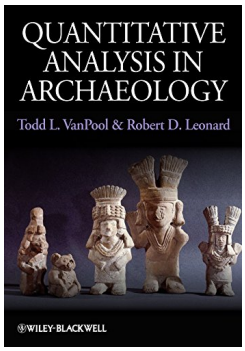
# Coding in R

## Reflection on the last week

### Tidy data principles

### Readings




## Objectives

Today's goals...

- Notion of functions vs. objects.
- Intro on R data types (ie. *what types of things are there*).
- Intro on R types of objects (ie. *how things are represented*).
- Subsetting data frames.
- Reading data into R.

## R is a smart calculator...

```r
2 + 40
```

```
[1] 42
```

```r
5^2
```

```
[1] 25
```

```r
round(6.48^2)
```

```
[1] 42
```

```r
5 < 10
```

```
[1] TRUE
```

```r
sqrt(1764)
```

```
[1] 42
```

```r
8 * 10^10
```

```
[1] 8e+10
```

```r
x <- 1
```

```r
x
```

```
[1] 1
```

```r
y <- 41
```

```r
x + y
```

```
[1] 42
```

```
x > y
```

```
[1] FALSE
```

```
z <- x - y
```

```
z
```

```
[1] -40
```

```
(x + y)^2
```

```
[1] 1764
```

## Functions and objects

### Objects

- *Anything* is an object.
- Objects contain **data** *(etc.)*
- Objects have **names**.
- You choose the names.
- Name your objects wisely.

```
x <- 1
x
```

```
[1] 1
```

```
pi
```

```
[1] 3.141593
```

```
pi + 1
```

```
[1] 4.141593
```

`<-` is an *assignment* operator
(use *Alt* + - in RStudio to write it).

### Functions

- End with parentheses `function(arguments)`.
- **Arguments** go in the parentheses.
- Functions **do** something with the inputs you give them.

```
sqrt(x = 1764)
```

```
[1] 42
```

```
args(sqrt)
```

```
function (x)
NULL
```

```
round(pi)
```

```
[1] 3
```

```
args(round)
```

```
function (x, digits = 0)
NULL
```

```
round(pi, digits = 2)
```

```
[1] 3.14
```

## Types of objects

### Vector

- Basic data structure.
- Contains single data type.
- Created using function `c()` *(combine, concatenate)*

```r
c("Fuu", "Bar")
```

```
[1] "Fuu" "Bar"
```

```r
x <- c(1, 3, 5, 8)
x
```

```
[1] 1 3 5 8
```

```r
x^2
```

```
[1]  1  9 25 64
```

```r
is.vector(x)
```

```
[1] TRUE
```

```r
x >= 4
```

```
[1] FALSE FALSE  TRUE  TRUE
```

```r
length(x)
```

```
[1] 4
```

## Types of objects

### Data frame & tibble

- A table.
- Has rows and columns.
- *Rectangular*, ie. identical number of rows in each column.

```r
dfr
```

```
   x y     z   w
1 95 a  TRUE 4.2
2 96 b FALSE 4.4
3 97 c FALSE 4.6
4 98 d  TRUE 4.8
```

```
is.data.frame(dfr)
```

```
[1] TRUE
```

```
ncol(dfr)
```

```
[1] 4
```

```
nrow(dfr)
```

```
[1] 4
```

```
head(dfr, n = 2)
```

```
   x y     z   w
1 95 a  TRUE 4.2
2 96 b FALSE 4.4
```

### Subsetting

- $ operator returns a single column.

```
colnames(dfr)
```

```
[1] "x" "y" "z" "w"
```

```
dfr$x
```

```
[1] 95 96 97 98
```

```
dfr$y
```

```
[1] "a" "b" "c" "d"
```

```
dfr$z
```

```
[1]  TRUE FALSE FALSE  TRUE
```

```
dfr$w / 2
```

```
[1] 2.1 2.2 2.3 2.4
```

```
dfr$x - dfr$w
```

```
[1] 90.8 91.6 92.4 93.2
```

```
str(dfr)
```

```
'data.frame':   4 obs. of  4 variables:
 $ x: int  95 96 97 98
 $ y: chr  "a" "b" "c" "d"
 $ z: logi  TRUE FALSE FALSE TRUE
 $ w: num  4.2 4.4 4.6 4.8
```

## Data types

```
str(dfr)
```

```
'data.frame':   4 obs. of  4 variables:
 $ x: int  95 96 97 98
 $ y: chr  "a" "b" "c" "d"
 $ z: logi  TRUE FALSE FALSE TRUE
 $ w: num  4.2 4.4 4.6 4.8
```

### Text strings

- **Character** data type, abbreviated as *chr*.
- Written in quotation marks (double or single).

```
"I am a string."
```

```
[1] "I am a string."
```

```
x <- 'I am also a string'
is.character(x)
```

```
[1] TRUE
```

- Functions with `is.` prefix:
  `is.numeric()`, `is.double()` etc.

### Numbers

- **Integers** (*whole* numbers)
- **Doubles** (decimal numbers)
- **Numeric** (class for all numbers in general)

```
dfr$x
```

```
[1] 95 96 97 98
```

```
is.numeric(dfr$x)
```

```
[1] TRUE
```

```
dfr$x + dfr$w
```

```
[1]  99.2 100.4 101.6 102.8
```

```
mean(dfr$x)
```

```
[1] 96.5
```

## Data types

### Dichotomies

- **Logical** data type.
- Binary/boolean values.
- As `TRUE` and `FALSE` in `R`.

### Special values

- Missing values as `NA`, ie. *not available*.
- `Inf` and `-Inf` for infinities.
- `NULL` for an object of a zero length.

