The Tidyverse!

HiDef
AERIAL SURVEYING

Overview

- This is going to be a ‘play as we learn’ lecture!
- Intro to tidyverse – WHY?!
- Data import
- Data wrangling
- Functional programming (escape the for loop)
- Visualisation?? (coming up soon)

Install the Tidyverse

WHILE WE ARE TALKING…

```
install.packages('tidyverse')

library(tidyverse)

data(starwars)
```

The tidyverse

Components

The tidyverse is a collection of R packages that share common philosophies and are designed to work together. This site is a work-in-progress guide to the tidyverse and its packages.

Why use tidyverse instead of base R??

- When importing data, tidyverse functions are FASTER, SMARTER (imports dates, etc…), and do not convert strings to factors

- Pipelines ('%>%') make code faster to write (and 'tidier') and allows for logical stringing of tasks (more EFFICIENT)

- tidyverse tables allow for users to more efficiently import and work with BIG DATA

- tidyverse is a verified R library (long-term support by Rstudio team guaranteed)

- Let's get into it!... Data import

```
TASK 1:  IMPORT the csv 'tidyverse_dummy_import.csv' in
base R
    - Using:  sapply(x,class) – what are the column
classes?

TASK 2:  IMPORT the csv 'tidyverse_dummy_import.csv'
with the readr::read_csv function
    - What are the column classes??

TASK 3:  IMPORT the xlsx 'tidyverse_dummy_import.xslx'
with the readxl::read_xlsx function
    - What are the column classes??
```

- Let's get into it!... Data import

TASK 1:

```
Occurrence_field     Number_field    Integer_field       Text_field       Date_field
       "integer"        "numeric"        "integer"         "factor"         "factor"
```

TASK 2:

```
cols(
  Occurrence_field = col_double(),
  Number_field = col_double(),
  Integer_field = col_double(),
  Text_field = col_character(),
  Date_field = col_character()
)
```

TASK 3:

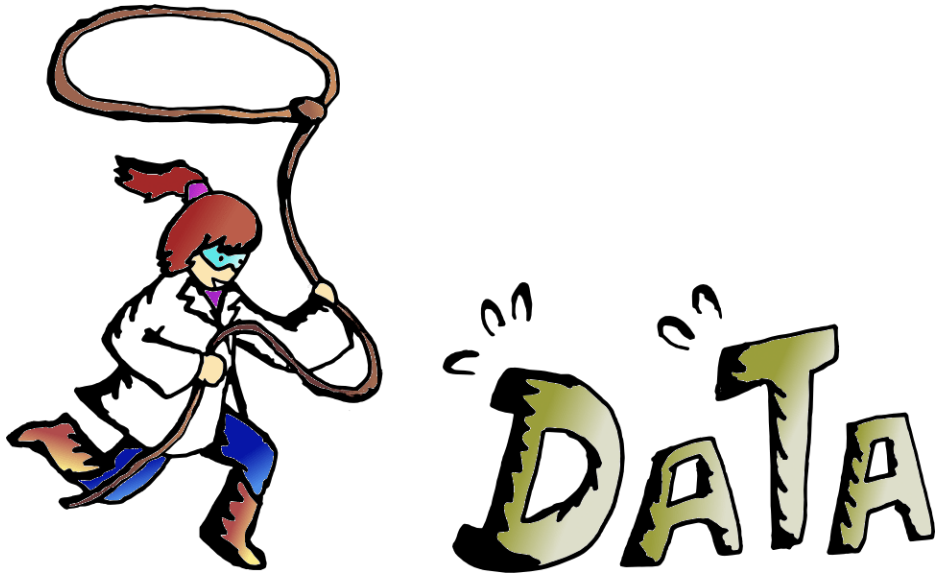
```
# A tibble: 16 x 5
   Occurrence_field Number_field Integer_field Text_field Date_field
              <dbl>        <dbl>         <dbl> <chr>      <dttm>
```

- Let's get into it!... Data import
  - you can set column types from the import

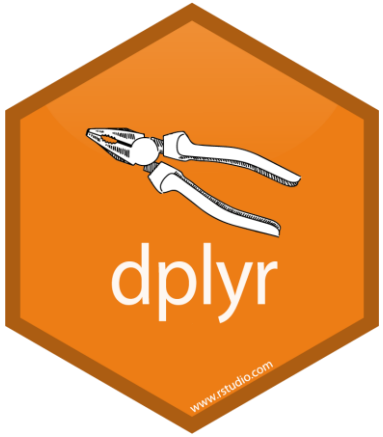
```
readr::read_csv('tidyverse_dummy_import.csv',
                          col_types=list(Occurrence_field=col_factor()))

readxl::read_xlsx('tidyverse_dummy_import.xlsx',
                          col_types=c("guess","numeric","numeric","text","date")))
```

