

# Biodiversity data analysis workshop – Day 2

---

Dominic Henry & Lizanne Roxburgh  
Conservation Science Unit

dominich@ewt.org.za  
lizanner@ewt.org.za



**ENDANGERED  
WILDLIFE TRUST**  
Protecting forever, together.



**science  
& technology**

Department:  
Science and Technology  
REPUBLIC OF SOUTH AFRICA



**National  
Research  
Foundation**



South African National Biodiversity Institute



ENDANGERED  
WILDLIFE TRUST

# Workshop goals

---

- Introduce foundational concepts
- Provide insight into the potential applications of R
- Overcome the first hurdles of using R
- Provide a solid foundation for data wrangling
- Introduce you to the R community
- Start you on your R journey & learn by doing

# Programme – Day 2

---



ENDANGERED  
WILDLIFE TRUST

08h30 – 10h00	Session 1 – Introduction to R, RStudio, basics of programming
10h00 – 10h30	Tea break
10h30 – 12h15	Session 2 – Data wrangling with the tidyverse
12h15 – 13h30	Lunch
13h30 – 15h00	Session 3 – Data visualisation using ggplot2
15h00 – 15h30	Tea break
15h30 – 17h00	Session 4 – Handling spatial data in R



ENDANGERED  
WILDLIFE TRUST

# What is R?

---

- A high-level functional programming language
- Open source (free!) software environment
- A diverse community
- Driven by the *R Development Core Team*

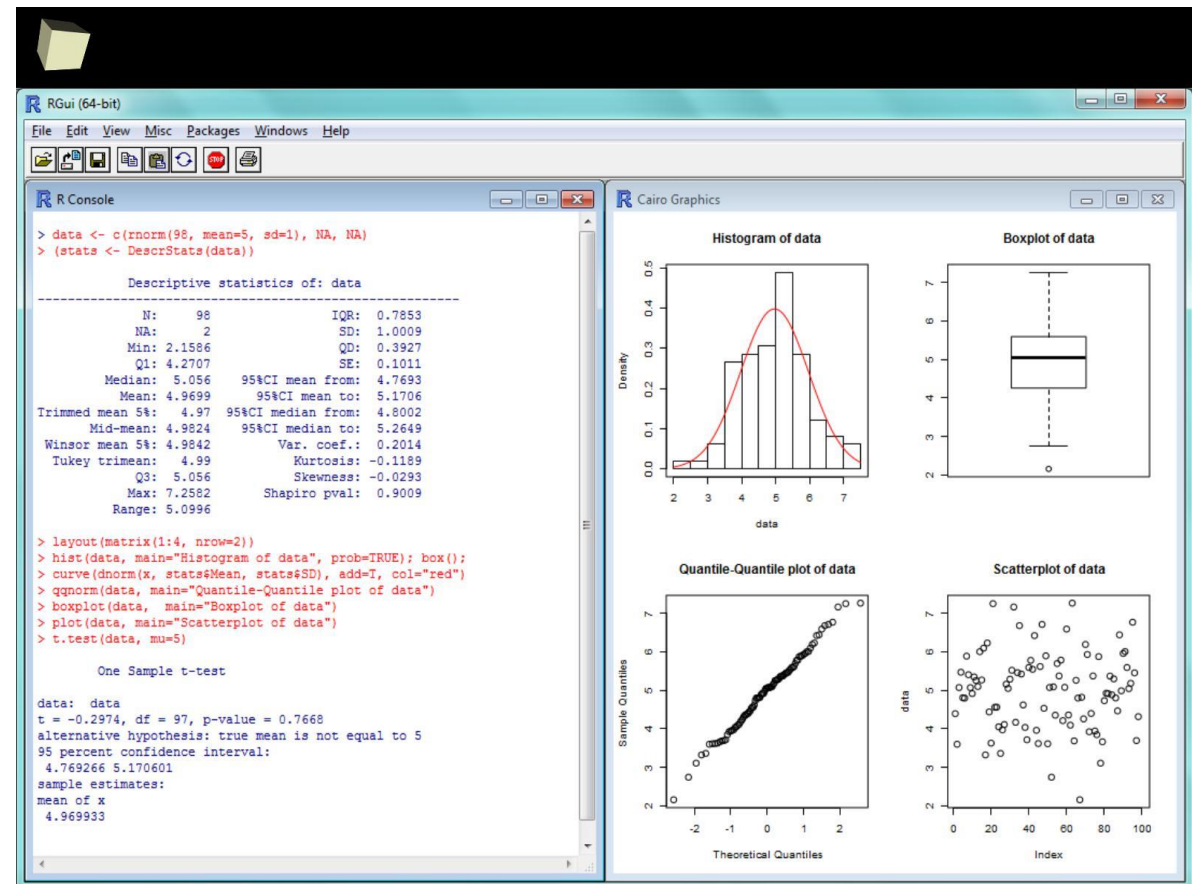
[www.r-project.org](http://www.r-project.org)

