Tame Your Data With R: ADVANCE Workshop 1

Elizabeth E. Esterly

October 23rd, 2017, University of New Mexico

datasets used in this workshop loosely based on several open-source ones provided by Jenny Bryan, University of British Colombia, RStudio and Tidyverse https://github.com/jennybc

DATASET 1: http://cs.unm.edu/~elizabeth/superheroes.csv
DATASET 2: http://cs.unm.edu/~elizabeth/characteristics.csv
Save these to your Desktop.

Getting Started

I didn't install R before the Workshop!
* No problem, go here and follow the instructions: https://goo.gl/Utw4KG
Coming from Stata?
* Open this in a new tab: https://goo.gl/XFgudR

As you follow along with this tutorial in the console, you can store all of the commands in an R script to use and run later. To create a new R script, click the green `+` button in the upper left corner of your window and select `RScript`.
A note about your working directory: R needs to know where you want to work from. To follow along with this workshop, it's easiest to work right from your Desktop. You can change this working directory at any time. Change your working directory to the Desktop now by going to `Session > Set Working Directory...` and choosing your Desktop.

In this workshop series, we'll be using the Tidyverse approach to data:
* *Import*
* *Tidy*
* *Transform -> visualize -> model* (repeat as necessary to gain understanding)
* *Communicate*
The Tidyverse set of libraries include *ggplot2* for plotting, *dplyr* for fast manipulation of dataframes, among others. Let's load them all as one set with:

```
library(tidyverse)
```

You might see some conflict messages here. That's fine.
Let's begin the Tidyverse workflow.

Import

It's great to get a quick feel for the data, especially if we have missing values.
`head` shows us the first few rows of our data:

```
#load dataset 1, superheroes, and assign it to the variable super
super <- read_csv("~/Desktop/superheroes.csv")
#look at the first few rows
head(super)
```

```
## # A tibble: 6 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       Storm      good female   300    Marvel
## 4    Mystique       bad female   120    Marvel
## 5        Thor      good   male    NA    Marvel
## 6        Loki       bad   male  1048    Marvel
```

`tail` shows us the last few rows:

```
#look at the last few rows
tail(super)
```

```
## # A tibble: 6 x 5
##           name alignment    sex   age  publisher
##          <chr>     <chr>  <chr> <int>      <chr>
## 1        Joker       bad   male    46         DC
## 2            0      good      0    NA         DC
## 3 Harley Quinn       bad female    27         DC
## 4    Supergirl      good female    24         DC
## 5     Superman      good   male    29         DC
## 6      Hellboy      good   male    10 Dark Horse
```

Looks like we've got some NAs–and 0 values too. We'll deal with those in a minute.

First, let's get some summary information:

```
#get a summary of the data: notice that some values are character and some are numeric
summary(super)
```

```
##      name            alignment            sex                 age        
##  Length:14          Length:14          Length:14          Min.   :  10.0  
##  Class :character   Class :character   Class :character   1st Qu.:  28.5  
##  Mode  :character   Mode  :character   Mode  :character   Median :  42.0  
##                                                           Mean   : 563.9  
##                                                           3rd Qu.: 165.0  
##                                                           Max.   :5000.0  
##                                                           NA's   :2       
##   publisher        
##  Length:14          
##  Class :character   
##  Mode  :character   
##                     
##                     
##                     
##
```

Tables will give us the whole picture of our factors and levels.

```
#build a contingency table of the counts at each combination of factor levels
#this is going to be huge
table(super)
```

How about trying this the Tidyverse way?

```
#use the pipe command to give the super data to the table function
super %>% table()
```

How about a two-way table?

```
super %>% select(age, sex) %>% table()
```

```
##       sex
## age    0 female male
##   10   0      0    1
##   24   0      1    0
##   27   0      1    0
##   29   0      0    1
##   32   0      0    1
##   38   0      1    0
##   46   0      0    1
##   93   0      0    1
##   120  0      1    0
##   300  0      1    0
##   1048 0      0    1
##   5000 0      1    0
```

Our Import is complete and we have a sense of the data! Let move on to Step 2, Tidy.

