# First Steps to R

**W. John Braun, UBC**

Workshop - Calgary

December 4, 2019

# An Overview of R

[1]

**These lectures introduce R, as originally developed as S, by John *Chambers* and others at Bell Laboratories in 1976, and implemented and made into an Open Source program by Robert *Gentleman* and Ross *Ihaka* in 1995.**

**As you learn R, there is nothing wrong with making errors when learning a programming language like R.**

**You learn from your mistakes, and there is no harm done.**

**Try out the code embedded into the accompanying text and experiment with new variations to discover how the system will respond.**

---

[1] `https://www.r-project.org/Licenses/GPL-2`

# Downloading and installing R and RStudio

**R can be downloaded for free from CRAN*.**

**A *binary version* is usually simplest to use and can be installed in Windows and Mac fairly easily.**

**A binary version is available for Windows from the web page** `http://cloud.r-project.org/bin/windows/base`**.**

**The "setup program"setup is usually a file with a name like** `R-3.6.1-win.exe`**.**

**Clicking on this file will start an almost automatic installation of the R system. Clicking "Next" several times is often all that is necessary in order to complete the installation.**

---

*`http://cloud.r-project.org`

## Downloading and installing R and RStudio

**An R icon (R) will appear on your computer's desktop upon completion.**

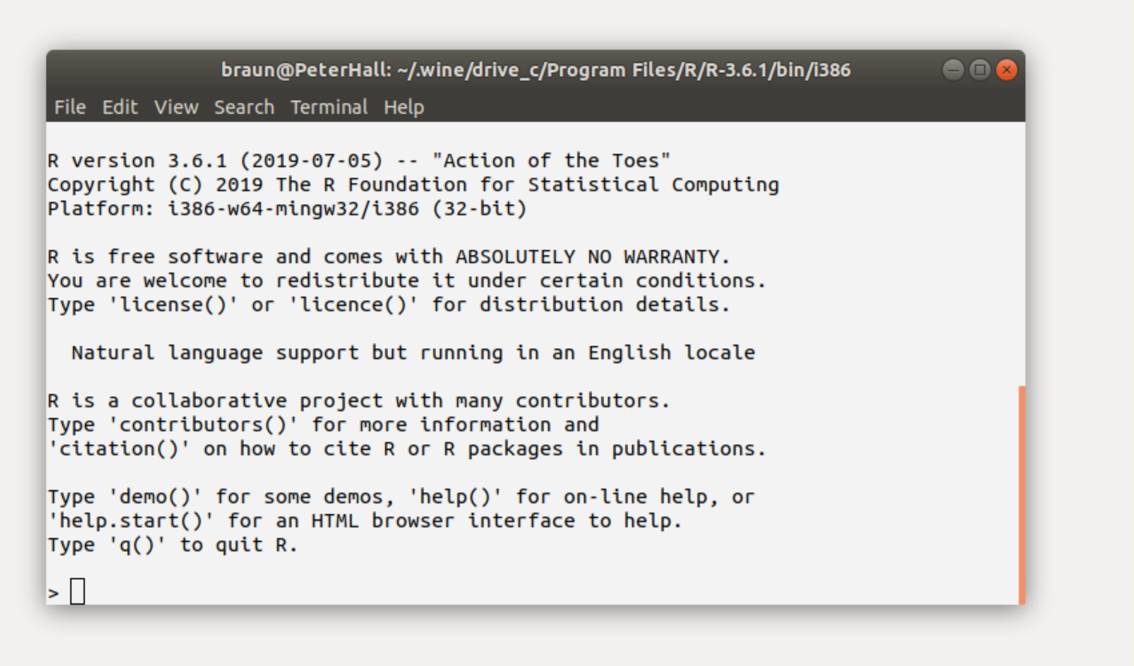
**RStudio is also very popular. You can download the "Open Source Edition" of "RStudio Desktop" from `http://www.rstudio.com/`, and follow the instructions to install it on your computer.**

**Although much or all of what is described here can be carried out in RStudio, there will be little further comment about that environment.**

**Thus, you might find that some of the instructions to be carried out at the command line can also be carried out with the menu system in RStudio.**

# Executing commands in R

**Clicking on the R icon, or opening RStudio similarly, should provide you with access to a window or pane, called the *R console* in which you can execute commands.**

