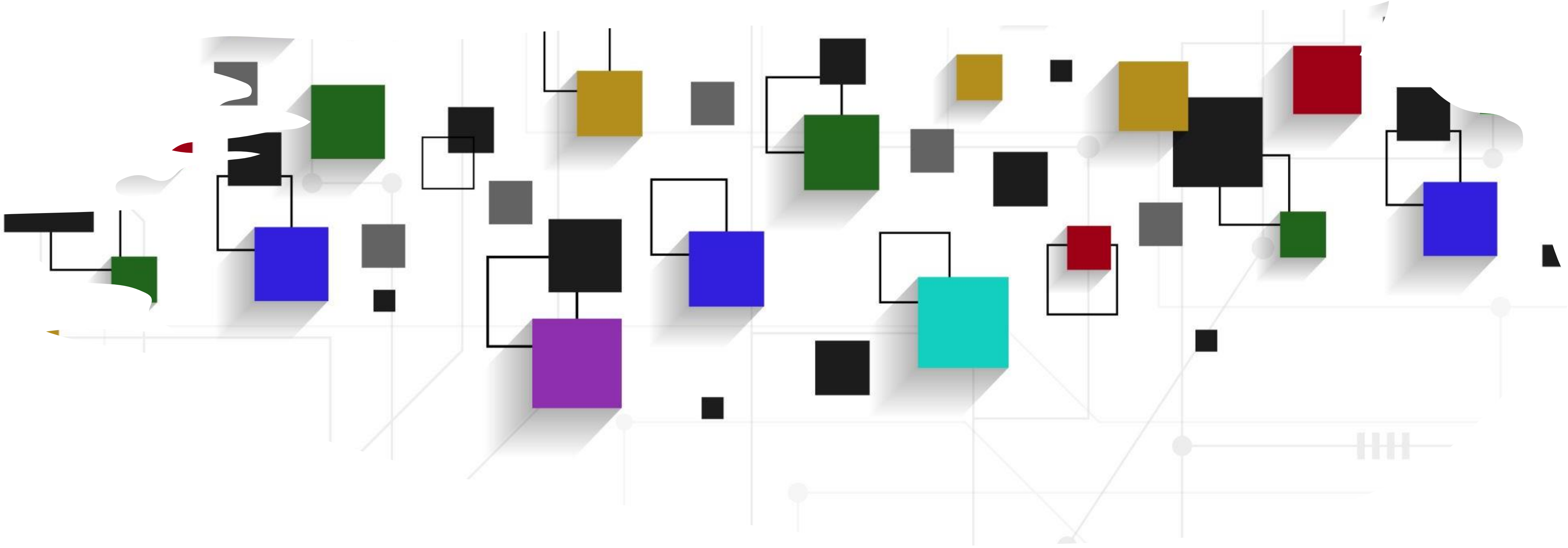
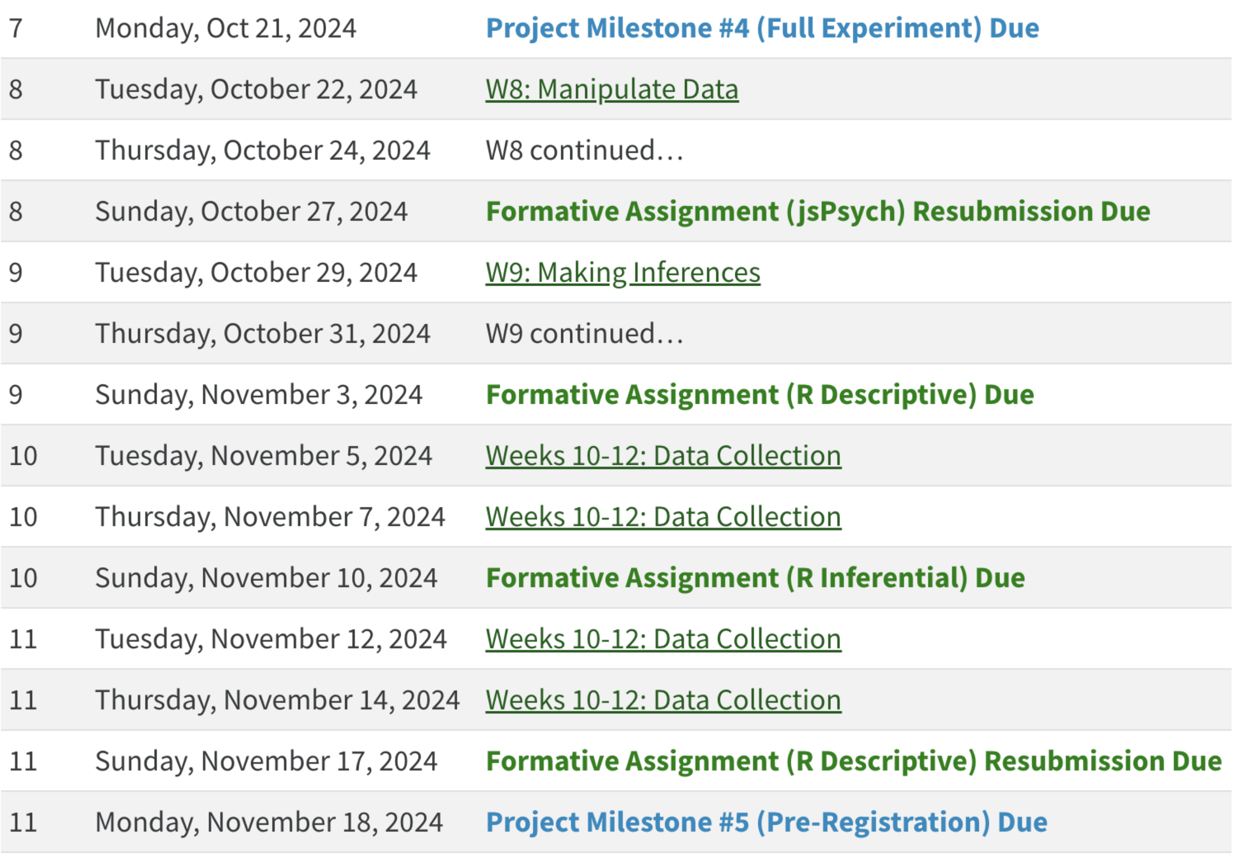


CogLab: Manipulate Data

WEEK 8

logistics: formative assignments

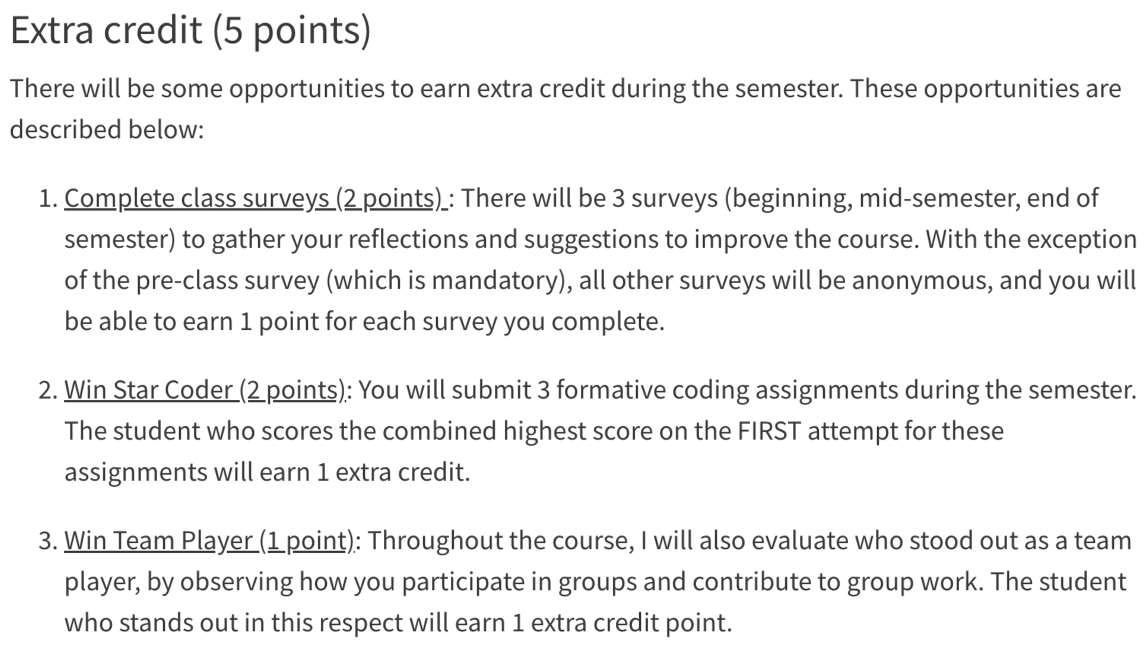
- formative assignment #1: resubmission due Sunday
- formative assignment #2: descriptive statistics and plotting in R
  - due Nov 3



