Introduction to R

Learning objectives

- Introduction to calculations, vectors, data frames, if/else, functions, etc
- Tutorial partially based on University of Michigan Genome Analysis and software carpentry courses
- www.r-project.org for download and help

R overview

- Free and has a large community of users
- R implements many common statistical procedures
- R provides excellent graphics functionality
- Useful for many data analysis and statistics projects
- R defaults to an interactive mode
- A prompt ">" is presented to users

Commenting

- Use # signs to comment.
- Anything to the right of a # is ignored by R.
- Helpful to comment code.

Assignment operator

- <- or = is the assignment operator
- Assigns values on the right to objects on the left
- Like an arrow that points from the value to the object
- Some people prefer <- and others prefer =
- For input in a function, you need = (i.e. rnorm(n=100))

```r
# Create the variable A, which equals 4
A<-4

A # Display the value of A
```

```
## [1] 4
```

```r
# Create the variable B, which equals 3
B=3

B # Display the value of B
```

```
## [1] 3
```

R as a Calculator

```
# Simple arithmetic
4+3
```

```
## [1] 7
```

```
# Multiplication, division, subtraction
4*((4-3)/3)
```

```
## [1] 1.333333
```

```
# Exponentiation
3 ^ 2
```

```
## [1] 9
```

```
# Square root
sqrt(9)
```

```
## [1] 3
```

```
# Basic mathematical functions are available
exp(1)
```

```
## [1] 2.718282
```

```
# The constant pi is predefined
pi
```

```
## [1] 3.141593
```

```
# Recall A=4, B=3
A+B
```

```
## [1] 7
```

```
# Check if A equals B
A==B
```

```
## [1] FALSE
```

```
# Check if A does not equal B
A!=B
```

```
## [1] TRUE
```

```
# Check if A is greater than B
A>B
```

```
## [1] TRUE
```

```
# Check is A is less than B
A<B
```

```
## [1] FALSE
```

```
# R is case sensitive
a
```

```
## Error in eval(expr, envir, enclos): object 'a' not found
```

R Vectors

- A series of numbers created with
  - c() to concatenate elements or sub-vectors
  - rep() to repeat elements or patterns
  - seq() or m:n to generate sequences
- Most mathematical functions and operators can be applied to vectors

```r
# Repeats the number 1,10 times
rep(1,10)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

```r
# Sequence of integers between 2 and 6
c(2,3,4,5,6)
```

```
## [1] 2 3 4 5 6
```

```r
# Equivalent to sequence above
c(2:6)
```

```
## [1] 2 3 4 5 6
```

```r
# Sequence of integers between 2 and 6 by 1
seq(from=2,to=6,by=1)
```

```
## [1] 2 3 4 5 6
```

```r
# Create 2 vectors x and y
x <- c(2,0,0,4)
y <- c(1,1,2,3)

# Check the length of x
length(x)
```

```
## [1] 4
```

```r
# Sum the elements of two vectors
x + y
```

```
## [1] 3 1 2 7
```

```r
# Multiply each element of x by 4
x * 4
```

```
## [1]  8  0  0 16
```

```r
# Multiply x and y
x*y
```

```
## [1]  2  0  0 12
```

```r
# Sum of x times y
sum(x*y)
```

```
## [1] 14
```

```r
# Square root of each element of x and round to 2 digits
round(sqrt(x),2)
```

```
## [1] 1.41 0.00 0.00 2.00
```

Accessing Vector Elements

- Use the [] operator to select elements
- To select specific elements:
  - Use index or vector of indexes to identify them
- To exclude specific elements:
  - Negate index or vector of indexes

```
x
```

```
## [1] 2 0 0 4
```

```
# Select the first element, equivalent to x[c(1)]
x[1]
```

```
## [1] 2
```

```
# Exclude the first element
x[-1]
```

```
## [1] 0 0 4
```

```
# Set the first element to 3 and display
x[1] <- 3 ; x
```

```
## [1] 3 0 0 4
```

```
# Set values of x less than 4 to 1
 x[x<4] = 1;  x
```

```
## [1] 1 1 1 4
```

Matrices

```
# Create a matrix from x and y
matXY<-cbind(x,y)
matXY
```

```
##      x y
## [1,] 1 1
## [2,] 1 1
## [3,] 1 2
## [4,] 4 3
```

```
# Dimension of the matrix
dim(matXY)
```

```
## [1] 4 2
```

```
# Name the columns of matXY
colnames(matXY)<-c("X","Y") #CAREFUL WITH QUOTES
matXY
```

```
##      X Y
## [1,] 1 1
## [2,] 1 1
## [3,] 1 2
## [4,] 4 3
```

```r
# Create a matrix using matrix
matT<-matrix(c(1:16),nrow=4,ncol=4,byrow=T)
matT
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
## [4,]   13   14   15   16
```

```r
# Dimension of matrix T
dim(matT)
```

```
## [1] 4 4
```

```r
# Element in row 1, column 2
matT[1,2]
```

```
## [1] 2
```

```r
# Row 1
matT[1,]
```

```
## [1] 1 2 3 4
```

```r
# Column 2
matT[,2]
```

```
## [1]  2  6 10 14
```

```r
matT %*% matXY # matrix multiplication (4x4)*(4x2)=(4x2)
```

```
##        X   Y
## [1,]  22  21
## [2,]  50  49
## [3,]  78  77
## [4,] 106 105
```

```r
dim(matT %*% matXY) #check dimensions
```

```
## [1] 4 2
```

```r
matXY %*% matT # (4x2) * (4x4)
```

```
## Error in matXY %*% matT: non-conformable arguments
```

Data Frames

- Group a collection of related vectors
- Most of the time, when data is loaded, it will be organized in a data frame

Loading Data Sets

- Load from a text file using read.table()
- Example: bp <-read.table("bp.txt",header=T)
- For help: ?read.table
- Parameters header, sep, and na.strings control useful options
- read.csv() and read.delim() have useful defaults for comma or tab delimited files

- Create from scratch using data.frame()
  - Example:

```
bp<-data.frame(height=c(150,160),weight=c(65,72))
# Careful w/ quotes in R & word formatting
colnames(bp)<-c("HEIGHT","WEIGHT")
bp
```

```
##   HEIGHT WEIGHT
## 1    150     65
## 2    160     72
```

Accessing Data Frames

- Multiple ways to retrieve columns
- The following all retrieve weight data:

```
bp["WEIGHT"]
```

```
##   WEIGHT
## 1     65
## 2     72
```

```
bp[,2]
```

```
## [1] 65 72
```

```
bp$WEIGHT
```

```
## [1] 65 72
```

Lists

- Collections of related variables
- Created with list function

```
point <- list(x = 1, y = 1)
point
```

```
## $x
## [1] 1
##
## $y
## [1] 1
```

- Access to components follows similar rules as for data frames, the following all retrieve x:

```
point$x; point["x"]; point[1]
```

```
## [1] 1
```

```
## $x
## [1] 1
```

```
## $x
## [1] 1
```

Programming Constructs

- if . . . else . . .
- for loops
- repeat loops
- while loops

Example: if . . . else . . .

```
vec<-rep(c(0,1),times=7)
vec
```

```
##  [1] 0 1 0 1 0 1 0 1 0 1 0 1 0 1
```

```
#Set i to 1
i<-1
if (vec[i] == 1){ print(paste("Yeah, the number", vec[i]))
}else{print(paste("Yippy, the number", vec[i]))
}
```

```
## [1] "Yippy, the number 0"
```

for

- Loop through variable in for statement

Example: for

```
# Cycle through the whole vec vector; paste in the whole section below
for(i in 1:length(vec)){
if (vec[i] == 1){
print(paste("Yeah, the number", vec[i]))
}else{
print(paste("Yippy, the number", vec[i]))
}# close the if else statement
} #close the for loop
```

```
## [1] "Yippy, the number 0"
## [1] "Yeah, the number 1"
## [1] "Yippy, the number 0"
## [1] "Yeah, the number 1"
## [1] "Yippy, the number 0"
## [1] "Yeah, the number 1"
## [1] "Yippy, the number 0"
## [1] "Yeah, the number 1"
## [1] "Yippy, the number 0"
## [1] "Yeah, the number 1"
## [1] "Yippy, the number 0"
## [1] "Yeah, the number 1"
## [1] "Yippy, the number 0"
## [1] "Yeah, the number 1"
```

repeat

- Continually evaluate expression
- Loop must be terminated with break statement

Example: repeat

```r
# Sample with replacement from a set of N=10 objects until the number 7 is sampled twice
N<-10
M <- matches <- 0
repeat{
# Keep track of total connections sampled, increase by 1 each time
M <- M + 1
# Sample a new number from the set of 1 to 10
p = sample(N, 1)
# Increment matches whenever we sample 7
if (p == 7){matches <- matches + 1}
# Stop after 2 matches
if (matches == 2){break}
}
print(paste("Loop ran",M,"times in order to find",matches,"matches for the number 7"))
```

```
## [1] "Loop ran 34 times in order to find 2 matches for the number 7"
```

while

- While expression 1 is false, repeatedly evaluate expression 2

Example: while

```r
# Sample with replacement from a set of N=10 objects until 7 and 8 are sampled consecutively
N<-10
match <- FALSE
while (match == FALSE)
{
# Sample a new element
p = sample(N, 1)
# If not 7, then go to next iteration
if (p != 7)
   next;
# Sample another element
q = sample(N, 1)
# If not 8, then go to next iteration
if (q != 8)
next;
   match = TRUE;
}
c(p,q)
```

```
## [1] 7 8
```

Functions in R

- As tasks become complex, it is a good idea to organize code into functions that perform defined tasks
- In R, it is good practice to give default values to function arguments

Function definitions

Of the form: name <- function(argument1, argument2, . . .){ expression } Arguments can be assigned default values Return value is the last evaluated expression or can be set explicitly with return()

Defining Functions

```
square <- function(x = 10){
 x * x
}

# Default value is 10
square()
```

```
## [1] 100
```

```
# Run with 2
square(2)
```

```
## [1] 4
```

Basic Utility Functions

- length() returns the number of elements
- mean() returns the sample mean
- median() returns the sample mean
- range() returns the largest and smallest values
- unique() removes duplicate elements
- summary() calculates descriptive statistics
- Basic power functions. You need to have the package pwr installed first. Enter install.packages("pwr") in the command prompt and pick a mirror. Don't need to install a package everytime. Use library to laod the package.

```
#install the package using install.packages("pwr")
#load the library
library(pwr)

# power for test of correlation
pwr.r.test(n = 100, r = 0.5, alternative = c("two.sided"))
```

```
##
##      approximate correlation power calculation (arctangh transformation)
##
##               n = 100
##               r = 0.5
##       sig.level = 0.05
##           power = 0.9997432
##     alternative = two.sided
```

R Help System

- R has a built-in help system with useful information and examples
- help() provides general help
- help(plot) will explain the plot function or ?plot
- help.search("histogram") will search for topics that include the word histogram
- example(plot) will provide examples for the plot function

Managing Workspaces

- As you generate functions and variables, these are added to your current workspace
- Use ls() to list workspace contents and rm() to delete variables or functions
- When you quit, with the q() function, you can save the current workspace for later use

Libraries for data example

```
# Use the commented command below if you do not have the package installed
# install.packages("boot")
# install.packages("table1")
# install.packages("corrplot")

# Don't use warn=FALSE in general, used here to save space
library(boot, warn=FALSE)

library(table1, warn=FALSE)

library(corrplot, warn=FALSE)
```

```
## corrplot 0.92 loaded
```

Dataset

- Load datasets using read.table, read.csv, fread (big datasets), etc
- Make sure you are in the right directory. Either use the path to load the data or pick the working directory where the data is stored (Session-Set Working directory -> or Choose directory or Misc-> change working directory)
- Example:

