UNIVERSITY OF CALGARY

BioSci UofC

R Wizardry

Winter 2018

Instructors:

Oscar Montoya, M.Sc.

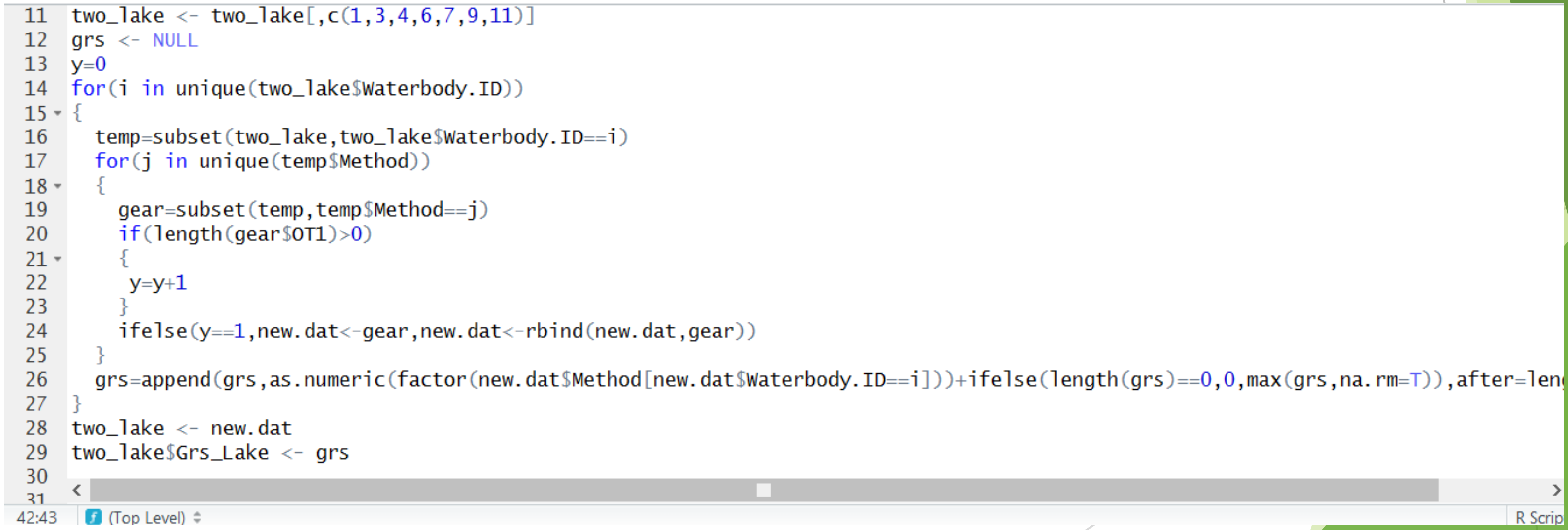
Ryan Tate, M.Sc.

Course coordinators:

Dr. Paul Galpern

Dr. Samuel Yeaman

Introduction to programming in R/RStudio



```
two_lake <- two_lake[,c(1,3,4,6,7,9,11)]
grs <- NULL
y=0
for(i in unique(two_lake$Waterbody.ID))
{
  temp=subset(two_lake,two_lake$Waterbody.ID==i)
  for(j in unique(temp$Method))
  {
    gear=subset(temp,temp$Method==j)
    if(length(gear$OT1)>0)
    {
     y=y+1
    }
    ifelse(y==1,new.dat<-gear,new.dat<-rbind(new.dat,gear))
  }
  grs=append(grs,as.numeric(factor(new.dat$Method[new.dat$Waterbody.ID==i]))+ifelse(length(grs)==0,0,max(grs,na.rm=T)),after=len
}
two_lake <- new.dat
two_lake$Grs_Lake <- grs
```

Disclaim:

This is NOT a statistics course...

We apologize if we disappointed you...

Communication channels:
Slack (preferred) or email

Office hours:
Mondays 15:00 – 17:00.
Room TBD

Learning to code

- Coding can be fun
  - Treat it as a game
  - Many ways to solve the same problem
  - R can be run line-by-line; play around!