# What can R be used for?



ENDANGERED  
WILDLIFE TRUST

- Statistical programming





ENDANGERED  
WILDLIFE TRUST

# What can R be used for?

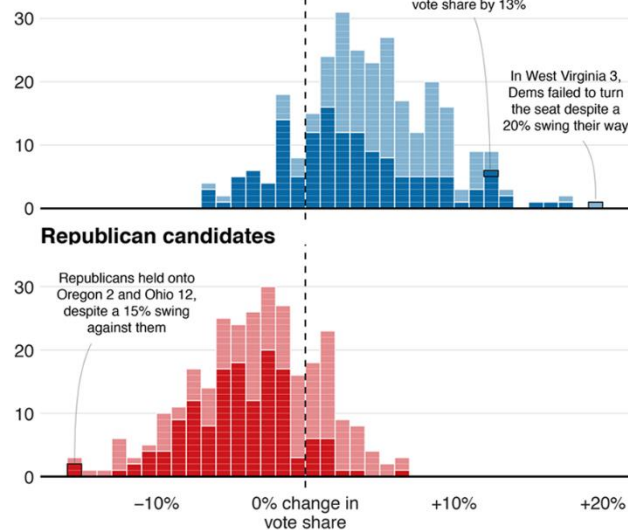
- Statistical programming
- Data wrangling and visualisation



## Blue wave

■ Won seat ■ Didn't win

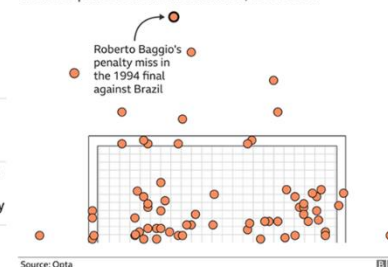
### Democrat candidates



Source: AP, 19:01 ET

## Where penalties are saved

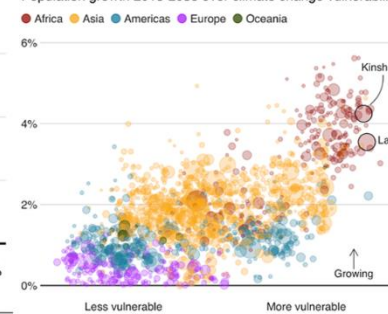
World Cup shootout misses and saves, 1982-2014



Source: Opta

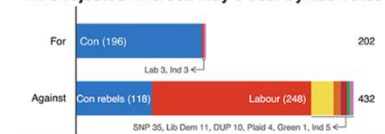
## Fast-growing cities face worse climate risks

Population growth 2018-2035 over climate change vulnerability



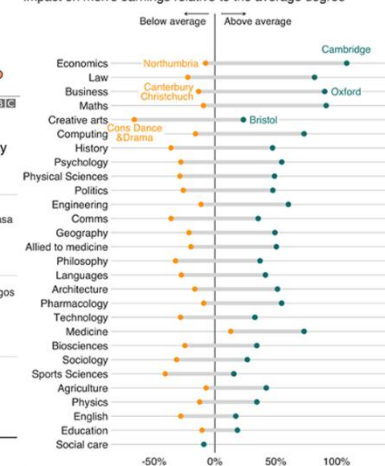
Source: Verisk Maplecroft. Circle size represents current population.

## MPs rejected Theresa May's deal by 230 votes



Source: Commons Votes Services. Excludes 'tellers', the Speaker and deputies

## Earnings vary across units even within subjects



Source: Institute for Fiscal Studies



ENDANGERED  
WILDLIFE TRUST

# What can R be used for?

---

- Statistical programming
- Data wrangling and visualisation
- Web apps





ENDANGERED  
WILDLIFE TRUST

# What can R be used for?

---

- Statistical programming
- Data wrangling and visualisation
- Web apps
- Spatial analysis







ENDANGERED  
WILDLIFE TRUST

# What can R be used for?

---

- Statistical programming
- Data wrangling and visualisation
- Web apps
- Spatial analysis
- Data science



ENDANGERED  
WILDLIFE TRUST

# What can R be used for?

---

- Statistical programming
- Data wrangling and visualisation
- Web apps
- Spatial analysis
- Data science
- Almost anything...