7	Monday, Oct 21, 2024	**Project Milestone #4 (Full Experiment) Due**
8	Tuesday, October 22, 2024	W8: Manipulate Data
8	Thursday, October 24, 2024	W8 continued…
8	Sunday, October 27, 2024	**Formative Assignment (jsPsych) Resubmission Due**
9	Tuesday, October 29, 2024	W9: Making Inferences
9	Thursday, October 31, 2024	W9 continued…
9	Sunday, November 3, 2024	**Formative Assignment (R Descriptive) Due**
10	Tuesday, November 5, 2024	Weeks 10-12: Data Collection
10	Thursday, November 7, 2024	Weeks 10-12: Data Collection
10	Sunday, November 10, 2024	**Formative Assignment (R Inferential) Due**
11	Tuesday, November 12, 2024	Weeks 10-12: Data Collection
11	Thursday, November 14, 2024	Weeks 10-12: Data Collection
11	Sunday, November 17, 2024	**Formative Assignment (R Descriptive) Resubmission Due**
11	Monday, November 18, 2024	**Project Milestone #5 (Pre-Registration) Due**

mid-semester survey

- available on canvas + course website
- counts towards extra credit
- due Monday
- your feedback and reflections are really important!
- **anonymous**

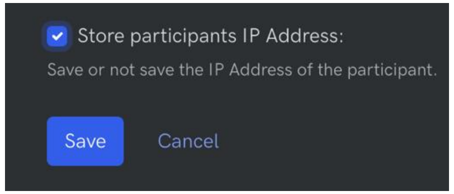


Extra credit (5 points)

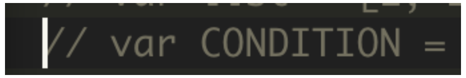
There will be some opportunities to earn extra credit during the semester. These opportunities are described below:

1. Complete class surveys (2 points) : There will be 3 surveys (beginning, mid-semester, end of semester) to gather your reflections and suggestions to improve the course. With the exception of the pre-class survey (which is mandatory), all other surveys will be anonymous, and you will be able to earn 1 point for each survey you complete.
2. Win Star Coder (2 points): You will submit 3 formative coding assignments during the semester. The student who scores the combined highest score on the FIRST attempt for these assignments will earn 1 extra credit.
3. Win Team Player (1 point): Throughout the course, I will also evaluate who stood out as a team player, by observing how you participate in groups and contribute to group work. The student who stands out in this respect will earn 1 extra credit point.

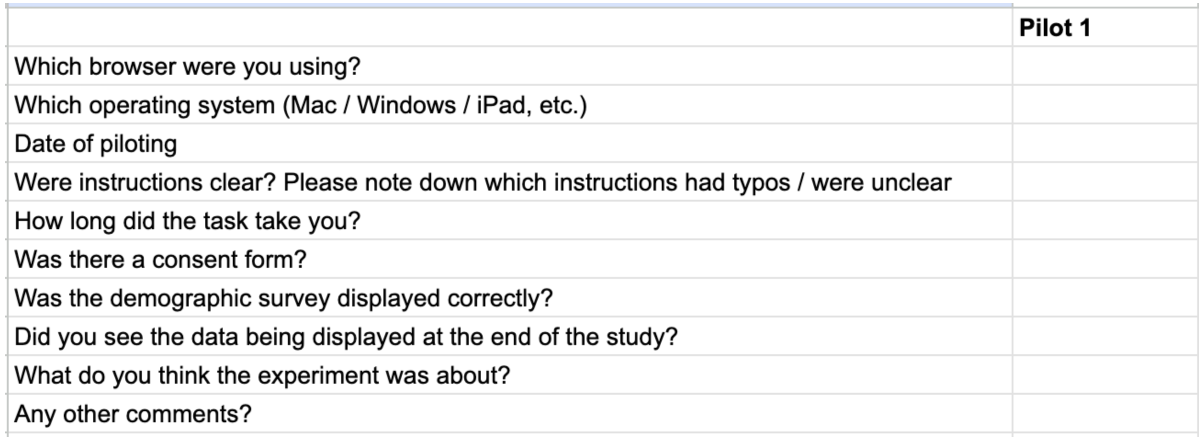
logistics: project


☑ Store participants IP Address:

Save or not save the IP Address of the participant.


```
// var CONDITION =
```

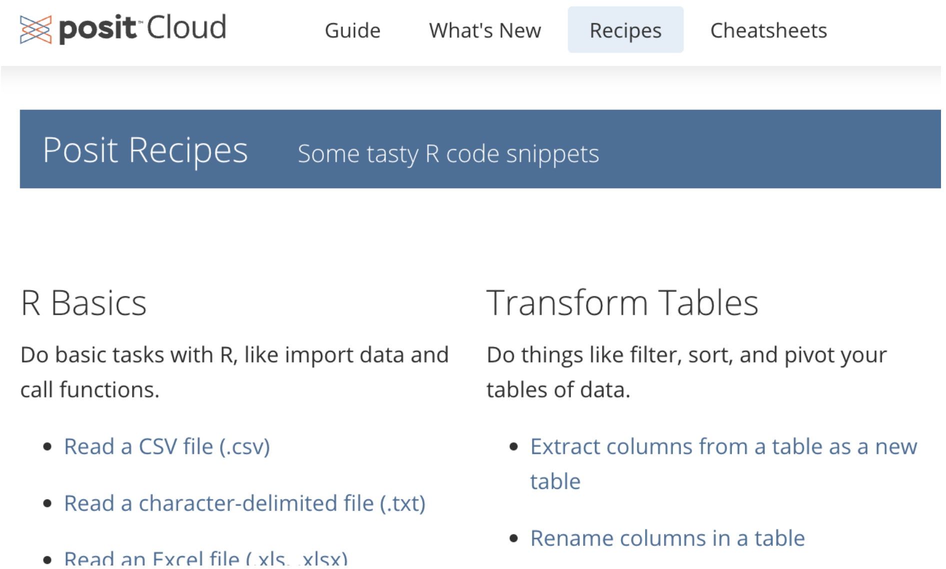
- next milestone #6:
  pre-registration (Nov 18: might move)
- **before** pre-registration:
  - providing accuracy feedback on priming trials
  - recording IP addresses
  - commenting the condition definition inside cognition.run
  - piloting your experiment (Uma + other group + 5 friends, N = 8), pilot feedback sheet
  - send cognition.run link by Nov 10
  - finalizing analysis plan + sample size



	Pilot 1
Which browser were you using?	
Which operating system (Mac / Windows / iPad, etc.)	
Date of piloting	
Were instructions clear? Please note down which instructions had typos / were unclear	
How long did the task take you?	
Was there a consent form?	
Was the demographic survey displayed correctly?	
Did you see the data being displayed at the end of the study?	
What do you think the experiment was about?	
Any other comments?	

recap

- what we covered:
  - R101, data analysis plan
  - visualizing data
- your to-do's were:
  - *apply:* project milestone 5 (full experiment)
  - *prep:* start Transform Tables recipes



Posit Recipes Some tasty R code snippets

R Basics

Do basic tasks with R, like import data and call functions.

- Read a CSV file (.csv)
- Read a character-delimited file (.txt)
- Read an Excel file (.xls, .xlsx)

Transform Tables

Do things like filter, sort, and pivot your tables of data.

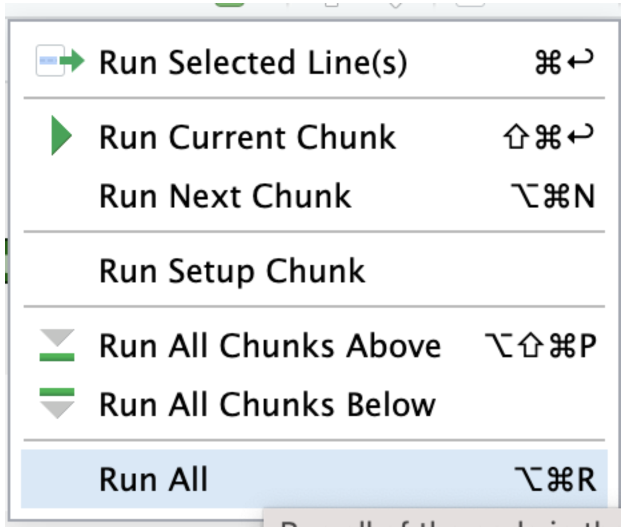
- Extract columns from a table as a new table
- Rename columns in a table

today’s agenda

- tidyverse verbs
  - select()
  - filter()
  - mutate()
  - summarize()
  - group_by()

