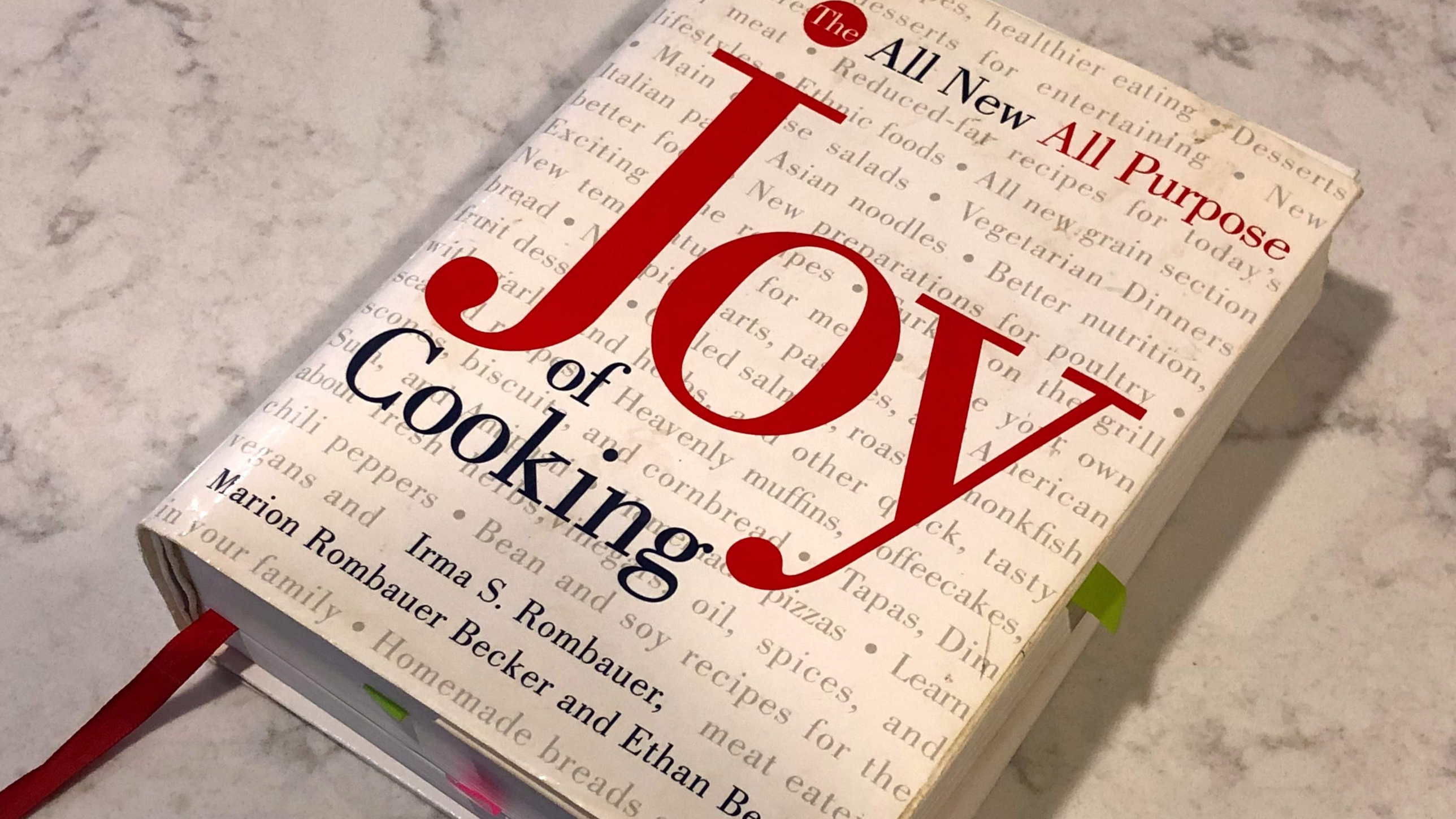
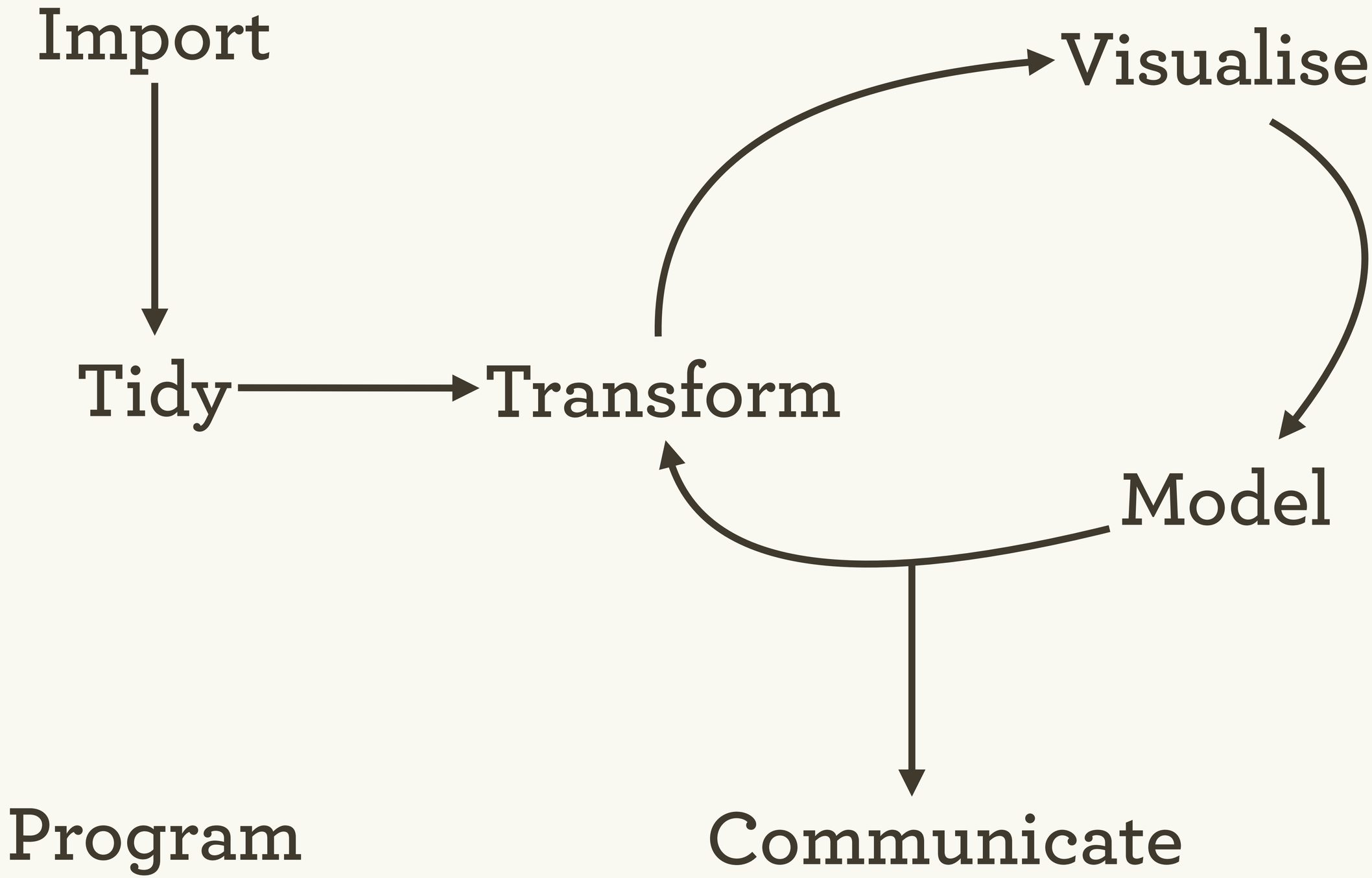
The joy of functional programming

