Introduction to R

Wildlife Population Ecology

University of Florida

Jennifer Moore

R Development Core Team (2008). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-90005107-0, URL http://www.R-project.org.

RStudio (2012). RStudio: Integrated development environment for R (Version 0.96.122) [Computer software]. Boston, MA. Retrieved September 27, 2014.

Available from http://www.rstudio.org/

Table of Contents

[Installing R 3](#_Toc36366)

[Getting Started 4](#_Toc36367)

[Help in R 5](#_Toc36368)

[Links to R Manuals/Basic Lessons 5](#_Toc36369)

[Setting the Working Directory 6](#_Toc36370)

[Saving your Work 6](#_Toc36371)

[R Packages & Libraries 7](#_Toc36372)

[Entering Data into R 8](#_Toc36373)

[Comments in R 10](#_Toc36374)

[Data Types and Structure 11](#_Toc36375)

[Basics in R - Assignment 17](#_Toc36376)

[Descriptive Statistics 19](#_Toc36377)

[Missing Values 20](#_Toc36378)

[Plotting 22](#_Toc36379)

[Writing Functions 36](#_Toc36380)

[Descriptive Statistics/Plotting – Assignment 37](#_Toc36381)

[Data Analysis – Simple Tests 40](#_Toc36382)

[Basics in R – Assignment Answers 45](#_Toc36383)

[Descriptive Statistics/Plotting – Assignment Answers 50](#_Toc36384)

# Installing R

Download R using one of the following links – depending on whether you have a PC or a Mac. After you have downloaded R, you can download R Studio. R Studio is an open source integrated development environment (IDE) for R. Both R and R studio are free.

Windows Users:

http://cran.at.r-project.org/bin/windows/base/

Mac Users:

http://cran.r-project.org/bin/macosx/

R-Studio:

http://www.rstudio.com/products/rstudio/download/



# 

# Getting Started

In R Studio, there are 4 main windows.

1. Console Window
2. Environment/History Window
3. Files/Plots/Packages/Help Window
4. Script Window (does not open automatically)

The Console Window is where the code is run. You can type R commands directly into this window, or you can run R code or script (which is just a collection of valid R commands) from a script file. All of your commands and outputs are also displayed in the console.

The Environment/History Window is in the top right of the screen. The

Environment tab lists all of the data and values that have been created and saved. This allows you to see what each variable you have created represents. You can also import data into R directly using this window. The History tab keeps track of all of the code that you have run. You can save this into a file.

The Files/Plots/Packages/Help Window is in the bottom right of the screen. The Files tab shows all of the files on your computer; you can also set working directory (see below for details), create new folders, and copy or move files from the Files tab. You can open data files directly from this window to look at, but they must be imported into R Studio for you to be able to use or analyze the data. The Plot tab is where all of the plots you create are displayed, and where they can be saved into files. The Packages tab is a list of all currently installed packages, the place to update packages, and the place to install new packages. The Help tab displays the help files for all of the commands and packages.

The Script Window is created by clicking File > New File > New Script. An R script can be saved, so you can reopen it to continue working on a project, and it can be sent to another person for them to open and edit.

It is best to organize all code within an R Script.

Press <CTRL Enter> to run code from script in the Console Window or use the ‘Run’ button at the top of the Script Window.

# Help in R

Introduction to R manual

http://cran.r-project.org/doc/manuals/R-intro.pdf

R help mailing list

http://r.789695.n4.nabble.com/R-help-f789696.html

R built in help

The help files can be searched in the Help tab in the bottom right corner. In addition, the following commands can be entered into the Console Window to open the specified files (in this case help on the plot function)

> help(plot)

> ?plot

> ??plot

The single question mark ? is used to search for a particular function, the double question mark ?? is used if you do not know the function you want to use, but want to search for potential options.

R built in examples

The example files can be accessed by entering the following command into the Console Window. This will bring up examples of how the function can be used.

> example(plot)

You can also find R help by searching Google!

# Links to R Manuals/Basic Lessons

Here are a few websites with tutorial on learning the basics in R. There are a lot of tutorials, manuals and presentations about various aspects of R available online, so if you aren’t interested in these, there are plenty more out there!

http://tryr.codeschool.com/levels/1/challenges/1 https://www.datacamp.com/courses/introduction-to-r

http://swirlstats.com/students.html

# Setting the Working Directory

Before beginning work in R/R studio it is important to set the working directory. This is the folder where all files that you are going to access are located, as well as where all files that you create will be saved.

You can set working directory in three ways: using Files tab and using R commands. In the Files tab in the bottom left corner, locate the folder that you want as your working directory. Once you have clicked on the folder and it is open, under ‘More’ (menu at the top of the Files tab) click ‘Set as Working Directory’. An alternative way of achieving the same goal is to use R Studio main menu -> Session -> Set working directory.

You can also set working directory using the setwd (stands for set working directory) command as follows (you need know the full name of the directory or folder):

> setwd(“C:/Statistics/R/”)

>setwd(“C:\\Statistics\\R\\”)

Note: All slashes needs to be forwards slashes (/) or double backward slashes (\\).

Also, R is case sensitive, so make sure capital letters are used only when needed.

# Saving your Work

You can save everything in all of the windows in R Studio (data, R objects etc.) using save.image command.

> save.image(“File.R”)

This will create a file called File.R in your working directory.

>load(“File.R”)

This will load your data back so you can keep working where you left off.

This is helpful where you are working with code that takes a long time to run, because you do not have to rerun it every time you want to view your results.

You can also save just the R script or single plots that you have created.

# R Packages & Libraries

The functions in R are organized in packages or libraries. The commonly used packages are preinstalled within R. However, if you need to use more specialized packages, you will need to install and then load the package before you can use functions available in that package.

Packages are installed and loaded in the Packages tab in the bottom right corner or through the dropdown menu at the top of the screen (Tools -> Install Packages).

If a package is already installed on your computer, it will be listed in this tab. To load this package so you can use it, hit the checkmark next to the name of the package.

If a package is not installed, click the ‘Install’ button at the top of the tab, type in the name of the package, and click ‘Install’. Once it has been installed click the checkmark next to the name to load it.

Installed packages can also be loaded from the command line or within a script using the library function.

>library(popbio)

This would load the previously installed package ‘popbio’.

A list of all available packages can be found here:

http://cran.r-project.org/web/packages/available\_packages\_by\_name.html

# Entering Data into R

Data can be entered into R using a variety of methods.

1. Manually – entered into the command line
2. Imported – entered in Excel (or other software – see format below), saved as

.csv or .txt files, and imported into R

* 1. Using the Environment tab
  2. Using ‘read’ commands

## Manual Entry

Data can be typed directly into an R script.

As a calculator

> 10 + 5

[1] 15

Compute a sum

> sum(1,5,10)

[1] 16

Create a vector of numbers called Numbers

> Numbers <- c(5,4,7)

> Numbers

[1] 5 4 7

Notes

* Numbers is the name of the vector containing the numbers 5, 4, 7. To display the contents of the vector type ‘Numbers’ into the console.

* The c used in the vector above stands for ‘concatenate’ or ‘combine’. It is used whenever a list of data points is entered into R.

## Importing CSV and TXT Files

Data files can only be imported into R if they are saved as a CSV or TXT file. You can enter data in Excel or other spreadsheet programs, and then save the file into one of these two formats before it can be imported for use in R.

A CSV file is recommended for use in R.

Using the Environment tab

* In the Environment tab, in the top right corner click on ‘Import Dataset’, and then ‘From Text File…’.
* Choose the file that you want to import.
* Choose a name for the data – enter into the ‘Name’ blank. (This name cannot contain any spaces)
* Choose yes or no depending on whether or not the file has a Heading (column names)
* Choose the separator, decimal, quote, and na.strings depending on your data  Click import

Read Commands

The following command can be typed into the console to import the data.

For a CSV file:

> data <- read.csv(“data.csv”, header = TRUE) For a text file:

> data <- read.table(“data.txt”, header = TRUE, sep=” ”)

The first argument is the file name (found within the working directory set earlier).

The second argument, header, refers to whether or not the column names are listed in the first line of the file, TRUE is used for yes, FALSE for no.

The third argument, sep, refers to what is separating the values. A CSV uses commas, but a text document could use semicolons, colons, etc.

Import the chimpanzee dataset into R using either of the two methods explained above. Make sure to save it as a csv file in Excel before importing it into R. When you import it into R, name the variable containing the data ‘chimp’ and ‘. We will use it later! You can view the data after it is imported in

R by using the command View, or by double clicking the file in the

Environment tab. This dataset gives the population number of chimpanzees in Gombe National Park, Tanzania from 1964-1973.



# Comments in R

Comments are notes that you include in your code, but that are not run. R does not run any commands included in your comments.

It is good practice to always put comments into your code, so you or anyone else who access your code later knows what you were doing. This might seem unnecessary for simple code, but will be very useful with more complicated code.

In R, you can comment a line by putting the # sign in front of the text. If you want to comment a large chunk of code all at once, highlight the code and then use the dropdown menu (Code -> Comment/Uncomment Lines).

>#this is a comment

>a = c(1,2,3)

>#creates a vector, a, containing the numbers 1, 2, and 3

>sum(a)

>#calculates the sum of the numbers in vector a

# Data Types and Structure

R can use many data types, such as numeric, categorical, and ordinal. They are organized in various structures, which dictates how they are manipulated, accessed and used by R.

Each data point or a variable can be numeric, a character, or a factor.

* Numeric – a number (e.g. 1, 5.3, -2, 4)
* Character – string of text (e.g. red, one, monkey)
* Factor – level (e.g. male/female, high/low)

Data in R can be organized as

1. vectors
2. matrices
3. data frames

## Vectors

Vectors are one-dimensional arrays containing numbers, or characters.

>a <- c(1,2,5.3,6,-2,4) # numeric vector

>b <- c("one","two","three") # character vector

>c <-c(1, “one”) #mixed vector

Vectors can also be created using sequences of numbers, or repetitions of the same number.

>5:9

[1] 5 6 7 8 9

>seq(5,9)

[1] 5 6 7 8 9

>seq(5,9,0.5)

[1] 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0

>rep(5,3)

[1] 5 5 5

Note: Characters must be entered inside double quotes.

Access the elements of the vectors using their position in the vector in []

> a[3] #3rd element of vector a

[1] 5.3

>a[c(2,4)] # 2nd and 4th elements of vector a [1] 2 6

>a[3:5] #return the 3rd, 4th, and 5th element of vector a

[1] 5.3 6.0 -2.0

Add elements to the vector

> a[7] <- 0 #add a 7th element to the vector

> a

[1] 1.0 2.0 5.3 6.0 -2.0 4.0 0.0

Change elements in the vector

> a[1] <- 0 #change the first element to 0 instead of 1

> a

[1] 0.0 2.0 5.3 6.0 -2.0 4.0 0.0

Add a value to every element of the vector

> a + 1

[1] 1.0 3.0 6.3 7.0 -1.0 5.0 1.0

Multiply a value to every element of the vector (you can also subtract or divide, etc.) > a \* 2

[1] 0.0 4.0 10.6 12.0 -4.0 8.0 0.0

Add two vectors together (or subtract, multiple, divide)

> b <- 1:7

> a+b

[1] 1.0 4.0 8.3 10.0 3.0 10.0 7.0

The above commands simply output the desired results – if you want to use the new vector it most be saved as a new variable.

> c <- a+b

[1] 1.0 4.0 8.3 10.0 3.0 10.0 7.0

#the sum of vectors a and b is now saved as variable c

## Matrices

Matrices are two-dimensional, for example a 4 x 5 matrix has 4 rows and 5 columns. All values within the matrix must be of the same type; i.e. all numeric or all characters

> mymatrix <- matrix(1:4, nrow=2, ncol=2)

> mymatrix

[,1] [,2]

[1,] 1 3

[2,] 2 4

The values of a matrix are accessed using their position, like for vectors, but for matrices it is necessary to designate the row and column.