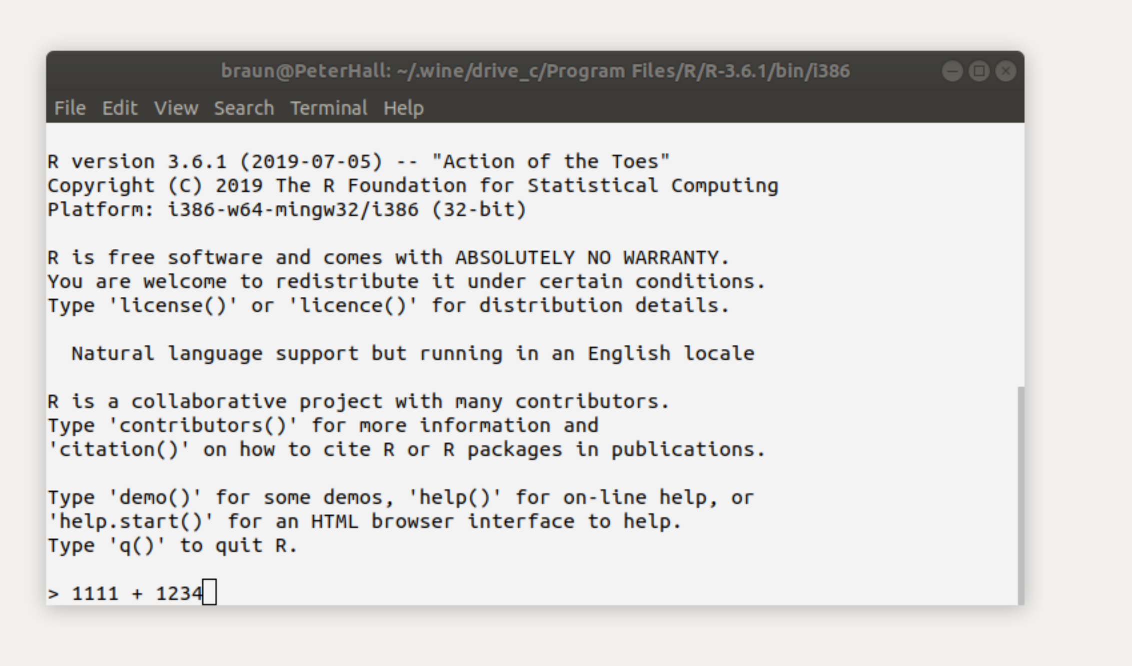


```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

**The $>$ sign is the R prompt which indicates where you can type in the command to be executed.**

# Executing commands in R

**You can do arithmetic of any type, including multiplication:**

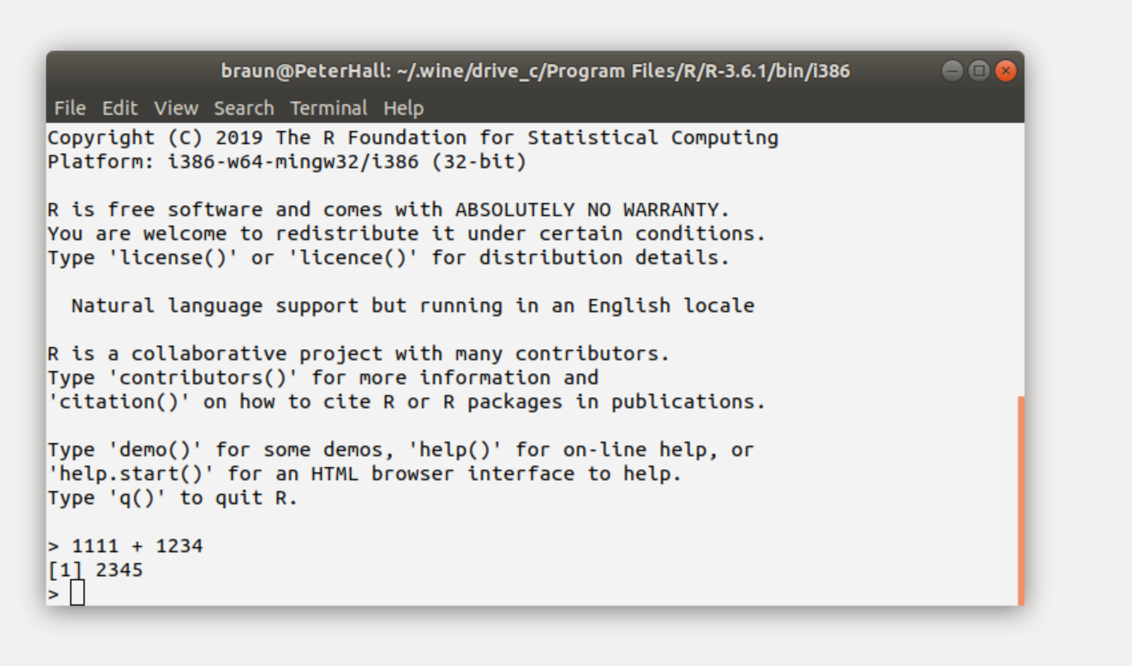


```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 1111 + 1234
```

**By hitting the “Enter” key, you are asking R to execute this calculation.**

# Executing commands in R

**The answer appears on the next line:**



```
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 1111 + 1234
[1] 2345
> 
```

**Often, you will type in commands such as this into a script window, as in RStudio, for later execution, through hitting "ctrl-R" or another related keystroke sequence.**

# Executing commands in R

**Objects that are built in to R or saved in your *workspace*, i.e. the environment in which you are currently doing your calculations, can be displayed, simply by invoking their name.**

**For example,**

```
> women
```

**the data set or *data frame* called** `women` **contains information on heights and weights of American women:**

