Tame Your Data With R: ADVANCE Workshop 2

Elizabeth E. Esterly

December 4th, 2017, University of New Mexico

Getting Started

Download these datasets to your desktop. DATASET 1: http://cs.unm.edu/~elizabeth/newSuperheroes.csv (http://cs.unm.edu/~elizabeth/newSuperheroes.csv)
DATASET 2: http://cs.unm.edu/~elizabeth/newCharacteristics.csv (http://cs.unm.edu/~elizabeth/newCharacteristics.csv)

```
#Load the Tidyverse
library(tidyverse)
```

```
#Read in the superhero data
super <- read_csv('newSuperheroes.csv')
```

Transform

We'll bring in a second dataset here, transform it, and then transform our original dataset by doing some different joins of these two datasets.

```
#Import the second set and take a look at the first few rows
second <- read_csv('newCharacteristics.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 more complete view…

```
second
```

```
## # A tibble: 20 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
## 11 Wonder Woman      mortality    immortal
## 12          Thor      mortality    immortal
## 13         Storm      mortality      mortal
## 14      Superman      mortality      mortal
## 15      Deadpool      mortality    immortal
## 16 Wonder Woman            age        5000
## 17          Thor            age          29
## 18         Storm            age         300
## 19      Superman            age          29
## 20      Deadpool            age          45
```

I think we should do a transformation here, but let's do it after the join, so we can transform all at once.

joins

Don't be scared! Take it slow.

inner join(x, y)

From the documentation:

```
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 `second`. `second` 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.

```
#Try an inner join
inner_join(super, second)
```

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

```
## # A tibble: 16 x 7
##             name alignment    sex   age publisher         status
##            <chr>     <chr>  <chr> <dbl>     <chr>          <chr>
##  1        Storm      good female   300    Marvel     homePlanet
##  2        Storm      good female   300    Marvel flyingSpeedmph
##  3        Storm      good female   300    Marvel      mortality
##  4        Storm      good female   300    Marvel            age
##  5         Thor      good   male    29    Marvel     homePlanet
##  6         Thor      good   male    29    Marvel flyingSpeedmph
##  7         Thor      good   male    29    Marvel      mortality
##  8         Thor      good   male    29    Marvel            age
##  9 Wonder Woman      good female  5000        DC     homePlanet
## 10 Wonder Woman      good female  5000        DC flyingSpeedmph
## 11 Wonder Woman      good female  5000        DC      mortality
## 12 Wonder Woman      good female  5000        DC            age
## 13     Superman      good   male    29        DC     homePlanet
## 14     Superman      good   male    29        DC flyingSpeedmph
## 15     Superman      good   male    29        DC      mortality
## 16     Superman      good   male    29        DC            age
## # ... with 1 more variables: measurement <chr>
```

```
#The Tidyverse way: use pipes for first function argument 'x'
#Same result
superInner <- super %>% inner_join(second)
```

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

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

spread and gather

This data is a great candidate for transformation. Let's break up that status column into separate factors by using spread to make it a wide format:

```
#Use the spread command to spread the status column out. (Go to "wide.")
superInnerSpread <- superInner %>% spread(status, measurement, convert=TRUE)
superInnerSpread
```

```
## # A tibble: 4 x 9
##           name alignment    sex   age publisher   age flyingSpeedmph
## *        <chr>     <chr>  <chr> <dbl>     <chr> <int>          <chr>
## 1        Storm      good female   300    Marvel   300           1000
## 2     Superman      good   male    29        DC    29      7,200,000
## 3         Thor      good   male    29    Marvel    29          24000
## 4 Wonder Woman      good female  5000        DC  5000      5,100,000
## # ... with 2 more variables: homePlanet <chr>, mortality <chr>
```

(We'll just have time to practice the spread function today, but 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.) Let's learn more joins and practice using the spread function to transform them.

semi_join(x, y)

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

```
superSemi <- super %>% semi_join(second)
```

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

```
superSemi
```

```
## # A tibble: 4 x 5
##           name alignment    sex   age publisher
##          <chr>     <chr>  <chr> <dbl>     <chr>
## 1        Storm      good female   300    Marvel
## 2         Thor      good   male    29    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 `second` dataset like we did in the inner join. There's nothing to spread on. Let's try a full join now.

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.

