Introduction to R – Transforming data, creating graphics

Jana Jarecki, Stephan Lewandowsky

11 July 2022

Agenda: What today teaches or repeats

Data Reading

- How to read data
- How to change column classes

Data Management

- How to manipulate strings
- How to transform data
- When summarize data by groups

Data Visualization

- The principles of graphics (*ggplot*)
- Two useful plots for modelers

A large package collection

The `tidyverse` collection

- A package collection, installing which downloads currently 25 packages at once (like MS Office that comes with Word, Excel, Powerpoint, etc.)
- `library(tidyverse)`, as of today, calls `library` for eight of those packages:
  - readr to read and write data
  - tibble to represent data.frames easier
  - dplyr to manipulate data e.g. long-to-wide and vice versa
  - ggplot2 to plot graphic
  - purr to apply functions
  - tidyr to tidy data
  - stringr to work with strings
  - forcats to deal with factors

R Tip: Reproducible Package Management

A usability problem

- `library(aaa)` can load package *aaa* only if it is installed
- Poor usability for you who want to execute my script
- I want a command that
  - checks: are package installed? not, installs them
  - and: Loads packages

A solution *pacman*

- `pacman::p_load(aaa, bbb)` checks: are packages *aaa*, *bbb*, installed? If not, installs them
- And: Loads the packages automatically

```
# delete this line afterwards!
install.packages("pacman")
# pacman::  calls a function in pacman
pacman::p_load(tidyverse)
```

Note the notation `pacman::`

Reading data

Reading data with `readr()`

Setup

```
pacman::p_load(readr)
setwd(dir=dirname(rstudioapi::getSourceEditorContext()$path))
```

Data Reading

Read the data *ggplot-data.csv* with base R's command `read.csv()` and the tidyr's command `read_csv()`, what are the differences?

Reading Data

```
setwd(dir=dirname(rstudioapi::getSourceEditorContext()$path))
a <- read.csv("data/ggplot-data.csv")
a
```

```
##     id condition    judgment  certainty choice model_X_pred model_Y_pred
## 1    1         A  5.11287534  2.6285737      0    0.4540471    0.5397072
## 2    2         A  0.30590549  1.1001238      1    0.7245752    0.3362662
## 3    3         A  2.08938523  2.5760403      1    0.5282888    0.1851453
## 4    4         A  3.89858781  1.4530127      1    0.5845568    0.2678840
## 5    5         A  2.21280497  0.4742799      1    0.7198482    0.4388933
## 6    6         A  1.68162645  2.1691433      1    0.6927926    0.6610837
## 7    7         A  5.53456599  1.6955200      1    0.5637879    0.3150401
## 8    8         A  1.71602288  3.1873058      1    0.5598316    0.6052043
## 9    9         A  7.05527114  2.3796080      1    0.6997092    0.5334455
## 10 10         A  1.81185770  2.4180194      1    0.8762260    0.5504416
## 11 11         A  4.91460896  2.7048471      1    0.6336925    0.6543931
## 12 12         A  8.85993618  2.3881483      1    0.5753737    0.1014269
## 13 13         A -3.16658210  2.3424111      1    0.6078865    0.1962579
## 14 14         A  1.16363370  2.2756260      1    0.7122976    0.3395638
## 15 15         A  0.60003599  1.6097678      1    0.5085801    0.5051917
## 16 16         A  3.90785119  2.4581210      1    0.8195041    0.3882232
## 17 17         A  0.14724124  2.1087371      0    0.3298549    0.6271167
## 18 18         A -5.96936626  0.2959596      1    0.7159193    0.2901701
## 19 19         A -6.32140079 -2.8573702      1    0.7284936    0.5888546
```

Reading Data

```
pacman::p_load(readr)
setwd(dir=dirname(rstudioapi::getSourceEditorContext()$path))
d <- read_csv("data/ggplot-data.csv")
d
```

```
## # A tibble: 40 x 11
##       id condition judgment certainty choice model_X_pred model_Y_pred
##    <dbl> <chr>          <dbl>     <dbl>  <dbl>        <dbl>        <dbl>
##  1     1 A             5.11      2.63      0        0.454        0.540
##  2     2 A             0.306     1.10      1        0.725        0.336
##  3     3 A             2.09      2.58      1        0.528        0.185
##  4     4 A             3.90      1.45      1        0.585        0.268
##  5     5 A             2.21      0.474     1        0.720        0.439
##  6     6 A             1.68      2.17      1        0.693        0.661
##  7     7 A             5.53      1.70      1        0.564        0.315
##  8     8 A             1.72      3.19      1        0.560        0.605
##  9     9 A             7.06      2.38      1        0.700        0.533
## 10    10 A             1.81      2.42      1        0.876        0.550
## # ... with 30 more rows, and 4 more variables: model_X_evidence <dbl>,
## #   model_Y_evidence <dbl>, alpha <dbl>, delta <dbl>
```

