CSCE 587
Introduction to R / RStudio

Part 2

RStudio Environment: Console / History / Script Files

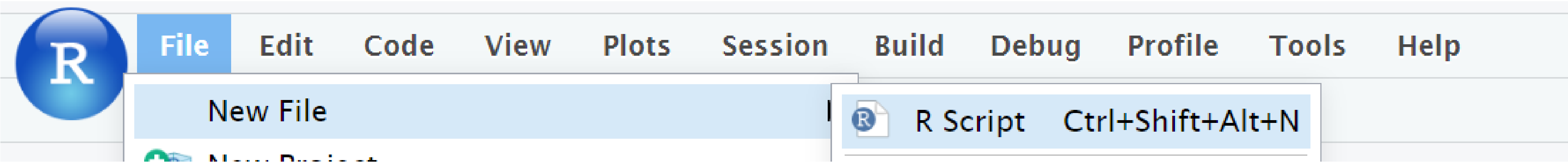
Loading Datasets and Basic Plots

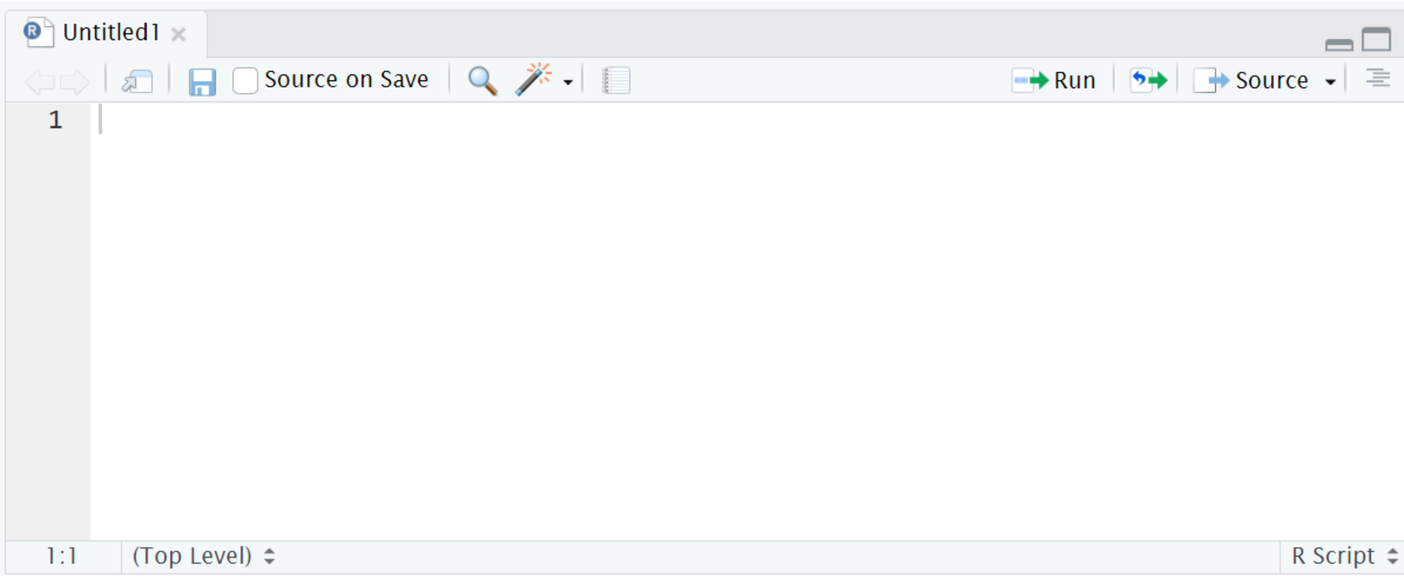
RStudio Environment

- Console – R commands are executed
- Terminal – Your Virtual Machine terminal

RStudio Environment

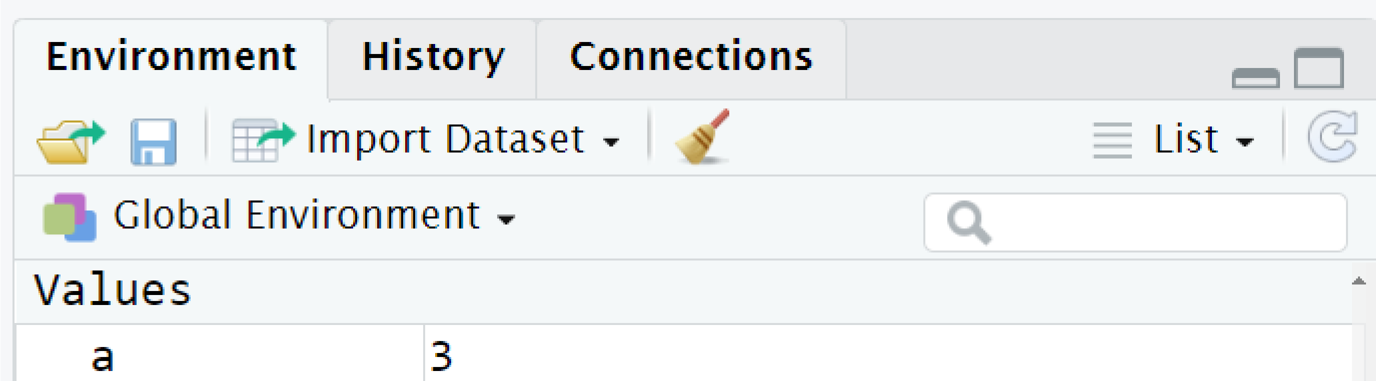