Tidy

It's time to clean up those 0 and NA values we saw earlier.
Dealing with two different types of NAs is tough, so let's reimport the data so those `0` values are converted to `NA` also:

```
#load dataset 1, superheroes, and re-assign it to the variable super, reusing this variable to save spa
#important for very large datasets
#add a list of entries we want to fill with NA
super <- read_csv("~/Desktop/superheroes.csv", na = c("","0","NA"))
#look at the first few rows again
super %>% head()
```

```
## # A tibble: 6 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       Storm      good female   300    Marvel
## 4    Mystique       bad female   120    Marvel
## 5        Thor      good   male    NA    Marvel
## 6        Loki       bad   male  1048    Marvel
```

```
#and the last to make sure everything was replaced correctly
super %>% tail()
```

```
## # A tibble: 6 x 5
##           name alignment    sex   age  publisher
##          <chr>     <chr>  <chr> <int>      <chr>
## 1        Joker       bad   male    46         DC
## 2         <NA>      good   <NA>    NA         DC
## 3 Harley Quinn       bad female    27         DC
## 4    Supergirl      good female    24         DC
## 5     Superman      good   male    29         DC
## 6      Hellboy      good   male    10 Dark Horse
```

Notice that the 0s in the row below `Joker` have been replaced by `NA`.

dropping rows with NA values

R allows us to mess around with our data without mutating the original. In other words, unless we assign something to `super` again, `super` will be in the same state as when we left it. This is both good and bad–once you made the change you want to your data, don't forget to either assign it to your old variable again with `<-` or create a new one, or your changes will be lost!

drop all rows with NAs

The nuclear option.

```
#all rows that contain NAs are dropped: Black Widow, Thor, and the nameless row below Joker
super %>% drop_na()
```

```
## # A tibble: 11 x 5
##            name alignment    sex   age  publisher
##           <chr>     <chr>  <chr> <int>      <chr>
##  1      Magneto       bad   male    93     Marvel
##  2        Storm      good female   300     Marvel
##  3     Mystique       bad female   120     Marvel
##  4         Loki       bad   male  1048     Marvel
##  5 Wonder Woman      good female  5000         DC
##  6       Batman      good   male    32         DC
##  7        Joker       bad   male    46         DC
##  8 Harley Quinn       bad female    27         DC
##  9    Supergirl      good female    24         DC
## 10     Superman      good   male    29         DC
## 11      Hellboy      good   male    10 Dark Horse
```

drop just rows that have NA's in certain columns

Give the column name to the drop_na() function:

```
#see ya, Black Widow!
super %>% drop_na(alignment)
```

```
## # A tibble: 13 x 5
##            name alignment    sex   age  publisher
##           <chr>     <chr>  <chr> <int>      <chr>
##  1      Magneto       bad   male    93     Marvel
##  2        Storm      good female   300     Marvel
##  3     Mystique       bad female   120     Marvel
##  4         Thor      good   male    NA     Marvel
##  5         Loki       bad   male  1048     Marvel
##  6 Wonder Woman      good female  5000         DC
##  7       Batman      good   male    32         DC
##  8        Joker       bad   male    46         DC
##  9         <NA>      good   <NA>    NA         DC
## 10 Harley Quinn       bad female    27         DC
## 11    Supergirl      good female    24         DC
## 12     Superman      good   male    29         DC
## 13      Hellboy      good   male    10 Dark Horse
```

```
#Bye, Thor!
super %>% drop_na(age)
```

```
## # A tibble: 12 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        Storm      good female   300     Marvel
##  4     Mystique       bad female   120     Marvel
##  5         Loki       bad   male  1048     Marvel
##  6 Wonder Woman      good female  5000         DC
##  7       Batman      good   male    32         DC
##  8        Joker       bad   male    46         DC
##  9 Harley Quinn       bad female    27         DC
## 10    Supergirl      good female    24         DC
## 11     Superman      good   male    29         DC
## 12      Hellboy      good   male    10 Dark Horse
```

```
#Or both of you!
super %>% drop_na(alignment, age)
```

```
## # A tibble: 11 x 5
##            name alignment    sex   age  publisher
##           <chr>     <chr>  <chr> <int>      <chr>
##  1      Magneto       bad   male    93     Marvel
##  2        Storm      good female   300     Marvel
##  3     Mystique       bad female   120     Marvel
##  4         Loki       bad   male  1048     Marvel
##  5 Wonder Woman      good female  5000         DC
##  6       Batman      good   male    32         DC
##  7        Joker       bad   male    46         DC
##  8 Harley Quinn       bad female    27         DC
##  9    Supergirl      good female    24         DC
## 10     Superman      good   male    29         DC
## 11      Hellboy      good   male    10 Dark Horse
```

```
#Let's drop that row where we don't know the identity of the entry and save the changes.
#When you assign the variable, just type its name again to show results.
super <- super %>% drop_na(name)
super
```

```
## # A tibble: 13 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        Storm      good female   300     Marvel
##  4     Mystique       bad female   120     Marvel
##  5         Thor      good   male    NA     Marvel
##  6         Loki       bad   male  1048     Marvel
##  7 Wonder Woman      good female  5000         DC
##  8       Batman      good   male    32         DC
##  9        Joker       bad   male    46         DC
## 10 Harley Quinn       bad female    27         DC
```

```
## 11    Supergirl      good female    24         DC
## 12     Superman      good   male    29         DC
## 13      Hellboy      good   male    10 Dark Horse
```