> mymatrix[1,2] #the value in row 1 column 2

[1] 3

> mymatrix[2,1] #the value in row 2 column 1

[1] 2

> mymatrix[,2] #the values for all rows, column 2

[1] 3 4

> mymatrix[1,] # the values for row 1, and all columns

[1] 1 3

> mymatrix[1:2,] #the values from row 1 to 2, and all columns

[,1] [,2]

[1,] 1 3

[2,] 2 4

Change the value in row 1 column 2 to 0

>mymatrix[1,2] <- 0

>mymatrix[1,2]

[1] 0

## Data Frames

Data frames are a more general form of a matrix. The values do not have to be of the same type, but instead one data frame can contain numbers, characters, and factors.

> d <- c(1,2,3,4)

> e <- c("red", "white", "red", "blue")

> f <- factor(c("high", "low", "high", "high"))

> mydata <- data.frame(d,e,f)

> names(mydata) <- c("ID", "Color", "Level")

> mydata

ID Color Level 1 1 red high

1. 2 white low
2. 3 red high
3. 4 blue high

The values of a data frame can be accessed in a variety of ways.

> mydata[1] #calls column 1

ID

1. 1
2. 2
3. 3
4. 4

> mydata["ID"] #another way to access column 1 – using the column name

ID

1. 1
2. 2
3. 3
4. 4

> mydata$ID #last way to access column 1 – also using the column name [1] 1 2 3 4

> mydata[1,] #access the first row

ID Color Level

1 1 red high

## Useful Functions for Data Types

Note: R has many datasets built into the program that you can use for practicing. Type data() into the console to see a list of all of the available datasets. We will use the Loblolly dataset. If you type help(Loblolly) you can read all about the data included in this dataset.



> str(Loblolly) # structure of the dataframe

Classes ‘nfnGroupedData’, ‘nfGroupedData’, ‘groupedData’ and 'data.frame':

84 obs. of 3 variables:

$ height: num 4.51 10.89 28.72 41.74 52.7 ...

$ age : num 3 5 10 15 20 25 3 5 10 15 ...

$ Seed : Ord.factor w/ 14 levels "329"<"327"<"325"<..: 10 10 10 10 10 10 13 13 13 13

...

* attr(\*, "formula")=Class 'formula' length 3 height ~ age | Seed

.. ..- attr(\*, ".Environment")=<environment: R\_EmptyEnv>

* attr(\*, "labels")=List of 2

..$ x: chr "Age of tree"

..$ y: chr "Height of tree"

* attr(\*, "units")=List of 2

..$ x: chr "(yr)"

..$ y: chr "(ft)"

>Loblolly #displays all of the data

> head(Loblolly) #displays first 6 rows of data

> tail(Loblolly) #displays last 6 rows of data

> nrow(Loblolly) #number of rows

[1] 84

> ncol(iris) #number of columns

[1] 3

> names(Loblolly) # column names

[1] "height" "age" "Seed"

> length(Loblolly$height) #number of data points in that column

[1] 84

## Subsetting Data

Sometimes if we have a large dataset for multiple categories, we might want to subset it into smaller datasets for analysis.

For example, lets look at the Loblolly dataset, which gives us information on the height and age of loblolly pine trees from different seeds.

We can create separate datasets for each of the seeds.

> seed301 <- subset(Loblolly, Loblolly$Seed == “301”)

#this creates a new dataframe called ‘seed301’ which contains all columns of data for just the seed numbered 301

> seed303 <- subset(Loblolly, Loblolly$Seed == “303”)

#this creates a new dataframe called ‘seed303’ which contains all columns of data for just the seed numbered 303

We can also subset based on criteria – in this case such as age or height.

> Loblolly2 <- subset(Loblolly, Loblolly$age < 5)

#this dataframe only contains the data on Loblolly trees at less than 5 years of age (only the trees at 3 years of age)

# Basics in R - Assignment



1. Open up R Studio and set your working directory to the folder you will be working from (the one that contains the data files you want to use).

1. Save your R script into this folder. Include comments throughout your script to explain what each command is doing.

1. What is one way that you can find help if you forget how to use the subset function?

1. Using R as a calculator, what is the sum of 15 and 32?

1. Using R as a calculator, what is the product of 34 and 543?

1. Using R as a calculator, what is the value of 24?

1. Create a vector of the numbers 3,8,4,2 and give it the name A.

1. Create a vector of the characters “monkey”, “rat”, “snake”, and give it the name B.

1. Create a vector of the sequence of numbers between 89 and 143, and give it the name C.

1. Create a vector containing 15 of the number 3 – use the rep command, and give it the name D.

1. What is the 3rd element in the vector A?

1. What are the last two elements of the vector B?

1. What is the 4th and 30th element in the vector C?

1. What is the sum of all of the elements in the vector D?

1. Add the character “bird” to the end of vector B.

1. Change the first element in vector D to the number 5.

1. Multiple vector A by 3 and save the results as vector A2.

1. Add 5 to every element of vector C and save the results as vector C2.

1. Import the ‘lionCrater’ file into R and save the dataframe as ‘Lion”. This file gives the population numbers for lions in the Ngorongoro Crater in Tanzania.

1. Import the ‘blackRhinoCrater’ file into R and save the dataframe as “Rhino”. This file gives the population numbers for black rhinos in the Ngorongoro Crater in Tanzania.

1. Display the structure of the Lion data frame. What is included? How many observations and how many variables? What are the variable names?

1. What are three ways to display the data in column 1 of the Lion data frame?

1. How do you access the 5th row of data in the Lion data frame?

1. Display the first 6 rows of the Rhino data frame.

1. Display the last 6 rows of the Rhino data frame.

1. How many rows are in the Rhino data frame?

1. How many columns are in the Rhino data frame?

1. Look at the Year column for both Lions and Rhinos. Which years overlap? Create two new data frames Lion2 and Rhino2 that only contain the overlapping years.

1. Create a new data frame called Lion3 that only contains the years when the population is over 50 individuals. How many rows are in this new data frame?

1. Create a new data frame called Rhino3 that only contains the years when the population of rhinos is less than 12 individuals. How many rows are in this new data frame?

# Descriptive Statistics

We will use the chimpanzee dataset that we imported earlier to explore basic descriptive statistics. But first, we need to look at the structure of the dataset.

> str(chimp)

'data.frame': 10 obs. of 2 variables:

$ Year : int 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973

$ Population: int 41 51 62 53 46 43 45 50 51 39

This dataset has two variables, Year and Population. Let’s explore the Population variable.

> chimp$Population #display the data in the Population column

[1] 41 51 62 53 46 43 45 50 51 39

Minimum – smallest value

> min(chimp$Population)

[1] 39

Maximum – largest value

> max(chimp$Population)

[1] 62

Range (Minimum and Maximum)

> range(chimp$Population)

[1] 39 62

Median – middle number when all of the values are organized from smallest to largest

> median(chimp$Population)

[1] 48

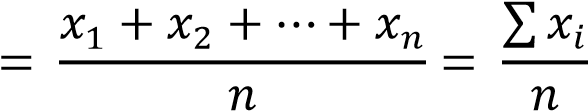
Quantile

> quantile(chimp$Population)

0% 25% 50% 75% 100%

39.0 43.5 48.0 51.0 62.0

Mean is a measure of central tendency of the data, and is calculated as:



> mean(chimp$Population)

[1] 48.1

Variance is a measure of spread of the data, and is calculated as

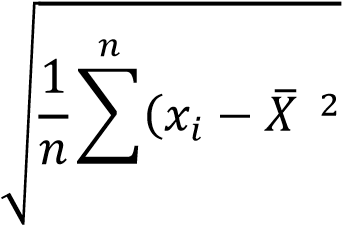
1

= − )

> var(chimp$Population)

[1] 45.65556

Standard deviation is just the square root of variance (how much the values typically vary from the average value):

= )

> sd(chimp$Population)

[1] 6.756889

The summary command calculates a group of the descriptive statistics all at once.

> summary(chimp$Population)

Min. 1st Qu. Median Mean 3rd Qu. Max.

39.0 43.5 48.0 48.1 51.0 62.0

All of the values are rounded to the nearest integer before these values are calculated, which is why they are slightly different than the values calculated above.

# Missing Values

Datasets are rarely perfect, and there is often data missing. First, here is a simple example with one value missing.

> a <- c(1,2,3,NA,5)

> mean(a)

[1] NA

> mean(a, na.rm = TRUE)

[1] 2.75

Next, let’s us see how to deal with missing data in a real dataset. Let us import the ‘lion’ dataset. This dataset has the population number of lions in the Serengeti Plains from 1966-1990.



First import the dataset and name it ‘lion’.

>lion <- read.csv(“lion.csv”, header = T)

If you look at the data, you’ll see that there are no population sizes for the years 1969-1973. Now, what happens if we try to find the mean of the population size?

> mean(lion$Population)

[1] NA

We can use the trick we just learned to get the mean population size.

> mean(lion$Population, na.rm = TRUE)

[1] 34.05

However, instead of having to include the ’na.rm’ argument every time, we can just delete the rows that are missing data.

> lion2 <- na.omit(lion)

# Plotting

## Basic Plot

We will plot the chimpanzee and lion population numbers.

>plot(chimp$Year, chimp$Population)

Creates a plot with years on the x-axis and chimpanzee population size on the y-axis.

1964

1966

1968

1970

1972

4045505560

chimp$Population

chimp$Year

What does this plot tell us?

The population increases to its highest point at 1966 and then decreases to 1969. The population then increases again until 1972 and decreases in 1973.

We can also create a plot for the lion population numbers.

1970

1975

1980

1985

1990

25303540

lion$Population

lion$Year

You will notice that there are no population sizes for the years 1969-1973 – these are the rows that we deleted earlier because the population sizes were unknown. However, you’ll notice that the x-axis still goes from 1966-1990 but there are not values for those years that are missing.

Now, let’s us work on customizing our plot. There are many more arguments that can be added to the plot command.

* Type – the type of plot to be drawn o Points (default) o Lines
* Main – the title of the plot
* Xlab, Ylab – the axis labels
* Xlim, Ylim – the range of values on each axis
* Col – color of symbols

Let’s add these arguments to our chimpanzee plot. Let’s now make a line plot, instead of a point plot – add a title and axis labels – and change the color to blue.

> plot(chimp$Year, chimp$Population, type = "l", main = "Chimpanzee Population in

Gombe National Park", xlab = "Year", ylab = "Population", col = "blue")

Chimpanzee Population in Gombe National Park

1964

1966

1968

1970

1972

4045505560

Population

Year

Next, let’s fix our lion plot. Let’s change it to a line plot, add a title and axis labels, and change the color to red.

> plot(lion$Year, lion$Population, type = "l", main = "Lion Population in the Serengeti

Plains", xlab = "Year", ylab = "Population", col = "red")

## Lion Population in the Serengeti Plains

1970

1975

1980

1985

1990

25303540

Population

Year

You will notice there is a break in the line for the years where there is missing data.

### Other Common Plots

Barplot

A barplot is a graphical way to display and compare the values of different categories within a dataset.