- Editor – We can edit R Script files (.R) to save on our virtual machine for future use



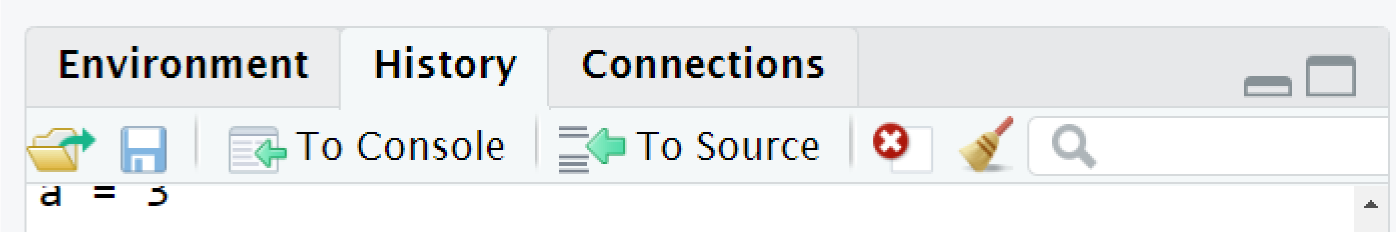


RStudio Environment

- Environment – View objects currently held in the environment

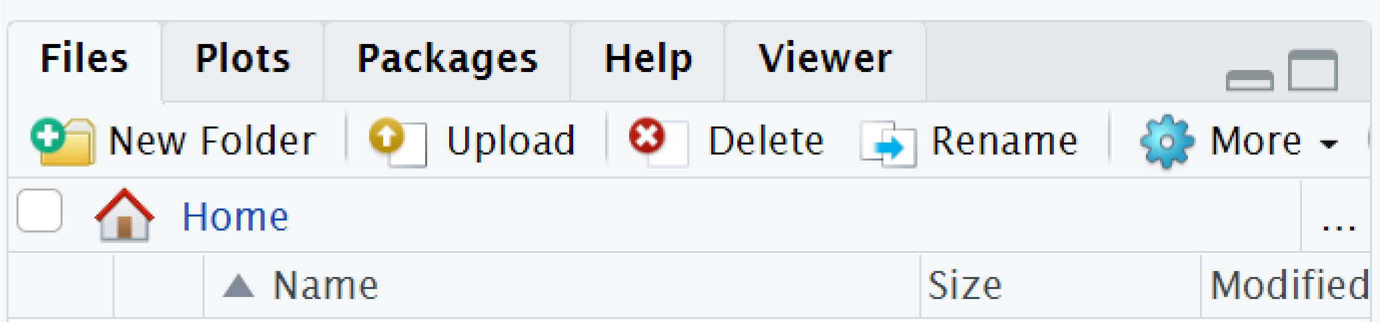


- History – View recently executed commands. Can load, save, execute, save, or remove/clear these commands



