

R you ready?

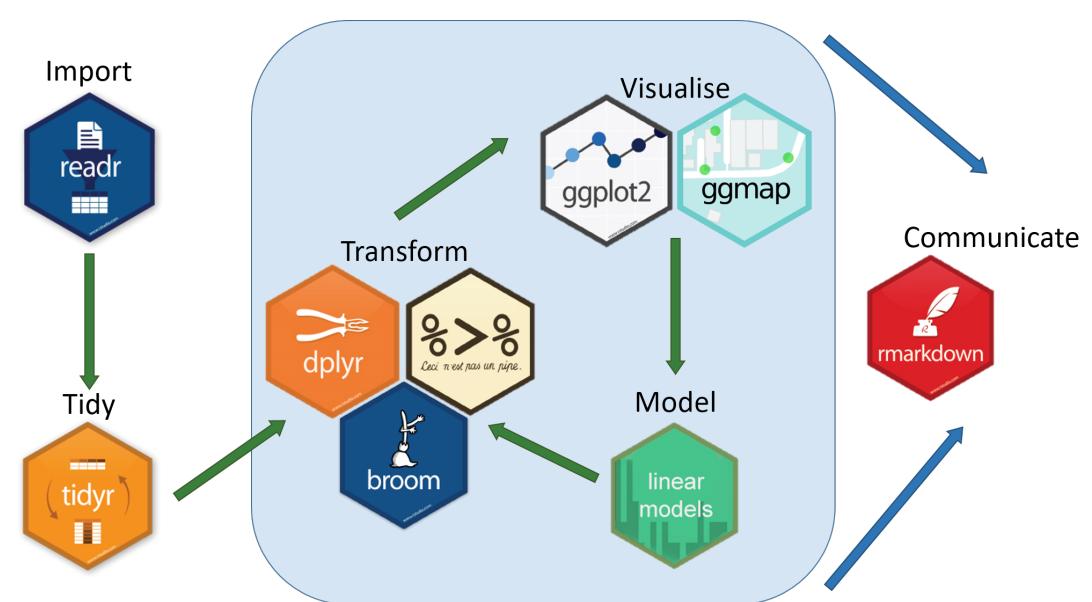
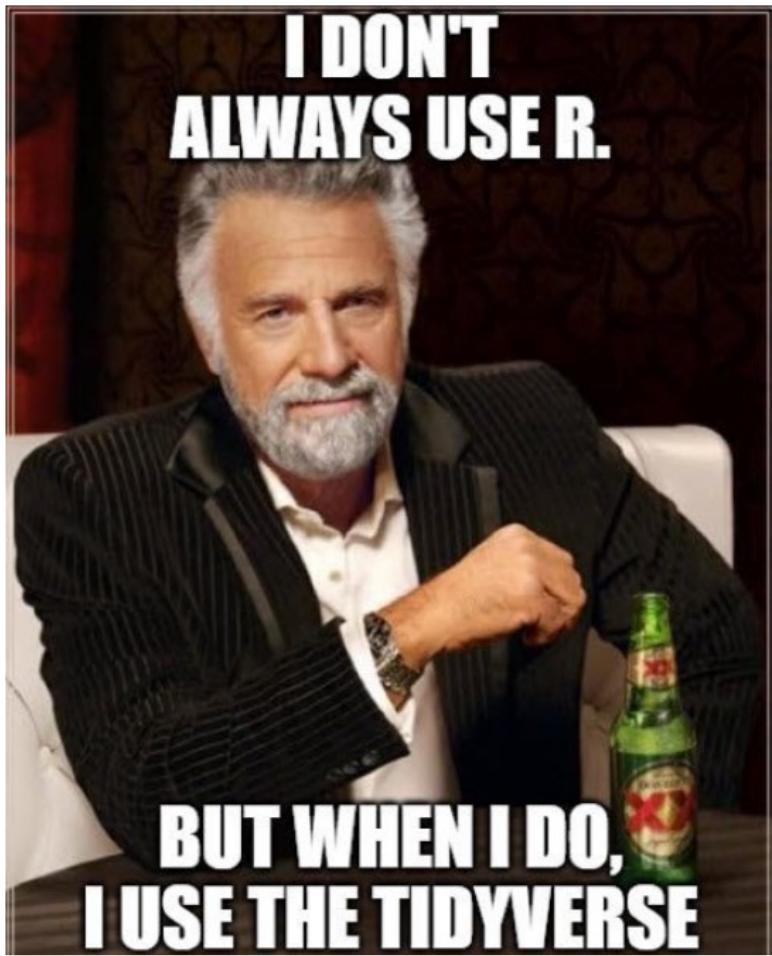
**IntRo to RStudio and R Markdown
for open data and reproducibility**

