Tame Your Data With R: ADVANCE Workshop 2

Elizabeth E. Esterly

December 4th, 2017, University of New Mexico

```
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr

## Conflicts with tidy packages ----------------------------------------------

## filter(): dplyr, stats
## lag():    dplyr, stats
```

```
super <- read_csv('superheroes.csv')
```

```
## Parsed with column specification:
## cols(
##   name = col_character(),
##   alignment = col_character(),
##   sex = col_character(),
##   age = col_integer(),
##   publisher = col_character()
## )
```

Transform

We'll bring in a second dataset here, transform it, and then transform our original dataset by doing a join on superhero names. For brevity's sake, assume this one is already tidied up.

```
#Import the second set and take a look at the first few rows
second <- read_csv("~/Desktop/characteristics.csv")
second %>% head()
```

```
## # A tibble: 6 x 3
##           name         status measurement
##          <chr>          <chr>       <chr>
## 1 Wonder Woman     homePlanet  Themyscira
## 2         Thor     homePlanet      Asgard
## 3        Storm     homePlanet       Earth
## 4     Superman     homePlanet     Krypton
## 5     Deadpool     homePlanet       Earth
## 6 Wonder Woman flyingSpeedmph   5,100,000
```

Huh. It looks like the name column repeats itself. Let's get a wider view…

```
second
```

```
## # A tibble: 10 x 3
##            name         status measurement
##           <chr>          <chr>       <chr>
##  1 Wonder Woman     homePlanet  Themyscira
##  2         Thor     homePlanet      Asgard
##  3        Storm     homePlanet       Earth
##  4     Superman     homePlanet     Krypton
##  5     Deadpool     homePlanet       Earth
##  6 Wonder Woman flyingSpeedmph   5,100,000
##  7         Thor flyingSpeedmph       24000
##  8        Storm flyingSpeedmph        1000
##  9     Superman flyingSpeedmph   7,200,000
## 10     Deadpool flyingSpeedmph           0
```

spread and gather

It definitely does. This data is a great candidate for transformation. Let's break up that status column into separate factors:

```
#Import the second set and take a look at the first few rows
secondWide <- second %>% spread(status, measurement)
secondWide
```

```
## # A tibble: 5 x 3
##           name flyingSpeedmph homePlanet
## *        <chr>          <chr>      <chr>
## 1     Deadpool              0      Earth
## 2        Storm           1000      Earth
## 3     Superman      7,200,000    Krypton
## 4         Thor          24000     Asgard
## 5 Wonder Woman      5,100,000 Themyscira
```

The Tidyverse `gather` function will help you do the opposite–go from wide to long. To learn more about that function, you can type:

```
help("gather")
```

You can do this with most functions in R to get some more details about them.

Note that the result of `spread` was sorted alphabetically by the first column. We really care about who's the fastest, though:

```
#Sort by speed
#ascending order
secondWide <- secondWide %>% arrange(flyingSpeedmph)
secondWide
```

```
## # A tibble: 5 x 3
##           name flyingSpeedmph homePlanet
##          <chr>          <chr>      <chr>
## 1     Deadpool              0      Earth
## 2        Storm           1000      Earth
## 3         Thor          24000     Asgard
## 4 Wonder Woman      5,100,000 Themyscira
## 5     Superman      7,200,000    Krypton
```

```
#descending order
secondWide <- secondWide %>% arrange(desc(flyingSpeedmph))
secondWide
```

```
## # A tibble: 5 x 3
##            name flyingSpeedmph homePlanet
##           <chr>          <chr>      <chr>
## 1      Superman      7,200,000    Krypton
## 2  Wonder Woman      5,100,000 Themyscira
## 3          Thor          24000     Asgard
## 4         Storm           1000      Earth
## 5      Deadpool              0      Earth
```

This looks like what we want. Let's merge with our `super` dataset.

joins

Don't be scared!

inner join(x, y)

From the documentation:

```r
help("inner_join")
```

> Return all rows from x where there are matching values in y, and all columns from x and y. If there are multiple matches >between x and y, all combination of the matches are returned. This is a mutating join.