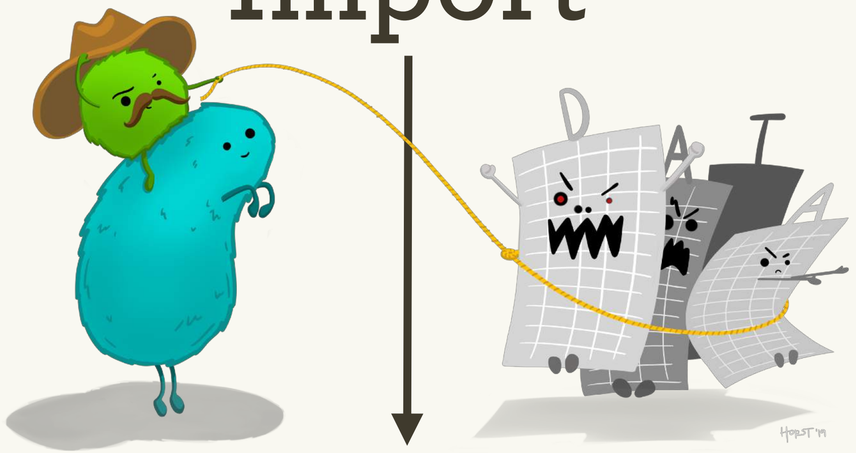
June 2019

Hadley Wickham
@hadleywickham
Chief Scientist, **RStudio**


The All New All Purpose
Joy of Cooking
Irma S. Rombauer,
Marion Rombauer Becker and Ethan Be