drop a column altogether by name (subsetting/selecting)

```
#Drop the publisher column and save the changes.
super <-super %>% subset(select = -c(publisher))
super
```

```
## # A tibble: 13 x 4
##            name alignment    sex   age
##           <chr>     <chr>  <chr> <int>
##  1  Black Widow      <NA> female    38
##  2      Magneto       bad   male    93
##  3        Storm      good female   300
##  4     Mystique       bad female   120
##  5         Thor      good   male    NA
##  6         Loki       bad   male  1048
##  7 Wonder Woman      good female  5000
##  8       Batman      good   male    32
##  9        Joker       bad   male    46
## 10 Harley Quinn       bad female    27
## 11    Supergirl      good female    24
## 12     Superman      good   male    29
## 13      Hellboy      good   male    10
```

fill a numerical NA value

How old is Thor? More sophisticated types of imputation are beyond the scope of this workshop. Let's get the simple summary stats:

```
#get the summary of the age column
super %>% select(age) %>% summary()
```

```
##       age
##  Min.   :  10.0
##  1st Qu.:  28.5
##  Median :  42.0
##  Mean   : 563.9
##  3rd Qu.: 165.0
##  Max.   :5000.0
##  NA's   :1
```

```
#how about a summary for just the guys?
super %>% filter(sex == 'male') %>% select(age) %>% summary()
```

```
##       age
##  Min.   :  10.00
##  1st Qu.:  29.75
##  Median :  39.00
##  Mean   : 209.67
##  3rd Qu.:  81.25
##  Max.   :1048.00
##  NA's   :1
```

```
#how about a summary for just the good guys?
super %>% filter(sex == 'male') %>% filter(alignment == 'good') %>% select(age) %>% summary()
```

```
##       age
##  Min.   :10.00
##  1st Qu.:19.50
##  Median :29.00
##  Mean   :23.67
##  3rd Qu.:30.50
##  Max.   :32.00
##  NA's   :1
```

Well, that helped us narrow it down a bit. We know Thor's a good guy, so we'll fill Thor's age with the median age for good guys, 29.0.
NOTE: (Filter before Select) Filter is slicing by rows, and select is slicing by columns. You always want to do your row slices first, or you might lose the identifying columns to do them on!

```
#replace NA in the age column with 29
super <- super %>% replace_na(list(age = 29))
#did it work?
super
```

```
## # A tibble: 13 x 4
##            name alignment    sex   age
##           <chr>     <chr>  <chr> <dbl>
##  1  Black Widow      <NA> female    38
##  2      Magneto       bad   male    93
##  3        Storm      good female   300
##  4     Mystique       bad female   120
##  5         Thor      good   male    29
##  6         Loki       bad   male  1048
##  7 Wonder Woman      good female  5000
##  8       Batman      good   male    32
##  9        Joker       bad   male    46
## 10 Harley Quinn       bad female    27
## 11    Supergirl      good female    24
## 12     Superman      good   male    29
## 13      Hellboy      good   male    10
```

fill a categorical NA value

Is Black Widow good or bad? Again, some simple imputation techniques for this binary category.

```
#what's the modal alignment?
super %>% select(alignment) %>% table() %>% which.max() %>% names()
```

```
## [1] "good"
```

```
#how about just the women?
super %>% filter(sex == 'female') %>% select(alignment) %>% table() %>% which.max() %>% names()
```

```
## [1] "good"
```