We'll give our `super` dataset as x, because we want to keep all of the characters in `super` that also appear in `secondWide`. `secondWide` will be given second as y.
What is a mutating join?: It's some database terminology. It just means you can add variables to the LHS.

```r
#Try an inner join
inner_join(super, secondWide)
```

```
## Joining, by = "name"
```

```
## # A tibble: 4 x 7
##            name alignment    sex   age publisher flyingSpeedmph homePlanet
##           <chr>     <chr>  <chr> <int>     <chr>          <chr>      <chr>
## 1         Storm      good female   300    Marvel           1000      Earth
## 2          Thor      good   male    NA    Marvel          24000     Asgard
## 3  Wonder Woman      good female  5000        DC      5,100,000 Themyscira
## 4      Superman      good   male    29        DC      7,200,000    Krypton
```

```r
#The Tidyverse way: use pipes for first function argument 'x'
#Doesn't make a difference here but a good habit to get into
super %>% inner_join(secondWide)
```

```
## Joining, by = "name"
```

```
## # A tibble: 4 x 7
##            name alignment    sex   age publisher flyingSpeedmph homePlanet
##           <chr>     <chr>  <chr> <int>     <chr>          <chr>      <chr>
## 1         Storm      good female   300    Marvel           1000      Earth
## 2          Thor      good   male    NA    Marvel          24000     Asgard
## 3  Wonder Woman      good female  5000        DC      5,100,000 Themyscira
## 4      Superman      good   male    29        DC      7,200,000    Krypton
```

Notice we were automatically joined on name. We kept all characters that appeared in *both* datasets and merged all attributes given in both.

semi_join(x, y)

return all rows from x where there are matching values in y, keeping just columns from x

```
super %>% semi_join(secondWide)
```

```
## Joining, by = "name"

## # A tibble: 4 x 5
##            name alignment    sex   age publisher
##           <chr>     <chr>  <chr> <int>     <chr>
## 1         Storm      good female   300    Marvel
## 2          Thor      good   male    NA    Marvel
## 3 Wonder Woman      good female  5000        DC
## 4      Superman      good   male    29        DC
```

Here, our names were filtered by just the ones that appeared in both datasets, but we didn't pick up the extra columns from the `secondWide` dataset.

anti_join(x, y)

return all rows from x where there are not matching values in y, keeping just columns from x.

```
super %>% anti_join(secondWide)
```

```
## Joining, by = "name"

## # A tibble: 10 x 5
##             name alignment    sex   age  publisher
##            <chr>     <chr>  <chr> <int>      <chr>
##  1  Black Widow      <NA> female    38     Marvel
##  2      Magneto       bad   male    93     Marvel
##  3     Mystique       bad female   120     Marvel
##  4         Loki       bad   male  1048     Marvel
##  5       Batman      good   male    32         DC
##  6        Joker       bad   male    46         DC
##  7            0      good      0    NA         DC
##  8 Harley Quinn       bad female    27         DC
##  9    Supergirl      good female    24         DC
## 10      Hellboy      good   male    10 Dark Horse
```

In this case, we exclude all characters that appear in both x and y but still keep just columns from x.

full_join(x, y)

return all rows and all columns from both x and y. Where there are not matching values, returns NA for the one missing.

```
super %>% full_join(secondWide)
```

```
## Joining, by = "name"

## # A tibble: 15 x 7
##             name alignment    sex   age  publisher flyingSpeedmph
##            <chr>     <chr>  <chr> <int>      <chr>          <chr>
##  1  Black Widow      <NA> female    38     Marvel           <NA>
##  2      Magneto       bad   male    93     Marvel           <NA>
##  3        Storm      good female   300     Marvel           1000
##  4     Mystique       bad female   120     Marvel           <NA>
##  5        Thor      good   male    NA     Marvel          24000
##  6        Loki       bad   male  1048     Marvel           <NA>
##  7 Wonder Woman      good female  5000         DC      5,100,000
##  8      Batman      good   male    32         DC           <NA>
##  9       Joker       bad   male    46         DC           <NA>
## 10           0      good      0    NA         DC           <NA>
## 11 Harley Quinn       bad female    27         DC           <NA>
## 12   Supergirl      good female    24         DC           <NA>
## 13    Superman      good   male    29         DC      7,200,000
## 14     Hellboy      good   male    10 Dark Horse           <NA>
## 15    Deadpool      <NA>   <NA>    NA       <NA>              0
## # ... with 1 more variables: homePlanet <chr>
```

