamonds$x > 10
diamonds[x_big, ]
diamonds[x_big, "x"]
diamonds[x_big, c("x", "y", "z")]

small <- diamonds[diamonds$carat < 1, ]
lowqual <- diamonds[diamonds$clarity
  %in% c("I1", "SI2", "SI1"), ]

# Comparison functions:
# < > <= >= != == %in%

# Boolean operators
small <- diamonds$carat < 1 &
  diamonds$price > 500
lowqual <- diamonds$colour == "D" |
  diamonds$cut == "Fair"
```

And	<code>a & b</code>
Or	<code>a b</code>
Not	<code>!b</code>
Xor	<code>xor(a, b)</code>

Saving results

```
# Prints to screen
```

```
diamonds[diamonds$x > 10, ]
```

```
# Saves to new data frame
```

```
big <- diamonds[diamonds$x > 10, ]
```

```
# Overwrites existing data frame. Dangerous!
```

```
diamonds <- diamonds[diamonds$x < 10, ]
```

```
diamonds <- diamonds[1, 1]  
diamonds
```

```
# Uh oh!
```

```
rm(diamonds)  
str(diamonds)
```

```
# Phew!
```

Your turn

Extract diamonds with equal x & y .

Extract diamonds with incorrect/unusual x , y , or z values.

```
equal <- diamonds[diamonds$x == diamonds$y, ]
```

```
y_big <- diamonds$y > 10
```

```
z_big <- diamonds$z > 6
```

```
x_zero <- diamonds$x == 0
```

```
y_zero <- diamonds$y == 0
```

```
z_zero <- diamonds$z == 0
```

```
zeros <- x_zero | y_zero | z_zero
```

```
bad <- y_big | z_big | zeros
```

```
dbad <- diamonds[bad, ]
```

Aside: strategy

The biggest problem I see new programmers make is trying to do too much at once.

Break the problem into pieces and solve the smallest piece first. Then check each piece before solving the next problem.

Making new variables

```
diamonds$pricepc <- diamonds$price /  
  diamonds$carat
```

```
diamonds$volume <- diamonds$x *  
  diamonds$y * diamonds$z
```

```
qplot(pricepc, carat, data = diamonds)  
qplot(carat, volume, data = diamonds)
```

Modifying values

Combination of subsetting and making new variables:

```
diamonds$x[x_zero] <- NA
```

```
diamonds$z[z_big] <- diamonds$z[z_big] / 10
```

These modify the data in place.

Be careful!

```
diamonds$volume <- diamonds$x *  
  diamonds$y * diamonds$z  
qplot(carat, volume, data = diamonds)
```

```
# Fix problems & replot
```

```
diamonds$x[x_zero] <- NA
```

```
diamonds$y[y_zero] <- NA
```

```
diamonds$z[z_zero] <- NA
```

```
diamonds$y[y_big] <- diamonds$y[y_big] / 10
```

```
diamonds$z[z_big] <- diamonds$z[z_big] / 10
```

```
diamonds$volume <- diamonds$x *
```

```
  diamonds$y * diamonds$z
```

```
qplot(carat, volume, data = diamonds)
```

Your turn

Fix the incorrect values and replot scatterplots of x , y , and z . Are all the unusual values gone?

Correct any other strange values.

Hint: If $\text{qplot}(a, b)$ is a straight line, $\text{qplot}(a, a / b)$ will be a flat line. Makes selecting strange values much easier!

```
qplot(carat, volume, data = diamonds)
qplot(carat, volume / carat, data = diamonds)

weird_density <-
  (diamonds$volume / diamonds$carat) < 140 |
  (diamonds$volume / diamonds$carat) > 180
weird_density <- weird_density & !is.na(weird_density)

diamonds[weird_density, c("x", "y", "z", "volume")] <- NA
```

Short cuts

You've been typing diamonds many many times. There are three shortcuts: **with**, **subset** and **transform**.

These save typing, but may be a little harder to understand, and will not work in some situations.

Useful tools, but don't forget the basics.

```
weird_density <-  
  (diamonds$volume / diamonds$carat) < 140 |  
  (diamonds$volume / diamonds$carat) > 180  
weird_density <- with(diamonds,  
  (volume / carat) < 140 | (volume / carat) > 180)
```

```
diamonds[diamonds$carat < 1]  
subset(diamonds, carat < 1)
```

```
equal <- diamonds[diamonds$x == diamonds$y, ]  
equal <- subset(diamonds, x == y)
```

```
diamonds$volume <- diamonds$x * diamonds$y *  
  diamonds$z
```

```
diamonds$pricepc <- diamonds$price /  
  diamonds$carat
```

```
diamonds <- transform(diamonds,  
  volume = x * y * z,  
  pricepc = price / carat)
```

Your turn

Try to convert your previous statements to use with, subset and transform. Which ones convert easily? Which are hard?

When is the shortcut actually a longcut?

Next time

Learning how to use latex: a scientific publishing program.

If you're using a laptop, please install latex from the links on the course webpage.

<code>a & b</code>	<code>intersect(c, d)</code>
<code>a b</code>	<code>union(c, d)</code>
<code>!b</code>	<code>setdiff(U, c)</code>
<code>xor(a, b)</code>	<code>union(setdiff(c, d), setdiff(d, c))</code>

`c = which(a)`
`d = which(b)`

`U = seq_along(a)`
`a = U %in% c`
`b = U %in% d`