# Why should we use R?

---

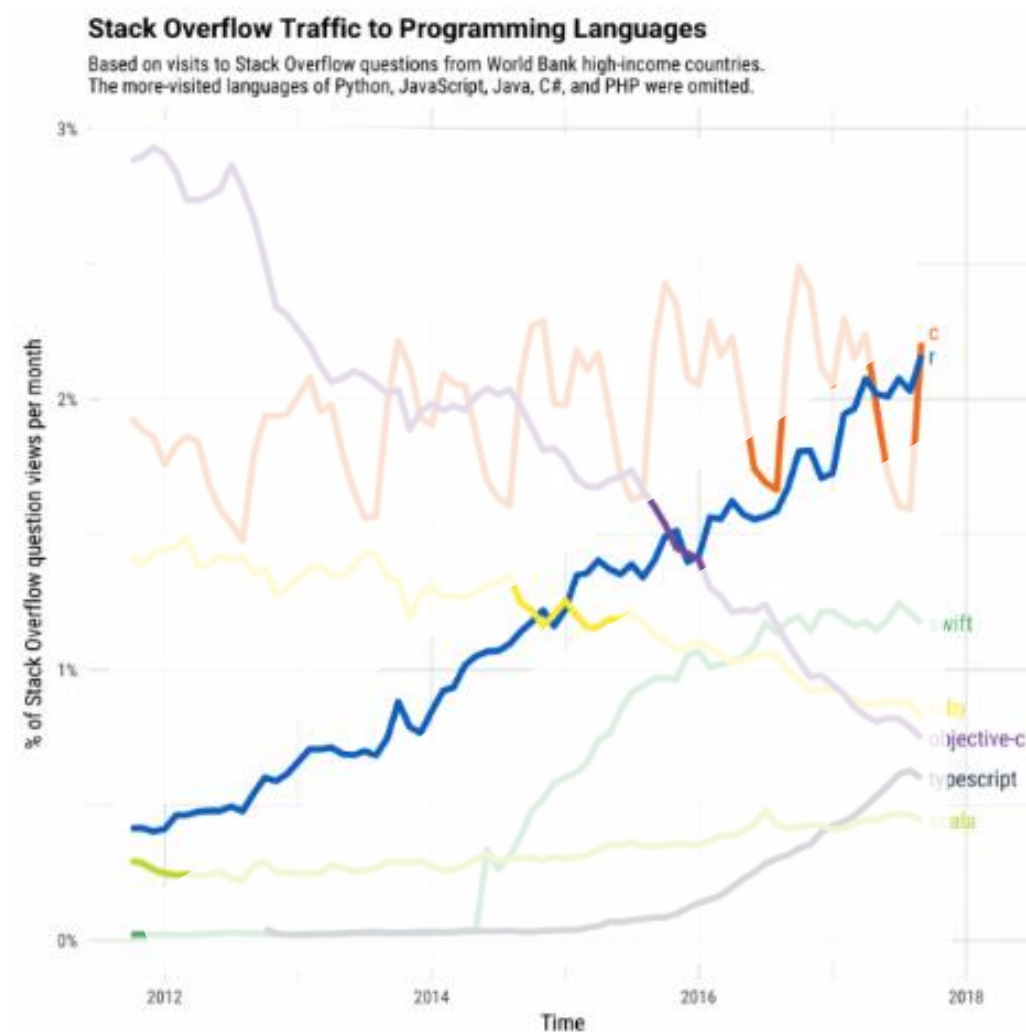
- Repetitive manual tasks are boring, time-consuming and prone to errors
- Allows for reproducible and transferable research, processes and analysis
- Free with incredible user support
- Unlimited (almost) capabilities
- It is a valuable career skill
- It will save you time!