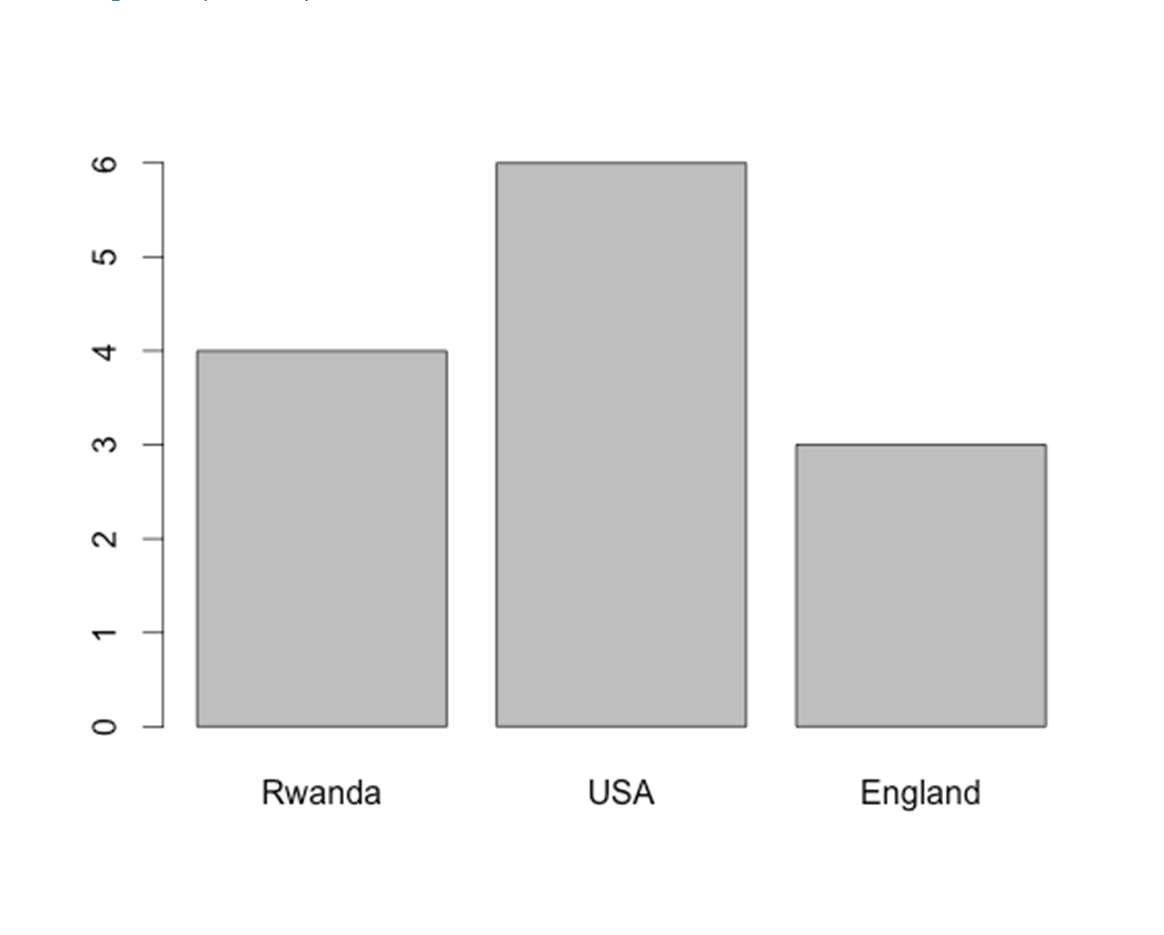
Create a barplot for the following vector and add labels for each bar.

> data <- c(4,6,3)

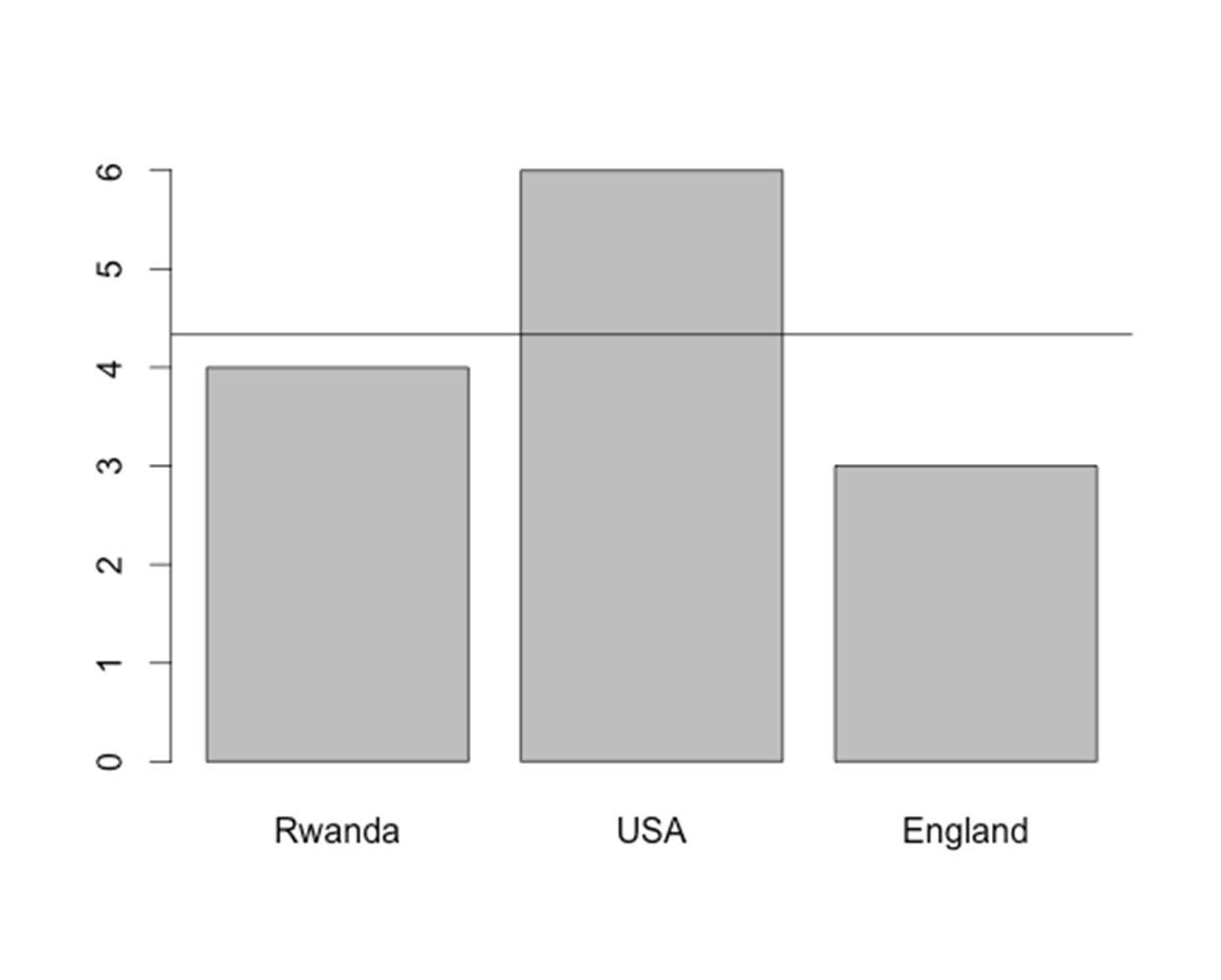
> barplot(data)

> names(data) <- c(“Rwanda”, “USA”, “England”)

> barplot(data)



>abline(h=mean(data)) #adds a horizontal line to the graph at the mean value



Boxplot

A boxplot shows the distribution of the data based on the five number summary: minimum, first quartile, median, third quartile, and maximum.

Create a side-by-side boxplot for the Loblolly dataset.

> boxplot(Loblolly$age, Loblolly$height, main = "Loblolly Pine", names = c("Age", "Height"))

## Loblolly Pine

Height

Age

102030405060

Histogram

A histogram is a graphical way to represent the distribution of a dataset. The bars represent the frequency of each value within the dataset.

Create a histogram of the Loblolly pine height from the Loblolly dataset.

> hist(Loblolly$height, xlab = "Height ", main = "Histogram of Loblolly pine height")

Histogram of Loblolly Pine Height

Frequency

0

10

20

30

40

50

60

70

05101520

Tree Height (ft)

Add a green vertical line to show the mean tree height.

>abline(v=mean(Loblolly$height), col = “green”)

## Histogram of Loblolly Pine Height

Frequency

10

20

30

40

50

60

70

0

05101520

Tree Height (ft)

### Multiple Plots

Let’s do a quick analysis of two new datasets (Blue Wildebeest and African Buffalo) – and then learn how to create multiple plots on the same page, as well as one plot with multiple sets of data.



Import the Blue Wildebeest (wildebeest.csv) and the African buffalo (buffalo.csv) datasets. These datasets give population numbers of wildebeest and buffalo from the Serengeti Plains.

> wildebeest <- read.csv("wildebeest.csv", header = T)

> buffalo <- read.csv("buffalo.csv", header = T)

Create a line plot of the population sizes for each of these species – make sure to include a title, axis labels, and make each plot a different color.

> plot(wildebeest$Year, wildebeest$Population, main = "Wildebeest Population Size", xlab = "Year", ylab = "Population", col = "orange", type = "l")

Wildebeest Population Size

1960

1965

1970

1975

1980

1985

400000600000800000100000012000001400000

Population

Year

> plot(buffalo$Year, buffalo$Population, main = "Buffalo Population Size", xlab =

"Year", ylab = "Population", col = "purple", type = "l")

## Buffalo Population Size

1966

1968

1970

1972

1974

1976

40000500006000070000

Population

Year

Now, let’s keep these same plots – but what if we want them both on the same page, for example for in a publication?

>par(mfrow=c(2,1)) #two rows and one column of plots on the page

> plot(wildebeest$Year, wildebeest$Population, main = "Wildebeest Population Size", xlab = "Year", ylab = "Population", col = "orange", type = "l")

> plot(buffalo$Year, buffalo$Population, main = "Buffalo Population Size", xlab =

"Year", ylab = "Population", col = "purple", type = "l")

## Wildebeest Population Size

1960

1965

1970

1975

1980

1985

4000001200000

Population

Year

## Buffalo Population Size

1966

1968

1970

1972

1974

1976

4000060000

Population

Year

If we want to instead have two columns and one row of plots (i.e. two plots side by side) we would use par(mfrow=c(1,2)). For two columns and three rows (i.e. six plots total on page) we would use par(mfrow=c(3,2)) and so forth.

Note: After changing the number of columns/rows on the page – it is necessary to change it back to 1 row and 1 column to have just one plot on the page again (par(mfrow=c(1,1)).

### Multiple Sets of Data on the Same Plot & Legends

Now, let’s use the same data again for wildebeest and buffalo, but create just one plot showing both sets of data. We will also include a legend to show which set of data relates to which animal.

These two datasets cover different time periods – so before we create our plot let’s subset the wildebeest dataset to cover the same years as the buffalo dataset.

> wildebeest2 <- subset(wildebeest, wildebeest$Year >1964 & wildebeest$Year < 1977)

Now let’s put the window back to one column and one row, and then create a plot of the wildebeest data in blue – the same way we created plots before. We will also add the ylim argument, which sets the limits of the y-axis. We will set the minimum value to the minimum value of the buffalo dataset and the maximum value to the maximum value of the wildebeest dataset – since these are the two extreme values.

>par(mfrow=c(1,1))

> plot(wildebeest2$Year, wildebeest2$Population, xlab = "Year", ylab = "Population

Size", col = "blue", type = "l", main = "Wildebeest & Buffalo Populations in the Serengeti Plains", ylim = c(41000, 1306603))

Now, let’s add the buffalo data to the same plot using the lines command. If we wanted to add the data as points instead of the lines we would use the points command. We will plot the buffalo data in red.

> lines(buffalo$Year, buffalo$Population, col = "red")

For the last step, let’s add a legend to the plot so that we know which set of data represent which animal. We will put the legend in the top left corner of the plot.

> legend("topleft", col = c("blue", "red"), lty = 1, legend = c("Wildebeest", "Buffalo"))

In the above R code, you’ll notice the lty argument. This stands for line type, with line type 1 as the solid line – try switching it to other numbers between 1 and 6 for other line types. If you had a plot using points instead of lines, you would use the pch command instead – which stands for point character. Point character 16 is a solid circle. Try using point characters 0-25 for different symbols!

## Wildebeest & Buffalo Populations in the Serengeti Plains

1966

1968

1970

1972

1974

1976

02000006000001000000

Population Size

Wildebeest

Buffalo

Year

# Writing Functions

In addition, we can write our own function to calculate all of the descriptive statistics at once – let’s try it with our chimpanzee dataset.

> summary.stat = function(x) {

+ n = length(x)

+ my.mean = mean(x)

+ my.var = var(x)

+ my.sd = sd(x)

+ my.min = min(x)

+ my.max = max(x)

+ results = list(mean=my.mean,sd=my.sd,max=my.max,min=my.min,n=n)

+ return(results)

+ }

> summary.stat(chimp$Population)

$mean

[1] 48.1

$sd

[1] 6.756889

$max

[1] 62

$min

[1] 39

$n

[1] 10

# Descriptive Statistics/Plotting – Assignment

1. Import the Lion and Black Rhino files into R that we were using in the last assignment (‘lionCrater.csv’ and ‘blackRhinoCrater.csv’).