open your RStudio project

- open the project and your .Rmd file
- run all chunks

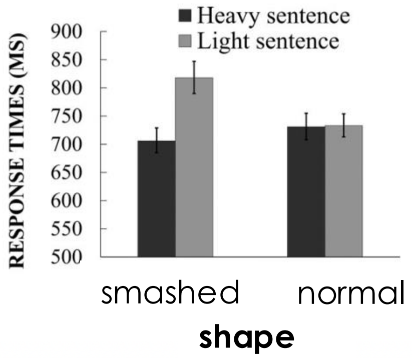


an experiment

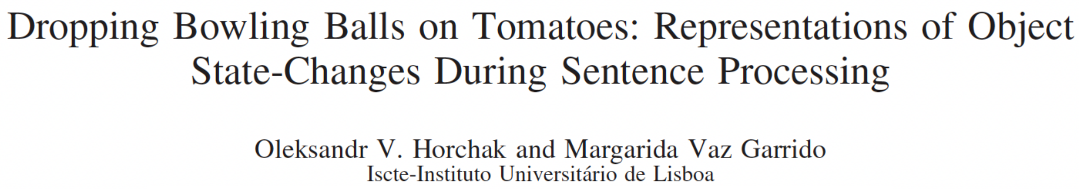
- I will show you a sentence
- then I will show you an image
- raise your dominant hand if the object shown was mentioned in the sentence
- raise your non dominant hand otherwise

you drop a bowling ball on a tomato

object state changes dataset

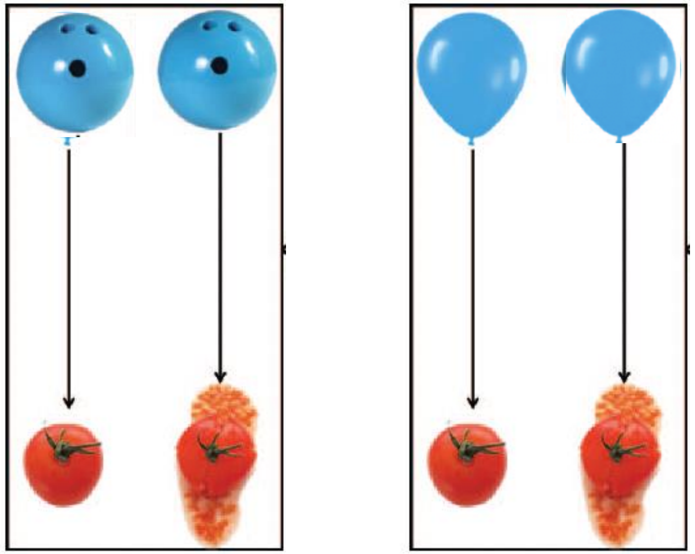


- task: object verification from sentences presented to participants
- research questions: do the events mentioned in the sentences influence response time?
- RT (bowling ball + squashed tomato) vs. RT (bowling ball + intact tomato)
- RT (balloon + squashed tomato) vs. RT (balloon + intact tomato)



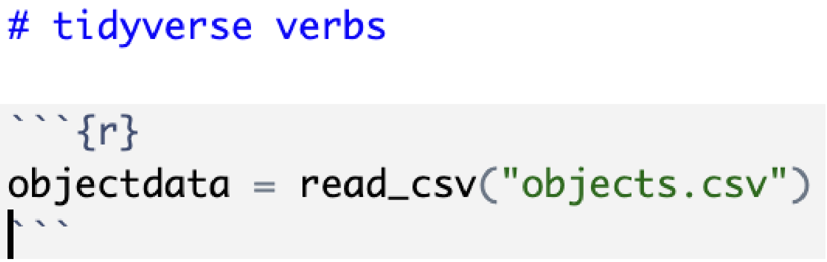
Dropping Bowling Balls on Tomatoes: Representations of Object State-Changes During Sentence Processing

Oleksandr V. Horchak and Margarida Vaz Garrido
Iscte-Instituto Universitário de Lisboa

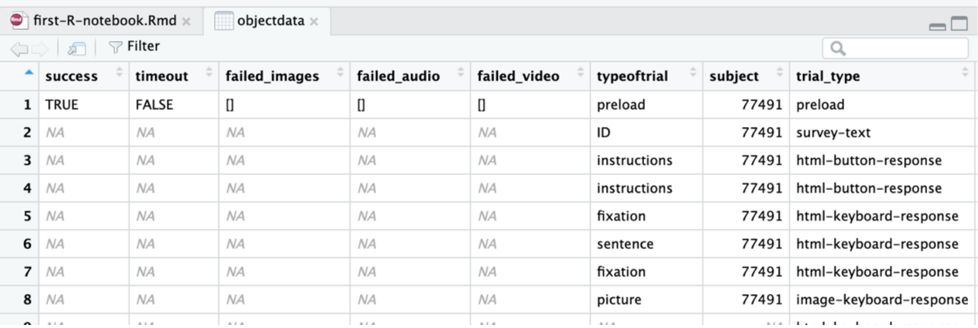


review: importing new data

- create a new a # tidyverse verbs heading and code chunk
- download objects.csv
- import this data into your notebook and name it objectdata
- how many rows and columns?



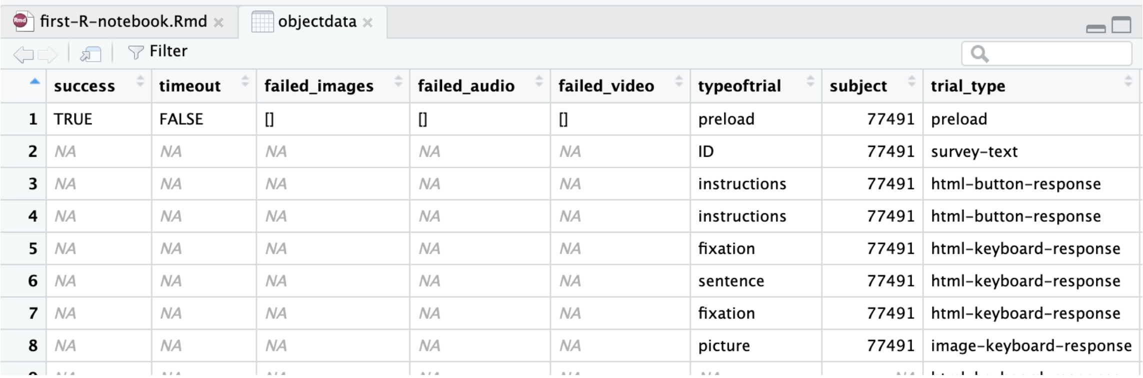
~~~
# tidyverse verbs

```{r}
objectdata = read_csv("objects.csv")
```
~~~



	success	timeout	failed_images	failed_audio	failed_video	typeoftrial	subject	trial_type
1	TRUE	FALSE	[]	[]	[]	preload	77491	preload
2	NA	NA	NA	NA	NA	ID	77491	survey-text
3	NA	NA	NA	NA	NA	instructions	77491	html-button-response
4	NA	NA	NA	NA	NA	instructions	77491	html-button-response
5	NA	NA	NA	NA	NA	fixation	77491	html-keyboard-response
6	NA	NA	NA	NA	NA	sentence	77491	html-keyboard-response
7	NA	NA	NA	NA	NA	fixation	77491	html-keyboard-response
8	NA	NA	NA	NA	NA	picture	77491	image-keyboard-response

tidyverse verbs

- often, your experiment data is not in analysis-ready format
- you may need to delete some rows, select some columns, arrange the data, etc.
- tidyverse verbs allow you to manipulate the dataframe and make it analysis and plotting-friendly



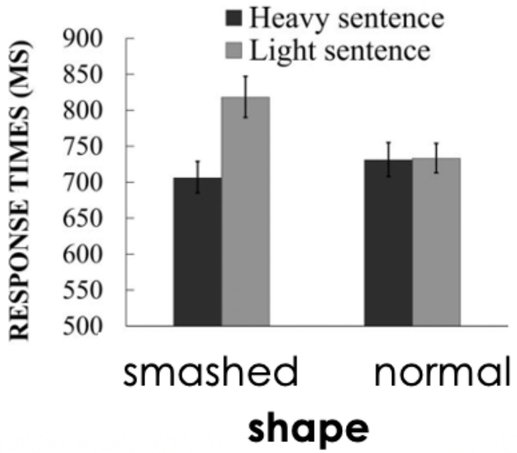
	success	timeout	failed_images	failed_audio	failed_video	typeoftrial	subject	trial_type
1	TRUE	FALSE	[]	[]	[]	preload	77491	preload
2	NA	NA	NA	NA	NA	ID	77491	survey-text
3	NA	NA	NA	NA	NA	instructions	77491	html-button-response
4	NA	NA	NA	NA	NA	instructions	77491	html-button-response
5	NA	NA	NA	NA	NA	fixation	77491	html-keyboard-response
6	NA	NA	NA	NA	NA	sentence	77491	html-keyboard-response
7	NA	NA	NA	NA	NA	fixation	77491	html-keyboard-response
8	NA	NA	NA	NA	NA	picture	77491	image-keyboard-response