# Why should we use R?



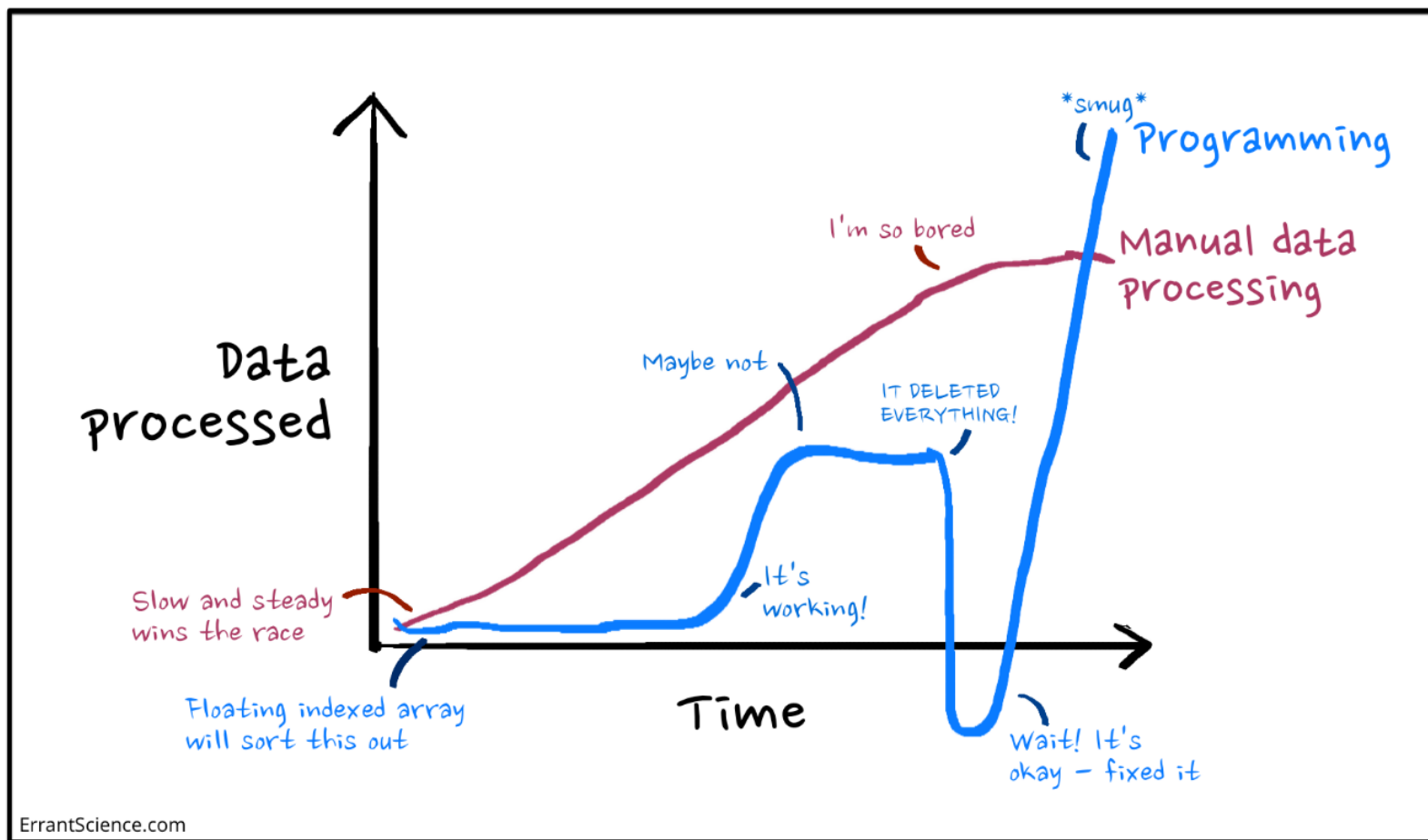
ENDANGERED  
WILDLIFE TRUST



# Is it worth the effort?



ENDANGERED  
WILDLIFE TRUST



Programming vs manual data processing



ENDANGERED  
WILDLIFE TRUST

# What is RStudio?

---

- Integrated development environment (IDE) for R
- Most popular IDE, free and open-source
- Enhances extends the core capabilities of R
- Provides an easy way to write & save scripts, interact with objects, visualise results and publish end products



# Layout of RStudio



ENDANGERED  
WILDLIFE TRUST

The screenshot displays the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar. The main workspace is divided into four panels:

- Source:** The top-left panel shows a script editor with a line of code: `Source`. Below the code, a text box explains: "Scripts are recipes (records of how to do things). Write and save recipes here so R knows how to cook."
- Environment:** The top-right panel shows the "Global Environment" with a search bar. Below the search bar, a text box explains: "Kitchen counter where you can put ingredients (data) and finished meals (model outputs) while you cook".
- Console:** The bottom-left panel shows the R console output. It displays the R version (3.5.1) and the R Foundation logo. Below the logo, a text box explains: "Where the cooking happens – send recipes here (run code) to cook them. You can cook without recipes but you'll struggle to remember how to make the dish in future."
- Files:** The bottom-right panel shows a file explorer view of the "C:\Users\DominicH\Google Drive" directory. It lists several files, including "Ecology and Conservation", "Functional programming.R", and "SAFON". Below the file list, a text box explains: "Ingredients in the cupboard (you need to get them into the kitchen counter to use them)".

# Layout of RStudio



ENDANGERED  
WILDLIFE TRUST

The screenshot displays the RStudio interface with the following components:

- Menu Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations (new, open, save, print), navigation (Go to file/function), and execution (Run, Source).
- Source Editor:** Displays the script `first analysis.R` with the following code:

```
1  
2 ## Read in the data  
3 birds <- read.csv("Day 2 - S2/data/bird_counts.csv")  
4  
5 ## Remove the first row of data  
6 birds <- birds[-1, ]  
7 |  
8 ## Check the number of rows  
9 nrow(birds)
```
- Console:** Shows the output of the executed code:

```
> birds <- read.csv("Day 2 - S2/data/bird_counts.csv")  
> birds <- birds[-1, ]  
> nrow(birds)  
[1] 475  
> |
```
- Environment Pane:** Shows the **Global Environment** with a search bar. Under the **Data** tab, it lists `birds` with 475 observations and 29 variables.
- Files Pane:** Shows the file explorer for the project `project > Day 2 - S2 > data`. It contains a table of files:

	Name	Size	Modified
	..		
<input type="checkbox"/>	bird_counts.csv	64.1 KB	Oct 17, 2019, 1:50 PM
<input type="checkbox"/>	birds.csv	62.4 KB	Oct 17, 2019, 1:50 PM
<input type="checkbox"/>	roadkill_data.csv	3.9 MB	Oct 17, 2019, 1:50 PM
<input type="checkbox"/>	first analysis.R	169 B	Oct 23, 2019, 12:06 PM



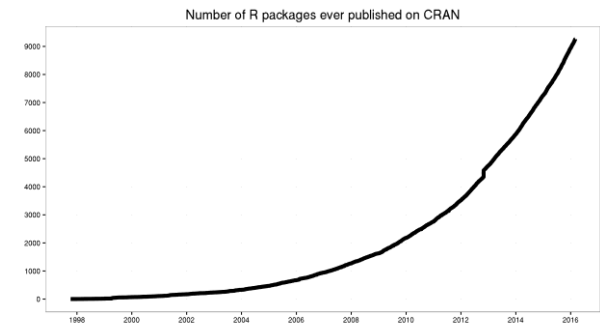
# R packages

---



ENDANGERED  
WILDLIFE TRUST

- Base R
- The building blocks of the R user experience
- Consists of functions & objects developed by members of the R community
- Completely free to use
- Download and install in a single line:  
`install.packages("tidyverse")`
- Load before starting each session: `library(tidyverse)`



