R Studio Data Visualisation Guide for REG Covid Projects

This guide will explain how to create a range of data visualisations on R Studio using any new data that comes in on the REG Covid Projects delivery partners. The current data visualisations that can be produced using the R Studio Script provided are:

- Bar Chart
- Stacked Chart
- Alluvial Graph
- Georgraphic Plot with pie charts

The software required to produce these visualisations are

1. R
2. R Studio

The Data

- The data required to produce the visualisations is stored on the REG sharepoint and can be found here: Sharepoint
- The excel document required is 'COVID Project Visualisation data_raw_20XX-XX'.
- Before new data is added to this document make a copy and archive it first. Once archived, the new data can be added to the original document and this can be resaved with the relevant month in the document name.
- Save a copy of the new data to the data folder in the following GitHub repository: Data-Visualisation-Project

Updating the COVID Partner Data

- When updating the COVID partners raw data document, ensure that all of the cells are filled with up-to-date date.
- The postcode and coordinates for the contributors, beneficiaries and coordinators can be found on google.

Creating the project from Github and setting the working directory

1. In your documents create a folder called 'R Studio - COVID Data Visualisations'.
2. Open up R Studio and select create a new project.
3. Select Version Control > Git > and paste in the following GitHub Repository Url link.
4. Browse your documents and find the folder 'R Studio - COVID Data Visualisations'. Create the project as a subdirectory of this folder.
5. Create project.
6. In R, go to the Files section and find the R Script ‘daisySampleScript’ in the src folder and open this. Still on the files tab select more > set as working directory.
7. The following packages need to be loaded or installed if not already present. This can be done by committing the following code, which is in the script, to the console.

```
library(readxl)
library(lubridate)
library(tidyr)
library(dplyr)
library(ggplot2)
library(ggalluvial)
library(tibble)
library(ggmap)
library(stringr)
library(scatterpie)
library(ggrepel)
library(ggfittext)
```

Reading the data into R Studio

1. In the working directory, find the data folder and the new data you have saved.
2. Update the following code with the name of your data document by changing 'September_data' to the file now and submit to the console.
3. This will read the data into the Environment.

```
data <- read_xlsx("../data/September_data.xlsx")
```

Transforming the data

1. To transform the data to make the visualisations submit all code from:

```
data$Start <- ymd(data$`Start date`)
data$End <- ymd(data$`End date`)
```

down to

```
d <- pivot_longer(data,cols=all_of(Months),names_to="ProjectMonth",values_to="MonthEffort",
                  names_repair = "unique")
```

to transform the data. What each step is doing is outlined by the green text in the R Script.