Let's do an anti join one more time and save the result:

```r
result <- super %>% anti_join(secondWide)
```

```
## Joining, by = "name"
```

```r
result
```

```
## # A tibble: 10 x 5
##            name alignment    sex   age  publisher
##           <chr>     <chr>  <chr> <int>      <chr>
##  1  Black Widow      <NA> female    38     Marvel
##  2      Magneto       bad   male    93     Marvel
##  3     Mystique       bad female   120     Marvel
##  4         Loki       bad   male  1048     Marvel
##  5       Batman      good   male    32         DC
##  6        Joker       bad   male    46         DC
##  7            0      good      0    NA         DC
##  8 Harley Quinn       bad female    27         DC
##  9    Supergirl      good female    24         DC
## 10      Hellboy      good   male    10 Dark Horse
```

Our dataset has come a long way since the beginning! Let's close by exploring some visualizations.

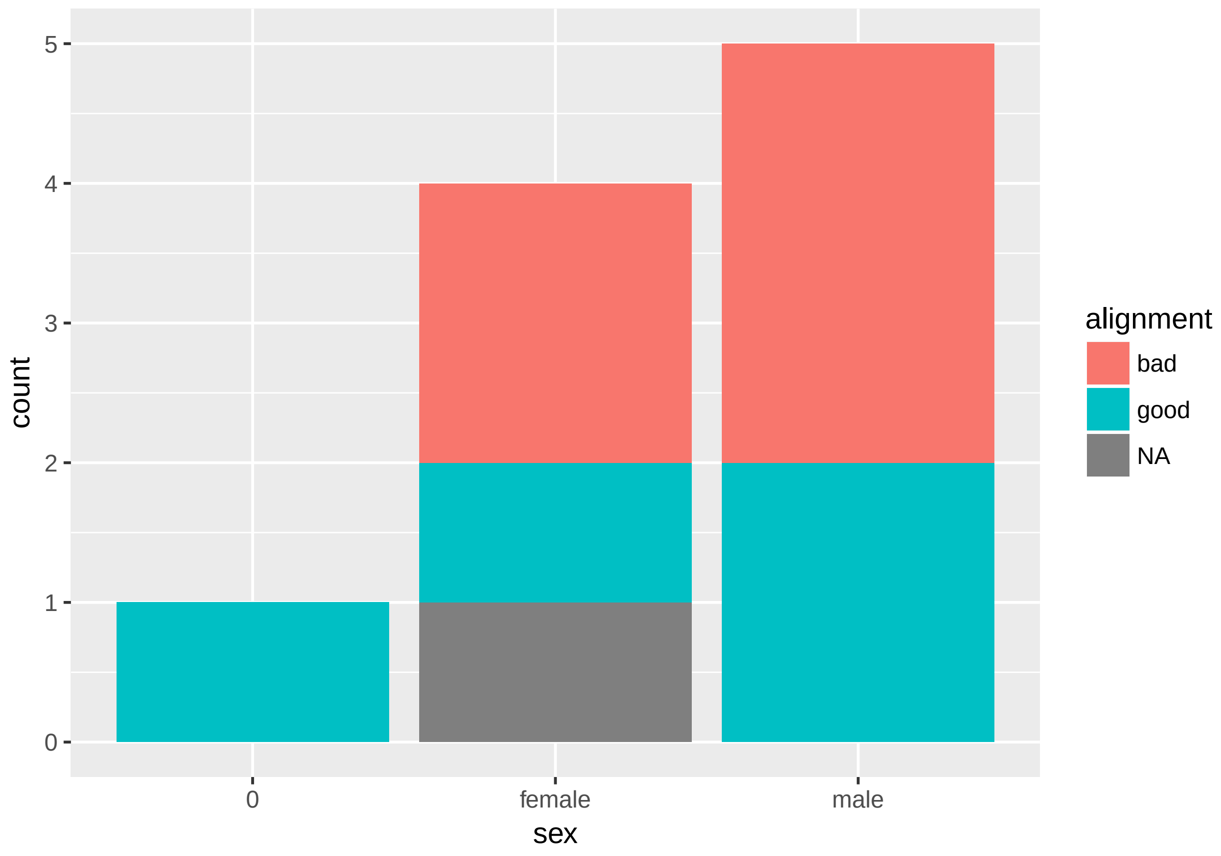
Visualize

The ggplot2 library provides an extensive set of tools to help you create beautiful visualizations. It can be hard to get started though, so I suggest looking at great examples, like from this website: http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Ordered%20Bar%20Chart and working backwards from there.