# R package repositories

---



ENDANGERED  
WILDLIFE TRUST

CRAN



<https://cran.r-project.org/>

GitHub



<https://github.com/>

# R package vignettes

---



ENDANGERED  
WILDLIFE TRUST

- An in-depth guide to a package
- Can describe the problem that a package is designed to solve, and then show the reader how to solve it
- A vignette divides functions into useful categories, and demonstrate how to coordinate multiple functions to solve problems

# sf package vignette



ENDANGERED  
WILDLIFE TRUST

## What is a feature?

Dimensions
Simple feature geometry types
Coordinate reference system
How simple features in R are organized
sf: objects with simple features
sfc: simple feature geometry list-column
Mixed geometry types
sfg: simple feature geometry
Well-known text, well-known binary, precision
Reading and writing
Coordinate reference systems and transformations
Conversion, including to and from sp
Geometrical operations
Non-simple and non-valid geometries

## 1. Simple Features for R

[Simple features](#) or [simple feature access](#) refers to a formal standard (ISO 19125-1:2004) that describes how objects in the real world can be represented in computers, with emphasis on the *spatial* geometry of these objects. It also describes how such objects can be stored in and retrieved from databases, and which geometrical operations should be defined for them.

The standard is widely implemented in spatial databases (such as PostGIS), commercial GIS (e.g., [ESRI ArcGIS](#)) and forms the vector data basis for libraries such as [GDAL](#). A subset of simple features forms the [GeoJSON](#) standard.

R has well-supported classes for storing spatial data ([sp](#)) and interfacing to the above mentioned environment has so far lacked a complete implementation of simple features, making conversions at times convoluted, inef. The package [sf](#) tries to fill this gap, and aims at succeeding [sp](#) in the long term.

This vignette:

- explains what is meant by features, and by simple features
- shows how they are implemented in R
- provides examples of how you can work with them
- shows how they can be read from and written to external files or resources
- discusses how they can be converted to and from sp objects
- shows how they can be used for meaningful spatial analysis



## What is a feature?

A feature is thought of as a thing, or an object in the real world, such as a building or a tree. As is the case with objects, they often consist of other objects. This is the case with features too: a set of features can form a single feature. A forest stand can be a feature, a forest can be a feature, a city can be a feature. A satellite image pixel can be a feature, a complete image can be a feature too.

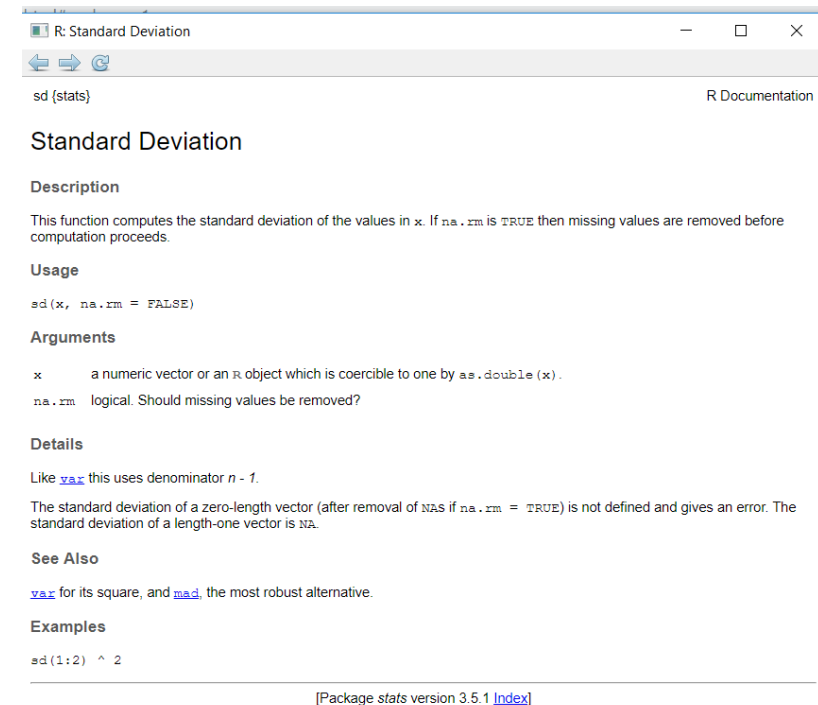
Features have a *geometry* describing *where* on Earth the feature is located, and they have attributes, which describe other properties. The geometry of a tree can be the delineation of its crown, of its stem, or the point indicating its centre. Other properties may include its height, color, diameter at breast height at a particular date, and so on.



# Getting help in R

---

- Help page for each function
- Divided into sections in a standardised format
- Access help: `?sd` or `F1`



# Getting help outside of R

---



<https://stackoverflow.com/>



<https://community.rstudio.com/>

# Getting help outside of R



ENDANGERED  
WILDLIFE TRUST

The image displays three screenshots from the Stack Overflow website, illustrating resources for getting help outside of R.