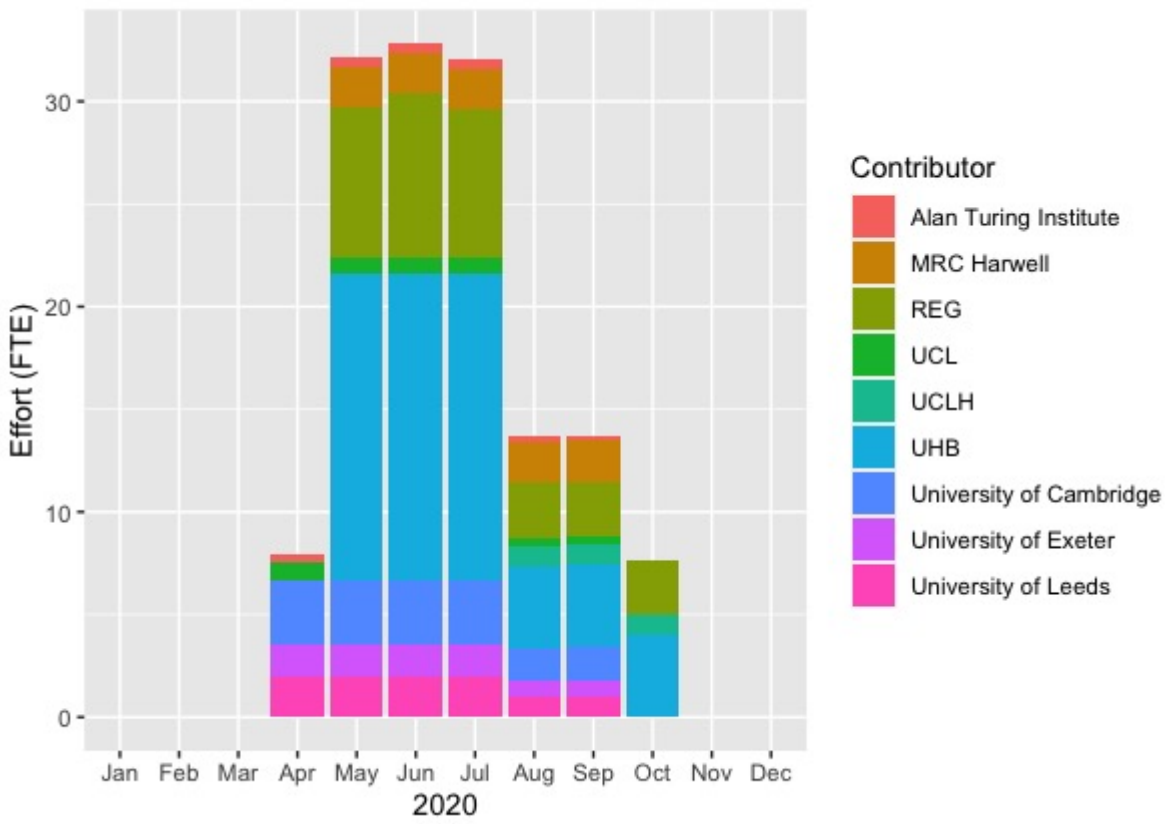
Generating the data visualisations

Bar Charts

Partner Effort

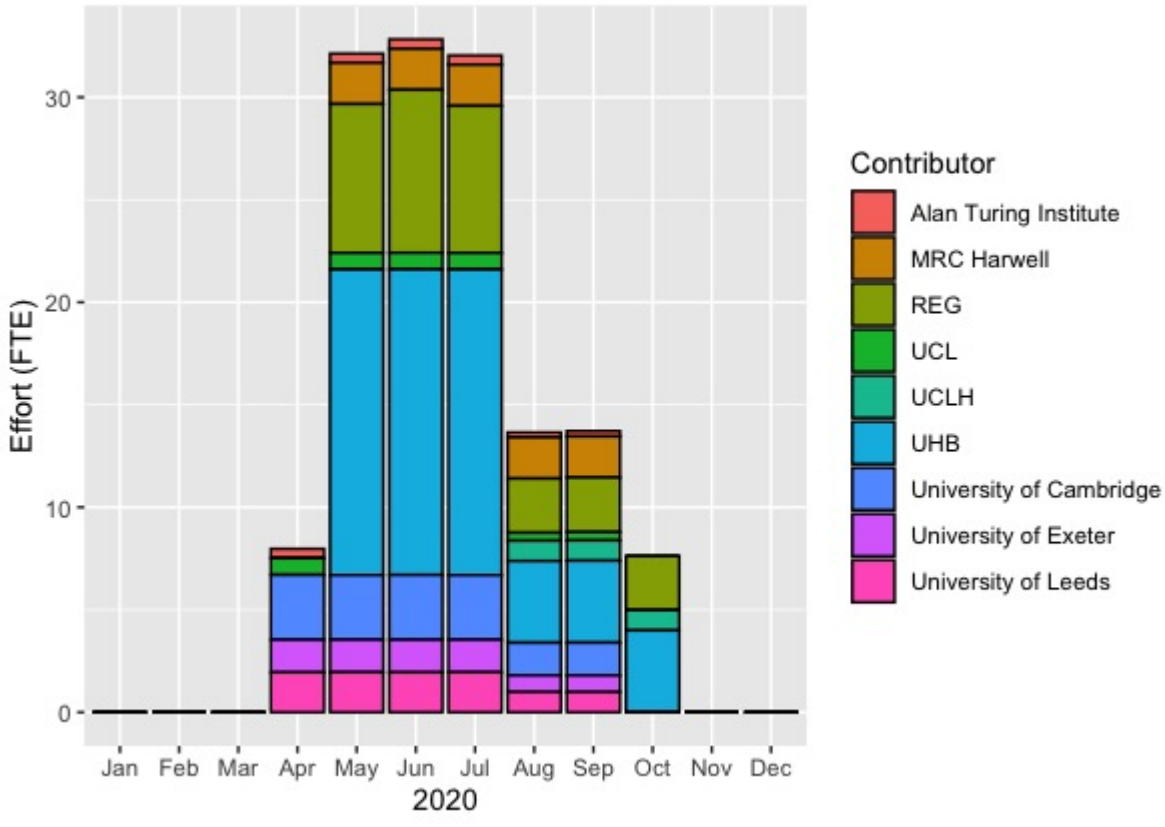
The following code will produce a bar chart which presents the partner effort per month, without black lines delineating the project partners.

```
d %>% dplyr::select(Contributor,ProjectMonth,MonthEffort) %>%
      ggplot(aes(x=factor(ProjectMonth,levels=Months),y=MonthEffort,fill=Contributor)) +
      geom_col(aes(group=Contributor),position="stack") + xlab("2020") +
      ylab("Effort (FTE)")
```