```
superFull <- super %>% full_join(second)
```

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

```
superFull
```

```
## # A tibble: 29 x 7
##           name alignment    sex   age publisher         status measurement
##          <chr>     <chr>  <chr> <dbl>     <chr>          <chr>       <chr>
##  1 Black Widow      good female    38    Marvel           <NA>        <NA>
##  2     Magneto       bad   male    93    Marvel           <NA>        <NA>
##  3       Storm      good female   300    Marvel     homePlanet       Earth
##  4       Storm      good female   300    Marvel flyingSpeedmph        1000
##  5       Storm      good female   300    Marvel      mortality      mortal
##  6       Storm      good female   300    Marvel            age         300
##  7    Mystique       bad female   120    Marvel           <NA>        <NA>
##  8        Thor      good   male    29    Marvel     homePlanet      Asgard
##  9        Thor      good   male    29    Marvel flyingSpeedmph       24000
## 10        Thor      good   male    29    Marvel      mortality    immortal
## # ... with 19 more rows
```

...and spread it out.

```
superFullSpread <- superFull %>% spread(status, measurement, convert=TRUE, fill="FILL
ME!")
superFullSpread
```

```
## # A tibble: 14 x 10
##            name alignment    sex   age  publisher      age flyingSpeedmph
##  *          <chr>     <chr>  <chr> <dbl>      <chr>    <chr>          <chr>
##  1       Batman      good   male    32         DC FILL ME!       FILL ME!
##  2  Black Widow      good female    38     Marvel FILL ME!       FILL ME!
##  3     Deadpool      <NA>   <NA>    NA       <NA>       45              0
##  4 Harley Quinn       bad female    27         DC FILL ME!       FILL ME!
##  5      Hellboy      good   male    10 Dark Horse FILL ME!       FILL ME!
##  6        Joker       bad   male    46         DC FILL ME!       FILL ME!
##  7         Loki       bad   male  1048     Marvel FILL ME!       FILL ME!
##  8      Magneto       bad   male    93     Marvel FILL ME!       FILL ME!
##  9     Mystique       bad female   120     Marvel FILL ME!       FILL ME!
## 10        Storm      good female   300     Marvel      300           1000
## 11    Supergirl      good female    24         DC FILL ME!       FILL ME!
## 12     Superman      good   male    29         DC       29      7,200,000
## 13         Thor      good   male    29     Marvel       29          24000
## 14 Wonder Woman      good female  5000         DC     5000      5,100,000
## # ... with 3 more variables: homePlanet <chr>, mortality <chr>,
## #   `<NA>` <chr>
```

anti_join(x, y)

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

```
superAnti <- super %>% anti_join(second)
```

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

```
superAnti
```

```
## # A tibble: 9 x 5
##           name alignment    sex   age  publisher
##          <chr>     <chr>  <chr> <dbl>      <chr>
## 1  Black Widow      good 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 Harley Quinn       bad female    27         DC
## 8    Supergirl      good female    24         DC
## 9      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.
Let's close by exploring some visualizations.

