Data science and analysis in Neuroscience

Kevin Allen

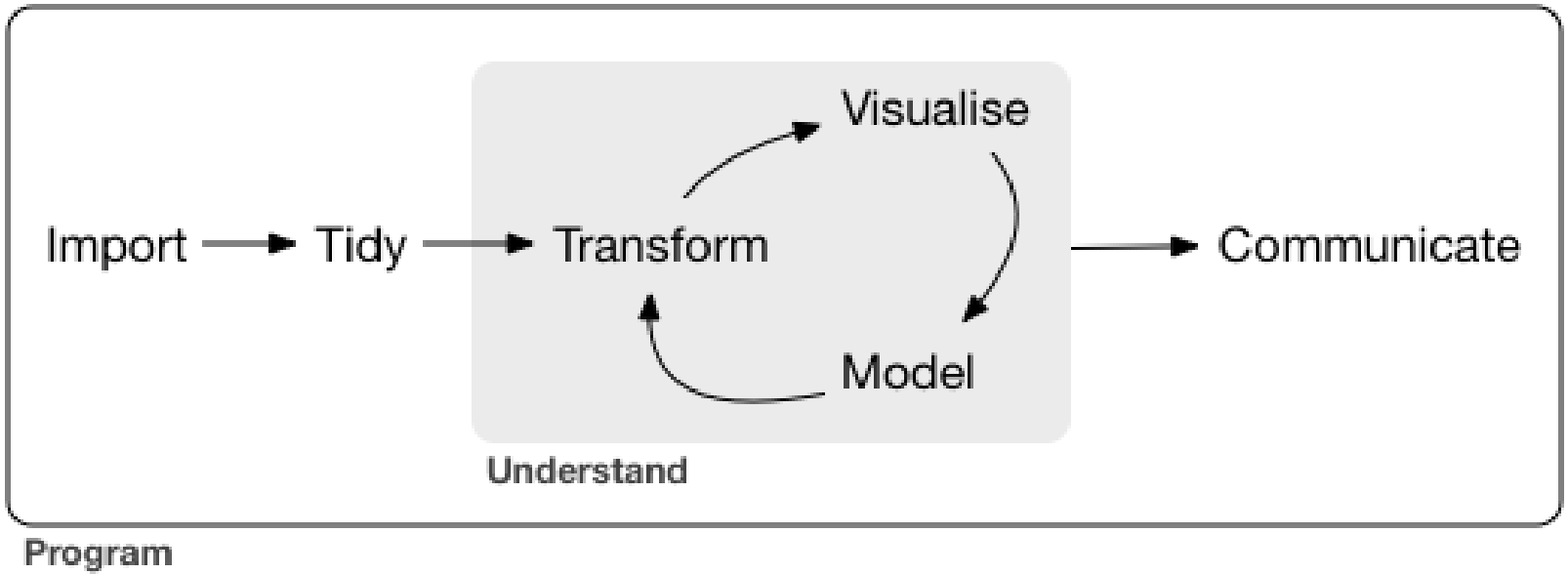
November 7, 2019

Today's plan

1. Overview of the course
2. Assessment of previous experience
3. Data science landscape and RStudio
4. Introduction to R
5. Introduction to ggplot2

What is data science?

Turn raw data into understanding, insight and knowledge.

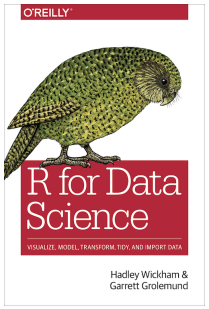


Course objectives

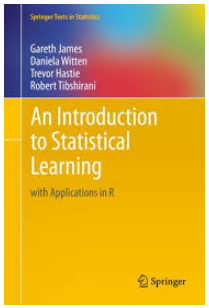
1. Identify the principal tools available in data science.
2. Import your data into R.
3. Visualize distributions of single variables and relations between several variables using ggplot2.
4. Transform and subset your data using tidyverse.
5. Understand what machine learning is and name its principal advantages and challenges.
6. Apply a machine-learning algorithm to test a hypothesis in neuroscience.
7. Harness the power of deep neural networks to quantify behavior with deeplabcut.

Supporting materials

1. R for Data Science



1. An Introduction to Statistical Learning: With Applications in R.



Both books are available online.

