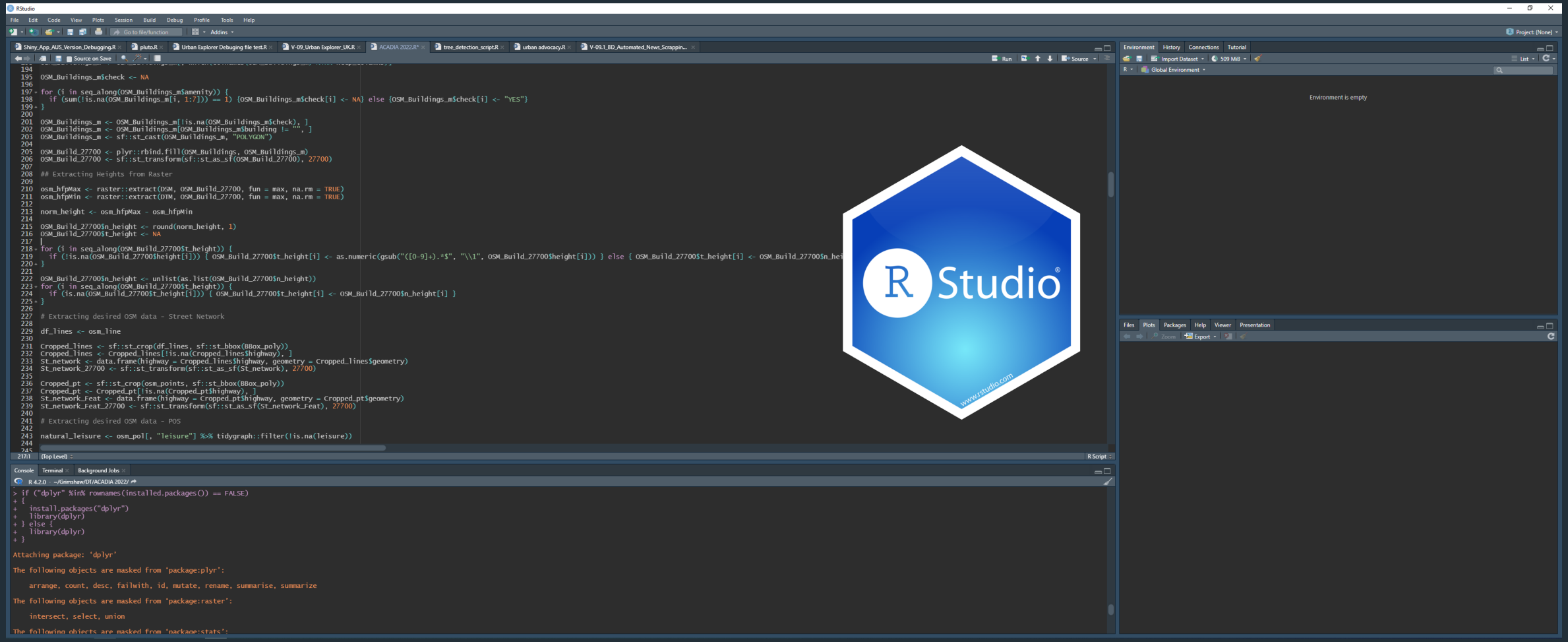
ACADIA 2022

A data-driven approach for urban design and master planning development

Jorge Sainz de Aja Curbelo – Esther Rubio Madronal

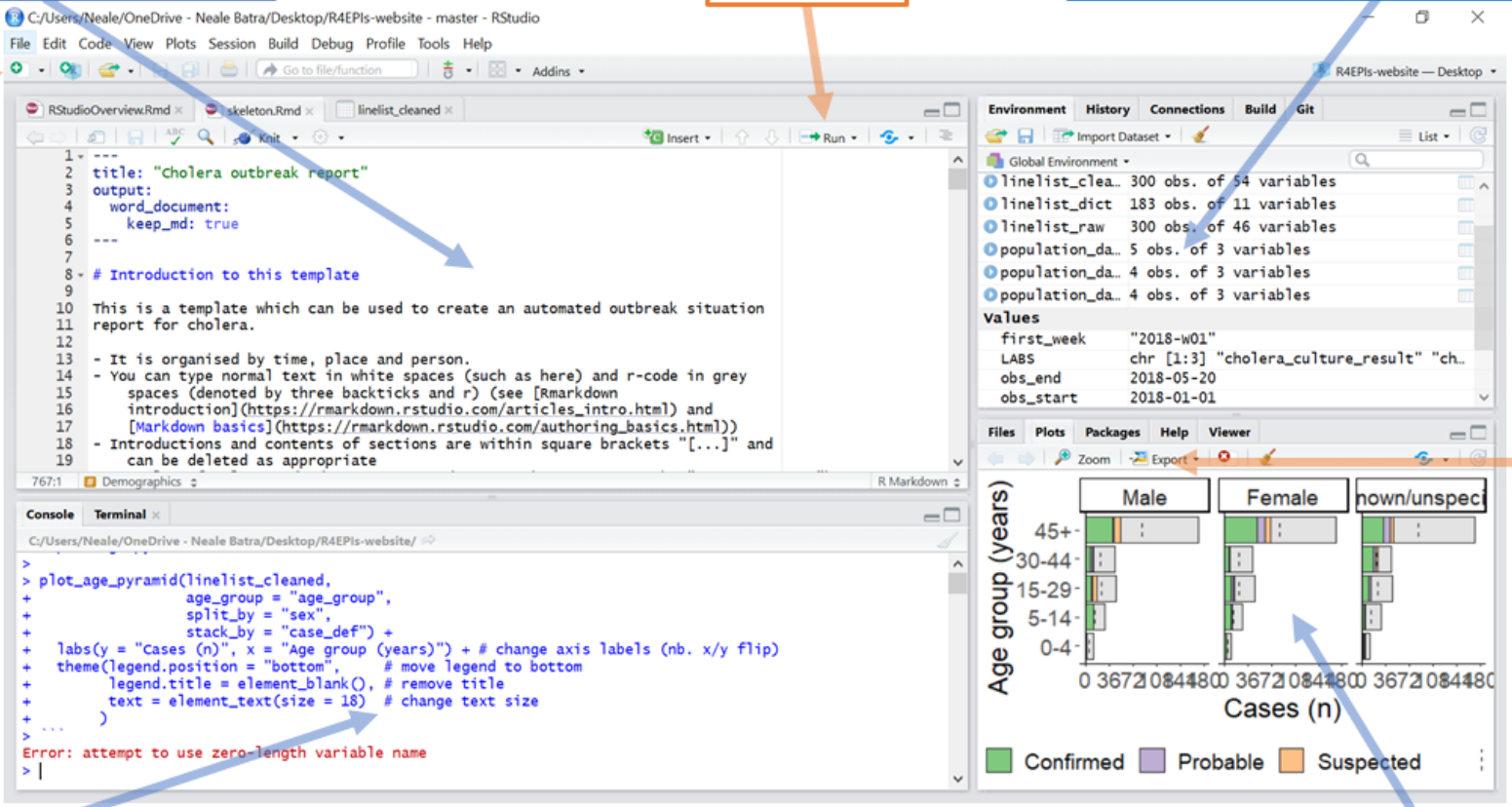

RStudio
Shiny_App_AUS_Version_Debugging.R
pluto.R
Urban Explorer Debuging file test.R
V-09_Urban Explorer_UK.R
ACADIA 2022.R*
tree_detection_script.R
urban advocacy.R
V-09.1_BD_Automated_News_Scrappin...
OSM_Buildings_m$check <- NA
for (i in seq_along(OSM_Buildings_m$amenity)) {
if (sum(!is.na(OSM_Buildings_m[i, 1:7])) == 1) {OSM_Buildings_m$check[i] <- NA} else {OSM_Buildings_m$check[i] <- "YES"}
}
OSM_Buildings_m <- OSM_Buildings_m[!is.na(OSM_Buildings_m$check),]
OSM_Buildings_m <- OSM_Buildings_m[OSM_Buildings_m$building != "",]
OSM_Buildings_m <- sf::st_cast(OSM_Buildings_m, "POLYGON")
OSM_Build_27700 <- plyr::rbind.fill(OSM_Buildings, OSM_Buildings_m)
OSM_Build_27700 <- sf::st_transform(sf::st_as_sf(OSM_Build_27700), 27700)
Extracting Heights from Raster
osm_hfpMax <- raster::extract(DSM, OSM_Build_27700, fun = max, na.rm = TRUE)
osm_hfpMin <- raster::extract(DTM, OSM_Build_27700, fun = max, na.rm = TRUE)
norm_height <- osm_hfpMax - osm_hfpMin
OSM_Build_27700$n_height <- round(norm_height, 1)
OSM_Build_27700$t_height <- NA
for (i in seq_along(OSM_Build_27700$t_height)) {
if (!is.na(OSM_Build_27700$height[i])) { OSM_Build_27700$t_height[i] <- as.numeric(gsub("([0-9]+).*$", "\\1", OSM_Build_27700$height[i])) } else { OSM_Build_27700$t_height[i] <- OSM_Build_27700$n_hei
}
OSM_Build_27700$n_height <- unlist(as.list(OSM_Build_27700$n_height))
for (i in seq_along(OSM_Build_27700$t_height)) {
if (is.na(OSM_Build_27700$t_height[i])) { OSM_Build_27700$t_height[i] <- OSM_Build_27700$n_height[i] }
}
# Extracting desired OSM data - Street Network
df_lines <- osm_line
Cropped_lines <- sf::st_crop(df_lines, sf::st_bbox(BBox_poly))
Cropped_lines <- Cropped_lines[!is.na(Cropped_lines$highway),]