```
# d1<-read.table("/Users/path/datasetname.txt", sep="\t",header=T, na.strings="")
```

- Can store and load data as rdata but may be cumbersome, but allows for more complex data formats

Melanoma dataset

- ?melanoma
- Andersen, P.K., Borgan, O., Gill, R.D. and Keiding, N. (1993) Statistical Models Based on Counting Processes. Springer-Verlag.

The data consist of measurements made on patients with malignant melanoma.

- Each patient had their tumour removed by surgery at the Department of Plastic Surgery, University Hospital of Odense, Denmark during the period 1962 to 1977.
- Surgery consisted of complete removal of the tumour together with about 2.5cm of surrounding skin.

- Among the measurements taken were the thickness of the tumour and whether it was ulcerated or not. These are thought to be important prognostic variables in that patients with a thick and/or ulcerated tumour have an increased chance of death from melanoma.
- Patients were followed until the end of 1977.

This data frame contains the following columns:

- time: Survival time in days since the operation, possibly censored.
- status: The patients status at the end of the study. 1 indicates that they had died from melanoma, 2 indicates that they were still alive and 3 indicates that they had died from causes unrelated to their melanoma.
- sex: The patients sex; 1=male, 0=female.
- age: Age in years at the time of the operation.
- year: Year of operation.
- thickness: Tumour thickness in mm.
- ulcer: Indicator of ulceration; 1=present, 0=absent.

What information would we need?

- Number/percent of subjects with each status
- Mean/var of age and thickness by status or median/range
- Maybe test is mean age differs by status (t-test/ANOVA)
- Number/percent for sex and ulcer by status
- Maybe test is status differs by sex (chi-square test)
- Consider if age is associated with sex (correlation)

First, examine the dataset.

```
# create a melanoma2 dataset based on melanoma (alternative: melanoma2 = melanoma)
melanoma2 <- melanoma

# dimensions of melanoma2 matrix
dim(melanoma2)
```

```
## [1] 205   7
```

```
# Check if there is missing data
# melanoma2<-na.omit(melanoma2) #creates data set without missing values
dim(na.omit(melanoma2))
```

```
## [1] 205   7
```

```
# view first 3 rows
melanoma2[1:3,]
```

```
##   time status sex age year thickness ulcer
## 1   10      3   1  76 1972      6.76     1
## 2   30      3   1  56 1968      0.65     0
## 3   35      2   1  41 1977      1.34     0
```

```
# view first 3 rows and first 2 columns
melanoma2[1:3,1:2]
```

```
##   time status
## 1   10      3
## 2   30      3
## 3   35      2
```

```r
# column names of variables
colnames(melanoma2)
```

```
## [1] "time"      "status"    "sex"       "age"       "year"      "thickness"
## [7] "ulcer"
```

```r
# Use $ for a given variable or melanoma2[,"age"]
melanoma2$age
```

```
##   [1] 76 56 41 71 52 28 77 60 49 68 53 64 68 63 14 72 46 72 95 54 89 25 37 43 68
##  [26] 67 86 56 16 42 65 52 58 60 68 75 19 66 56 46 58 74 65 64 27 73 56 63 69 77
##  [51] 80 76 65 61 26 57 45 31 36 46 43 68 57 57 55 58 20 67 44 59 32 83 55 15 58
##  [76] 47 54 55 38 41 56 48 44 70 40 53 65 54 71 49 55 69 83 60 40 77 35 46 34 69
## [101] 60 84 66 56 75 36 52 58 39 68 71 52 55 66 35 44 72 58 54 33 45 62 72 51 77
## [126] 43 65 63 60 50 40 67 69 74 49 47 42 54 72 45 67 48 34 44 31 42 24 58 78 62
## [151] 70 35 61 54 29 64 47 62 32 49 25 49 64 36 58 37 54 61 31 61 60 43 68  4 60
## [176] 50 20 54 29 56 60 46 42 34 56 12 21 46 49 35 42 47 69 52 52 30 22 55 26 19
## [201] 29 40 42 50 41
```

- Determine the number of subjects

```r
# How many subjects?
nrow(melanoma2)
```

```
## [1] 205
```

```r
# How many subjects with status 1: died from melanoma? Percent?
nrow(melanoma2[melanoma2$status==1,])
```

```
## [1] 57
```

```r
nrow(melanoma2[melanoma2$status==1,])/nrow(melanoma2)
```

```
## [1] 0.2780488
```

```r
# How many subjects with status 2: alive? Percent?
length(melanoma2$status[melanoma2$status==2])
```

```
## [1] 134
```

```r
length(melanoma2$status[melanoma2$status==2])/length(melanoma2$status)
```

```
## [1] 0.6536585
```

```r
# How many subjects with status 3 and did not have missing data?
nrow(melanoma2[melanoma2$status==3 & !is.na(melanoma2$status),])
```

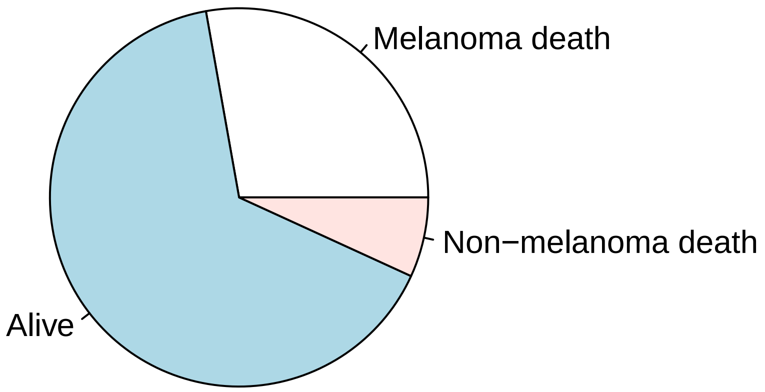
```
## [1] 14
```

```
#View status as a piechart
x<-c(nrow(melanoma2[melanoma2$status==1,]),nrow(melanoma2[melanoma2$status==2,]),
        nrow(melanoma2[melanoma2$status==3,]))
x
```

