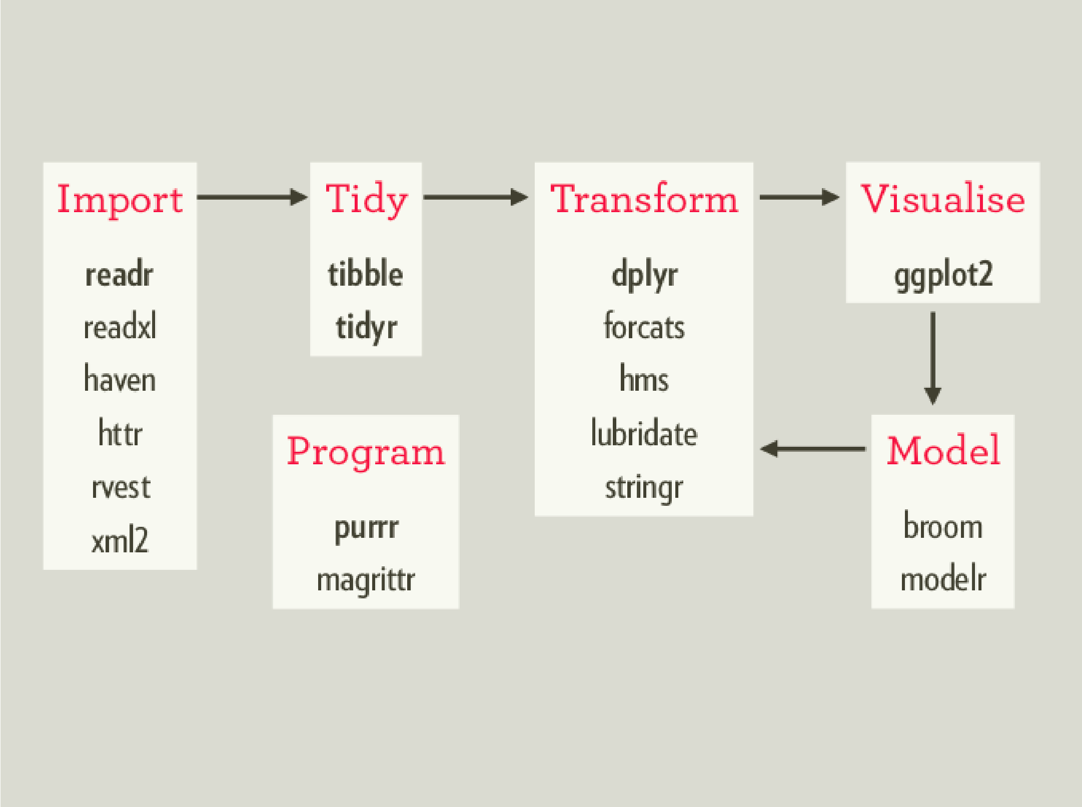
MARK OKELLO

@okello_mark

28/04/2018

DATA WRANGLING AND VISUALIZATION USING TIDYVERSE PACKAGE

Overview



H. Wickham - R for data science, licence CC

Tidy

Tibble

```
library(tibble)

*
```

Why tibble

i. No creation of row names

ii. Printing

claims

No conversion of strings to factors

No changes with names of variables

Tidy data with tidyr

Tidy datasets are all alike, but every messy dataset is messy in its own way - Hadley Wickham

- What is tidy data?

```
uaceexams <- as_tibble(read.csv("uaceexams.csv", stringsAsFactors =
    F,check.names = F))
```

- Is our dataset tidy?

```
library(tidyr)
```

- More like the familiar reshape2 package but the difference is its simplicity

Part 1

Gathering – Happens when some **column names** are not variable but values of variables.

To gather those columns into a new pair of variables Soln. gather() **aka** melt() in our reshape2 library

```
library(magrittr)

uaceexams %>% gather(`2011`, `2012`, `2013`, `2014`,`2015`,key = "year", value = "Students")

newtestuace <- uaceexams %>% gather(`2011`, `2012`, `2013`, `2014`,`2015`,key = "year", value = "Students")
```

Part 2

Spreading – Rows are not observations

```
women_abused <- as_tibble(read.csv("sexualviolence.csv",stringsAsFactors =
    F, check.names = F))
```

```
women_abused
```

Is our data tidy?

```
women_a <- spread(women_abused, key = Type, value = Count)
```

* Summary of gather() and spread()

- Others functions in tidyr package are spread and unite

Pipe (%>%)

Pipes help us write code in a way that is easier to read and understand

- Lets see.

function composition would be

```
  walk_outbox(
     pay_money(
         masawo(
             board_taxi()
                )
                  ,-- )
      ,-- )
```

```
board_taxi() %>%masawo() %>%pay_money()%>%walk_outbox()
```

Pipes don’t work:

- Functions that use the current environment example

```
   assign("x", 10)
 "x" %>% assign(100)
```

- Functions that use lazy evaluation

example of such function tryCatch()

But to assign use %<>%

*assign example adopted from H. Wickham- R for data science

When not to use a pipe

- Greater than 10 of its assignments
- expressing complex relationships example directed graphs
- ggplot2 functions
- multiple inputs or outputs

Mapping with Purrr

- The benefit of using **map()** function to the **for loop** is not speed, but clarity: it makes your code easier to write and to read.