St_network <- data.frame(highway = Cropped_lines$highway, geometry = Cropped_lines$geometry)
St_network_27700 <- sf::st_transform(sf::st_as_sf(St_network), 27700)
Cropped_pt <- sf::st_crop(osm_points, sf::st_bbox(BBox_poly))
Cropped_pt <- Cropped_pt[!is.na(Cropped_pt$highway),]
St_network_Feat <- data.frame(highway = Cropped_pt$highway, geometry = Cropped_pt$geometry)
St_network_Feat_27700 <- sf::st_transform(sf::st_as_sf(St_network_Feat), 27700)
# Extracting desired OSM data - POS
natural_leisure <- osm_pol[, "leisure"] %>% tidygraph::filter(!is.na(leisure))
Environment
History
Connections
Tutorial
Environment is empty
Files
Plots
Packages
Help
Viewer
Presentation
Console
Terminal
Background Jobs
R 4.2.0 · ~/Grimshaw/DT/ACADIA 2022/
> if ("dplyr" %in% rownames(installed.packages()) == FALSE)
+ {
+ install.packages("dplyr")
+ library(dplyr)
+ } else {
+ library(dplyr)
+ }
Attaching package: 'dplyr'
The following objects are masked from 'package:plyr':
arrange, count, desc, failwith, id, mutate, rename, summarise, summarize
The following objects are masked from 'package:raster':
intersect, select, union
RStudio
www.rstudio.com

Source Pane
Edit and run scripts (e.g. Rmarkdown templates), and view datasets

Tip: Run script

Environment Pane
Overview of objects (datasets, parameters, lists, etc.) you have imported or created.

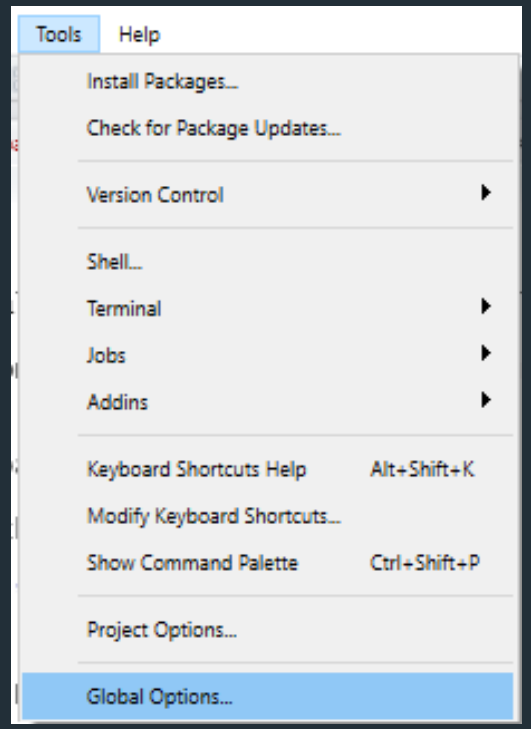
Tip:
Start new script

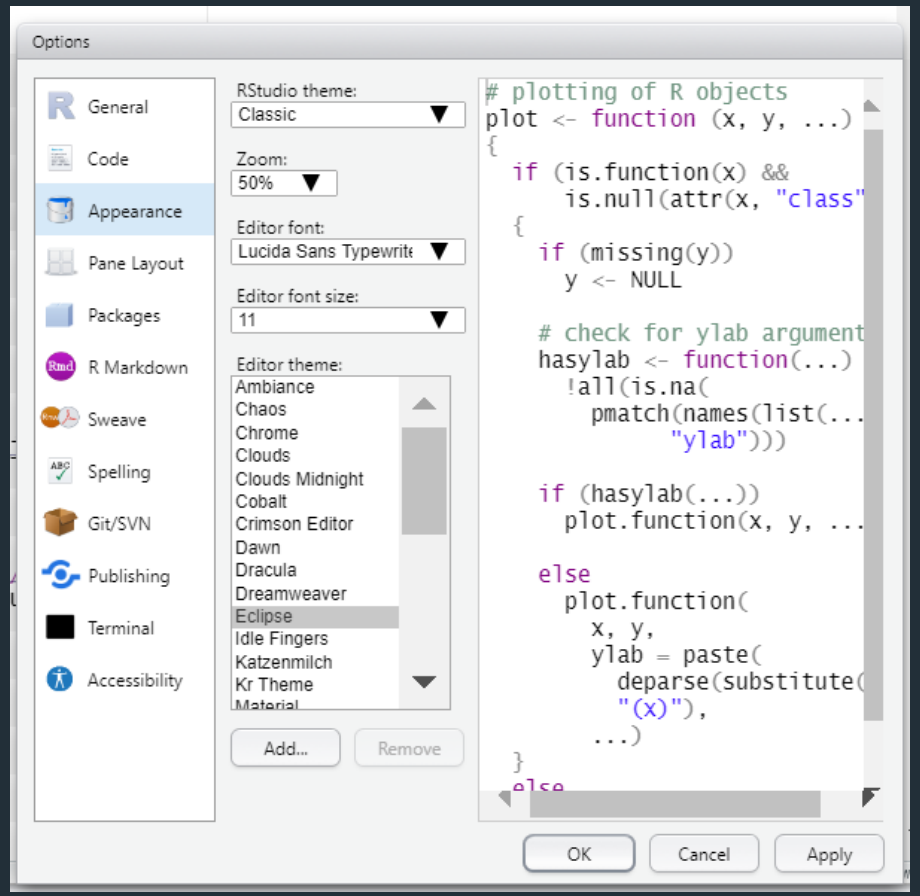


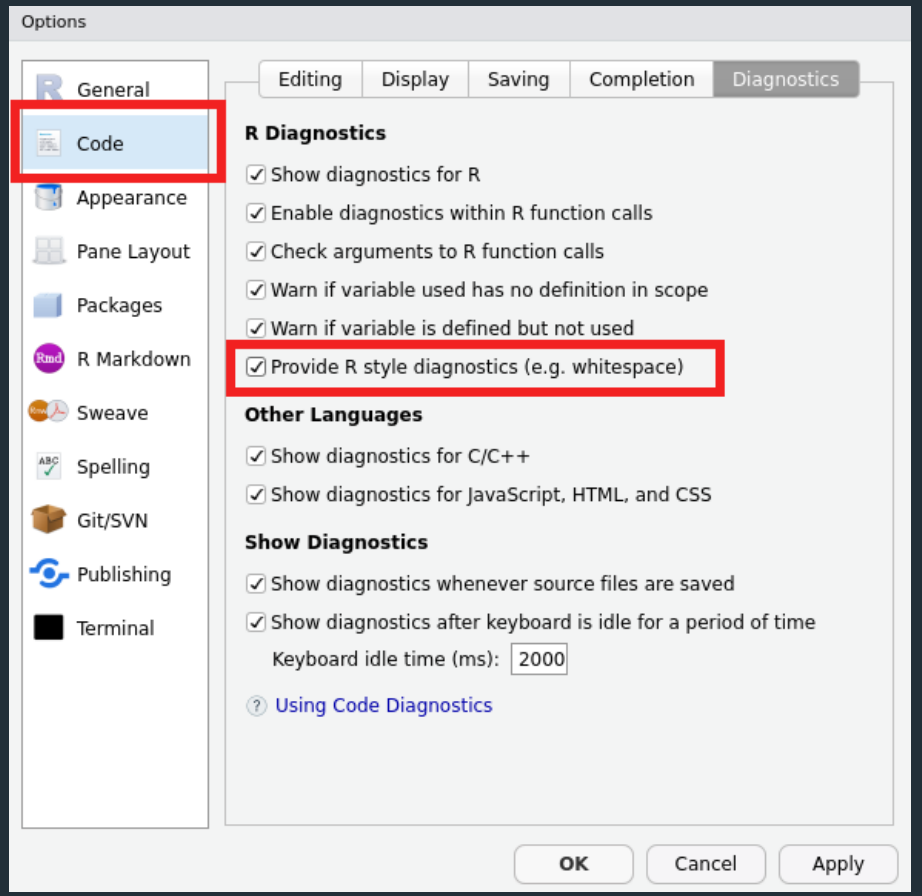
Tip: Zoom and export plots

R Console Pane
R commands run are shown here, and non-graphic output and errors are displayed

Plots, Packages, and Help Pane
Commonly used to view graphics, install packages, and view help


Tools
Help
Install Packages...
Check for Package Updates...
Version Control
Shell...
Terminal
Jobs
Addins
Keyboard Shortcuts Help Alt+Shift+K
Modify Keyboard Shortcuts...
Show Command Palette Ctrl+Shift+P
Project Options...
Global Options...