- Coding is an investment
  - May be slow at first but easy to repeat and modify.
  - Add a valuable set of skills to your C.V.
  - Fastest growing job market

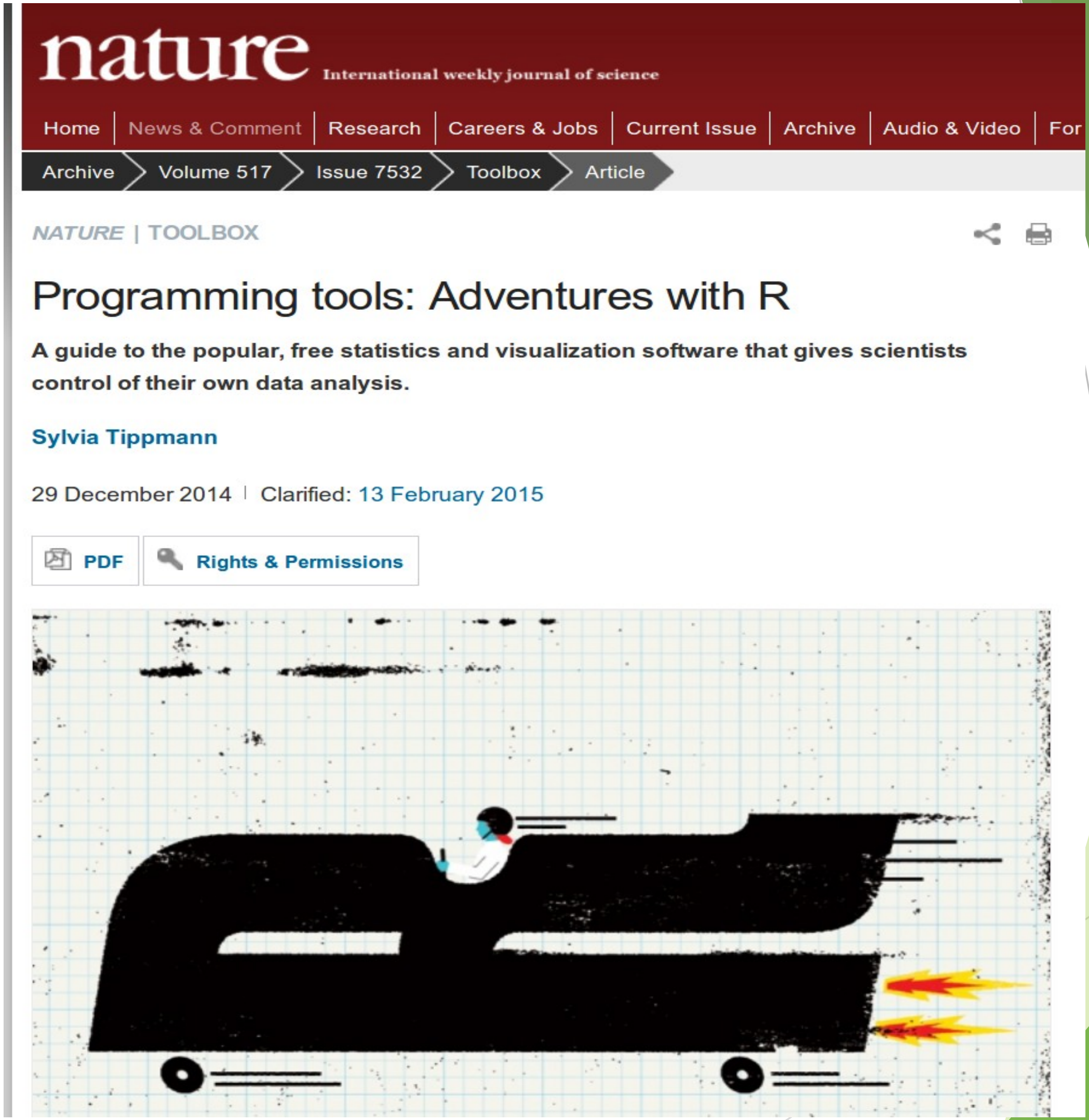
Remember: You're learning a new language!

“R is becoming the “lingua franca” of data science” (R-Bloggers)

- R, a dialect of S/S+, is widely used for advanced statistics/plotting in data science/analytics.
- Codes can be recycled (“the best bioinformatician is the laziest one”).
- Hundreds of books and (many are open source!) with examples and available data sets.
- Several free courses (edx, coursera, Code School, Code Academy, DataCamp, among others).

- Helpful websites: The Comnprehensive R Archive Network (CRAN), Stack Overflow, R-Bloggers, discussion blogs, among others.
- Need more motivation to start using R?

It's open source = free!



NATURE | TOOLBOX

Programming tools: Adventures with R

A guide to the popular, free statistics and visualization software that gives scientists control of their own data analysis.