"Good" it is. Try using `replace_na` again, but this time, combine everything into one statement. You don't even have to know the value in advance:

```
super <- super %>% replace_na(list(alignment = super %>% filter(sex == 'female') %>% select(alignment) %
#did it work?
super
```

```
## # A tibble: 13 x 4
##            name alignment    sex   age
##           <chr>     <chr>  <chr> <dbl>
##  1  Black Widow      good female    38
##  2      Magneto       bad   male    93
##  3        Storm      good female   300
##  4     Mystique       bad female   120
##  5         Thor      good   male    29
##  6         Loki       bad   male  1048
##  7 Wonder Woman      good female  5000
##  8       Batman      good   male    32
##  9        Joker       bad   male    46
## 10 Harley Quinn       bad female    27
## 11    Supergirl      good female    24
## 12     Superman      good   male    29
## 13      Hellboy      good   male    10
```

Goodbye NAs! Let move on to Step 3, Tranform.

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:

```
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.

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

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

```
## # A tibble: 4 x 6
##           name alignment    sex   age flyingSpeedmph homePlanet
##          <chr>     <chr>  <chr> <dbl>          <chr>      <chr>
## 1        Storm      good female   300           1000      Earth
## 2         Thor      good   male    29          24000     Asgard
## 3 Wonder Woman      good female  5000      5,100,000 Themyscira
## 4     Superman      good   male    29      7,200,000    Krypton
```

```
#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 6
##           name alignment    sex   age flyingSpeedmph homePlanet
##          <chr>     <chr>  <chr> <dbl>          <chr>      <chr>
## 1        Storm      good female   300           1000      Earth
## 2         Thor      good   male    29          24000     Asgard
## 3 Wonder Woman      good female  5000      5,100,000 Themyscira
## 4     Superman      good   male    29      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 4
##           name alignment    sex   age
##          <chr>     <chr>  <chr> <dbl>
## 1        Storm      good female   300
## 2         Thor      good   male    29
## 3 Wonder Woman      good female  5000
## 4     Superman      good   male    29
```

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

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: 14 x 6
##            name alignment    sex   age flyingSpeedmph homePlanet
##           <chr>     <chr>  <chr> <dbl>          <chr>      <chr>
##  1  Black Widow      good female    38           <NA>       <NA>
##  2      Magneto       bad   male    93           <NA>       <NA>
##  3        Storm      good female   300           1000      Earth
##  4     Mystique       bad female   120           <NA>       <NA>
##  5         Thor      good   male    29          24000     Asgard
##  6         Loki       bad   male  1048           <NA>       <NA>
##  7 Wonder Woman      good female  5000      5,100,000 Themyscira
##  8       Batman      good   male    32           <NA>       <NA>
##  9        Joker       bad   male    46           <NA>       <NA>
## 10 Harley Quinn       bad female    27           <NA>       <NA>
## 11    Supergirl      good female    24           <NA>       <NA>
```

```
## 12     Superman      good   male    29      7,200,000    Krypton
## 13      Hellboy      good   male    10            <NA>       <NA>
## 14     Deadpool      <NA>   <NA>    NA               0      Earth
```