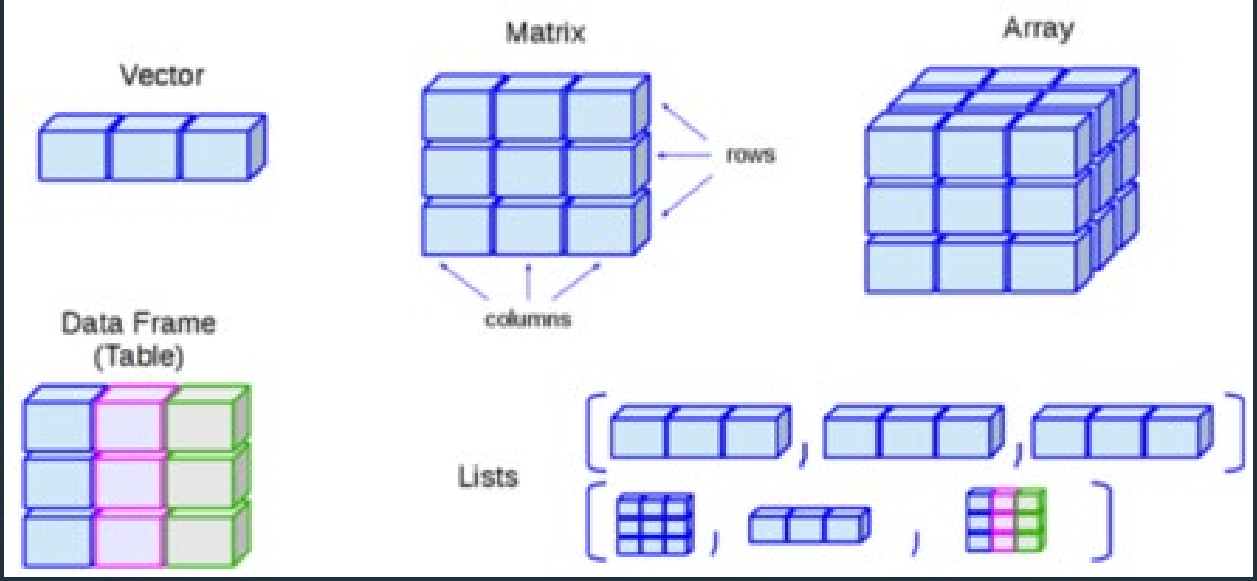

Options
General
Code
Appearance
Pane Layout
Packages
R Markdown
Sweave
Spelling
Git/SVN
Publishing
Terminal
Accessibility
RStudio theme:
Classic
Zoom:
50%
Editor font:
Lucida Sans Typewrite
Editor font size:
11
Editor theme:
Ambiance
Chaos
Chrome
Clouds
Clouds Midnight
Cobalt
Crimson Editor
Dawn
Dracula
Dreamweaver
Eclipse
Idle Fingers
Katzenmilch
Kr Theme
Material
Add...
Remove
# plotting of R objects
plot <- function (x, y, ...)
{
if (is.function(x) &&
is.null(attr(x, "class"
{
if (missing(y))
y <- NULL
# check for ylab argument
hasylab <- function(...)
!all(is.na(
pmatch(names(list(...
"ylab")))
if (hasylab(...))
plot.function(x, y, ...
else
plot.function(
x, y,
ylab = paste(
deparse(substitute(
"(x)"),
...)
}
else
OK
Cancel
Apply


Options
General
Code
Appearance
Pane Layout
Packages
R Markdown
Sweave
Spelling
Git/SVN
Publishing
Terminal
Editing
Display
Saving
Completion
Diagnostics
R Diagnostics
Show diagnostics for R
Enable diagnostics within R function calls
Check arguments to R function calls
Warn if variable used has no definition in scope
Warn if variable is defined but not used
Provide R style diagnostics (e.g. whitespace)
Other Languages
Show diagnostics for C/C++
Show diagnostics for JavaScript, HTML, and CSS
Show Diagnostics
Show diagnostics whenever source files are saved
Show diagnostics after keyboard is idle for a period of time
Keyboard idle time (ms): 2000
Using Code Diagnostics
OK
Cancel
Apply

https://epirhandbook.com/en/r-basics.html


```
Bread <- dinner::eat(pastry::decorate(cooker::oven_bake(...), ...), ...)
```


```
Bread <- cooker::oven_bake(...) %>% pastry::decorate(...) %>% dinner::eat(...)
```



id	Fruits	Dairies	Meat
1	apple	milk	NA
2	banana	cheese	NA
3	strawberry	cream	NA

Fruits <- c("apple", "banana", "strawberry",...) VECTOR

Groceries <- data.frame(Fruits = Fruits, Dairies = c("milk", "cheese", "cream"), Meat = NA) DATA FRAME

```
> Groceries <- data.frame(Fruits = c("apple", "banana", "strawberry"), Dairies = c("milk", "cheese", "cream"), Meat = NA)
> Groceries[1, 2]
[1] "milk"
> Groceries[1, "Dairies"]
[1] "milk"
> Groceries$Dairies[1]
[1] "milk"
> Groceries$Dairies
[1] "milk"   "cheese" "cream"
> |
```

Class	Explanation	Examples
Character	These are text/words/sentences **"within quotation marks"**. Math cannot be done on these objects.	"Character objects are in quotation marks"
Integer	Numbers that are **whole only** (no decimals)	-5, 14, or 2000
Numeric	These are numbers and **can include decimals**. If within quotation marks they will be considered character class.	23.1 or 14
Factor	These are vectors that have a **specified order** or hierarchy of values	An variable of economic status with ordered values
Date	**Once R is told that certain data are Dates**, these data can be manipulated and displayed in special ways. See the page on Working with dates for more information.	2018-04-12 or 15/3/1954 or Wed 4 Jan 1980
Logical	Values must be one of the two special values TRUE or FALSE (note these are **not** "TRUE" and "FALSE" in quotation marks)	TRUE or FALSE
data.frame	A data frame is how R stores a **typical dataset**. It consists of vectors (columns) of data bound together, that all have the same number of observations (rows).	The example AJS dataset named linelist_raw contains 68 variables with 300 observations (rows) each.
tibble	tibbles are a variation on data frame, the main operational difference being that they print more nicely to the console (display first 10 rows and only columns that fit on the screen)	Any data frame, list, or matrix can be converted to a tibble with as_tibble()
list	A list is like vector, but holds other objects that can be other different classes	A list could hold a single number, and a dataframe, and a vector, and even another list within it!

A NAN value in R represents "NOT A NUMBER", It is basically any numeric calculations with an undefined result, such as '0/0'.This exists only in vectors with the numeric datatype.

A NA value in R represents "NOT AVAILABLE". This can exist in any sort of numeric or character vector. It is generally interpreted as missing values.

NULL is an object and is returned when an expression or function results in an undefined value.

