R for psychologists

Alejandro de la Vega

What is R?

- R is an free & powerful, interactive statistical programming language

Why R?

FREE as in freedom and free beer

It is a *programming language*

Powerful

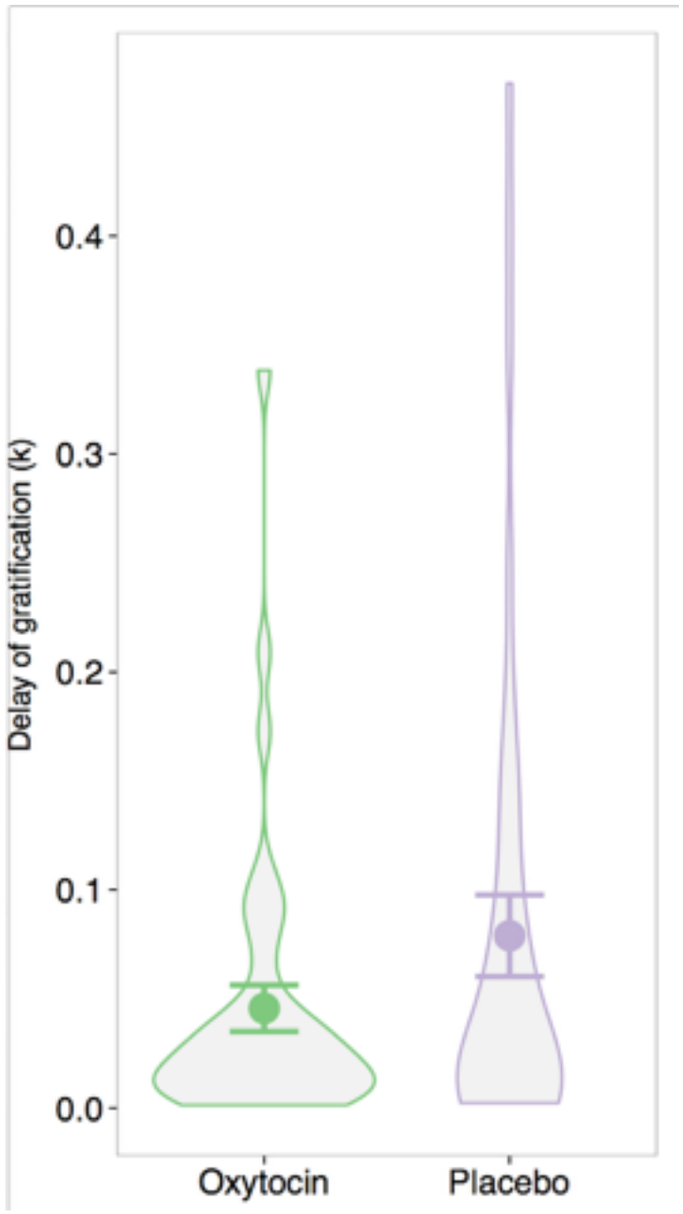
- "There are very few things that SAS or SPSS will do that R cannot, while R can do a wide range of things that the others cannot."
  - Robert A. Muenchen
    Author, R for SAS and SPSS Users

