HOW TO CODE MISSING DATA

- **NA** = "Not Available"
- Makes certain calculations impossible

Pseudocode:

About NAs in Data

- Create a dataset with NA. Summary() shows separate column with NAs; however, functions like mean() do not work with NAs.

How to get rid of Missing values

- Find missing values with **which()** function: `which(method(dataset))`
    - `which(is.na(x1)` gives index number of NAs
- Ignore missing values with `na.rm = T:` use with functions, like mean()
- Replace missing values with 0:(or other number)
    - using `is.na` or `ifelse`

Imputation

- Guess what # should go in NA. Easiest is to put mean of that variable there.
- **imputation method** : replace value sames as above (using `is.na` & `ifelse`)
    - except, instead of a 0 value, use a **function** to replace the missing value

SCRIPTS SUMMARY

DATA

```
x1 <- c(1, 2, 3, NA, 5)
summary(x1)
mean(x1)
```

MISSING VALUES

```
which(is.na(x1))
mean(x1, na.rm = T)
x2 <- x1
x2[is.na(x2)] <- 0
x2
x3 <- ifelse(is.na(x1), 0, x1)
x3
```

IMPUTATION

```
browseURL("http://cran.r-project.org/web/packages/mi/index.html")
browseURL("http://cran.r-project.org/web/packages/mice/index.html")
browseURL("http://cran.r-project.org/web/packages/imputation/index.html")
```

CLEAN UP

```
rm(list = ls())
```

SCRIPTS & NOTES

DATA

Create dataset with NAs:

```
x1 <- c(1, 2, 3, NA, 5)
```

- → workspace: values x1 numeric[5] (even with NA)

```
summary(x1)  # works with NA (shows # of NAs)
#       Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#       1.00    1.75    2.50    2.75    3.50    5.00    1
mean(x1)        # does not work - default assumes all are numeric
#[1] NA
```

- Error - bc default version of mean assumes that these are all valid values

FIX MISSING VALUES

Find missing values:

```
which(is.na(x1))    # gives index number of NAs
```

- **which** function (method (dataset))
    - give row to look for NA → returns index value of NA
        - for variable x1
        - → find values that are NA
        - ⇒ return which index # that is

```
# [1] 4
```

the 4th value in the set

Ignore missing values with na.rm = T:

```
mean(x1, na.rm = T)
```

- when have missing value (NA)
- → tell function mean that have NAs in dataset - to remove them
    - na not available
    - rm remove
    - T true (can write word TRUE)

```
#[1] 2.75
```

same as in summary data above

Replace missing values with 0:(or other number)

option 1: using "is.na"

- IF something is NA THEN zero goes into it ⇒ 1, 2, 3, 0, 5

```
x2 <- x1                    # put x1 into x2
x2[is.na(x2)] <- 0          # in set x2, put 0 where is not a number
x2
# [1] 1 2 3 0 5
```

- often put mean value of dataset in NA place

option 2: using "`ifelse`"

- IF something is NA THEN put in `0` ELSE put in value of dataset `x1`

x3 <- ifelse(is.na(x1), 0, x1)

- goes to variable x1 (x1)
- IF there is an NA (is.na)
    - THEN put in 0
- IF the number is not an NA
    - THEN take its number form x1
- ⇒ feed it all into x3

x3

#[1] 1 2 3 0 5

IMPUTATION

- **imputation**: replace missing data NA with another number
    - imputation - guess what # should go in there
        - easiest - put mean of that variable there
- **imputation method** : replace value sames as above (using is.na & ifelse)
    - except, instead of a 0 value, use a **function** to replace the missing value

For data frames, R has many packages to deal intelligently with missing data via imputation. These are just three:

- **`mi`**: Missing Data Imputation and Model Checking
    - **browseURL("http://cran.r-project.org/web/packages/mi/index.html")**
    - CRAN - Package mi
    - Sophisticated procedures: exps
        - mean imputation
        - regression imputation
        - multiple imputation which maintains the probability distributions of variables
- **`mice`**: Multivariate Imputation by Chained Equations
    - **browseURL("http://cran.r-project.org/web/packages/mice/index.html")**
    - CRAN - Package mice
- **`imputation`**
    - **browseURL("http://cran.r-project.org/web/packages/imputation/index.html")**
    - Archived on 2014-01-14 for policy violation (using all the processors on a large system).

CLEAN UP

rm(list = ls())