Sylvia Tippmann

29 December 2014 | Clarified: 13 February 2015

R or a spreadsheet software? Actually, both!

When to use Excel-like packages:

- When you have something that needs nice, quick presentation.

- Inputing data is easier in Excel than in R.

- When you have quick and “dirty” number crunching to do: Handful of descriptive stats, you need to look something up, run a quick sort/filter, etc.

When to use R

- Heavy data management.

- Complex, robust statistical analyses.

- When you need serious statistical capabilities.

- When reproducibly is needed, e.g. for publication purposes
  - “Talk is cheap. Show me the code.”, Linus Torvalds)

- Building a code for repetitive tasks that you routinely perform (automation saves a lot of time).

“If you are using R and you think you're in hell, this is a map for you” (Patrick Burns, The R Inferno)

Recommended articles:

http://www.r-bloggers.com/why-you-should-learn-r-first-for-data-science/

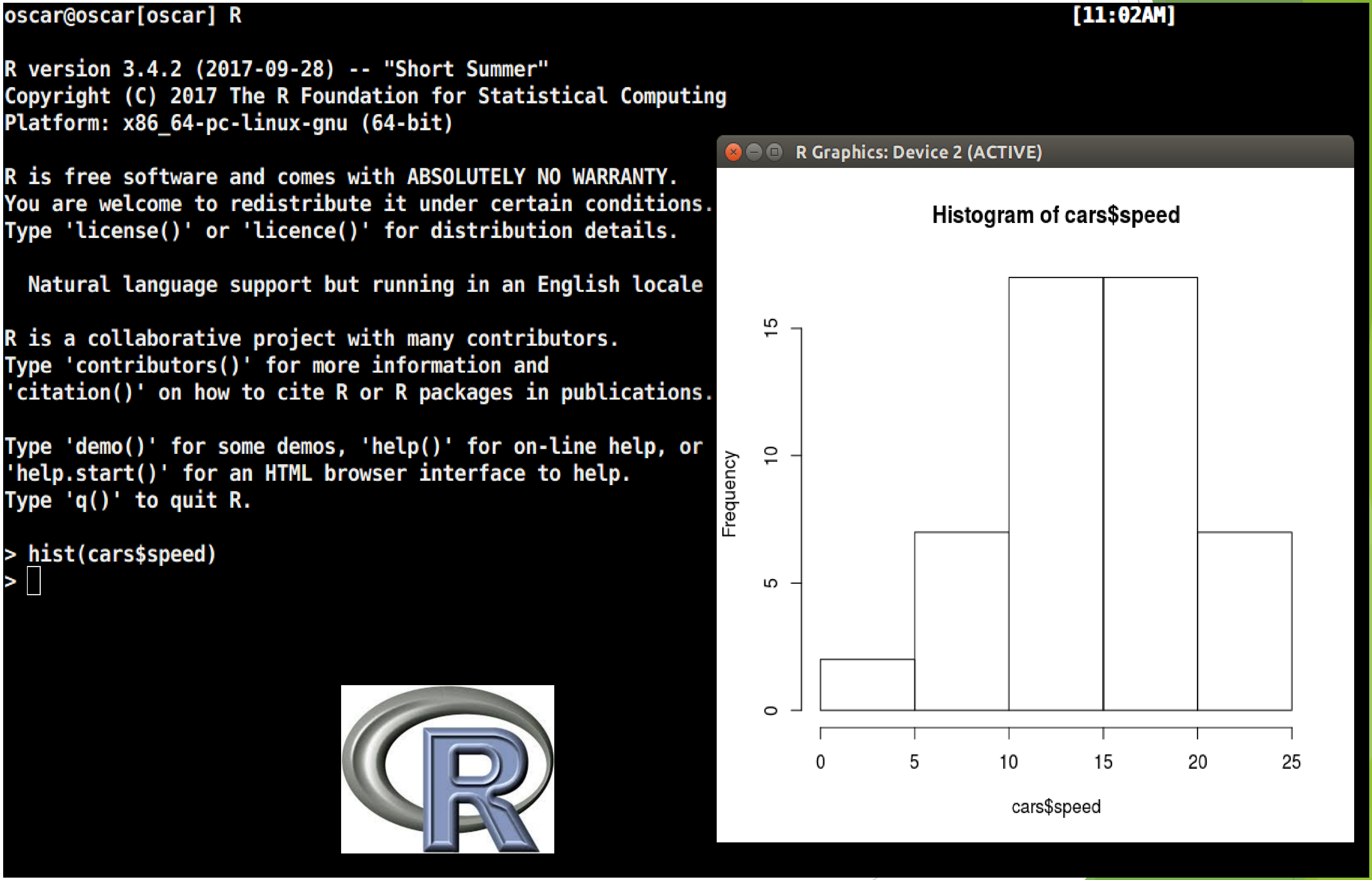
http://www.nature.com/news/programming-tools-adventures-with-r-1.16609