```
## [1]  57 134  14
```

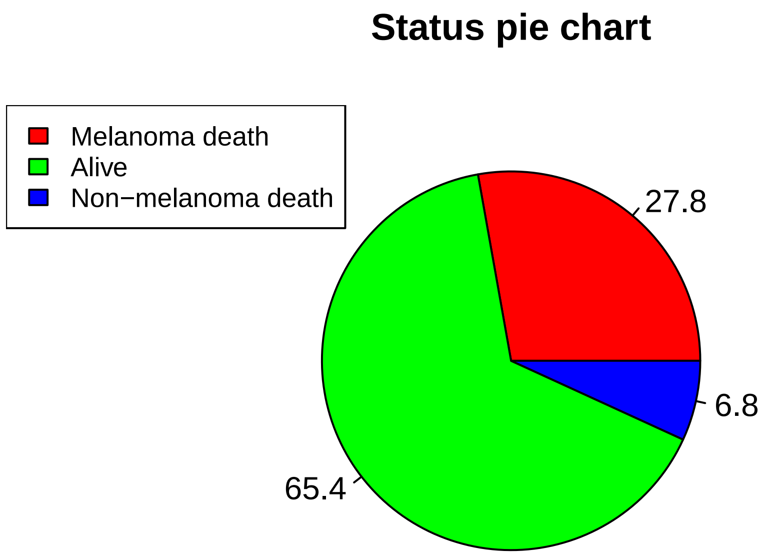
```
labels=c("Melanoma death","Alive","Non-melanoma death")

#regular pie chart
pie(x,labels)
```



```
# pie chart with percents
piepercent<- round(100*x/sum(x), 1)

pie(x, labels = piepercent, main = "Status pie chart",col = rainbow(length(x)))
legend("topleft", c("Melanoma death","Alive","Non-melanoma death"), cex = 0.8,
   fill = rainbow(length(x)))
```



Mean, standard deviation, variance, median

- Consider age. What is the average age?

```
# mean for age overall or sum(melanoma2$age)/length(melanoma2$age)
mean(melanoma2$age)
```

```
## [1] 52.46341
```

```
#mean age if there where missing values in dataset
mean(melanoma2$age,na.rm=T)
```

```
## [1] 52.46341
```

```
# round mean age to 1 decimal place
round(mean(melanoma2$age),1)
```

```
## [1] 52.5
```

```
# mean age for status 1: died from melanoma
mean(melanoma2$age[melanoma2$status==1])
```

```
## [1] 55.08772
```

```
# mean age for status 2: alive
mean(melanoma2$age[melanoma2$status==2])
```

```
## [1] 50.00746
```

```
# mean age for status 3: died not from melanoma
mean(melanoma2$age[melanoma2$status==3])
```

```
## [1] 65.28571
```

- What is the variance and standard deviation?

```
# sd for age overall
sd(melanoma2$age)
```

```
## [1] 16.67171
```

```
# sd for age if there where missing values in dataset
sd(melanoma2$age,na.rm=T)
```

```
## [1] 16.67171
```

```
# variance for age overall
sd(melanoma2$age)^2
```

```
## [1] 277.946
```

```
var(melanoma2$age)
```

```
## [1] 277.946
```

```
# variance for age by status
round(c(var(melanoma2$age[melanoma2$status==1]),var(melanoma2$age[melanoma2$status==2]),
        var(melanoma2$age[melanoma2$status==3])),1)
```

```
## [1] 320.7 253.3 118.8
```

- What is the median and range for age?

```
# median for age overall
median(melanoma2$age)
```

```
## [1] 54
```

```
# minimum age
min(melanoma2$age)
```

```
## [1] 4
```

```
#maximum age
max(melanoma2$age)
```

```
## [1] 95
```

```
#range
range(melanoma2$age)
```

```
## [1]  4 95
```

```
# create new variables
AgeStatus1<-melanoma2$age[melanoma2$status==1]
AgeStatus2<-melanoma2$age[melanoma2$status==2]
AgeStatus3<-melanoma2$age[melanoma2$status==3]

#median age by status
c(median(AgeStatus1),median(AgeStatus2),median(AgeStatus3))
```

```
## [1] 56 52 65
```

```
#matrix of ranges
matRange<-matrix(c(range(AgeStatus1),range(AgeStatus2),range(AgeStatus3)),nrow=2,ncol=3)
colnames(matRange)<-c("Status1","Status2","Status3")
rownames(matRange)<-c("min","max")
matRange
```

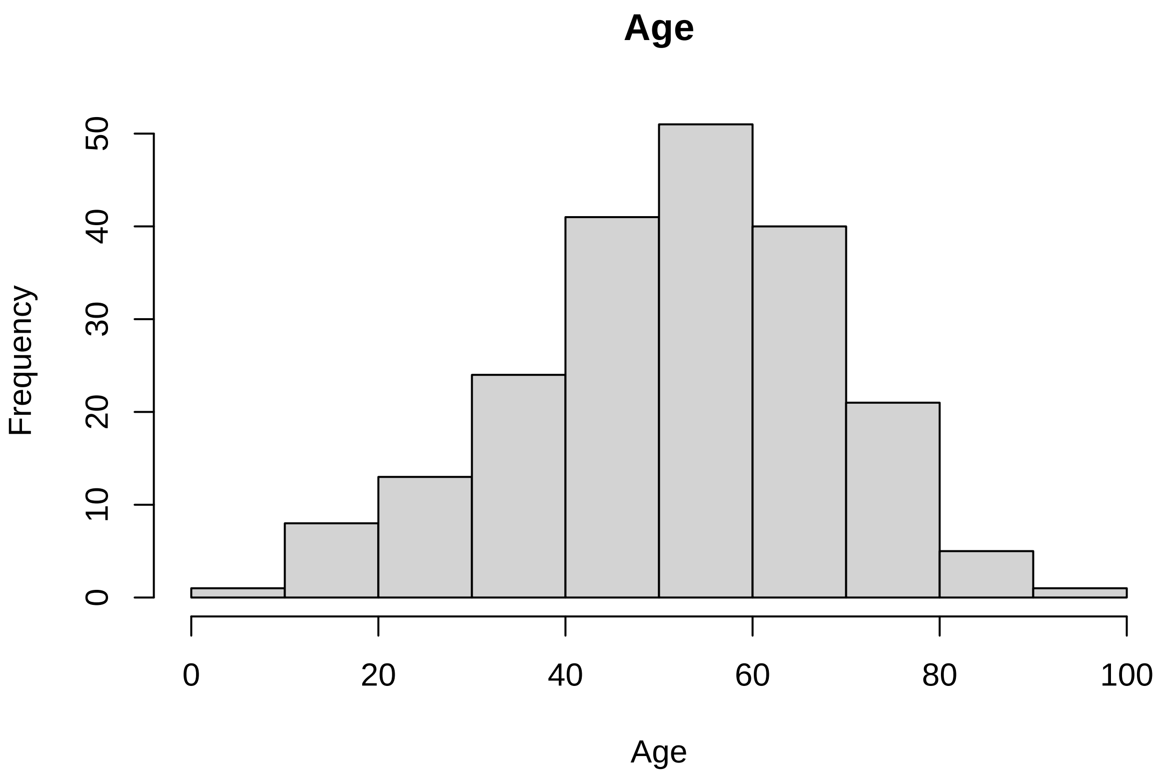
```
##     Status1 Status2 Status3
## min      14       4      49
## max      95      84      86
```