```
##    height weight
## 1      58    115
## 2      59    117
## 3      60    120
## 4      61    123
## 5      62    126
## 6      63    129
## 7      64    132
## 8      65    135
## 9      66    139
## 10     67    142
## 11     68    146
## 12     69    150
## 13     70    154
## 14     71    159
## 15     72    164
```

## Packages

One of the major strengths of R is the availability of add-on *packages* that have been created by statisticians and computer scientists from around the world.

There are thousands of packages, e.g. *graphics*, *ggplot2*, and *MPV*.

A package contains functions and data which extend the abilities of R.

Every installation of R contains a number of packages by default (e.g. *base*, stats, and *graphics*) which are automatically loaded when you start R.

## Packages

**To load an additional package, for example, called *DAAG*, type**

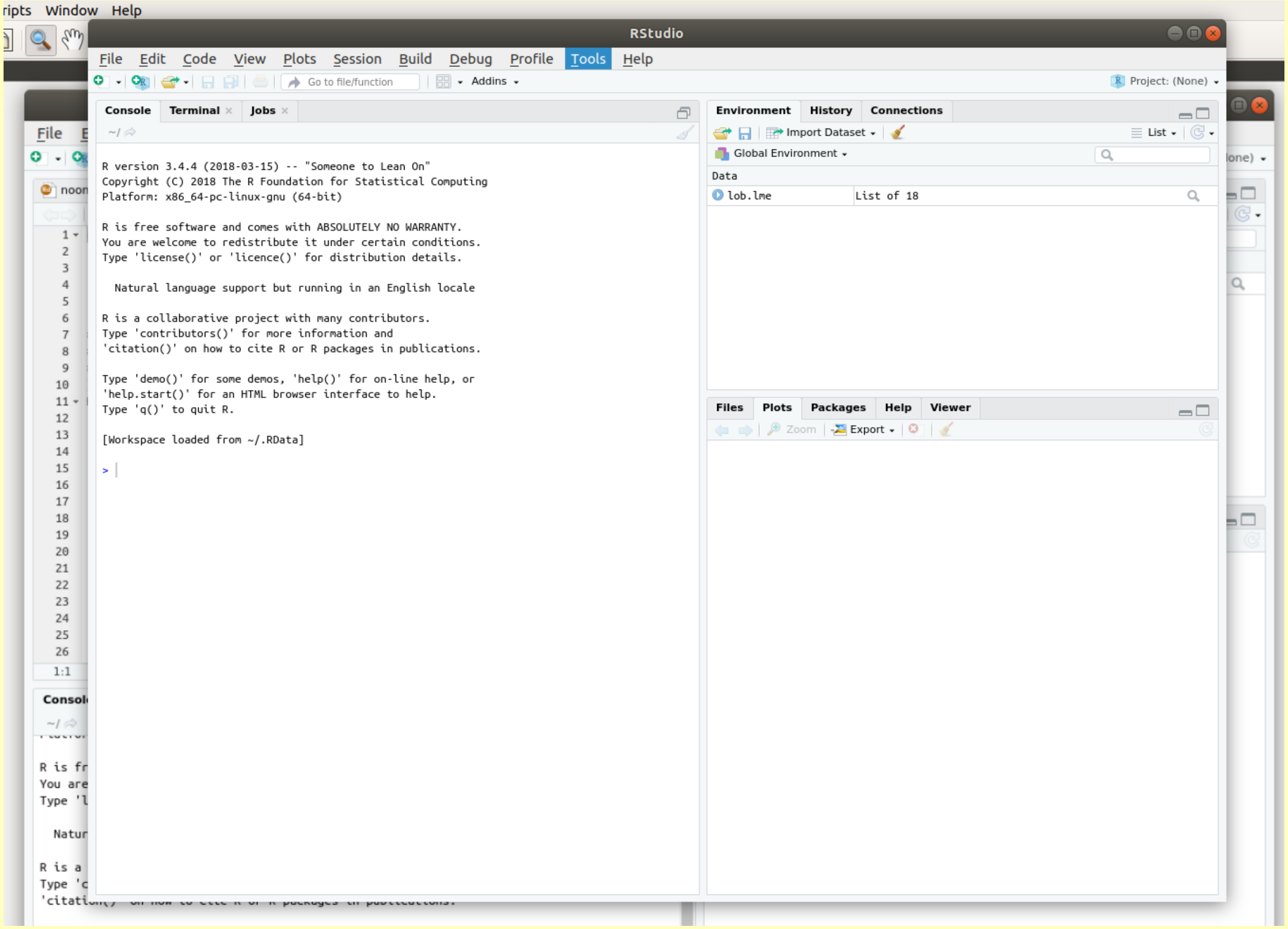
```
library(DAAG)
```

**If you get a warning that the package is can’t be found, then the package doesn’t exist on your computer, but it can likely be installed. Try**

```
install.packages("DAAG")
```

# Packages

**In *RStudio*, it may be simpler to use the `Tools` menu.**



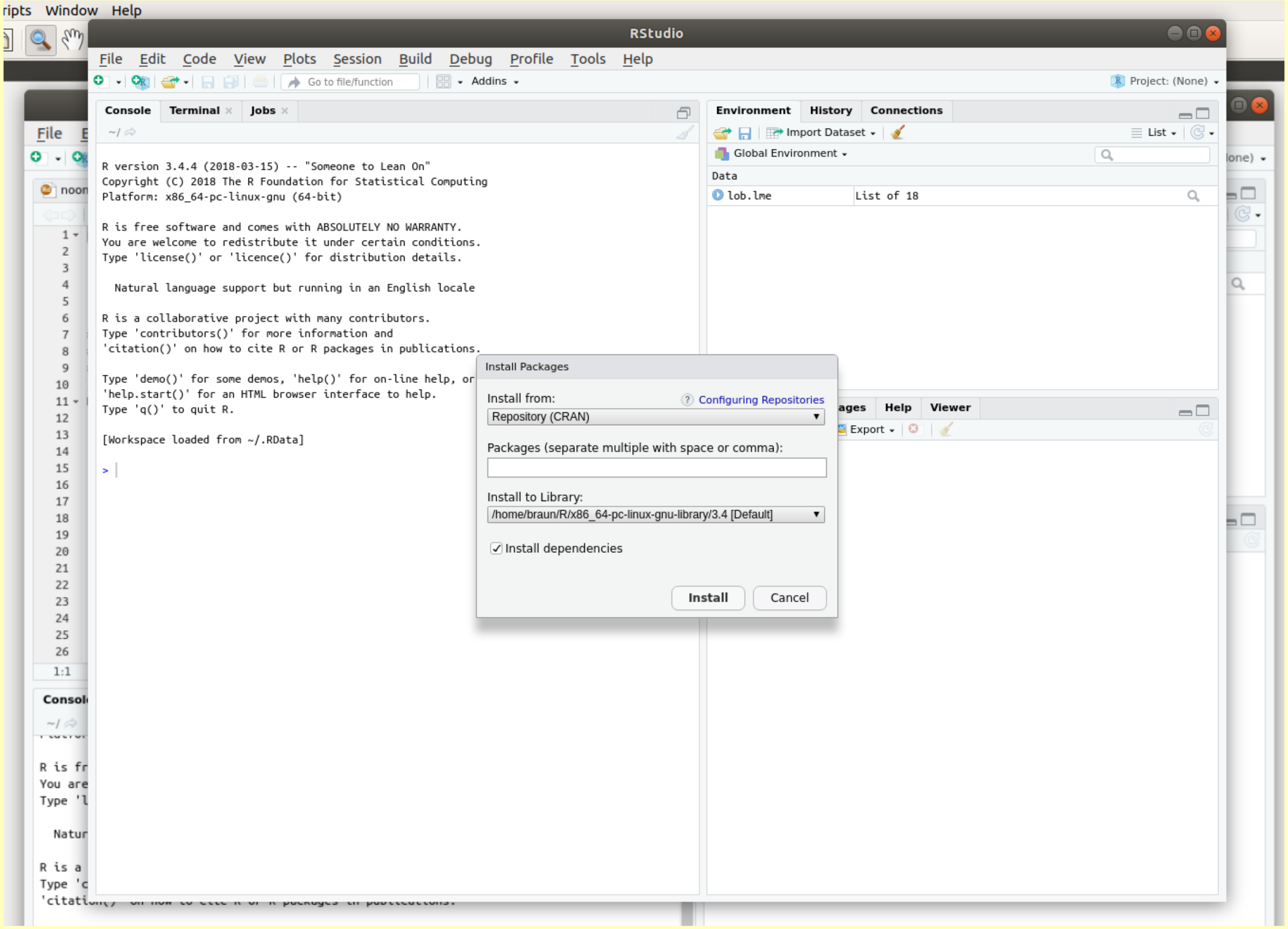
```
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