Reading Data: Column Types

```
# subject "id" are in 99% of all cognitive data not meaningful numbers but IDs
d <- read_csv("data/ggplot-data.csv",
              col_type = cols(
                id = col_character()))
d
```

```
## # A tibble: 40 x 11
##    id    condition judgment certainty choice model_X_pred model_Y_pred
##    <chr> <chr>        <dbl>     <dbl>  <dbl>        <dbl>        <dbl>
##  1 1     A            5.11      2.63       0        0.454        0.540
##  2 2     A            0.306     1.10       1        0.725        0.336
##  3 3     A            2.09      2.58       1        0.528        0.185
##  4 4     A            3.90      1.45       1        0.585        0.268
##  5 5     A            2.21      0.474      1        0.720        0.439
##  6 6     A            1.68      2.17       1        0.693        0.661
##  7 7     A            5.53      1.70       1        0.564        0.315
##  8 8     A            1.72      3.19       1        0.560        0.605
##  9 9     A            7.06      2.38       1        0.700        0.533
## 10 10    A            1.81      2.42       1        0.876        0.550
## # ... with 30 more rows, and 4 more variables: model_X_evidence <dbl>,
## #   model_Y_evidence <dbl>, alpha <dbl>, delta <dbl>
```

String Manipulation

String manipulation

```
s <- c("ab_1", "xy_1", "uv_1")
print(s)
```

```
## [1] "ab_1" "xy_1" "uv_1"
```

```
pacman::p_load(stringr)
str_flatten(s)
```

```
## [1] "ab_1xy_1uv_1"
```

```
str_split_fixed(s, pattern="_", n=2)
```

```
##      [,1] [,2]
## [1,] "ab" "1"
## [2,] "xy" "1"
## [3,] "uv" "1"
```

```
d
```

```
## # A tibble: 40 x 11
##    id    condition judgment certainty choice model_X_pred model_Y_pred
```

Exersise: A Small Meaningless String Manipulation

- Make a new variable in the data `d` that holds the condition and choice in one string.
- (Optional) Substitute the pseudo-numeric `id` variable in `d` by a string of randomly selected 3 characters such as “nwt” for each subject.

```
pacman::p_load(stringr)
str_flatten(s)
str_split_fixed(s, pattern="_", n=2)
```

R Tip: De-numerize your IDs

Problem:

- If I don’t transform the ID to a character, it can be summed up, averaged, etc which makes no sense
- Code can assume ID = 1.5 can exis

Solution

- In my preprocessing I de-numerize IDs

```
pacman::p_load(stringr)
# letters is an inbuilt
# R object
d$id <- replicate(
  nrow(d),
  str_flatten(sample(letters, 3)))
```

Note an inbuilt vector: `letters`

Selecting and Transforming

Data transformations

Selecting parts of data

```
pacman::p_load(dplyr)
filter(d, id=="9")
```

```
## # A tibble: 1 x 11
##   id    condition judgment certainty choice model_X_pred model_Y_pred
##   <chr> <chr>        <dbl>     <dbl>  <dbl>        <dbl>        <dbl>
## 1 9     A             7.06      2.38      1        0.700        0.533
## # ... with 4 more variables: model_X_evidence <dbl>, model_Y_evidence <dbl>,
## #   alpha <dbl>, delta <dbl>
```

```
filter(d, condition=="A", judgment > 6)
```

```
## # A tibble: 2 x 11
##   id    condition judgment certainty choice model_X_pred model_Y_pred
##   <chr> <chr>        <dbl>     <dbl>  <dbl>        <dbl>        <dbl>
## 1 9     A             7.06      2.38      1        0.700        0.533
## 2 12    A             8.86      2.39      1        0.575        0.101
## # ... with 4 more variables: model_X_evidence <dbl>, model_Y_evidence <dbl>,
## #   alpha <dbl>, delta <dbl>
```

```
select(d, model_X_pred)
```

