Using Pipes

For cleaner, more readable code

Mary van Valkenburg
@MaryLvV
R-Ladies Nashville
September 2017

<table>
<tr><td>Unix pipe:</td><td>|</td><td>(1973)</td></tr>
<tr><td>F# pipe:</td><td>|&gt;</td><td>(2005)</td></tr>
<tr><td><b>R pipe:</b></td><td><b>%&gt;%</b></td><td><b>(2014)</b></td></tr>
</table>

- introduced in November 2014
- **magrittr** package – Stefan Bache & Hadley Wickham

The Tidy Tools Manifesto:

1. Reuse existing data structures.
2. **Compose simple functions with the pipe.**
3. Embrace functional programming.
4. **Design for humans.**

https://cran.r-project.org/web/packages/tidyverse/vignettes/manifesto.html

```
my_data <- do_one_last_thing(
             do_something_else(
               do_something(data_in)
             )
           )
```

☹

```
temp_data_1<- do_something(data_in)
temp_data_2 <-do_something_else(temp_data_1)
my_data <- do_one_last_thing(temp_data_2)
```

☹

The pipe operator indicates that an object is “piped in” to a function or expression as its first argument:

```
object %>% function()
```

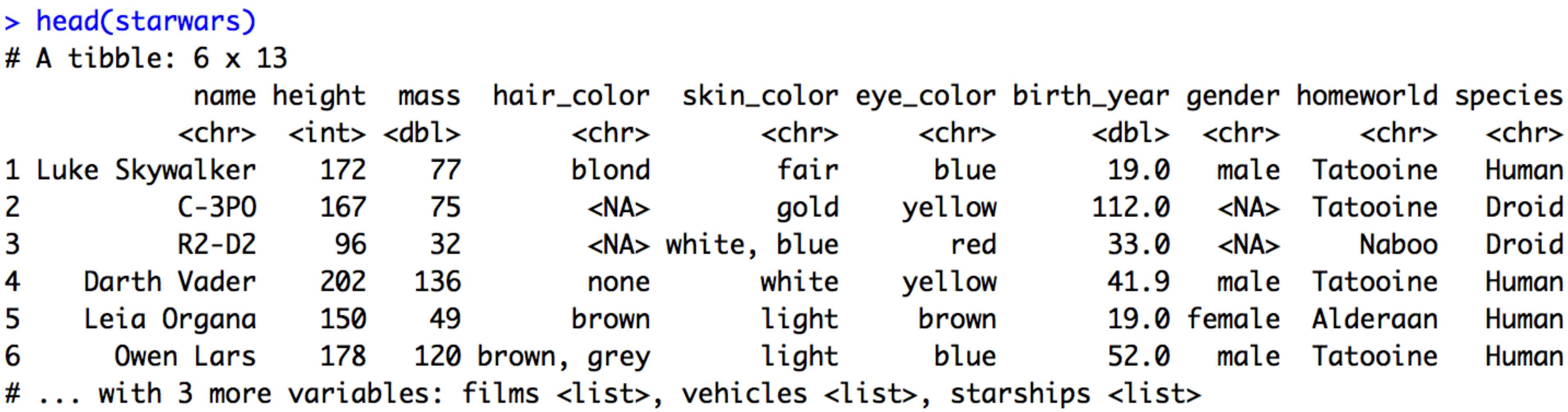
R functions are designed (mostly) with data as the first argument - which works really nicely with pipes:

```
my_data <- data_in %>%
              do_something() %>%
              do_something_else() %>%
              do_one_last_thing()
```

You’ve been given a task:

Create a list of 100 random, normally distributed numbers
with a mean of 0 and a standard deviation of 2,
that have been transformed to:
First - find their absolute values, and
Second - round to two decimal places.