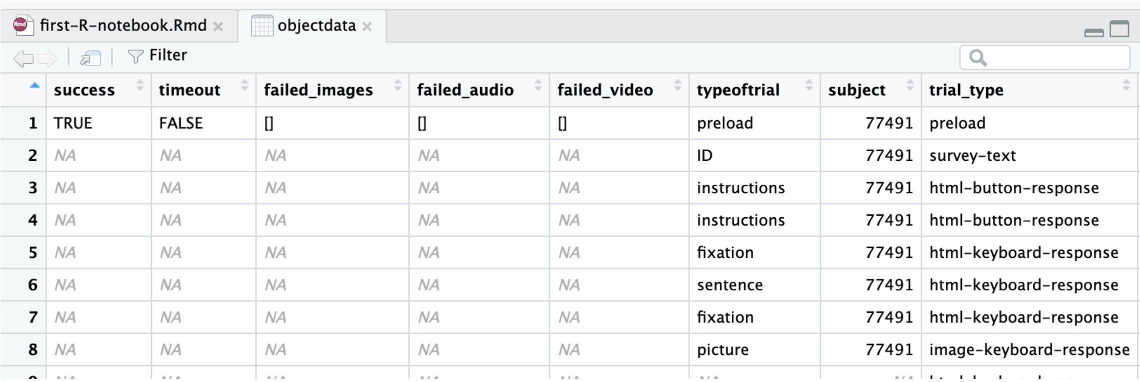
tidyverse piping

- piping is a way to define a sequence of operations in R
- this is accomplished using %>%
- the idea is that you use the same data but perform multiple operations on it using the pipe
- we will use piping to combine different operations together

tidyverse: select()

- select() allows you to retain only specific columns from your dataframe
- useful when your data contains too many unnecessary columns that are not relevant for analysis
- what columns might be important in this dataset?
- print the column names and let's make a list!

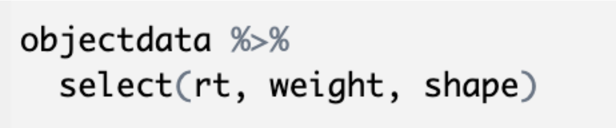




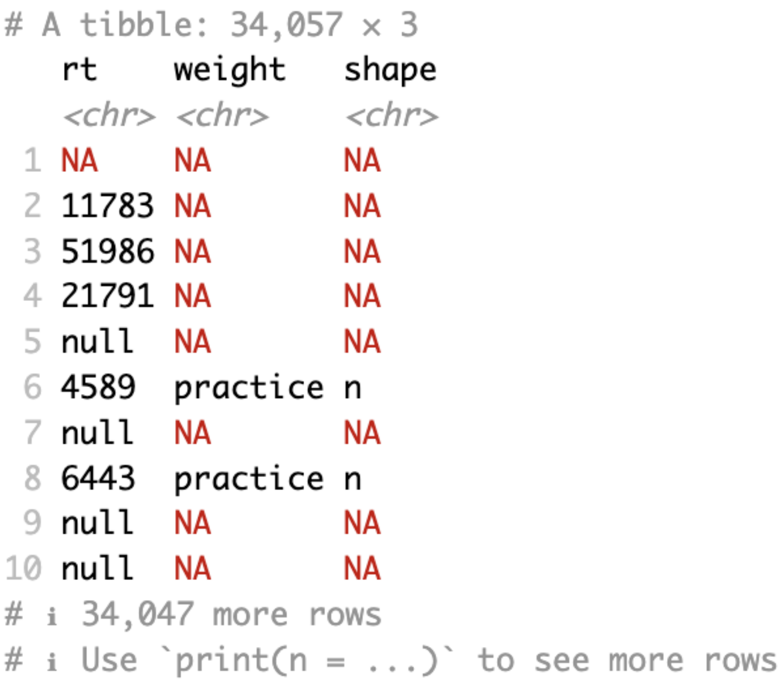
	success	timeout	failed_images	failed_audio	failed_video	typeoftrial	subject	trial_type
1	TRUE	FALSE	[]	[]	[]	preload	77491	preload
2	NA	NA	NA	NA	NA	ID	77491	survey-text
3	NA	NA	NA	NA	NA	instructions	77491	html-button-response
4	NA	NA	NA	NA	NA	instructions	77491	html-button-response
5	NA	NA	NA	NA	NA	fixation	77491	html-keyboard-response
6	NA	NA	NA	NA	NA	sentence	77491	html-keyboard-response
7	NA	NA	NA	NA	NA	fixation	77491	html-keyboard-response
8	NA	NA	NA	NA	NA	picture	77491	image-keyboard-response

tidyverse: select()

- logic of piping:
  - start with the dataset
  - add a pipe
  - specify an action
- select RT, weight, and shape from objectdata
- run the chunk
- what do you see?
- ALL trials are being included because select only picks the columns, not the rows



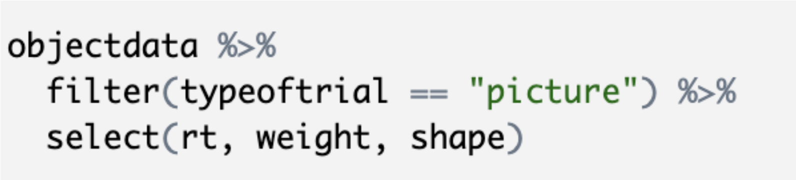
```
objectdata %>%
  select(rt, weight, shape)
```



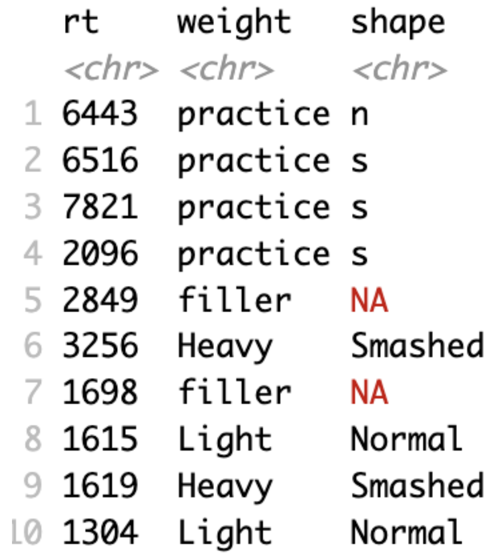
```
# A tibble: 34,057 × 3
   rt    weight   shape
   <chr> <chr>    <chr>
 1 NA    NA       NA
 2 11783 NA       NA
 3 51986 NA       NA
 4 21791 NA       NA
 5 null  NA       NA
 6 4589  practice n
 7 null  NA       NA
 8 6443  practice n
 9 null  NA       NA
10 null  NA       NA
# ℹ 34,047 more rows
# ℹ Use `print(n = ...)` to see more rows
```

tidyverse: filter()

- filter() allows you to retain only specific rows from your dataframe
- if we need only the picture trials, we can use filter to do this *before* we select our columns
- notice how we've used the pipe to continue our code
- run this chunk again!
- what do you notice now?



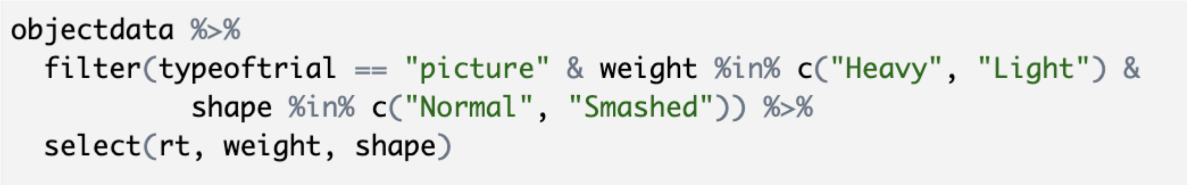
```
objectdata %>%
  filter(typeoftrial == "picture") %>%
  select(rt, weight, shape)
```