Transformation

Transforming data: wide to long

```
select(d, id, model_X_pred, model_Y_pred)[1:3,]
```

```
## # A tibble: 3 x 3
##   id    model_X_pred model_Y_pred
##   <chr>        <dbl>        <dbl>
## 1 1            0.454        0.540
## 2 2            0.725        0.336
## 3 3            0.528        0.185
```

We need a *long* format of this data:

```
## # A tibble: 80 x 3
##    id    model         pred
##    <chr> <chr>        <dbl>
##  1 1     model_X_pred 0.454
##  2 1     model_Y_pred 0.540
##  3 2     model_X_pred 0.725
##  4 2     model_Y_pred 0.336
##  5 3     model_X_pred 0.528
##  6 3     model_Y_pred 0.185
##  7 4     model_X_pred 0.585
##  8 4     model_Y_pred 0.268
##  9 5     model_X_pred 0.720
```

Transforming data: wide to long

```
pacman::p_load(tidyr)
dl <- pivot_longer(
  data = d,
  cols=contains("pred"),
  names_to = "model",
  values_to = "pred")
select(dl, id, model, pred, choice)
```

```
## # A tibble: 80 x 4
##    id    model         pred choice
##    <chr> <chr>        <dbl>  <dbl>
##  1 1     model_X_pred 0.454      0
##  2 1     model_Y_pred 0.540      0
##  3 2     model_X_pred 0.725      1
##  4 2     model_Y_pred 0.336      1
##  5 3     model_X_pred 0.528      1
##  6 3     model_Y_pred 0.185      1
##  7 4     model_X_pred 0.585      1
##  8 4     model_Y_pred 0.268      1
##  9 5     model_X_pred 0.720      1
## 10 5     model_Y_pred 0.439      1
## # ... with 70 more rows
```

Transforming data: long to wide

```
dw <- pivot_wider(
  data = dl,
  names_from = model,
  values_from = pred)
select(dw, id, model_X_pred, model_Y_pred)
```

```
## # A tibble: 40 x 3
##    id    model_X_pred model_Y_pred
##    <chr>        <dbl>        <dbl>
##  1 1            0.454        0.540
##  2 2            0.725        0.336
##  3 3            0.528        0.185
##  4 4            0.585        0.268
##  5 5            0.720        0.439
##  6 6            0.693        0.661
##  7 7            0.564        0.315
##  8 8            0.560        0.605
##  9 9            0.700        0.533
## 10 10           0.876        0.550
## # ... with 30 more rows
```

Transforming data: summary by groups

- *M* judgment by condition?
- *SD* of judgments by condition?
- *SE* of judgments by condition?

```
d[1:5]
```

```
## # A tibble: 40 x 5
##    id    condition judgment certainty choice
##    <chr> <chr>        <dbl>     <dbl>  <dbl>
##  1 1     A            5.11      2.63       0
##  2 2     A            0.306     1.10       1
##  3 3     A            2.09      2.58       1
##  4 4     A            3.90      1.45       1
##  5 5     A            2.21      0.474      1
##  6 6     A            1.68      2.17       1
##  7 7     A            5.53      1.70       1
##  8 8     A            1.72      3.19       1
##  9 9     A            7.06      2.38       1
## 10 10    A            1.81      2.42       1
## # ... with 30 more rows
```

Exersise: summary by groups

- Compute the *M* and *SE* of the `choice` for each `condition` and store it as `d_obs_msd`
- Compute the *M* of the predictions for each `condition` and for each model and store it in `d_pred`

```r
# Code reminder
dg <- group_by(d, condition)
d_m <- summarise(dg, across(
    .cols="judgment",
    .fns=list(M=mean)))
```

Two Dialects in the R Language

This code is the same …

```r
dg <- group_by(d, condition)
d_m <- summarise(dg, across(
    .cols="judgment",
    .fns=list(M=mean)))
d_m
```

```
## # A tibble: 2 x 2
##   condition judgment_M
##   <chr>          <dbl>
## 1 A              2.08
## 2 B              0.687
```

… as this code

```r
d_m <- d %>%
  group_by(condition) %>%
  summarise(across(
    .cols="judgment",
    .fns=list(M=mean)))
d_m
```

```
## # A tibble: 2 x 2
##   condition judgment_M
##   <chr>          <dbl>
## 1 A              2.08
## 2 B              0.687
```