Conventions in the slides

Code appears as the next 3 line.

```
x <- 10
y <- 5
x + y
```

Comments start with **#**, output starts with **##**.

```
# This is a comment, for humans only
x <- 10
y <- 5
x + y
```

```
## [1] 15
```

Course online repository

The content of the course is in a GitHub repository.

https://github.com/kevin-allen/dataNeuroCourse

We will cover git and online repositories later on.

To copy the repository to your computer:

```
git clone https://github.com/kevin-allen/dataNeuroCourse
```

To update your local version, from within your local repository directory

```
git pull
```

Lecture format

1. Short introduction of topics
2. Exercises in small groups
3. Review of exercises
4. Homeworks: readings or some exercises
5. Review quizzes

It is your course, interrupt me as often as you like.

Previous experience

A short survey of the classroom.

https://tinyurl.com/y2rtpf2h

Data science landscape

Matlab

Stable environment, singal processing, matrix operations

Weakness: Proprietary programming language -> Implications in science

Python

Open-source language, general-purpose, popularity rising, TensorFlow for machine learning

R

Open-source language, statistical analysis, visualization, R packages, RStudio

Data science landscape

Recommendation: Use open-source languages (R and `python`).

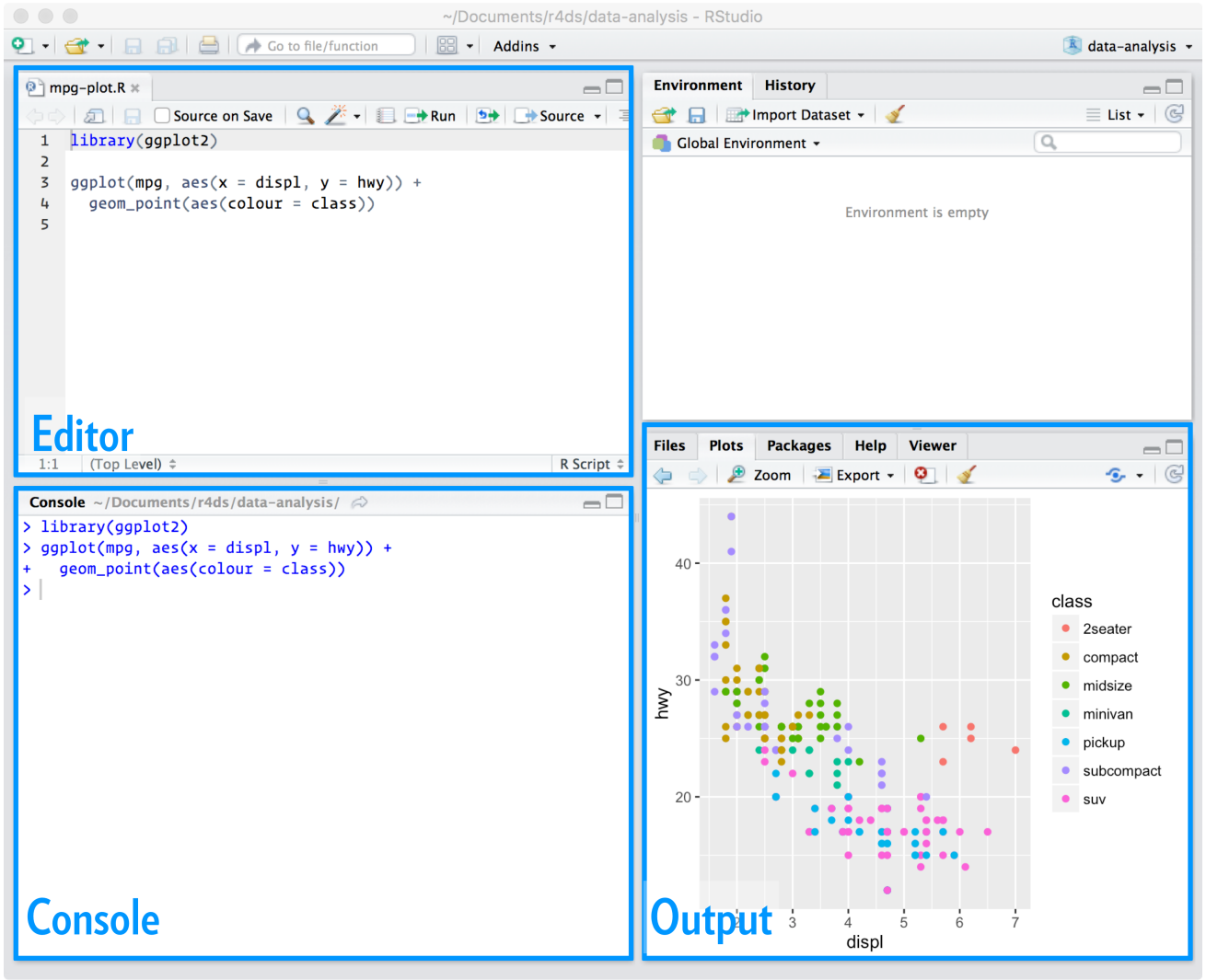
Both R and `python` are great tools for data analysis.

