Data Practical ENVT3361/ENVT4461: maps in R

Basic maps in R using the maptiles, sf, sp, and prettymapr packages

load packages needed to make maps

```r
library(sp)
library(sf)
```

```
## Linking to GEOS 3.9.1, GDAL 3.4.3, PROJ 7.2.1; sf_use_s2() is TRUE
```

```r
library(maptiles)
library(prettymapr)
```

Code to read data copied from an Excel worksheet into an R data frame (or from a file):

```r
## afs19 <- read.table('clipboard', header=T, sep="\t", stringsAsFactors = TRUE)
# alternative code reading data from a file:
afs19 <- read.csv(file = "afs19.csv", stringsAsFactors = TRUE) # un-comment me
```

Preparing to make maps

First we define some commonly-used coordinate reference systems which define the projection of or GPS data

```r
LongLat <- CRS("+proj=longlat +ellps=WGS84
           +datum=WGS84 +no_defs") # uses Earth ellipsis specs from WGS84 datum
UTM50 <- CRS("+proj=utm +zone=50 +south") # just for Zone 50, S hemisphere!
```

We will use these coordinate reference system objects later. For now we're going to work in UTM coordinates.

We define the area we need for our map and save it in an object called 'extent' 'extent' is a 'simple features' object which refers to a formal standard (ISO 19125-1:2004) that describes how objects in the real world can be represented in computers - see https://r-spatial.github.io/sf/articles/sf1.html (this is why we need the 'sf' or 'Simple Features' package by Pebesma (2018))

```r
extent <-
  st_as_sf(SpatialPoints(coords =
                         data.frame(x = c(399800,400700),
                                    y = c(6467900,6468400)),
                         proj4string = UTM50))
```

NOTE: The projection we specify here will be the one the map plots in!

Getting the map tile data

We now need some functions from the 'maptiles' package (Giraud 2021). We're using one of the OpenStreetMap tile options, but the following tile providers also work:
OpenStreetMap, OpenStreetMap.HOT, Esri.WorldStreetMap, Esri.WorldTopoMap, Esri.WorldImagery, Esri.WorldGrayCanvas, Stamen.Toner, Stamen.TonerBackground, Stamen.TonerHybrid, Stamen.TonerLines, Stamen.TonerLabels, Stamen.TonerLite, CartoDB.Positron, CartoDB.DarkMatter, CartoDB.Voyager (all CartoDB... tiles have variants which work), OpenTopoMap
The option `crop = TRUE` is included to crop the tiles to our defined area in the `extent` object. If we leave it out, the map may change shape as it will use only square (uncropped) map tiles.

```r
aftiles <- get_tiles(extent, provider = "OpenStreetMap.HOT", crop = TRUE)
```

Plotting the map

The 'aftiles' object we created is a 'SpatRaster' object which needs the maptiles package loaded to be able to plot it. We need to set the outer margins of the plot area (by default they are zeros) to allow plotting of axes *etc.*

```
par(oma=c(3,3,1,1), lend="square")
plot(aftiles)

box(lwd=6,col="white") # tidies up skewness of UTM tiles a bit
axis(1, tcl=-0.2, mgp = c(1.6,0.3,0))
mtext("Easting (UTM Zone 50, m)", side = 1, line = 1.5, font=2)
axis(2, tcl=-0.2, mgp = c(1.6,0.3,0))
mtext("Northing (UTM Zone 50, m)", side = 2, line = 1.5, font=2)
box()
```

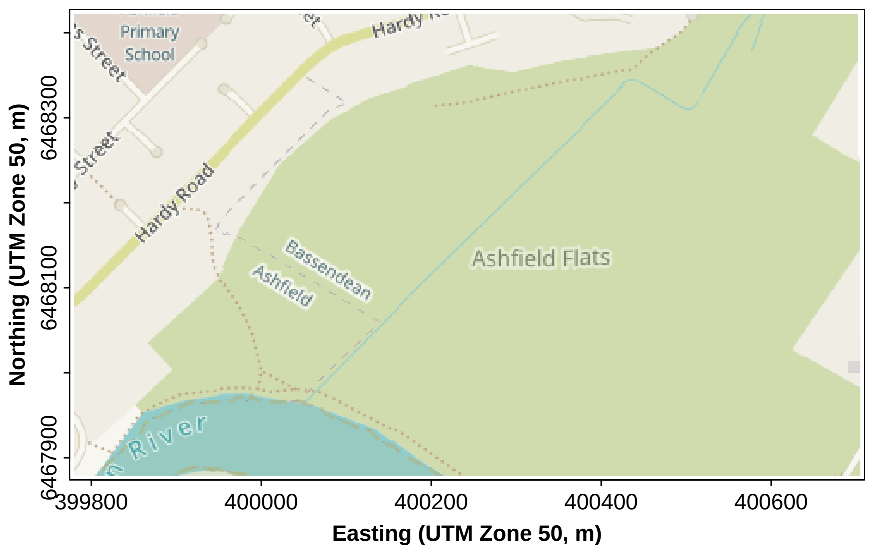


Figure 1: Map of Ashfield Flats in the UTM projection, generated using the maptiles R package, with OpenStreetMap tiles.