Note the notation %>%

Two Dialects in the R Language

```
a %>% f(b)
# will execute
f(a, b)
```

`%>%` passes what’s left of it automatically as **first argument** to the function right of it, before any arguments that we give the function

In this case `a` becomes the first argument of `f`

- `%>%` is called a pipe
- comes from the *magrittr* package by Stefan Milton Bache

Plotting

Plotting with *ggplot2*

- *ggplot2* is a powerful and flexible package for making and saving graphics by Hadley Wickham (and others)

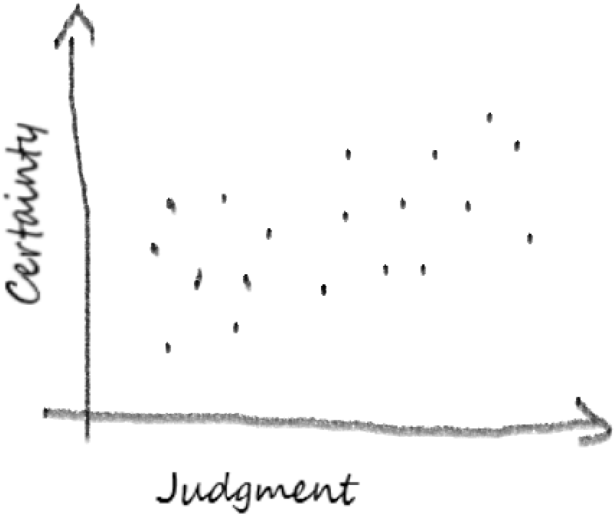
- it follows a quite straightforward logic

- it produces (almost) any type of static and animated 2D graphics

```r
setwd(dir=dirname(rstudioapi::getSourceEditorContext()$path))
pacman::p_load(ggplot2) # install.package(pacman)
```

Logic of Graphics

In principle, a plot …



… involves three questions

- *point*, *line*, *bar*, *boxplot*, *errorbar*, etc.
  Which geometric objects?

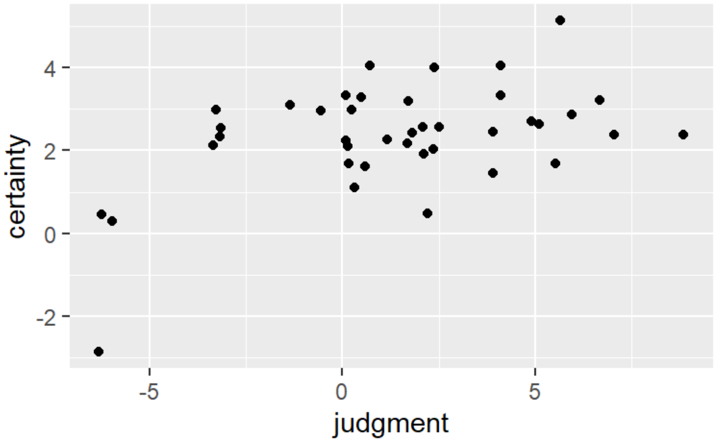
- *x-axis* = ?, *y-axis* = ?, *colors* = ?, etc.
  Which aesthetic dimensions map which variables?

- *title*, *color scales*, *axis texts*, …
  E.g., color-scale/theme adjustments that don't map variables

A Simple Graphic

Implemented in *ggplot*

```
d <- read.csv("data/ggplot-data.csv")

ggplot(data=d) +
  geom_point(mapping=aes(
    x=judgment,
    y=certainty))
```



… involves three questions

- *point*, *line*, *bar*, *boxplot*, *errorbar*, etc.
  Which geometric objects?
- *x-axis* = ?, *y-axis* = ?, *colors* = ?, etc.
  Which aesthetic dimensions map which variables?
- *title*, *color scales*, *axis texts*, …
  Layout adjustments that do not map any variables

Exersise: Some Extra Simple Plots

- Plot points with the `condition` on the x-axis and `judgment` on the y-axis
- Plot boxplots of `judgment` with the `condition` on the x-axis

```
head(d, n=2)
```