>
```

Data

| | |
|---|---|
| lob.lme | List of 18 |

# Packages

## Choose “Install Packages”:



```
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

>
```

lob.lme List of 18

Install from: Repository (CRAN)

Configuring Repositories

Packages (separate multiple with space or comma):

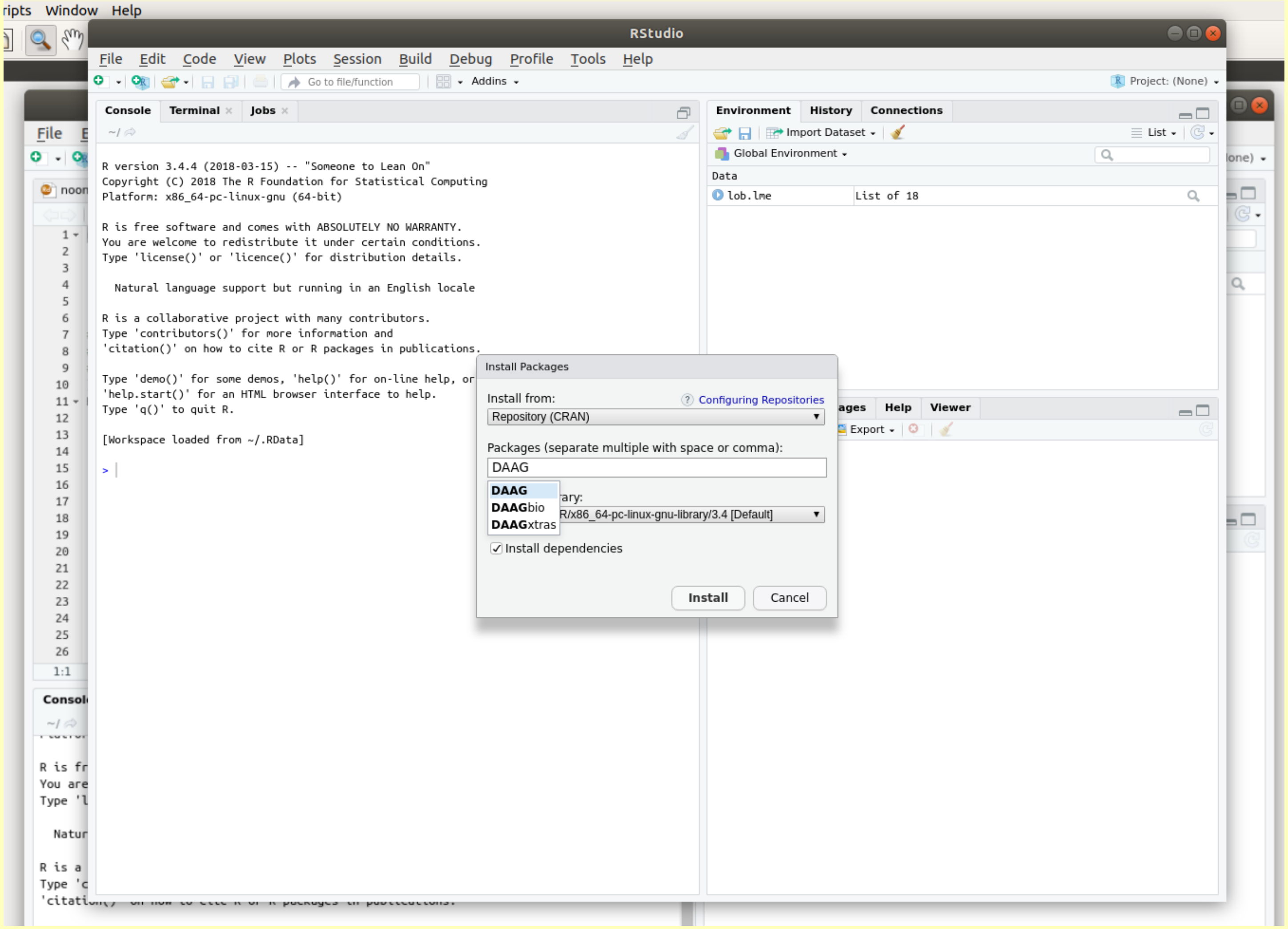
Install to Library: /home/braun/R/x86_64-pc-linux-gnu-library/3.4 [Default]

☑ Install dependencies

Install   Cancel

# Packages

**Type in the name of the package you are requesting, and click “Install”:**