Categorical bar chart

Let's compare alignment between the sexes.

```r
result %>% ggplot(aes(sex)) +
  geom_bar(aes(fill=alignment))
```

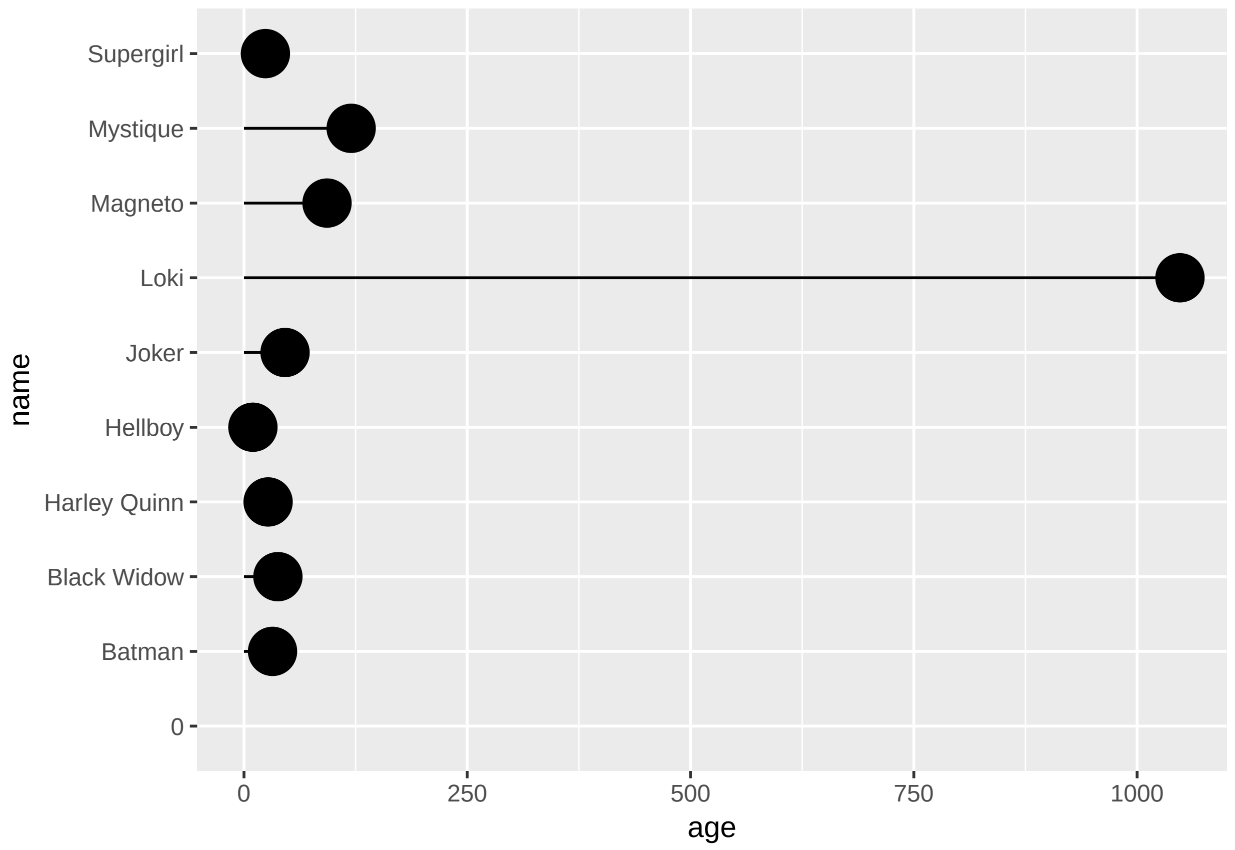


Lollipop charts

```
result %>% ggplot(aes(x=name, y=age, label=age )) +
  geom_point(stat='identity', fill="black", size=8) +
  geom_segment(aes(y = 0,
                   x = name,
                   yend = age,
                   xend= name),
               color = "black") +
  coord_flip()
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 rows containing missing values (geom_segment).
```