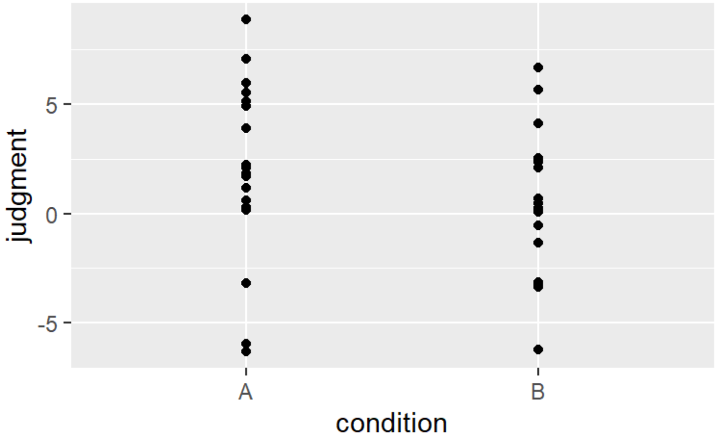
```
##   id condition  judgment certainty choice model_X_pred model_Y_pred
## 1  1         A 5.1128753  2.628574      0    0.4540471    0.5397072
## 2  2         A 0.3059055  1.100124      1    0.7245752    0.3362662
##   model_X_evidence model_Y_evidence     alpha    delta
## 1       0.13571107        0.8642889 0.4244226 4.582245
## 2       0.09960571        0.9003943 0.3057628 1.385302
```

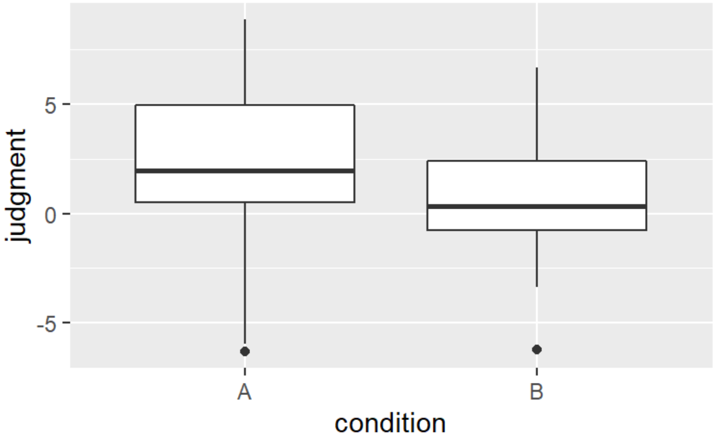
- (Advanced, optional) Plot the mean judgment by condition as single points. Add errorbars representing the SD to the plot. Make the errorbar whisker smaller. Plot bars of `choice` on the x-axis, and check what exactly do you see? Try out `geom_dotplot(x = judgment)`.

Solution Some Extra Simple Plots

```
ggplot(data=d) +
  geom_point(mapping=aes(
    x=condition,
    y=judgment))
```

```
ggplot(data=d) +
  geom_boxplot(mapping=aes(
    x=condition,
    y=judgment))
```





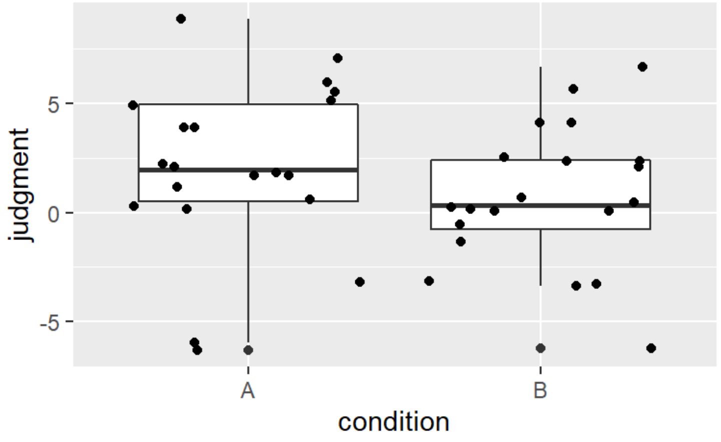
Exersise: Some More Simple Plots

- Add `+geom_jitter(aes(x=condition, y=judgment))` with a `+` to the boxplot, see what it does