```
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

>
```

lob.lme List of 18

Install Packages

Install from: Repository (CRAN)

Packages (separate multiple with space or comma): DAAG

DAAG
DAAGbio
DAAGxtras

Install to Library: R/x86_64-pc-linux-gnu-library/3.4 [Default]

☑ Install dependencies

## Packages

**Once `DAAG` is installed, it can be loaded using the `library()` function, and you can access data frames and functions that were not avaiable previously. For example, the `seedrates` data frame is now available:**

```
seedrates
```

```
##   rate grain
## 1   50  21.2
## 2   75  19.9
## 3  100  19.2
## 4  125  18.4
## 5  150  17.9
```

# Using one object from a package at a time

The ***MPV*** **package is installed on my system, but I have not loaded it. I only want to access the** `p2.12` **data frame and nothing else.**

**To do this, just type the package name (MPV), followed by two colons (**`::`**) and the object name you seek.**

```
MPV::p2.12
##    temp  usage
## 1    21 185.79
## 2    24 214.47
## 3    32 288.03
## 4    47 424.84
## 5    50 454.68
## 6    59 539.03
## 7    68 621.55
## 8    74 675.06
## 9    62 562.03
## 10   50 452.93
## 11   41 369.95
## 12   30 273.98
```

# Calculations in R

**You can control the number of digits in the output with the `options()` function.**

**This is useful when reporting final results such as means and standard deviations, since including excessive numbers of digits can give a misleading impression of the accuracy in your results.**

**Compare**

```
583/31
## [1] 18.80645
```

**with**

```
options(digits=3)
583/31
## [1] 18.8
```

# Calculations in R

**Observe the patterns in the following calculations.**

```
options(digits = 18)
1111111*1111111
## [1] 1234567654321
11111111*11111111
## [1] 123456787654321
111111111*111111111
## [1] 12345678987654320
```

**The error in the final calculation is due to the way R stores information about numbers.**

**There are around 17 digits of numeric storage are avaiable.**

## Data frames

**Most data sets are stored in R as *data frames*, such as the `women` object we encountered earlier.**

**Data frames are like matrices, but where the columns have their own names.**

**You can obtain information about a built-in data frame by using the `help()` function. For example, observe the outcome to typing `help(women)`.**

**It is generally unwise to inspect data frames by printing their entire contents to your computer screen, as it is far better to use graphical procedures to display large amounts of data or to exploit numerical summaries.**

# Data frames

**The *summary()* function provides information about the main features of a data frame:**

```
summary(women)

##       height          weight
##   Min.   :58.0   Min.   :115
##   1st Qu.:61.5   1st Qu.:124
##   Median :65.0   Median :135
##   Mean   :65.0   Mean   :137
##   3rd Qu.:68.5   3rd Qu.:148
##   Max.   :72.0   Max.   :164
```

# Data frames

**Columns can be of different types from each other. An example is the built-in `chickwts` data frame:**

```
summary(chickwts)

##      weight             feed   
##  Min.   :108   casein   :12  
##  1st Qu.:204   horsebean:10  
##  Median :258   linseed  :12  
##  Mean   :261   meatmeal :11  
##  3rd Qu.:324   soybean  :14  
##  Max.   :423   sunflower:12  
```

**One column is of factor type while the other is numeric.**

## Data frames

**If you want to see the first few rows of a data frame, you can use the *head()* function:**

```
head(chickwts)
```

```
##   weight      feed
## 1    179 horsebean
## 2    160 horsebean
## 3    136 horsebean
## 4    227 horsebean
## 5    217 horsebean
## 6    168 horsebean
```

**The *tail()* function displays the last few rows.**

# Data frames

**The number of rows can be determined using the *nrow()* function:**

```
nrow(chickwts)

## [1] 71
```

**Similarly, the *ncol()* function counts the number of columns.**

# Data frames

**The `str()` function is another way to extract information about a data frame:**

```
str(chickwts)

## 'data.frame':	71 obs. of  2 variables:
##  $ weight: num  179 160 136 227 217 168 108 124 143 140 ...
##  $ feed  : Factor w/ 6 levels "casein","horsebean",..: 2 2 2 2 2 2 2
```

## Reading data into a data frame from an external file

**If you have prepared the data set yourself, you could simply type it into a text file, for example called `mydata.txt`, perhaps with a header indicating column names, and where you use blank spaces to separate the data entries.**

**The *read.table()* function will read in the data for you as follows:**

```
mydata <- read.table("mydata.txt", header = TRUE)
```

**The object `mydata` now contains the data read in from the external file.**

## Reading data into a data frame from an external file

**You could use any name that you wish in place of `mydata`, as long as the first element of its name is an alphabetic character.**

**If the data entries are separated by commas and there is no header row, as in the file *wx_l3_2006.txt*, you would type:**

```
wx1 <- read.table("wx_l3_2006.txt", header=F, sep=",")
```

## Reading data into a data frame from an external file

**Often, your data will be in a spreadsheet.**

**If possible, export it as a `.csv` file and use something like the following to read it in.**

```
wx2 <- read.table("wx_13_fwi_2006-2011.csv",
    header=FALSE, sep=",")
```

**If you cannot export to `.csv`, you can leave it as `.xlsx` and use the *read.xslx()* command in the *xlsx* package (Dragulescu and Arendt, 2018).**

# Reading data into a data frame from an external file

**When reading in a file with columns separated by blanks with blank missing values, you can use code such as**

```
dataset1 <- read.table("file1.txt", header=TRUE,
     sep=" ", na.string=" ")
```

**This tells R that the blank spaces should be read in as missing values.**

## Reading data into a data frame from an external file

**Observe the contents of `dataset1`:**