In the end, the most crucial point is to get the job done.

We will use mainly R.

We will use `python` for our `deeplabcut` project.

RStudio


~/Documents/r4ds/data-analysis - RStudio
Go to file/function
Addins
data-analysis
mpg-plot.R
Source on Save
Run
Source
1 library(ggplot2)
2
3 ggplot(mpg, aes(x = displ, y = hwy)) +
4 geom_point(aes(colour = class))
5
Editor
1:1
(Top Level)
R Script
Environment
History
Import Dataset
List
Global Environment
Environment is empty
Console ~/Documents/r4ds/data-analysis/
> library(ggplot2)
> ggplot(mpg, aes(x = displ, y = hwy)) +
+ geom_point(aes(colour = class))
>
Console
Files
Plots
Packages
Help
Viewer
Zoom
Export
hwy
displ
class
2seater
compact
midsize
minivan
pickup
subcompact
suv
Output

Introduction to R

Perform simple calculation as you would expect.

```
4 + 9
(360 + 50) / 2
```

Variables

Create an object to store a value using `<-`. You can display the value stored in an object by simply writing its name.

Variable names must to start with a letter and can also contain numbers and `_`.

```
x <- 10 + 50
x
```

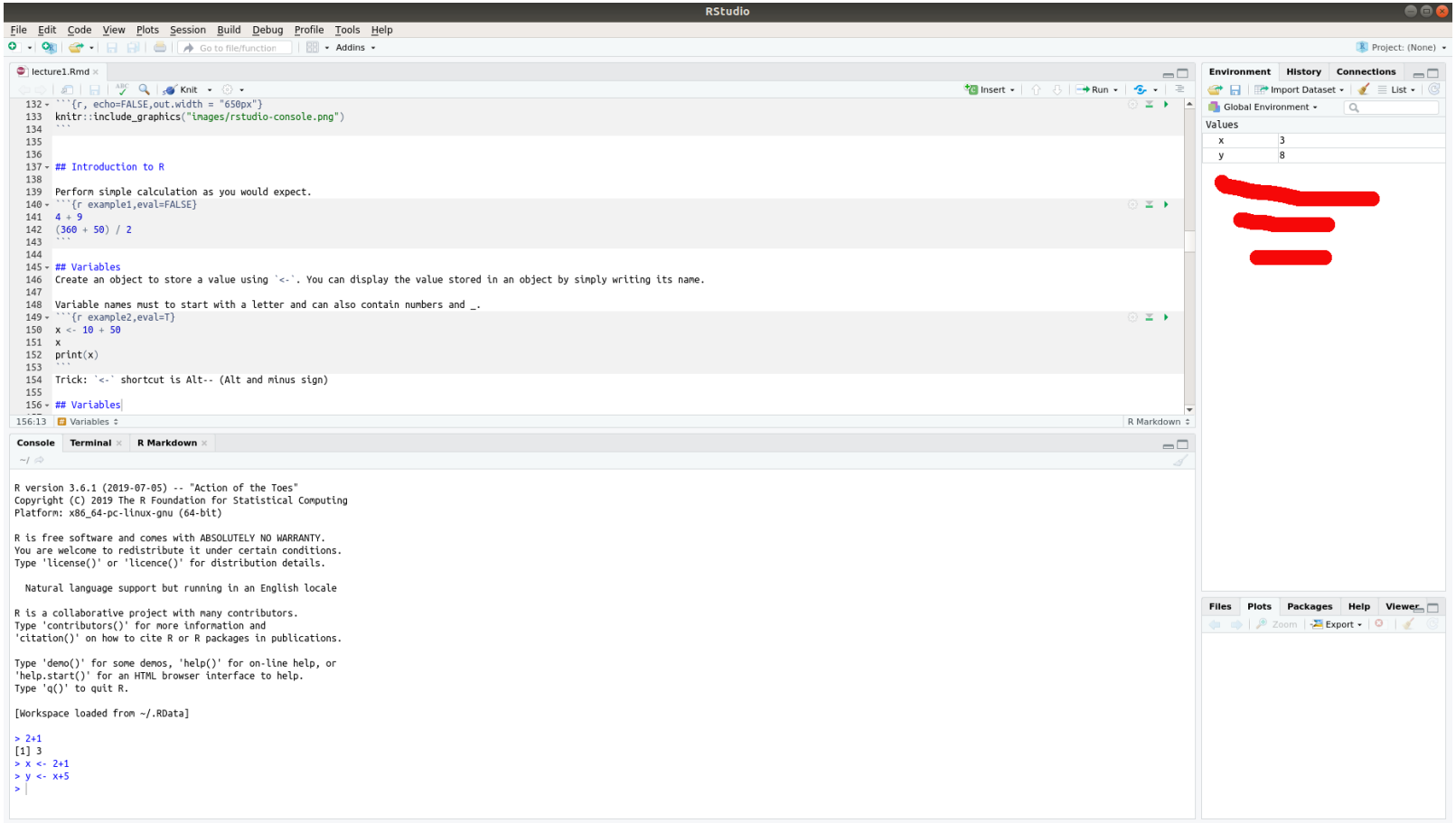
```
## [1] 60
```

```
print(x)
```

```
## [1] 60
```

