

Stat405

Subsetting and missing values

Hadley Wickham

Final poster session
Thursday November 29th
4–6pm

1. Subsetting

2. Missing values

3. Team projects & this week's
homework

Subsetting

Your turn

With your neighbour, run the code on the following slide **IN YOUR HEADS**

```
# Predict what the following code will do
```

```
# DON'T RUN IT!
```

```
x <- c(6, 1, 3, 6, 10, 5)
```

```
x[-(1:4)]
```

```
x[c(5, 6)]
```

```
x[x > 5]
```

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

```
diamonds[1:10, c("carat", "cut")]
```

blank

include all

integer

+ve: include

0: include none

-ve: exclude

logical

keep TRUEs

character

lookup by name

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!
```

```
# Blank
```

```
str(diamonds[, ])
```

```
# Positive integers
```

```
# Positive integers & nothing
```

```
diamonds[1:6, ] # same as head(diamonds)
```

```
diamonds[, 1:4] # watch out!
```

```
# Two positive integers in rows & columns
```

```
diamonds[1:10, 1:4]
```

```
# Repeating input repeats output
```

```
diamonds[c(1,1,1,2,2), 1:4]
```

```
# Negative integers
```

```
diamonds[-(1:53900), -1]
```

```
# Logical vectors
```

```
# Use logical comparisons to describe which values  
you want. Comparison functions:
```

```
# < > <= >= != == %in%
```

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

```
head(x_big)
```

```
sum(x_big)
```

```
mean(x_big)
```

```
table(x_big)
```







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

```
diamonds[x_big, ]
```

```
# Logical vectors
```

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

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

	a
	b
	a b
	a & b
	a & !b
	xor(a, b)

Your turn

Select the diamonds that have:

Equal x and y dimensions.

Depth between 55 and 70.

Carat smaller than the mean.

Cost more than \$10,000 per carat.

Are of very good quality or better.

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

diamonds[diamonds$depth >= 55 & diamonds$depth <= 70, ]

diamonds[diamonds$carat < mean(diamonds$carat), ]

diamonds[diamonds$price / diamonds$carat < 10000, ]

diamonds[diamonds$cut > "Good", ]
# OR
diamonds[diamonds$cut %in%
  c("Very Good", "Premium", "Ideal"), ]
```



```
# Common mistakes
```

```
diamonds[diamonds$color == "D" | "E" | "F", ]
```

```
diamonds[diamonds$color %in% c("D", "E", "F"), ]
```

```
diamonds[diamonds$depth >= 55 & <= 75]
```

```
diamonds[diamonds$cut = "Good", ]
```

```
good_cut <- diamonds$cut = "Good"
```

```
# Character subsetting
```

```
diamonds[1:5, c("carat", "cut", "color")]
```

```
diamonds[1:5, c(4, 9, 3)]
```

```
# Useful technique: change labelling
labels <- c("Fair" = "C", "Good" = "B",
  "Very Good" = "B+", "Premium" = "A",
  "Ideal" = "A+")
labels
labels["Fair"]
labels["Very Good"]

first_10 <- diamonds$cut[1:10]
first_10
labels[first_10]

all <- labels[diamonds$cut]
table(all)
```

```
# Can also be used to collapse levels
table(c("Fair" = "C", "Good" = "B", "Very Good" =
"B", "Premium" = "A", "Ideal" = "A")[diamonds$cut])

# If you're confused by a big statement, break it
# up in to smaller pieces

grades <- c("Fair" = "C", "Good" = "B", "Very Good"
= "B", "Premium" = "A", "Ideal" = "A")
grades
cuts <- diamonds$cut
head(grades[cuts])
table(grades[cuts])

# (see ?cut for continuous to discrete equivalent)
```

Your turn

The `f1` variable gives the type of fuel (r = regular, d = diesel, p = premium, c = cng, e = ethanol). Modify `f1` to spell out the fuel type explicitly, collapsing c, d and e into a single “other” category.

(Warning: run `mpg$f1 <- as.character(mpg$f1)` first)

```
mpg$fl <- as.character(mpg$fl)
```

```
mpg$fl <- c("r" = "regular", "p" = "premium", "d" =  
"other", "c" = "other", "e" = "other")[mpg$fl]
```

```
# Note this is a little dangerous, so you might  
# want to do it in multiple steps:
```

```
table(mpg$fl)
```