Unit 4:

**The tidier the better: Basics of coding
with the Tidyverse**

Mason A. Wirtz





Installing the tidyverse

Let's go ahead and install the package tidyverse

```
> install.packages("tidyverse")
> library(tidyverse)
```

The tidyverse package is a handy way of installing and loading a lot of different packages at the same time

You could also just load in the individual packages from the tidyverse package, like so

```
> library(tibble)
> library(readr)
> library(dplyr)
> library(magrittr)
> library(ggplot2)
```

Why tidyverse?

Stuff we need to do with data (frames)

tidy up a data frame

create new variables

calculate summary statistics

extract model outputs



Tibbles

Load in the data frame Vampires

```
> Vampires = read.csv("Vampires.csv")  
> Vampires
```

	idVampire	gender	ageOfVampire	deadOrAlive	hasFangs	bornIn	visitedCities	numberOfChildren
1	1	Male	85	Dead	Yes	South America	107	1
2	2	Female	73	Alive	No	Australia	66	3
3	3	Male	100	Alive	Yes	Australia	15	8
4	4	Female	75	Alive	No	Antarctica	11	2
5	5	Male	101	Alive	Yes	Australia	11	2
6	6	Female	87	Dead	Yes	North America	19	4
7	7	Male	82	Alive	No	North America	83	6
8	8	Female	68	Dead	Yes	Australia	50	5
9	9	Female	99	Dead	No	Australia	7	5
10	10	Female	44	Alive	Yes	Australia	66	1

Tibbles

And then coerce it into a tibble

```
> tibble(Vampires)
```

```
> tibble(Vampires)
# A tibble: 100 x 9
  idVampire gender ageOfVampire deadOrAlive hasFangs bornIn      visitedCities numberOfChildren numberChangedToVamp
  <fct>     <chr>    <dbl> <chr>       <chr>      <chr>      <dbl>          <dbl>           <dbl>
1 1          Male      85   Dead        Yes        South America 107            1             16
2 2          Female    73   Alive       No         Australia   66             3              6
3 3          Male      100  Alive       Yes        Australia   15             8              4
4 4          Female    75   Alive       No         Antarctica 11             2              4
5 5          Male      101  Alive       Yes        Australia   11             2              7
6 6          Female    87   Dead        Yes        North America 19             4             10
7 7          Male      82   Alive       No         North America 83             6              9
8 8          Female    68   Dead        Yes        Australia   50             5             16
9 9          Female    99   Dead        No         Australia   7              5             11
10 10        Female    44   Alive       Yes        Australia   66             1              3
# ... with 90 more rows
```

	idVampire	gender	ageOfVampire	deadOrAlive	hasFangs	bornIn	visitedCities	numberOfWorks	numberOfWorks
1	1	Male	85	Dead	Yes	South America	107	1	1
2	2	Female	73	Alive	No	Australia	66	3	3
3	3	Male	100	Alive	Yes	Australia	15	8	8
4	4	Female	75	Alive	No	Antarctica	11	2	2
5	5	Male	101	Alive	Yes	Australia	11	2	2
6	6	Female	87	Dead	Yes	North America	19	4	4
7	7	Male	82	Alive	No	North America	83		
8	8	Female	68	Dead	Yes	Australia			
9	9	Female	99	Dead	No	Australia			
10	10	Female	44	Alive	Yes	Australia			

What differences do we see?