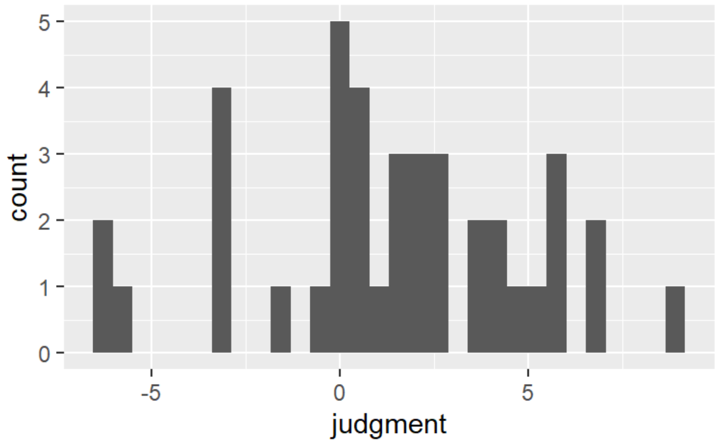
- Use a histogram as geometric object to show the distribution of `judgment`

Solution: Some Extra Simple Plots

```
ggplot(data=d) +
  geom_boxplot(mapping=aes(
    x=condition,
    y=judgment)) +
  geom_jitter(mapping=aes(
    x=condition,
    y=judgment))
```



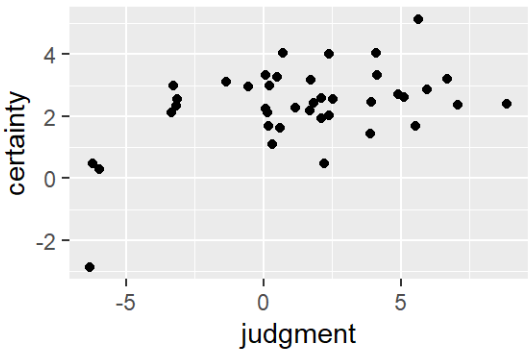
```
ggplot(data=d) +
  geom_histogram(mapping=aes(
    x=judgment))
```



Changes the overall look

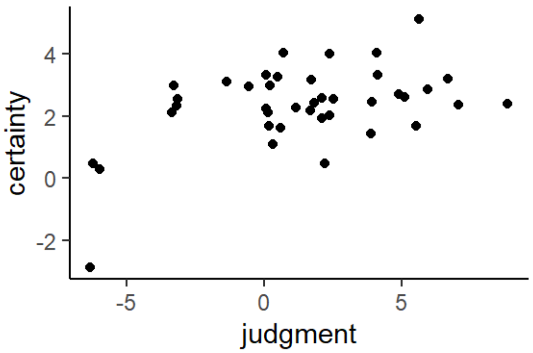
Get rid of this grey background

```
# Without a new theme being set
ggplot(data=d) +
  geom_point(mapping=aes(
    x=judgment,
    y=certainty))
```



A theme is set defining the look of *all subsequent* plots

```
theme_set(theme_classic()) # sets theme
ggplot(data=d) +
  geom_point(mapping=aes(
    x=judgment,
    y=certainty))
```



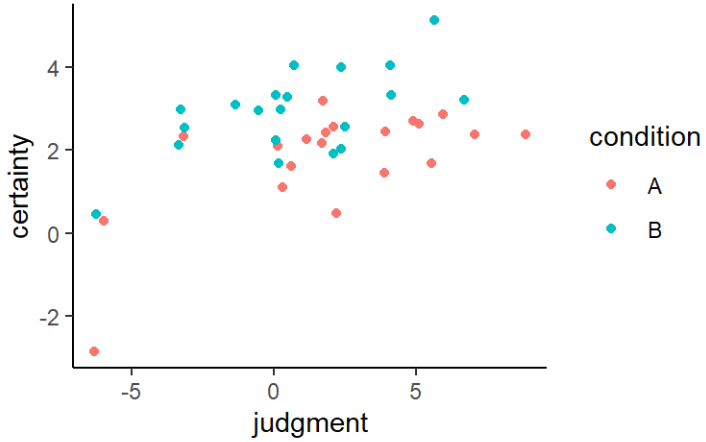
Note the (): `theme_classic()`

More themes: https://jrnold.github.io/ggthemes/

Aesthetic mapping to dimensions beyond x and y

This plot …

```
ggplot(data=d) +
  geom_point(mapping=aes(
    x=judgment,
    y=certainty,
    color=condition)
    )
```