```
dataset1
##      x  y   z
## 1    3  4  NA
## 2   51 48  23
## 3   23 33 111
```

**Note the appearance of *NA*.**
**This represents a *missing value*.**
**Functions such as `is.na()` are important for *detecting missing values* in vectors and data frames.**

**For more information about handling of missing values, check out the `See Also` section of `help(is.na)` and the *mice* package (van Buuren and Groothuis-Oudshoorn, 2011).**

# Reading data into a data frame from an external file

**Sometimes, external software exports data files that are tab-separated. When reading in a file with columns separated by tabs with blank missing values, you could use code like**

```
dataset2 <- read.table("file2.txt", header=TRUE,
    sep="\t", na.string=" ")
```

**Again, observe the result:**

```
dataset2
##      x   y  z
## 1 33 223 NA
## 2 32  88  2
## 3  3  NA NA
```

**If you need to skip the first 3 lines of a file to be read in, use the** `skip=3` **argument.**

# Extracting information from data frames

**To extract the `height` column from the `women` data frame, use the `$` operator:**

```
women$height

##  [1] 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
```

## Extracting information from data frames

**If you want only the chicks who were fed horsebean, you can apply the *subset()* function to the `chickwts` data frame:**

```
chickHorsebean <- subset(chickwts, feed == "horsebean")
chickHorsebean
```

```
##    weight      feed
## 1     179 horsebean
## 2     160 horsebean
## 3     136 horsebean
## 4     227 horsebean
## 5     217 horsebean
## 6     168 horsebean
## 7     108 horsebean
## 8     124 horsebean
## 9     143 horsebean
## 10    140 horsebean
```

# Extracting information from data frames

**You can now calculate the mean and standard deviation, and so on, of these weights:**

```
mean(chickHorsebean$weight) # mean

## [1] 160.2

sd(chickHorsebean$weight)     # standard deviation

## [1] 38.626
```

## Extracting information from data frames

**In order to extract the 4th row from the `chickHorsebean` data frame, type**

```
chickHorsebean[4, ]
```

```
##   weight      feed
## 4    227 horsebean
```

**To extract the element in the 2nd column of the 7th row of `women`, type**

```
women[7, 2]
```

```
## [1] 132
```

# Extracting information from data frames

**If we want the elements in the 4th through 7th row of the 2nd column of `women`, we can use**

```
women[4:7, 2]

## [1] 123 126 129 132
```

**Note the use of the *:* operator:**

```
4:7

## [1] 4 5 6 7
```

# Extracting information from data frames

**Another built-in data frame is `airquality`.**

**If we want to compute the mean for each of the first 4 columns of this data frame, we can use the *sapply()* function:**

```
sapply(airquality[, 1:4], mean)
##    Ozone Solar.R    Wind    Temp
##       NA      NA  9.9575 77.8824
```

**The `sapply()` function applies the same function to all columns of the supplied data frame.**

# Factors

**Factors offer an alternative, often more efficient, way of storing character data.**

**For example, a factor with 6 elements and having the two levels,** `control` **and** `treatment` **can be created using:**`factor()`

```
grp <- c("control", "treatment",  "control", "treatment",
    "treatment", "control")
grp
```

```
## [1] "control"   "treatment" "control"   "treatment"
## [5] "treatment" "control"
```

```
grp <- factor(grp)
grp
```

```
## [1] control   treatment control   treatment treatment
## [6] control
## Levels: control treatment
```

# Factors

**Consider the built-in data frame *InsectSprays***

```
summary(InsectSprays)

##      count      spray
##  Min.   : 0.0   A:12
##  1st Qu.: 3.0   B:12
##  Median : 7.0   C:12
##  Mean   : 9.5   D:12
##  3rd Qu.:14.2   E:12
##  Max.   :26.0   F:12
```

**The second column of this data frame is a *factor* representing the different types of spray used in the associated experiment.**

# Factors

**The levels of a factor can be listed using the *levels()* function:**

```
levels(InsectSprays$spray)
## [1] "A" "B" "C" "D" "E" "F"
```

# Factors

**Factors are a more efficient way of storing character data when there are repeats among the vector elements.**

**This is because the levels of a factor are internally coded as integers.**

**To see what the codes are for the `spray` factor, we can type**

```
as.integer(InsectSprays$spray)

##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
## [24] 2 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4
## [47] 4 4 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6
## [70] 6 6 6
```

**The labels for the levels are only stored once each, rather than being repeated.**

# Factors

**We can change the labels for the factor using the *levels()* function as follows:**

```
levels(InsectSprays$spray)[3] <- "Raid"
```

**Observe the effect of the change in**

```
summary(InsectSprays$spray)


##      A      B  Raid     D      E      F
##     12     12     12     12     12     12
```

# Factors

**The `levels()` function also offers a simple way to collapse categories.**

**Suppose we are interested in comparing the first three levels with the last three levels.**

**We can create a new factor for this purpose as follows:**

```
InsectSprays$newFactor <- InsectSprays$spray
levels(InsectSprays$newFactor) <- c("A", "A", "A",
     "B", "B", "B")
```

# Factors

## Check the result:

```
summary(InsectSprays)

##      count           spray    newFactor
##  Min.   : 0.0   A    :12   A:36
##  1st Qu.: 3.0   B    :12   B:36
##  Median : 7.0   Raid:12
##  Mean   : 9.5   D    :12
##  3rd Qu.:14.2   E    :12
##  Max.   :26.0   F    :12
```

# Tibbles

**A *tibble* can be created from an existing data frame, using the `as_tibble()` function, found in the *tibble* package (Wickham, 2017).**

```
library(tibble) # install.packages("tibble"), if needed
trees.tbl <- as_tibble(trees) # trees is a data frame
```

# Tibbles

**Tibbles are like data frames, but they prevent you from doing silly things, like printing a whole data set to the screen:**