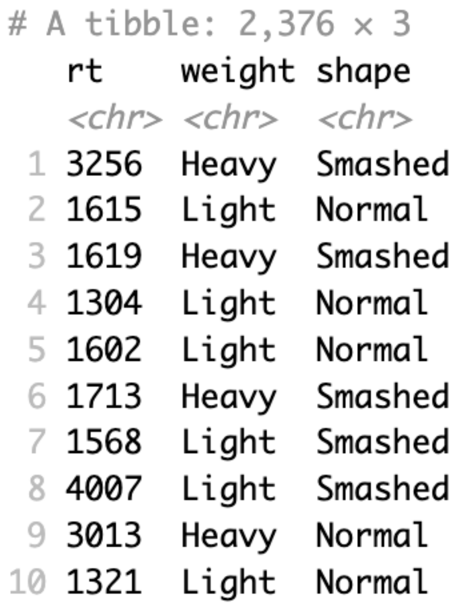
```
   rt    weight   shape
   <chr> <chr>    <chr>
 1 6443  practice n
 2 6516  practice s
 3 7821  practice s
 4 2096  practice s
 5 2849  filler   NA
 6 3256  Heavy    Smashed
 7 1698  filler   NA
 8 1615  Light    Normal
 9 1619  Heavy    Smashed
10 1304  Light    Normal
```

tidyverse: filter()

- the data is a lot better now but still contains filler and practice trials
- we could add an additional conditions in our filter statement that restrict the values of weight and shape
- the & operator combines different constraints we want to apply to the data



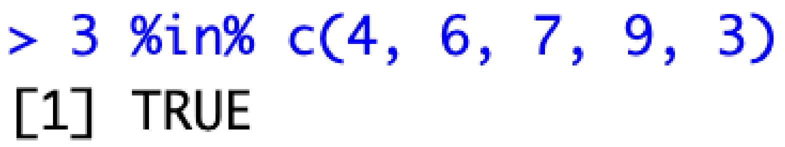
```
objectdata %>%
  filter(typeoftrial == "picture" & weight %in% c("Heavy", "Light") &
           shape %in% c("Normal", "Smashed")) %>%
  select(rt, weight, shape)
```



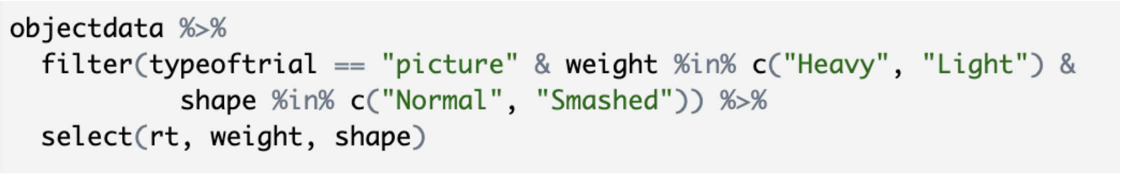
```
# A tibble: 2,376 × 3
   rt    weight shape
   <chr> <chr>  <chr>
 1 3256  Heavy  Smashed
 2 1615  Light  Normal
 3 1619  Heavy  Smashed
 4 1304  Light  Normal
 5 1602  Light  Normal
 6 1713  Heavy  Smashed
 7 1568  Light  Smashed
 8 4007  Light  Smashed
 9 3013  Heavy  Normal
10 1321  Light  Normal
```

tidyverse: %in%

- %in% is a useful tidyverse operator that checks whether an element belongs to a vector
- in your console: check if 3 is inside a vector containing 4, 6, 7, 9, 3
- each part of filter() is a condition being evaluated as TRUE or FALSE



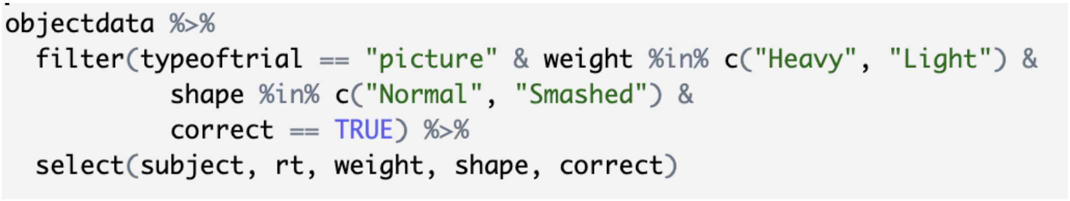
```
> 3 %in% c(4, 6, 7, 9, 3)
[1] TRUE
```



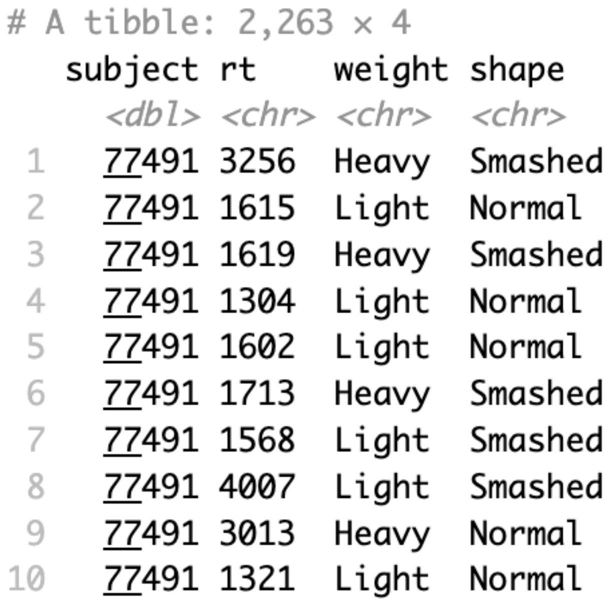
```
objectdata %>%
  filter(typeoftrial == "picture" & weight %in% c("Heavy", "Light") &
           shape %in% c("Normal", "Smashed")) %>%
  select(rt, weight, shape)
```

exercise: more constraints

- we want to evaluate only correct trials, use filter() to do this
- we want to retain the subject/participant identifier in the resulting dataframe: use select() to do this



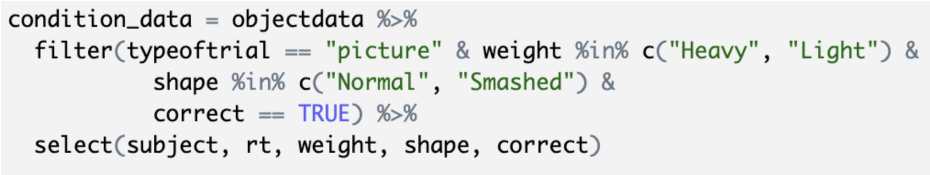
```
objectdata %>%
  filter(typeoftrial == "picture" & weight %in% c("Heavy", "Light") &
           shape %in% c("Normal", "Smashed") &
           correct == TRUE) %>%
  select(subject, rt, weight, shape, correct)
```



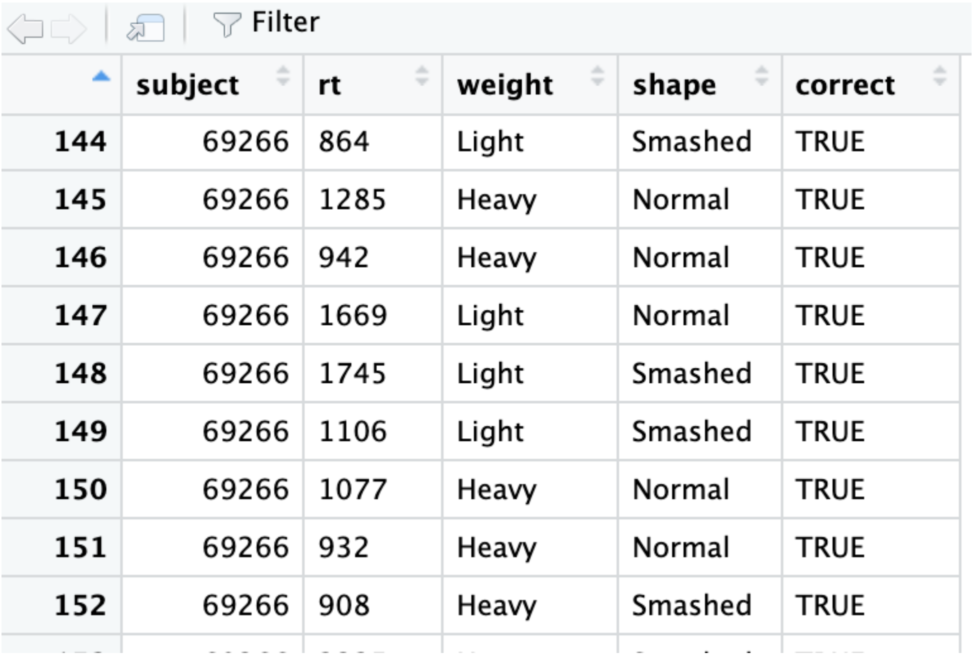
```
# A tibble: 2,263 × 4
   subject rt    weight shape  
     <dbl> <chr> <chr>  <chr>  
 1   77491 3256  Heavy  Smashed
 2   77491 1615  Light  Normal 
 3   77491 1619  Heavy  Smashed
 4   77491 1304  Light  Normal 
 5   77491 1602  Light  Normal 
 6   77491 1713  Heavy  Smashed
 7   77491 1568  Light  Smashed
 8   77491 4007  Light  Smashed
 9   77491 3013  Heavy  Normal 
10   77491 1321  Light  Normal 
```

storing filtered data

- we not only want to subset the data but also store it so that we can do more analyses on the data
- name the filtered data as condition_data
- this should create condition_data in the environment
- click and examine that data