## Reading data in CSV into R

### Comma separated values (CSV)

- **Plain-text** document.
- Practical **exchange** and **preservation** format for data sets.
- Most open and commercial softwares will allow export in CSV.
- Americas: separated by commas (`,`), period (`.`) as a decimal mark.
- Europe: separated by semicolon (`;`), comma (`,`) as a decimal mark.

### Reading CSV into R

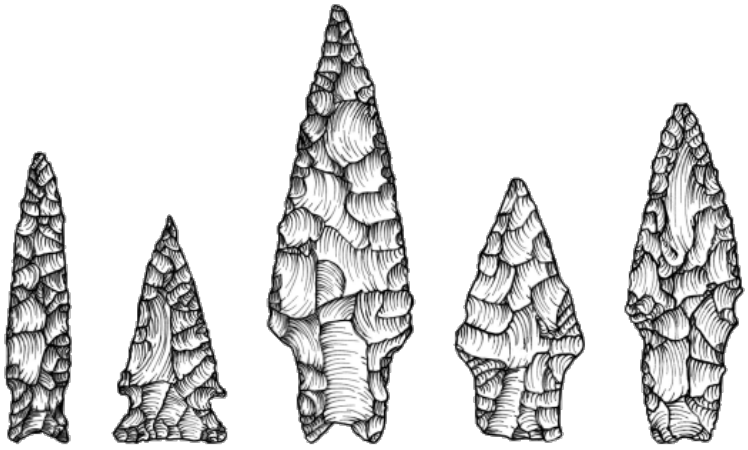
Comma separated:
```
read.csv(file = "path")
```

Semicolon separated:
```
read.csv2(file = "path")
```

Other delimiter:
```
read.table(file = "path", sep = "separator")
```

## Practice

### Dart points



*Adapted from Carlson 2011*

## Dart points

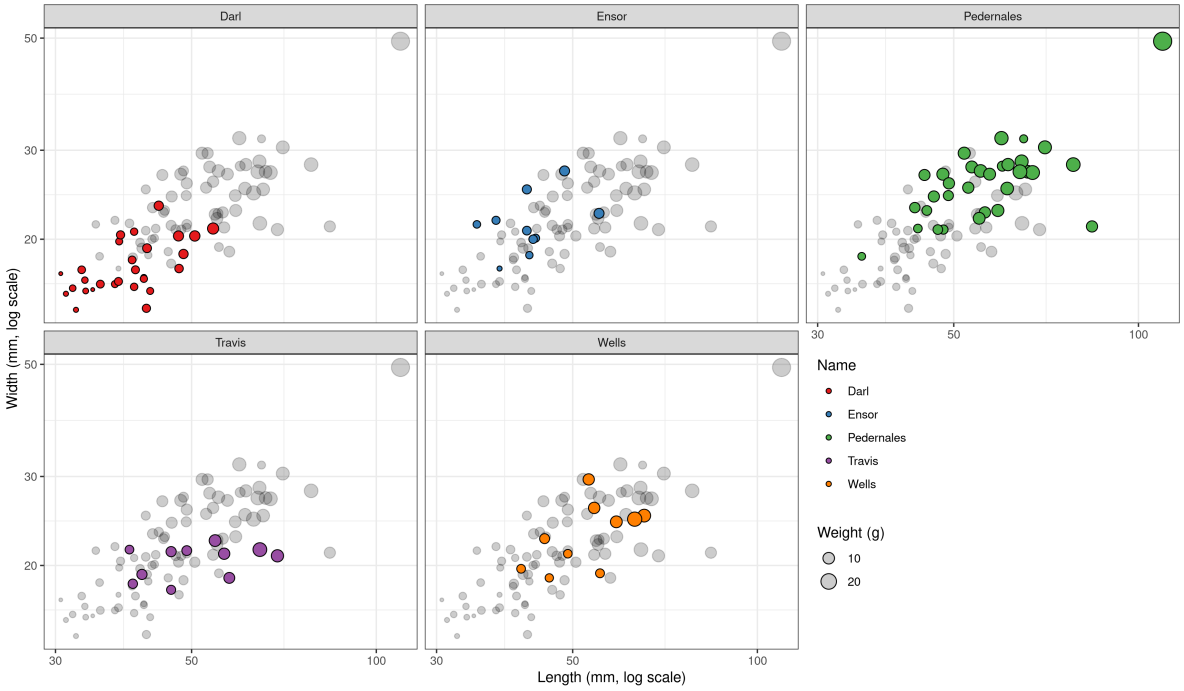



Figure 1: Scatter plot of dart points

Measurements on five types of dart points from **Fort Hood** in central Texas (**Darl, Ensor, Pedernales, Travis, and Wells**). The points were recovered during 10 different pedestrian survey projects during the 1980's and were classified and measured by H. Blaine Ensor…

## Exercise

1. Download the dataset dartpoints.csv
2. Explore the dataset using spreadsheet editor
3. Save the dataset somwhere you can find it
4. Start **RStudio**
5. Create a script (*Ctrl + Shift + n*)
6. Read the dataset from the path where you saved it

7. Save it as `darts` object
8. What kind of an object is it?
9. How many rows and columns does it have?

10. What columns does it have?
11. What data types are there? What is the structure?
12. Read details about the dataset using `?archdata::DartPoints`
    Hints: `read.csv()`, `ncol()` and `nrow()`, `str()`, `colnames()`

### Solution

```
url <- "https://petrpajdla.github.io/stat4arch/lect/w02/data/dartpoints.csv"
download.file(url, "./dartpoints.csv")
darts <- read.csv("./dartpoints.csv")
class(darts)
dim(darts)
nrow(darts)
ncol(darts)
colnames(darts)
head(darts)
str(darts)
```