```
range(AgeStatus1)
```

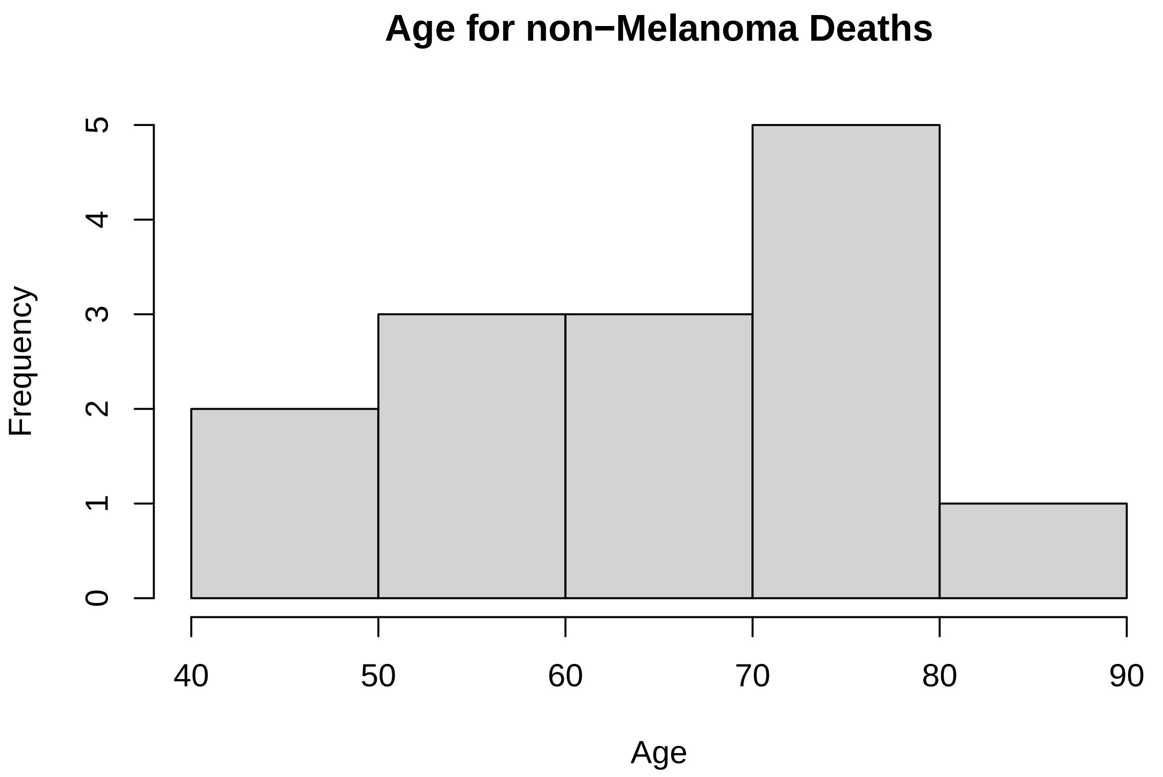
```
## [1] 14 95
```

- Should we be using medians or means? Check normality (Kolmogorov–Smirnov or Shapiro–Wilk test)