Import
Visualise
Tidy
Transform
Model
Program
Communicate

Import

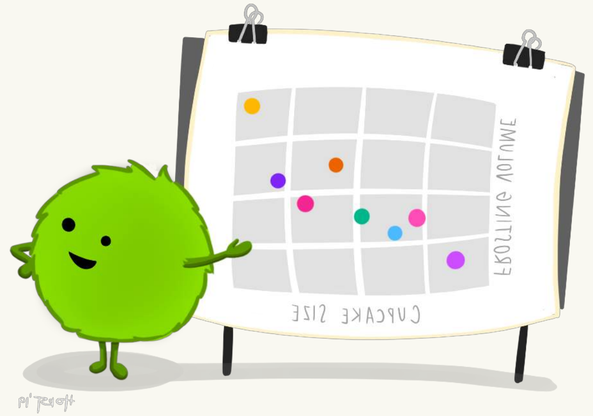


Tidy

Transform

Visualise

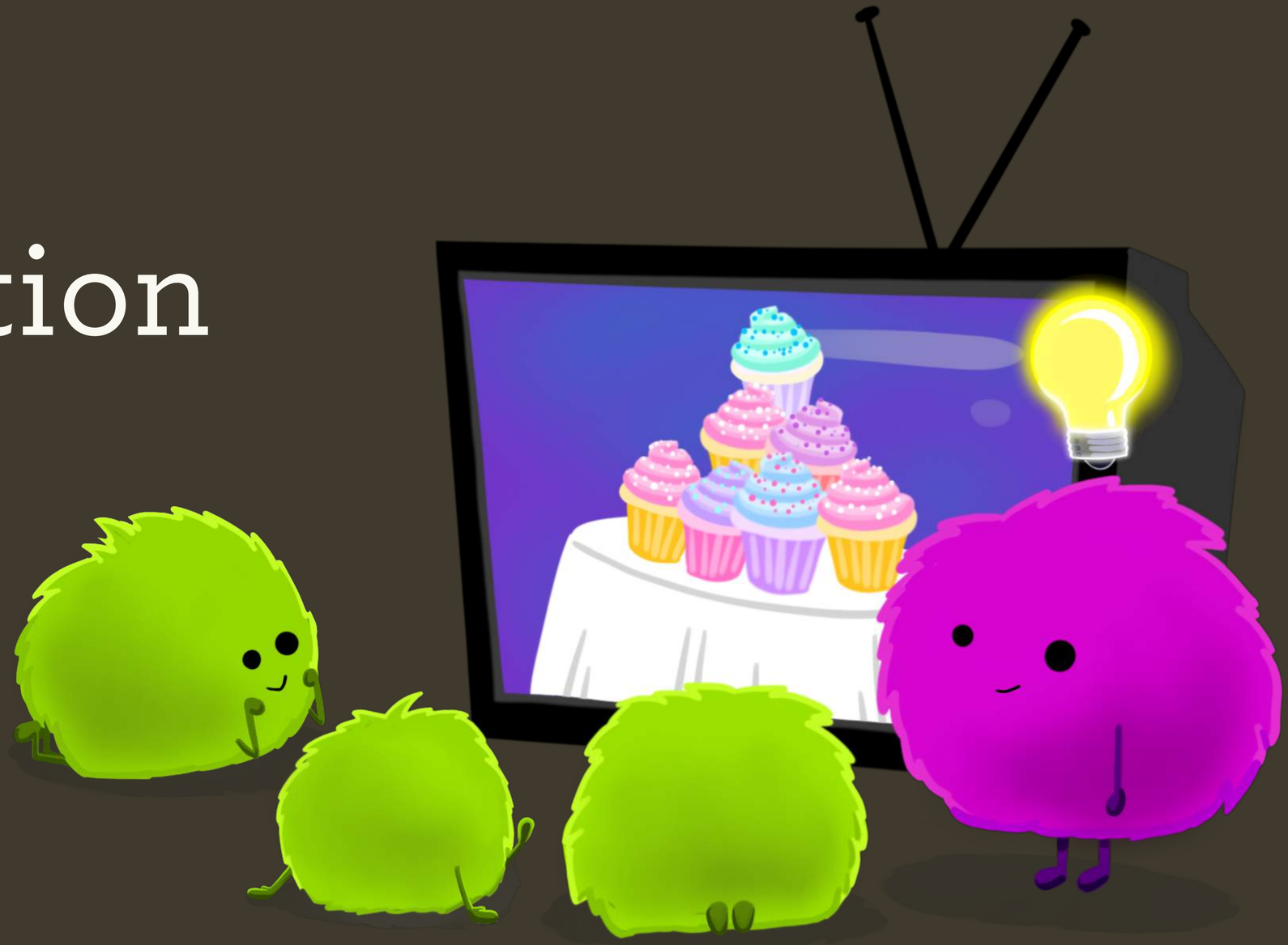
Model



Communicate

Program

Motivation



Imagine we want to read in a bunch of csv files

```
# Find all the csv files in the current directory
paths <- dir(pattern = "\\.csv$")


# And read them in as data frames
data <- vector("list", length(paths))
for (i in seq_along(paths)) {
  data[[i]] <- read.csv(paths[[i]])
}
```

Imagine we want to read in a bunch of csv files

```
# Find all the csv files in the current directory
paths <- dir(pattern = "\\.csv$")

# And read them in as data frames
data <- vector("list", length(paths))
for (i in seq_along(paths)) {
  data[[i]] <- read.csv(paths[[i]])
}
```

R uses <- for assignment

A loop always has three components

```
data <- vector("list", length(paths))
for (i in seq_along(paths)) {
  data[[i]] <- read.csv(paths[[i]])
}
```

1. Space for the output

Create a new list of the correct size

```
data <- vector("list", length(paths))
for (i in seq_along(paths)) {
  data[[i]] <- read.csv(paths[[i]])
}
```

2. A vector to iterate over

```
data <- vec                                    s))
for (i in seq_along(paths)) {
  data[[i]] <- read.csv(paths[[i]])
}
```

Creates an integer vector from 1 to length(paths)

Avoid 1:length(paths) because it fails in unhappy way if paths has length 0

3. Code that's run for every iteration

```
data <- vector("list", length(paths))
for (i in seq_along(paths)) {
  data[[i]] <- read.csv(paths[[i]])
}
```

Extract element i from paths

Use [[whenever you get or set a single element

There’s nothing wrong with using a loop

```
library(purrr)

# But the FP equivalent is much shorter
data <- map(paths, read.csv)

# And has convenient extensions
data <- map_dfr(paths, read.csv, id = "path")
```

Why not for loops?

Vanilla cupcakes

The hummingbird bakery cookbook

1 cup flour
a scant ¾ cup sugar
1 ½ t baking powder
3 T unsalted butter
½ cup whole milk
1 egg
¼ t pure vanilla extract

Preheat oven to 350°F.

Put the flour, sugar, baking powder, salt, and butter in a freestanding electric mixer with a paddle attachment and beat on slow speed until you get a sandy consistency and everything is combined.

Whisk the milk, egg, and vanilla together in a pitcher, then slowly pour about half into the flour mixture, beat to combine, and turn the mixer up to high speed to get rid of any lumps.

Turn the mixer down to a slower speed and slowly pour in the remaining milk mixture. Continue mixing for a couple of more minutes until the batter is smooth but do not overmix.

Spoon the batter into paper cases until 2/3 full and bake in the preheated oven for 20-25 minutes, or until the cake bounces back when touched.

Chocolate cupcakes

The hummingbird bakery cookbook

¾ cup + 2T flour
2 ½ T cocoa powder
a scant ¾ cup sugar
1 ½ t baking powder
3 T unsalted butter
½ cup whole milk
1 egg
¼ t pure vanilla extract

Preheat oven to 350°F.

Put the flour, cocoa, sugar, baking powder, salt, and butter in a freestanding electric mixer with a paddle attachment and beat on slow speed until you get a sandy consistency and everything is combined.

Whisk the milk, egg, and vanilla together in a pitcher, then slowly pour about half into the flour mixture, beat to combine, and turn the mixer up to high speed to get rid of any lumps.

Turn the mixer down to a slower speed and slowly pour in the remaining milk mixture. Continue mixing for a couple of more minutes until the batter is smooth but do not overmix.

Spoon the batter into paper cases until 2/3 full and bake in the preheated oven for 20-25 minutes, or until the cake bounces back when touched.

Chocolate cupcakes

The hummingbird bakery cookbook

¾ cup + 2T flour
2 ½ T cocoa powder
a scant ¾ cup sugar
1 ½ t baking powder
3 T unsalted butter
½ cup whole milk
1 egg
¼ t pure vanilla extract

Preheat oven to 350°F.

Put the flour, cocoa, sugar, baking powder, salt, and butter in a freestanding electric mixer with a paddle attachment and beat on slow speed until you get a sandy consistency and everything is combined.

Whisk the milk, egg, and vanilla together in a pitcher, then slowly pour about half into the flour mixture, beat to combine, and turn the mixer up to high speed to get rid of any lumps.

Turn the mixer down to a slower speed and slowly pour in the remaining milk mixture. Continue mixing for a couple of more minutes until the batter is smooth but do not overmix.

Spoon the batter into paper cases until 2/3 full and bake in the preheated oven for 20-25 minutes, or until the cake bounces back when touched.