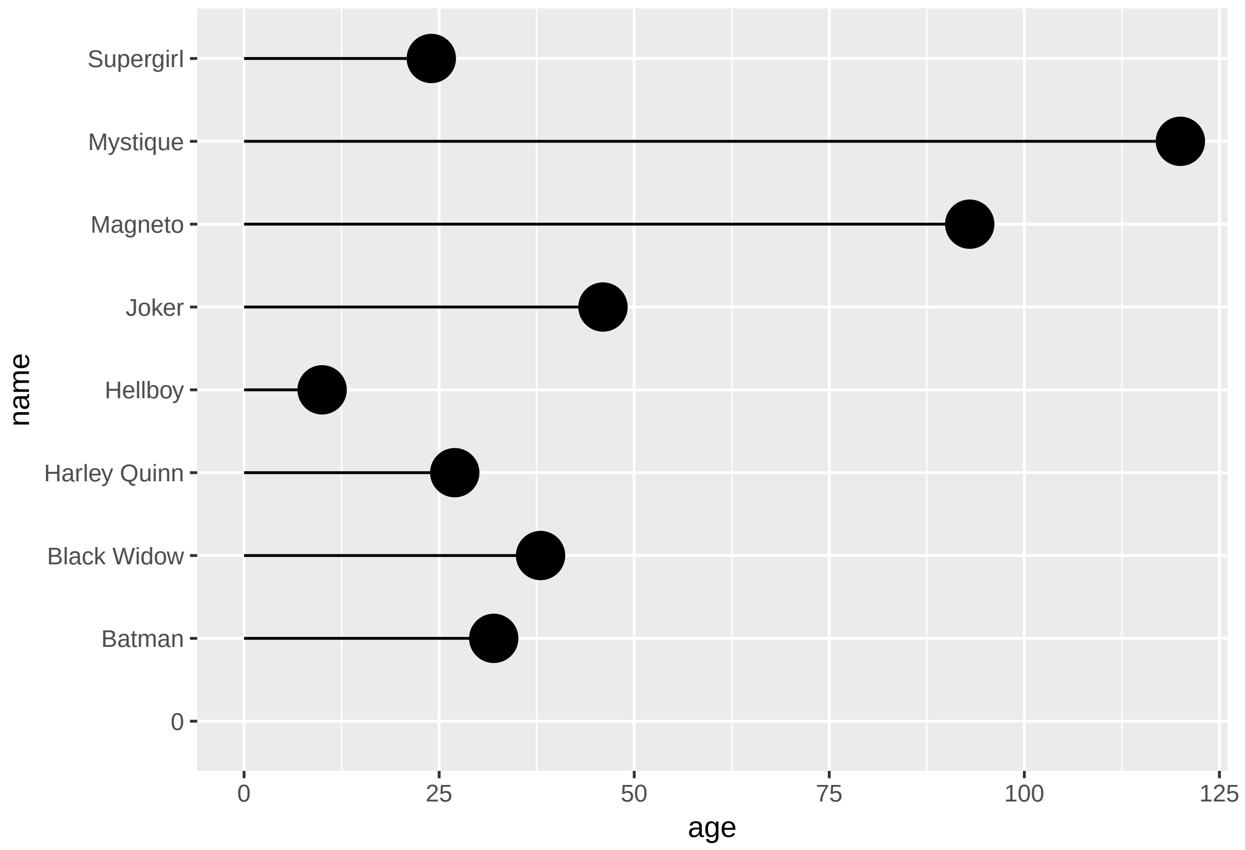


Loki is throwing things off here. Let's take him out:

```
result %>% filter(name != "Loki") %>% ggplot(aes(x=name, y=age, label=age )) +
  geom_point(stat='identity', fill="black", size=8) +
  geom_segment(aes(y = 0,
                   x = name,
                   yend = age,
                   xend= name),
               color = "black") +
  coord_flip()
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 rows containing missing values (geom_segment).
```



Much better!