1. What is the minimum and maximum population size for lions?

1. What is the mean, median, and standard deviation of the population size of lions?

1. What is the minimum and maximum population size for black rhinos?

1. What are the quantile (0%,25%,50%,75%,100%) population sizes for black rhinos?

1. Use the summary command to find the descriptive statistics of population size for black rhinos. What is the mean and median?

1. What is the variance and standard deviation of population size for black rhinos?

1. Create a plot of the lion population size with the year on the x-axis. Include a title, axis labels, and make the points blue. Can you figure out how to change the open circle points to another shape? Try and change them to closed circles. Is the population increasing or decreasing? What year is the minimum and maximum population sizes?

1. Create a plot of the black rhino population size with the year on the x-axis. This time make it a line plot. Include a title, axis labels, and the make the line dark green. Can you figure out how to change the line type to a dotted line? Is the population increasing or decreasing? What year is the minimum and maximum population sizes?

1. Put the two plots that you just made onto one page with one column and two rows.

1. Create two new data frames Lion2 and Rhino2 that contain the subset of years that overlap between the two datasets (like we did in the last assignment). These new datasets will be used for the next plot.

1. Create a line plot of the lion population sizes. On the same plot add the black rhino population sizes as points. Make the lion population line purple and the black rhino population points red. Includes a title and axis labels. You will need to set the limits of the y-axis from (0,100) otherwise the rhino points will not be visible when you add them to the plot.

1. Add a legend to the top right of the plot you just made – make sure next to lion you have a purple line, and next to black rhino you have a red point.

1. Create a barplot showing the minimum population size for black rhino and lion. Make sure to include a title and labels for each of the bars.

1. Create a side-by-side boxplot of the population size for lions and black rhinos – make sure to include a title, and a label below each box to show which animal it represents.

1. Create a histogram of the population size for black rhinos. Add an orange vertical line to show the mean population size.

39

# Data Analysis – Simple Tests

## 2 Sample t-test

Goal: Compare the means of two independent samples, X1 and X2

: =

: ≠

: <

: >

Let’s look at two species of iris from the built-in iris dataset, and compare the means of the petal lengths.

> data(iris)

First, subset the data for each of the two species.

> setosa <- subset(iris, iris$Species == "setosa")

> versicolor <- subset(iris, iris$Species == "versicolor")

Next, use the t-test to compare the means.

First, use a 2-sided t-test to look at whether to accept the null hypothesis (means are equal) or the reject the null for the first alternate hypothesis (means are not equal).

2-sided t-test (alternative hypothesis 1)

> t.test(setosa$Petal.Length, versicolor$Petal.Length)

Welch Two Sample t-test

data: setosa$Petal.Length and versicolor$Petal.Length

t = -39.4927, df = 62.14, p-value < 2.2e-16 alternative hypothesis: true difference in means is not equal to 0 95 percent confidence interval: -2.939618 -2.656382 sample estimates: mean of x mean of y

1.462 4.260

With a p-value of <2.2e-16 we can reject the null hypothesis, and conclude that mean petal length significantly differs between the two species (alternate hypothesis 1).

From here, we can then run one-sided t-tests to see if alternate hypotheses 2 or 3 is accepted.

1-sided t-test (alternative hypothesis 2 – less than)

> t.test(setosa$Petal.Length, versicolor$Petal.Length, alternative = "less")

Welch Two Sample t-test

data: setosa$Petal.Length and versicolor$Petal.Length

t = -39.4927, df = 62.14, p-value < 2.2e-16 alternative hypothesis: true difference in means is less than 0 95 percent confidence interval:

-Inf -2.679701 sample estimates: mean of x mean of y

1.462 4.260

1 sided t-test (alternative hypothesis 3 – greater than)

> t.test(setosa$Petal.Length, versicolor$Petal.Length, alternative = "greater")

Welch Two Sample t-test

data: setosa$Petal.Length and versicolor$Petal.Length

t = -39.4927, df = 62.14, p-value = 1 alternative hypothesis: true difference in means is greater than 0 95 percent confidence interval:

-2.916299 Inf sample estimates: mean of x mean of y

1.462 4.260

The results show that the means petal lengths of the two species of iris are significantly different (P <2.2e-16). The petal length of Setosa is significantly shorter than that of Versicolor. We accept hypothesis 2.

Note: The above analyses assume that the variances are unequal, to specify equal variance add var.equal = TRUE to the end of the argument.

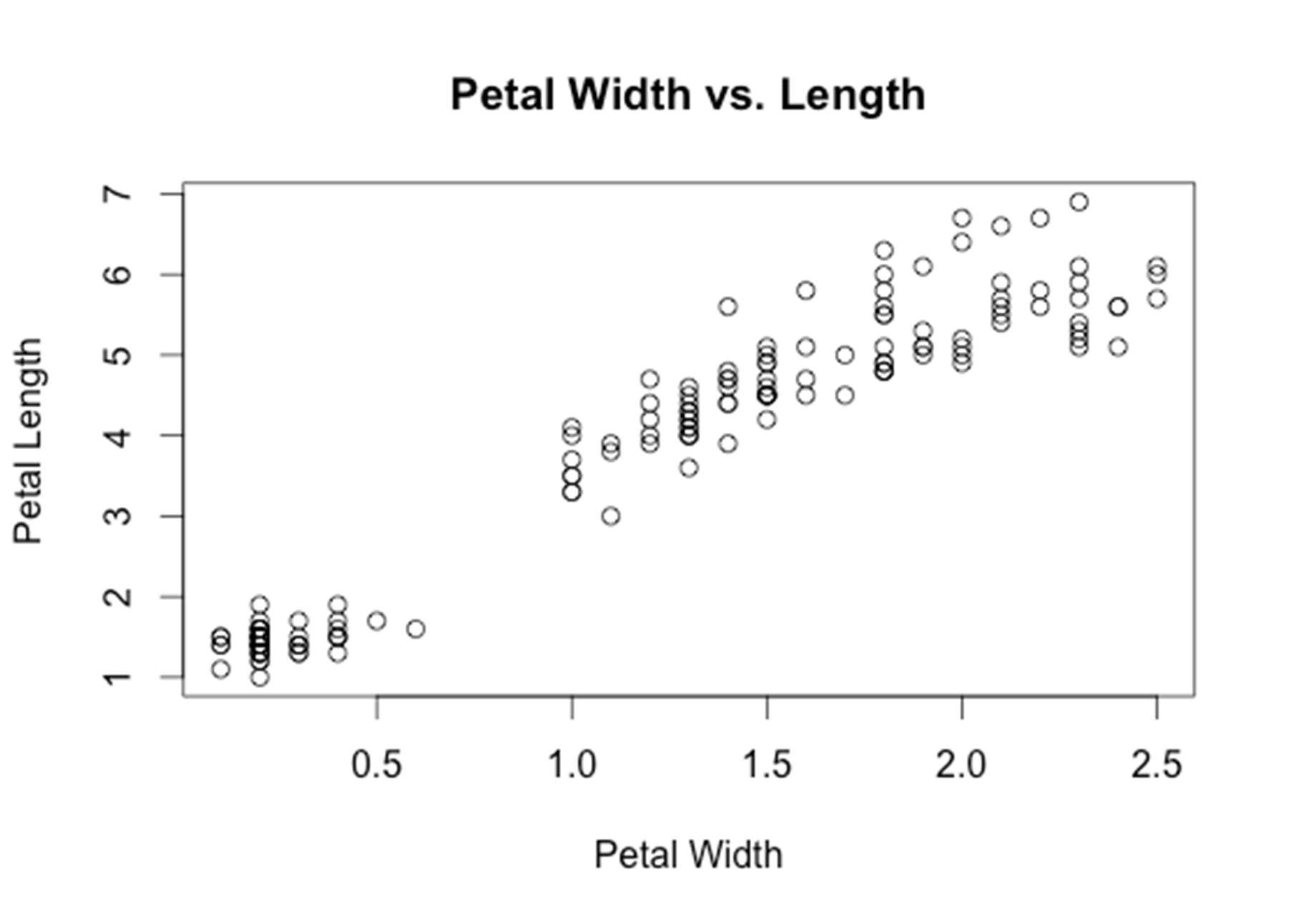
## Correlation

We will use the iris data to look at the relationship between petal length and width.

First, we can look at the relationship on a graph.

>plot(iris$Petal.Width, iris$Petal.Length, main = "Petal Width vs. Length", xlab =

"Petal Width", ylab = "Petal Length")



The plot shows that there is a positive, linear relationship between petal length and width.

If we want to know if the relationship is significant, we can calculate the correlation between the two variables.

> cor.test(iris$Petal.Width, iris$Petal.Length)

Pearson's product-moment correlation

data: iris$Petal.Width and iris$Petal.Length t = 43.3872, df = 148, p-value < 2.2e-16

alternative hypothesis: true correlation is not equal to 0 95 percent confidence interval: 0.9490525 0.9729853 sample estimates:

cor

0.9628654

Since the p-value is less than 0.05 there is a significant correlation between these two variables. The variables are 96% correlated.

## Linear Regression (LR)

Goal of LR: Model linear relationship between an independent (X) and a dependent (Y) variable.

If we know the petal width can we estimate the petal length?

The simple linear regression model is:

= + +

~ 0, ),

First, fit a linear model.

> model <- lm(iris$Petal.Length ~ iris$Petal.Width)

Summarize the model.

> summary(model)

Call:

lm(formula = iris$Petal.Length ~ iris$Petal.Width)

Residuals:

Min 1Q Median 3Q Max

-1.33542 -0.30347 -0.02955 0.25776 1.39453

Coefficients:

Estimate Std. Error t value Pr(>|t|) (Intercept) 1.08356 0.07297 14.85 <2e-16 \*\*\* iris$Petal.Width 2.22994 0.05140 43.39 <2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4782 on 148 degrees of freedom

Multiple R-squared: 0.9271, Adjusted R-squared: 0.9266

F-statistic: 1882 on 1 and 148 DF, p-value: < 2.2e-16

The results show us:

The intercept ( ) = 1.08356

The slope ( ) = 2.22994

The F statistic = 1882

P-value = <2.2e-16

R-squared = 0.9271

So, our predictive equation is:

Petal Length = 1.08356 + 2.22994(Petal Width)

These results tell us that:

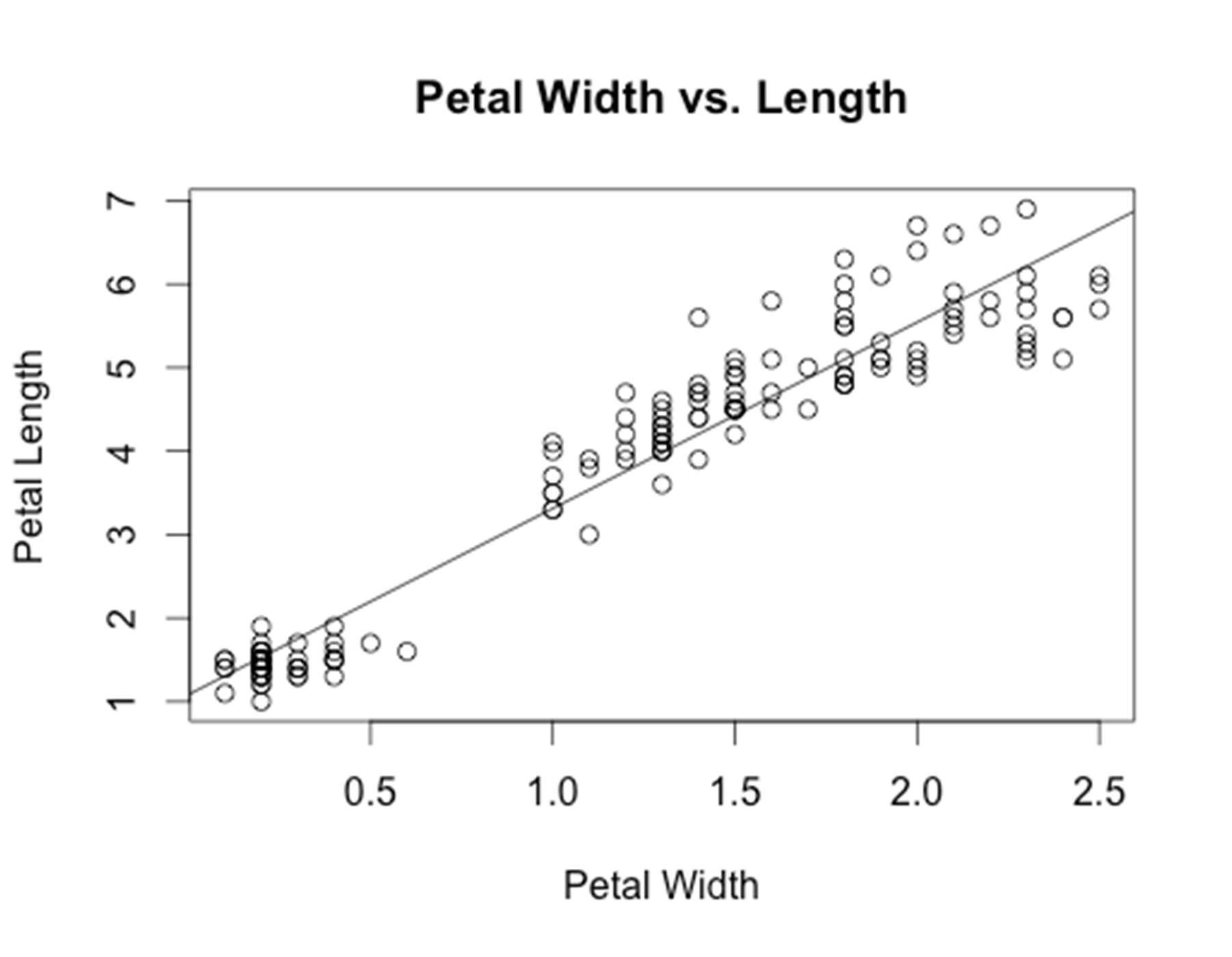
* the intercept is significantly different than 0 (P < 0.05);
* the slope is significantly different than 0 (P < 0.05)
* the model is a good fit for the data (R2 = 0.9271). This means that 93% of the variation in petal length is explained by the petal width.

We can also add a linear regression line directly in the plot of the two variables:

> plot(iris$Petal.Width, iris$Petal.Length, main = "Petal Width vs. Length", xlab =

"Petal Width", ylab = "Petal Length")

> abline(model)



# Basics in R – Assignment Answers

1. Open up R Studio and set your working directory to the folder you will be working from (the one that contains the data files you want to use).

In the window on the lower right corner – under Files – find the folder in the window – and then under More > Set As Working Directory

1. Save your R script into this folder. Include comments throughout your script to explain what each command is doing.

> save.image(“BasicsAssignments.R”)

or

File > Save As

1. What is one way that you can find help if you forget how to use the subset function?

> help(subset)

or

> ?subset

1. Using R as a calculator, what is the sum of 15 and 32?

> 15+32

[1] 47

5) Using R as a calculator, what is the product of 34 and 543?

> 34\*543

[1] 18462

6) Using R as a calculator, what is the value of 24?

> 2^4

[1] 16

7) Create a vector of the numbers 3,8,4,2 and give it the name A.

> A <- c(3,8,4,2)

> A

[1] 3 8 4 2

8) Create a vector of the characters monkey, rat, snake, and give it the name B.

> B <- c("monkey", "rat", "snake")

> B

[1] "monkey" "rat" "snake"

9) Create a vector of the sequence of numbers between 89 and 143, and give it the name C.

> C <- seq(89,143)

> C

[1] 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103

[16] 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118

[31] 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133

[46] 134 135 136 137 138 139 140 141 142 143

10) Create a vector containing 15 of the number 3 – use the rep command, and give it the name D.

> D <- rep(3,15)

> D

[1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

11) What is the 3rd element in the vector A?

> A[3]

[1] 4

12) What are the last two elements of the vector B?

> B[2:3]

[1] "rat" "snake"

13) What is the 4th and 30th element in the vector C?

> C[c(4,30)]

[1] 92 118

14) What is the sum of all of the elements in the vector D?

> sum(D)

[1] 45

15) Add the character “bird” to the end of vector B.

> B[4] <- "bird"

> B

[1] "monkey" "rat" "snake" "bird"

16) Change the first element in vector D to the number 5.

> D[1] <- 5

> D

[1] 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3

17) Multiple vector A by 3 and save the results as vector A2.

> A2 <- A\*3

> A2

[1] 9 24 12 6

18) Add 5 to every element of vector C and save the results as vector C2.

> C2 <- C+5

> C2

[1] 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108

[16] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123

[31] 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138

[46] 139 140 141 142 143 144 145 146 147 148

1. Import the ‘lionCrater’ file into R and save the dataframe as ‘Lion”. This file gives the population numbers for lions in the Ngorongoro Crater in Tanzania.

> Lion <- read.csv("lionCrater.csv", header = T)

1. Import the ‘blackRhinoCrater’ file into R and save the dataframe as “Rhino”. This file gives the population numbers for black rhinos in the Ngorongoro Crater in Tanzania.

> Rhino <- read.csv("blackRhinoCrater.csv", header = T)

1. Display the structure of the Lion data frame. What is included? How many observations and how many variables? What are the variable names?

> str(Lion)

'data.frame': 30 obs. of 2 variables:

$ Year : int 1963 1964 1965 1966 1967 1968 1968 1970 1971 1972 ...

$ Population: int 13 12 31 25 39 46 60 61 53 69 ...

There are 30 observations in the data frame of 2 variables. The variables are year and population.

1. What are three ways to display the data in column 1 of the Lion data frame?

> Lion[1]

> Lion["Year"]

> Lion$Year

1. How do you access the 5th row of data in the Lion data frame?

> Lion[5,]

Year Population

5 1967 39

24) Display the first 6 rows of the Rhino data frame.

> head(Rhino)

Year Population

1. 1980 20
2. 1981 21
3. 1982 20
4. 1983 15
5. 1984 10
6. 1985 11

25) Display the last 6 rows of the Rhino data frame.

> tail(Rhino)

Year Population

1. 1990 11
2. 1991 12
3. 1992 14
4. 1993 15
5. 1994 16
6. 1995 16

26) How many rows are in the Rhino data frame?

> nrow(Rhino)

[1] 16

27) How many columns are in the Rhino data frame?

> ncol(Rhino)

[1] 2

28) Look at the Year column for both Lions and Rhinos. Which years overlap? Create two new data frames Lion2 and Rhino2 that only contain the overlapping years.

1980-1992 are the overlapping years.

> Lion2 <- subset(Lion, Lion$Year > 1979)

> Rhino2 <- subset(Rhino, Rhino$Year < 1993)

29) Create a new data frame called Lion3 that only contains the years when the population is over 50 individuals. How many rows are in this new data frame?

> Lion3 <- subset(Lion, Lion$Population > 50)

> nrow(Lion3)

[1] 23

30) Create a new data frame called Rhino3 that only contains the years when the population of rhinos is less than 12 individuals. How many rows are in this new data frame?

> Rhino3 <- subset(Rhino, Rhino$Population < 12)

> nrow(Rhino3)

[1] 6

# Descriptive Statistics/Plotting – Assignment Answers

1. Import the Lion and Black Rhino files into R that we were using in the last assignment (‘lionCrater.csv’ and ‘blackRhinoCrater.csv’).

> Lion <- read.csv("lionCrater.csv", header = T)

> Rhino <- read.csv("blackRhinoCrater.csv", header = T)

1. What is the minimum and maximum population size for lions?

> min(Lion$Population)

[1] 12

> max(Lion$Population)

[1] 123

3) What is the mean, median, and standard deviation of the population size of lions?

> mean(Lion$Population)

[1] 69.2

> median(Lion$Population)

[1] 72.5

> sd(Lion$Population)

[1] 27.80523

4) What is the minimum and maximum population size for black rhinos?

> min(Rhino$Population)

[1] 10

> max(Rhino$Population)

[1] 21

5) What are the quantile (0%,25%,50%,75%,100%) population sizes for black rhinos?

> quantile(Rhino$Population)

0% 25% 50% 75% 100%

1. 11 14 16 21

6) Use the summary command to find the descriptive statistics of population size for black rhinos. What is the mean and median?

> summary(Rhino$Population)

Min. 1st Qu. Median Mean 3rd Qu. Max.

10.00 11.00 14.00 14.12 16.00 21.00

The mean is 14.12 rhinos and the median is 14 rhinos.

7) What is the variance and standard deviation of population size for black rhinos?

> var(Rhino$Population)

[1] 13.98333

> sd(Rhino$Population)

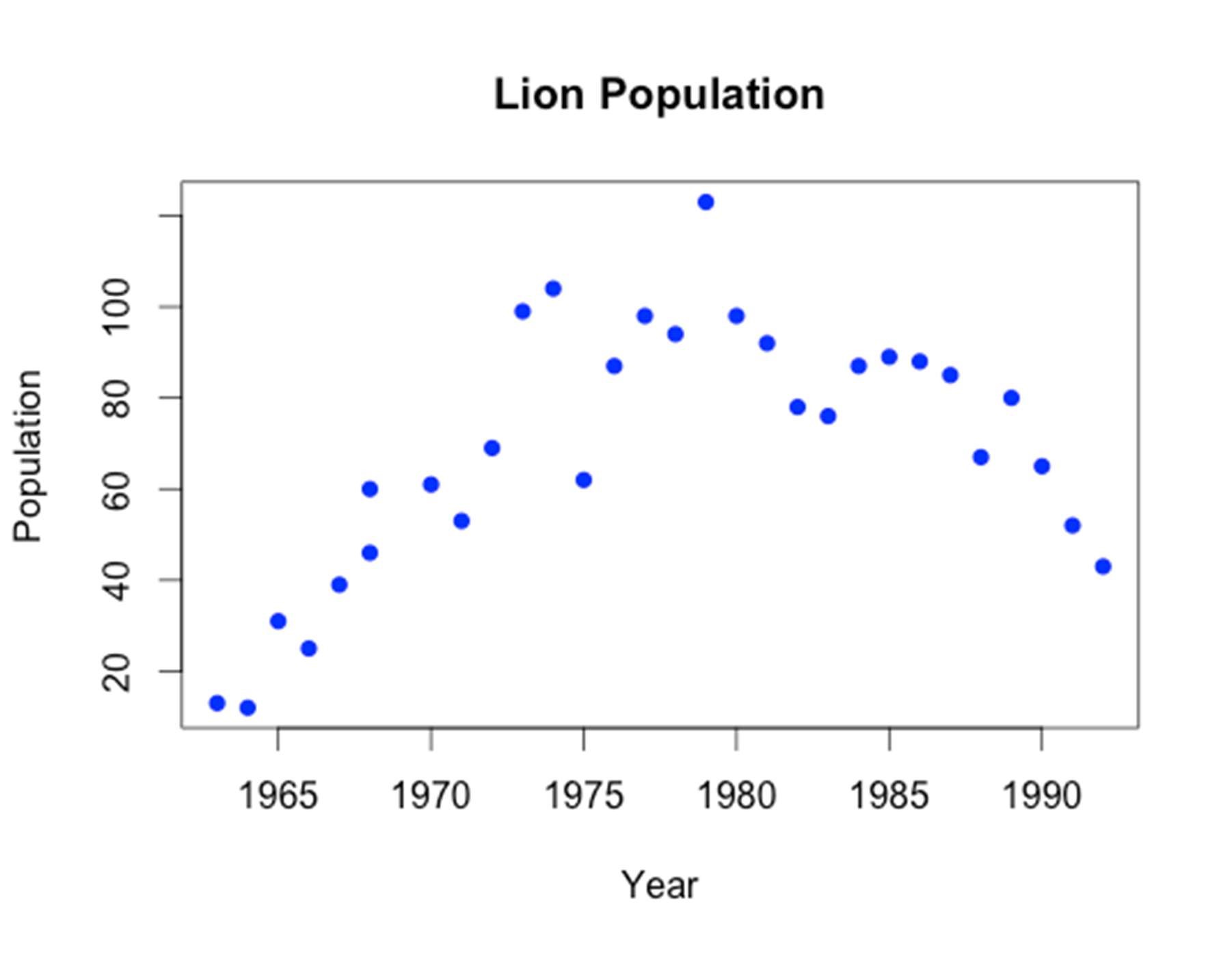
[1] 3.73943

1. Create a plot of the lion population size with the year on the x-axis. Include a title, axis labels, and make the points blue. Can you figure out how to change the open circle points to another shape? Try and change them to closed circles. Is the population increasing or decreasing? What year is the minimum and maximum population sizes?