Vanilla cupcakes

120g flour
140g sugar
1.5 t baking powder
40g butter
120ml milk
1 egg
0.25 t vanilla

Preheat oven to 350°F.

Put the flour, sugar, baking powder, salt, and butter in a freestanding electric mixer with a paddle attachment and beat on slow speed until you get a sandy consistency and everything is combined.

Whisk the milk, egg, and vanilla together in a pitcher, then slowly pour about half into the flour mixture, beat to combine, and turn the mixer up to high speed to get rid of any lumps.

Turn the mixer down to a slower speed and slowly pour in the remaining milk mixture. Continue mixing for a couple of more minutes until the batter is smooth but do not overmix.

Spoon the batter into paper cases until 2/3 full and bake in the preheated oven for 20-25 minutes, or until the cake bounces back when touched.

Vanilla cupcakes

The hummingbird
bakery cookbook

120g flour
140g sugar
1.5 t baking powder
40g butter
120ml milk
1 egg
0.25 t vanilla

Beat flour, sugar, baking powder, salt, and butter until sandy.

Whisk milk, egg, and vanilla. Mix half into flour mixture until smooth (use high speed). Beat in remaining half. Mix until smooth.

Bake 20-25 min at 170°C.

Vanilla cupcakes

The hummingbird bakery cookbook

120g flour
140g sugar
1.5 t baking powder
40g butter
120ml milk
1 egg
0.25 t vanilla

Beat dry ingredients + butter until sandy.

Whisk together wet ingredients. Mix half into dry until smooth (use high speed). Beat in remaining half. Mix until smooth.

Bake 20-25 min at 170°C.

Cupcakes

Beat dry ingredients + butter until sandy.

Whisk together wet ingredients. Mix half into dry until smooth (use high speed). Beat in remaining half. Mix until smooth.

Bake 20-25 min at 170°C.

Vanilla

120g flour

140g sugar

1.5t baking powder

40g butter

120ml milk

1 egg

0.25 t vanilla

Chocolate

100g flour

20g cocoa

140g sugar

1.5t baking powder

40g butter

120ml milk

1 egg

0.25 t vanilla

Cupcakes

Beat dry ingredients + butter until sandy.

Whisk together wet ingredients. Mix half into dry until smooth (use high speed). Beat in remaining half. Mix until smooth.

Bake 20-25 min at 170°C.

Vanilla

120g flour

140g sugar

1.5t baking powder

40g butter

120ml milk

1 egg

0.25 t vanilla

Chocolate

100g flour

20g cocoa

140g sugar

1.5t baking powder

40g butter

120ml milk

1 egg

0.25 t vanilla

Espresso

120g flour

140g sugar

1.5t baking powder

40g butter

120ml milk + 10g espresso powder

1 egg

What do these for loops do?

```
out1 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out1[[i]] <- mean(mtcars[[i]], na.rm = TRUE)
}
```

Extracts column i

```
out2 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out2[[i]] <- median(mtcars[[i]], na.rm = TRUE)
}
```

	mpg	cyl	disp	hp	drat	
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	
1	21	6	160	110	3.9	...
2	21	6	160	110	3.9	...
3	22.8	4	108	93	3.85	...
4	21.4	6	258	110	3.08	...
5	18.7	8	360	175	3.15	...
.	...	.	...	...	....	...

For loops emphasise the objects

```
out1 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out1[[i]] <- mean(mtcars[[i]], na.rm = TRUE)
}

out2 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out2[[i]] <- median(mtcars[[i]], na.rm = TRUE)
}
```

Not the actions

```
out1 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out1[[i]] <- mean(mtcars[[i]], na.rm = TRUE)
}

out2 <- vector("double", ncol(mtcars))
for(i in seq_along(mtcars)) {
  out2[[i]] <- median(mtcars[[i]], na.rm = TRUE)
}
```

Functional programming weights action and object equally

```
out1 <- map_dbl(mtcars, mean, na.rm = TRUE)
out2 <- map_dbl(mtcars, median, na.rm = TRUE)
```

And combines well with the pipe

```
out1 <- mtcars %>% map_dbl(mean, na.rm = TRUE)
out2 <- mtcars %>% map_dbl(median, na.rm = TRUE)
```

Which is particularly important for harder problems

```
diamonds %>%
  split_by(diamonds$color) %>%
  map(~ lm(log(price) ~ log(carat), .x)) %>%
  map_dfr(broom::tidy, .id = "color")
```

Of course someone has to write loops. It doesn't have to be **you**.