The next chunk of code adds the `prettymapr` features:

```
addnortharrow(text.col=1, border=1)
addscalebar(plotepsg = 32750, label.col = 1, linecol = 1,
            label.cex = 1.2, htin=0.15, widthhint = 0.15)
```

Adding our data and map annotations

Since we have a map in UTM coordinates, we can now add plots of our data based on UTM locations (with a legend, of course).

```
with(afs19, points(Easting, Northing, lwd = 2,
                   pch = c(0,1,2,3,5,6,15,17,18,19)[Group],
                   col = rainbow(10,v=0.6,end=0.75)[Group]))
legend("bottomright", legend=levels(as.factor(afs19$Group)),
       pch = c(0,1,2,3,5,6,15,17,18,19), col = rainbow(10,v=0.6,end=0.75),
       pt.lwd = 2, title = "Group", inset = 0.02, ncol = 2)
```

We can also add digitized map features such as wetland ponds, drains, *etc.* Ideally we would add these *before* plotting the data.

```
with(afr_map, lines(drain_E, drain_N, col = "cadetblue3", lwd = 2))
with(afr_map, lines(wetland_E, wetland_N, col = "cadetblue3", lwd = 1, lty = 2))
```

Finally, we would most likely want to add some text. Text labels should also be added *before* plotting the data.

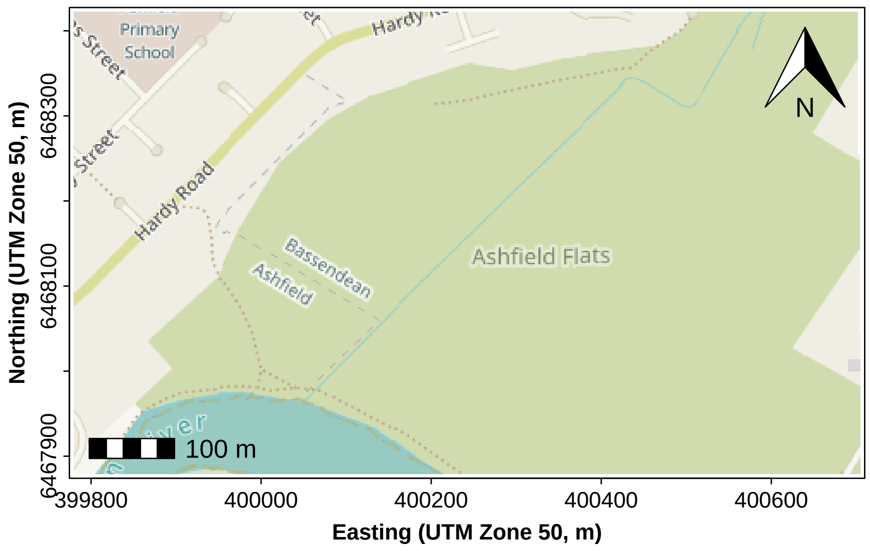


Figure 2: Map of Ashfield Flats (UTM), with added North arrow and scale bar from prettymapr, over OpenStreetMap tiles acquired using maptiles.

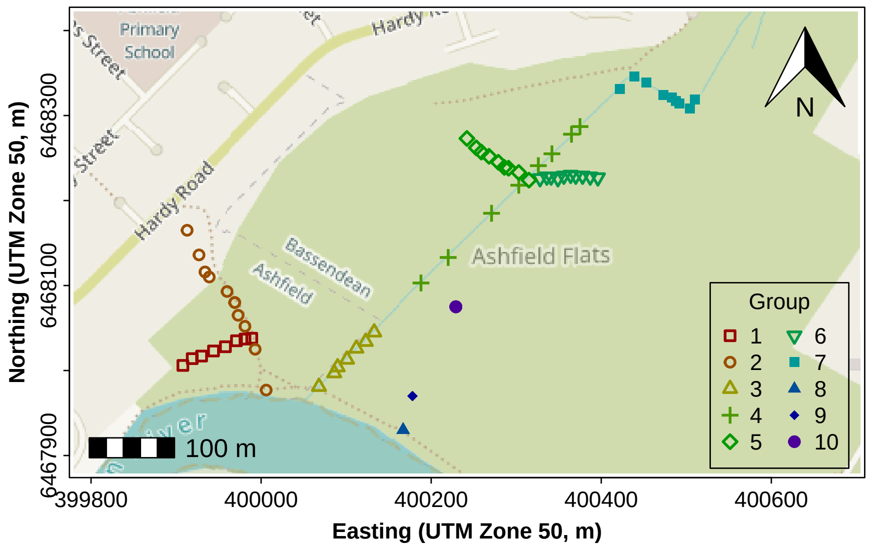


Figure 3: Map of Ashfield Flats (UTM), with prettymapr features, 2019 sediment locations, and legend, over OpenStreetMap tiles acquired using maptiles.

```
text(c(400263, 399962, 400047), c(6468174, 6468083, 6468237),
     labels = c("Chapman Drain","Kitchener Drain", "Woolcock Drain"),
     pos = c(2,2,4), cex = 0.8, font = 3, col = "cadetblue3")
```