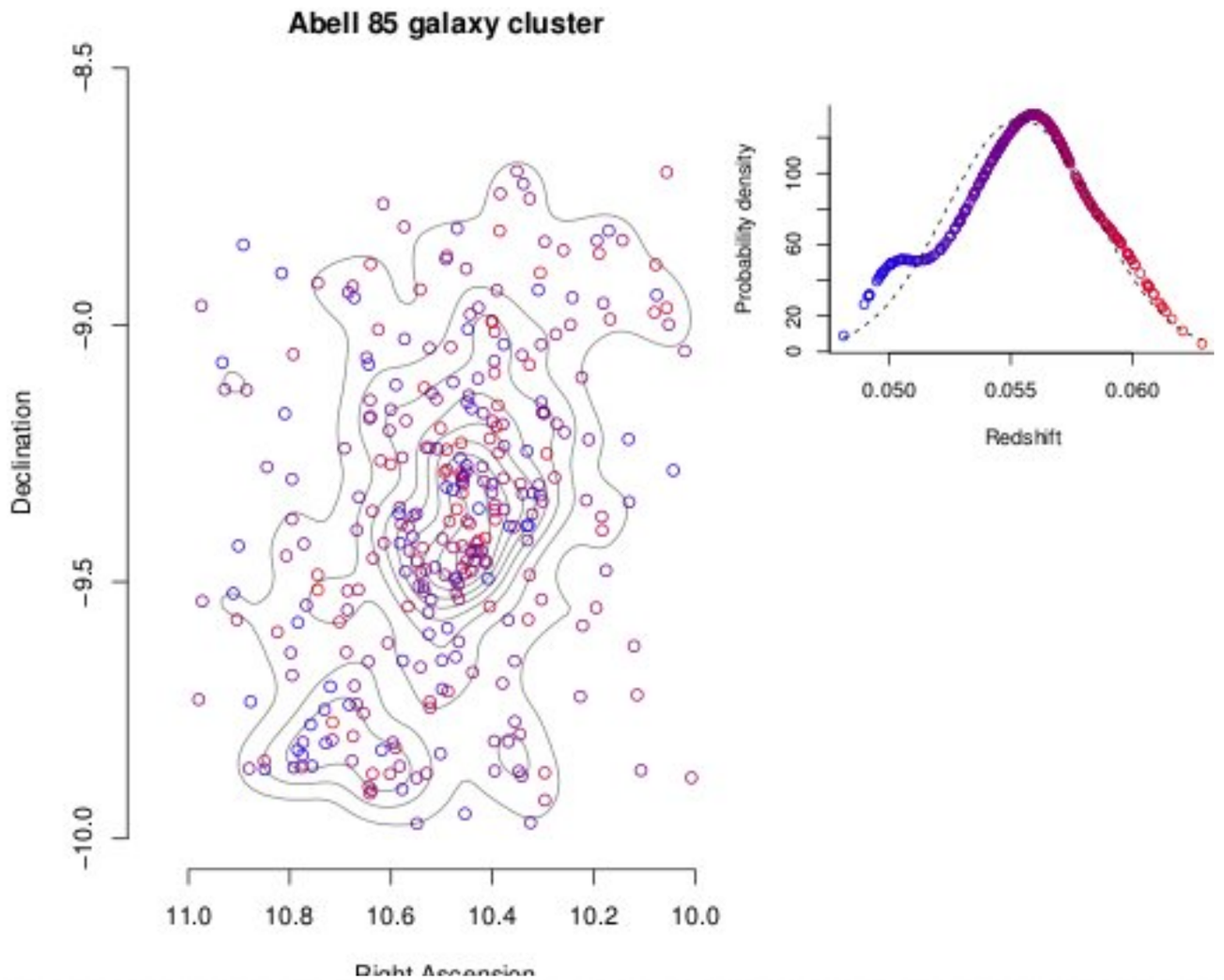
- Thanks to being a fully-fledged programming language
  - Could theoretically program anything in R