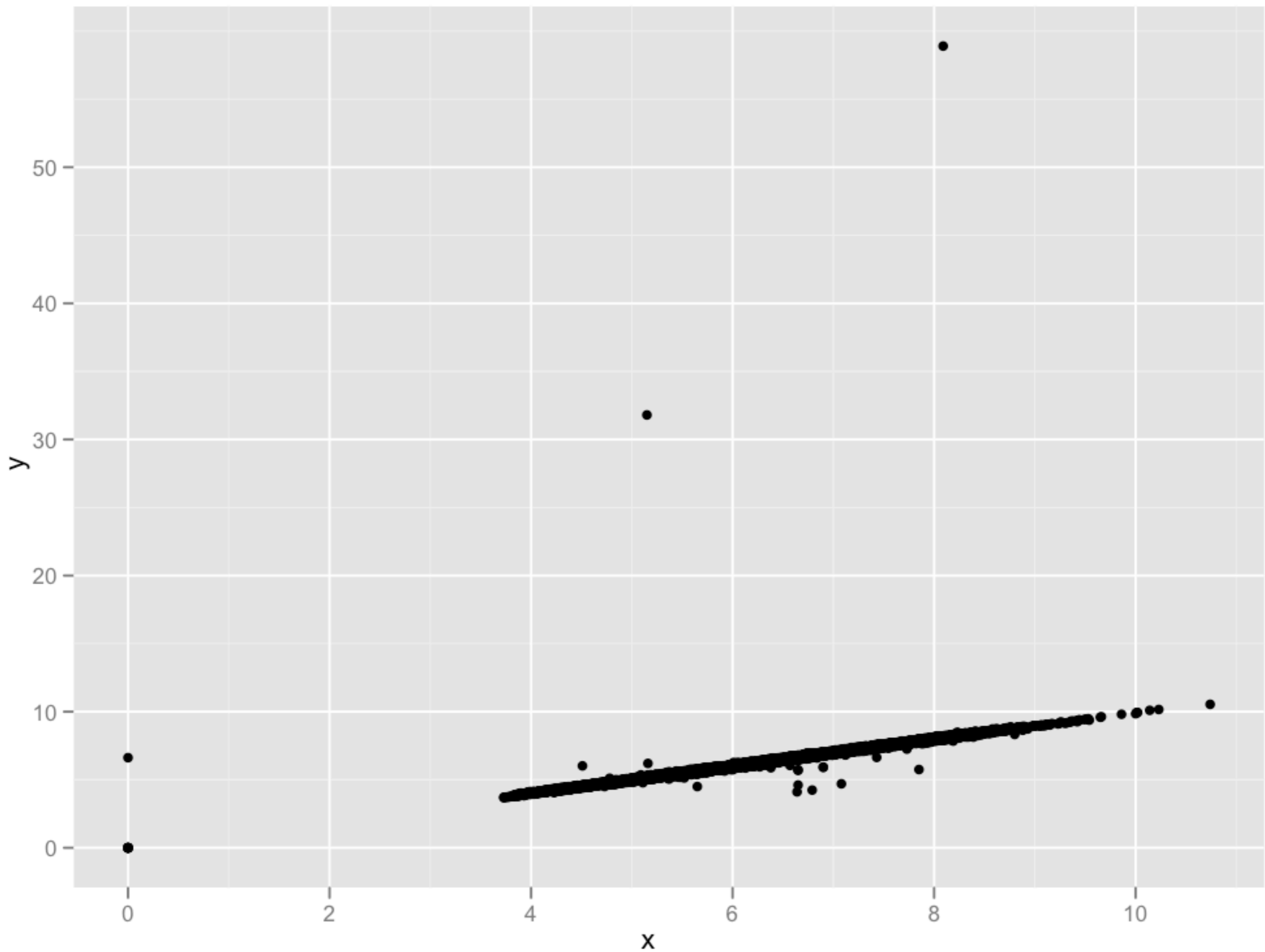
```
fl_map <- c("r" = "regular", "p" = "premium", "d" =  
"other", "c" = "other", "e" = "other")
```