The R User Interface

“R gives you a language to speak in. RStudio gives you a way to talk to your computer” (RStudio team)

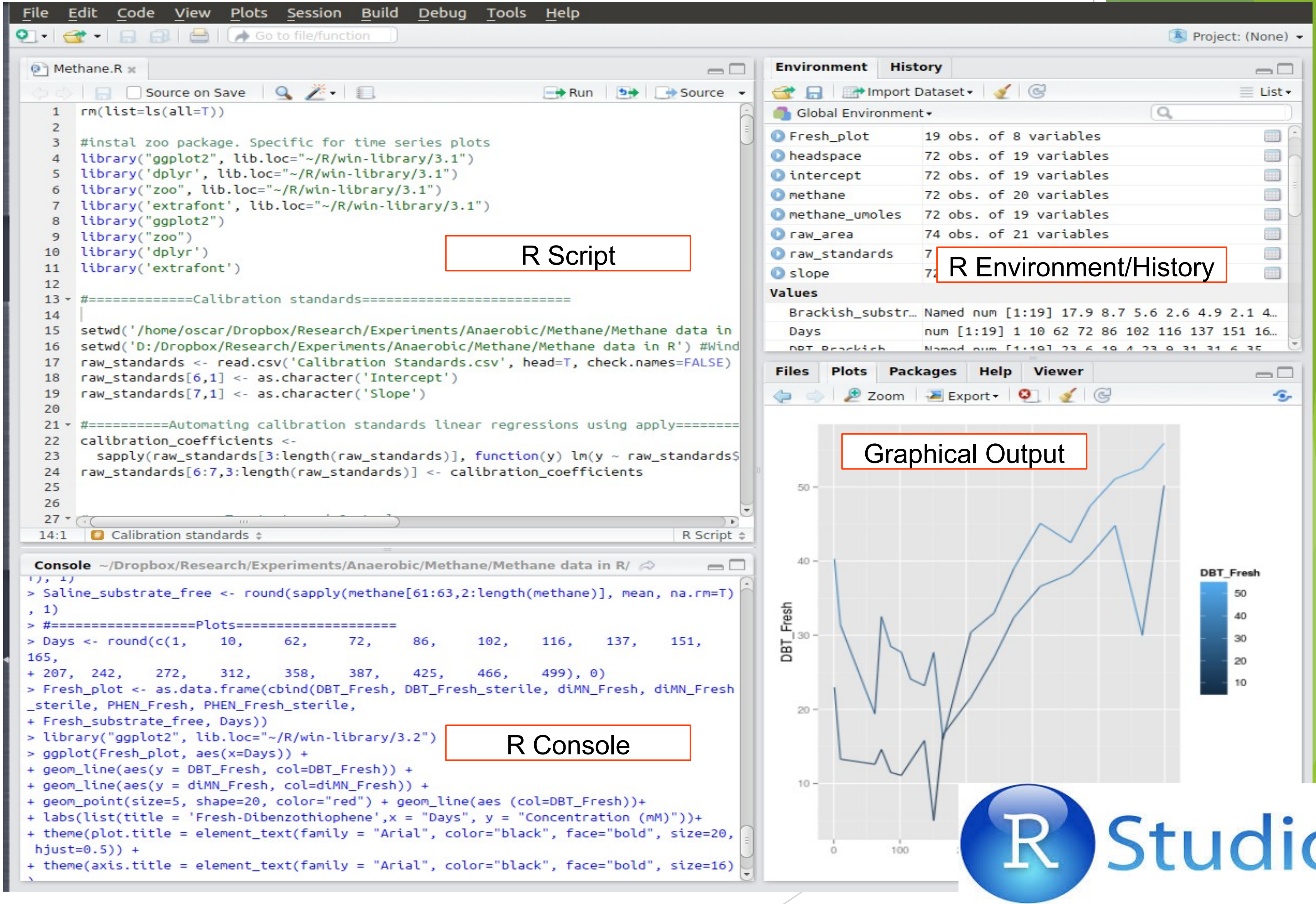
Although R can be ran from a terminal, integrated development environments (IDE) like RStudio offer a Graphic User Interface (GUI) that simplifies the visualization of the coding.

