Stat405

Problem solving

Hadley Wickham

1. Homework & project updates
2. Saving data
3. Slot machine challenge

Homework

(Common problems)

```
library(ggplot2)
mpg2 <- read.csv("mpg2.csv.bz2", stringsAsFactors = FALSE)

# Be sceptical
recent <- subset(mpg2, year >= 1998 &
   fueltype %in% c("CNG", "Diesel", "Regular", "Premium"))
qplot(year, cty, data = recent, colour = fueltype,
  geom = "smooth")
qplot(year, cty, data = recent, colour = fueltype,
  geom = "jitter")

# Be curious
qplot(year, cty, data = recent, geom = "boxplot", group = year) +
  facet_wrap(~ fueltype) +
  geom_smooth(colour = "red")
```

Project

- Due on Tuesday Sep 25
- Make sure to meet with Barret, Shaya or myself for project review
- This week’s homework is pretty light: practice code styling and loading and saving data. Work on the project!
- **Recommendation**: reserve next week (Thursday-Tuesday) for final polishing, printing etc.

Project review

- Meetings will last about 15 minutes
- We'll give you feedback on your current direction, ask questions and offer suggestions. The more you have to bring the better.
- *Barret* tomorrow 12-2, *Me* 2-5 tomorrow (in the pavilion), *Yeshaya* 11-3 Monday
- Email all three of us and cc your team members.
- If one of those slots doesn't work, please provide three time slots that work for your team.

Saving data

Quiz

How do you load a csv file into R?

What’s the difference between a character vector (string) and a factor? When do you use strings? When do you use factors?

```
# Make sure your working directory is set correctly!

slots <- read.delim(”slots.txt”, sep = ” ”, header = F,
  stringsAsFactors = F)
names(slots) <- c(”w1”, ”w2”, ”w3”, ”prize”, ”night”)

levels <- c(0, 1, 2, 3, 5, 6, 7)
labels <- c(”0”, ”B”, ”BB”, ”BBB”, ”DD”, ”C”, ”7”)

slots$w1 <- factor(slots$w1, levels = levels, labels = labels)
slots$w2 <- factor(slots$w2, levels = levels, labels = labels)
slots$w3 <- factor(slots$w3, levels = levels, labels = labels)
```

Your turn

Guess the name of the function you might use to write an R object back to a csv file on disk. Use it to save `slots` to `slots-2.csv`.

What happens if you now read in `slots-2.csv`? Is it different to your `slots` data frame? How?

```
write.csv(slots, ”slots-2.csv”)
slots2 <- read.csv(”slots-2.csv”)

head(slots)
head(slots2)

str(slots)
str(slots2)

# Better, but still loses factor levels
write.csv(slots, file = ”slots-3.csv”, row.names = F)
slots3 <- read.csv(”slots-3.csv”)
```

Saving data

```
# For long-term storage
write.csv(slots, file = ”slots.csv”,
  row.names = FALSE)

# For short-term caching
# Preserves factors etc.
saveRDS(slots, ”slots.rds”)
slots2 <- readRDS(”slots.rds”)
```

.csv	.rds
read.csv()	readRDS()
write.csv(row.names = FALSE)	saveRDS()
Only data frames	Any R object
Plain text	Binary

Plain text	**Binary**
Human readable	Machine readable
Easy to understand	Very fast to load
Big	Small
Long term storage	Short term caching

Slot machine payoffs

Slots

Casino claims that slot machines have prize payout of 92%. Is this claim true?

```
mean(slots$prize)
```

```
t.test(slots$prize, mu = 0.92)
```

```
qplot(prize, data = slots, binwidth = 1)
```

How can we do better?

Idea

We have enough information (distribution of windows and payoffs) to simulate the slot machine.

We could write code to simulate a single pull, then run it thousands of times and compare to 92%.

Strategy

1. Break complex tasks into smaller parts

2. Use words to describe how each part should work

3. Translate words to R

4. When all parts work, combine into a function (next class)

DD	DD	DD	800
7	7	7	80
BBB	BBB	BBB	40
BB	BB	BB	25
B	B	B	10
C	C	C	10
Any bar	Any bar	Any bar	5
C	C	*	5
C	*	C	5
C	*	*	2
*	C	*	2
*	*	C	2

```
# Challenge: given e.g.
windows <- c(”7”, ”C”, ”C”)
# how can we calculate the
# payoff in R?
```

DD doubles any winning combination. Two DD quadruples. DD is wild.

Your turn

We can simplify this table into 3 basic cases of prizes. What are they? Take 3 minutes to brainstorm with a partner.