> plot(Lion$Year, Lion$Population, main = "Lion Population", xlab = "Year", ylab =

"Population", col = "blue", pch = 16)

The population is increasing until 1979 and then decreasing. The minimum value is at 1964 and the maximum value is at 1979.

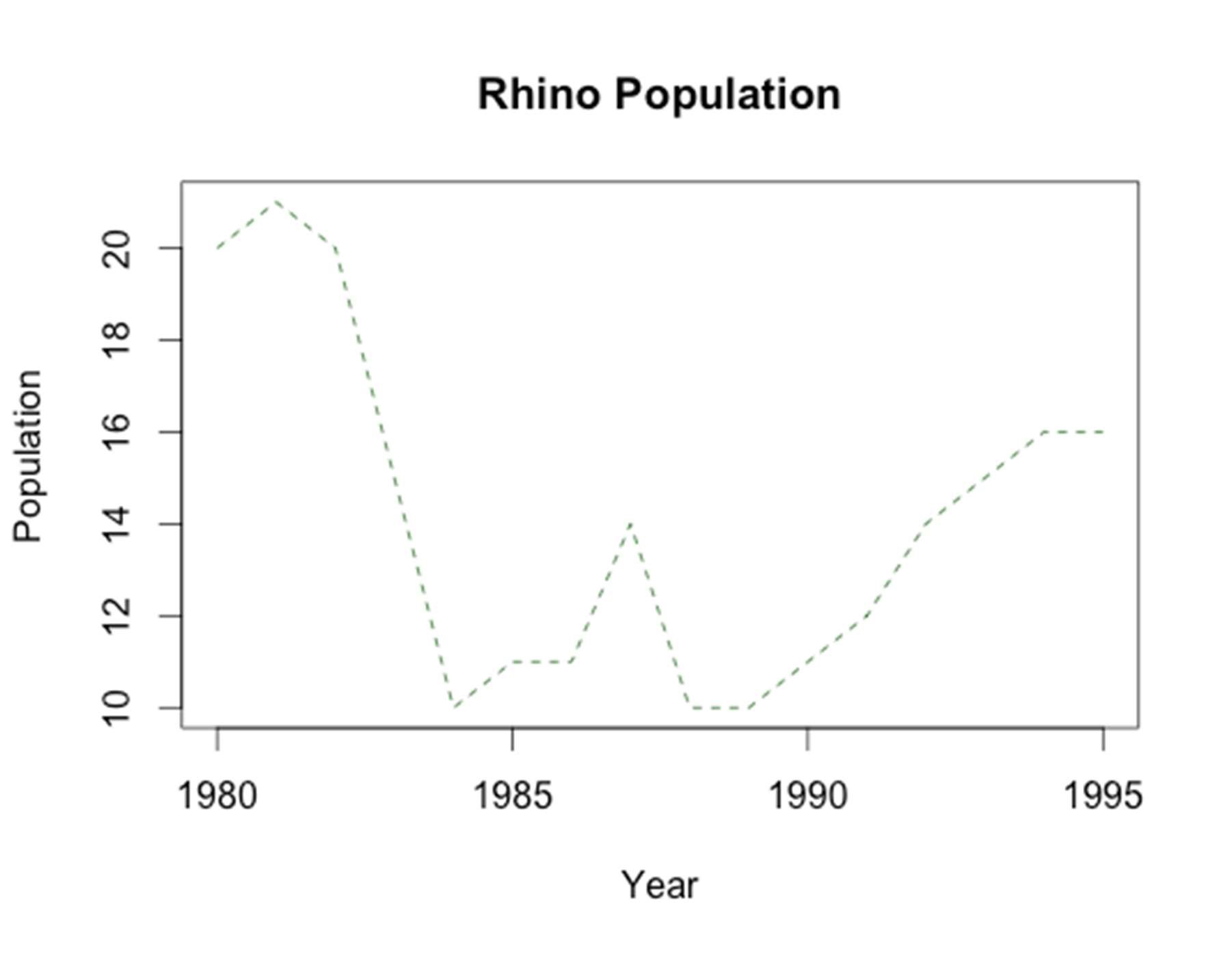


1. Create a plot of the black rhino population size with the year on the x-axis. This time make it a line plot. Include a title, axis labels, and the make the line dark green. Can you figure out how to change the line type to a dotted line? Is the population increasing or decreasing? What year is the minimum and maximum population sizes?

> plot(Rhino$Year, Rhino$Population, main = "Rhino Population", xlab = "Year", ylab =

"Population", col = "darkgreen", type = "l", lty = 2)

The population increases to 1981 then decreases to 1984 then increases to 1987 then decreases to 1988 and then increases to 1995. The minimum population size is at year 1984, 1988, and 1989, and the maximum population size is at 1981.

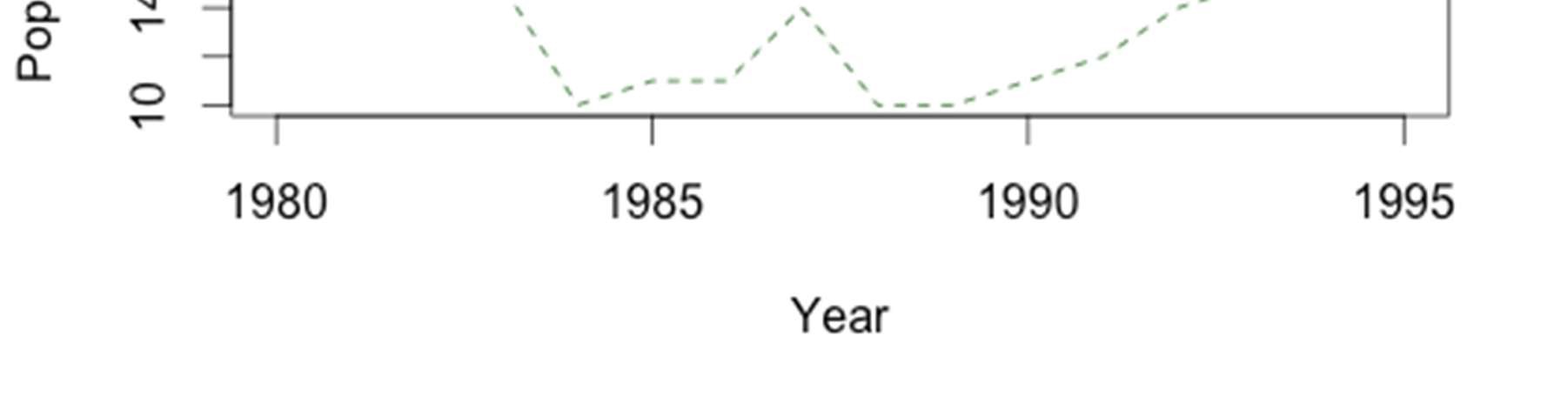
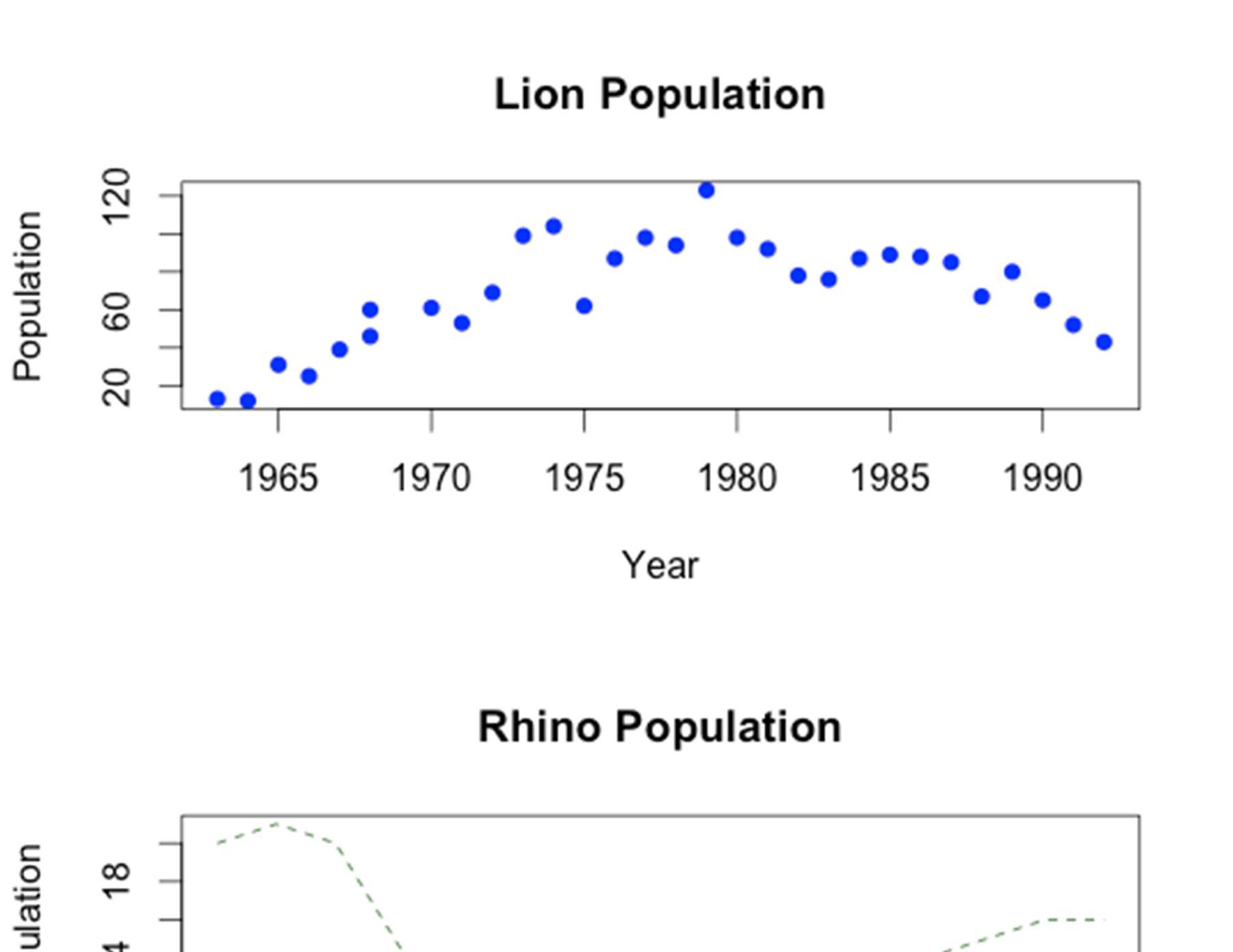


1. Put the two plots that you just made onto one page with one column and two rows.

> par(mfrow = c(2,1))

> plot(Lion$Year, Lion$Population, main = "Lion Population", xlab = "Year", ylab = "Population", col = "blue", pch = 16)

> plot(Rhino$Year, Rhino$Population, main = "Rhino Population", xlab = "Year", ylab = "Population", col = "darkgreen", type = "l", lty = 2)



1. Create two new data frames Lion2 and Rhino2 that contain the subset of years that overlap between the two datasets (like we did in the last assignment). These new datasets will be used for the next plot.