Trick: `<-` shortcut is Alt– (Alt and minus sign)

Variables


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
lecture1.Rmd
```{r, echo=FALSE,out.width = "650px"}
knitr::include_graphics("images/rstudio-console.png")
```
Introduction to R
Perform simple calculation as you would expect.
```{r example1,eval=FALSE}
4 + 9
(360 + 50) / 2
```
Variables
Create an object to store a value using `<-`. You can display the value stored in an object by simply writing its name.
Variable names must to start with a letter and can also contain numbers and _.
```{r example2,eval=T}
x <- 10 + 50
x
print(x)
```
Trick: `<-` shortcut is Alt-- (Alt and minus sign)
Variables
Console Terminal R Markdown
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 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]
> 2+1
[1] 3
> x <- 2+1
> y <- x+5
Environment History Connections
Import Dataset
Global Environment
Values
x 3
y 8
Files Plots Packages Help Viewer
Export

Arithmetic operators

Operator	Description	Example	Results
+	Addition	1+1	2
-	Subtraction	5-1	4
*****	Multiplication	2*5	10
/	Division	10/5	2
^	Exponent	2^3	8
%%	Modulus (Remainder)	5%%2	1
%/%	Integer division	5%/%2	2

Relational operators

Operator	Description	Example	Results
<	Less than	1<2	TRUE
>	Greater than	5>4	TRUE
<=	Less than or equal	5<=5	TRUE
>=	Greater than or equal	3>=1	TRUE
==	Equal to	2==2	TRUE
!=	Not equal to	2!=2	FALSE

Operators and vectors

```
x <- 1:5 # vector from 1 to 5
y <- c(2,2,2,2,2) # c for concatenate
x + y
```

```
## [1] 3 4 5 6 7
```

```
x == y
```

```
## [1] FALSE  TRUE FALSE FALSE FALSE
```

```
which(x == y)
```

```
## [1] 2
```

Quizz

1. What is the difference between **=** and **==**?

Built-in functions

```
function_name(arg1 = val1, arg2 = val2)
```

```
x <- c(1, 2, 3, 6)
sin(x)
```

```
## [1]  0.8414710  0.9092974  0.1411200 -0.2794155
```

```
seq(1, 10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
sum(x)
```

```
## [1] 12
```

Your own functions

Define a function instead of copying and pasting blocks of code.

```
# define a function called addTwo
addTwo <- function(x) {
  return(x + 2)
}
# use the function
addTwo(2)
```

```
## [1] 4
```