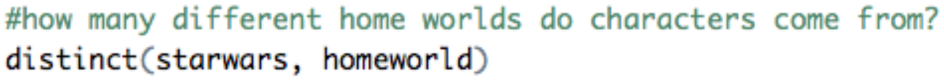
The starwars dataset - built from https://swapi.co/ - is included with dplyr 0.7.2



```
> head(starwars)
# A tibble: 6 x 13
            name height  mass  hair_color  skin_color eye_color birth_year gender homeworld species
           <chr>  <int> <dbl>       <chr>       <chr>     <chr>      <dbl>  <chr>     <chr>   <chr>
1 Luke Skywalker    172    77       blond        fair      blue       19.0   male  Tatooine   Human
2          C-3PO    167    75        <NA>        gold    yellow      112.0   <NA>  Tatooine   Droid
3          R2-D2     96    32        <NA> white, blue       red       33.0   <NA>     Naboo   Droid
4    Darth Vader    202   136        none       white    yellow       41.9   male  Tatooine   Human
5    Leia Organa    150    49       brown       light     brown       19.0 female  Alderaan   Human
6      Owen Lars    178   120 brown, grey       light      blue       52.0   male  Tatooine   Human
# ... with 3 more variables: films <list>, vehicles <list>, starships <list>
```

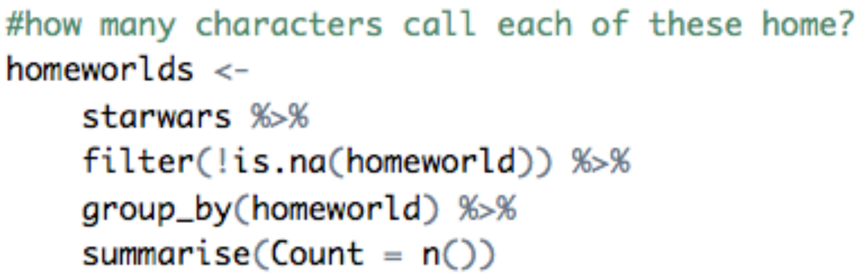
Let’s play with pipes!

How many different home worlds
do Star Wars characters come from?



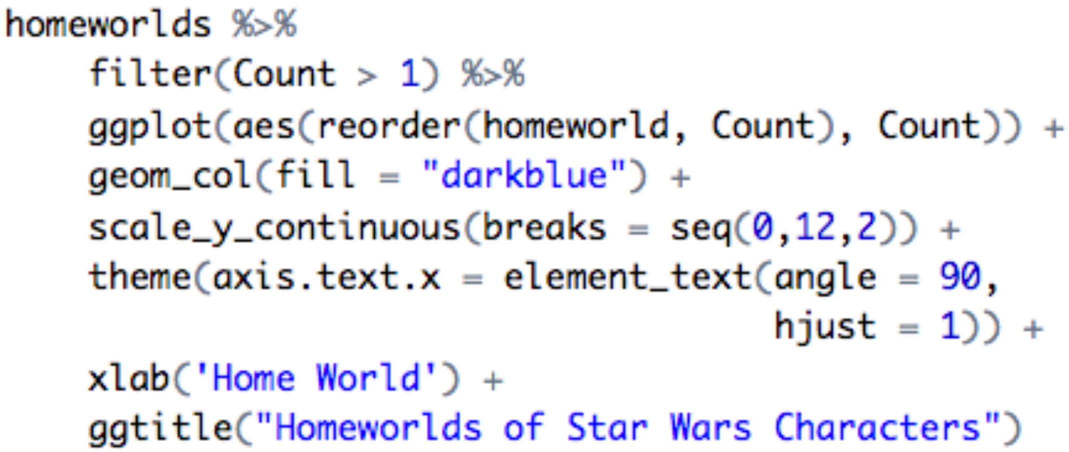
```
#how many different home worlds do characters come from?
distinct(starwars, homeworld)
```

49!!