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

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

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

```
result
```

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

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

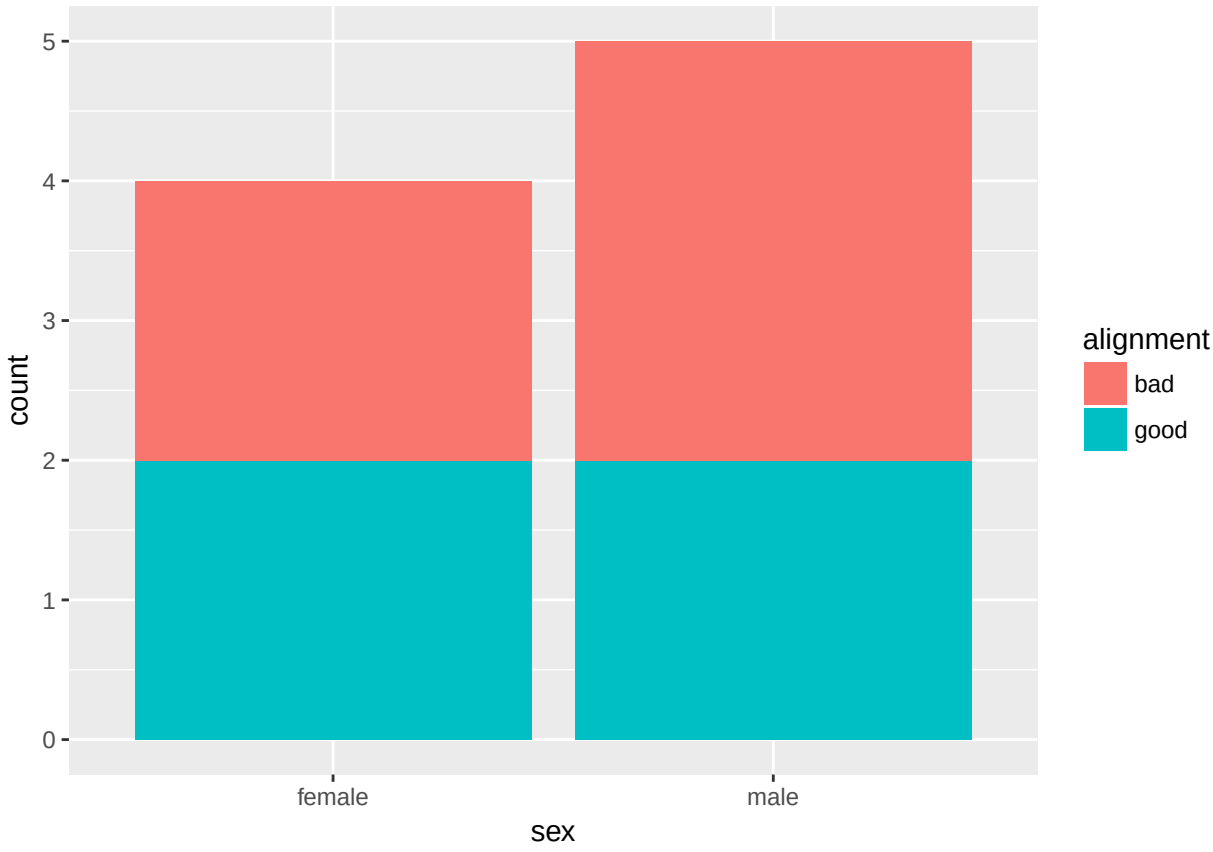
Visualize

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

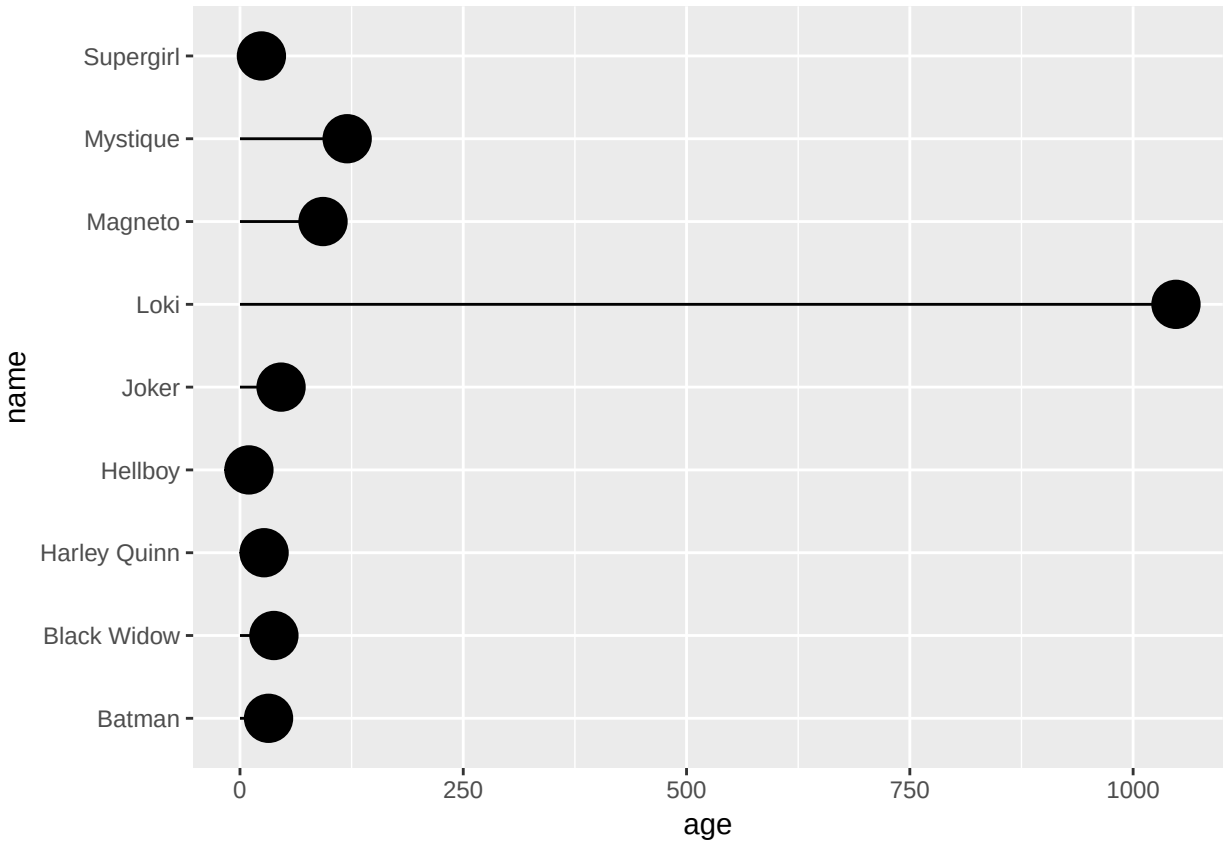
Let's compare alignment between the sexes.