Using R from a terminal



```
oscar@oscar[oscar] R                                                    [11:02AM]

R version 3.4.2 (2017-09-28) -- "Short Summer"
Copyright (C) 2017 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.

> hist(cars$speed)
> 
```

RStudio (IDE)



R Script

```
rm(list=ls(all=T))

#instal zoo package. Specific for time series plots
library("ggplot2", lib.loc="~/R/win-library/3.1")
library('dplyr', lib.loc="~/R/win-library/3.1")
library("zoo", lib.loc="~/R/win-library/3.1")
library('extrafont', lib.loc="~/R/win-library/3.1")
library("ggplot2")
library("zoo")
library('dplyr')
library('extrafont')

#============Calibration standards=========================

setwd('/home/oscar/Dropbox/Research/Experiments/Anaerobic/Methane/Methane data in
setwd('D:/Dropbox/Research/Experiments/Anaerobic/Methane/Methane data in R') #Wind
raw_standards <- read.csv('Calibration Standards.csv', head=T, check.names=FALSE)
raw_standards[6,1] <- as.character('Intercept')
raw_standards[7,1] <- as.character('Slope')

#=========Automating calibration standards linear regressions using apply=======
calibration_coefficients <-
  sapply(raw_standards[3:length(raw_standards)], function(y) lm(y ~ raw_standards$
raw_standards[6:7,3:length(raw_standards)] <- calibration_coefficients
```

R Environment/History

```
Fresh_plot         19 obs. of 8 variables
headspace          72 obs. of 19 variables
intercept          72 obs. of 19 variables
methane            72 obs. of 20 variables
methane_umoles     72 obs. of 19 variables
raw_area           74 obs. of 21 variables
raw_standards      7
slope              72
Values
Brackish_substr…   Named num [1:19] 17.9 8.7 5.6 2.6 4.9 2.1 4…
Days               num [1:19] 1 10 62 72 86 102 116 137 151 16…
```

Graphical Output

R Console

```
Console ~/Dropbox/Research/Experiments/Anaerobic/Methane/Methane data in R/
> Saline_substrate_free <- round(sapply(methane[61:63,2:length(methane)], mean, na.rm=T)
, 1)
> #=================Plots====================
> Days <- round(c(1,    10,    62,    72,    86,    102,    116,    137,    151,
165,
+ 207,  242,    272,    312,    358,    387,    425,    466,    499), 0)
> Fresh_plot <- as.data.frame(cbind(DBT_Fresh, DBT_Fresh_sterile, diMN_Fresh, diMN_Fresh
_sterile, PHEN_Fresh, PHEN_Fresh_sterile,
+ Fresh_substrate_free, Days))
> library("ggplot2", lib.loc="~/R/win-library/3.2")
> ggplot(Fresh_plot, aes(x=Days)) +
+ geom_line(aes(y = DBT_Fresh, col=DBT_Fresh)) +
+ geom_line(aes(y = diMN_Fresh, col=diMN_Fresh)) +
+ geom_point(size=5, shape=20, color="red") + geom_line(aes (col=DBT_Fresh))+
+ labs(list(title = 'Fresh-Dibenzothiophene',x = "Days", y = "Concentration (mM)"))+
+ theme(plot.title = element_text(family = "Arial", color="black", face="bold", size=20,
 hjust=0.5)) +
+ theme(axis.title = element_text(family = "Arial", color="black", face="bold", size=16)
```

Types of data

Numeric: Can be either continuous or count data

Character: Study site is: “Alberta” v. “British Columbia”

Genetic data stored as basepairs:

“GATTACA” vs. “CTGCCAC”

Factors: Special way to store character data with a ‘numeric’ rank: “Alberta” v. “British Columbia”

- Computer interprets “Alberta” as 1, “British Columbia” as 2

Types of objects in R

Vector – a sequence of at least one stored value(s):

Matrix – 2-d form of a vector that can be indexed by rows and columns; data must be of the same type

Data Frame – like a matrix, but rows/columns can vary in type; character and numeric data allowed

Array – N-dimensional form of a vector that can be indexed by rows, columns, depth, etc.