The graph can also be produced with a black line delineating the delivery partners by using the following code:

```
d %>% dplyr::select(Contributor,ProjectMonth,MonthEffort,Project)     %>%
   group_by(ProjectMonth,Contributor)                                 %>%
   summarise(Effort=sum(MonthEffort),.groups="keep")                  %>%
   ggplot(aes(x=factor(ProjectMonth,levels=Months),y=Effort,fill=Contributor)) +
   geom_col(aes(group=Contributor),position="stack",colour="black") + xlab("2020") +
   ylab("Effort (FTE)")
```

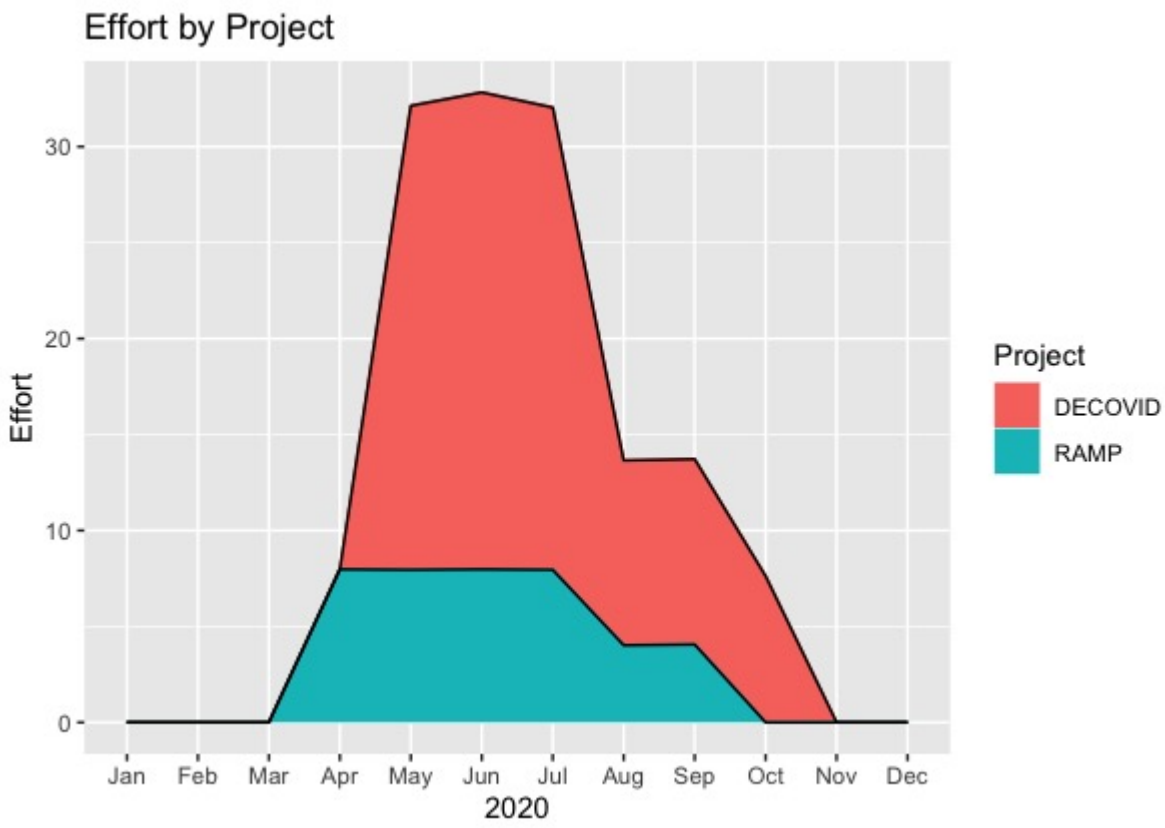


Stacked Graphs

Monthly effort by project

The following code will produce a stacked graph that presents the overall effort on each project across the project timeline.

```
d %>% dplyr::select(Contributor,ProjectMonth,MonthEffort,Project)     %>%
      group_by(ProjectMonth, Project)                                 %>%
      summarise(Effort=sum(MonthEffort),.groups="keep")               %>%
      ggplot(aes(x=factor(ProjectMonth,levels=Months),y=Effort,group=Project,fill=Project)) +
      geom_area(aes(colour=Project)) + geom_line(position="stack",colour="black") +
      xlab("2020") + ggtitle("Effort by Project")
```