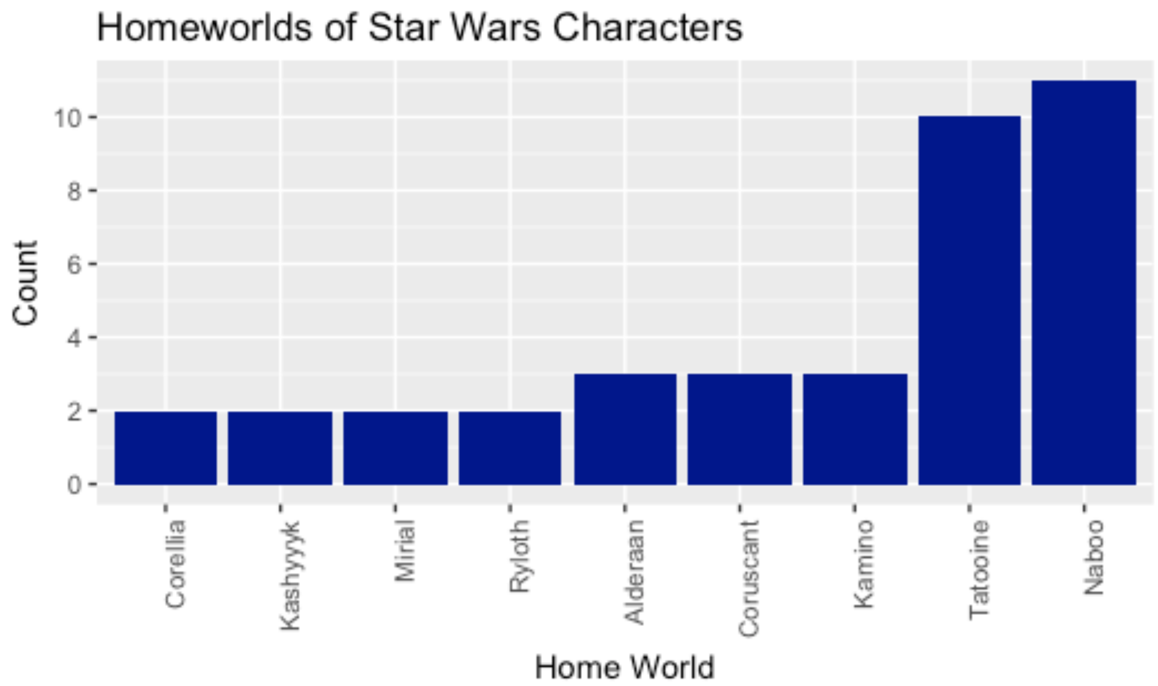


```
#how many characters call each of these home?
homeworlds <-
    starwars %>%
    filter(!is.na(homeworld)) %>%
    group_by(homeworld) %>%
    summarise(Count = n())
```