**Top Left Screenshot: Stack Overflow Tags Page**

- Search bar: [r]
- Navigation: Home, PUBLIC, Stack Overflow, Tags, Users, Jobs, Teams Q&A for work (Learn More).
- Tags Section: A tag is a keyword or label that categorizes your question with other tags. Search for `r`. Statistics: `r` × 289562, `javascript` × 180940.
- Featured Tag: `r` × 132.2k watchers, 289.6k questions. Description: R is a free, open-source programming language and software environment for statistical computing, bioinformatics, visualization, and general computing. Please provide minimal and reproducible example(s) along with the desired output. Use `dput()` for data and specify all non-base packages with `library()` calls. Do not embed pictures for data or code, use indented code blocks instead. For statistics related questions, use <https://stats.stackexchange.com>. View tag.
- Buttons: Frequent, Votes, Active, Unanswered.

**Top Right Screenshot: Stack Overflow Question**

- Search bar: [ ]
- Navigation: Home, PUBLIC, Stack Overflow, Tags, Users, Jobs, Teams Q&A for work (Learn More).
- Question Title: How to make a great R reproducible example
- Views: 2476
- Content: When discussing performance with colleagues, teaching, sending a bug report or searching for guidance on mailing lists and here on Stack Overflow, a [reproducible example](#) is often asked and always helpful. What are your tips for creating an excellent example? How do you paste data structures from `r` in a text format? What other information should you include? Are there other tricks in addition to using `dput()`, `dump()` or `structure()`? When should you include `library()` or `require()` statements? Which reserved words should one avoid, in addition to `c`, `df`, `data`, etc.? How does one make a great `r` reproducible example?
- Tags: `r`, `r-faq`
- Buttons: share, edited Aug 19 '18 at 17:12, community wiki (15 revs, 10 users 41% Hack-R)

**Bottom Right Screenshot: Stack Overflow Question**

- Search bar: [ ]
- Navigation: Home, PUBLIC, Stack Overflow, Tags, Users, Jobs, Teams Q&A for work (Learn More).
- Question Title: 23 Answers
- Views: 1595
- Content: A minimal reproducible example consists of the following items:
  - a minimal dataset, necessary to reproduce the error
  - the minimal **runnable** code necessary to reproduce the error, which can be run on the given dataset.
  - the necessary information on the used packages, R version, and system it is run on.
  - in the case of random processes, a seed (set by `set.seed()`) for reproducibilityLooking at the examples in the help files of the used functions is often helpful. In general, all the code given there fulfills the requirements of a minimal reproducible example: data is provided, minimal code is provided, and everything is runnable.
- Section: Producing a minimal dataset
- Text: For most cases, this can be easily done by just providing a vector/data frame with some values. Or you can use one of the built-in datasets, which are provided with most packages. A comprehensive list of built-in datasets can be seen with `library(help = "datasets")`. There is a short description to every dataset and more information can be obtained for example with `?mtcars` where `'mtcars'` is one of the datasets in the list. Other packages might contain additional datasets.

# R projects

---



ENDANGERED  
WILDLIFE TRUST

- Basis for all workflows and analyses
- Best practice – relative directories and file paths
- Makes code and data transferable
- Create skeleton folder structures

Two specific slides generated much discussion and consternation in #rstats Twitter:

*If the first line of your R script is*

```
setwd("C:\\Users\\jenny\\path\\that\\only\\I\\have")
```