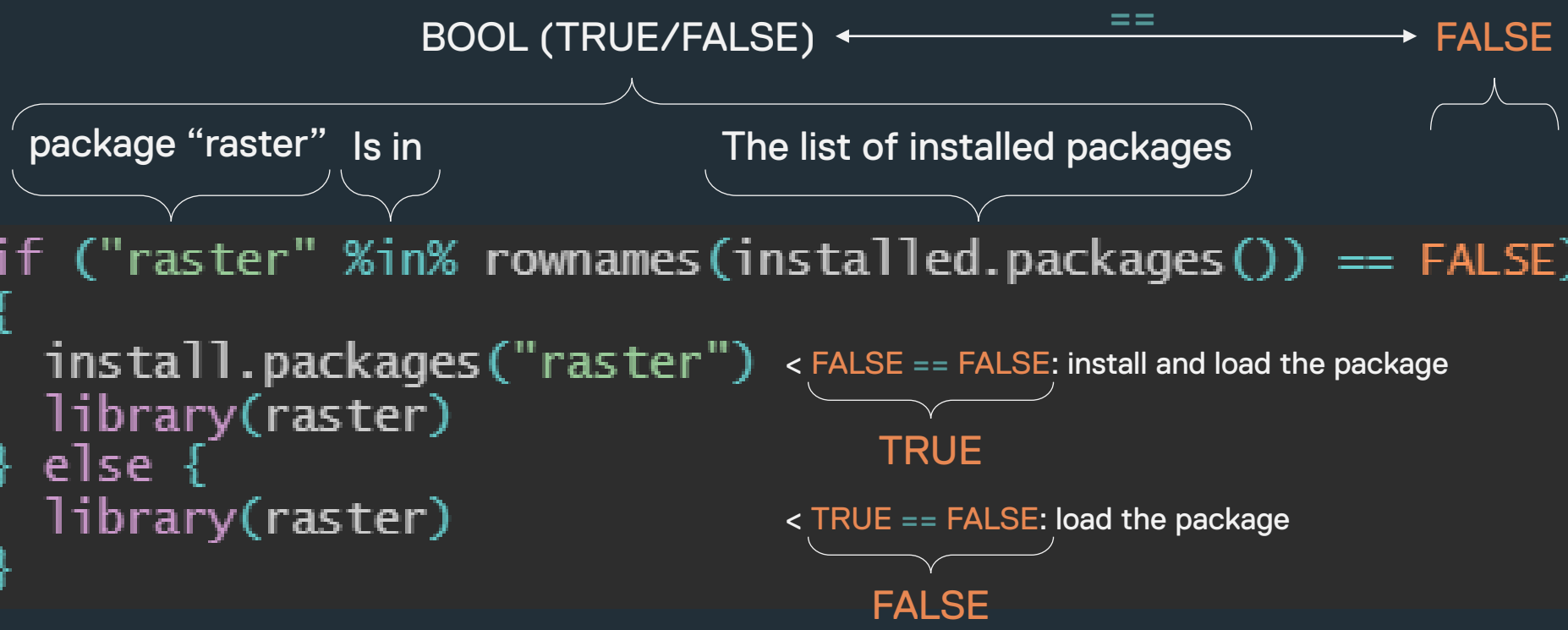
Inf and -Inf stands for infinity (or negative infinity) and is a result of storing either a large number or a product that is a result of division by zero.

In numeric operations, you can avoid errors due to undefined values with the argument na.rm = TRUE

e.g:

sum(1, 2, NA) = NA
sum(1, 2, NA, na.rm = TRUE) = 3

1. Loading all necessary packages



If (bool) { bool = True, do something } else { bool = False, do something else }

If (bool) { bool = True, do something } else if (if initial condition is False, test a second condition) {do...}

& = and, | = or > if (apple is green & apple is sweet), if (apple is green | apple is sweet)

2. Setting up your working folder

setwd("C:/Users/user_name/Documents/Folder/ACADIA 2022")

e.g: C:/Users/user_name/Documents/Folder/ACADIA 2022/file.shp

Read("file.shp")

getwd() give back the folder on setwd – informative

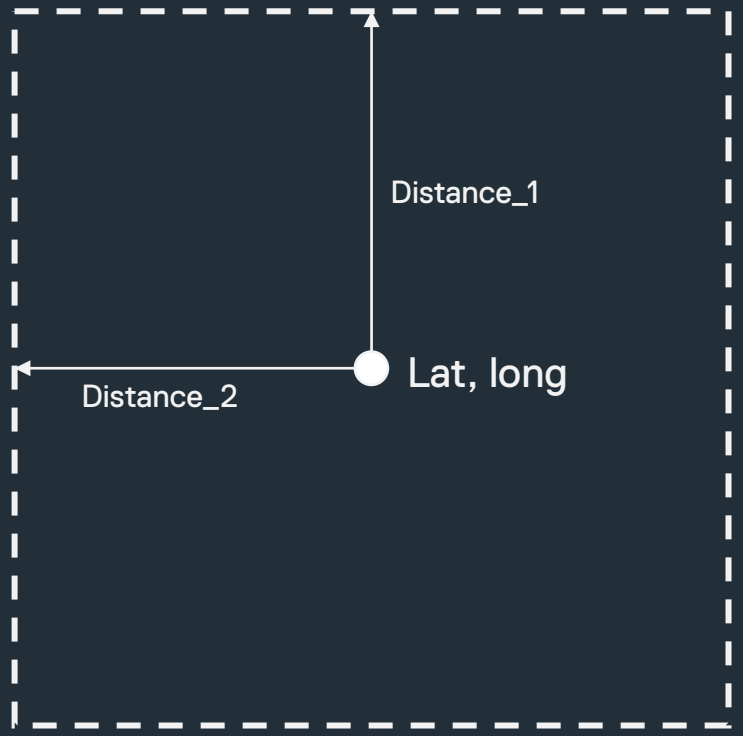
R language does not recognize back slash as they are reserved for special characters, please change back slashes on your folder path by a single or double forward slash;

- The "forward slash" / is actually more common as it used by Unix, Linux, and macOS
- The "backward slash" \ is actually somewhat painful as it is also an escape character. So whenever you want one, you need to type two in the string: "C:\\TEMP".

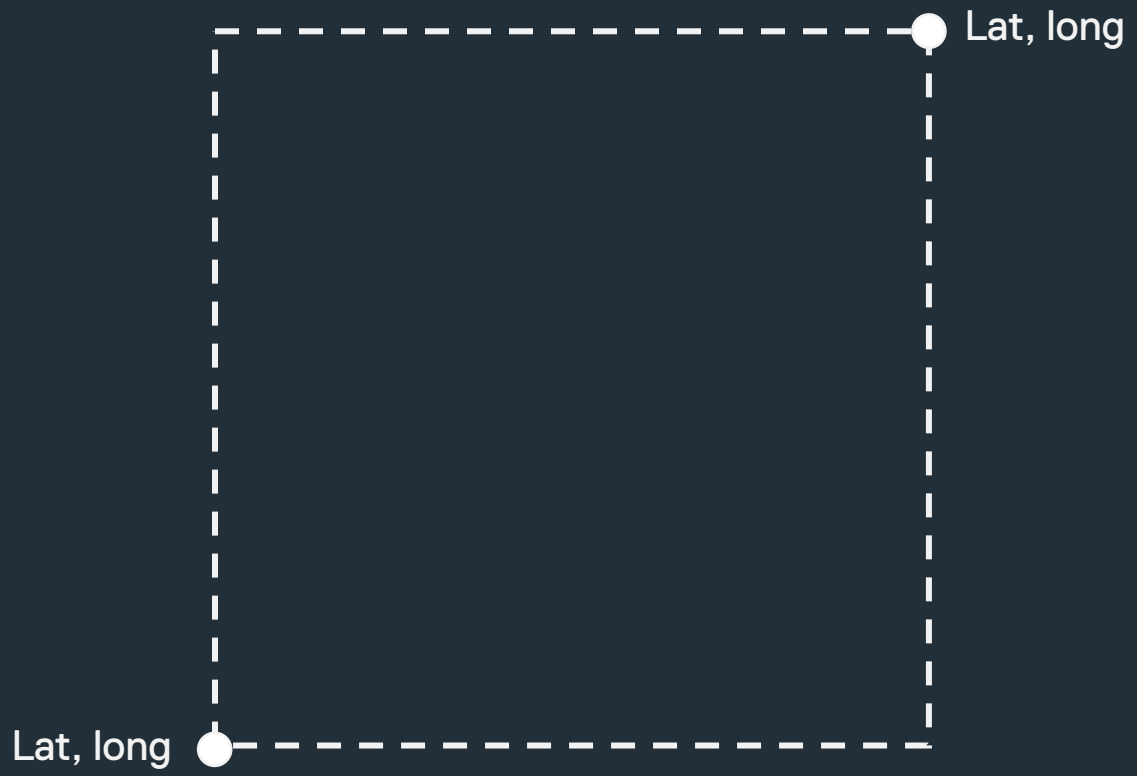
*REGEX (regular expression) Special characters

3. Setting up our area of analysis

Bounding box (bbox) method 1

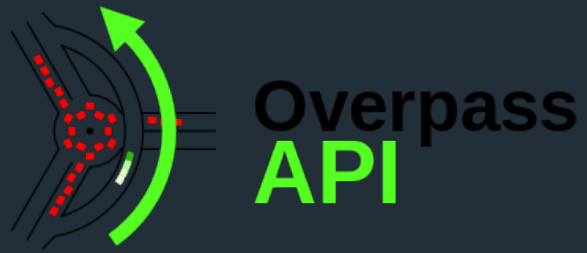


Bounding box (bbox) method 2



bb_poly: transform a bounding box (area) into a polygon which can be used to split/clip/crop other polygons etc.