… shows more than two variables

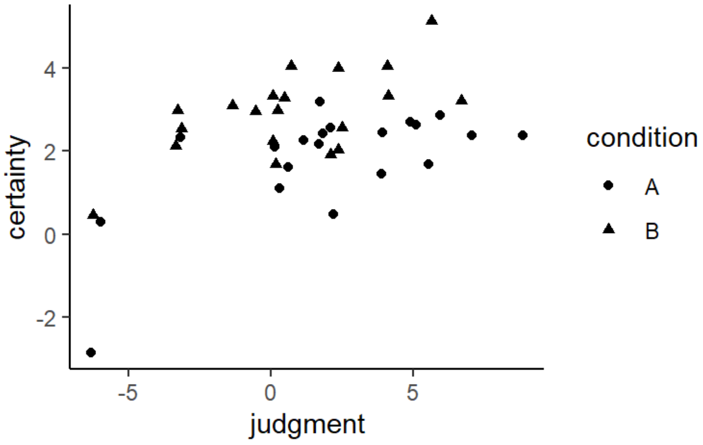
- *x- and y-axes* are the core aesthetic dimensions mapping variables `aes(x=judgment, y=certainty)`
- But they can map only two variables
- Beyond the axes, *color, shapes, linetypes, etc* are additional aesthetic dimensions mapping variables `aes(color=condition)`

Exersise: Mapping > 2 variables

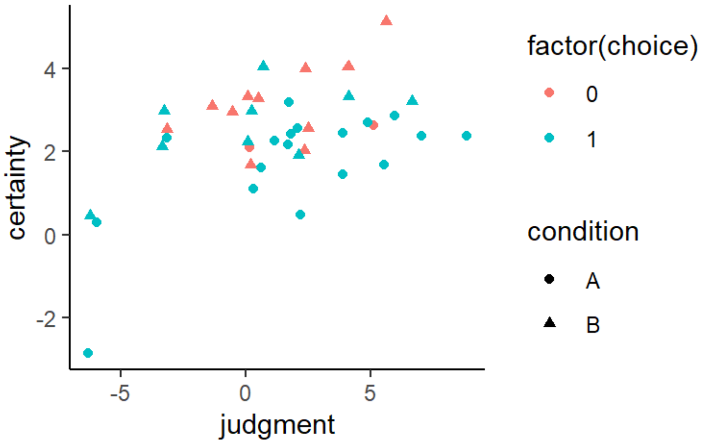
- Show the condition by point shape, rather than than color
- Show the variable choice by color
- (Optional) Change the shape of the points to a cross and a triangle, change the color scale to be grey to white, change the color scale to be discrete, change the title of the scales to “Observations” and “Experimental Condition”

Exersise: Mapping 2+ variables

```
ggplot(data=d) +
  geom_point(mapping=aes(
    x=judgment,
    y=certainty,
    shape=condition)
    )
```



```
ggplot(data=d) +
geom_point(mapping=aes(
  x=judgment,
  y=certainty,
  shape=condition,
  color=factor(choice))
  )
```



Advanced Exersises

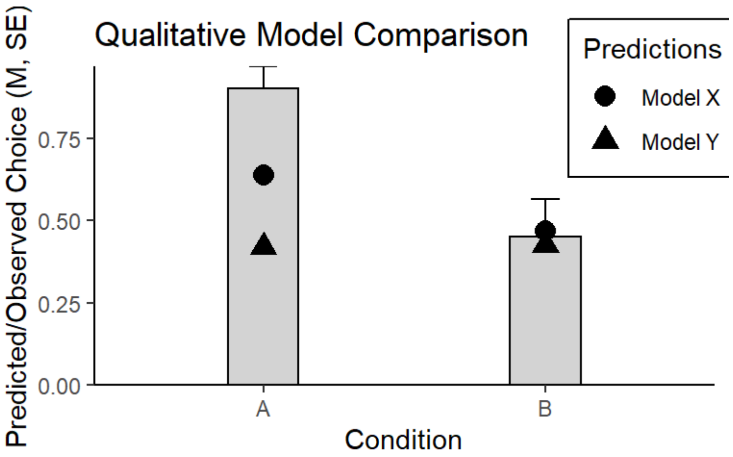
Exersise: Qualitative Prediction Plots

```
# Transformation
se <- function(x) {sd(x) / sqrt(length(x))}
obs_msd <- d %>%
  group_by(condition) %>%
  summarise(across(choice,
            .fns=list(M=mean, SD=sd, SE=se))

pred_msd <- d %>%
  group_by(condition) %>%
  summarise(
    across(ends_with("pred"),mean)) %>%
  pivot_longer(
    cols=contains("pred"),
    names_to = "model",
    values_to = "M")
N <- length(unique(d$id))
```

```
p <- ggplot(data=obs_msd, aes(
    x=condition,
    y=choice_M)) +
  geom_col(width=.25,
           color="black",
           fill="lightgrey") +
  geom_errorbar(aes(
    ymin=choice_M,
    ymax=choice_M + choice_SE),
    width=.1) +
  geom_point(data=pred_msd, aes(
    x=condition,
    y=M,
    shape=model),
    size=3.5)
# --> look at p
```