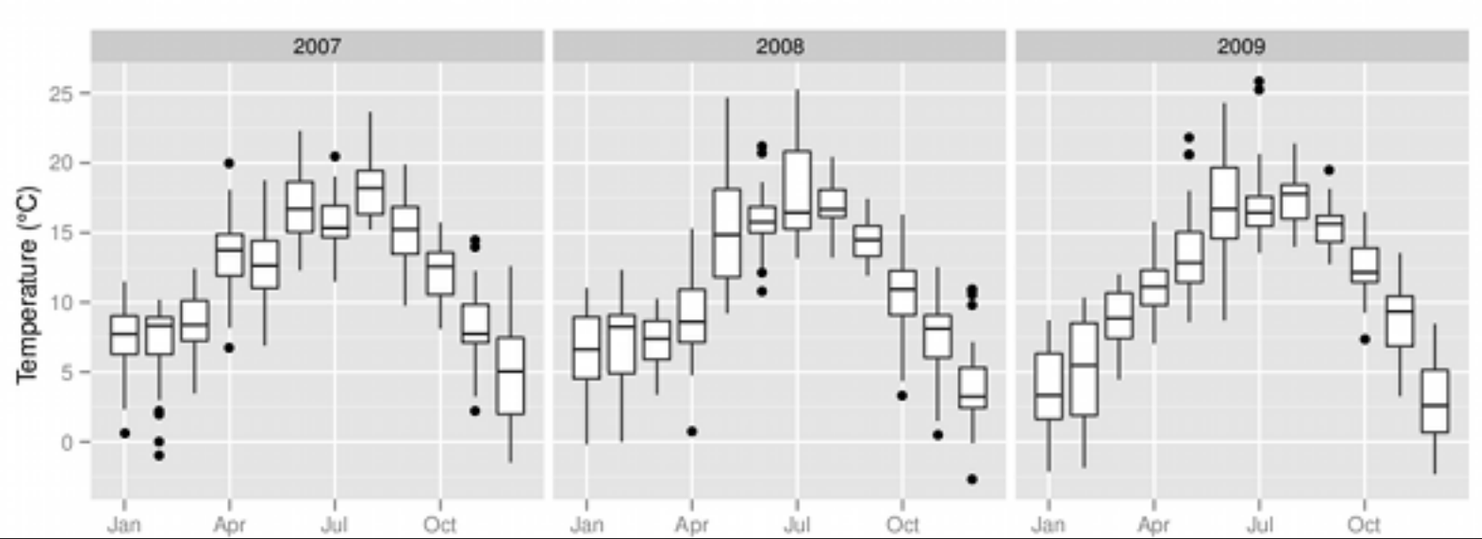
Community

- "R is the most discussed software by roughly a 2:1 margin, followed by Stata then SAS."
- Many online resources and discussions available
- R users are passionate and helpful
- Free + powerful + community = many, many, freely available useful packages