```r
hist(melanoma2$age,main="Age",xlab="Age")
```

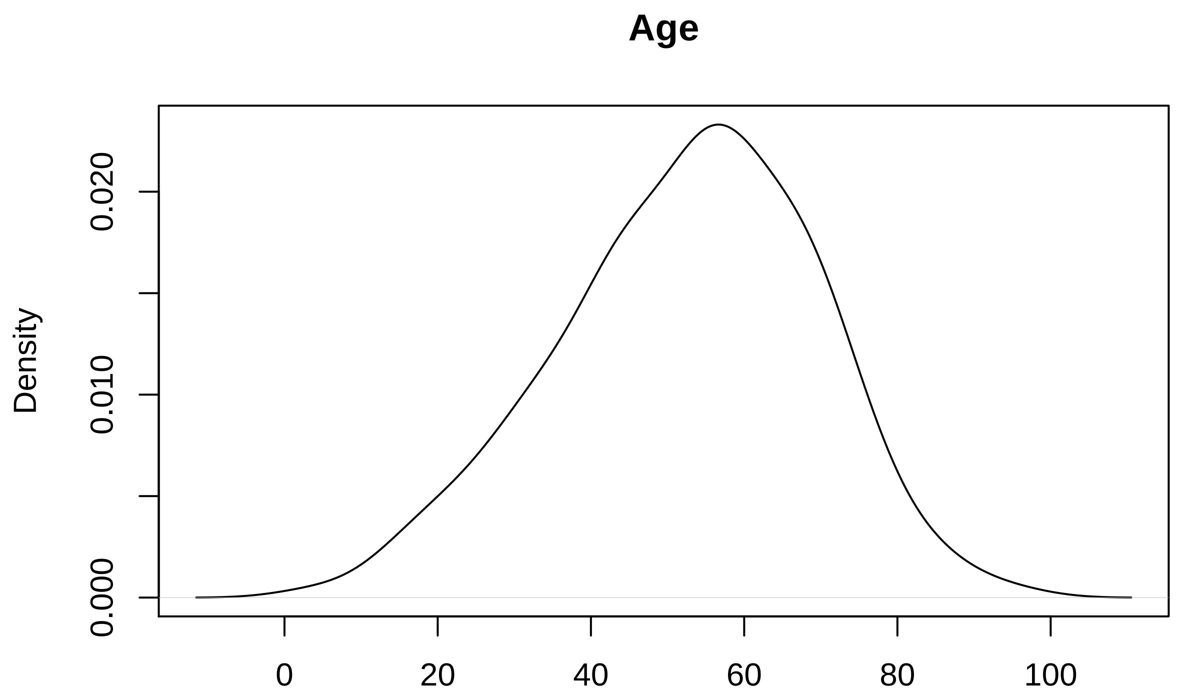


```r
hist(melanoma2$age[melanoma2$status==3],main="Age for non-Melanoma Deaths",xlab="Age")
```

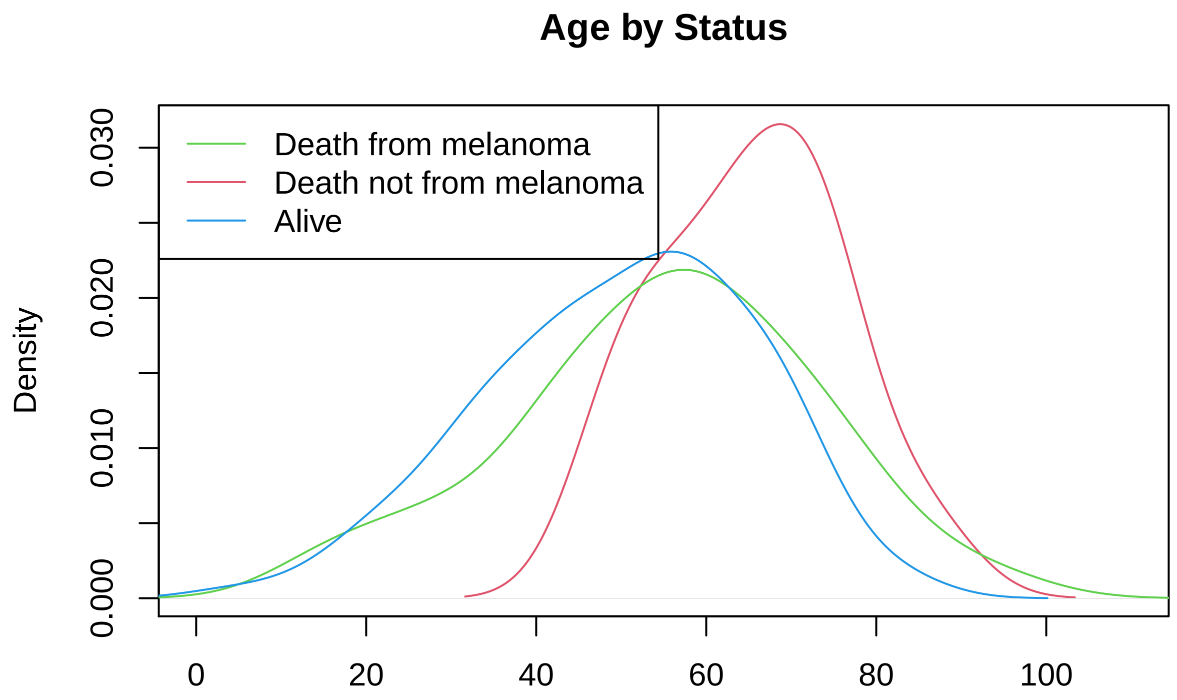


- Density plots (status samples size= 57,134,14)

```
#plot density for age overall
plot(density(melanoma2$age),main="Age",xlab="")
```

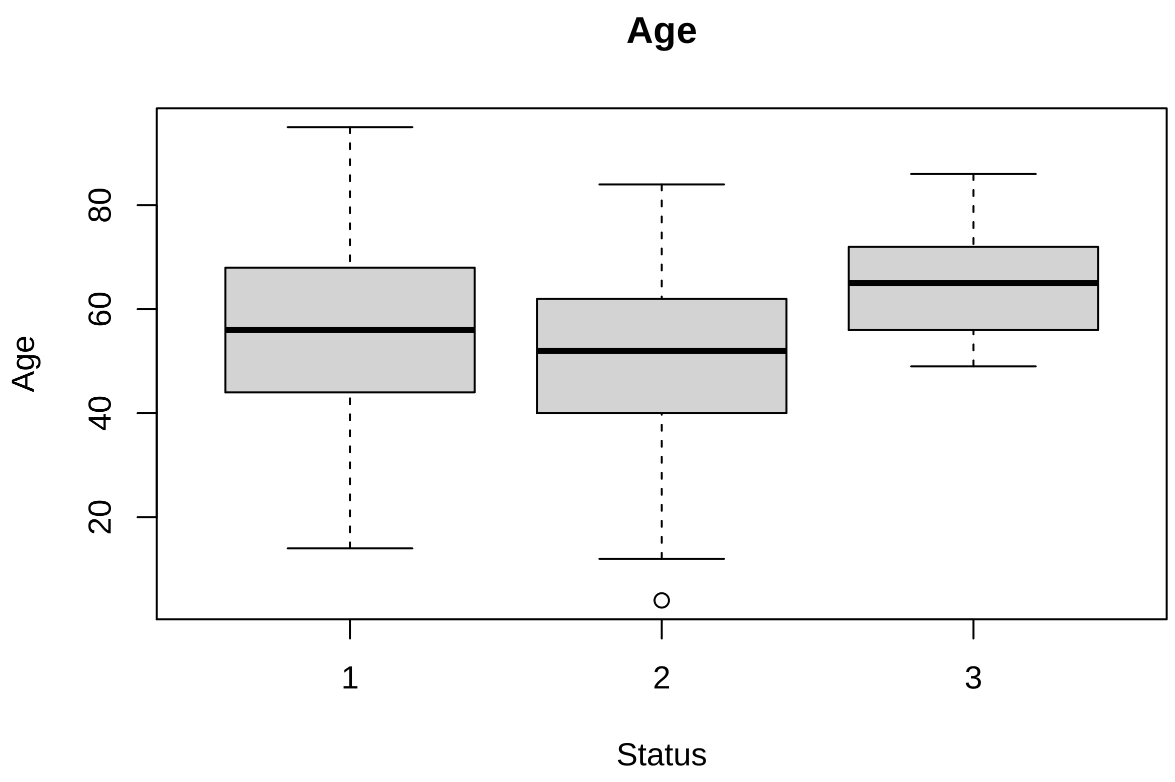