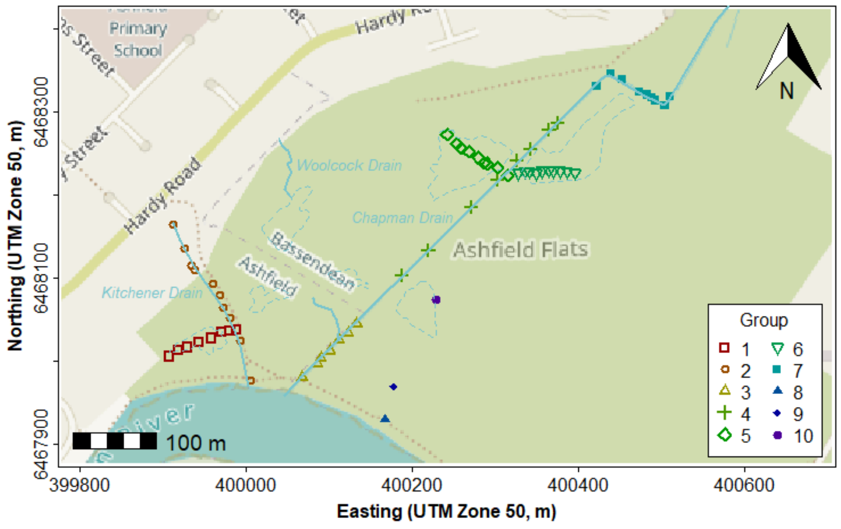


Figure 4: Map of Ashfield Flats (UTM projection) showing sediment and soil samples collected in 2019 by the ENVT3361 class. Generated using the maptiles R package, with OpenStreetMap map tiles.

Alternative - use OpenStreetMap to make an initial map object

We use the OpenStreetMap R package (Fellows, 2019) to make a plot-able map object, based on the coordinates which bound a rectangular area.

The easiest way to get bounding coordinates (latitude, longitude) is by using Google Earth or Google Maps. We need the north-west (upper left) and south-east (lower right) coordinates of the rectangular map we want to plot.

Note that south latitudes (and west longitudes) are negative, and we want decimal degrees rather than degrees-minutes-seconds.

```
require(OpenStreetMap)
UWA_osm <- openmap(upperLeft = c(-31.974, 115.812),
                   lowerRight = c(-31.988, 115.828),
                   zoom = 16,
                   type = "osm")
plot(UWA_osm)
```

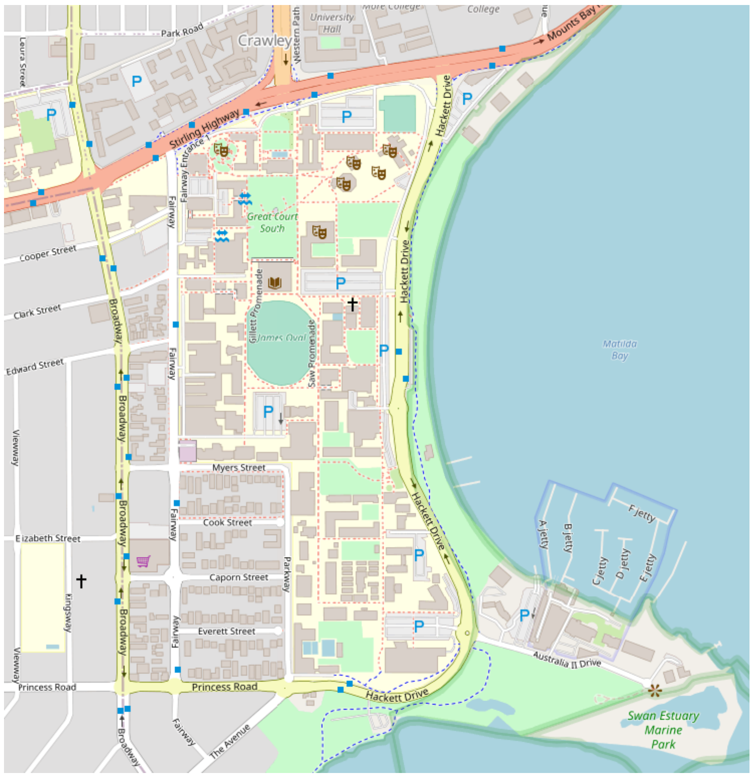


Figure 1 Map of the University of Western Australia Crawley campus using the default OpenStreetMap settings.

The map in Figure 1 needs some attention. Without some prior knowledge, we don't know where on Earth's surface this map represents, we don't know the direction it's oriented in, and we don't know the scale. We can add a north arrow to indicate direction, a scale bar to show the scale, and axes to show any coordinates which should show the location on Earth.

Plot the UWA campus map with margins, axes, and annotations

```r
require(OpenStreetMap)
require(prettymapr)
par(mar = c(3, 3, 0.5, 0.5), mgp = c(1.6, 0.3, 0), tcl = 0.4)
plot(UWA_osm, removeMargin = FALSE)
axis(1)
axis(2)
addnortharrow()
addscalebar()
box()
```