Tibbles

Modern take on data frames:

Tibbles **default to character vectors** (rather than factor vectors) → character vectors easier to manipulate

Only first 10 rows are displayed, saving `head()` function calling time

Tibbles **display row and column numbers** (saves `nrow()` and `ncol()` function calls)

Tibble automatically **displays how each column is coded**

These are some small changes,
but they save you a lot of time
and typing in the long run.

Loading in tibbles from the getgo

Save yourself turning each data frame into a tibble:

**Use the `read_csv()` function from the
readr package.**

```
> Vampires = read_csv("Vampires.csv")
```

Manipulating data (the good kind)

Useful functions in dplyr

filter()

select()

mutate()

group_by()

summarize()

summarize_at()

*absolute
powerhouse in
manipulating data*

Operators in data manipulation

Operator	Description
<	Less than
>	Greater than
\leq	Less than or equal to
\geq	Greater than or equal to
$=$	Equal to
\neq	Not equal to

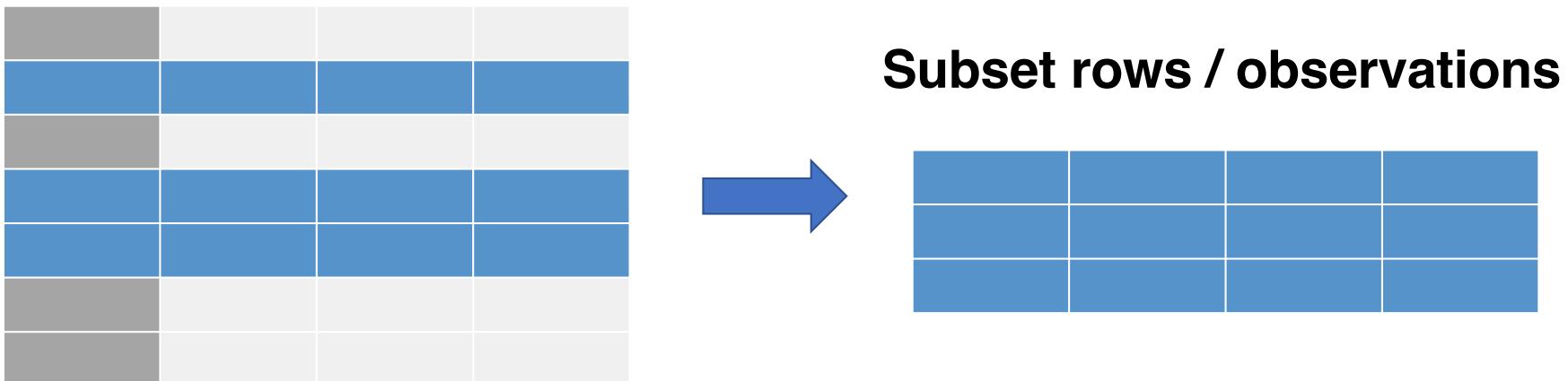
These operators
are useful when
manipulating data



NOT to be confused with the assignment
operator (only one =)

Manipulating data: filter()

Imagine we are working with our Vampires data frame and we want to take a closer look at all the **vampires who are older than 100**



```
> Vampires %>%  
  filter(ageOfVampire > 100)
```

Manipulating data: filter()

Tibble (data frame)

```
> Vampires %>%  
  filter(ageOfVampire > 100)
```

Operator + condition

The pipe:
*String together
functions*

Function

Variable in
tibble

Manipulating data: filter()

We can also **filter** the data frame according to two terms:

We want a data frame of **vampires** who are older than 100
AND **still alive**

```
> Vampires %>%  
  filter(ageOfVampire > 100, deadOrAlive == "Alive")
```



When subsetting CHARACTER vectors,
make sure to put the **condition in quotation marks!!**



Manipulating data: select ()