```
trees.tbl # trees.tbl is a tibble

## # A tibble: 31 x 3
##    Girth Height Volume
##    <dbl>  <dbl>  <dbl>
##  1   8.3     70   10.3
##  2   8.6     65   10.3
##  3   8.8     63   10.2
##  4  10.5     72   16.4
##  5  10.7     81   18.8
##  6  10.8     83   19.7
##  7  11       66   15.6
##  8  11       75   18.2
##  9  11.1     80   22.6
## 10  11.2     75   19.9
## # ... with 21 more rows
```

## Getting a glimpse of a tibble

**The `glimpse` function is similar to `str` but a little friendlier:**

```
glimpse(trees.tbl)

## Observations: 31
## Variables: 3
## $ Girth  <dbl> 8.3, 8.6, 8.8, 10.5, 10.7, 10....
## $ Height <dbl> 70, 65, 63, 72, 81, 83, 66, 75...
## $ Volume <dbl> 10.3, 10.3, 10.2, 16.4, 18.8, ...
```

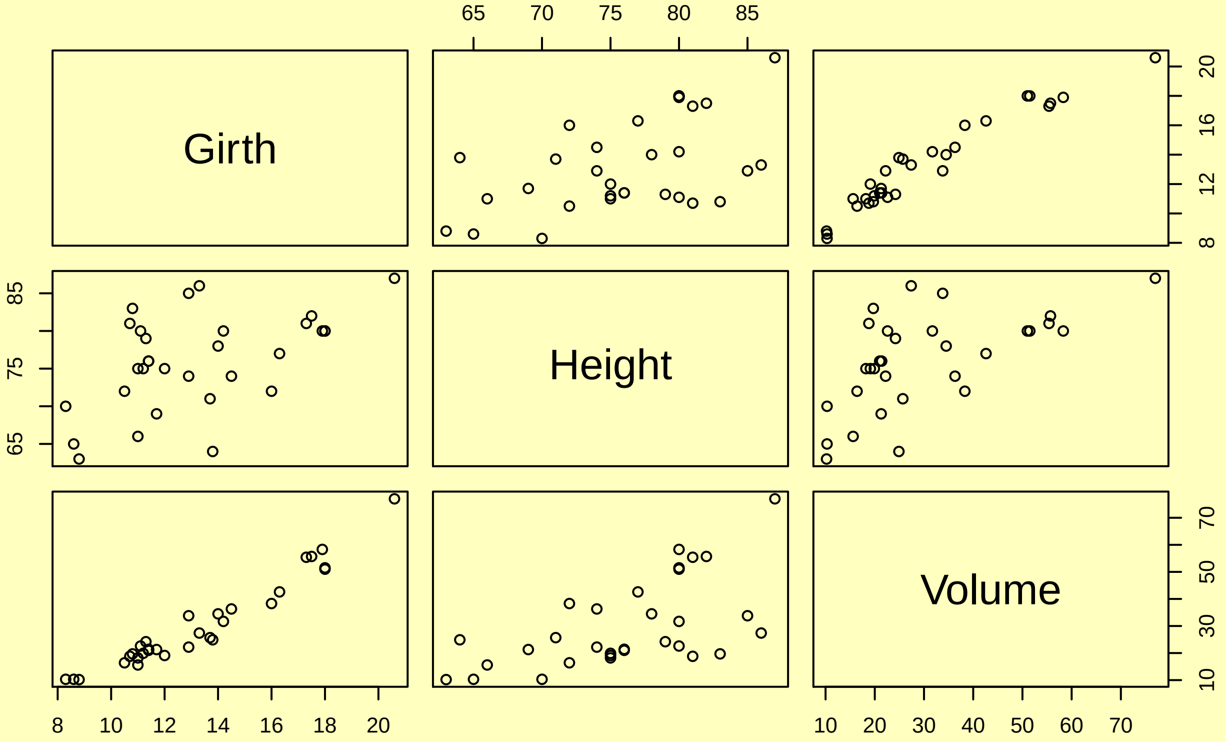
# Tibbles are like data frames

**Tibbles act like data frames in some ways. Functions such as *summary()* and `str()` are still useful. For example,**

```
str(trees.tbl)


## Classes 'tbl_df', 'tbl' and 'data.frame': 31 obs. of  3 va
##  $ Girth : num  8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2
##  $ Height: num  70 65 63 72 81 83 66 75 80 75 ...
##  $ Volume: num  10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22
```

# The `plot()` function

```
plot(trees.tbl)
```



**This is a scatterplot matrix.**

**The effect of `plot(trees)` would have been the same.**

# Tibbles are not like data frames

**The girth of a tree is like its circumference, so we might expect the volume of the tree to be related to the square of girth times height.**

**Specifically, we might predict volume from girth and height using the following formula:**

$$V = \frac{G^2 H}{4\pi}$$

**We can calculate this prediction from the given data and see how much error there is.**

**To do this, we need functions in the *tidyr* and *dplyr* packages.**

```
library(tidyr)
library(dplyr)     # or just use library(tidyverse)
```

# Tibbles are not like data frames

```
trees.tbl <- trees.tbl %>%
  mutate(VolumePredicted = Girth^2*Height/(4*pi))
```

```
trees.tbl

## # A tibble: 31 x 4
##     Girth Height Volume VolumePredicted
##     <dbl>  <dbl>  <dbl>           <dbl>
##  1    8.3     70   10.3            384.
##  2    8.6     65   10.3            383.
##  3    8.8     63   10.2            388.
##  4   10.5     72   16.4            632.
##  5   10.7     81   18.8            738.
##  6   10.8     83   19.7            770.
##  7   11       66   15.6            636.
##  8   11       75   18.2            722.
##  9   11.1     80   22.6            784.
## 10   11.2     75   19.9            749.
## # ... with 21 more rows
```

# Tibbles are not like data frames

**Why are the predicted volumes off by so much?**

**To find the answer, read the help file to find that the Girth measurements are actually diameter measurements in inches.**

**The other variables are in terms of feet.**

**Re-doing the calculation with diameter, instead of girth, we have**

$$V = \frac{\pi * D^2 H}{4(12)^2}$$

**We can calculate this prediction from the given data and see how much error there is:**