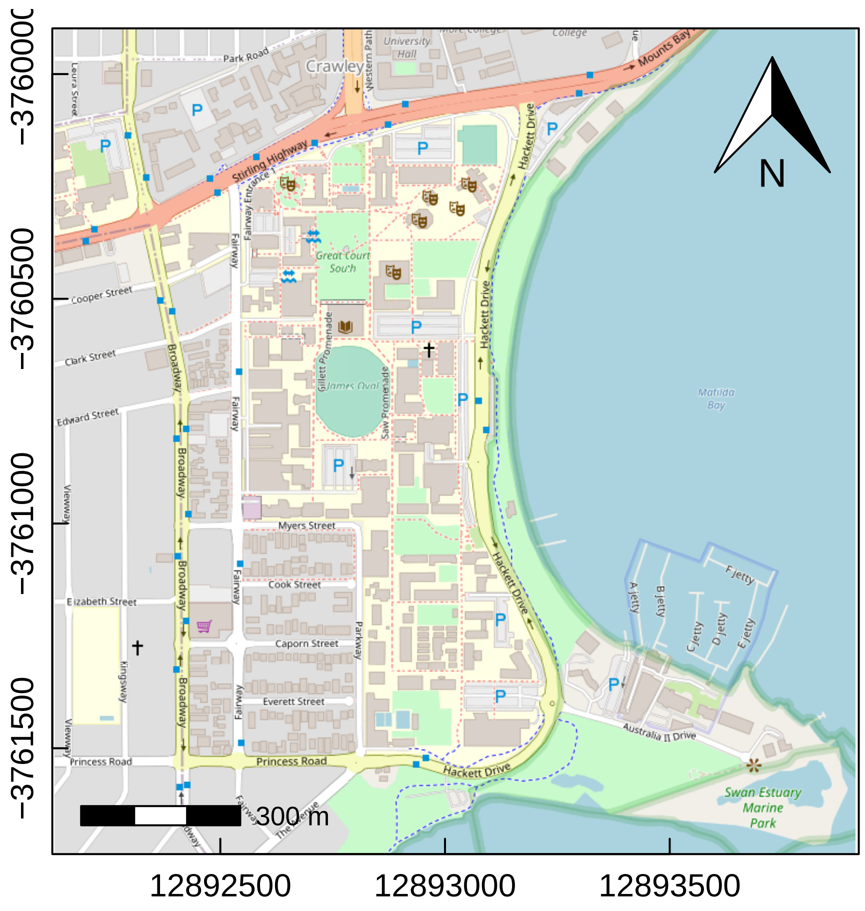


Figure 2 Map of the University of Western Australia Crawley campus showing scale, north arrow, and axes, but retaining some default OpenStreetMap settings.

The map in Figure 2 is better – we now know scale and direction, but finding our location on Earth is tricky because the OpenStreetMap map projection is “Google Mercator” which is not a projection most people are familiar with. We need a map in a commonly-used projection such as Longitude-Latitude, or Universal Transverse Mercator (UTM). We can change projections using the openproj() function in the OpenStreetMap package.

Plot a re-projected UWA campus map in UTM with margins, axes, and annotations

First convert the projection…

```
require(OpenStreetMap)
UWA_utm <- openproj(UWA_osm, projection = "+proj=utm +zone=50 +south")
{cat("Show upper left (p1) and lower right (p2) coordinates\n")
UWA_utm$bbox} # show bounding coordinates to check
```

```
## Show upper left (p1) and lower right (p2) coordinates
```

```
## $p1
## [1]  387741.6 6461856.6
##
## $p2
## [1]  389290 6460267
```

...then plot the map in its new projection.

```
require(OpenStreetMap)
require(prettymapr)

# plot the map in its new projection
par(mar = c(3, 3, 0.5, 0.5), mgp = c(1.6, 0.3, 0), tcl = 0.4)
plot(UWA_utm, removeMargin = FALSE)
axis(1)
mtext("Easting (UTM Zone 50, m)", side = 1, line = 1.6, font = 2, cex = 1.2)
axis(2)
mtext("Northing (UTM Zone 50, m)", side = 2, line = 1.6, font = 2, cex = 1.2)
addnortharrow(scale = 1.2)
addscalebar(plotepsg = 32750, htin = 0.15, label.cex = 1.2)
box()
```