4. Loading all necessary data



Packages: OSMAR & OSMDATA

```
Query <- bbox %>% opq() %>% add_osm_feature("building")

buildings <- osmdata_sf(Query)
```

Building

This is used to identify individual buildings or groups of connected buildings. See the page **Buildings** for further details on the usage of this tag and `man_made=*` for tagging of various other structures. The building tags are intended for the physical description of a building: for functions in the building (e.g. police station, church, townhall, museum) you should add additional tags like `amenity=*`, `tourism=*`, `shop=*` etc.

For example mapping `building=supermarket` is not enough to mark place as having an active supermarket shop, it just marks that building has form typical for supermarket buildings. `shop=supermarket` must be mapped to indicate an active supermarket shop. On the other hand `shop=*`/`amenity=*` is not indicating building by itself, building must be mapped with `building=*` tag.

Key	Value	Comment	Photo
Accommodation			
building	apartments	A building arranged into individual dwellings, often on separate floors. May also have retail outlets on the ground floor.	
building	barracks	Buildings built to house military personnel or laborers.	

https://wiki.openstreetmap.org/wiki/Map_features

```
Query <- bbox %>% opq()

OSM_sf <- osmdata_sf(Query) # Returns whole osm features
for the defined bounding box
```

Points Lines Polygons Relationships (Multilines, multipolygons)


```
mypoints <- OSM_sf$points
```

4. Loading all necessary data

Building

This is used to identify individual buildings or groups of connected buildings. See the page **Buildings** for further details on the usage of this tag and `man_made=*` for tagging of various other structures. The building tags are intended for the physical description of a building: for functions in the building (e.g. police station, church, townhall, museum) you should add additional tags like `amenity=*`, `tourism=*`, `shop=*` etc.

For example mapping `building=supermarket` is not enough to mark place as having an active supermarket shop, it just marks that building has form typical for supermarket buildings. `shop=supermarket` must be mapped to indicate an active supermarket shop. On the other hand `shop=*`/`amenity=*` is not indicating building by itself, building must be mapped with `building=*` tag.

Key	Value	Comment	Photo
		Accommodation	
building	apartments	A building arranged into individual dwellings, often on separate floors. May also have retail outlets on the ground floor.	
building	barracks	Buildings built to house military personnel or laborers.	

mypolygons$building[3] = “yes”

OSM_sf

- Points
- Lines
- Polygons
- Multilines
- Multipolygons

mypolygons <- OSM_sf$polygons

KEY = Columns

id	Osm_id	KEY	building	KEY	geometry
1	10987	Value	apartments	Value	Sf_polygon
2	11987	Value	NA	Value	Sf_polygon
3	12654	Value	yes	Value	Sf_polygon

VALUES = Rows

4. Loading all necessary data

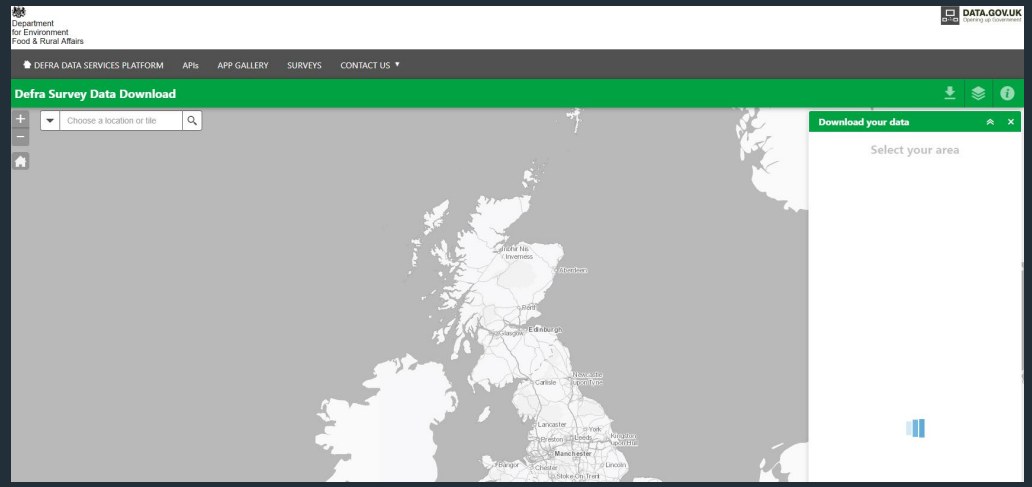
https://cran.r-project.org/web/packages/available_packages_by_name.html

Available CRAN Packages By Name

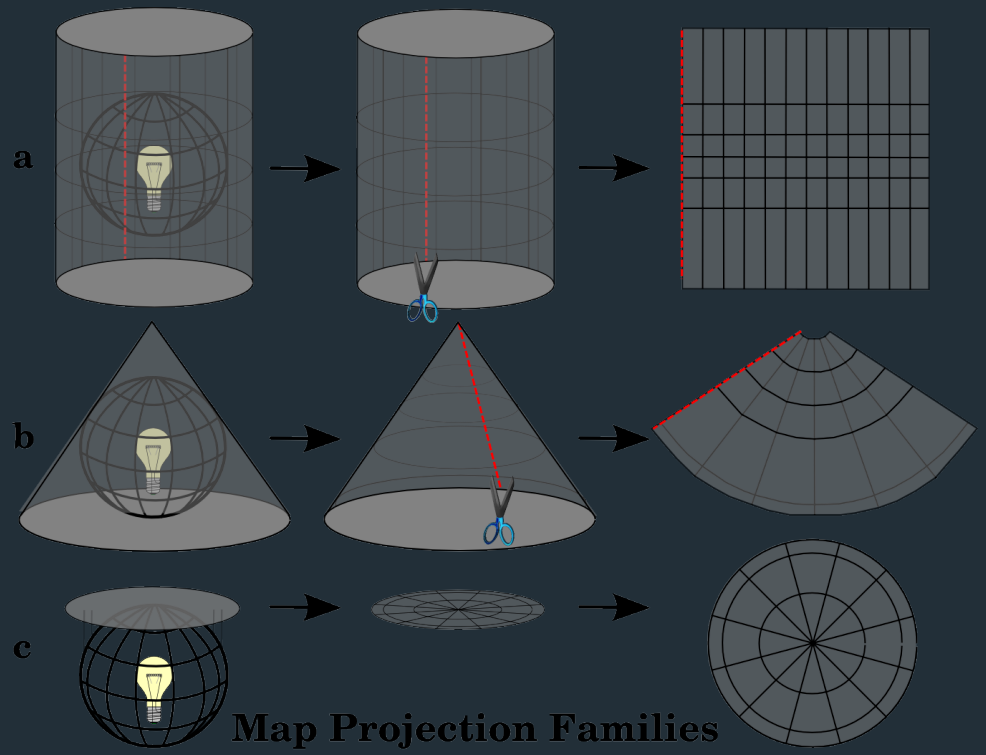
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A3	Accurate, Adaptable, and Accessible Error Metrics for Predictive Models
AATtools	Reliability and Scoring Routines for the Approach-Avoidance Task
ABACUS	Apps Based Activities for Communicating and Understanding Statistics
abbreviate	Readable String Abbreviation
abbyyR	Access to Abbyy Optical Character Recognition (OCR) API
abc	Tools for Approximate Bayesian Computation (ABC)
abc.data	Data Only: Tools for Approximate Bayesian Computation (ABC)
ABC.RAP	Array Based CpG Region Analysis Pipeline
abcADM	Fit Accumulated Damage Models and Estimate Reliability using ABC
ABCanalysis	Computed ABC Analysis
abclass	Angle-Based Large-Margin Classifiers
ABCoptim	Implementation of Artificial Bee Colony (ABC) Optimization
ABCp2	Approximate Bayesian Computational Model for Estimating P2
abcrf	Approximate Bayesian Computation via Random Forests
abcrlda	Asymptotically Bias-Corrected Regularized Linear Discriminant Analysis
abctools	Tools for ABC Analyses
abd	The Analysis of Biological Data
abdiv	Alpha and Beta Diversity Measures
abe	Augmented Backward Elimination
abess	Fast Best Subset Selection
abglasso	Adaptive Bayesian Graphical Lasso
ABHgenotypeR	Easy Visualization of ABH Genotypes
abind	Combine Multidimensional Arrays
abjData	Databases Used Routinely by the Brazilian Jurimetrics Association
abjutils	Useful Tools for Jurimetrical Analysis Used by the Brazilian Jurimetrics Association
abmR	Agent-Based Models in R
abn	Modelling Multivariate Data with Additive Bayesian Networks
abnormality	Measure a Subject's Abnormality with Respect to a Reference Population
abodOutlier	Angle-Based Outlier Detection
ABPS	The Abnormal Blood Profile Score to Detect Blood Doping

acs	Download, Manipulate, and Present American Community Survey and Decennial Data from the US Census
Census2016	Data from the Australian Census 2016
censusapi	Retrieve Data from the Census APIs
censusGeography	Changes United States Census Geographic Code into Name of Location
censusr	Collect Data from the Census API
censusxy	Access the U.S. Census Bureau's Geocoding A.P.I. System
nomisr	Access 'Nomis' UK Labour Market Data
wdnr.gis	Pull Spatial Layers from 'WDNR ArcGIS REST API'
galah	Atlas of Living Australia (ALA) Data and Resources in R
usincometaxes	Calculate Federal and State Income Taxes in the United States