*I will come into your office and SET YOUR COMPUTER ON FIRE 🔥.*

*If the first line of your R script is*

```
rm(list = ls())
```

*I will come into your office and SET YOUR COMPUTER ON FIRE 🔥.*

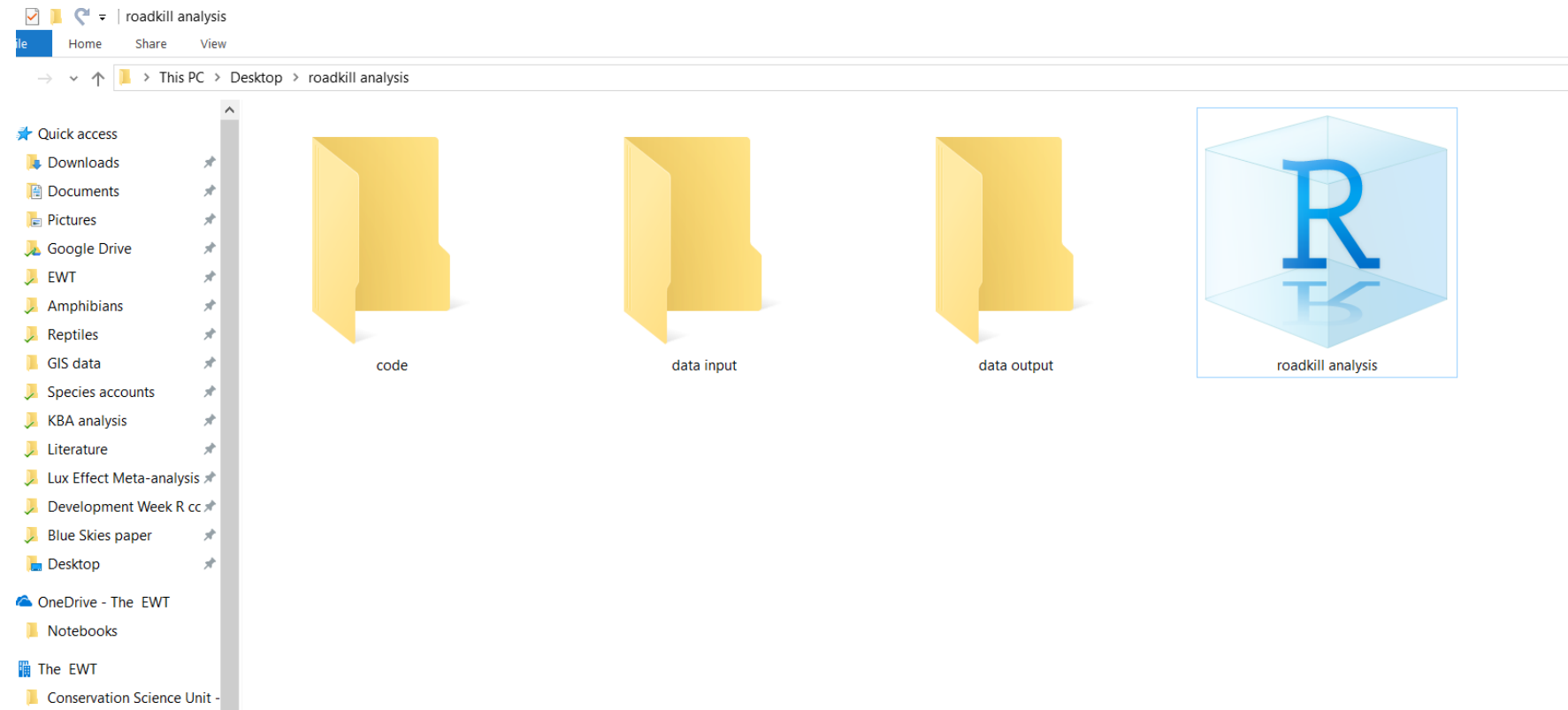


# R projects



ENDANGERED  
WILDLIFE TRUST

## Step 1: Create folder and subfolders



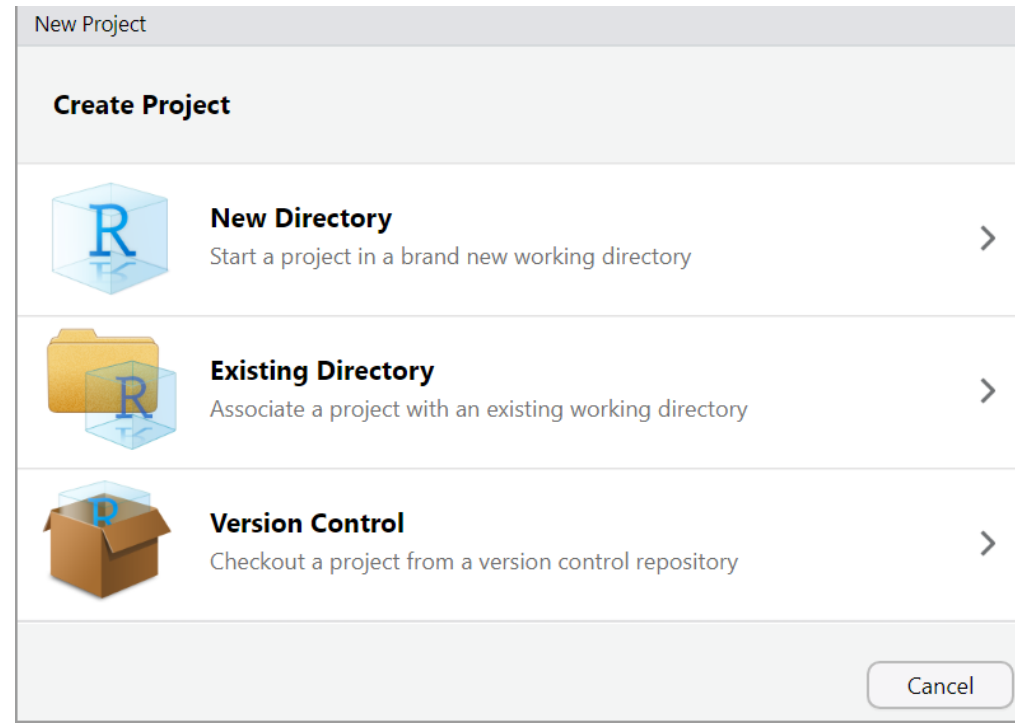
# R projects

---