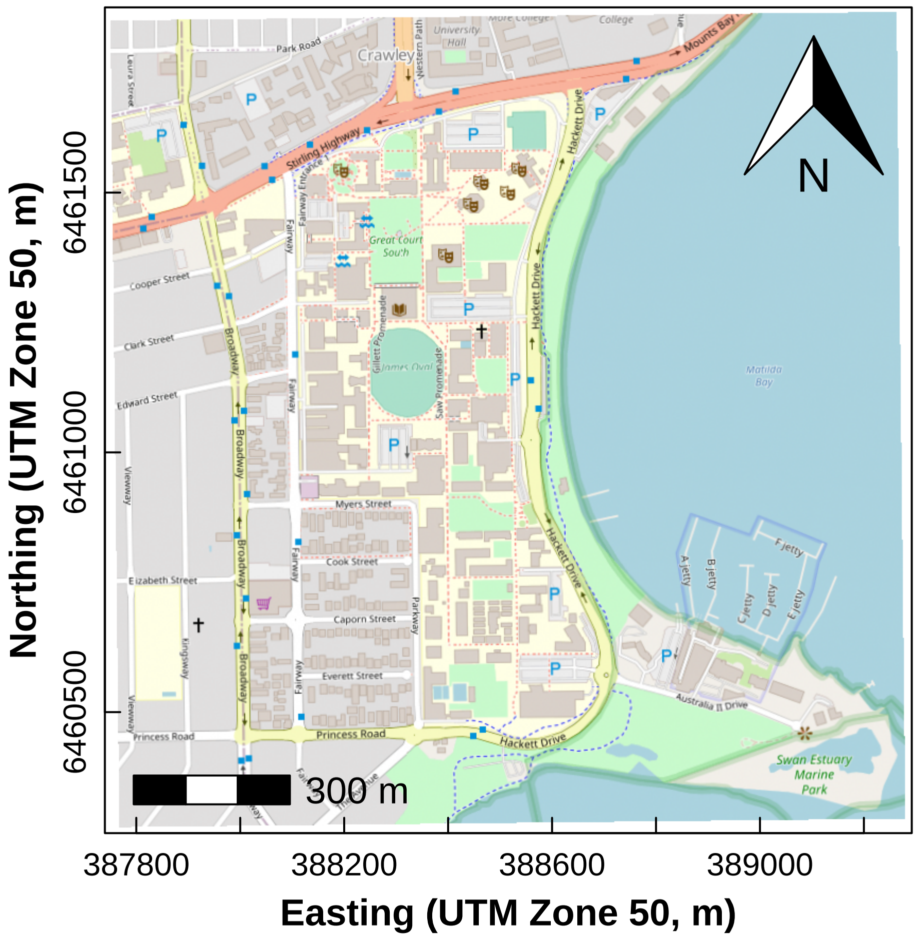


Figure 3 Map of the University of Western Australia Crawley campus showing scale, north arrow, and axes, and with the map projection changed from the OpenStreetMap default to Universal Transverse Mercator (UTM).

The map in Figure 3 is now just what we need – we know scale and direction, and we have axes in the Universal Transverse Mercator (UTM) projection with units of metres, which we would normally set up our field GPS to show.

Very often we would like to **add our own information to a map**, such as the location of samples, often with different sizes or shapes of symbols to represent numerical information.

Making some data to plot on our map

```r
Longitude <-  c(115.8211, 115.8186, 115.8186, 115.8189, 115.8178, 115.8183,
                115.8209, 115.8204, 115.8199, 115.8194, 115.8198, 115.8181)
Latitude <- c(-31.9822, -31.98303, -31.98265, -31.98109, -31.979, -31.9765,
              -31.986, -31.97688, -31.97971, -31.97638, -31.97834, -31.97872)
Easting <- c(388624.5, 388387.9, 388383.5, 388415.6, 388308.3, 388354.6,
             388603.9, 388551.3, 388501.3, 388452.6, 388493.5, 388334)
Northing <- c(6460930, 6460836, 6460878, 6461051, 6461282, 6461559,
              6460509, 6461519, 6461205, 6461574, 6461357, 6461313)
Name <- c("Matilda Bay", "Barry Marshall", "Science", "Guild", "Reid", "Hackett",
          "Business", "Music", "Law", "Rec Centre", "UniClub", "Reid")
Type <- c("Cafe", "Library", "Cafe", "Cafe", "Library", "Cafe",
          "Cafe", "Library", "Library", "Cafe", "Cafe", "Cafe")
Usage <- c(10,10,8,4,2,7,1,0.1,0.1,1,2,4)
places <- as.data.frame(cbind(Longitude, Latitude, Easting, Northing, Usage))
places$Name <- Name
places$Type <- as.factor(Type)
# str(places)
rm(list = c("Longitude", "Latitude", "Easting", "Northing", "Name", "Type", "Usage"))
print(places)
```

```
##    Longitude  Latitude  Easting Northing Usage           Name    Type
## 1   115.8211 -31.98220 388624.5  6460930  10.0    Matilda Bay    Cafe
## 2   115.8186 -31.98303 388387.9  6460836  10.0 Barry Marshall Library
## 3   115.8186 -31.98265 388383.5  6460878   8.0        Science    Cafe
## 4   115.8189 -31.98109 388415.6  6461051   4.0          Guild    Cafe
## 5   115.8178 -31.97900 388308.3  6461282   2.0           Reid Library
## 6   115.8183 -31.97650 388354.6  6461559   7.0        Hackett    Cafe
## 7   115.8209 -31.98600 388603.9  6460509   1.0       Business    Cafe
## 8   115.8204 -31.97688 388551.3  6461519   0.1          Music Library
## 9   115.8199 -31.97971 388501.3  6461205   0.1            Law Library
## 10  115.8194 -31.97638 388452.6  6461574   1.0     Rec Centre    Cafe
## 11  115.8198 -31.97834 388493.5  6461357   2.0        UniClub    Cafe
## 12  115.8181 -31.97872 388334.0  6461313   4.0           Reid    Cafe
```

Plot simple data on our best map

An OpenStreetMap plot is essentially an x-y plot with a map image as the plot background.