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



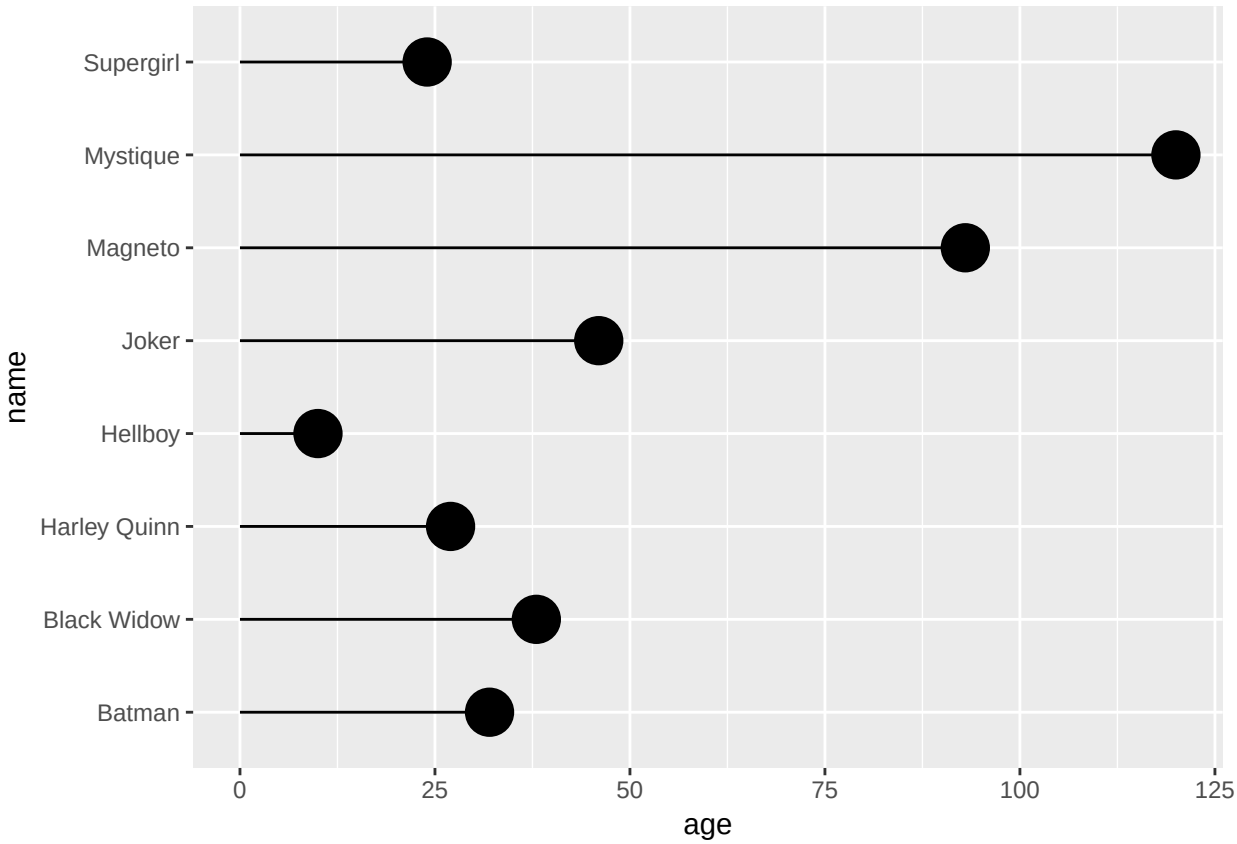
Lollipop charts

```
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()
```