```
#plot density for age by status
plot(density(melanoma2$age[melanoma2$status==3]),main="Age by Status",xlab="",xlim=c(0,110),col=2)
lines(density(melanoma2$age[melanoma2$status==1]),col=3)
lines(density(melanoma2$age[melanoma2$status==2]),col=4)
legend("topleft",c("Death from melanoma","Death not from melanoma","Alive"),col=c(3,2,4),lwd=1)
```



- Boxplot

```
boxplot(age~status,data=melanoma2, main="Age",
   xlab="Status", ylab="Age")
```



t-tests and Wilcoxon rank sum test

- Does the average age differ by status 1 and 2?
- Which test to use? t-test assumes normality. t-test with equal variance or unequal?

```
# t test
t.test(melanoma2$age[melanoma2$status==1],melanoma2$age[melanoma2$status==2],
       alternative = "two.sided",paired = FALSE, var.equal = FALSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  melanoma2$age[melanoma2$status == 1] and melanoma2$age[melanoma2$status == 2]
## t = 1.853, df = 95.424, p-value = 0.06698
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.3623352 10.5228485
## sample estimates:
## mean of x mean of y
##  55.08772  50.00746
```

- p-value=0.07. Average age does not significantly differ for subjects who died by melanoma versus subjects who were still alive.

- Wilcoxon rank sum test may be approriate if normality assumption is not met or samlple size is small. Used to test if the median differs in the 2 groups.

```
#Wilcoxon rank sum exact test (signed rank for paired =True)
wilcox.test(melanoma2$age[melanoma2$status==1],melanoma2$age[melanoma2$status==2],
            alternative = "two.sided",paired = FALSE)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  melanoma2$age[melanoma2$status == 1] and melanoma2$age[melanoma2$status == 2]
## W = 4458.5, p-value = 0.06749
## alternative hypothesis: true location shift is not equal to 0
```

ANOVA and Kruskal-Wallis

- Does the average age differ by status?
- ANOVA assumes normality and vairance equal in all 3 groups.
- The Kruskal-Wallis test is a non parametric test used to test the null hypothesis which states that 'k' number of samples has been drawn from the same population or the identical population with the same or identical median.

```
#ANOVA
oneway.test(age ~ status,data = melanoma2,var.equal = TRUE)
```

```
##
##  One-way analysis of means
##
## data:  age and status
## F = 6.6498, num df = 2, denom df = 202, p-value = 0.001596
```

```
#Kruskall-Wallis
kruskal.test(age ~ status,data = melanoma2)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  age by status
## Kruskal-Wallis chi-squared = 13.028, df = 2, p-value = 0.001483
```

- p-value = 0.00159. The average age differs by status for at least 2 groups.

Chi-square tests

- Is sex associated with status? Yes, p-value = 0.03.

```
table(melanoma2$sex, melanoma2$status)
```

```
##
##      1  2  3
##   0 28 91  7
##   1 29 43  7
```

```r
chisq.test(melanoma2$sex, melanoma2$status)
```

```
##
##  Pearson's Chi-squared test
##
## data:  melanoma2$sex and melanoma2$status
## X-squared = 6.793, df = 2, p-value = 0.03349
```

Correlations

- Is age correlated with sex?

```r
cor(melanoma2$sex,melanoma2$age)
```

```
## [1] 0.06833741
```

```r
# Test correlation (pearson, could use method = "spearman")
cor.test(melanoma2$sex,melanoma2$age,alternative = "two.sided",method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  melanoma2$sex and melanoma2$age
## t = 0.97594, df = 203, p-value = 0.3303
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.06934701  0.20346704
## sample estimates:
##        cor
## 0.06833741
```

- Age is note significantly correlated with sex (p-value=0.33).

Linear regression, just an example to give the corresponding code

- Is age associated with sex?

```r
modelLinear<-lm(age~sex,data=melanoma2)
summary(modelLinear)
```

```
##
## Call:
## lm(formula = age ~ sex, data = melanoma2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.563  -9.899   2.101  13.101  41.101
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   51.563      1.485  34.713   <2e-16 ***
## sex            2.335      2.393   0.976     0.33
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 16.67 on 203 degrees of freedom
## Multiple R-squared:  0.00467,    Adjusted R-squared:  -0.0002331 
## F-statistic: 0.9525 on 1 and 203 DF,  p-value: 0.3303
```

Logistic regression, just an example to give the corresponding code

- Is sex associated with age?