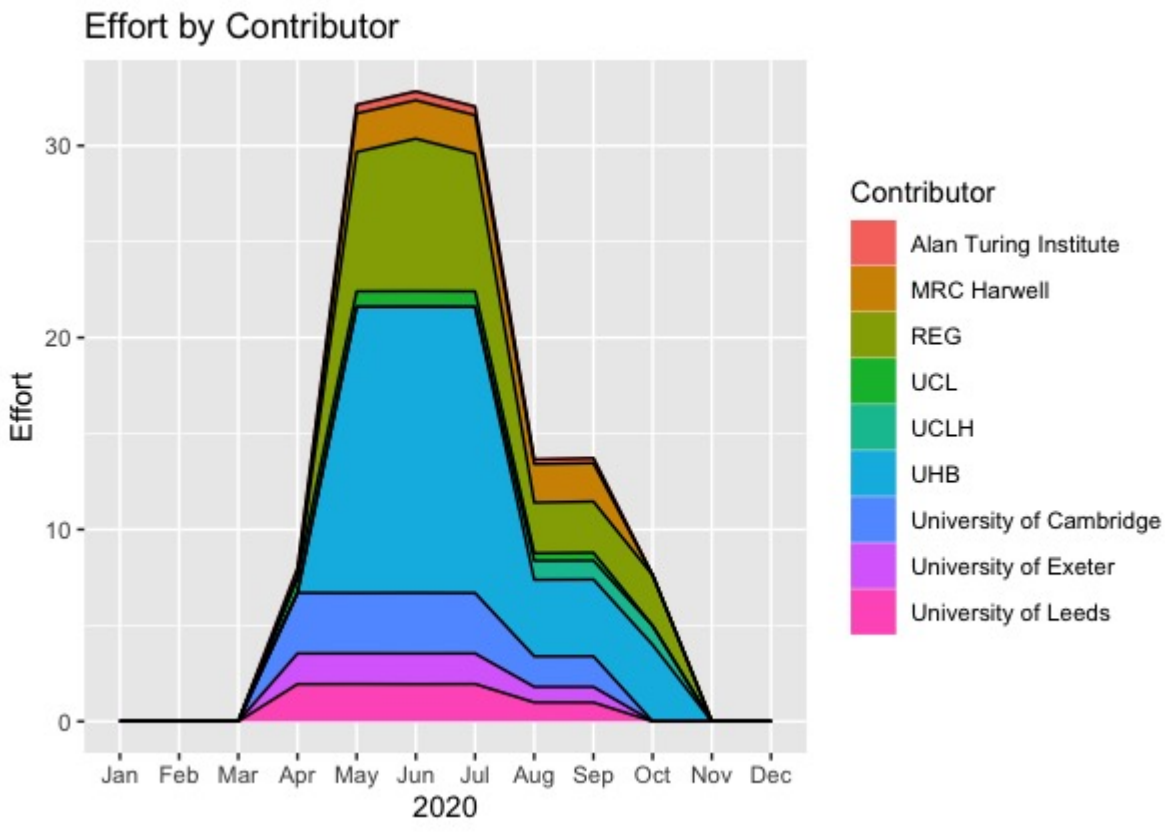
To normalise the above graph, run the following code:

```
d %>% dplyr::select(Contributor,ProjectMonth,MonthEffort,Project)  %>%
   group_by(ProjectMonth, Project)                                 %>%
   summarise(Effort=sum(MonthEffort),.groups="keep")               %>%
   ggplot(aes(x=factor(ProjectMonth,levels=Months),y=Effort,group=Project,fill=Project)) +
   geom_area(aes(colour=Project),position="fill") + geom_line(position="fill",colour="black") +
   xlab("2020") + ggtitle("Normlised effort by Project") +
   ylab("% Effort for project") + scale_y_continuous(labels = scales::percent)
```

