R the ESSENTIALS

For those of you who are new to R. Here are some basics to get you started.

The '>' is your prompt to type something in the console to make things happen. Try this:

This creates a data object in R, known as a vector with 5 items, with values 1, 2, 3, and so on.

> x = c(1,2,3,4,5)

Here's another data item:

> y = seq(1, 100)

and another - a vector of 1000 random numbers between 0 and 1:

> z = runif(1000)

By typing the name of any data object, you can inspect its contents:

> x

[1] 1 2 3 4 5

Try examining the others. You'll see that y is a sequence of values from 1 to 100, while z is a collection of 1000 random numbers between 0 and 1.

The power of R lies in its ability to statistically manipulate such collections of data. Using the examples above where you have created some data now try these:

> mean(y)

[1] 50.5

> hist(z)

> boxplot(z)

Now create a vector of 1000 normally distributed random numbers:

> zz = rnorm(1000)

and try examining histograms and boxplots.

R recognizes both x = seq(1,100), and x <- seq(1,100) as equivalent ways to assign values to objects. As a beginner, you're likely to feel more comfortable with the '=' sign, but you should be aware of the other assignment operator as you will likely see it in the help materials. To access the help type (for example) ?hist. For more information go to: **http://cran.r-project.org/doc/manuals/R-intro.pdf**

Adapted from Blanford (2024) Geographic information, geospatial technologies and spatial data science for health. Pp376. CRC Taylor & Francis.

Set working directory

Get the working directory

>getwd()

Set the working directory

>setwd("c://documents//lessondir//")

Working with data files

List Files	> list.files() to show hidden files > list.files(all.files=TRUE)
Reading in files	There are many different things you can do here. Here are some of the ways you can do this: > newfile <- read.table("filename",header=TRUE, sep=",") where sep= can be set to ":", "," or any other special character that might be used. If the file doesn't have a header then leave out the header or set to FALSE. OR > newfile <- read.csv("filename",header=TRUE)
Writing files and saving outputs	We are not really going to be writing any files this week but if you do need to write to a file you may be interested in the following: > outfile <-write.csv(x, file="filename", row.names=FALSE) > outfile <-write.table(x, file="filename")
Saving plots and graphs	To write your plots to a file. For a list of available functions use help(Devices) > savePlot(filename="filename.*ext*", type=*"type"*) where type = png or jpeg. OR > png("plot.png",width=x,height=y) OR Specify the height and width of the plot file you would like to create. > png("plot.png",width=650,height=400) > plot(x,y,main="Scatterplot Example") > dev.off()
Converting files	Convert DBF to CSV file library("foreign") data = read.dbf("path/to/file.dbf") write.csv(data, "path/to/file.csv", row.names=F)

Statistics and Statistical tests

There are a variety of tests available. Here we will provide a summary to get you started. Again some good online resources for learning about different statistical tests include:

https://www.statmethods.net/stats/index.html

https://www.statmethods.net/advstats/index.html

Statistic	**R code**	**Additional information**
Descriptive Statistics	> **summary(x)** > **sum(x)** > **mean(x)** > **median(x)** where x is a column heading in the file newfile Measures of variability > **sd(x)** **>IQR(x)**	Median absolution deviation from median. This measure is less sensitive to outliers than the variance and standard deviation **>mad(x)** In package matrixStats > **weighted.mean(x,w=x2)** > **weightedMedian(x,w=x2)** Trimmed mean drops outliers at either end of the data. In the example below where 10% is dropped from each end (lower end and higher end) > **mean(x, trim=0.1)**
Converting data to z-score	> **(y-mean(x)) / sd(x)**	
Testing the mean of a sample: t-test	> **t.test(x, mu=m)** where mu = mean of the population **Confindence interval for the mean** > **t.test(x, conf.level=0.95)**	p<0.05 indicates the population is not likely to be m; p>0.05 indicates no evidence. if the sample size is small (n<30) then the population must be normally distributed in order to derive meaningful results from the t-test.
Confidence interval for a median	> **wilcox.test(x, conf.int=TRUE)**	
Testing a sample proportion	> **prop.test(x,n,p)**	p<0.05 indicates the proportion is not likely to be p p>0.05 indicates no evidence.
Comparing the means of two samples	> **t.test(x,y)** if paired > **t.test(x,y, paired=TRUE)**	p<0.05 indicates the means are likely different; p>0.05 indicates no evidence. n < 20, the data must be normally distributed; if two populations have the same variance, specify var.equal=TRUE to obtain a less conservative test.