4. Loading all necessary data



Cylindrical

Conical

Planar

Latitude and Longitude coordinates do NOT have any projections, those are spherical coordinates

sf::sf_use_s2(FALSE), avoids using spherical calculations from the package “sp”

Lat & Lng can be directly translated to EPSG: 4326, CRS Mercator (e.g: google maps)

sf::st_transform(sf_geometry, 27700), allows you to reproject your geometry from one CRS to another

Measurements may change from one CRS to another

The full CRS string could be obtained if required by googling your EPSG code + “proj.4” or at EPSG.io

“+proj=tmerc +lat_0=49 +lon_0=-2 +k=0.9996012717 +x_0=400000 +y_0=-100000 +ellps=airy +nadgrids=OSTN15_NTv2_OSGBtoETRS.gsb +units=m +no_defs +type=crs”

Projection name	Use extension	Scale factor	Use extension (easting/northing)	Ellipsoid name	Filename of NTv2 grid file to use for datum transforms	Units

4. Loading all necessary data

LIDAR
(Light Detection and Ranging)
Point Cloud (.LAZ)

DEM
(Digital Elevation Model)

DSM
(Digital Surface Model)

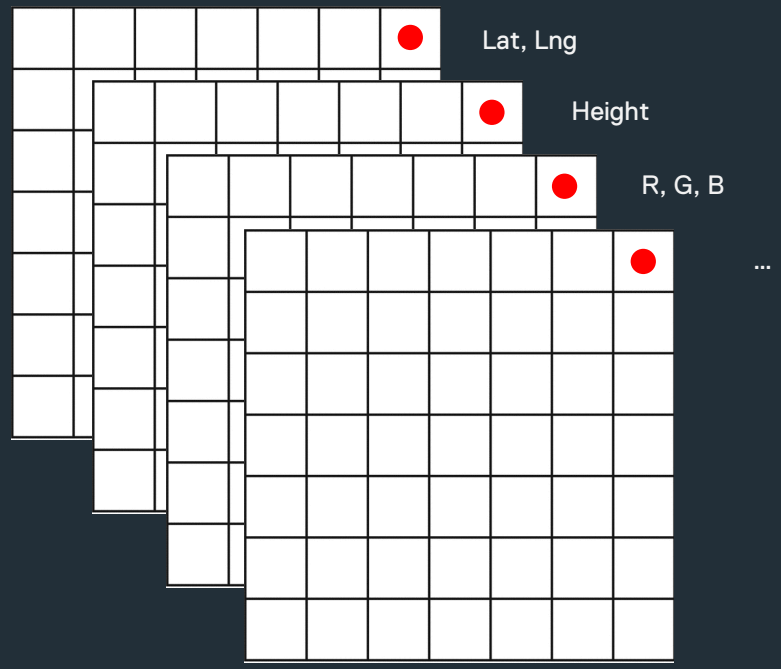
DTM
(Digital Terrain Model)

VOM
(Vegetation Object Model)

...

GEOTIFF (.Tif)

Form detection and classification – heavy computational power



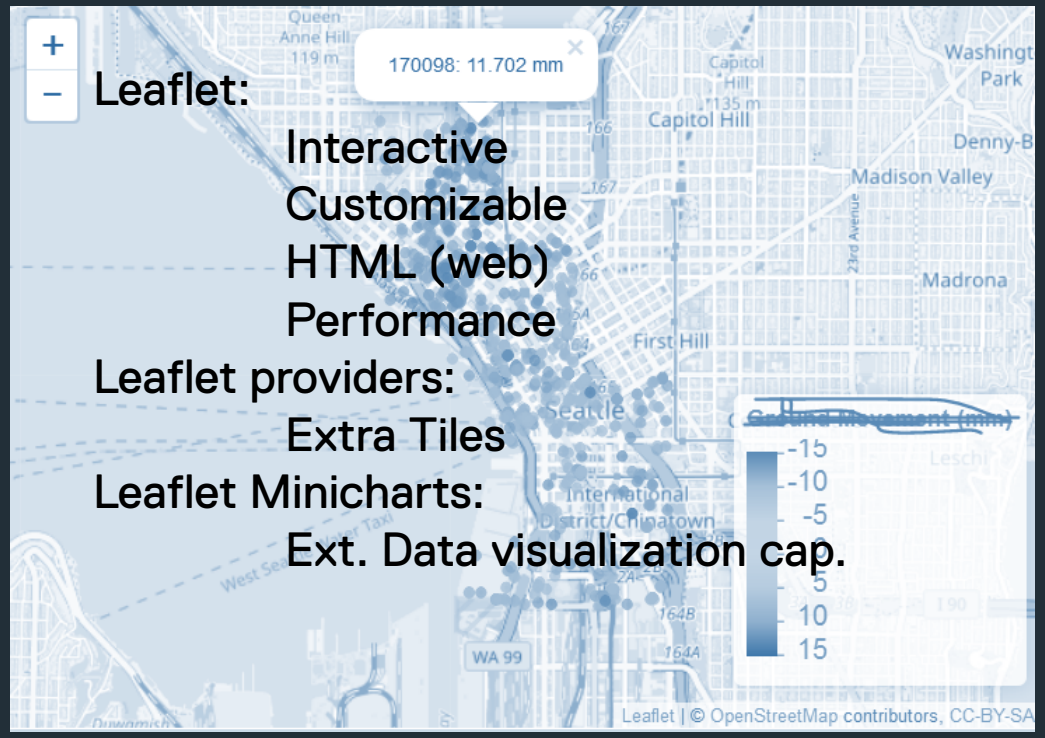
5. Processing and visualizing loaded data

[Row, Column]

```
Buildings_sf <- osm_pol[!is.na(osm_pol$building), ]
```

! == “is not”
!is.na() == “is not NA”

Select all ROWS that are not NA
at COLUMN building from
osm_pol, keeping all the columns




```
leaflet::leaflet() %>% leaflet::addTiles() %>% leaflet::addPolygons(data = Buildings_sf, popup = ~ building)
```

Empty map (initialize)

Add background map

Add building polygons, pop-up building column information when feature clicked (leaflet uses ~ rather than $)

LEAFLET only plot geometry in lat and lng, EPSG 4326
(Mercator CRS)

5. Processing and visualizing loaded data

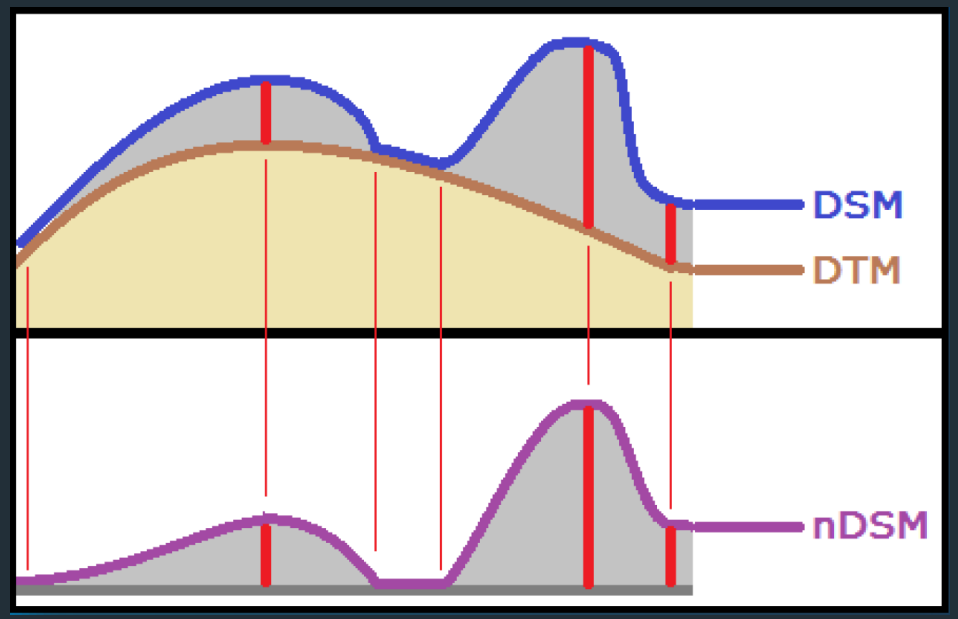
```
osm_hfpMax <- raster::extract(DSM, OSM_Build_27700, fun = max, na.rm = TRUE)
```