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

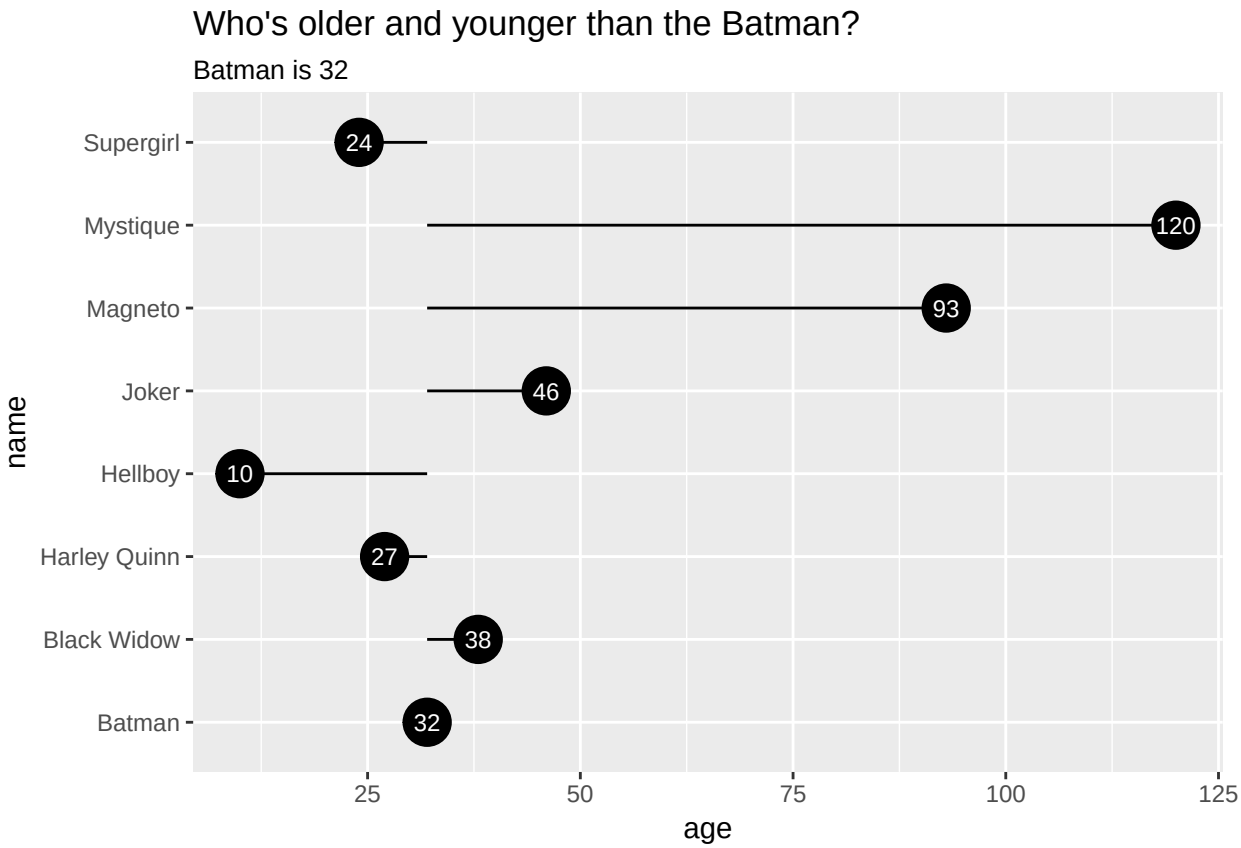
```
result %>% filter(name != "Loki") %>% ggplot(aes(x=name, y=age, label=age )) +
  geom_point(stat='identity', fill="black", size=8) +
  geom_segment(aes(y = 0,
                   x = name,
                   yend = age,
                   xend= name),
               color = "black") +
  coord_flip()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



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()
```



That's it for Workshop 1!