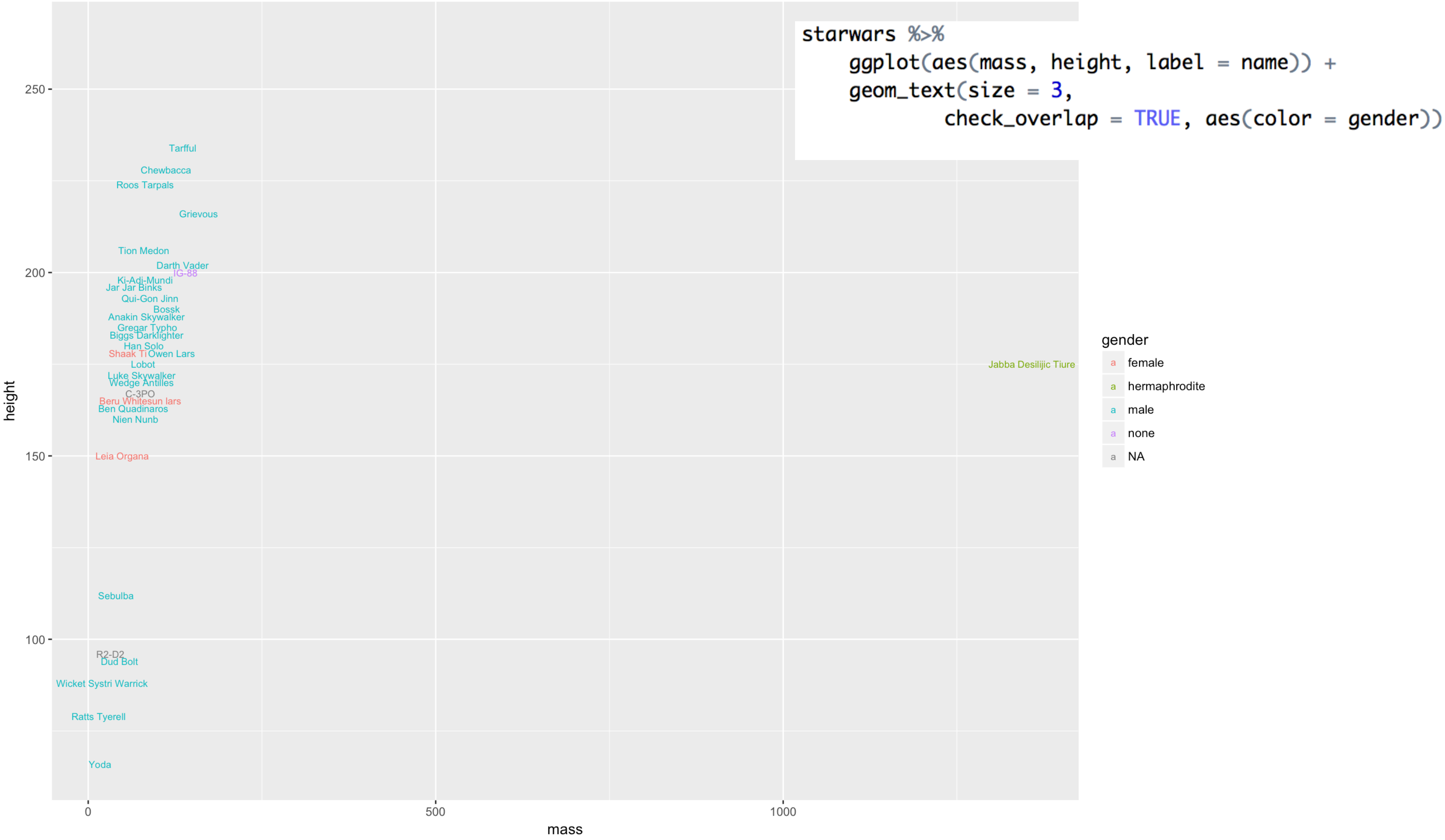
For worlds that two or more characters call home,
how many characters list that as their place of birth?



```
homeworlds %>%
    filter(Count > 1) %>%
    ggplot(aes(reorder(homeworld, Count), Count)) +
    geom_col(fill = "darkblue") +
    scale_y_continuous(breaks = seq(0,12,2)) +
    theme(axis.text.x = element_text(angle = 90,
                                     hjust = 1)) +
    xlab('Home World') +
    ggtitle("Homeworlds of Star Wars Characters")
```