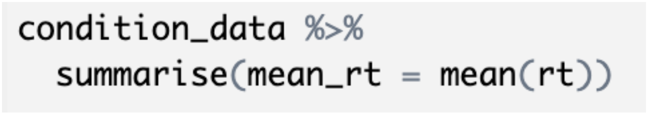
```
condition_data = objectdata %>%
  filter(typeoftrial == "picture" & weight %in% c("Heavy", "Light") &
           shape %in% c("Normal", "Smashed") &
           correct == TRUE) %>%
  select(subject, rt, weight, shape, correct)
```



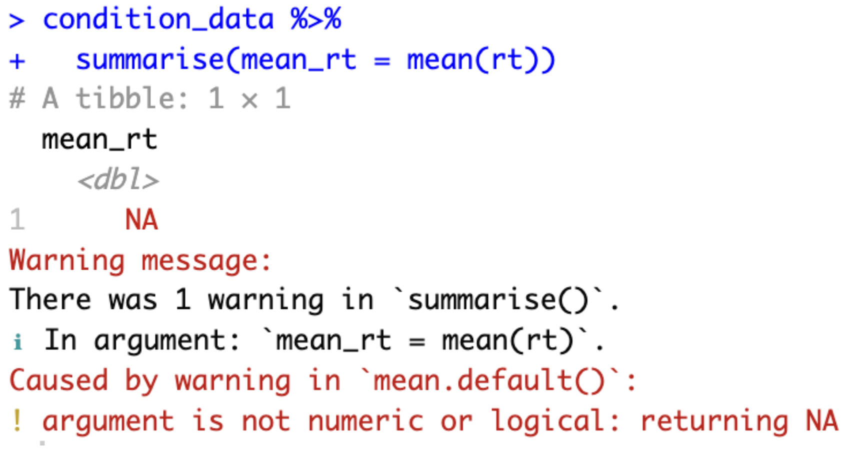
	subject	rt	weight	shape	correct
144	69266	864	Light	Smashed	TRUE
145	69266	1285	Heavy	Normal	TRUE
146	69266	942	Heavy	Normal	TRUE
147	69266	1669	Light	Normal	TRUE
148	69266	1745	Light	Smashed	TRUE
149	69266	1106	Light	Smashed	TRUE
150	69266	1077	Heavy	Normal	TRUE
151	69266	932	Heavy	Normal	TRUE
152	69266	908	Heavy	Smashed	TRUE

tidyverse: summarize()

- summarize() calculates descriptive statistics for your data
- we can compute the mean reaction time across all trials and all participants for condition_data
- NAs are produced when the mean cannot be computed



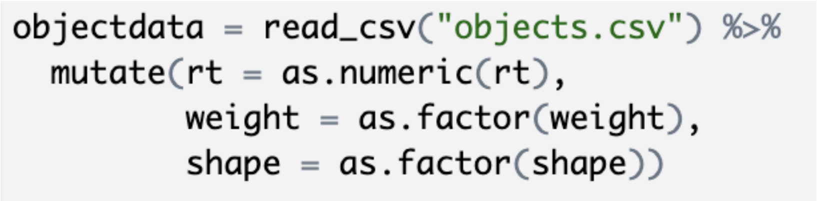
```
condition_data %>%
  summarise(mean_rt = mean(rt))
```



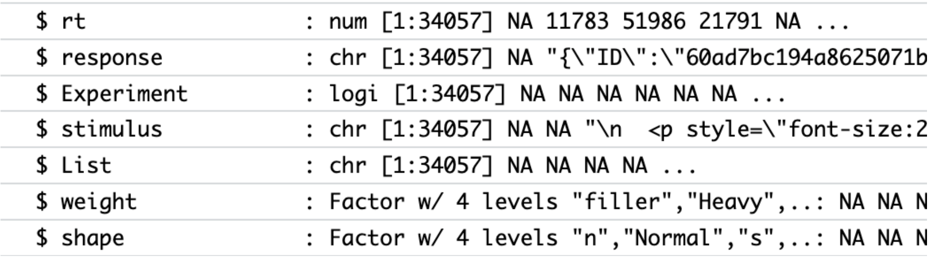
```
> condition_data %>%
+   summarise(mean_rt = mean(rt))
# A tibble: 1 × 1
  mean_rt
    <dbl>
1      NA
Warning message:
There was 1 warning in `summarise()`.
ℹ In argument: `mean_rt = mean(rt)`.
Caused by warning in `mean.default()`:
! argument is not numeric or logical: returning NA
```

tidyverse: mutate()

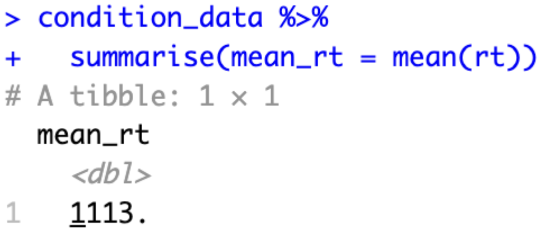
- mutate() allows you to create new columns in your dataframe or change/replace existing columns
- we can use mutate() to change the data type of important columns when we read in the object data
- re-run your chunk



```r
objectdata = read_csv("objects.csv") %>%
  mutate(rt = as.numeric(rt),
         weight = as.factor(weight),
         shape = as.factor(shape))
```



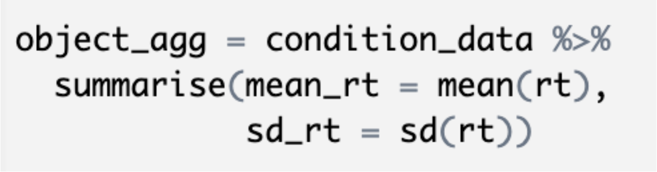
```
 $ rt                 : num [1:34057] NA 11783 51986 21791 NA ...
 $ response           : chr [1:34057] NA "{\"ID\":\"60ad7bc194a8625071b
 $ Experiment         : logi [1:34057] NA NA NA NA NA NA ...
 $ stimulus           : chr [1:34057] NA NA "\n  <p style=\"font-size:2
 $ List               : chr [1:34057] NA NA NA NA ...
 $ weight             : Factor w/ 4 levels "filler","Heavy",..: NA NA N
 $ shape              : Factor w/ 4 levels "n","Normal","s",..: NA NA N
```



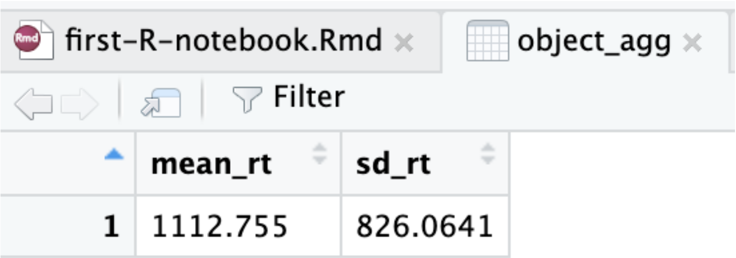
```
> condition_data %>%
+   summarise(mean_rt = mean(rt))
# A tibble: 1 × 1
  mean_rt
    <dbl>
1   1113.
```

tidyverse: more summarize()

- compute the standard deviation of reaction time
- store it all in a dataframe called object_agg



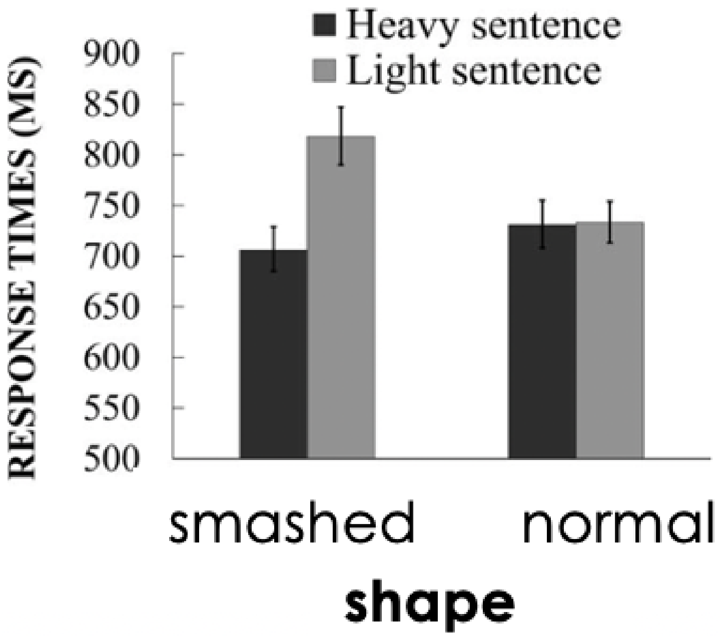
```
object_agg = condition_data %>%
  summarise(mean_rt = mean(rt),
            sd_rt = sd(rt))
```



	mean_rt	sd_rt
1	1112.755	826.0641

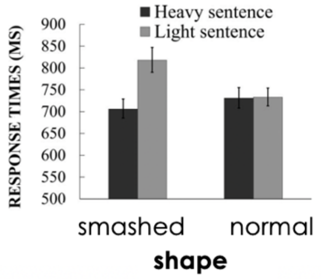
tidyverse: group_by()

- group_by() allows you to group the data based on specific values within a column
- if we want to obtain reaction times for our conditions, which columns should we use to group the data?