Data wrangling

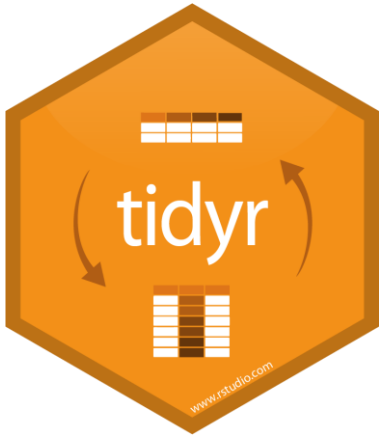


Data wrangling = the process of cleaning, organizing and structuring data for use in analysis and visualization



dplyr: Handles much of the data cleaning (filtering, selecting columns, summarizing, etc..)

Tibble: Table types used by most tidyverse libraries/functions



tidyr: Handles restructuring tibbles (tables) ; pivot tables, etc…

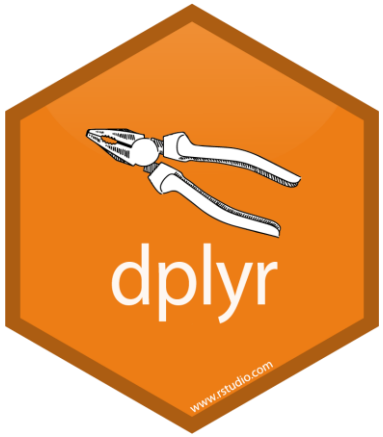
THE PIPELINE

the key to the tidyverse

%>%

```
TIBBLE OBJECT %>%
   Function1 %>%
   Function2 %>%
   Function3
```

Data wrangling



- **select():** Select columns from your dataset
- **filter():** Filter out certain rows that meet your criteria(s)
- **group_by():** Group different observations together such that the original dataset does not change. Only the way it is represented is changed in the form of a list
- **summarise():** Summarise data based on arguments
- **arrange():** Arrange your column data in ascending or descending order
- **join():** Perform left, right, full, and inner joins in R
- **mutate():** Create new columns by preserving the existing variables

```
starwars %>% dplyr::select(name,gender,homeworld)
```

```
starwars %>% dplyr::filter(homeworld=='Tatooine’)
```

```
starwars %>%
   dplyr::select(name,gender,homeworld) %>%
   dplyr::filter(homeworld==‘Tatooine’)
```

What are the r base equivalents?

Namespace prefixes

- some function names are used in multiple packages, e.g. both the dplyr and raster packages have a "select" function
- if we have both the raster and dplyr packages loaded, and we just write:

```
starwars %>% select(name,gender,homeworld)
```

we can't be certain which "select" function will be used. It will depend on the order in which the packages were loaded

- for such functions, it can therefore be useful to use the namespace prefix:

```
starwars %>% dplyr::select(name,gender,homeworld)
```

```
starwars %>%
    group_by(homeworld) %>%
    summarise(mean_height = mean(height))

starwars %>%
    group_by(homeworld) %>%
    summarise(mean_height = mean(height),
              sample_size = n())

starwars %>%
    group_by(homeworld,gender) %>%
    summarise(mean_height = mean(height),
              sample_size = n())

starwars %>%
    group_by(homeworld,gender) %>%
    summarise(mean_height = mean(height),
              sample_size = n()) %>%
    dplyr::filter(!is.na(mean_height))
```

Data wrangling - tasks

YOUR MISSION, should you choose to accept it…

Tasks:

1. What is the average mass of brown-haired humans in the Star Wars universe? Median mass? Standard deviation?

2. Who is the tallest non-human character? The shortest?

3. How many droids (in this dataset) were in “Attack of the clones”? ** Bonus

Data wrangling - tasks

1. What is the average mass of brown-haired humans in the Star Wars universe? Median mass? Standard deviation?

```
starwars %>%
  group_by(hair_color,species) %>%
  summarise(meanmass=mean(mass,na.rm = T),
            medmass=median(mass,na.rm = T),
            sdmass=sd(mass,na.rm = T)) %>%
  filter(species == 'Human',hair_color=='brown')
```