If we want only **select COLUMNS** in a data frame

Say we want to select **ONLY** idVampire, gender **and** ageOfVampire



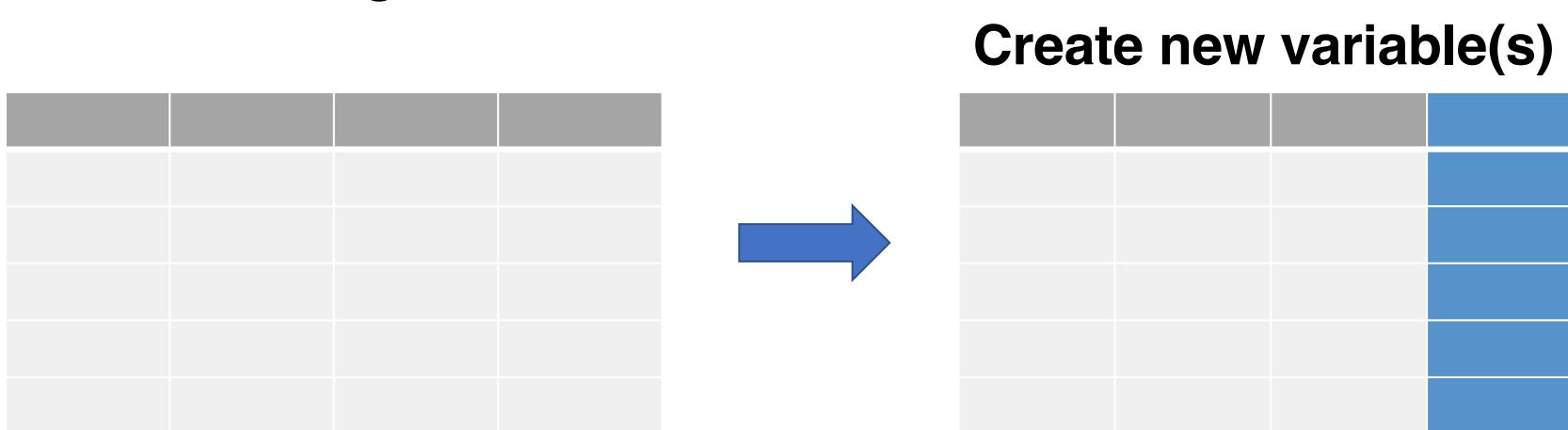
Subset columns


```
> Vampires %>%  
  select(idVampire, gender, ageOfVampire)
```

Manipulating data: `mutate()`

Create **new variables** that are **functions of existing variable(s)**

Since the variable `visitedCities` is a count variable, maybe we want to log transform the variable



```
> Vampires %>%  
  mutate(visitedCitiesLogged = log(visitedCities))
```

Manipulating data: mutate()

Tibble (data frame)



```
> Vampires %>%  
  mutate(visitedCitiesLogged = log(visitedCities))
```

Function
↑

NEW
variable name
↑

Assignment
operator
↑

Variable from
Vampires
data frame
↑

Manipulating data: summarize()

Collapses a data frame to a single row and summarizes it according to the argument we supply it with

Say we want the mean age of all vampires

ageOfVampire
1
2
3
4
5



Summarize a group/variable

mean
3

```
> Vampires %>%
```

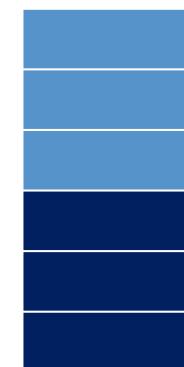
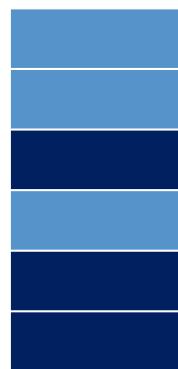
```
  summarize(mean = mean(ageOfVampire))
```

Manipulating data: group_by()

Group variables consisting of factors and summarize these grouped factors

We want to know the **mean age of the male and female vampires**

Group variables by unique values



```
> Vampires %>%
  group_by(gender) %>%
  summarize(mean = mean(ageOfVampire))
```

LET'S GET OUR HANDS DIRTY



makeameme.org