```
trees.tbl <- trees.tbl %>%
  mutate(VolumePredicted = Girth^2*Height/(4*12^2))
```

# Tibbles are not like data frames

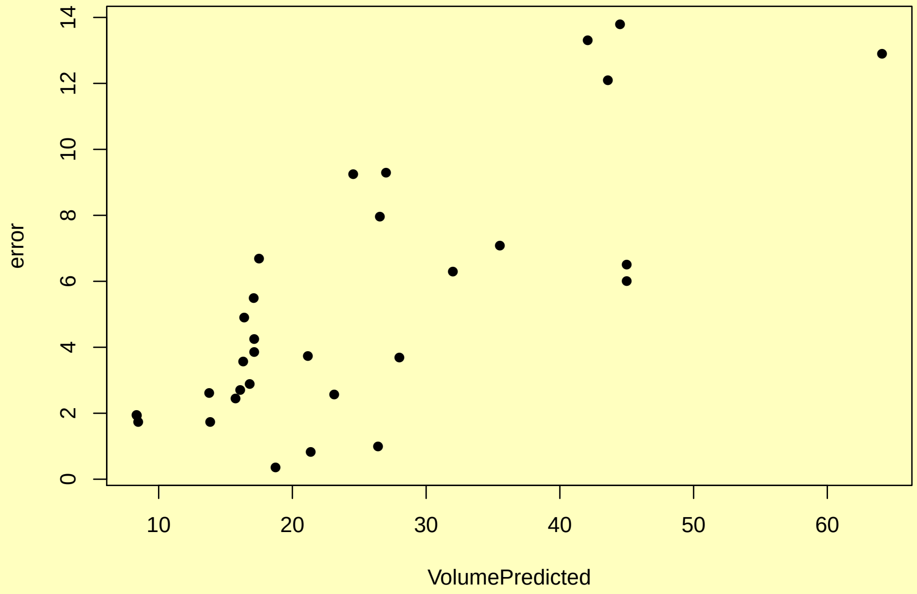
```
trees.tbl
```

```
## # A tibble: 31 x 4
##    Girth Height Volume VolumePredicted
##    <dbl>  <dbl>  <dbl>           <dbl>
##  1   8.3     70   10.3            8.37
##  2   8.6     65   10.3            8.35
##  3   8.8     63   10.2            8.47
##  4  10.5     72   16.4           13.8 
##  5  10.7     81   18.8           16.1 
##  6  10.8     83   19.7           16.8 
##  7  11       66   15.6           13.9 
##  8  11       75   18.2           15.8 
##  9  11.1     80   22.6           17.1 
## 10  11.2     75   19.9           16.3 
## # ... with 21 more rows
```

# Visualizing the errors in the volume predictions

**The code below causes the errors or *residuals* to be plotted against the predicted volumes as below.**

```
trees.tbl <- trees.tbl %>%
     mutate(error = Volume - VolumePredicted)
plot(error ~ VolumePredicted, data = trees.tbl)
```



**Note that we are still systematically under-predicting the volume and the prediction error is increasing with diameter.**

# References

1. Ernesto Barrios (2016). BHH2: Useful Functions for Box, Hunter and Hunter II. R package version 2016.05.31. URL `https://CRAN.R-project.org/package=BHH2`.

2. W.J. Braun (2019). MPV: Data Sets from Montgomery, Peck and Vining. R package version 1.55. URL `https://CRAN.R-project.org/package=MPV`.

3. W. John Braun and Duncan Murdoch (2016). *A First Course in Statistical Programming with R* Second Edition. Cambridge University Press.

4. Adrian A. Dragulescu and Cole Arendt (2018). xlsx: Read, Write, Format Excel 2007 and Excel 97/2000/XP/2003 Files. R package version 0.6.1. URL `https://CRAN.R-project.org/package=xlsx`.

5. John H. Maindonald and W. John Braun (2006). *Data Analysis and Graphics.* Third Edition. Cambridge University Press.

6. John H. Maindonald and W. John Braun (2015). DAAG: Data Analysis and Graphics Data and Functions. R package version 1.22. URL `https://CRAN.R-project.org/package=DAAG`.

7. R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL `https://www.R-project.org/`.

8. R Core Team (1999-2018). An Introduction to R. Version 3.6.1 (2019-07-05).

9. Sarkar, Deepayan (2008) *Lattice: Multivariate Data Visualization with R.* Springer, New York. ISBN 978-0-387-75968-5

10. **Stef van Buuren, Karin Groothuis-Oudshoorn (2011). mice: Multivariate Imputation by Chained Equations in R. Journal of Statistical Software, 45(3), 1-67. URL `https://www.jstatsoft.org/v45/i03/`.**

11. **Hadley Wickham (2011). The Split-Apply-Combine Strategy for Data Analysis. Journal of Statistical Software, 40(1), 1-29. URL `http://www.jstatsoft.org/v40/i01/`.**

12. **Hadley Wickham (2017). tidyverse: Easily Install and Load the ’Tidyverse’. R package version 1.2.1. `https://CRAN.R-project.org/package=tidyverse`**

13. **Hadley Wickham, Peter Danenberg, Gábor Csárdi, Manuel Eugster (2019). roxygen2: In-Line Documentation for R. R package version 7.0.1. `https://CRAN.R-project.org/package=roxygen2`**