— *Jenny Bryan*

Getting data

https://www.gov.uk/government/statistics/family-food-open-data

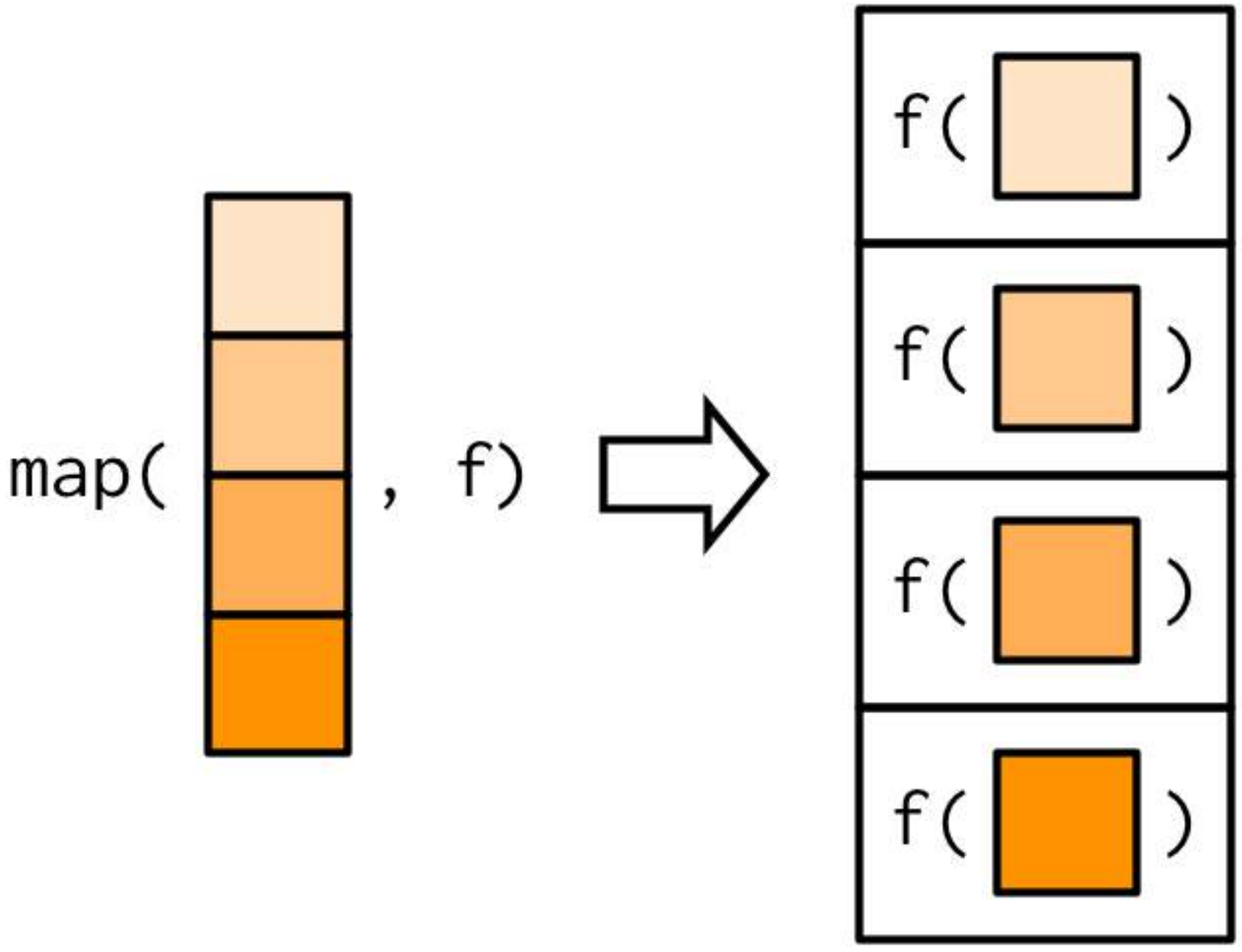
EXAMPLE PAGE

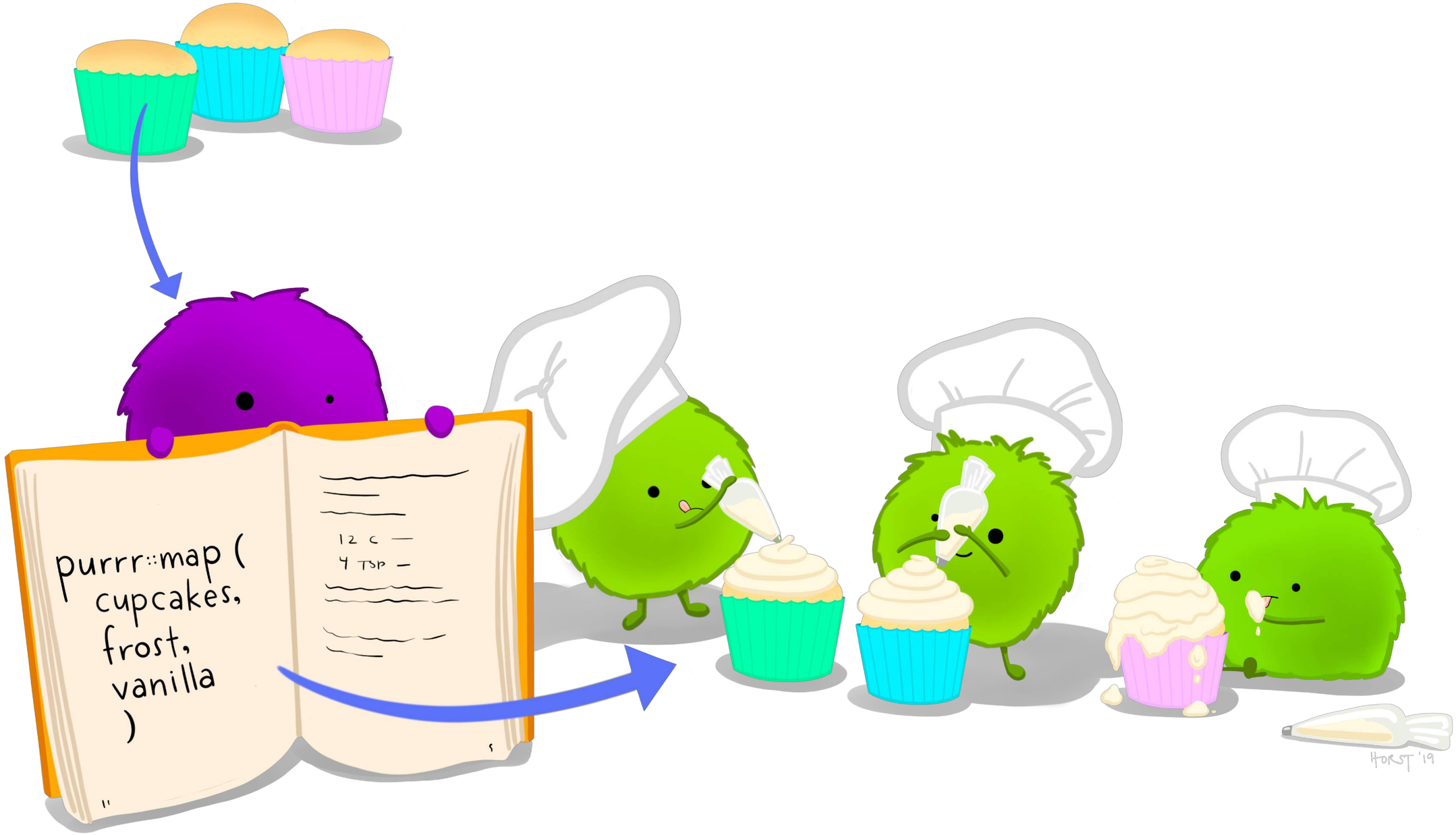
1. FOOD AND DRINKS BOUGHT AND BROUGHT HOME

(include soft drinks, alcoholic drinks, sweets, takeaways brought home and milk delivered today)