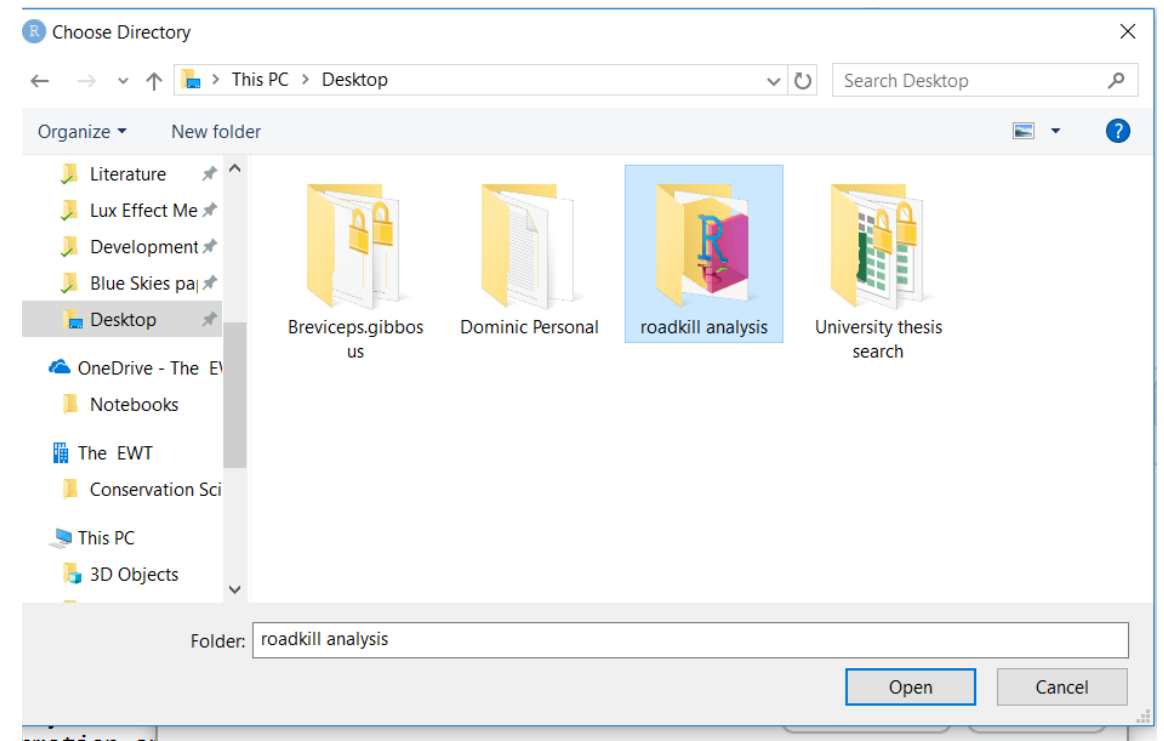
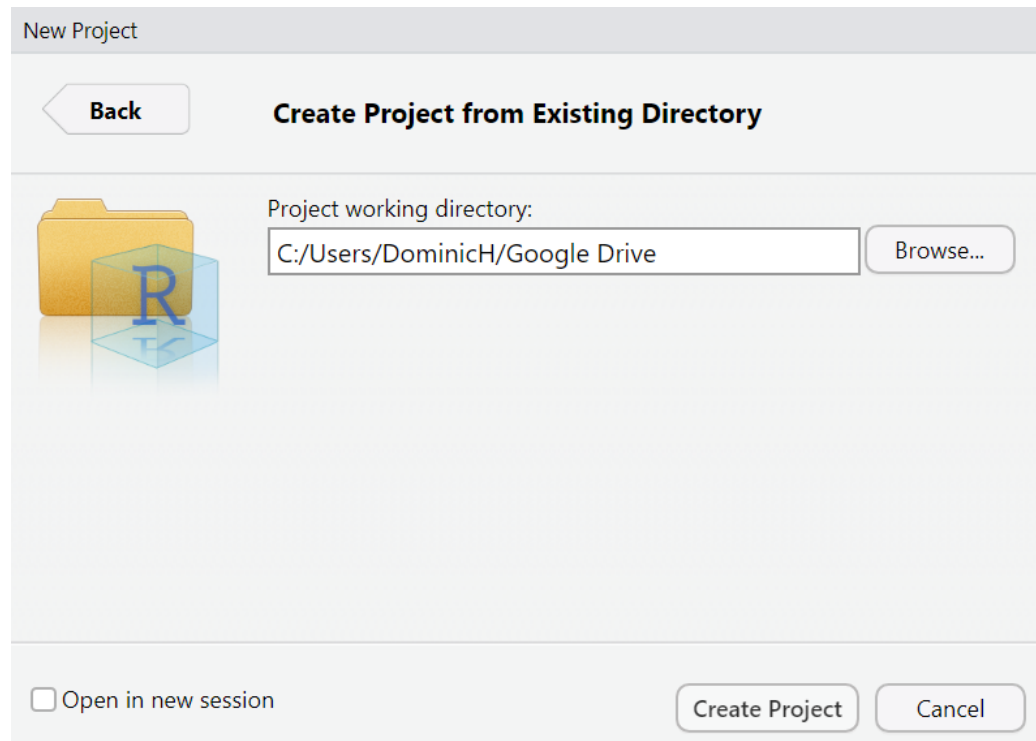
ENDANGERED  
WILDLIFE TRUST

Step 2: Open RStudio, Select *File -> New Project*



# R projects

## Step 3: Select *Existing directory*



# RStudio Cloud

---



ENDANGERED  
WILDLIFE TRUST

- Demo
- Explanation of assignments
- Interactive introductory tutorials (spare time)

# R basics

---



ENDANGERED  
WILDLIFE TRUST

- Rmd file link