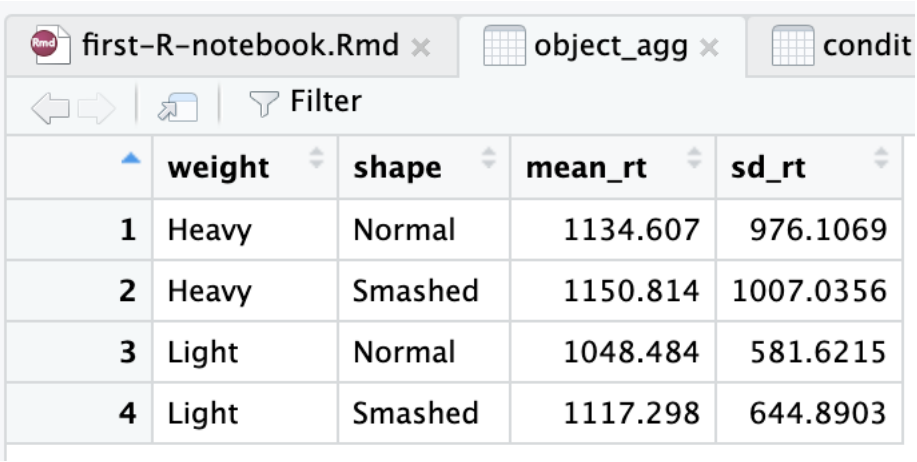


tidyverse: group_by()

- modify object_agg
- group by weight and shape
- compute the mean and sd of reaction time
- are we in business??



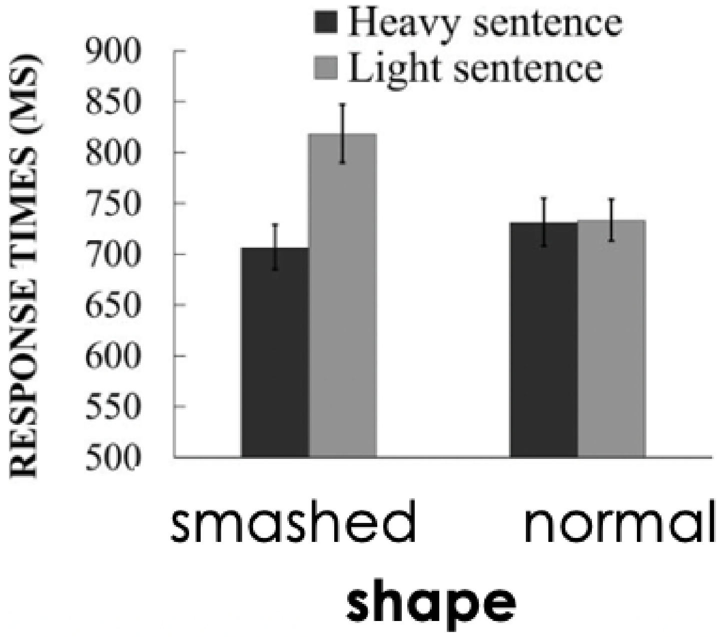
```
object_agg = condition_data %>%
  group_by(weight, shape) %>%
  summarise(mean_rt = mean(rt),
            sd_rt = sd(rt))
```



	weight	shape	mean_rt	sd_rt
1	Heavy	Normal	1134.607	976.1069
2	Heavy	Smashed	1150.814	1007.0356
3	Light	Normal	1048.484	581.6215
4	Light	Smashed	1117.298	644.8903

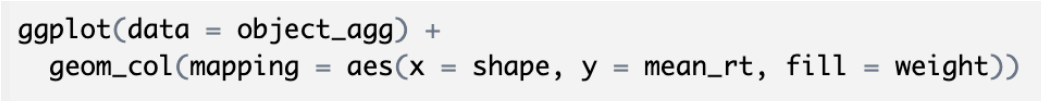
we're in business!

- we can now plot the means using our favorite plotting function
- recall the grammar of graphics…what 3 things do we need?
- data?
- geom?
- mapping/aes?

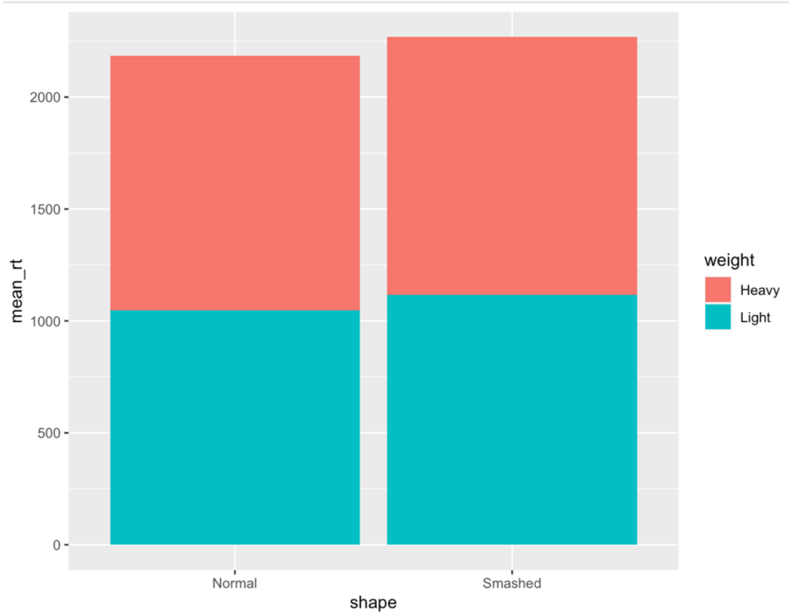


plotting the means

- use ggplot() to plot the data
- notice the + sign, not %>% for plotting
- notice the fill is inside the aes() because it is a column from the data
- close…?

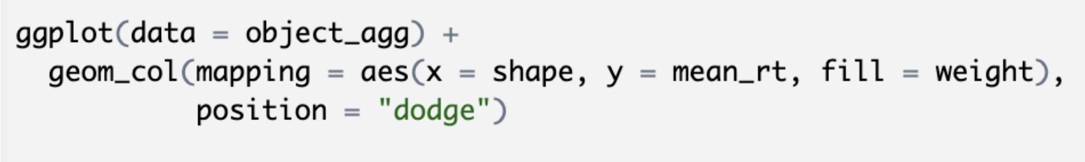


```
ggplot(data = object_agg) +
  geom_col(mapping = aes(x = shape, y = mean_rt, fill = weight))
```

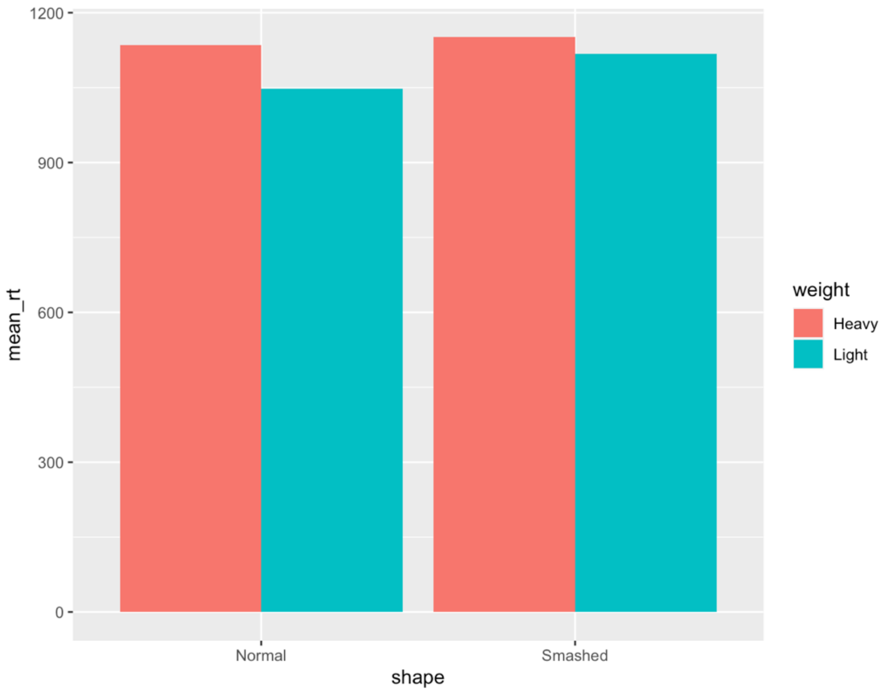


stacked vs. unstacked plots

- stacked bar charts display the grouped data on top of each other
- unstacked bar charts separate the bars
- use position = “dodge” inside geom_col(), after the mapping