.....DAY

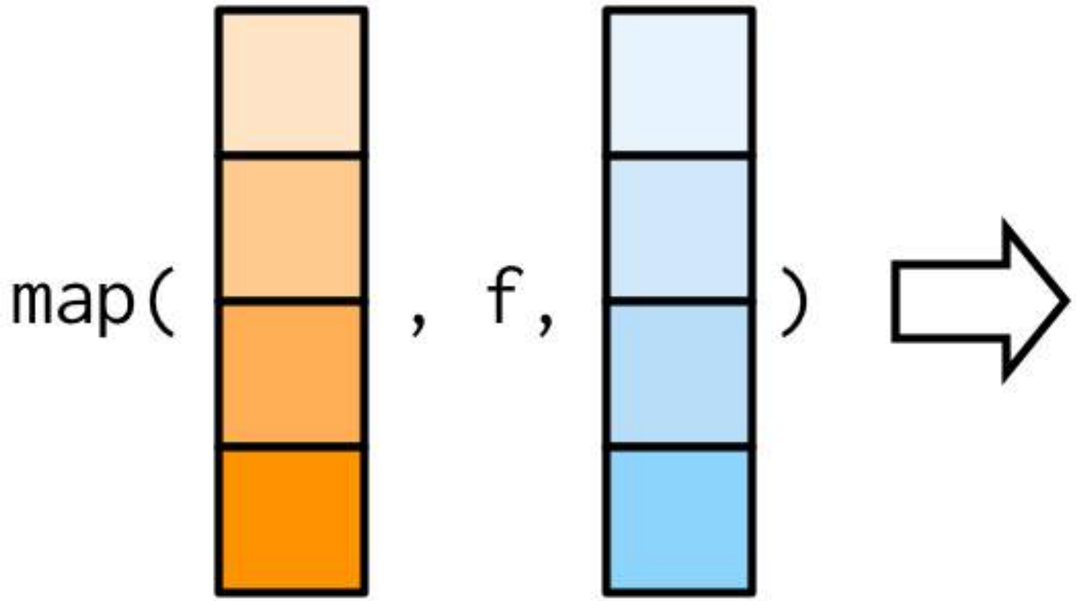
Check	TOTAL WEIGHT Oz, lbs, gms, kilos, pints, litres	DESCRIPTION OF FOOD OR DRINK *Please describe in FULL and give BRAND; Use one line for each ITEM*	TOTAL COST £	p	INTERVIEWER USE ONLY PLEASE LEAVE BLANK Food code	Qty	Unit	
	1pt	*Vita Pint (Blue Carton semi-skimmed milk)*		*38*				1
	1600g	*2 Sunblest sliced white loaves @ 800gms*	*1*	*20*				1
	450g	*Weetabix – Family Pack x 24 biscuits*	*1*	*39*				1
	1lb	*Frying steak – Fresh*	*5*	*25*				1
	1lb	*Pork chops – on bone – fresh*	*2*	*83*				1
	1lb	*1 pack danish bacon, streaky, pre-packed*	*2*	*74*				1
	12oz	*6 Doughnuts @ 2ozs – fresh from bakers*	*1*	*80*				1
	840g	*2 tins Heinz baked beans @ 420gms*		*84*				1
	200g	*Birds Eye frozen cod steaks in natural crumbs*	*2*	*09*				1
	1lb	*Walls beef sausages – frozen*	*1*	*32*				1
	493g	*McCain Deep Pizza, frozen - pepperoni*	*2*	*29*				1
	150g	*Ski fruit yogurt, low fat with pieces of fruit*		*42*				1
	500g	*2 packets Krona Reduced fat 2 @ 250gms*	*1*	*74*				1
	250g	*1 packet Kerrygold butter – Irish*		*95*				1
	5Kg	*"Old" Potatoes, fresh, prepacked*	*2*	*50*				1
	1lb 4oz	*Cauliflower – fresh*		*75*				1
	1½lb	*Eating apples – fresh*		*64*				1
	1 litre	*Robinson's orange squash – low calo*	*1*	*19*				1