We can use standard R graphics functions to add points, lines, polygons, and text (including legends) to the map.

```r
require(OpenStreetMap)
require(prettymapr)

# plot the map in its new projection
par(mar = c(3, 3, 0.5, 0.5), mgp = c(1.6, 0.3, 0), tcl = 0.4)
plot(UWA_utm, removeMargin = FALSE)
axis(1)
mtext("Easting (UTM Zone 50, m)", side = 1, line = 1.6, font = 2, cex = 1.2)
axis(2)
mtext("Northing (UTM Zone 50, m)", side = 2, line = 1.6, font = 2, cex = 1.2)
addnortharrow(scale = 1.2)
addscalebar(plotepsg = 32750, widthhint = 0.1, htin = 0.1, label.cex = 1.2)
box()
#
# plotting our data
points(places$Easting, places$Northing,
       pch = c(15, 19)[places$Type],
       col = c("red3", "blue2")[places$Type],
       cex = c(1.3, 1.5)[places$Type])
text(places$Easting, places$Northing,
     labels = paste(places$Name, places$Type),
     cex = 0.8,
     pos = c(4,2,4,4,2,2,4,4,4,4,4,4),
     col = c("red3","blue2","red3","red3","blue2","red3",
               "red3","blue2","blue2","red3","red3","red3"))
legend("right",
       legend = levels(places$Type),
       pch = c(15, 19),
       col = c("red3", "blue2"),
       pt.cex = c(1.3, 1.5),
       cex = 0.9,
       inset = 0.03)
```

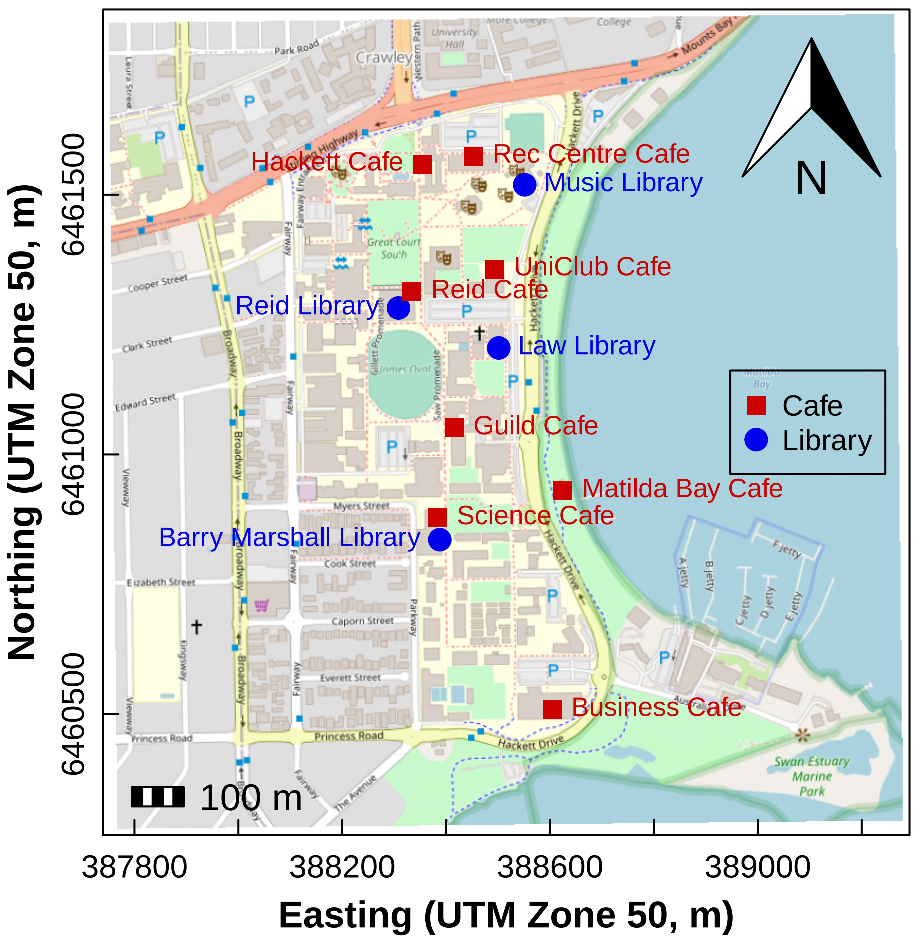


Figure 4 Data on locations of some libraries and cafes plotted on a map of the University of Western Australia Crawley campus showing scale, north arrow, and axes, with the map projection changed to UTM.

Plot numerical data on our best map

One way of representing quantities at different spatial locations is to use a "bubble plot", using the symbols() function in R.

We calculate the square root of the quantity being represented (see code below), to make our circle area proportional to quantity – area is more intuitive for human perception of amounts than are radius or diameter.

```
require(OpenStreetMap)
require(prettymapr)
```

```
# plot the map in its new projection
par(mar = c(3, 3, 0.5, 0.5), mgp = c(1.6, 0.3, 0), tcl = 0.4,
    lend = "square", ljoin = "mitre")
plot(UWA_utm, removeMargin = FALSE)
axis(1)
mtext("Easting (UTM Zone 50, m)", side = 1, line = 1.6, font = 2, cex = 1.2)
axis(2)
mtext("Northing (UTM Zone 50, m)", side = 2, line = 1.6, font = 2, cex = 1.2)
addnortharrow(scale = 1)
addscalebar(plotepsg = 32750, widthhint = 0.18, htin = 0.1,
            label.cex = 1, padin = c(0.4, 0.2))
box()
#
# plotting our data
points(places$Easting, places$Northing, pch = 3)
symbols(places$Easting, places$Northing,
        circles = sqrt(places$Usage),
        fg = "purple",
        bg = "#b0009940",
        inches = 0.2,
        lwd = 2,
        add = TRUE) # 'add = TRUE' required to overplot map
text(places$Easting, places$Northing,
     labels = paste(places$Name, places$Type, round(places$Usage)),
     cex = 0.8,
     pos = c(4,4,2,4,2,2,4,4,4,4,4,4),
     col = "purple3",
     offset = (places$Usage/8))
```