RStudio Environment

- Manage Files and Packages, view plots and help



Loading Data Sets

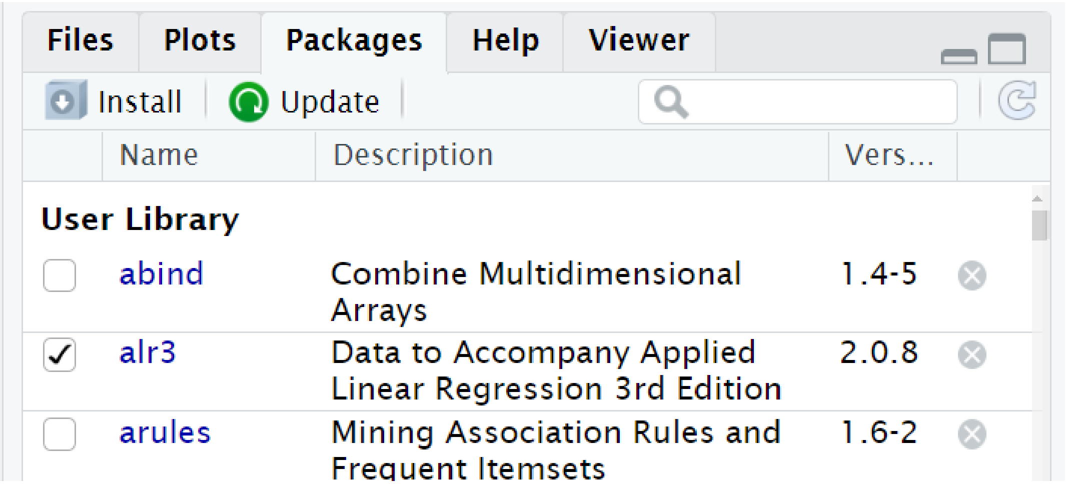
- Several ways to load data sets
  - Data sets in R packages
  - Data sets from the web
  - Data sets on the VM

Loading Data Sets from R Packages

Click on the Packages button

The second package listed is alr3

Click in the box at the left to load the package



Clicking the box in the GUI causes the R command to load the library to be executed at the console.

```
> library("alr3", lib.loc="~/R/x86_64-redhat-linux-gnu-library/3.5")
Loading required package: car
Loading required package: carData
```

Loading Data Sets from R Packages

- We'll use data() to access the UN2 data set in the alr3 package

```
> data(UN2)
> UN2
            logPPgdp logFertility Purban
Afghanistan 6.614710    2.7655347     22
Albania     10.363040   1.1890338     43
Algeria     10.800900   1.4854268     58
Angola      9.529431    2.8479969     35
Argentina   12.806348   1.2868811     88
Armenia     9.424166    0.2016339     67
Australia   14.197524   0.7655347     91
Austria     14.505563   0.3561438     67
```

Working with Data Sets

This data set looks like a matrix with row and column labels

Let’s examine the first row:

```
> UN2[1,]
            logPPgdp logFertility Purban
Afghanistan  6.61471     2.765535     22
```

Note: R shows us the row and column labels in the output

Working with Data Sets

Row and Column labels can be used for subsetting

```
> UN2['Algeria',]
        logPPgdp logFertility Purban
Algeria  10.8009     1.485427     58
```

This is better than hard coding an index since the addition of another row to the dataset won't affect which row our command accesses!

Working with Data Sets

We can calculate the mean for each column using the column names

```
> mean(UN2[,'logPPgdp'])
[1] 10.99309
> mean(UN2[,'logFertility'])
[1] 1.468687
> mean(UN2[,'Purban'])
[1] 55.53886
```

Working with Data Sets

Since all the values are numeric, we can compute the variance-covariance:

> var(UN2)

```
                logPPgdp logFertility    Purban
logPPgdp        5.640839   -1.2475279  44.55587
logFertility   -1.247528    0.6009038 -11.00879
Purban         44.555873  -11.0087939 579.19770
```