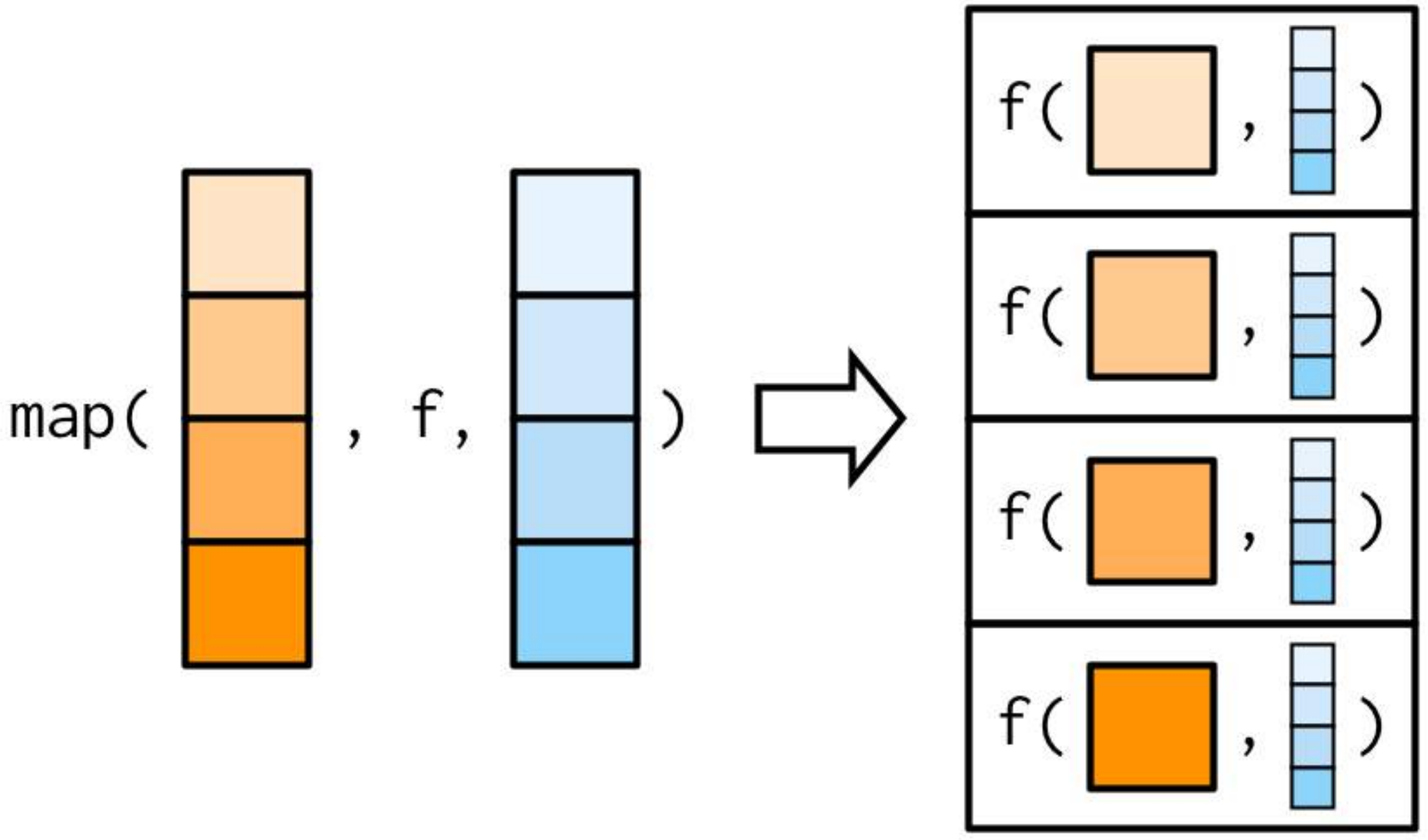

map(
, f)
f(
)
f(
)
f(
)
f(
)

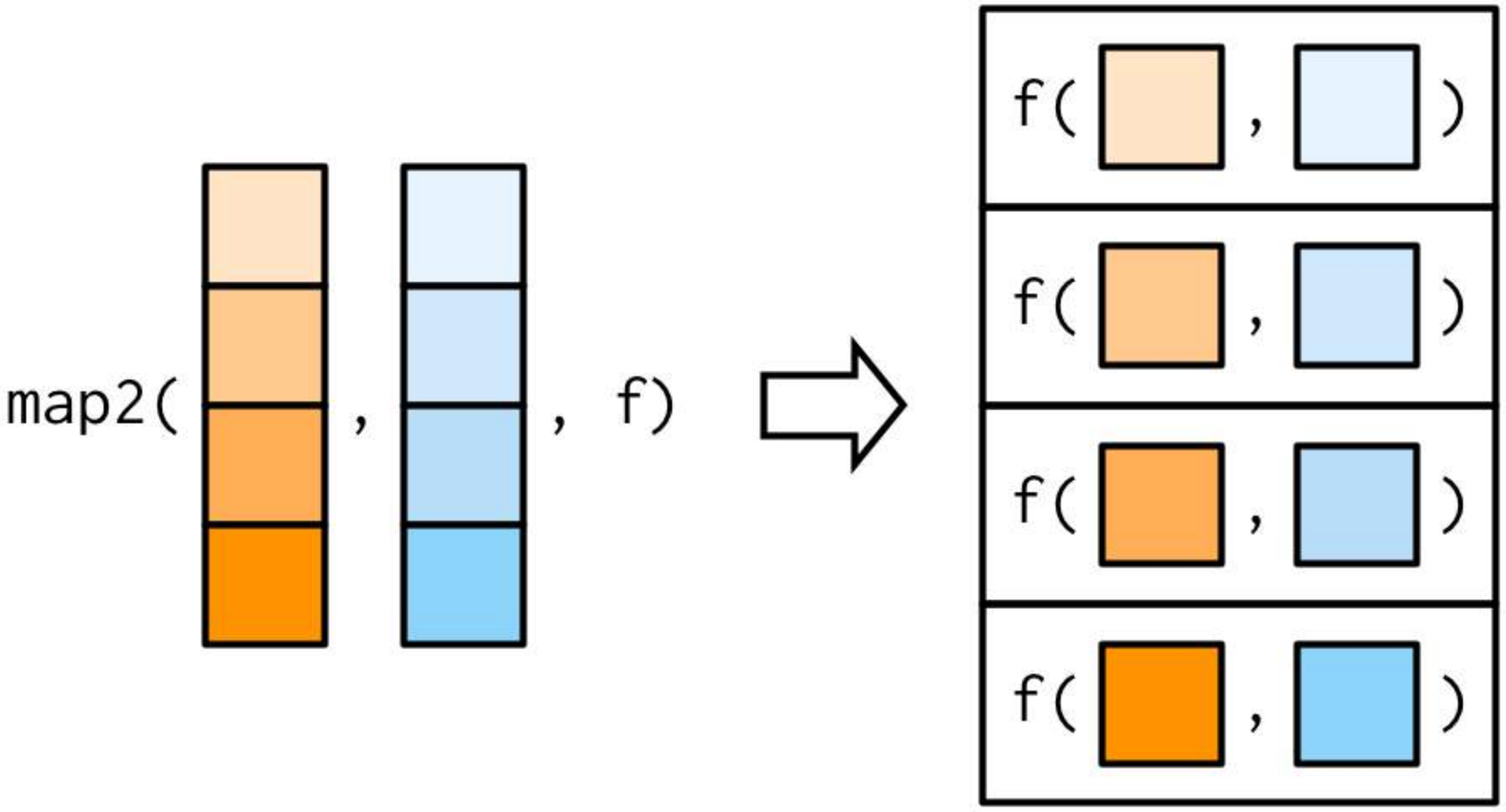

purrr::map(
cupcakes,
frost,
vanilla
)
12 C
4 TSP
HORST '19

Demo

Generating reports


map(
, f,
)


map(
, f,
)
f(
,
)
f(
,
)
f(
,
)
f(
,
)