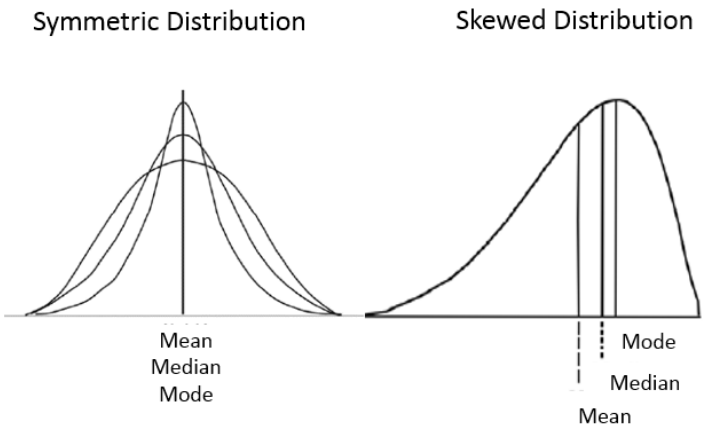
Nonparametric test for comparing two samples	**> wilcox.test(x,y)** if paired **> wilcox.test(x,y, paired=TRUE)**	compare two populations but don't know the distribution but the shapes are similar
Correlation between two variables	Pearson's method is used for normally distributed populations **> cor.test(x,y)** The Spearman's is used method for non-normal populations **> cor.test(x,y, method="Spearman")**	$p<0.05$ indicates the correlations are likely significant; $p>0.05$ indicates correlations are not significant
Regression	Linear regression **> lm(y ~ x, data=dataframe)** **glm**	To store results for different regression models set the outputs to an object **> lm1 <- lm(y ~ x, data=dataframe)** to recall the results and formula and see the residual plots and fit **> plot(lm1)** **> summary(lm1)** Non-normal data and specify the distribution type (e.g. poisson, binomial)

Adapted from Blanford (2024) Geographic information, geospatial technologies and spatial data science for health. Pp376. CRC Taylor & Francis.

Testing for Normality

Many statistical tests require for the data to be normally distributed. Below is an illustration of how the mean, median and mode are affected when the data is not symmetrical.

Figure 4.24 Illustration of a skewed and symmetric distribution and how the mean, median and mode are affected.



To test whether the data is normally distributed you should run a statistical test as well as visualize the data.

Statistical test	**Visualizing the data distribution**
>**shapiro.test(x)** p<0.05 indicates the population is not likely normally distributed p>0.05 indicates no evidence so the population is likely normally distributed.	Histogram >**hist(x)** Boxplot >**boxplot(x)** Q-Q plot (Quantile-Quantile plot) >**qqnorm(x)** >**qqline(x)** If the data were normally distributed the points would fall on the diagonal line. If the data is not normally distributed you may need to transform the data. If there are too many points above the line, the data is skewed to the left. In such cases you may want to transform the data using a logarthmic transformation. To see if a log transformation would make the data normally distributed you can apply the following: >**qqnorm(log(x))** >**qqline(log(x))** If a transformation doesn't prove useful then use a non-parametric test.

Grouping data

I find that I need to group data regularly. In ArcGIS this is easily done using the **Summary Statistics** command in **Analysis- Statistics**. In R there are several different ways you can do this

Creating a group so that you can summarize by groups.	> **split(x,f)** x=variable/header/vector and f=factor/groupToGroupby > **lapply(x,f)** x=variable/header/vector, f=function; returns results in a list > **sapply(x,f)** x=variable/header/vector, f=function; returns results in a vector or matrix *function* can be: length, mean, sum, range, median, sd, t.test
Applying a function to every row or column:	> **apply(matrix,row,function)** where matrix=matrix, row (1=row by row; 2=column by column), function is the function
Applying a function to groups of data	> **tapply (variable/header/dataframe,group,function)**
Applying a function to groups of rows	> **by(variable/header/dataframe,group,function)**

Adapted from Blanford (2024) Geographic information, geospatial technologies and spatial data science for health. Pp376. CRC Taylor & Francis.