List – form of a vector in which elements need not be of the same type; elements can be vectors, matrices, arrays, or lists themselves

All data is either numeric or character

Hence:

- Methods you learn to deal with numeric data in R will be the same regardless of the field
  - Physiology, biochemical, genetic, or ecological data

- Methods you learn to deal with character data will be the same across all character data

- Programming languages have similarities
  - C++, ADMB, R, S, SAS, etc.

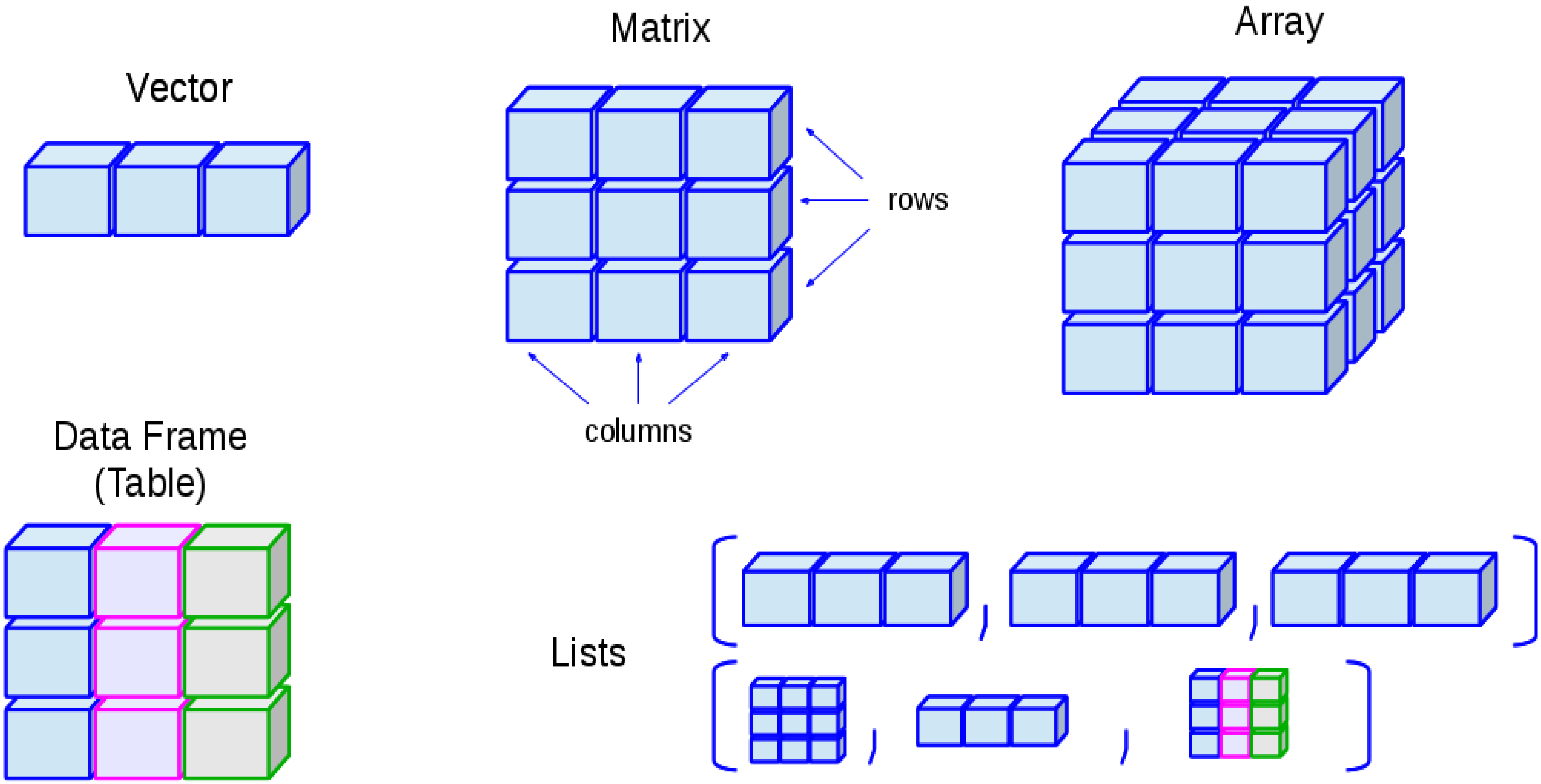
Vectors

Vectors can be thought of as contiguous cells containing data.

R has six basic (‘atomic’) vector types: Logical, integer, double (often called numeric), and character.