*There is one function for each type of output

example

```
library(purrr)

testmap <- tibble(a=rnorm(15),b = rnorm(15), c= rnorm(15),d = rnorm(15))


map_dbl(testmap, mean)
```

Alternatively

```
testmap %>% map_dbl(mean)
```

dplyr package

We shall look at:

- Filter() - pick observations based on their values
- Arrange() - reordering rows based on a certain criteria
- Select() - pick variables based on their values
- Mutate() - create a new variable from the existing ones
- Summarize() - summarize values to a single value
- Joins and grouping data

filter()

Lets use our previous women abused - assume we want to filter where **percentage > 23.5**

```
women_a <- spread(women_abused, key = Type, value = Count)

filter(women_a, Percentage > 23.5)

filter(women_a, women_a$Percentage > 23.5)
```

lets load our library and see the difference

```
library(dplyr)

filter(women_a, Percentage > 23.5)
```

arrange() and select()

Assume we want to **order** our data based on **highest** abused women

```
arrange(women_a, desc(Percentage))
```

```
head(arrange(women_a, desc(Percentage)))
```

```
tail(arrange(woman_a, desc(Percentage)))
```

What could be the problem? **Resume the discussion later**

What if we want variables which only start with **P**. Is it possible?

```
select(women_a, starts_with("P"))
```

* Note select() works on variables not observations

other functions you can use with select are: contains(" "), ends_with(""), matches(" "), num_range("A", 1:5)

Mutate() and summarize()

Lets create a new variable called Mens_percentage

```
mutate(women_a, Mens_percentage = (1/3)*Percentage)
```

- Mutate creates a new variable, you can create many variables at ago

```
women_a%>% summarise(Total = sum(Women))
```

* either summarise() or summarize() work the same

Joining datasets

```
men_a <- as_tibble(read.csv("men_abuse.csv",stringsAsFactors = F,
    check.names = F))
```

```
bind_cols(women_a,men_a)
```

```
bind_rows(women_a,men_a)
```

To see it better lets look at it as a dataframe

```
as.data.frame(bind_rows(women_a,men_a))
```

Joins continue……

```
union(women_a,men_a)
```

Tired!

```
left_join(women_a,men_a, by= c("Period","Background","Particulars"))
```

Is it what we want?

```
sexviolence <-
    left_join(women_a,men_a,by=c("Period","Background","Particulars"))
```

```
inner_join(women_a,men_a, by= c("Period","Background","Particulars"))
```

```
#Try semi_join() and anti_join()
```

Factors with forcats

forcats provides tools for dealing with categorical variables

```
library(forcats)

monthvector <- c("Feb", "Apr", "Jan", "Mar")

sort(monthvector)
```

What has happened?

```
months <- c("Jan", "Feb", "Mar", "Apr", "May", "Jun","Jul", "Aug", "Sep", "Oct", "Nov", "Dec")

factor(monthvector, levels = months)

sort(factor(monthvector, levels = months))
```

Factor reordering and modification

```
ggplot(women_a, aes(Percentage, fct_reorder(Particulars, Percentage))) +
    geom_point()
```

```
ggplot(men_a, aes(Percent_Men, fct_reorder(Particulars, Percent_Men))) +
    geom_point()
```

```
women_a%>%mutate(Particulars= fct_collapse(Particulars,Western = c("Bakiga",
    "Bakonzo", "Banyankore", "Banyoro","Batoro"), Eastern = c("Basoga",
    "Iteso","Bagisu"),Northern = c("Acholi", "Lango"), WestNile = c("Alur",
    "Lugbara"), Other = "Others"))
```

```
women_a%>%mutate(Particulars= fct_collapse(Particulars,Western = c("Bakiga",
    "Bakonzo", "Banyankore", "Banyoro","Batoro"), Eastern = c("Basoga",
    "Iteso","Bagisu"),Northern = c("Acholi", "Lango"), WestNile = c("Alur",
    "Lugbara"), Other = "Others")) %>%count(Particulars)
```

Part 2: Visualization using ggplot2

The greatest value of a picture is when it forces us to notice what we never expected to see.- John Tukey

Aesthetics – Visual characteristics that can be mapped to data(**color, fill, shape, size and alpha**)

```
ggplot(data = newtestuace) +     geom_point(mapping = aes(x = Gender, y = Students, color= year), na.rm = T)
```

What do you see about our data?

