**UNIT TESTING**

**Reproducibility**

-Unit testing is about testing functions

-Unit testing is just a test small bits of code

-Writing test to test the code

- unit test specifications

Testing a function which check if the year is a leap year

Expect\_equal -

Expect\_error – we expect to get an error

Task - 5 minutesTake the following test code and write down an explanation of all the different parts:  
- What is the test name? -  
 What is the expectation?  
- What are you expecting this test to do?Think carefully about the logic of this one before you write down your assumptions… you may want to try and run it to see what happens first.

# second test : one expectation  
test\_that("Character input returns an error", {  
 expect\_error(is\_leap\_year("year"))  
})

#test passes

```{r}

is\_leap\_year <- function(year){

if (year %% 100 == 0){ #check

if (year %% 400 == 0){

return(TRUE)

} else{

return(FALSE)

}

}

if (year %% 4 == 0){

return(TRUE)

} else {

return(FALSE)

}

}

```

```{r}

is\_leap\_year(1992)

undebug(is\_leap\_year)

```

```{r}

library(testthat)

```

```{r}

a <- 10

```

```{r}

expect\_equal(a, 11) #checks if the a is equal to 11 . If error displays which means the test has passed

```

```{r}

test\_that("Real leap years to yield TRUE",{

expect\_equal(is\_leap\_year(2020),TRUE)

expect\_equal(is\_leap\_year(2000),TRUE) # we expect two arguments to be

})

```

```{r}

# second test : one expectation

test\_that("Character input returns an error", {

expect\_error(is\_leap\_year("year"))

})

```

```{r}

test\_that("Non- numeric input returns an error",{

expect\_error(is\_leap\_year("banana"))

expect\_error(is\_leap\_year(TRUE))

})

```

```{r}

is\_leap\_year <- function(year){

if(!is.numeric(year)){

stop("Non - numeric input")

}

if (year %% 100 == 0){ #check

if (year %% 400 == 0){

return(TRUE)

} else{

return(FALSE)

}

}

if (year %% 4 == 0){

return(TRUE)

} else {

return(FALSE)

}

}

```

```{r}

#Task - 5 minutes

#Write a test that checks whether 2017 and 1900 are correctly identified as not being leap years by our is\_leap\_year() function

test\_that("2017 and 1900 returns an error",{

expect\_equal(is\_leap\_year(2017), FALSE)

expect\_equal(is\_leap\_year(1900), FALSE)

})

```

```{r}

```

WHEN SHOULD YOU USE TESTING

* When you are using lot of functions

**TEST DRIVEN DEVELOPMENT**

Write a code -> Write a code to test ->

**ASSERTIVE PROGRAMMING**

```{r}

library(tidyverse)

```

```{r}

weather\_one <- read\_csv("data/delhi\_weather\_one.csv")

```

```{r}

weather\_report <- function(weather){

# Our average temperature, humidity and wind speed

weather\_means <-

weather %>%

summarise(

mean\_temp = mean(temp),

mean\_humidity = mean(humidity),

mean\_windspeed = mean(wind\_speed)

)

# Show hot and humid days

hot\_and\_humid <-

weather %>%

filter(temp > 20) %>%

filter(humidity > 55)

return(list(weather\_means, hot\_and\_humid))

}

```

```{r}

weather\_report(weather\_one)

```

```{r}

weather\_two <- read\_csv("data/delhi\_weather\_two.csv")

```

```{r}

weather\_report(weather\_two)

```

```{r}

library(assertr)

```

#highest ever temperature in India was 51C and the lowest ever was -34C

#highest ever wind speed in the world is 108mph

weather\_report <- function(weather){

weather %>%

verify(temp <= 52 & temp >= -35) %>%

verify(wind\_speed >= 1 & wind\_speed <= 108) %>%

verify(humidity >= 0 & humidity <= 100)

# Our average temperature, humidity and wind speed

weather\_means <-

weather %>%

summarise(

mean\_temp = mean(temp),

mean\_humidity = mean(humidity),

mean\_windspeed = mean(wind\_speed)

)

# Show hot and humid days

hot\_and\_humid <-

weather %>%

filter(temp > 20) %>%

filter(humidity > 55)

return(list(weather\_means, hot\_and\_humid))

}

```

```{r}

weather\_report(weather\_two)

```