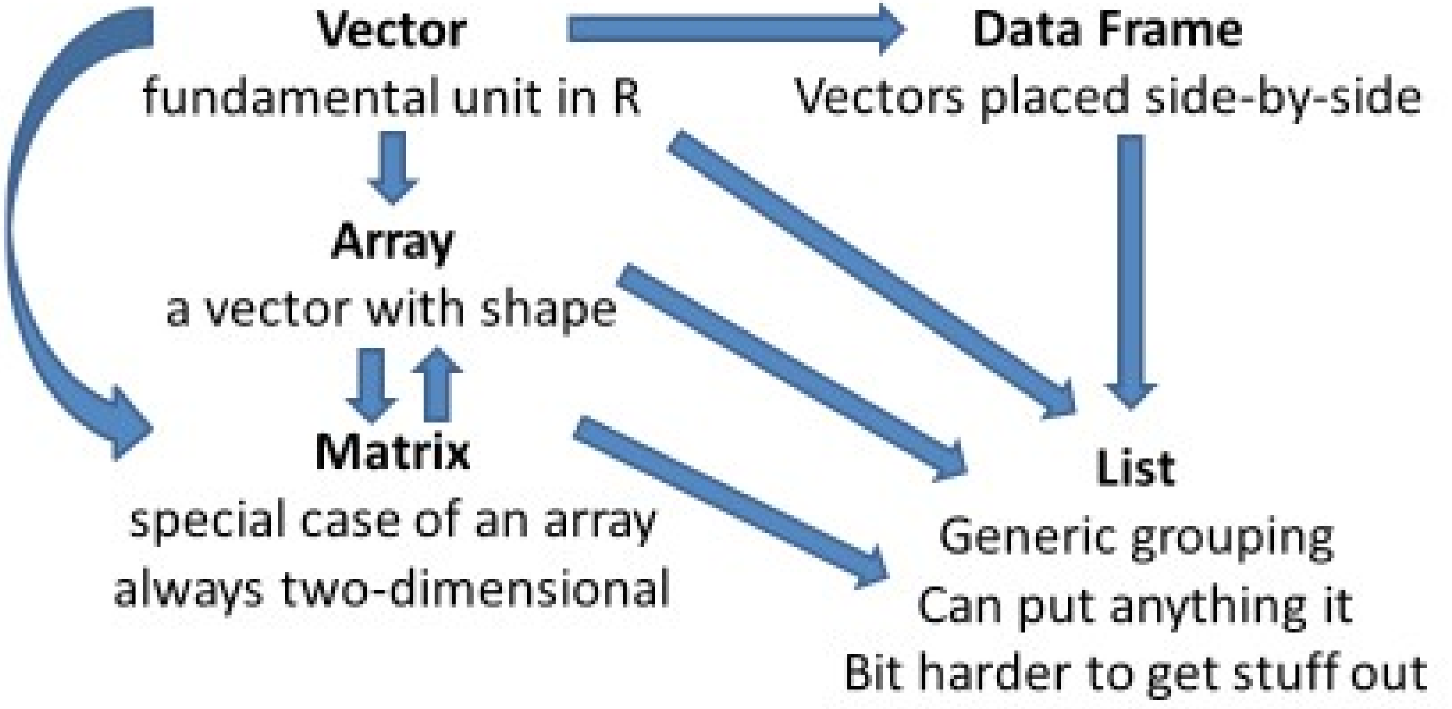
There are two other “rare” types: complex and raw.

Data structures



http://venus.ifca.unican.es/Rintro/dataStruct.html

Types of objects



Vector

fundamental unit in R

Data Frame

Vectors placed side-by-side

Array

a vector with shape

Matrix

special case of an array

always two-dimensional

List

Generic grouping

Can put anything it

Bit harder to get stuff out

R data Types

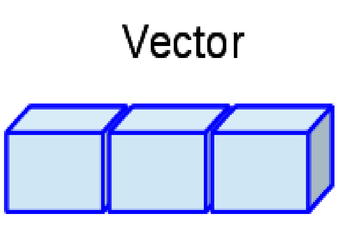
R has a wide variety of data types including:

- Scalars (a single number)
- Vectors (numerical, character, logical)
- Matrices
- Arrays
- Data frames
- Lists
- S4

Scalars and Vectors

- Scalar:

  X <- 5　　　　　　b <- 10

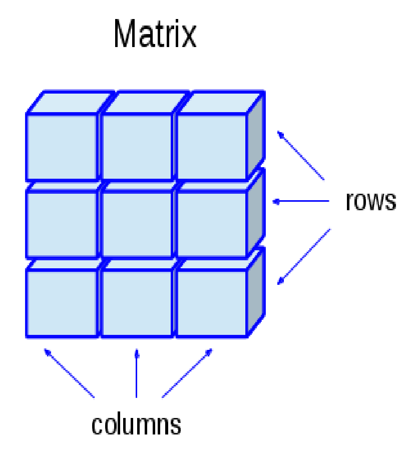


Vectors (numerical, character, logical):

```
a <- c(1,2,5.3,6,-2,4) # numeric vector
b <- c("one","two","three") # character vector
c <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE)
```

Matrices

All columns in a matrix must have the same mode (numeric, character, etc.) and the same length.