> Lion2 <- subset(Lion, Lion$Year >1979)

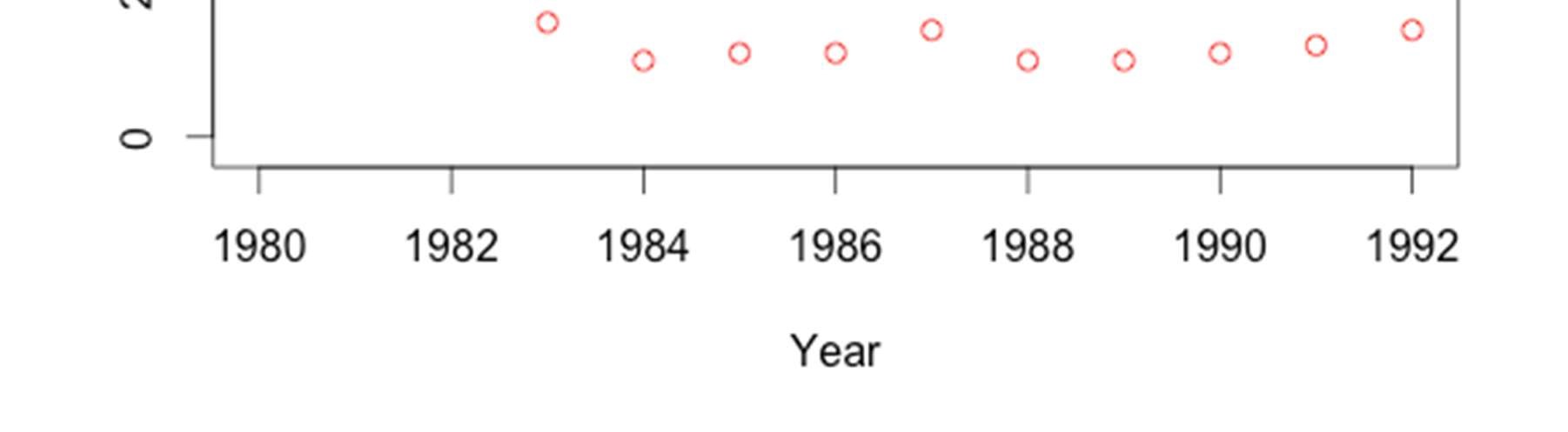
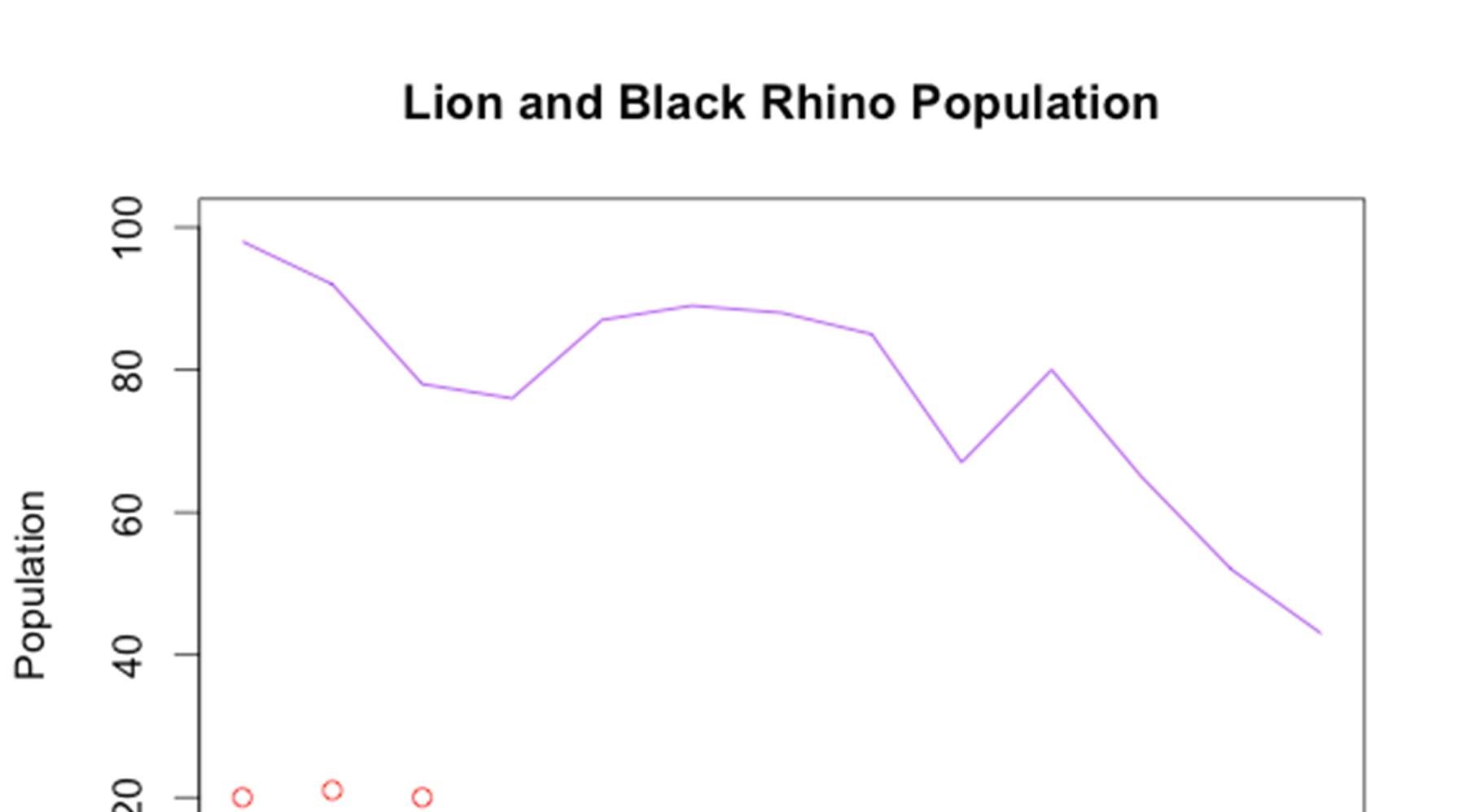
> Rhino2 <- subset(Rhino, Rhino$Year < 1993)

1. Create a line plot of the lion population sizes. On the same plot add the black rhino population sizes as points. Make the lion population line purple and the black rhino population points red. Includes a title and axis labels. You will need to set the limits of the y-axis from (0,100) otherwise the rhino points will not be visible when you add them to the plot.

> plot(Lion2$Year, Lion2$Population, type = "l", col = "purple", main = "Lion and Black

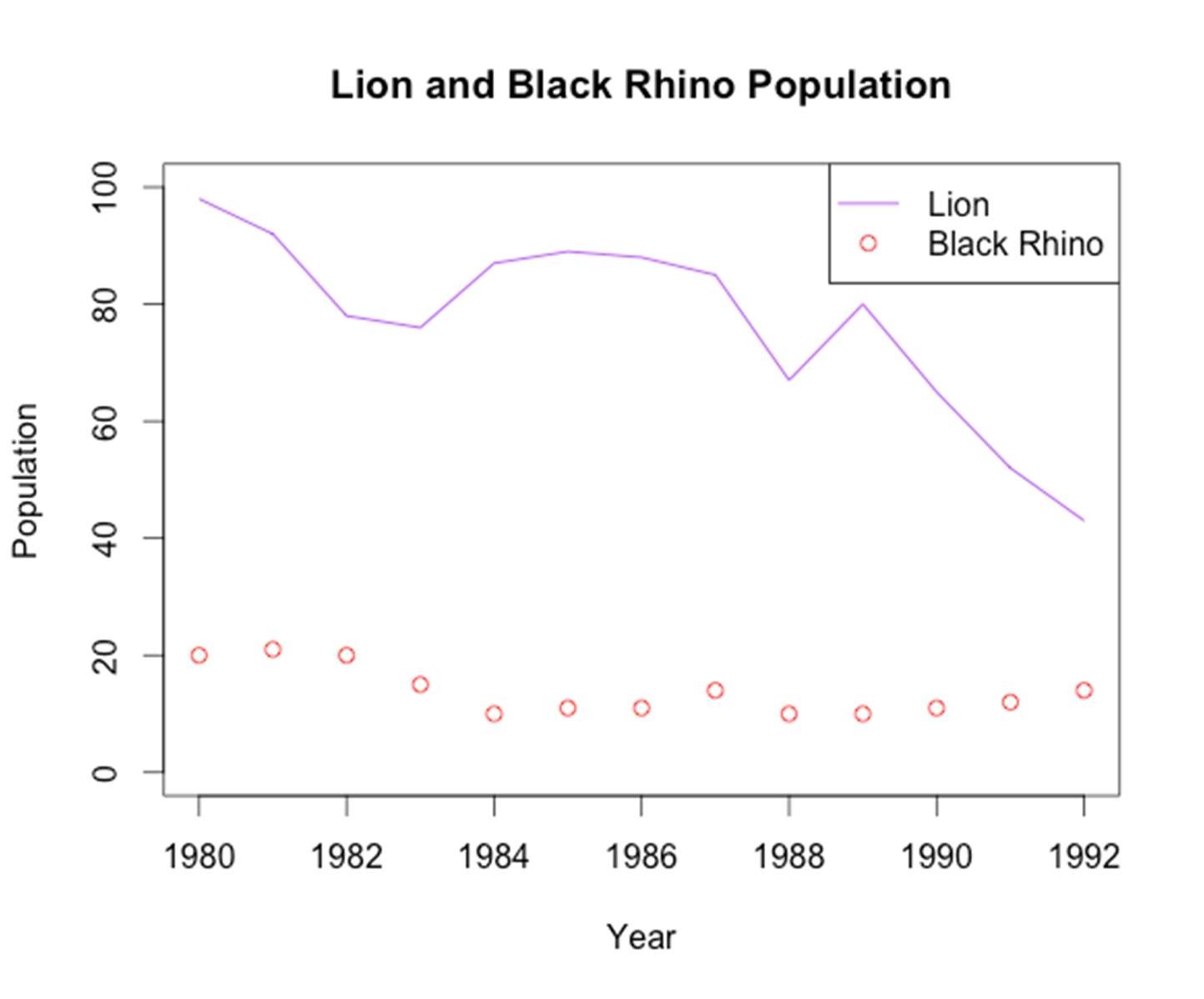
Rhino Population", xlab = "Year", ylab = "Population", ylim = c(0,100))

> points(Rhino2$Year, Rhino2$Population, col = "red")



1. Add a legend to the top right of the plot you just made – make sure next to lion you have a purple line, and next to black rhino you have a red point.

