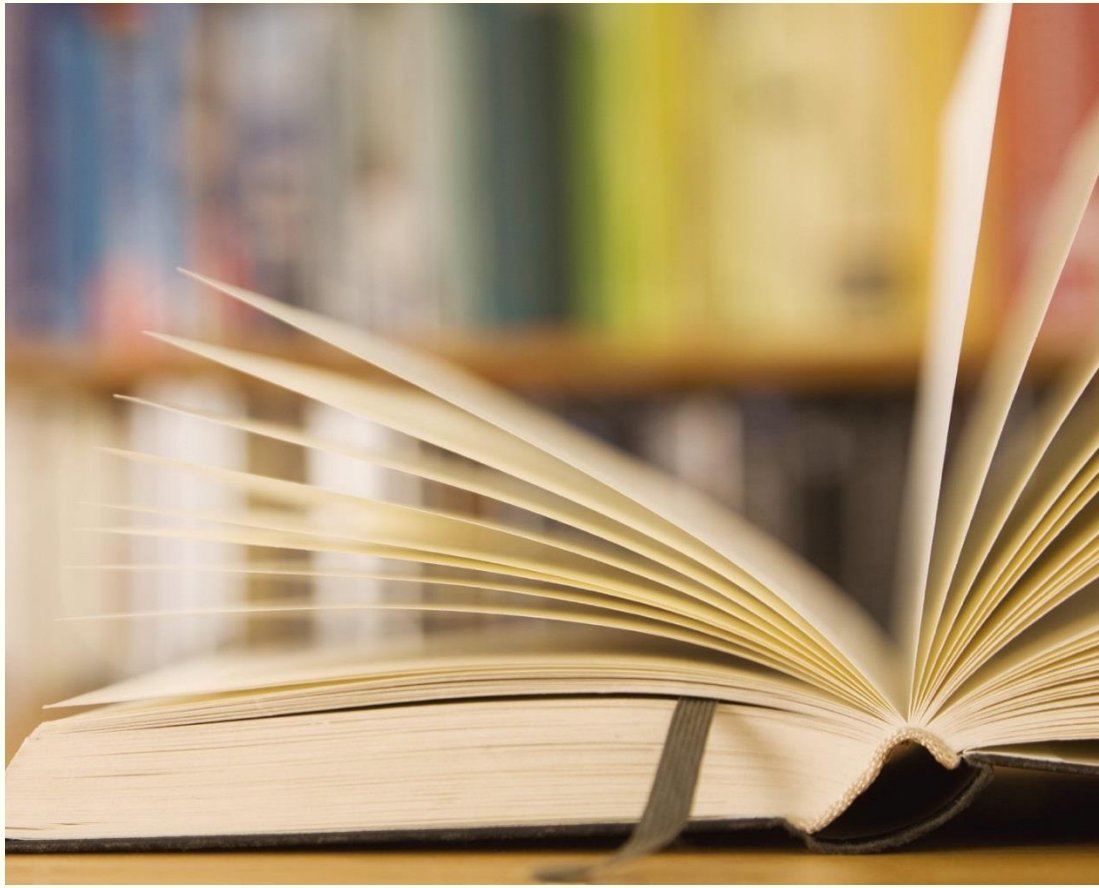
STA 4320 HW 4



Write a function y=x^2

square is the function name

defines the function

input value ; a number

```
square = function(x){
   return(x^2)
}
```

starting and ending of a function

the value to be returned

is x^2 ;

no space between "(" and "x^2"