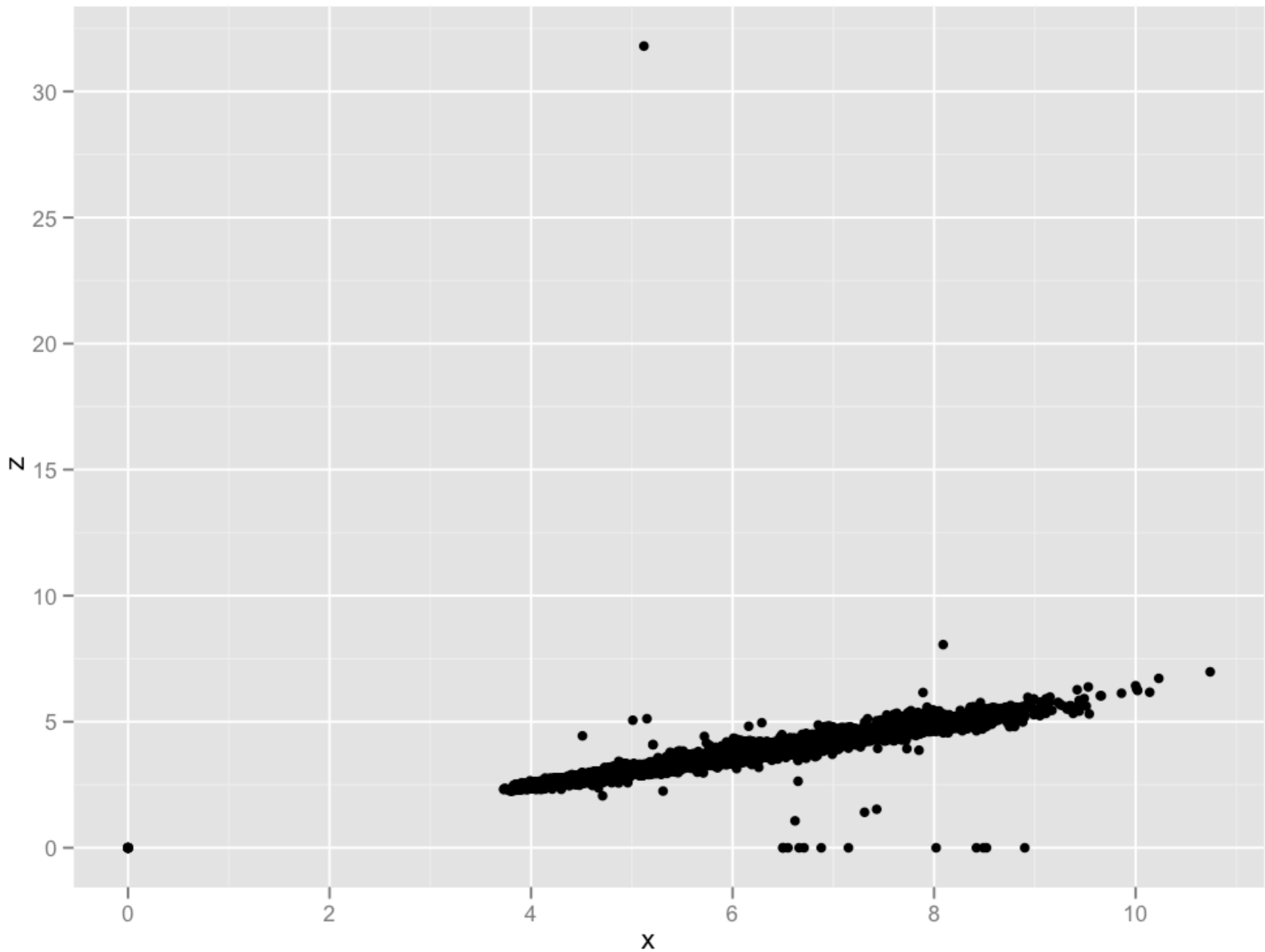
```
table(fl_map[mpg$fl])
```

```
mpg$fl <- fl_map[mpg$fl]
```

```
# Better to take small steps and check after each one
```

Missing values





```
y_big <- diamonds$y > 20
z_big <- diamonds$z > 20

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
good <- diamonds[!bad, ]

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

Data errors

Typically removing the entire row because of one error is overkill. Better to selectively replace problem values with missing values.

In R, missing values are indicated by NA

Expression	Guess	Actual
5 + NA		
NA / 2		
sum(c(5, NA))		
mean(c(5, NA))		
NA < 3		
NA == 3		
NA == NA		

NA behaviour

Missing values propagate

Use `is.na()` to check for missing values

Many functions (e.g. `sum` and `mean`) have `na.rm` argument to remove missing values prior to computation.

```
# Can use subsetting + <- to change individual  
# values
```

```
diamonds$x[diamonds$x == 0] <- NA  
diamonds$y[diamonds$y == 0] <- NA  
diamonds$z[diamonds$z == 0] <- NA
```

```
y_big <- diamonds$y > 20  
diamonds$y[y_big] <- NA  
z_big <- diamonds$z > 20  
diamonds$z[y_big] <- NA
```

Team projects

Teams

Assigned by Hadley. **Why?** No friends. Diverse abilities and backgrounds. Like real life.

Teams automatically dissolved after first project **unless** all team members submit signed request to stay together.

A team may **fire** a non-performing member. You may **resign** if you feel you are doing all the work.

Firing and resigning

- 1) The entire team meets with Hadley.
- 2) One week after meeting, if problems continue, send formal notice to fire/resign.

If you are fired, you need to find a team that will take you on, or work on all projects yourself.

Project grades

Each project will receive one overall grade.

Each team member will receive a grade adjusted by **team citizenship**, as assessed by team members.

Will discuss in more detail next week.

This week

Set up initial team meeting and decide on team name.

Plan common meeting time (weekly meeting recommended) and exchange contact details.

Discuss your expectations. How much time do you expect each team member to put into the project?

How will you communicate between team meetings?

How will you turn individual efforts into an integrate whole? How should you prepare for group meetings?

What will you do if a team member does not do the work?

Project details

Like homework on steroids - 3-4 interconnected questions.

Larger fuel economy dataset. (But primary question can not be about fuel economy)

Meet with intermediate questions/results to discuss with Yeshaya/Barrett/Hadley.

More information on website.