Exersise: Qualitative Prediction Plots (cont.)

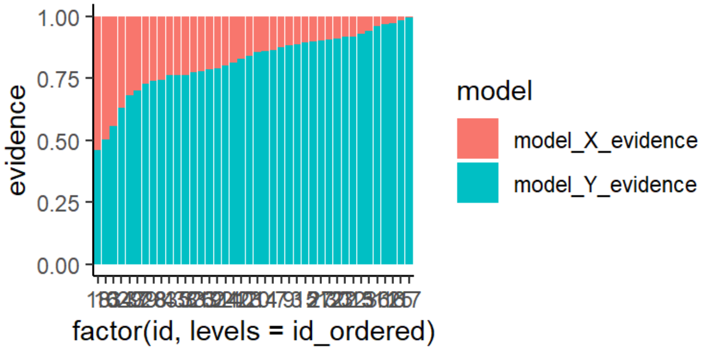
```r
# More cosmetics to the plot
p + scale_y_continuous(expand=c(0,NA)) +
  scale_shape_discrete(name="Predictions", labels=c("Model X","Model Y")) +
  ylab("Predicted/Observed Choice (M, SE)") +
  xlab("Condition") +
  theme(legend.position=c(.9,.9),
        legend.background=element_rect(color="black")) +
  ggtitle("Qualitative Model Comparison")
```



Advanced Exersise: Evidence Plots

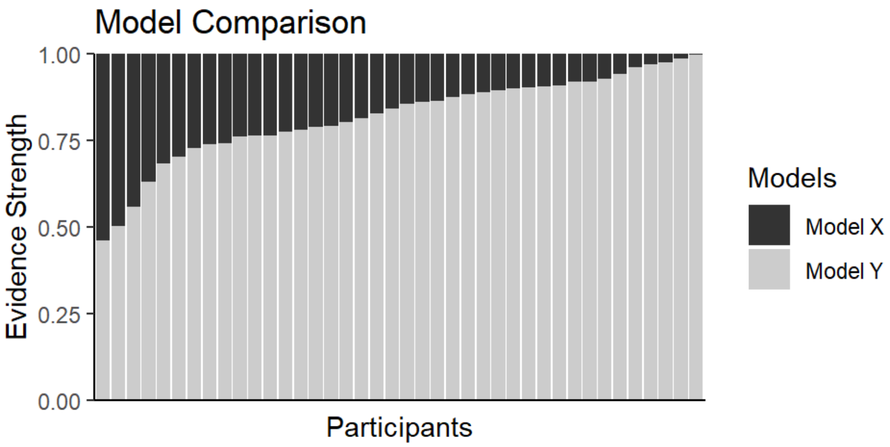
```r
# Transformation
dl <- pivot_longer(
  data = d,
  cols= c("model_X_evidence",
          "model_Y_evidence"),
  names_to = "model",
  values_to = "evidence")

id_order <- order(d$model_Y_evidence)
id_ordered <- d$id[id_order]
```

```r
p <- ggplot(data=dl) +
  geom_col(mapping=aes(
    x=factor(id, levels=id_ordered),
    y=evidence,
    fill=model))
p
```



Advanced Exersise: Evidence Plots (cont.)

```
# more cosmetics
p + scale_y_continuous(expand=c(0,0)) +
  scale_fill_grey("Models", labels=c("Model X", "Model Y")) +
  theme(axis.text.x = element_blank(),
    axis.ticks.x = element_blank()) +
  ylab("Evidence Strength") +
  xlab("Participants") +
  ggtitle("Model Comparison")
```


Model Comparison
Evidence Strength
Participants
Models
Model X
Model Y