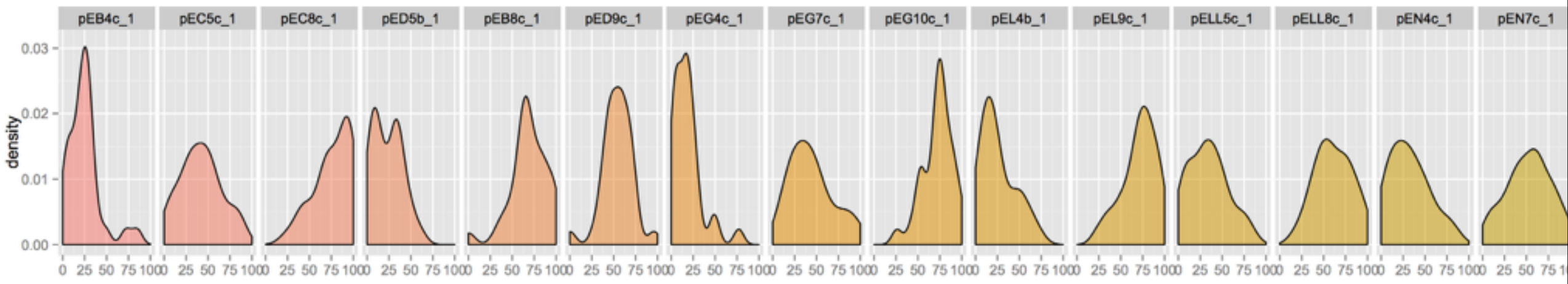
Reproducible pretty pictures







Also easily automated

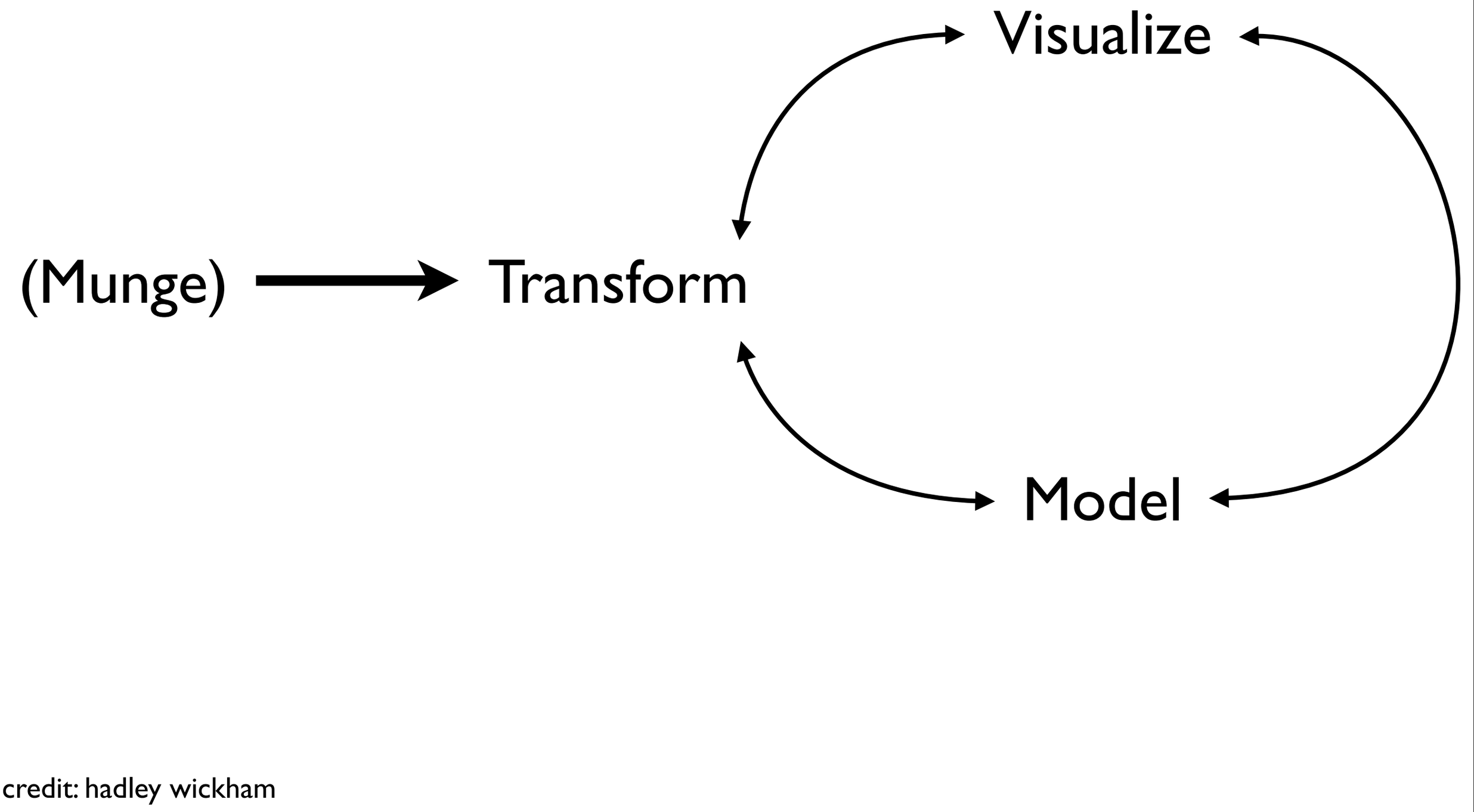


If that wasn't enough…

SKILL	2013	YR/YR CHANGE
R	$ 115,531	n/a
NoSQL	$ 114,796	1.6%
MapReduce	$ 114,396	n/a

data scientist using open-source tools earned a higher salary ($130,000) than those using proprietary tools ($90,000)

Too many choices



credit: hadley wickham

The following is a bit boring
but NECESSARY

Basic data types

- Classes of "objects"
  - character, numeric, integer, logical
- Vectors are series of objects in the same class
  - Lots of R functions *vectorize* -> apply to entire vector
- Lists CAN contain objects of different classes
- Objects have attributes such as length, dimensions, etc...

Basic expressions

- Assignment

```
> x <- 1
> x
[1] 1
```

- Functions

```
> print(1:10)
[1] 1 2 3 4 5
[6] 6 7 8 9 10
```

Making vectors

c() - *creates* vectors

```
> x <- c(0.5, 0.5)          ## numeric
> x <- c(TRUE, FALSE)       ## logical
> x <- c("a", "b", "c")     ## character
```

credit: roger peng

Coercing objects

- Coercion means to change types
- For example, if data is a character, function “mean()” won’t work properly

```
> x <- 0:6
> class(x)
[1] “integer”
> as.character(x)
[1] "0" "1" "2" "3" "4" "5" "6"
```

- Try: as.complex, as.logical

Matrices

Matrices are like vectors but with more than one dimension

```
> m <- matrix(nrow = 2, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
> dim(m)
[1] 2 3
```

Creating matrices from vectors

- cbind (columns) and rbind (rows)

```
> x <- 1:3
> y <- 10:12
> cbind(x, y)
     x  y
[1,] 1 10
[2,] 2 11
[3,] 3 12
> rbind(x, y)
  [,1] [,2] [,3]
x    1    2    3
y   10   11   12
```

Lists

Remember: can have different types

```
> list("a", 5)
[[1]]
[1] "a"

[[2]]
[1] 5
```

Factors - categorical

- Factors are like vectors but have categorical "labels". Represented by numbers under the hood in R

```
> factor(c("Male", "Female", "Other", "Male"))
[1] Male   Female Other  Male
Levels: Female Male Other
```

- ! Numerical order is by alphabetical

Data frames!