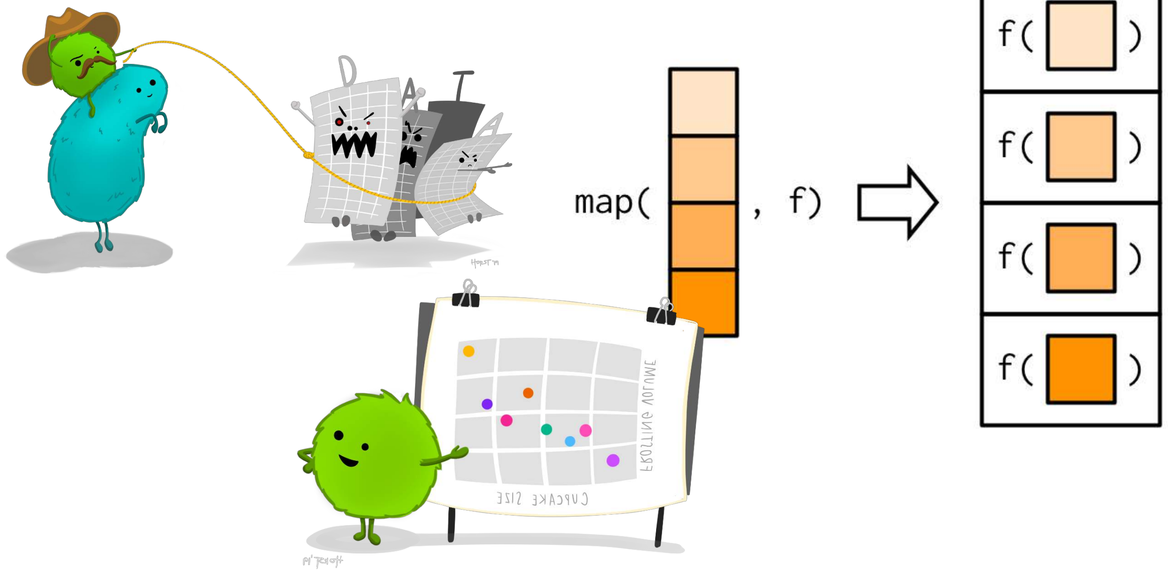

map2(, , f)
f(,)
f(,)
f(,)
f(,)

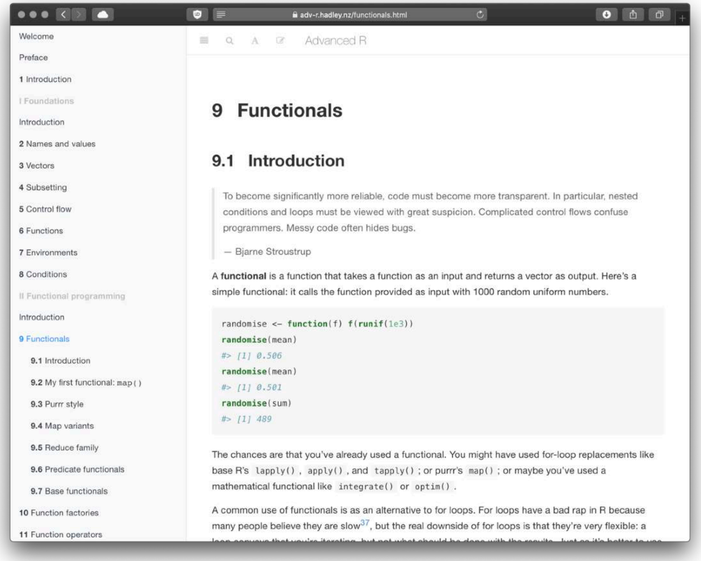

purrr::map2(
cupcakes,
frost,
c("vanilla",
"chocolate",
"strawberry"
)
)
12 c
4 tsp
HORST '19

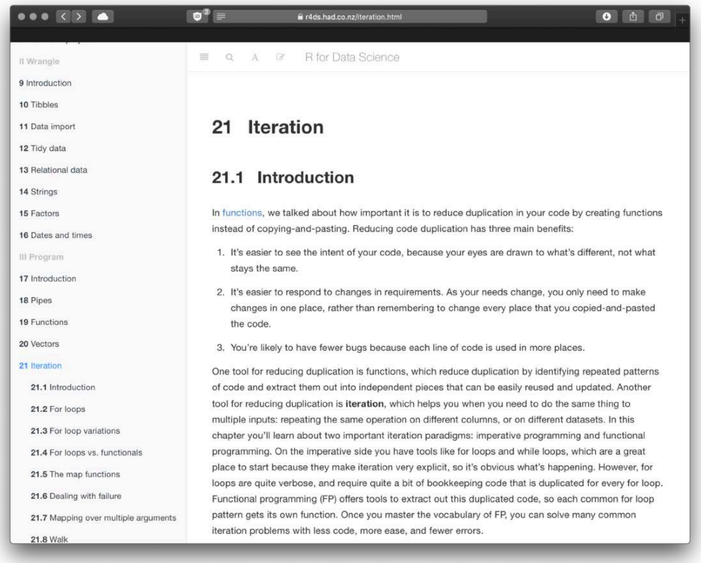
Demo

Conclusion

For loops aren't bad; but duplicated code can conceal important differences, and why do more work than you have to?

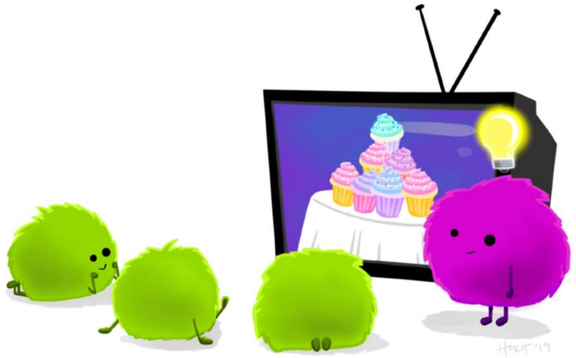


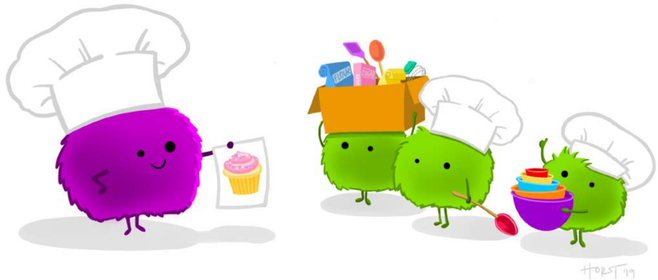




https://adv-r.hadley.nz/functionals.html

https://r4ds.had.co.nz/iteration.html





With big thanks to
Allison Horst!

https://github.com/allisonhorst