Mean = 75.5
Med = 78
Sd = 20.8

2. Who is the tallest non-human character? The shortest?

```
starwars %>%
  filter(species != 'Human') %>%
  arrange(height)
```

Yoda

```
starwars %>%
  filter(species != 'Human') %>%
  arrange(-height)
```

Yarael Poof

3. How many droids (in this dataset) were in “Attack of the clones”? ** Bonus

```
starwars %>%
  unnest(films) %>%
  count(species,films) %>%
  filter(species=='Droid',films=='Attack of the Clones')
```

2

```
starwars %>% count(homeworld,gender)
```

count() is equivalent to: group_by + summarise(n())

```
starwars %>% unnest(films)
```

unnest() is a TIDYR function that expands list columns

```
starwars %>% mutate(hm_ratio = height/mass)
```

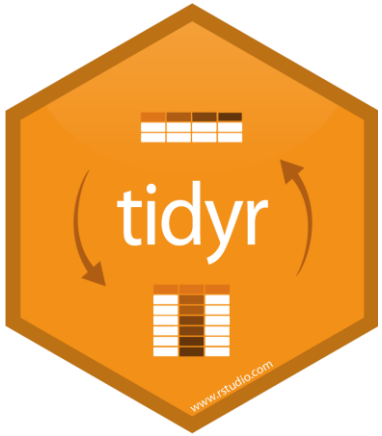
mutate() is a way to add new columns to a tibble

```
df1 <- tibble(
  category = c('a','a','b','c','c','a','b'),
  id = c(1,2,3,4,5,6,7)
)

df2 <- tibble(
  category = c('b','a','b','a','a','c','c'),
  id = c(7,6,5,4,3,2,1)
)

df1 %>% left_join(df2,by='id')
```

```
# A tibble: 7 x 3
  category.x    id category.y
  <chr>      <dbl> <chr>
1 a              1 c
2 a              2 c
3 b              3 a
4 c              4 a
5 c              5 b
6 a              6 a
7 b              7 b
```



- **gather():** The function “gathers” multiple columns from your dataset and converts them into key-value pairs

- **spread():** *This* takes two columns and “spreads” them into multiple columns

- **separate():** As the name suggests, this function helps in separating or splitting a single column into numerous columns

- **unite():** Works completely opposite to the *separate()* function. It helps in combining two or more columns into one

- **nest():** Takes a dataset with a repeating key and condenses the data into a column

- **unnest():** The opposite of nest(); will split out a list column into its elements

Gather: Going from WIDE to LONG

```
sst_data <- data.frame(
  time = as.Date('2009-01-01') + 0:9,
  unit_A = round(runif(10, 6, 9),2),
  unit_B = round(runif(10, 7, 10),2),
  unit_C = round(runif(10, 6, 10),2)
)

sst_data %>%
  gather(unit,sst,-time)
```

	time	unit_A	unit_B	unit_C
1	2009-01-01	7.61	8.61	7.51
2	2009-01-02	6.87	8.67	8.53
3	2009-01-03	8.65	8.42	6.47
4	2009-01-04	6.59	7.55	6.78
5	2009-01-05	8.17	9.06	9.36
6	2009-01-06	6.09	7.93	9.02

	time	unit	sst
1	2009-01-01	unit_A	7.61
2	2009-01-02	unit_A	6.87
3	2009-01-03	unit_A	8.65
4	2009-01-04	unit_A	6.59
5	2009-01-05	unit_A	8.17
6	2009-01-06	unit_A	6.09
7	2009-01-07	unit_A	6.66
8	2009-01-08	unit_A	8.72
9	2009-01-09	unit_A	8.28
10	2009-01-10	unit_A	7.09
11	2009-01-01	unit_B	8.61
12	2009-01-02	unit_B	8.67
13	2009-01-03	unit_B	8.42
14	2009-01-04	unit_B	7.55
15	2009-01-05	unit_B	9.06

Spread: Going from LONG to WIDE

```
sst_data %>%
  gather(unit,sst,-time) %>%
  spread(unit,sst)
```

	time	unit_A	unit_B	unit_C
1	2009-01-01	7.61	8.61	7.51
2	2009-01-02	6.87	8.67	8.53
3	2009-01-03	8.65	8.42	6.47
4	2009-01-04	6.59	7.55	6.78
5	2009-01-05	8.17	9.06	9.36
6	2009-01-06	6.09	7.93	9.02

	time	unit	sst
1	2009-01-01	unit_A	7.61
2	2009-01-02	unit_A	6.87
3	2009-01-03	unit_A	8.65
4	2009-01-04	unit_A	6.59
5	2009-01-05	unit_A	8.17
6	2009-01-06	unit_A	6.09
7	2009-01-07	unit_A	6.66
8	2009-01-08	unit_A	8.72
9	2009-01-09	unit_A	8.28
10	2009-01-10	unit_A	7.09
11	2009-01-01	unit_B	8.61
12	2009-01-02	unit_B	8.67
13	2009-01-03	unit_B	8.42
14	2009-01-04	unit_B	7.55
15	2009-01-05	unit_B	9.06