Trick: use tab to complete names

Your own functions

```
# definition of a more useful function
fahrenheit_to_celsius <- function(temp_F) {
  temp_C <- (temp_F - 32) * 5 / 9
  return(temp_C)
}
# use it, the name explains what the function does
fahrenheit_to_celsius(32)
```

```
## [1] 0
```

Your own functions

```
# set default values of function parameters
mySum <- function(iOne=1, iTwo = 3){
  return(iOne + iTwo)
}

# use the default parameters
mySum()
```

```
## [1] 4
```

Object types

There are different types of objects (variables): numerics, characters, factors, logicals, lists, data.frames. Inspect with `class()`.

```
x
```

```
## [1] 1 2 3 6
```

```
class(x)
```

```
## [1] "numeric"
```

```
y <- "neuroscience"
class(y)
```

```
## [1] "character"
```

TRICK: Use tab to complete the name of a function in RStudio.

Data frames

Data are often organized as data frames.

```
# create a data frame
df<-data.frame(firstName=c("Sonia","Bruno","Paul"),
           height=c(170,188,150),
           weight=c(72,90,50),
           married=c(TRUE,FALSE,FALSE))
df
```

```
##   firstName height weight married
## 1     Sonia    170     72    TRUE
## 2     Bruno    188     90   FALSE
## 3      Paul    150     50   FALSE
```

Trick: Use `view(df)` to get a more excel-like look.

Data frames

```
##   firstName height weight married
## 1     Sonia    170     72    TRUE
## 2     Bruno    188     90   FALSE
## 3      Paul    150     50   FALSE
```

1. Data frames are like tables.
2. A `row` usually represents one observation (e.g., one subject).
3. A `column` often contains a variable (e.g., height).
4. A `cell` contains a single value.

TRICK: Organize your data frame with one entry per row whenever possible.

Subsetting data frames

```
# one variable
df$height
```

```
## [1] 170 188 150
```

```
# or
df[,"height"]
```

```
## [1] 170 188 150
```

Subsetting data frames

```
# one row
df[1,]
```

```
##   firstName height weight married
## 1     Sonia    170     72    TRUE
```

```
# two rows
df[c(1,2),]
```

```
##   firstName height weight married
## 1     Sonia    170     72    TRUE
## 2     Bruno    188     90   FALSE
```

Subsetting data frames

```
df[df$height>165,]
```

```
##   firstName height weight married
## 1     Sonia    170     72    TRUE
## 2     Bruno    188     90   FALSE
```

```
df[df$weight<80,]
```

```
##   firstName height weight married
## 1     Sonia    170     72    TRUE
## 3      Paul    150     50   FALSE
```

What does `df[df$married==TRUE]` do? Is it what you expected?

Get help

1. A google search: what you want to do + R
2. ?function()
3. Ask someone
4. Cheat sheets

R packages

Many useful functions are found in R packages. These are free libraries of functions.

To download and install a package from CRAN (Comprehensive R Archive Network).

```
install.packages("package_name")
```

To load the package in your R session

```
library("package_name")
```

Exercise

We will use the mpg data set that comes with tidyverse, so load tidyverse. You can get information with `?mpg`.

1. What is the mean city miles per gallon of all cars?
2. What is the maximum number of cylinders in the data set?
3. What is the range of the highway miles per gallon in the data?
4. What is the mean city miles per gallon for the first 20 cars?
5. What is the mean city miles per gallon for vehicles produced in 2008?

Solutions

```
mean(mpg$cty)
max(mpg$cyl)
range(mpg$hwy)
mean(mpg$cty[1:10])
mean(mpg$cty[mpg$year==2008])
```

Next week

I am in Magdeburg for a collaborative grant application.

Solution

We have slightly longer sessions on the 21st, 28th of November 5th of December.

To do before the next lecture

1. Install git on your computer.
2. Reading: R for Data Science, Chapters 1 and 2.

Next lecture

1. Review exercises (15 minutes)
2. RStudio editor and .R files. (10 minutes)
3. git repositories (15 minutes)
4. Introduction to ggplot2 (30 minutes)
5. Introduction to dplyr (30 minutes)
6. Example: Analysis of behavioral data (30 minutes)