Apply y = x^2

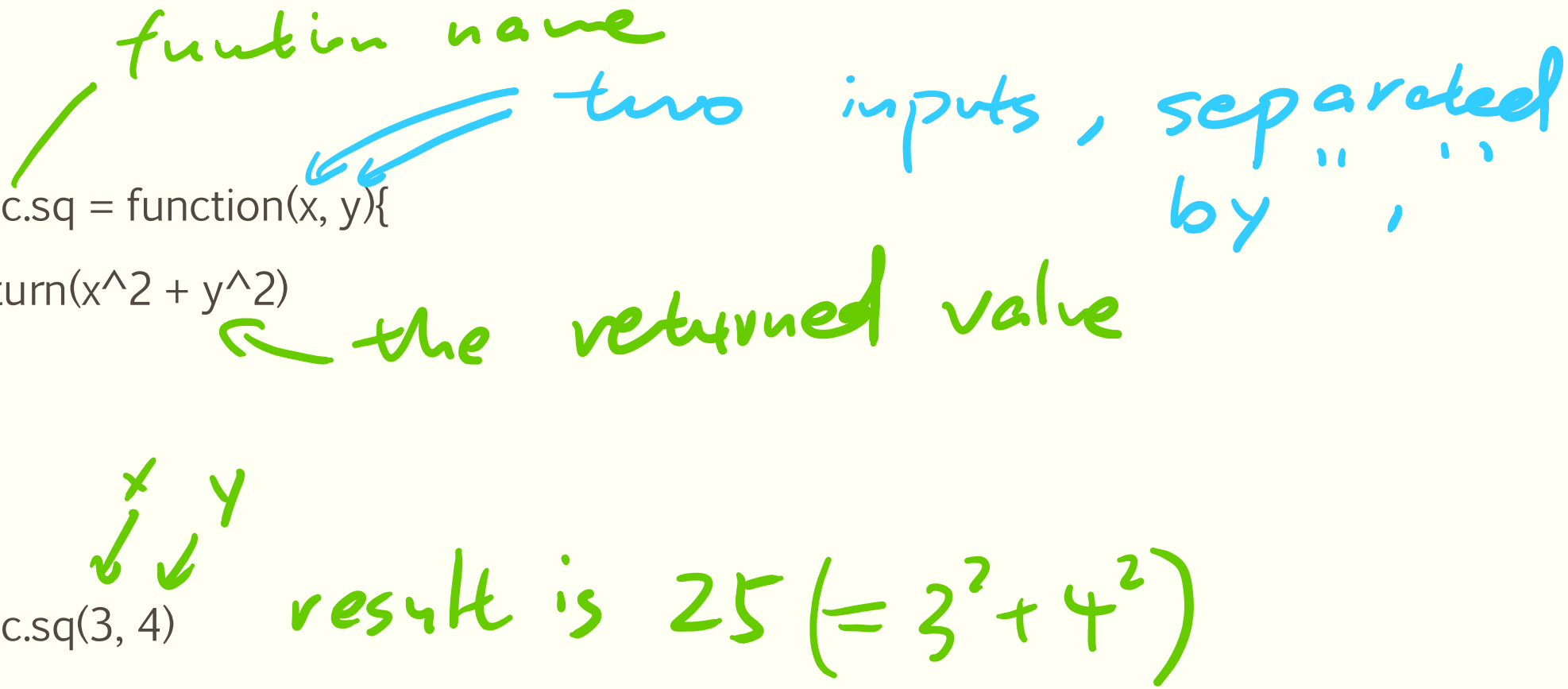
- square(3)

the x value where we want to evaluate x^2

function name

Exercise

- How do we write a function that computes x^2+y^2:



- euc.sq = function(x, y){
- return(x^2 + y^2)
- }
- euc.sq(3, 4)

dist command

- require(stats) # dist command

$$\begin{bmatrix} 0 & 0 \\ 3 & 4 \end{bmatrix} \qquad \vec{a} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \vec{b} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

the "dist" command computes the relevant distance between the top vector and the bottom vector

- rbind(a, b)

means put vector a on top of vector b; forming a matrix

- dist(rbind(a, b), method = "euclidean")

gives 5

Part a

- two.dist = function(a, b, m = "euclidean"){
- X = rbind(a, b)
- return(as.numeric(dist(X, method = m)))
- }

m = "euclidean" is the default method; meaning if we call two.dist(a.b) the m value is by fault "euclidean" so we call the function with only a.b vectors

removes the row and column names

- two.dist(a, b, m = "manhattan")

#7

- To use another distance, we just put that distance name in the m entry

Part b

X : data matrix without response

y : response vector

k : the number of nearest neighbors in KNN

x_new: a new data point that we want to predict $f(\vec{x}_{new})$

```
knn.reg = function(X, y, k, x_new){
   n = nrow(X)

   dist = numeric(n)
   for (i in 1:n){
      dist[i]  = two.dist(X[i, ], x_new, "euclidean")
   }

   k_near_loc = order(dist)[1:k]
   return(mean(y[k_near_loc]))
}
```

store

extract the statistics pairwise distances between Xnew and each row of X

← apply the helper function to compute pairwise distances

← order these pairwise distances and select the k smallest

find all y values corresponding to the k nearest data points and take average

Part c

- X = matrix(1:5, nrow = 5, ncol = 1)
- y = 11:15
- x_new = 2.1
- k = 3
- knn.reg(X, y, k, x_new)

the entries in the matrix

matrix size

call the function name and the arguments

Part d

- require(ISLR2)
- X = Auto[, 2:5]
- y = Auto[, 1]
- k = 10
- x_new = c(6, 310, 140, 3500)
- knn.reg(X, y, k, x_new)

package with Auto data

data matrix of size n by 4

response vector

new $\vec{x}$ to make prediction; the ordering matters

call the function in the appropriate order