Try other aesthetics

Facets- subplots that each display one subset of the data

```
ggplot(data = newtestuace) +     geom_point(mapping = aes(x = Gender, y =
    Students)) +     facet_wrap(~ year, nrow = 2)
```

In case there is need to facet two variable, use **facet_grid()**

Position Adjustment

```
ggplot(data = newtestuace) +    geom_bar(mapping = aes(x = Students, fill =
    year))
```

```
newtestuace1 <- filter(newtestuace, Students < 100)
```

```
ggplot(data = newtestuace1) +     geom_bar(mapping = aes(x = Students, fill =
    Gender), position = "dodge")
```

what do you see from our graph

Interpreting boxplots

```
ggplot(data = newtestuace, mapping = aes(x = year, y = Students)) + geom_boxplot()
```

Summary: Layered grammar of graphics

```
ggplot(data = <DATA>) +
<GEOM_FUNCTION>(
mapping = aes(<MAPPINGS>),
stat = <STAT>,
position = <POSITION>
) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION>
```

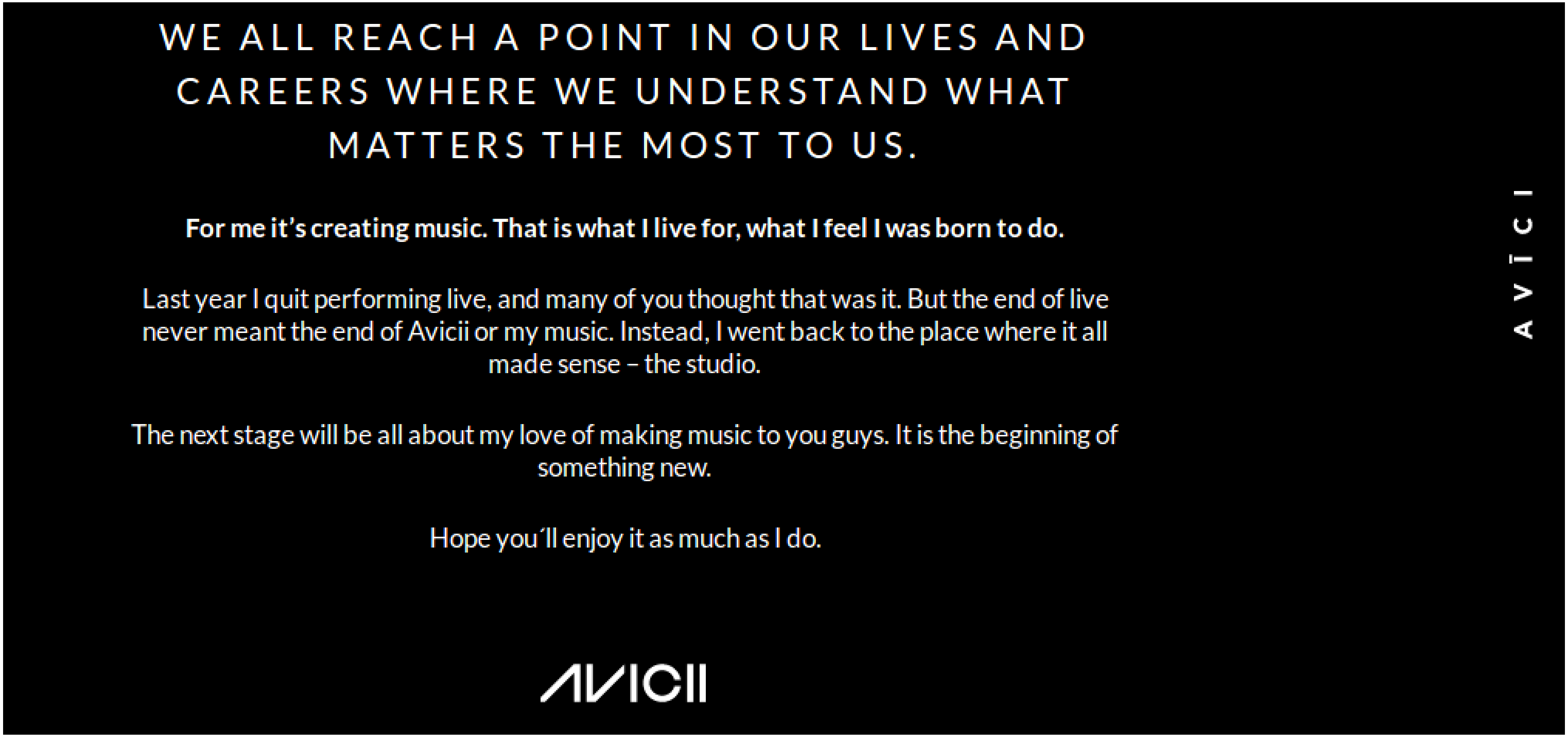
5 MINS DISCUSSION ABOUT OUR DATA

QUESTIONS?

Resources for more learning

- Doing data science from the front line
- Hands on Programming with R
- R for data science

Tips



WE ALL REACH A POINT IN OUR LIVES AND CAREERS WHERE WE UNDERSTAND WHAT MATTERS THE MOST TO US.

For me it’s creating music. That is what I live for, what I feel I was born to do.

Last year I quit performing live, and many of you thought that was it. But the end of live never meant the end of Avicii or my music. Instead, I went back to the place where it all made sense – the studio.

The next stage will be all about my love of making music to you guys. It is the beginning of something new.

Hope you´ll enjoy it as much as I do.