Data Sets from the Web

Example URL for a binary data file:

https://cse.sc.edu/~bhipp/587/WavesBasicR.RData

To download to our virtual machine, select the Terminal window in RStudio and enter:

wget https://cse.sc.edu/~bhipp/587/WavesBasicR.RData --no-check-certificate

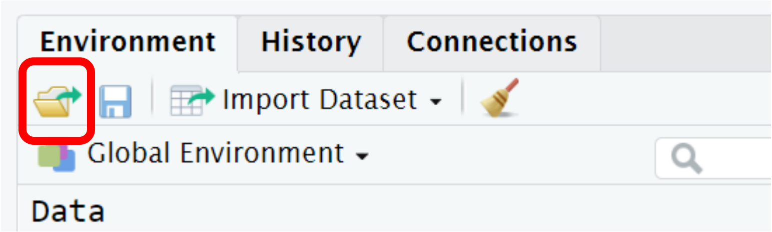
Excerpt from the terminal window showing command and result

```
[student@sandbox-host ~]$ wget https://cse.sc.edu/~bhipp/587/WavesBasicR.RData --no-check-certificate
--2024-08-25 19:29:08--  https://cse.sc.edu/~bhipp/587/WavesBasicR.RData
Resolving cse.sc.edu (cse.sc.edu)... 129.252.138.13
Connecting to cse.sc.edu (cse.sc.edu)|129.252.138.13|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1472151 (1.4M)
Saving to: ‘WavesBasicR.RData’

100%[======================================>] 1,472,151   --.-K/s   in 0.01s

2024-08-25 19:29:08 (102 MB/s) - ‘WavesBasicR.RData’ saved [1472151/1472151]
```

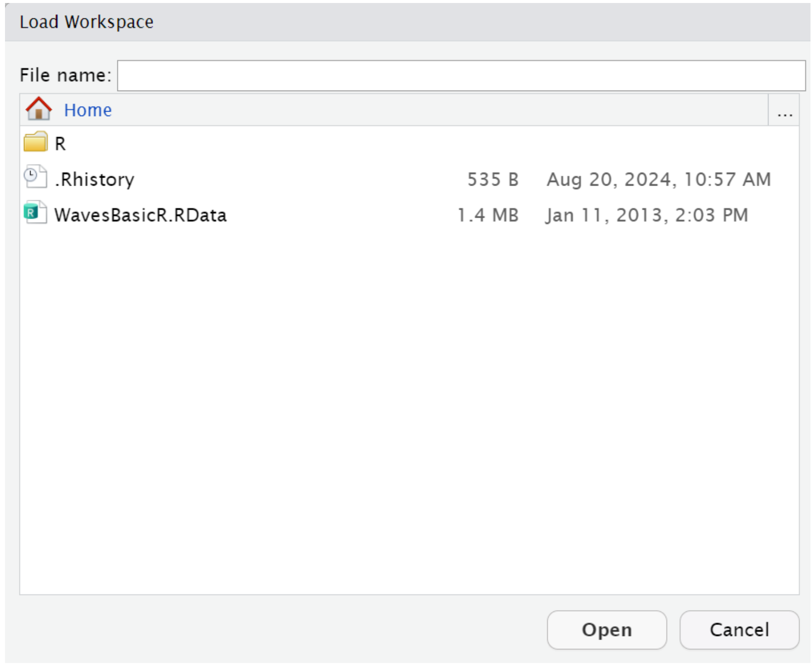
Data Sets from the Web

Now we can load the data set into RStudio

Click on the load workspace icon



This will open a dialog box for us to select the WavesBasicR.Rdata file

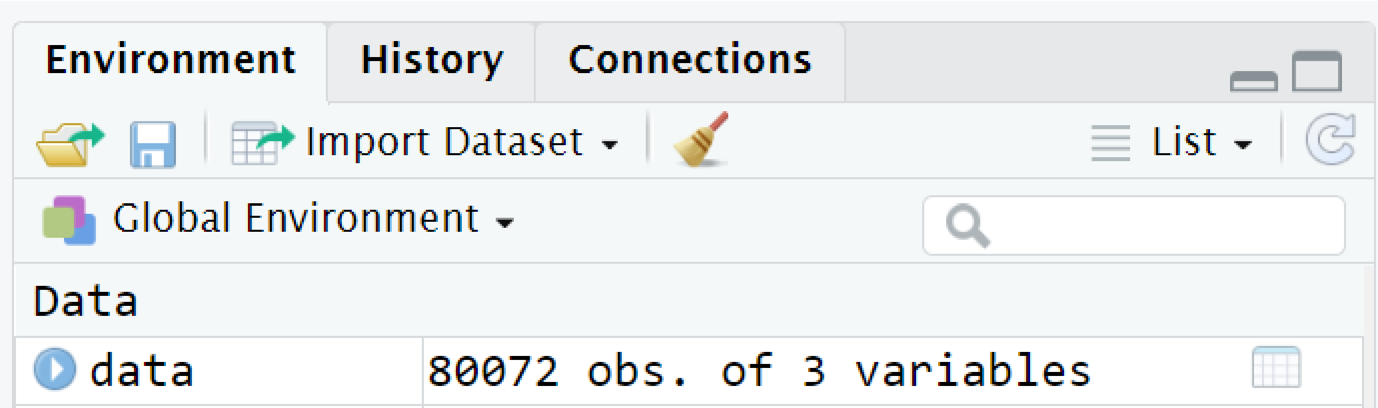


Data Sets from the Web

Notice that opening the file through the GUI causes the load command to be execute in the console window:

> load("~/WavesBasicR.RData")

The data set is a data.frame called "data"



Data Sets from the Web

A data.frame is similar to a matrix, but columns don't have to all be of the same type (though the values within an individual column are all the same type).

The summary function is useful to provide some basic descriptive statistics for numeric fields in a dataframe.

Example:

```
> summary(data)
       y                  x             wave_height    
 Min.   :-66.15   Min.   :-179.998   Min.   : 3.500  
 1st Qu.:-57.96   1st Qu.:-103.803   1st Qu.: 3.917  
 Median :-50.00   Median :  -2.345   Median : 4.462  
 Mean   :-42.71   Mean   :  -4.924   Mean   : 4.841  
 3rd Qu.:-39.50   3rd Qu.:  88.488   3rd Qu.: 5.327  
 Max.   : 66.15   Max.   : 179.999   Max.   :19.986  
```

Exercise (borrowed from Irina Kukuyeva)

Using this data set, how many waves are higher than the mean height?

Let's start by looking at the first few rows of data

```
> data[1:10,]
              y        x wave_height
15   -66.13506 18.82452       5.408
30   -66.09778 20.76402       5.053
45   -66.03464 22.69563       4.663
60   -65.94584 24.61539       4.937
75   -65.83165 26.51951       4.653
90   -65.69244 28.40436       4.765
105  -65.52863 30.26658       4.370
120  -65.34071 32.10304       4.299
135  -65.12923 33.91093       4.066
150  -64.89478 35.68771       3.682
```

Exercise (borrowed from Irina Kukuyeva)

Note: the third column is the wave height

1. Start by calculating the mean (use the column label)

```
> mean(data[,'wave_height'])
[1] 4.841396
```

2. Determine “which” observations are larger than the mean?

Hint: use which() and save the resulting vector

```
> larger = which(data[,'wave_height']>mean(data[,'wave_height']))
```

3. Get length of vector. This is our answer!

```
> length(larger)
[1] 29662
```

Plot – From Base R

Let’s look at the help first

```
> ?plot
```

Plot the waves

```
> plot(data$x, data$y, main=“Coordinates of wave heights")
```

Histograms – From Base R

We can use hist() to plot a histogram

Syntax: hist(data, xlab, main)

```
> hist(data$wave_height, xlab="wave heights", main="Our first histogram")
```

Plotting Histograms

Let's use the ggplot2 package to produce a nicer looking histogram

Load the ggplot2 package (using the library function)

Use the following command:

```
> ggplot(data=data, aes(data$wave_height)) + geom_histogram()
```

Plotting Histograms

Let’s add our own labels and title

```
> ggplot(data=data, aes(data$wave_height)) +
geom_histogram() +
labs(title="Our histogram using ggplot", x="wave height", y="count")
```

Plotting Histograms

Using hist() to plot a histogram can be easier.

```
> hist(data$wave_height, xlab="wave heights", main="Our first histogram")
```

Let’s add a yellow dashed vertical line were the mean is in the histogram

```
> abline(v=mean(data$wave_height), col="yellow", lwd=3, lty=2)
```

Plotting Histograms with ggplot2 package

Let’s update the color and number of bins

```
>  ggplot(data=data, aes(data$wave_height)) +
geom_histogram(bins=30, col="red", fill="blue") +
labs(title="Our ggplot histogram",x="wave height", y="count")
```

Plotting Histograms ggplot2 package

Let’s add a yellow dashed vertical line were the mean is in the histogram

```
> ggplot(data=data, aes(data$wave_height)) +
geom_histogram(bins=30, col="red", fill="blue") +
labs(title="Our ggplot histogram",x="wave height", y="count") +
geom_vline(aes(xintercept=mean(data$wave_height)),
color="yellow", linetype="dashed", size=1)
```