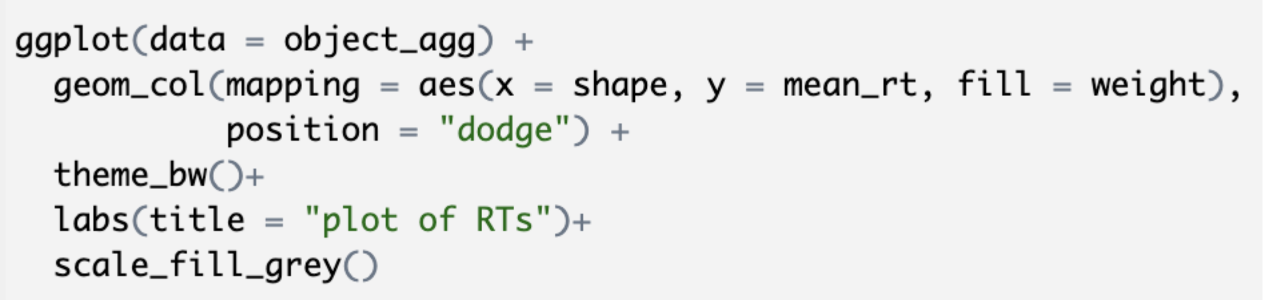


```
ggplot(data = object_agg) +
  geom_col(mapping = aes(x = shape, y = mean_rt, fill = weight),
           position = "dodge")
```

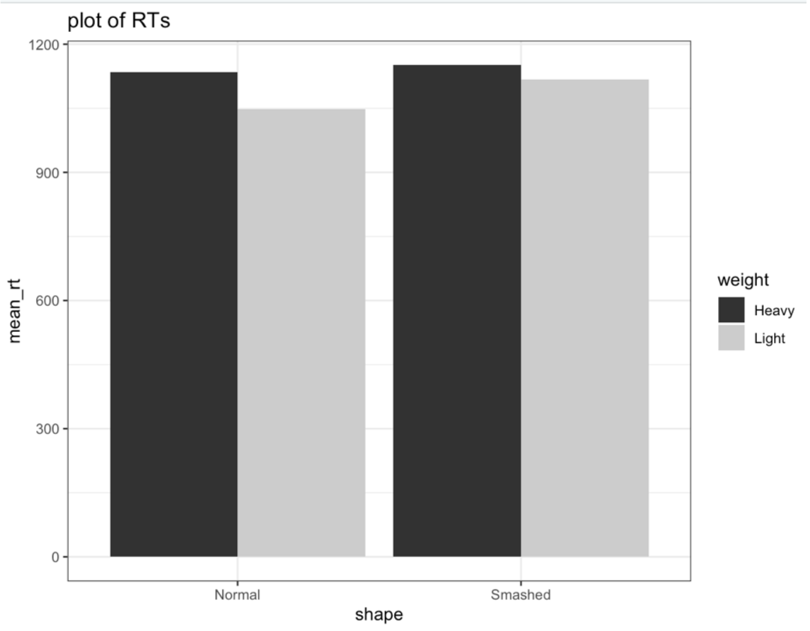


prettify your plot!

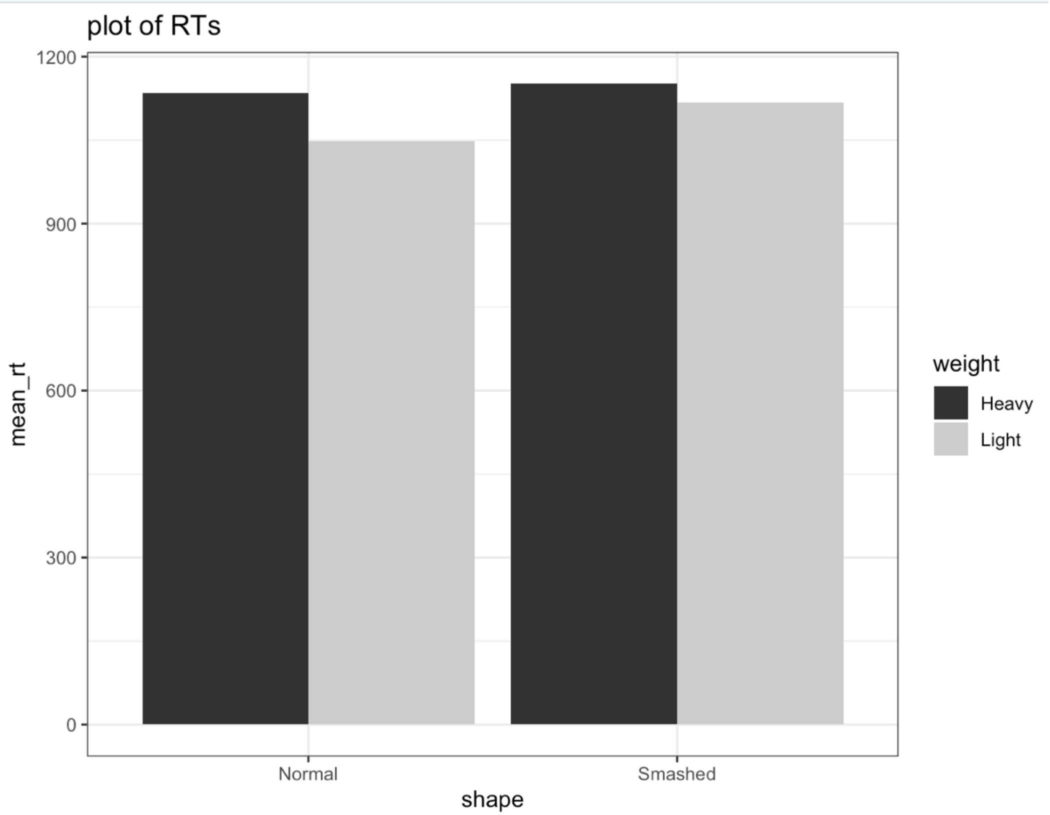
- add a theme
- add a title
- change color palette
- if aesthetics focus on filling, then use scale_fill_ otherwise use scale_color_

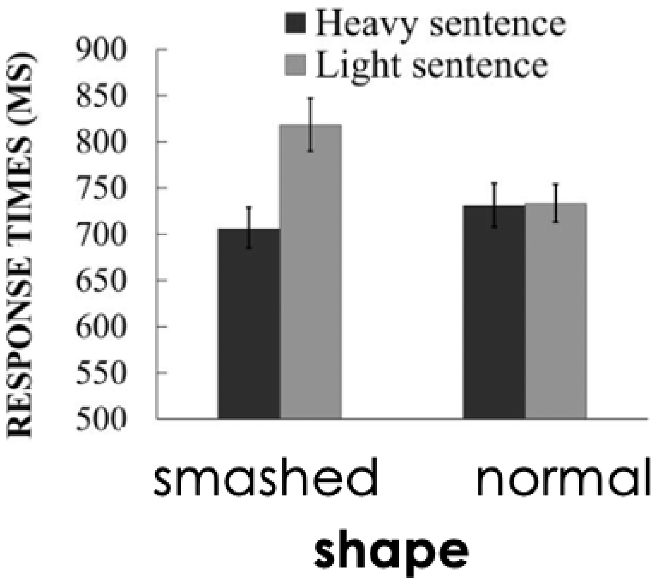


```r
ggplot(data = object_agg) +
  geom_col(mapping = aes(x = shape, y = mean_rt, fill = weight),
           position = "dodge") +
  theme_bw()+
  labs(title = "plot of RTs")+
  scale_fill_grey()
```



interpreting the plot





HW: exercises

- what if I wanted RTs for each condition for each participant?
- before I analyzed the RTs, what if I wanted to first filter out participants who failed an attention check?

next class

- **before** class
  - *brainstorm:* group project code (accuracy feedback)
  - *complete*: formative assignment #1 resubmission
  - *prep:* Transform Tables recipes
- **during** class
  - more data wrangling (for your experiments)