Visualize

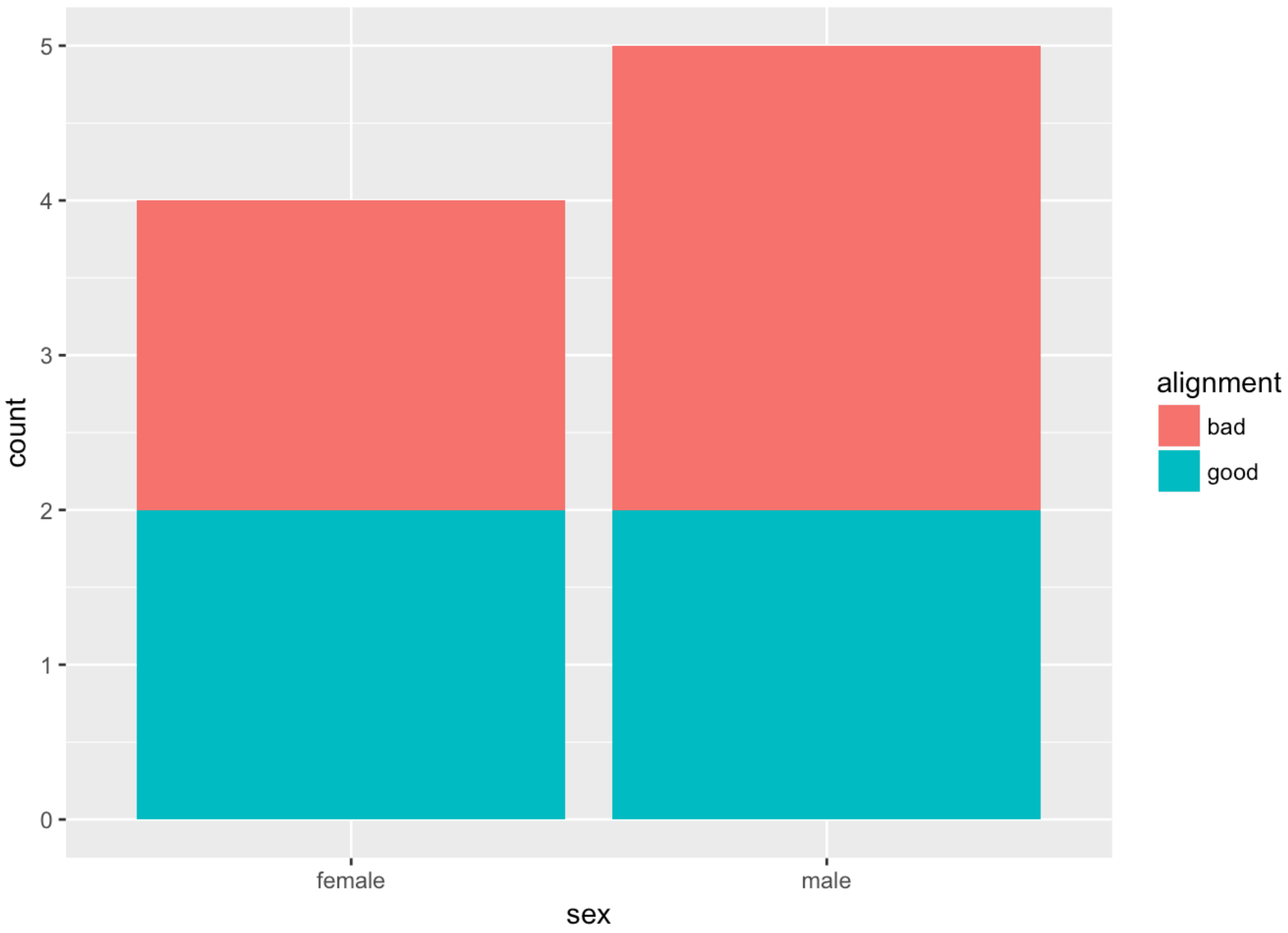
The `ggplot2` library provides an extensive set of tools to help you create beautiful visualizations. It can be hard to get started though, so I suggest looking at great examples, like from this website: http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Ordered%20Bar%20Chart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

QUESTION 1: WHO'S BAD?

Categorical bar chart

Let's compare alignment between the sexes. Who's more evil in our anti-joined result?

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



Exercise:

How about our spread full-joined result?

Our semi-join result?