Monthly effort by Partner

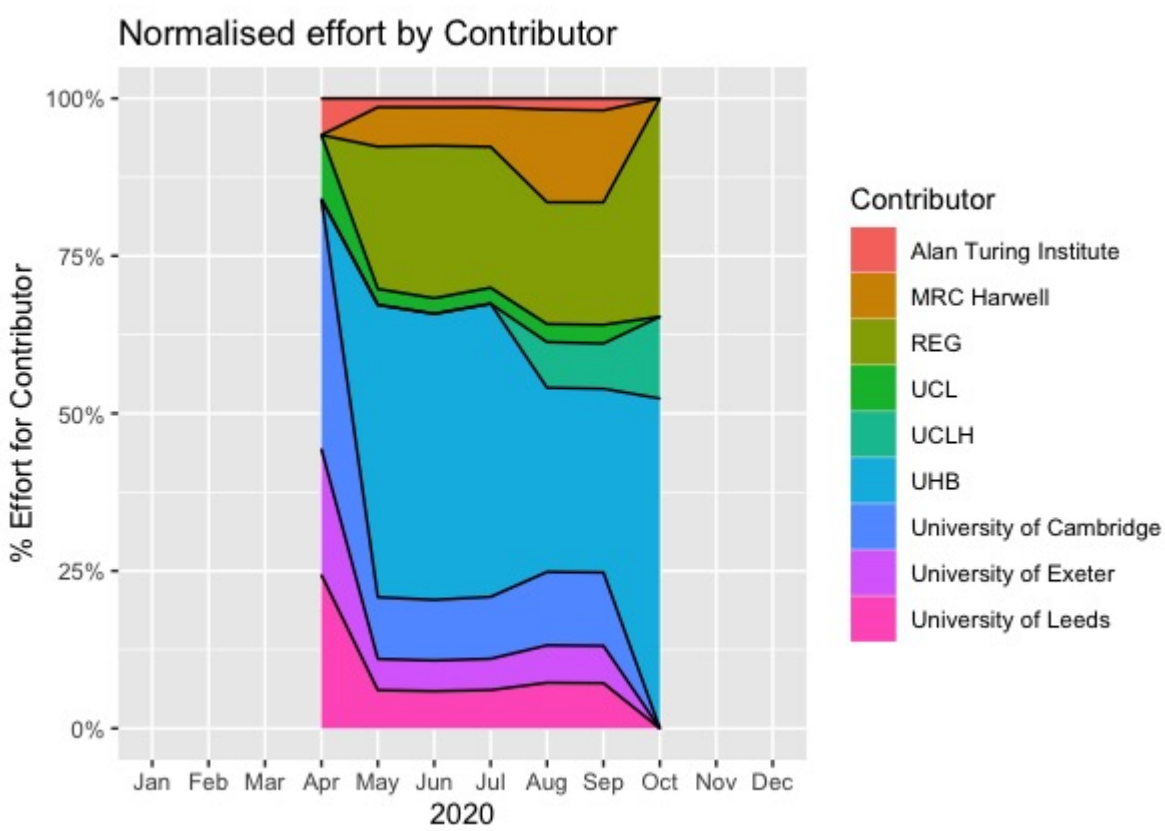
To produce a stacked graph presenting the effort by partner run the following code:

```
d %>% dplyr::select(Contributor,ProjectMonth,MonthEffort,Project)  %>%
   group_by(ProjectMonth, Project)                                 %>%
   summarise(Effort=sum(MonthEffort),.groups="keep")               %>%
   ggplot(aes(x=factor(ProjectMonth,levels=Months),y=Effort,group=Project,fill=Project)) +
   geom_area(aes(colour=Project),position="fill") + geom_line(position="fill",colour="black") +
   xlab("2020") + ggtitle("Normlised effort by Project") +
   ylab("% Effort for project") + scale_y_continuous(labels = scales::percent)
```



To normalise the above graph, run the following code:

```
group_by(ProjectMonth, Contributor,.drop=FALSE)                    %>%
summarise(Effort=sum(MonthEffort),.groups="keep")                  %>%
ggplot(aes(x=factor(ProjectMonth,levels=Months),y=Effort,group=Contributor,fill=Contributor)) +
geom_area(aes(colour=Contributor),position="fill") + geom_line(position="fill",colour="black") +
xlab("2020") + ggtitle("Normalised effort by Contributor") +
ylab("% Effort for Contributor") + scale_y_continuous(labels = scales::percent)
```