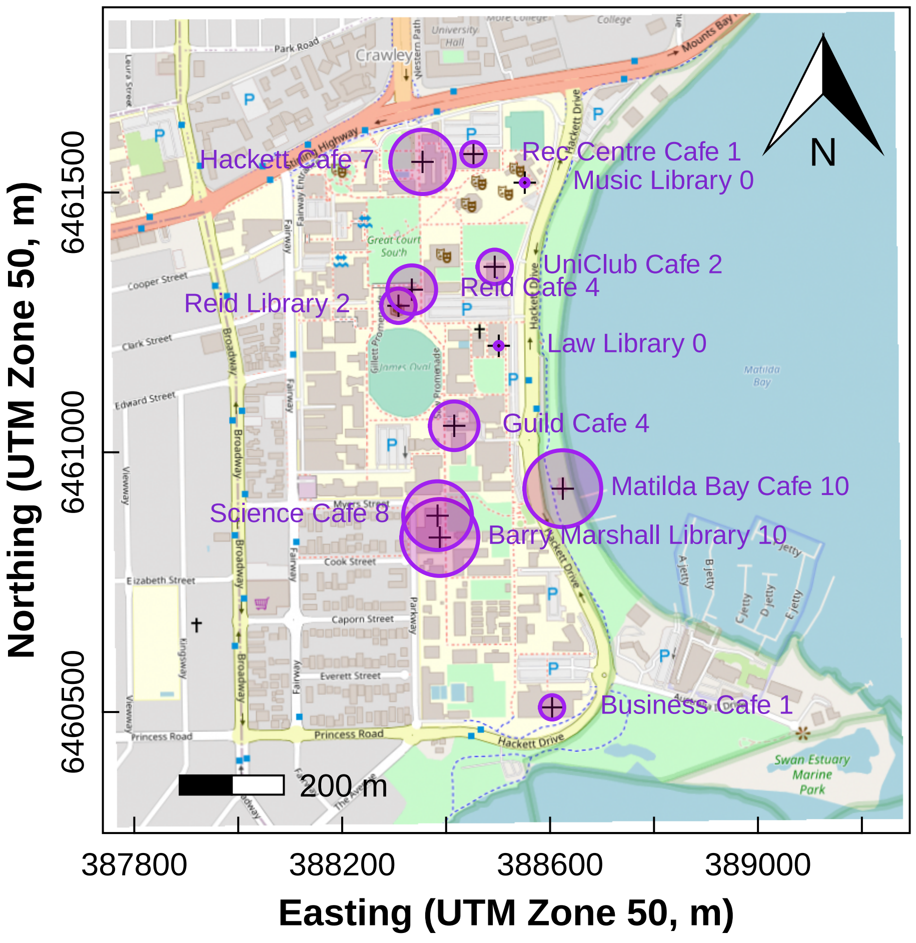


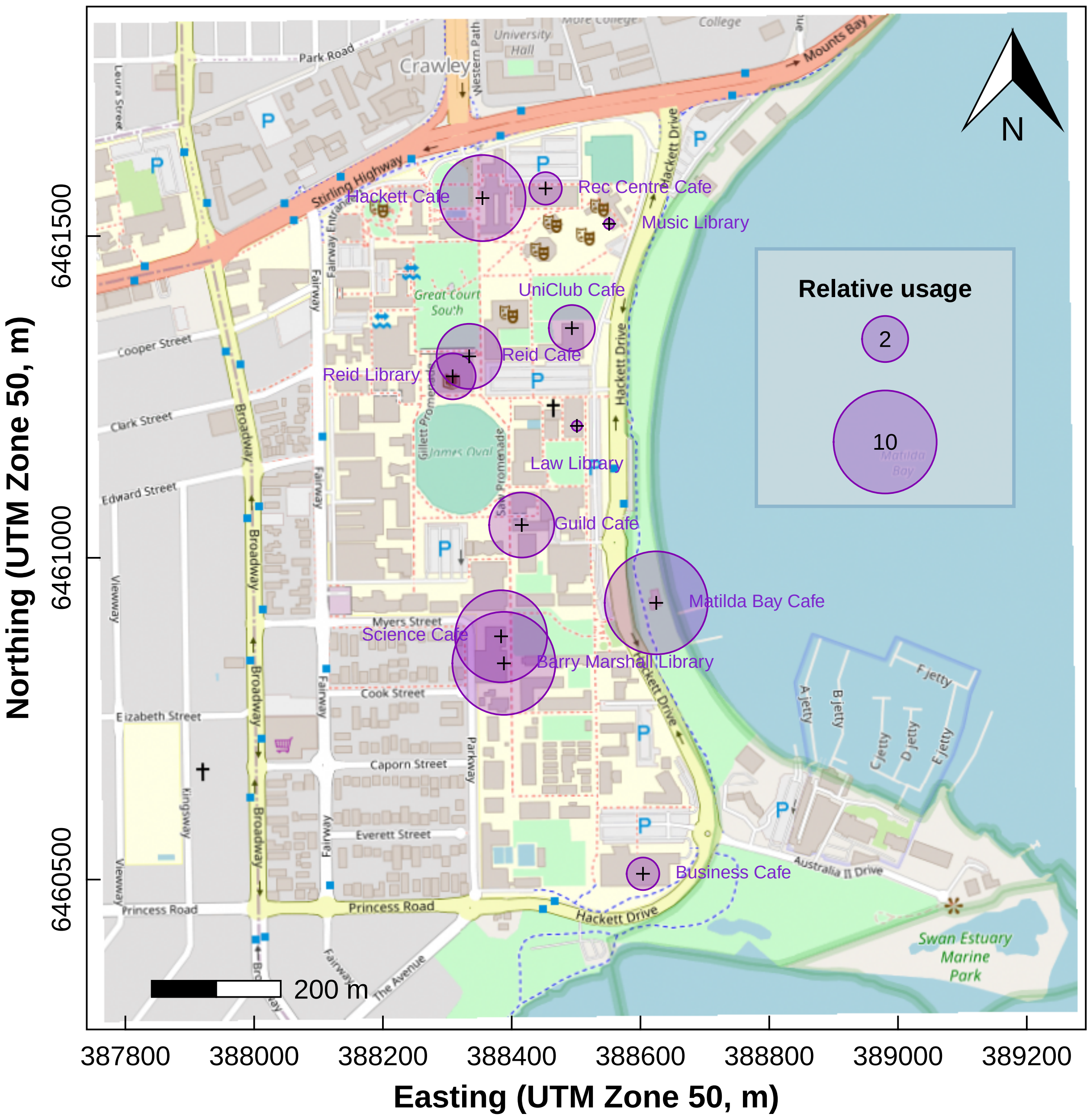
Figure 4 Data on relative usage (by the unit coordinator) of some libraries and cafes plotted on a map of The