The nest: dataframes in columns??

```
nested_sst <- sst_data %>%
  gather(unit,sst,-time) %>%
  nest(unit,sst)
```

```
# A tibble: 10 x 2
   time       data
   <date>     <list>
 1 2009-01-01 <tibble [3 x 2]>
 2 2009-01-02 <tibble [3 x 2]>
 3 2009-01-03 <tibble [3 x 2]>
```

```
nested_sst$data[[1]]
```

```
# A tibble: 3 x 2
  unit     sst
  <chr>  <dbl>
1 unit_A  7.61
2 unit_B  8.61
3 unit_C  7.51
```

unnest() is the reverse

Why would we do this?? – functional programming :D

Another mission!

```
film.rankings <- tibble(
  films = c('The Phantom Menace','Attack of the Clones','Revenge of the Sith',
            'A New Hope','The Empire Strikes Back','Return of the Jedi',
            'The Force Awakens'),
  rank = c(7,6,5,3,4,1,2)
)
```

1. What films does the character Plo Koon appear in, ordered by rank? The final result should be a single table drawn from a list column. (**hint:** use these functions in order: select(), unnest(), left_join(),arrange(),nest(),filter() -> store in a variable called 'Plo.Koon')

2. Create a table that nests height and mass data for only humans in a column nested by film name. (**hint:** you will need unnest(), nest(), filter() and select()) -> store in a variable called 'nested.mass.height'

Data wrangling - tasks

What films does the character Plo Koon appear in, ordered by rank? The final result should be a single table drawn from a list column. (**hint**: use these functions in order: select(), unnest(), left_join(),arrange(),nest(),filter() -> store in a variable called 'Plo.Koon')

```
Plo.Koon <-starwars %>%
  select(name,films) %>%
  unnest(films) %>%
  left_join(film.rankings,by='films') %>%
  arrange(name,rank) %>%
  nest(films,rank) %>%
  filter(name == 'Plo Koon')
Plo.Koon$data[[1]]
```

```
# A tibble: 3 x 2
  films                      rank
  <chr>                     <dbl>
1 Revenge of the Sith           5
2 Attack of the Clones          6
3 The Phantom Menace            7
```

Data wrangling - tasks

Create a table that nests height and mass data for only humans in a column nested by film name.

(**hint**: you will need unnest(), nest(), filter() and select()) -> store in a variable called 'nested.mass.height'

```
nested.mass.height <- starwars %>%
  unnest(films)%>%
  select(name,height,mass,films,species) %>%
  filter(species=='Human')%>%
  nest(name,height,mass)
```

```
# A tibble: 7 x 3
  films                     species data
  <chr>                     <chr>   <list>
1 Revenge of the Sith       Human   <tibble [14 x 3]>
2 Return of the Jedi        Human   <tibble [11 x 3]>
3 The Empire Strikes Back   Human   <tibble [10 x 3]>
4 A New Hope                Human   <tibble [12 x 3]>
5 The Force Awakens         Human   <tibble [6 x 3]>
6 Attack of the Clones      Human   <tibble [17 x 3]>
7 The Phantom Menace        Human   <tibble [8 x 3]>
```

Functional programming

Functional programming through the tidyverse means we can quickly explore patterns in datasets!

For this last bit, we're going to work together! So code along.