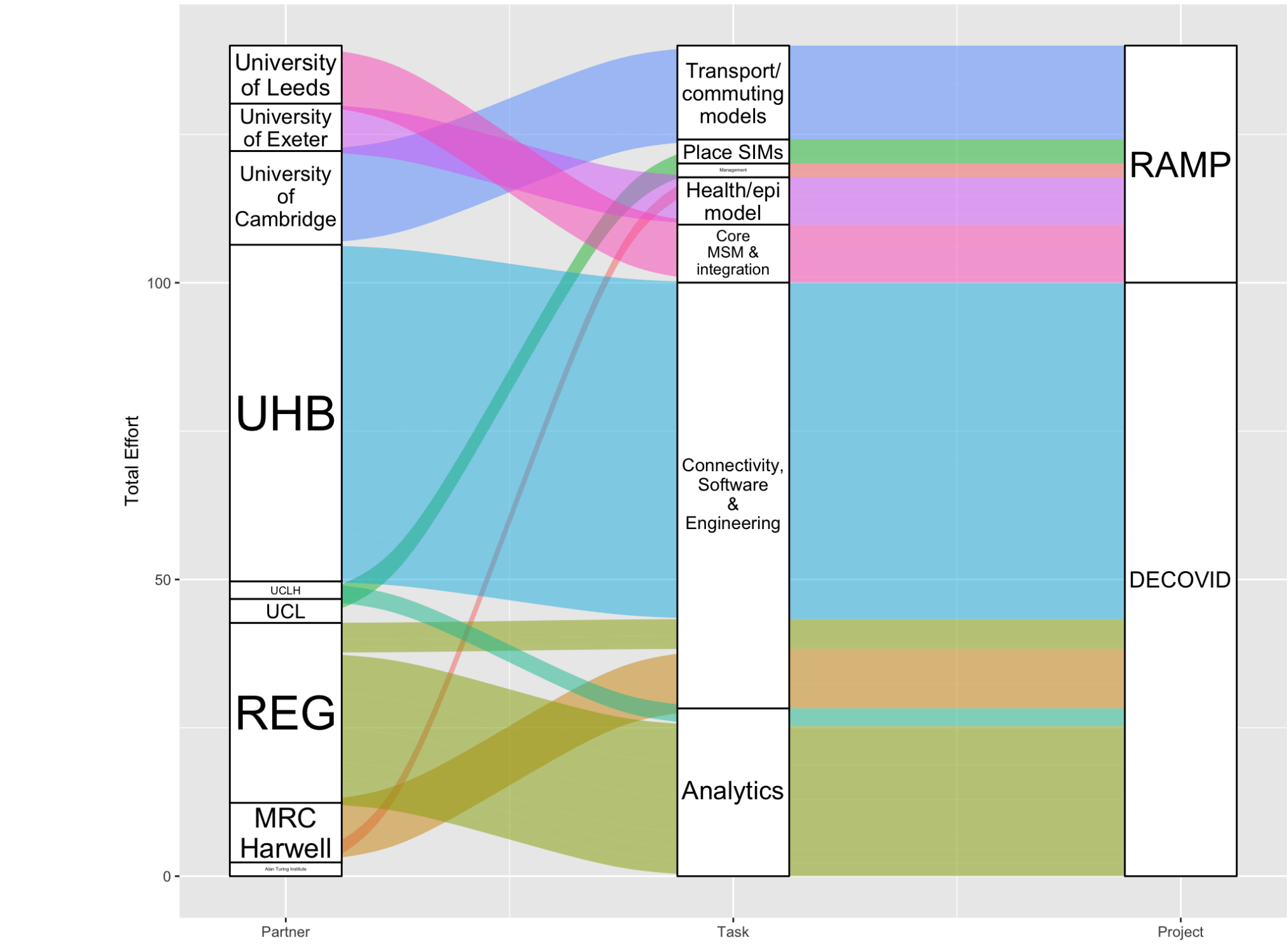


Alluvial Charts

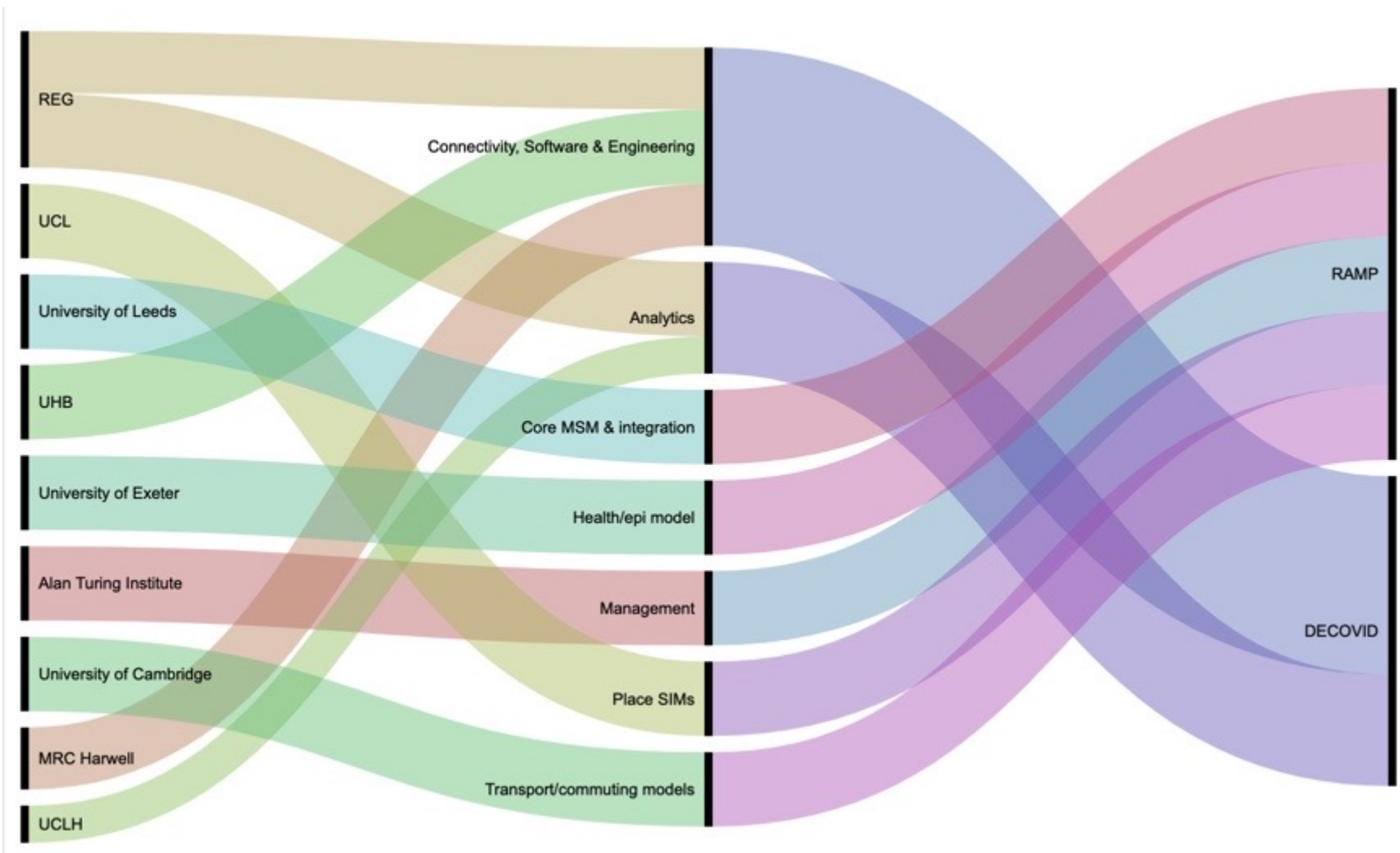
There are two options in which to generate the alluvial chart. One method is on R Studio, and the second method is on rawgraphs.io. I personally think using Raw Graphs generates a better presented graph but I will outline below how to make it on R Studio.

Run the following code to produce an alluvial chart which links together the partners, tasks and project.

```
d %>% dplyr::select(Partner=Contributor,Task, Project,MonthEffort)                  %>%
   group_by(Partner, Task, Project, MonthEffort)                                   %>%
   summarise(TotEffort=sum(MonthEffort), .groups="keep")                           %>%
   ggplot(aes(axis1=Partner, axis2=Task, axis3=Project,y=TotEffort)) +
   geom_alluvium(aes(fill=Partner), width = 0, reverse = FALSE) +
   guides(fill = FALSE) +
   geom_stratum(width = 1/4, reverse = FALSE) +
   ggfittext::geom_fit_text(aes(label = after_stat(stratum)),
                stat = "stratum", outside = TRUE,reflow=TRUE, grow=TRUE, reverse = FALSE,
                min.size=2, width = 1/4) +
   scale_x_continuous(breaks = 1:3, labels = c("Partner", "Task", "Project")) +
   ylab("Total Effort")
```



To produce an alluvial chart on rawgraphs.io follow the attached link. Raw Graphs. From here you can produce the following alluvial chart using the data.