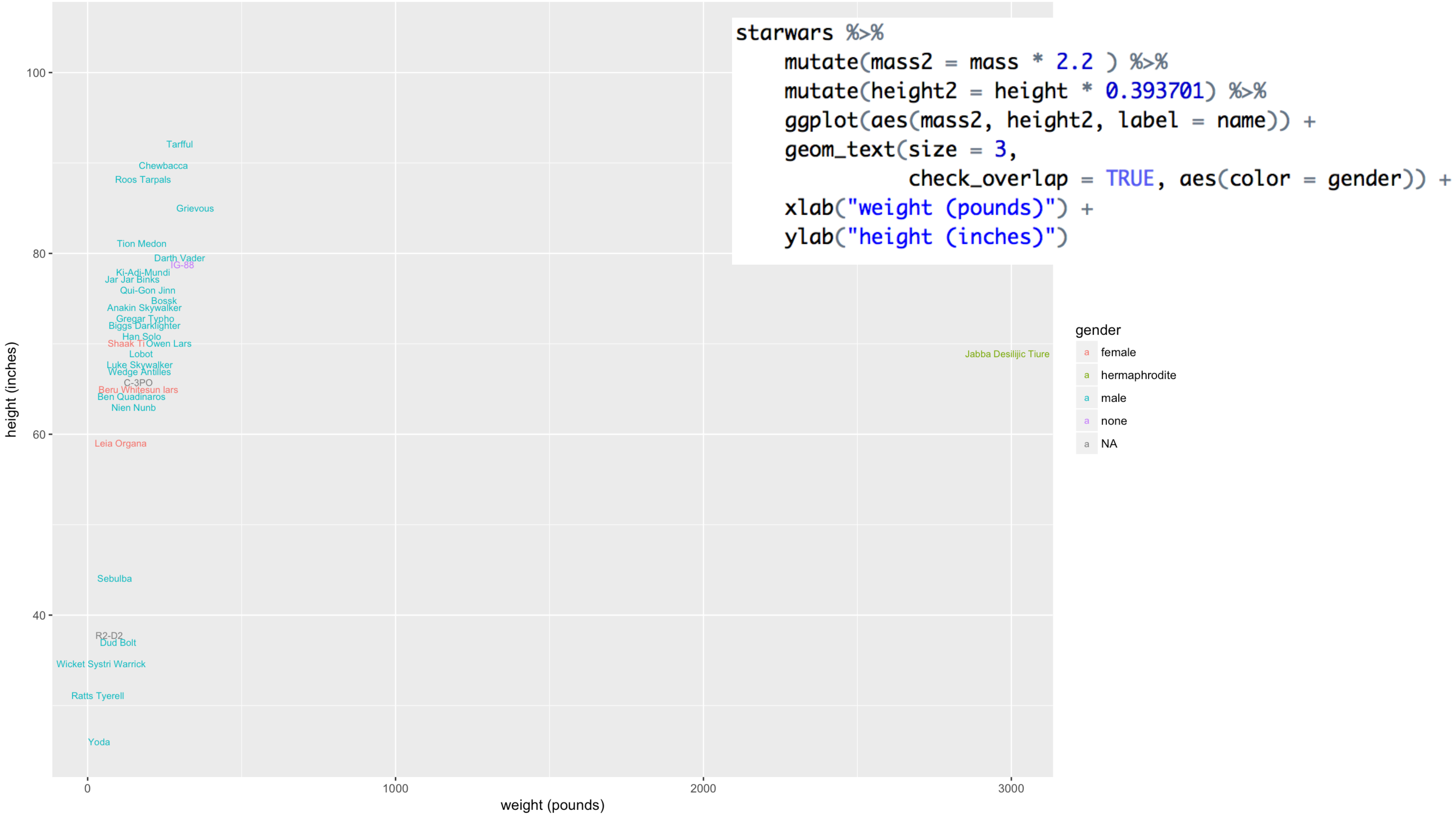


Which character is the largest? The smallest? Is it a male character? A female character? Neither?



```
starwars %>%
    ggplot(aes(mass, height, label = name)) +
    geom_text(size = 3,
            check_overlap = TRUE, aes(color = gender))
```

Metric is hard! What if I want to see inches and pounds?



```
starwars %>%
    mutate(mass2 = mass * 2.2 ) %>%
    mutate(height2 = height * 0.393701) %>%
    ggplot(aes(mass2, height2, label = name)) +
    geom_text(size = 3,
              check_overlap = TRUE, aes(color = gender)) +
    xlab("weight (pounds)") +
    ylab("height (inches)")
```

Magrittr has other pipe operators:

%<>% **Assigns the resulting value on the right hand side back to the left hand side:**

```
numbers <- rnorm(10)
numbers %<>%
   abs() %>%
   round(digits = 0)
```

%$% **Exposes the names in the left hand side to the expression on the right hand side:**

```
data.frame(z = rnorm(100)) %$% ts.plot(z)
```

%T>% **Pipes the left hand side to the right hand side expression, but returns the left hand side value:**

```
numbers <- rnorm(10)
numbers %T>% plot(numbers)
```

https://cran.r-project.org/web/packages/magrittr/magrittr.pdf

Why is this package called magrittr??

Naming things is hard.

The name for the package was inspired by Rene Magritte's painting 'A Treachery of Images.'

Ceci n'est pas une pipe.
Magritte

Magritte talks about the painting in this short video: https://www.sfmoma.org/watch/what-is-cheese-a-reality-lesson-from-rene-magritte/

For more on pipes in R, see Bob Rudis' RStudio Conference talk: https://www.rstudio.com/resources/videos/writing-readable-code-with-pipes/