Saving Our Plot

Step 1: Open a pdf file using pdf()

Syntax: pdf(filename, width, height)

Step 2: draw the plot

Step 3: close the file using dev.off()

Saving Our Plot

Step 1: open the file

```
> pdf("myHist.pdf", width=6, height=8)
```

Step 2: Draw the plot

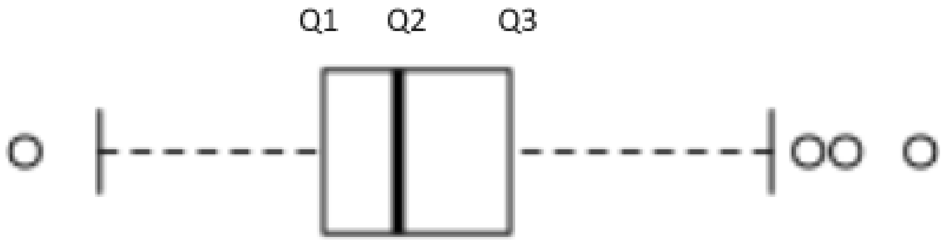
```
>  ggplot(data=data, aes(data$wave_height)) +
geom_histogram(bins=30, col="red", fill="blue") +
labs(title="Our ggplot histogram",x="wave height", y="count") +
geom_vline(aes(xintercept=mean(data$wave_height)),
color="yellow", linetype="dashed", size=1)
```

Step 3: close the file

```
> dev.off()
```

Boxplots

What is represented in the boxplot?



Q1 = First Quartile / 25^{th} Percentile

Q2 = Second Quartile / 50^{th} Percentile / Median

Q3 = Third Quartile / 75^{th} Percentile

IQR = Q3 – Q1 = Interquartile Range

Whiskers extend to smallest and largest values in the dataset within 1.5 *IQR of Q1 and Q3

Outliers are values more than 1.5 * IQR below Q1 or above Q3

Boxplots

Use boxplot() to draw a boxplot:

Syntax: boxplot(dataset, x-label, title)

```
> boxplot(data, xlab= "Variable Names", main= "Boxplot of wave height data")
```

Try y alone

```
> boxplot(data$y, xlab= "Variable Names", main= "Boxplot of wave height data")
```

Exercise: Plots

Let’s plot the wave data again:

```
> plot(data$x, data$y, main="Coordinates of wave heights")
```

Now, lets emphasis larger waves: use a different color:

Step 1: find waves larger than some threshold

Step 2: keep the indices of those “larger” waves

Step 3: add those larger waves using a different color

Exercise: Plots

Step 1: find waves larger than some threshold. How?

```
> data$wave_height > 7
```

Step 2: keep the indices of those "larger" waves

```
> ind = which(data$wave_height > 7)
```

How many?

```
> length(ind)
```

Exercise: Plots

Step 3: Add those larger waves using a different color

Use the points() operation

Syntax: points(x, y, col, symbol)

```
> points(data$x[ind], data$y[ind], col="red", pch=20)
```

Installing a Package

Install the “maps” package by either selecting “Install Packages…” from the Tools menu, or clicking the Install button on the Packages tab

Note: We only have to install a package once. To use the package, we’ll load it with the library function

Exercise: Plots

Add the map to our plot

```
> map("world", add= TRUE)
```

Another Exercise (borrowed from Irina Kukuyeva)

Determine if the northern or southern hemisphere has the largest waves.

Step 1: Choose an appropriate threshold

Step 2: Select those waves larger than the threshold

Step 3: Highlight these observations in a plot

Step 4: Determine which hemisphere has the largest waves.

Another Exercise (borrowed from Irina Kukuyeva)

Step 1: Choose an appropriate threshold. How?

1) Guess? Could take a while.
2) Maybe look at the boxplot?

```
> boxplot(data$wave_height)
```

Another Exercise (borrowed from Irina Kukuyeva)

Step 2: Select those waves larger than the threshold

```
> large = which(data$wave_height > 15)
```

How many waves are larger than 15?

Another Exercise (borrowed from Irina Kukuyeva)

Step 3: Highlight these observations in a plot

```
> points(data$x[large], data$y[large], col="yellow", pch=20)
```

Step 4: Determine which hemisphere has the largest waves.

Looks like a toss up. Try a larger threshold?