Our task is going to be to look at the relationship between mass and height for humans in each of the 7 Star Wars films. (we'll keep it simple and use a linear model!)

```
map()-> This function will 'map' a function
across a list! Essentially replaces the for-
loop inside of tidyverse (similar to lapply).

X <- c(list(seq(1,50,5)),list(seq(40,100,2)))
lapply(X,mean)  # in base R
map(X,mean)     # in tidyverse
```

```
# The tibble we created in the last task

nested.mass.height$data[[1]]

# A tibble: 14 x 3
   name                height  mass
   <chr>                <int> <dbl>
 1 Luke Skywalker         172    77
 2 Darth Vader            202   136
 3 Leia Organa            150    49
 4 Owen Lars              178   120
 5 Beru Whitesun lars     165    75
 6 Obi-Wan Kenobi         182    77
 7 Anakin Skywalker       188    84
 8 Wilhuff Tarkin         180    NA
 9 Palpatine              170    75
```

Functional programming

```
nested.mass.height %>%
  mutate(
    model = map(
      data,~lm(mass~height,data=.x)
    ),
    slope = map_dbl(
      model, ~coef(.x)['height']
    ),
    r.sq = map_dbl(
      model,~glance(.x)$'r.squared'
    ),
    sample_size = map_dbl(
      data,~nrow(.x)
    ),
    tallest = map(
      data,~.x[which.max(.x$height),'name']
    ),
    heaviest = map(
      data,~.x[which.max(.x$mass),'name']
    )
  ) %>%
  unnest(tallest,heaviest) %>%
  rename(tallest=name,heaviest=name1)
```

Create new columns to store output

Run a linear model function over lists

Extract some model information (r squared, slope, sample size)

How would we get the tallest human in each film?

Now we tidy up the output

Functional programming

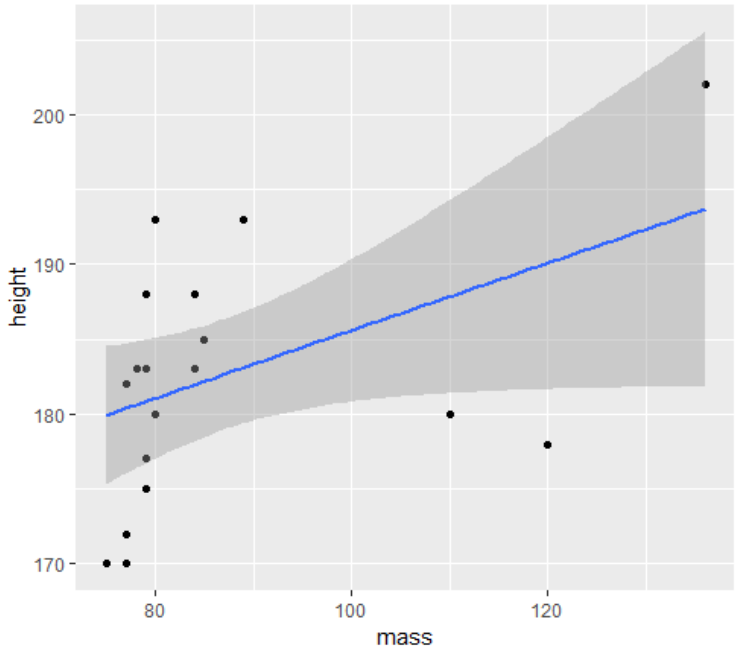
```
# A tibble: 7 x 9
  films                   species data              model    slope  r.sq sample_size tallest      heaviest
  <chr>                   <chr>   <list>            <list>   <dbl> <dbl>       <dbl> <chr>        <chr>
1 Revenge of the Sith     Human   <tibble [14 x 3]> <S3: lm> 1.21  0.494          14 Darth Vader  Darth Vader
2 Return of the Jedi      Human   <tibble [11 x 3]> <S3: lm> 1.48  0.820          11 Darth Vader  Darth Vader
3 The Empire Strikes Back Human   <tibble [10 x 3]> <S3: lm> 1.48  0.820          10 Darth Vader  Darth Vader
4 A New Hope              Human   <tibble [12 x 3]> <S3: lm> 1.38  0.557          12 Darth Vader  Darth Vader
5 The Force Awakens       Human   <tibble [6 x 3]>  <S3: lm> 1.08  0.971           6 Han Solo     Han Solo
6 Attack of the Clones    Human   <tibble [17 x 3]> <S3: lm> 0.718 0.157          17 Dooku        Owen Lars
7 The Phantom Menace      Human   <tibble [8 x 3]>  <S3: lm> 1.25  0.771           8 Qui-Gon Jinn Qui-Gon Jinn
```

Briefly: Visualisation

%>% usage looks very similar to the usage in ggplot2!

That's because it is!! You can pipe ggplot2 directly from any tidyverse tibbles!

```
starwars %>%
  filter(species=='Human',gender=='male') %>%
  ggplot(aes(x=mass,y=height))+
    geom_point()+
    geom_smooth(method=lm)
```



MORE ON

VISUALISATIONS

LATER!!!

QUESTIONS?