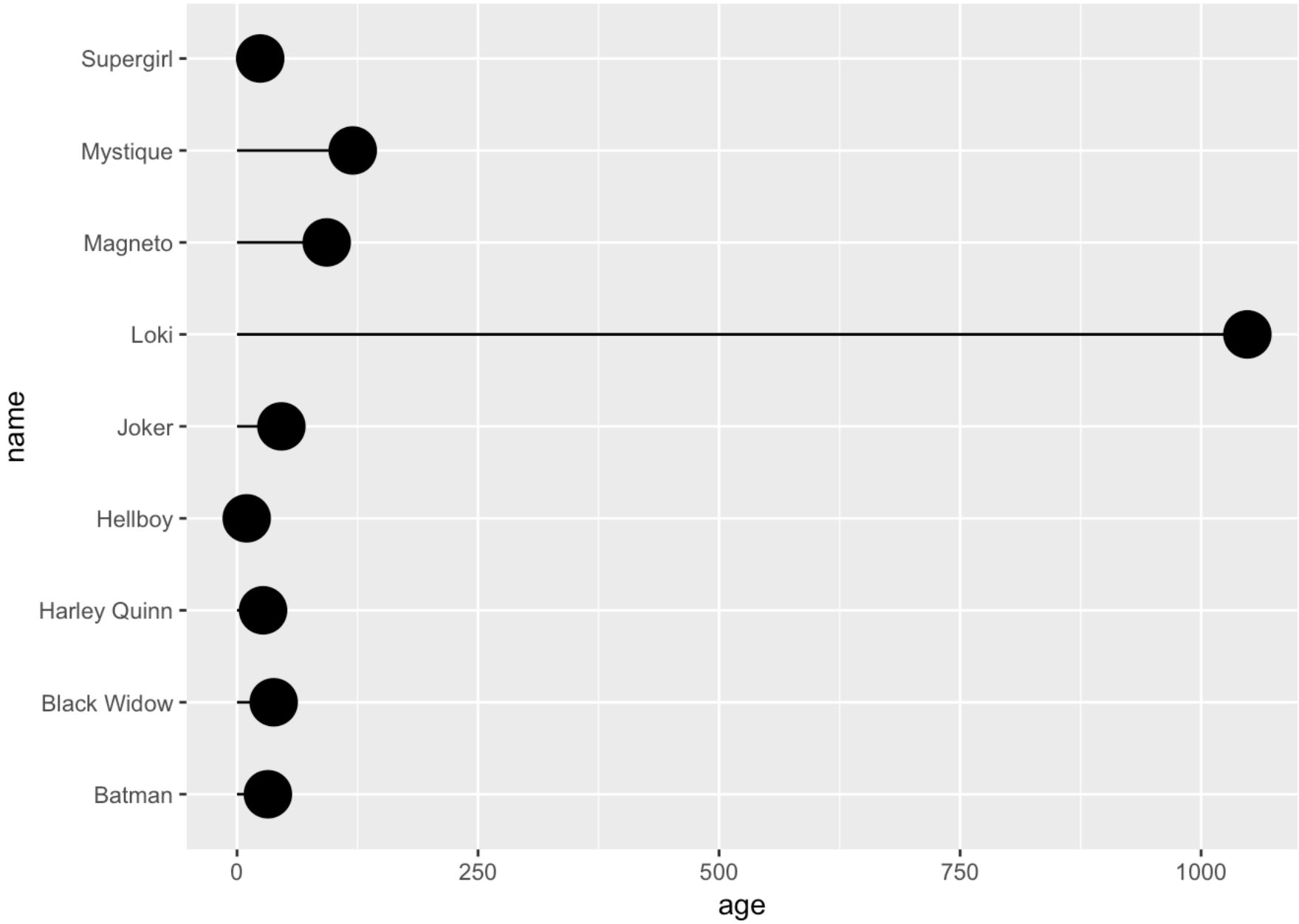
Our spread inner-joined result?

How can we handle NA's?

QUESTION 2: WHO'S OLDER AND YOUNGER THAN BATMAN?