University of Western Australia Crawley campus showing scale, north arrow, and axes, with the map projection changed to UTM.

Try adding a legend manually (tricky, optional!)

```r
require(OpenStreetMap)
require(prettymapr)

# plot the map in its new projection
par(mar = c(3, 3, 0.5, 0.5), mgp = c(1.6, 0.3, 0), tcl = 0.4,
    lend = "square", ljoin = "mitre")
plot(UWA_utm, removeMargin = FALSE)
axis(1)
mtext("Easting (UTM Zone 50, m)", side = 1, line = 1.6, font = 2, cex = 1.2)
axis(2)
mtext("Northing (UTM Zone 50, m)", side = 2, line = 1.6, font = 2, cex = 1.2)
addnortharrow(scale = 1)
addscalebar(plotepsg = 32750, widthhint = 0.18, htin = 0.1,
            label.cex = 1, padin = c(0.4, 0.2))
box()
#
# manual legend
# set up your bubble sizes
#[fudgf is a scaling factor- adjust to suit by trial-and-error]
fudgf <- 80
b0 <- round(fudgf/(max(sqrt(places$Usage), na.rm=T)),2)
if (pretty(places$Usage)[1] < 0.001) {
  bublo <- pretty(places$Usage)[2]
} else {
  bublo <- pretty(places$Usage)[1]
}
bubhi <- pretty(places$Usage)[NROW(pretty(places$Usage))]
# draw the bubble symbols
symbols(places$Easting,places$Northing,
        circles=b0*sqrt(places$Usage), add=T,
        lwd=1, inches=F, fg="#8000B0", bg="#8000B040")
#
# optionally plot sample locations as symbols
points(places$Northing~places$Easting,
       pch=3, cex=0.7)
#
# optionally draw a box around where your legend will be
# positioning the rectangle takes more trial-and-error!
rect(388780,6461080,389180,6461480,
     border = "lightskyblue3", col = "#E0E0E080", lwd = 2)
# add a legend title
text(388980,6461380,labels="Relative usage",
     col=1, font=2, cex=0.9, pos = 3)
#
# choose coordinates where legend bubbles go (trial-and-error!)
symbols(c(388980,388980),c(6461340,6461180), circles=b0*sqrt(c(bublo,bubhi)), add=T,
        lwd=1, inches=F, fg="#8000B0", bg="#8000B040")
text(c(388980,388980),c(6461340,6461180),
     labels=c(bublo,bubhi),
     cex=0.85)
# add place name labels
text(places$Easting, places$Northing,
     labels = paste(places$Name, places$Type),
     cex = 0.7,
     pos = c(4,4,2,4,2,2,4,4,1,4,3,4),
     col = "purple3",
     offset = 1)
```



```
# remove temporary objects
rm(list = c("fudgf","b0","bublo","bubhi"))
```

Figure 5 Data on relative usage (by the unit coordinator) of some libraries and cafes plotted on a map of The University of Western Australia Crawley campus showing scale, north arrow, axes, and a manual and slightly unattractive legend, with the map projection changed to UTM.

References

Dunnington, Dewey (2017). prettymapr: Scale Bar, North Arrow, and Pretty Margins in R. R package version 0.2.2. https://CRAN.R-project.org/package=prettymapr

Fellows, Ian and using the JMapViewer library by Jan Peter Stotz (2019). OpenStreetMap: Access to Open Street Map Raster Images. R package version 0.3.4. https://CRAN.R-project.org/package=OpenStreetMap