```
modelLogistic<-glm(sex~age,data=melanoma2,family=binomial)
summary(modelLogistic)
```

```
## 
## Call:
## glm(formula = sex ~ age, family = binomial, data = melanoma2)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -1.1112  -1.0009  -0.9318   1.3460   1.5367  
## 
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)  
## (Intercept) -0.916253   0.484627  -1.891   0.0587 .
## age          0.008521   0.008729   0.976   0.3289  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 273.32  on 204  degrees of freedom
## Residual deviance: 272.36  on 203  degrees of freedom
## AIC: 276.36
## 
## Number of Fisher Scoring iterations: 4
```

- Plots for correlation. Need one of the traits to be continuous.

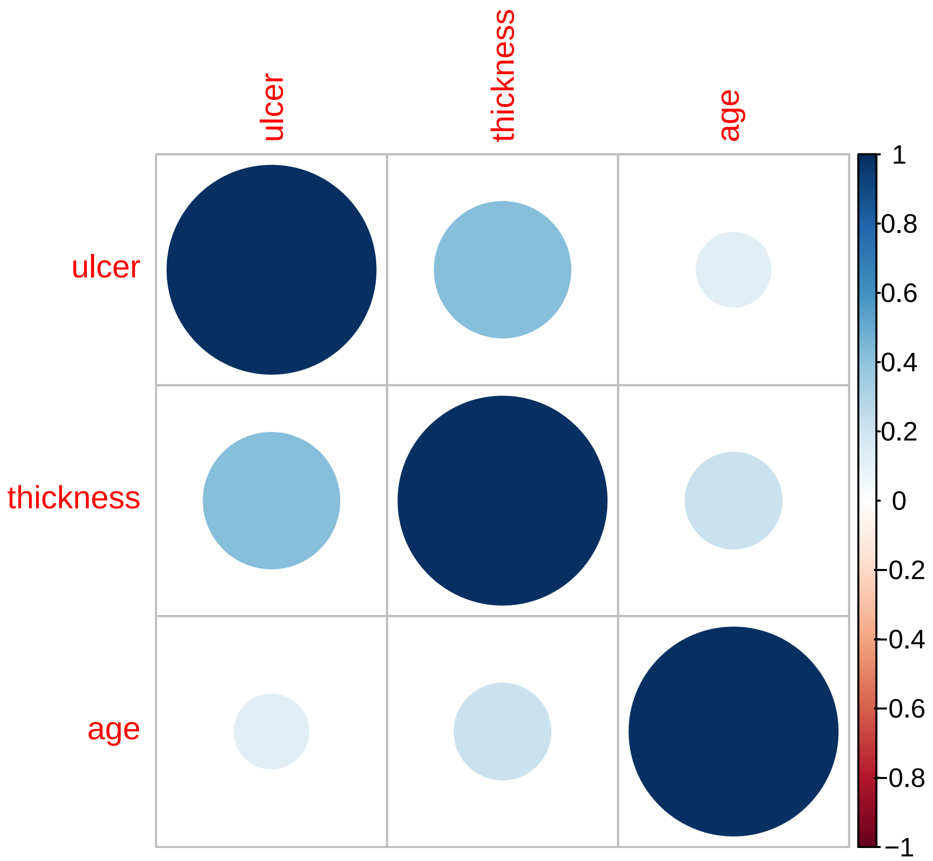
```
#https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html

# Create new matrix with only age, thickness, ulcer
melanoma3<-melanoma2[,c("age","thickness","ulcer")]

# Check
melanoma3[1:5,]
```

```
##   age thickness ulcer
## 1  76      6.76     1
## 2  56      0.65     0
## 3  41      1.34     0
## 4  71      2.90     0
## 5  52     12.08     1
```

```
# Plot correlation
M = cor(melanoma3)
corrplot(M, order = 'AOE')
```



Existing packages

```
# https://cran.r-project.org/web/packages/table1/vignettes/table1-examples.html

# change status variable to text
melanoma2$status <-
   factor(melanoma2$status,
          levels=c(2,1,3),
          labels=c("Alive", # Reference
                   "Melanoma death",
                   "Non-melanoma death"))
melanoma2[1:5,]
```

```
##   time             status sex age year thickness ulcer
## 1   10 Non-melanoma death   1  76 1972      6.76     1
## 2   30 Non-melanoma death   1  56 1968      0.65     0
## 3   35              Alive   1  41 1977      1.34     0
## 4   99 Non-melanoma death   0  71 1968      2.90     0
## 5  185     Melanoma death   1  52 1965     12.08     1
```

```
table1(~ factor(sex) + age + factor(ulcer) + thickness | status, data=melanoma2)
```

```
## Get nicer `table1` LaTeX output by simply installing the `kableExtra` package
```

	Alive	Melanoma death	Non-melanoma death	Overall
	(N=134)	(N=57)	(N=14)	(N=205)
factor(sex)				
0	91 (67.9%)	28 (49.1%)	7 (50.0%)	126 (61.5%)
1	43 (32.1%)	29 (50.9%)	7 (50.0%)	79 (38.5%)
age				
Mean (SD)	50.0 (15.9)	55.1 (17.9)	65.3 (10.9)	52.5 (16.7)
Median [Min, Max]	52.0 [4.00, 84.0]	56.0 [14.0, 95.0]	65.0 [49.0, 86.0]	54.0 [4.00, 95.0]
factor(ulcer)				
0	92 (68.7%)	16 (28.1%)	7 (50.0%)	115 (56.1%)
1	42 (31.3%)	41 (71.9%)	7 (50.0%)	90 (43.9%)
thickness				
Mean (SD)	2.24 (2.33)	4.31 (3.57)	3.72 (3.63)	2.92 (2.96)
Median [Min, Max]	1.36 [0.100, 12.9]	3.54 [0.320, 17.4]	2.26 [0.160, 12.6]	1.94 [0.100, 17.4]

R resources

- Harvard Catalyst or other free Harvard courses https://online-learning.harvard.edu/subject/r
- There are online courses through coursera https://www.coursera.org/learn/r-programming
- Software carpentry offers really fun 2 day workshops. You can check when there is one in Boston and make sure to sign up right away because they fill up quickly https://software-carpentry.org/workshops/