Generates 5 x 4 numeric matrix :

```
y <- matrix(1:20, nrow = 5, ncol = 4)
```

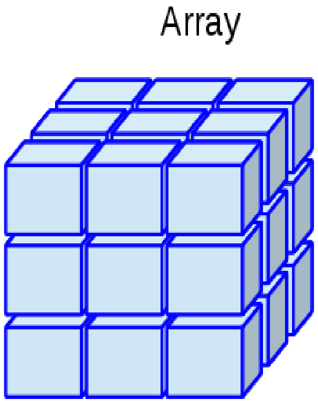
Another example

```
cells <- c(1, 26, 24, 68)
rnames <- c("R1", "R2")
cnames <- c("C1", "C2")
mymatrix <- matrix(cells, nrow = 2, ncol = 2,
                        byrow = T, dimnames=list(rnames, cnames))
```

Arrays

An **n-dimensional array** is a set of stacked matrices of identical dimensions

```
a <- matrix(8, 2, 3) # Creates a 2 x 3 matrix populated
with 8's.
b <- matrix(9, 2, 3) # Creates a 2 x 3 matrix populated

with 9's.
```

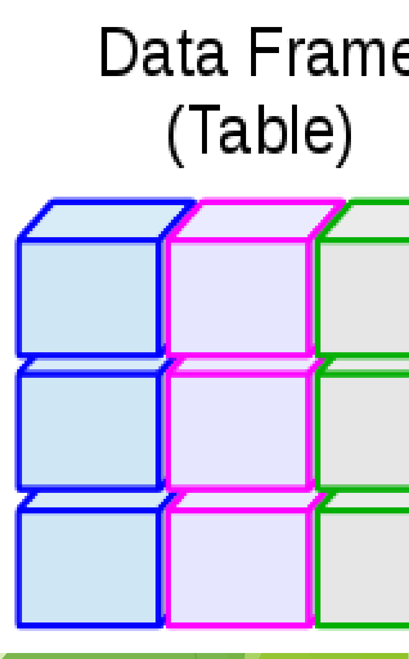


Data Frames

A data frame is more general than a matrix, in that d columns can have different modes (numeric character, factor, etc.).

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

e <- c("red", "white", "red", NA)

f <- c(TRUE,TRUE,TRUE,FALSE)

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

```
>     d    e       f
      1   red     TRUE
      2  white    TRUE
      3   red     TRUE
      4  <NA>     FALSE
```



Lists

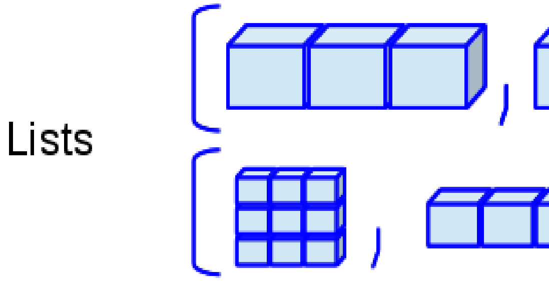
- An ordered collection of objects (components). A list allows yo gather a variety of (possibly unrelated) objects under one nam

Example of a list with 4 components -a string, a numeric vector, a matrix, and a scalar:

```
w <- list(name = "Fred", mynumbers = a, mymatrix = y,
= 5.3)
```

Example of a list containing two lists

```
v <- c(list1, list2)
```



S4 Objects

- Similar to a vector, except it has slots that can have different types of variables (character, numeric, etc.)
- Little more work to set up than other data types.

Example:

```
setClass("fieldsite", slots=list(name="character",
size="numeric", species="integer"))

s <- new("fieldsite", name="lakeawesome",
size=3.14, species=9L)
```

For more information and replicable examples go to:

- http://www.statmethods.net/input/datatypes.html

“R in Action” by Robert I. Kabacoff