- Basically, a big table
- A type of list where each element of the list has same length
- Each COLUMN has to be the same type, but different types across columns
- Typically you create it by loading a csv table

subject	condition	rt	asleep
1	Future	440	FALSE
2	Past	300	TRUE
3	Future	120	FALSE
4	Past	80	FALSE

What are the types?

str()

```
> str(TDdata)

'data.frame': 7020 obs. of  8 variables:

 $ sub          : int  10 10 10 10 10 10 10 10 10 10 ...

 $ condition    : Factor w/ 3 levels "FUTURE","PAST",..: 3 2 3 1 2 2
3 1 3 1 ...

 $ delay        : int  10 5 21 5 5 5 21 120 120 42 ...

 $ later_value : int  27 11 13 18 13 14 11 14 22 18 ...

 $ choiceRT     : int  1832 298 456 0 291 1117 543 628 565 298 ...

 $ later_choice: int  1 0 0 0 0 0 0 0 0 0 ...

 $ primeRT      : int  888 1635 949 935 905 1055 4728 271 1713
2436 ...

 $ cue          : Factor w/ 90 levels "APE","ARTIST",..: 19 61 16 12
6 11 44 82 23 37 ...

 $ later_choice: int  1 1 1 1 1 1 1 1 1 1 ...
```

Loading from .csv

- `dataFrame <- read.csv("file.csv")`
- Often will load strings as factors and numbers as integers and numeric automatically (which is what you want most of the time)
- Always check data type if acting up

Accessing & manipulating data

Indexing

- Numeric indexing to access based on “location”
- `data[ROW, COLUMN]`
  - `data[2,]` - Second row
  - `data[, 4]` - Fifth column
  - `data[1, 2]` - Specific item
  - `data[1:5, ]` - ????

```
> trustData[1:5,]
        ResponseID condition face_number later_value delay
1 R_eqRGUi5hxVbn49L   Neutral        006a          11     4
2 R_eqRGUi5hxVbn49L   Neutral        006a          18    90
3 R_eqRGUi5hxVbn49L   Neutral        006a          22    14
4 R_eqRGUi5hxVbn49L   Neutral        006a          14     4
5 R_eqRGUi5hxVbn49L   Neutral        006a          14     7
```

Access by name

- data$column_name
- data$age

```
> trustData$delay

 [1]  4 90 14  4  7  7  7  7 90  4

> class(trustData$delay)

[1] "integer"
```

attach()

- attach(data) will make it so you can type variable names directly
- warning: workspace can easily get out of hand so this usage is NOT recommended.

Vectorized operations

- One of the "special" features of R
- Will try to vectorize operations wherever possible

```
> data$miles / data$hours

 [1] 0.3636364 5.0000000 0.6363636 0.2857143

 [5] 0.5000000 0.6363636 0.2333333 0.2058824

> data$mph <- data$miles / data$hours
> data$rt <- scale(data$rt)
```

transform

```
> data <- transform(data, mph = miles/hours)
> data <- transform(data, tooLong = choiceRT > 1000)
   sub condition choiceRT tooLong
1   10      SIZE     1832    TRUE
2   10      PAST      298   FALSE
3   10      SIZE      456   FALSE
4   10    FUTURE        0   FALSE
5   10      PAST      291   FALSE
6   10      PAST     1117    TRUE
7   10      SIZE      543   FALSE
```

Remember not to mutate objects

subset

- allows you to select a subset of a data frame based on a logical test

```
subset(data,condition, select=c("column_1")
```

subset

```
> subset(data, choiceRT<200
& choiceRT != 0 & condition == "SIZE")

     sub condition choiceRT
1771  30      SIZE      186
1866  31      SIZE      183
1868  31      SIZE      192
1869  31      SIZE      172
1875  31      SIZE       21
2029  33      SIZE      113
2034  33      SIZE      131
```