Exploring data using visualizations

Visualizations play an important role in exploring data as well as telling a story about the data. There are many types of graphs and plots that you can create as you can see at the following links

Some good resources are:
Text: GGplots2:Guide to Create Beautiful Graphics in R available from
http://www.sthda.com/english/download/3-ebooks/5-guide-to-create-beautiful-graphics-in-r-book/ OR visit http://www.sthda.com/english/wiki/ggplot2-essentials
https://www.statmethods.net/graphs/index.html
https://www.statmethods.net/advgraphs/index.html

https://www.r-graph-gallery.com/index.html

Graphs and plots

To display several graphs or plots together you will need to create a grid to fill.

To divide the plot window into N x M matrix

> par(mfrow=(c(N,M))

After creating a matrix, call plot n times until each quadrant has been filled. For example if you have a 2 x 2 matrix then call plot 4 times to draw the figure in each quadrant. The plots will go from left to right starting at the top and move down.

To fill in the quadrants by column then us mfcol instead of mfrow

> par(mfcol=c(N,M))

We have provided a list of graphs and plots that might be of use. Again refer to the resources provided on the previous page for more details and also on how to pretty these up.

Graph	**> plot(x, main="Title of Graph", xlab="Label X", ylab="Label y")** Plotting a line from x and y points **> plot(x,y, type="1"")** **> plot(dataframe, type="1")** Adding a vertical or horizontal line draw a vertical line at x **> abline(v=x)** draw a horizontal line at y **> abline(h=y)** **> curve(x)**
Scatterplot	**> plot(x,y, main="Title of Graph", xlab="Label X", ylab="Label y")** where x is one variable and y is another variable
Bar Chart	**> barplot2(x)** With confidence intervals **> barplot2(x, plot.ci=TRUE, ci.l=lower, ci.u=upper)** With error bars **> data1 <- read.table("filename", header=T)** **> attach(data1)** **> names(data1)** Calculate the heights of the bars **> means <- tapply(field1, field2, mean)** **> barplot (means,xlab="x label", ylab="y label", col="green")**
Histogram	**> hist(x)** **> hist(x,freq=FALSE)** Density estimate which can be plotted over the histogram **> lines(density(x),lwd=3,col="blue")**
Boxplots	**> boxplot(x)** Boxplot for multiple variables/vectors comparisons **> boxplot(x~f)** **> boxplot (x ~ f, data=dataset, main="Title", xlab="x label", ylab="y label")** where x = numeric valriable and f= the factor

	OR try **> plot(response~factpr(fact), notch=TRUE)** Order the factor levels **> index <- order(tapply(response,fact,mean))** **> ordered <- factor(rep(index,rep(20,8)))** **> boxplot (response~ordered, notch=T, names=as.charcter(index))** **> ggplot(datafile, aes(x = field, y = field)) + geom_boxplot()** **> ggplot(datafile, aes(x = reorder(field, field2, FUN = median), y= field2)) +geom_boxplot()**
Q-Q plot (Quantile-Quantile plot)	**> qqnorm(x)** **> qqline(x)**

Adapted from Blanford (2024) Geographic information, geospatial technologies and spatial data science for health. Pp376. CRC Taylor & Francis.

Additional Resources

Wickham, H. and Grolembund, G. (2017) R for Data Science. O'Reilly
http://r4ds.had.co.nz/index.html

Kipp, A. (2017) Shinyapps.io - Getting started.
https://shiny.rstudio.com/articles/shinyapps.html

Anonymous (2017) The basic parts of a Shiny app.
https://shiny.rstudio.com/articles/basics.html

http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html

https://www.statmethods.net/advstats/index.html