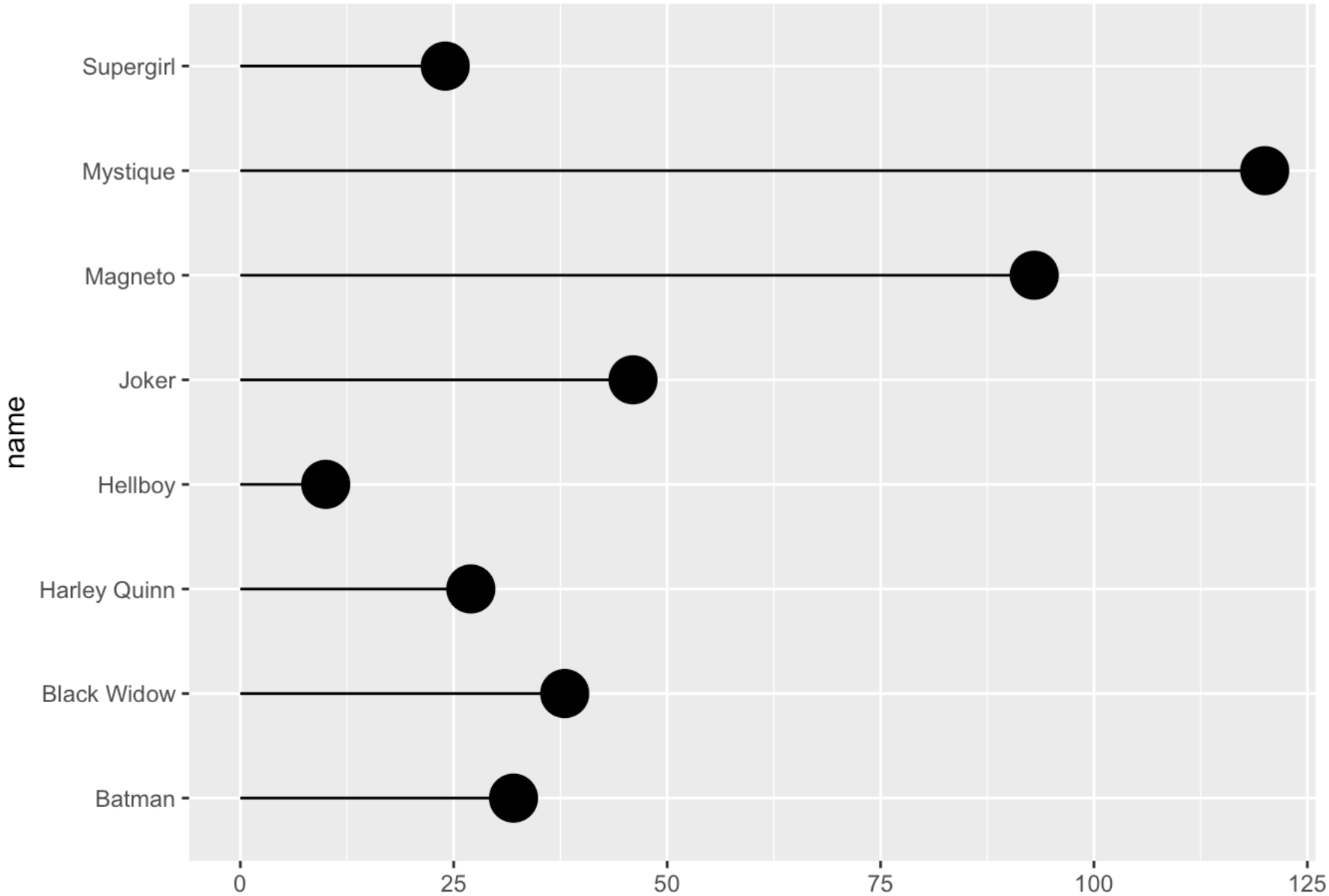
Lollipop charts

```
superAnti %>% 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()
```