Cases

1. All windows have same value
2. A bar (B, BB, or BBB) in every window
3. Cherries and diamonds
4. (No prize)

Same values

Same values

1. Check whether all windows are the same. How?

Same values

1. Check whether all windows are the same. How?
2. If so, look up prize value. How?

Same values

1. Check whether all windows are the same. How?
2. If so, look up prize value. How?

With a partner, brainstorm for 2 minutes on how to solve one of these problems

```r
# Same value
same <- length(unique(windows)) == 1


# OR
same <- windows[1] == windows[2] &&
        windows[2] == windows[3]


if (same) {
  # Lookup value
}
```

&&, || vs. &, |

Use `&&` and `||` to combine sub-conditions and return a single `TRUE` or `FALSE`. `&&` and `||` are “short-circuiting”: they do the minimum amount of work

Different to `&` and `|` - these return vectors when given vector.

If

```
if (condition) {
  expression
}
```

`Condition` should be a logical vector of length 1

```
if (TRUE) {
  # This will be run
}

if (FALSE) {
  # This will be run
} else {
  # This will be
}

# Single line form: (not recommended)
if (TRUE) print(”True!)
if (FALSE) print(”True!)
if (TRUE) {
  # This will be run
}

if (FALSE) {
  # This will be run
} else {
  # This will be
}

#              orm: (not recommended)
if (TRUE) print("True!)
if (FALSE) print("True!)
```

Note indenting.
Very important!

```
x <- 5
if (x < 5) print(”x < 5”)
if (x == 5) print(”x == 5”)

x <- 1:5
if (x < 3) print(”What should happen here?”)

if (x[1] < x[2]) print(”x1 < x2”)
if (x[1] < x[2] && x[2] < x[3]) print(”Asc”)
if (x[1] < x[2] || x[2] < x[3]) print(”Asc”)
if (window[1] == "DD") {
  prize <- 800
} else if (windows[1] == "7") {
  prize <- 80
} else if (windows[1] == "BBB") ...

# Or use subsetting
c("DD" = 800, "7" = 80, "BBB" = 40)
c("DD" = 800, "7" = 80, "BBB" = 40)["BBB"]
c("DD" = 800, "7" = 80, "BBB" = 40)["0"]
c("DD" = 800, "7" = 80, "BBB" = 40)[window[1]]
```

Your turn

Complete the previous code so that if all the values in win are the same, then prize variable will be set to the correct amount.

All bars

How can we determine if all of the windows are B, BB, or BBB?

```
(windows[1] == ”B” ||
 windows[1] == ”BB” ||
 windows[1] === ”BBB”) && ... ?
```

All bars

How can we determine if all of the windows are B, BB, or BBB?

```
(windows[1] == ”B” ||
 windows[1] == ”BB” ||
 windows[1] === ”BBB”) && ... ?
```

Take 1 minute to brainstorm possible solutions

```
windows[1] %in% c("B", "BB", "BBB")
windows %in% c("B", "BB", "BBB")

allbars <- windows %in% c("B", "BB", "BBB")
allbars[1] & allbars[2] & allbars[3]
all(allbars)


# See also ?any for the complement
```

Your turn

Complete the previous code so that the correct value of prize is set if all the windows are the same, or they are all bars

```
payoffs <- c(”DD” = 800, ”7” = 80, ”BBB” = 40,
  ”BB” = 25, ”B” = 10, ”C” = 10, ”0” = 0)

same <- length(unique(windows)) == 1
allbars <- all(windows %in% c(”B”, ”BB”, ”BBB”))

if (same) {
  prize <- payoffs[windows[1]]
} else if (allbars) {
  prize <- 5
}
```

Cherries

Need numbers of cherries, and numbers of diamonds (hint: use `sum`)

Then need to look up values (like for the first case) and multiply together

```
cherries <- sum(windows == ”C”)
diamonds <- sum(windows == ”DD”)

c(0, 2, 5)[cherries + 1] *
  c(1, 2, 4)[diamonds + 1]
```

```
payoffs <- c("DD" = 800, "7" = 80, "BBB" = 40,
  "BB" = 25, "B" = 10, "C" = 10, "0" = 0)

same <- length(unique(windows)) == 1
allbars <- all(windows %in% c("B", "BB", "BBB"))

if (same) {
  prize <- payoffs[windows[1]]
} else if (allbars) {
  prize <- 5
} else {
  cherries <- sum(windows == "C")
  diamonds <- sum(windows == "DD")

  prize <- c(0, 2, 5)[cherries + 1] *
    c(1, 2, 4)[diamonds + 1]
}
```

Writing a function

Now we need to wrap up this code in to a reusable fashion. We need a `function`

Have used functions a lot, next time we'll learn how to write one.