> legend("topright", legend = c("Lion", "Black Rhino"), col = c("purple", "red"), lty = c(1,NA), pch = c(NA,1))

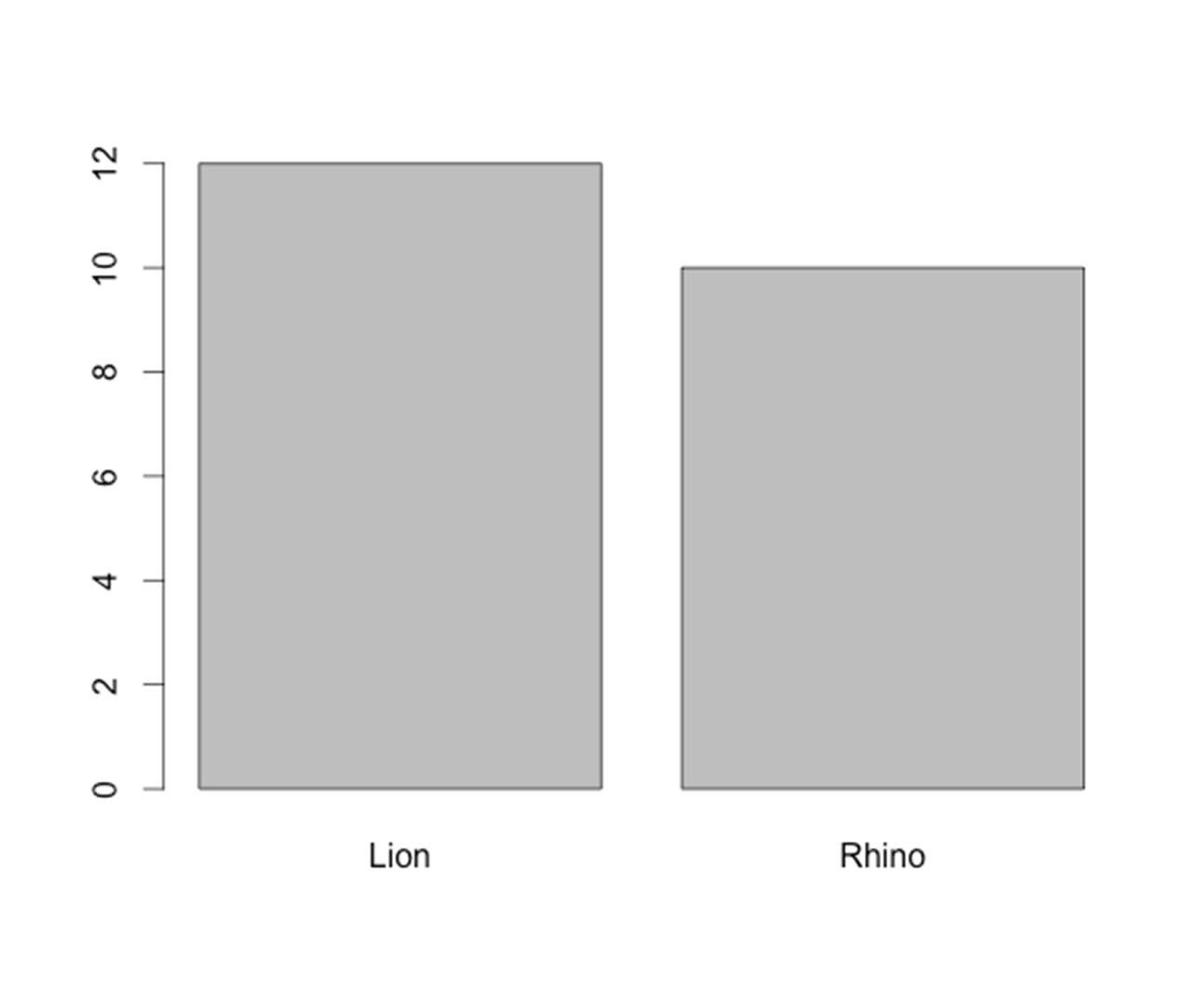


1. Create a barplot showing the minimum population size for black rhino and lion. Make sure to include a title and labels for each of the bars.

> data <- c(min(Lion$Population), min(Rhino$Population))

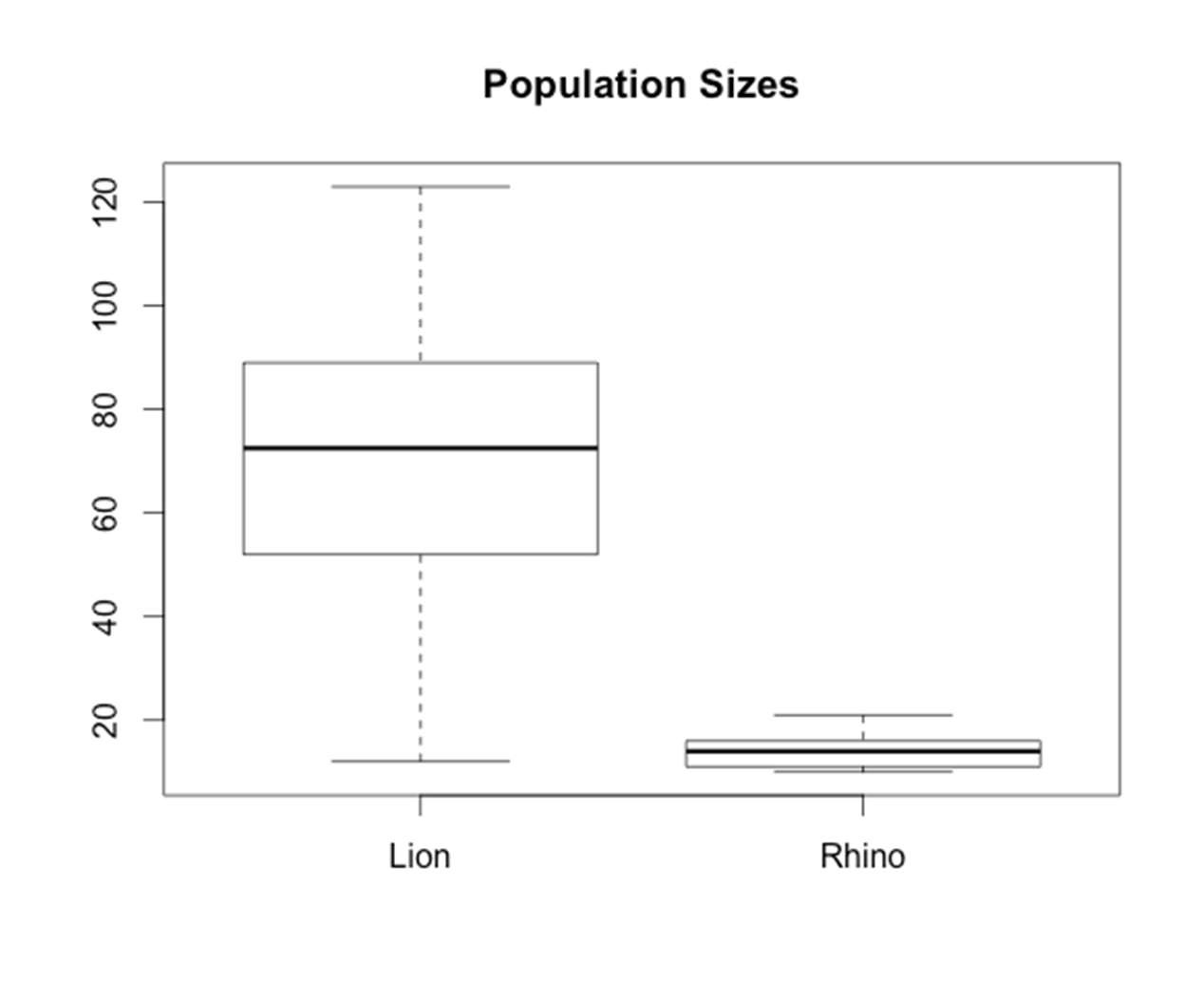
> names(data) <- c("Lion", "Rhino")

> barplot(data)



1. Create a side-by-side boxplot of the population size for lions and black rhinos – make sure to include a title, and a label below each box to show which animal it represents.

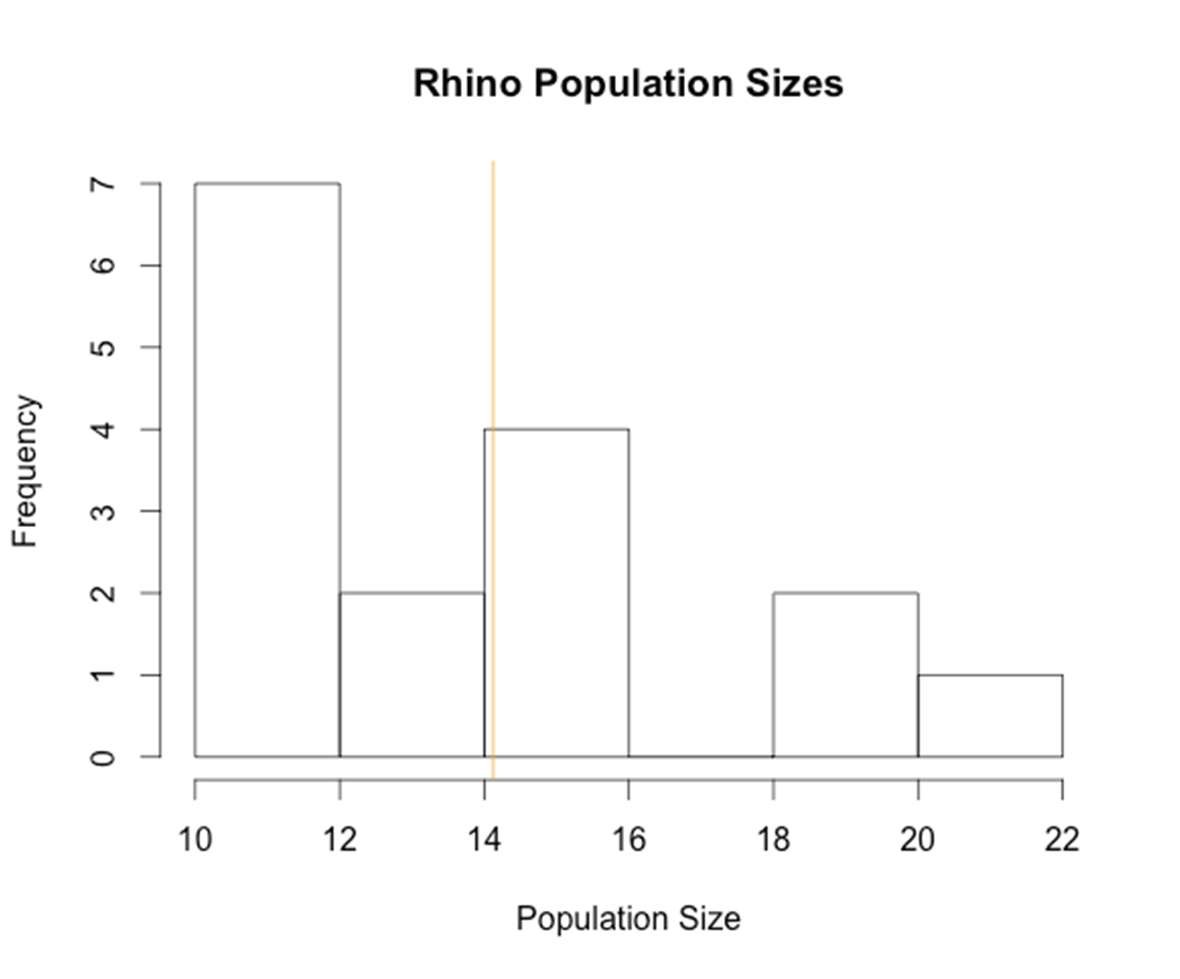
> boxplot(Lion$Population, Rhino$Population, names = c("Lion", "Rhino"), main = "Population Sizes")



1. Create a histogram of the population size for black rhinos. Add an orange vertical line to show the mean population size.

> hist(Rhino$Population, main = "Rhino Population Sizes", xlab = "Population Size")

> abline(v=mean(Rhino$Population), col = "orange")



Data Sources:

Brooks, B., Traill, L. W., and Bradshaw, C. J. A. 2006. Minimum viable population sizes and global extinction risk are unrelated. Ecology Letters 9 (4): 375-382. White Rhinocerous (Ceratotherium simum) population numbers from Matobo National Park, Zimbabwe from 1962-1994.

Lion (Panthera leo) population numbers from Ngorongoro Crater, Tanzania from 1963-1992.

Black Rhino (Diceros bicornis) population numbers from Ngorongoro Crater, Tranzania from 1980-1995.

Dublin, H. T., Sinclair, A. R. E., Boutin, S., Anderson, E., Jago, M., and Arcese, P. 1990. Does competition regulate ungulate populations? Further evidence from Serengeti, Tanzania. Oecologia 82: 283-288.

Blue Wildebeest (Connochaetes taurinus) population numbers in the Serengeti Plains from 1961-1986.

African Buffalo (Syncerus caffer) population numbers in the Serengeti Plains from 1965-1976.

Hanby, J. P., Boycott, J. D. and Packer, C. 1995. Ecology, demography, and behavior of lions in two contasting habitat: Ngorongoro Crater and the Serengeti Plains. In:

Serengeti II by Sinclair, A. R. E. and Arcese, P. pp 315-331.

Lion (Panthera leo) population numbers in the Serengeti Plains from 19661990.

Teleki, G., Hunt Jr., E. E. and Pfifferling, J. H. 1976. Demographic Observations (19631973) on the Chimpanzees of Gombe National Park, Tanzania.

Chimpanzee (Pan troglodytes) population numbers in Gombe National Park, Tanzania from 1964-1973.