		10	10	10	10	
	10	30	31	33	10	
	10	30	31	33	10	10
	10	30	31	33	35	10
	30	30	31	33	35	10
	10	10	10	10	10	10

fun = max == 35

fun = min == 30

fun = mean == 31.7



Norm_DSM = DSM – DTM

DSM and DTM heights are based on sea levels we need to subtract DTM heights from DSM heights to obtain relative building heights

5. Processing and visualizing loaded data

```
df_lines <- osm_line
```

Line features for column Highway represent classified street segments, such as primary, secondary...

```
Cropped_lines <- sf::st_crop(df_lines, sf::st_bbox(BBox_poly)) ←→ Crop all lines with bounding box polygon
Cropped_lines <- Cropped_lines[!is.na(Cropped_lines$highway), ] ←→ select only those which are NON NA values on Highway column
St_network <- data.frame(highway = Cropped_lines$highway, geometry = Cropped_lines$geometry) ←→ create a new data frame only with geometry and Highway column
St_network_27700 <- sf::st_transform(sf::st_as_sf(St_network), 27700) ←→ reproject to EPSG: 27700 (British)
```

Repeat the same operation with points features.
Points features for column Highway represent crosswalks, traffic lights, etc.

```
Cropped_pt <- sf::st_crop(osm_points, sf::st_bbox(BBox_poly))
Cropped_pt <- Cropped_pt[!is.na(Cropped_pt$highway), ]
St_network_Feat <- data.frame(highway = Cropped_pt$highway, geometry = Cropped_pt$geometry)
St_network_Feat_27700 <- sf::st_transform(sf::st_as_sf(St_network_Feat), 27700)
```

5. Processing and visualizing loaded data

POI – Point of Interest

Based on OSM values, we should create a list of all amenities, this list might change based on different locations, project types and other considerations

```
amenity_list <- c("yes", "bar", "biergarten", "cafe", "fast_food“, ...) ←→ list of amenities considered

amenity_pol <- osm_pol[, "amenity"] %>% tidygraph::filter(!is.na(amenity)) ←→ selecting amenity column and filtering by non NA values
amenity_filt <- amenity_pol[amenity_pol$amenity %in% amenity_list, ] ←→ filtering again based on the list creates previously
amenity_pol_filt <- sf::st_centroid(amenity_filt) ←→ calculating the centroid of each polygon so we have a list of points

amenity_pt <- osm_points[, c("amenity", "geometry")] %>% tidygraph::filter(!is.na(amenity))
amenity_pt_filt <- amenity_pt[amenity_pt$amenity %in% amenity_list, ]  Same process as before but with the point data-frame

amenity_pt_total <- as.data.frame(rbind.fill(amenity_pt_filt, amenity_pol_filt)) ←→ we now can join the result of the points data-frame and the polygon(centroid) data-frame
```

5. Processing and visualizing loaded data

POI – Point of Interest

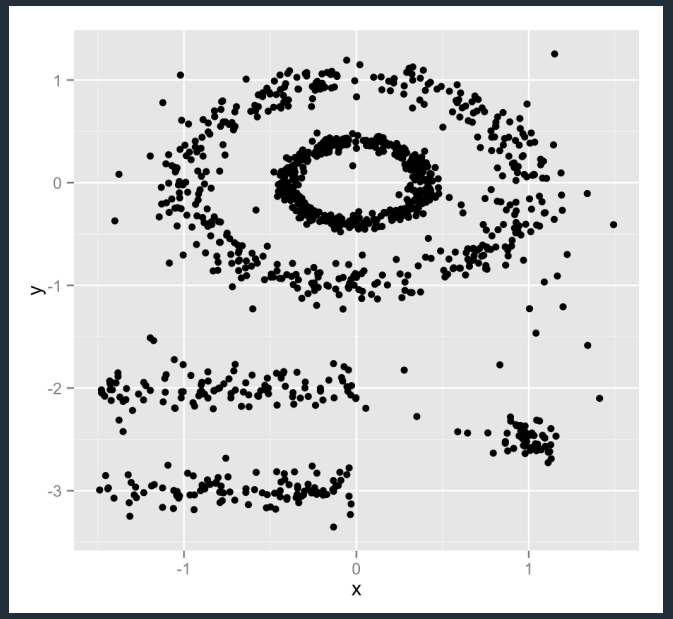
```
if (!is.null(amenity_pt_total)) {

  amenity_sustenance <- c(...)
  amenity_education <- c(...)
  amenity_finance <- c(...)
  amenity_healthcare <- c(...)
  amenity_culture <- c(...)
  amenity_public <- c(...)
  amenity_others <- c(...)
```

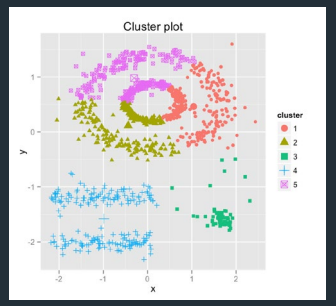
amenity_list <- c()

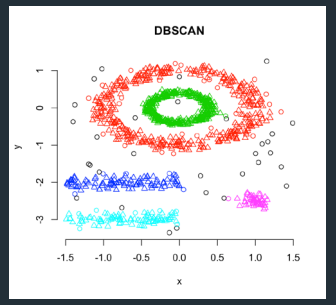
```
for (i in seq_along(amenity_pt_total$amenity)) {
    if (amenity_pt_total$amenity[i] %in% amenity_sustenance) {
      amenity_pt_total$category[i] <- "sustenance"
    } else if (amenity_pt_total$amenity[i] %in% amenity_education) {
      amenity_pt_total$category[i] <- "education"}
    else if (amenity_pt_total$amenity[i] %in% amenity_finance) {
      amenity_pt_total$category[i] <- "finance"}
    else if (amenity_pt_total$amenity[i] %in% amenity_healthcare) {
      amenity_pt_total$category[i] <- "healthcare"}
    else if (amenity_pt_total$amenity[i] %in% amenity_public) {
      amenity_pt_total$category[i] <- "public"}
    else if (amenity_pt_total$amenity[i] %in% amenity_others) {
      amenity_pt_total$category[i] <- "others"}
  }

}
```

If the value in column amenity is in the list amenity_sustenance, then assign the value “sustenance” to the column Category. If not, then test another condition (else if)

5. Processing and visualizing loaded data







data: data matrix, data frame or dissimilarity matrix (dist-object). Specify method = “dist” if the data should be interpreted as a dissimilarity matrix or object. Otherwise, Euclidean distances will be used.

eps: Reachability maximum distance

MinPts: Reachability minimum number of points

scale: If TRUE, the data will be scaled

method: Possible values are:

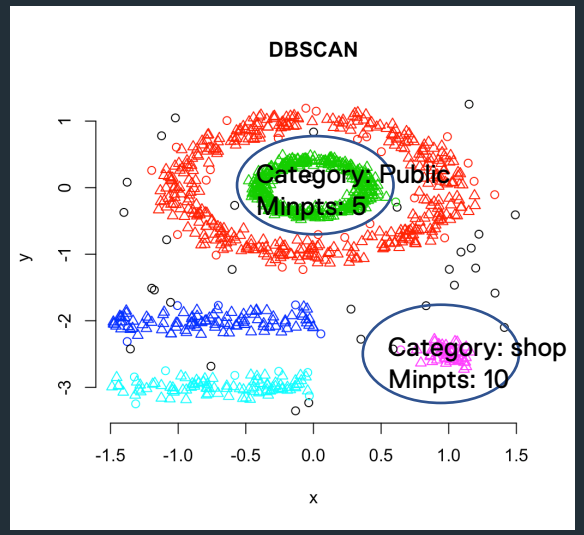
- dist: Treats the data as a distance matrix
- raw: Treats the data as raw data
- hybrid: Expect also raw data, but calculates partial distance matrices

DBSCAN is a **density-based clustering algorithm**, introduced by Ester et al. 1996, which can be used to identify clusters of any shape in data set containing noise and outliers. DBSCAN stands for Density-Based Spatial Clustering and Application with Noise.