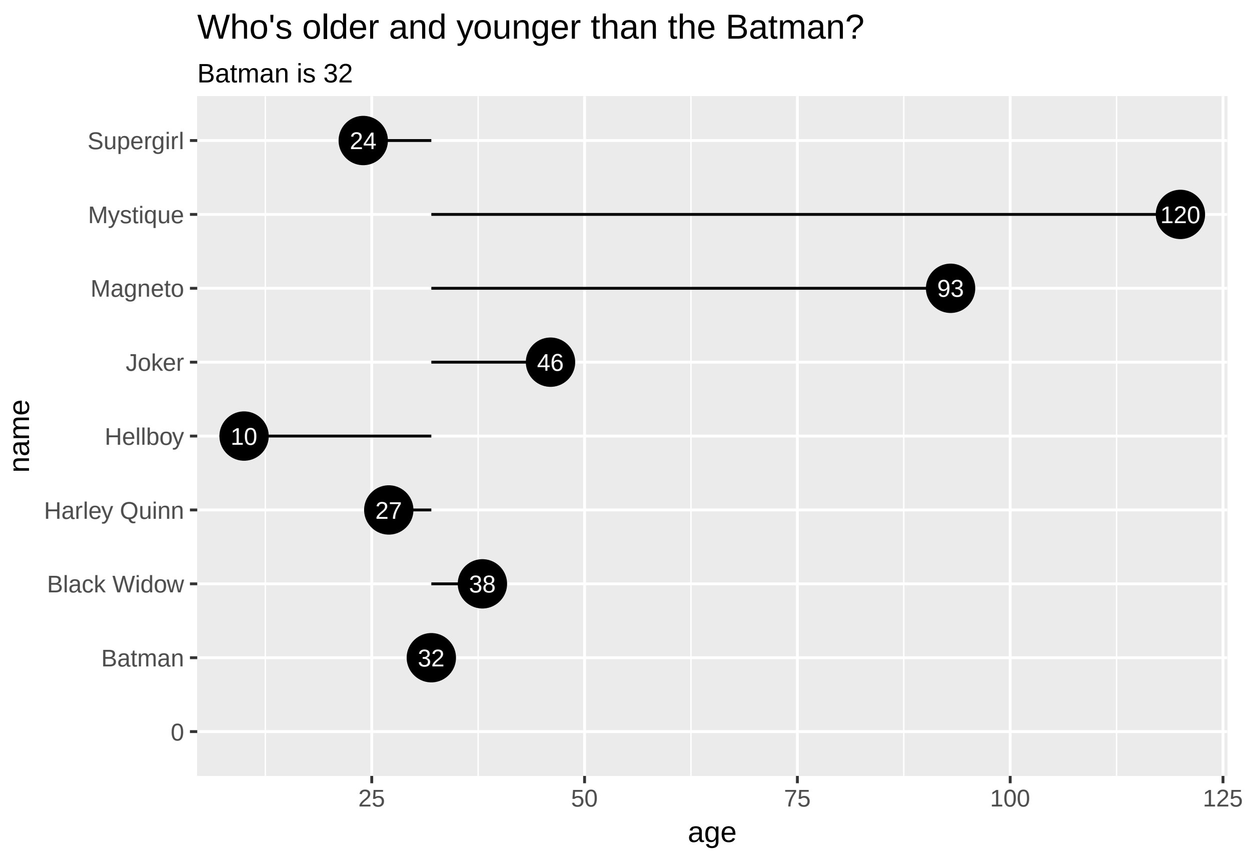
Let's add a little text in the dots for clarity and set our 'baseline' age to 32:

```
result %>% filter(name != "Loki") %>% ggplot(aes(x=name, y=age, label=age )) +
  geom_point(stat='identity', fill="black", size=8) +
  geom_segment(aes(y = 32,
                   x = name,
                   yend = age,
                   xend= name),
               color = "black") +
                geom_text(color="white", size=3)+
  labs(title="Who's older and younger than the Batman?", subtitle="Batman is 32")+
  coord_flip()
```

```
## Warning: Removed 1 rows containing missing values (geom_point).

## Warning: Removed 1 rows containing missing values (geom_segment).

## Warning: Removed 1 rows containing missing values (geom_text).
```


Who's older and younger than the Batman?
Batman is 32
Supergirl
Mystique
Magneto
Joker
Hellboy
Harley Quinn
Black Widow
Batman
0
24
120
93
46
10
27
38
32
name
25
50
75
100
125
age