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

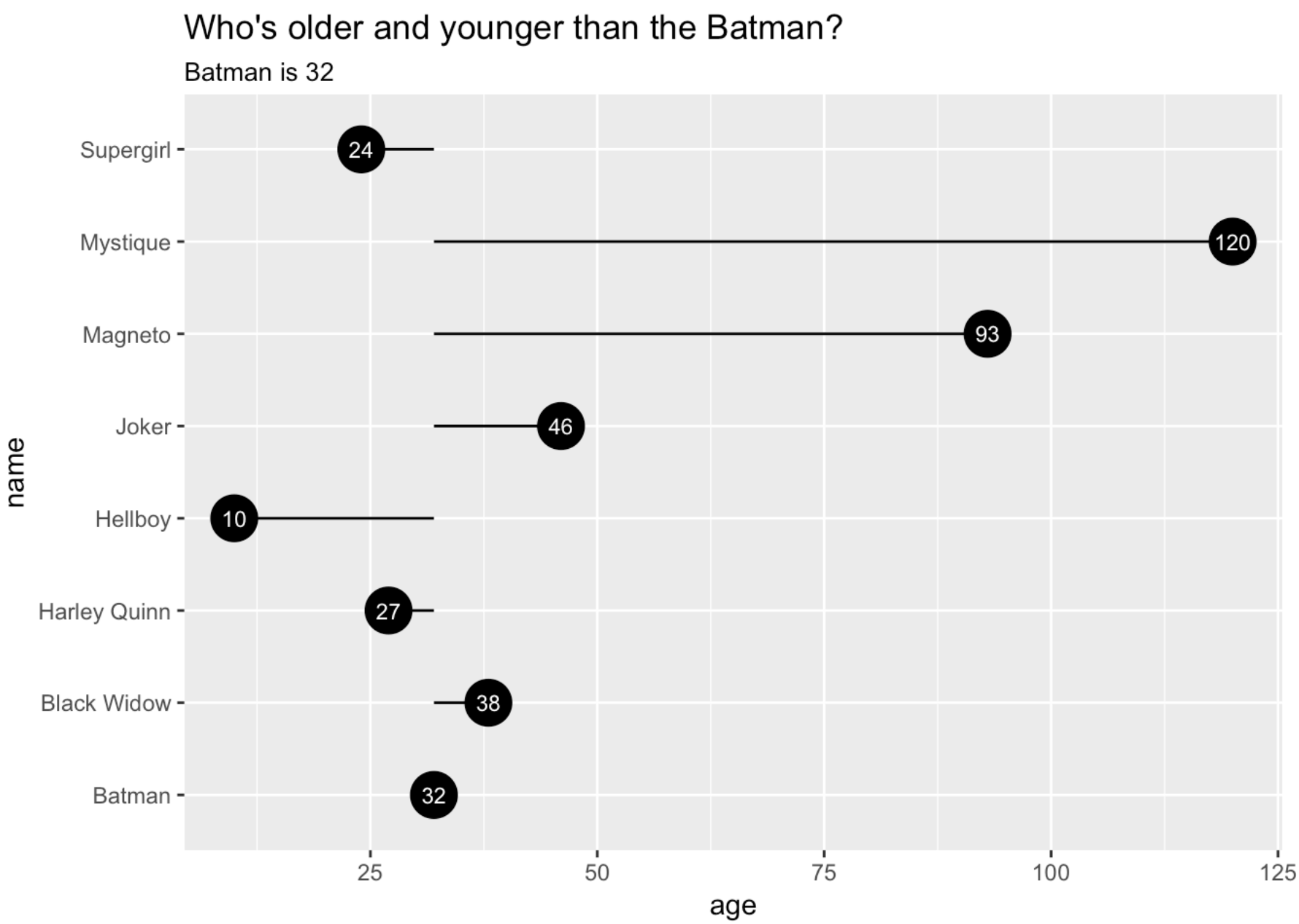
```
superAnti %>% filter(name != "Loki") %>% ggplot(aes(x=name, y=age, label=age )) +
  geom_point(stat='identity', fill="black", size=8) +
  geom_segment(aes(y = 0,
                   x = name,
                   yend = age,
                   xend= name),
               color = "black") +
  coord_flip()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



Much better!

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

```
superAnti %>% 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()
```


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