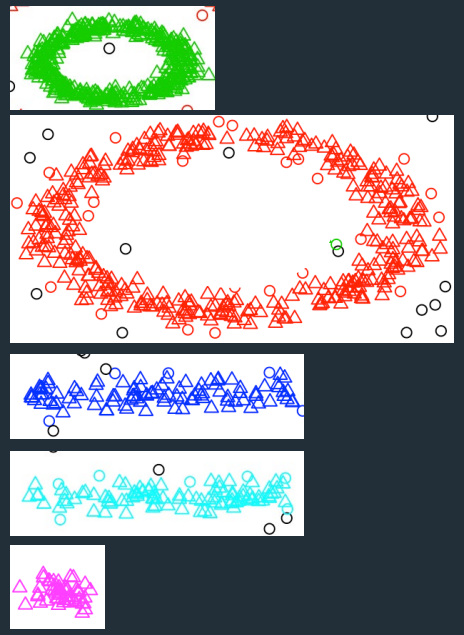
Clustering analysis is an unsupervised learning method that separates the data points into several specific bunches or groups, such that the data points in the same groups have similar properties and data points in different groups have different properties in some sense.

5. Processing and visualizing loaded data

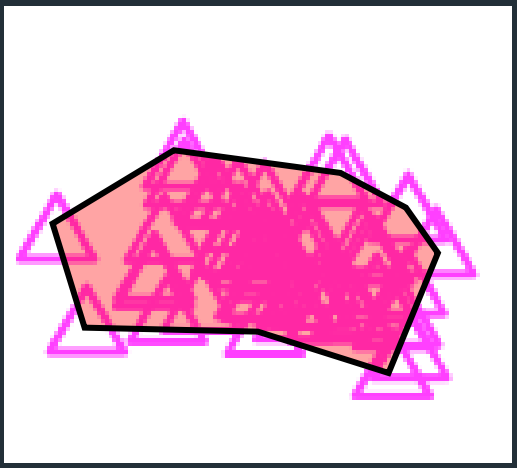
assign_clusters()



hulls()



Get_hull()



5. Processing and visualizing loaded data

ArcGIS RESTFUL API

Method 1 (package arcpullr):

```
arcpullr::get_layer_by_poly("https://services5.arcgis.com/GfwWNkhOj9bNBqoJ/arcgis/rest/services/MAPPLUTO/FeatureServer/0", BBox_sf, sp_rel = "esriSpatialRelIntersects")
```

ArcGIS Server

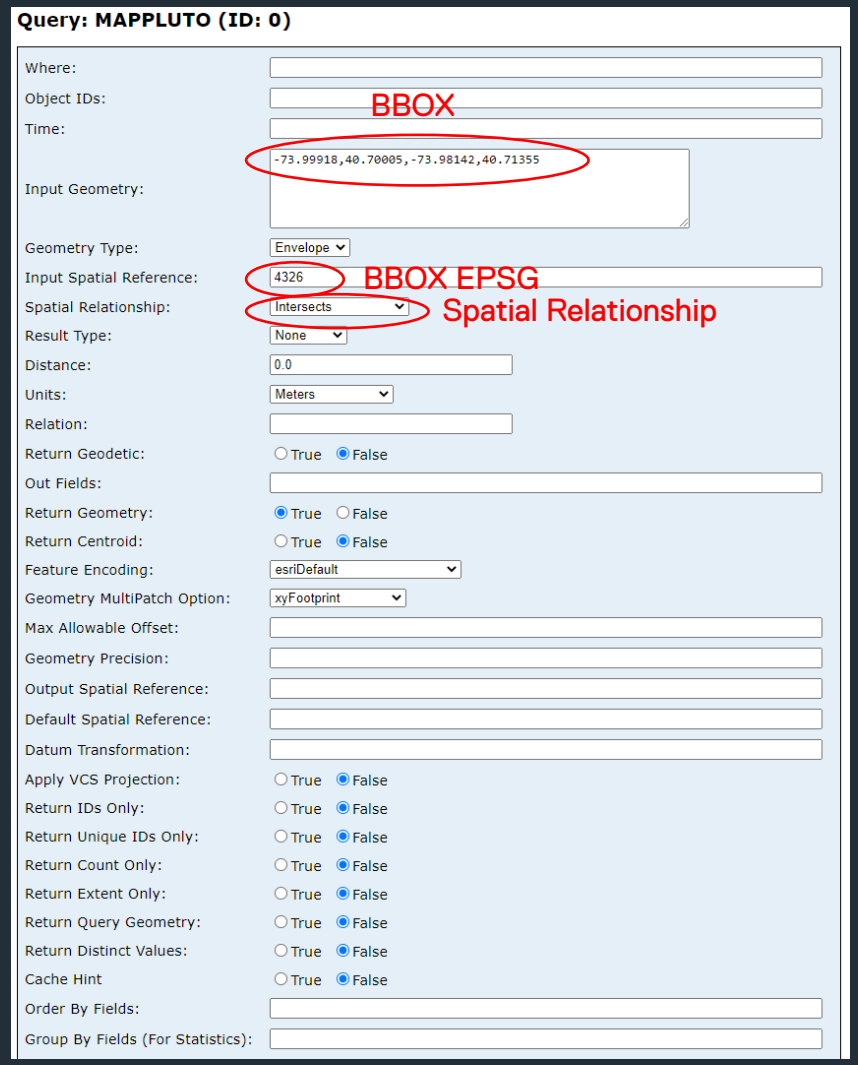
sf polygon

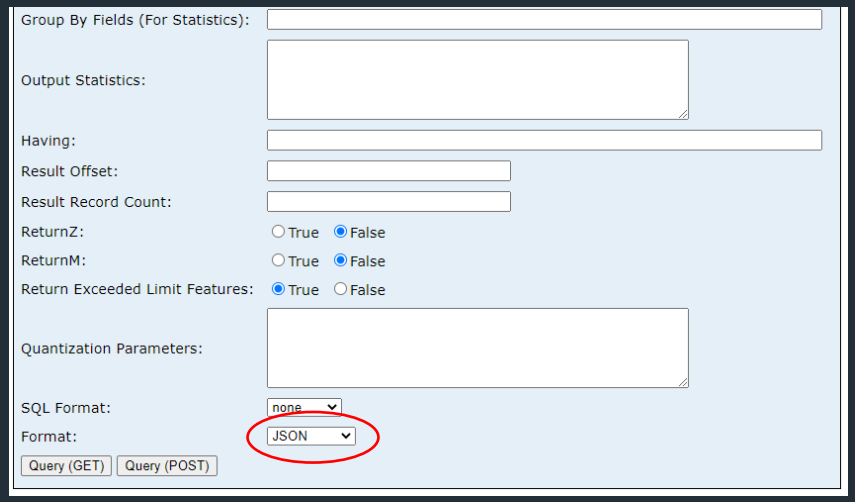
Spatial relationship: intersects, contains ...

Method 2 (url):

```
ESRI_query <-
paste("https://onsinspire.esriuk.com/arcgis/rest/services/Census_Boundaries/Lower_Super_Output_Areas_December_2011_Centroids/MapServer/0/query?where=1%3D1&outFields=*&
geometry=", as.character(bbox_coord$left), "%2C", as.character(bbox_coord$bottom), "%2C", as.character(bbox_coord$right), '%2C', as.character(bbox_coord$top),
"&geometryType=esriGeometryEnvelope&inSR=4326&spatialRel=esriSpatialRelIntersects&outSR=4326&f=json", sep = "")

query_result <- jsonify::from_json(ESRI_query)
```

5. Processing and visualizing loaded data





https://services5.arcgis.com/GfwWNkhOj9bNBqoJ/arcgis/rest/services/MAPPLUTO/FeatureServer/0/query?where=&objectIds=&time=&geometry=-73.99918%2C40.70005%2C-73.98142%2C40.71355&geometryType=esriGeometryEnvelope&inSR=4326&spatialRel=esriSpatialRelIntersects&resultType=none&distance=0.0&units=esriSRUnit_Meter&relationParam=&returnGeodetic=false&outFields=&returnGeometry=true&returnCentroid=false&featureEncoding=esriDefault&multipatchOption=xyFootprint&maxAllowableOffset=&geometryPrecision=&outSR=&defaultSR=&datumTransformation=&applyVCSProjection=false&returnIdsOnly=false&returnUniqueIdsOnly=false&returnCountOnly=false&returnExtentOnly=false&returnQueryGeometry=false&returnDistinctValues=false&cacheHint=false&orderByFields=&groupByFieldsForStatistics=&outStatistics=&having=&resultOffset=&resultRecordCount=&returnZ=false&returnM=false&returnExceededLimitFeatures=true&quantizationParameters=&sqlFormat=none&f=pjson&token=

5. Saving the Data

```
Save_folder <- "C://Users//your_name//Documents//ACADIA 2022//Results//"

sf::st_write(OSM_Build_27700, paste0(Save_folder, "Buildings.shp"), delete_dsn = TRUE)
```

SF object to save

Path

Re-write if exist

```
write.csv(CENSUS_df$age_structure, file = paste0(Save_folder, "CENSUS_age_structure.csv"), sep = ",")
```

DF object to save

Path

Separator (CSV = comma separated values)