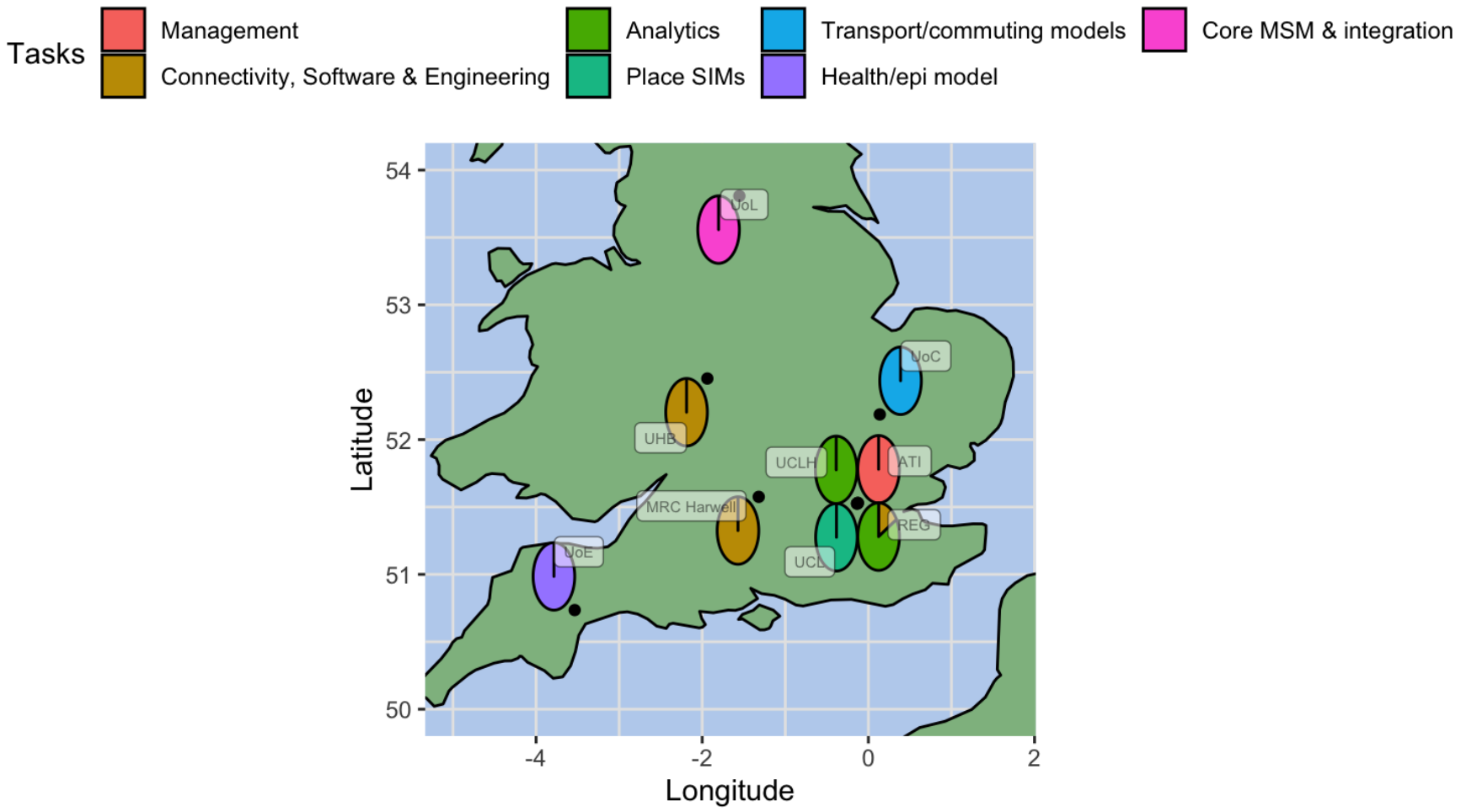


Geographic Plot

The geographic plot will map the location of each partner in the UK. For each partner there is a pie chart which will show the different tasks the partner took part in across all COVID projects.

To produce the graph, run following the code:

```
d$lat <- as.numeric(str_split_fixed(d$Contributor_Lat_Long ,",",2)[,1])
d$lon <- as.numeric(str_split_fixed(d$Contributor_Lat_Long,",",2)[,2])


ggmap(ukmap) + geom_point(data=d,aes(x=lon,y=lat))

world = map_data("world", resolution=0)


d %>% dplyr::select(Contributor, Task, MonthEffort, lon, lat)  %>%
      group_by(Contributor, lon, lat, Task,.drop = FALSE)      %>%
      summarise(TotEffort=sum(MonthEffort),.groups="drop")     %>%
      pivot_wider(names_from = Task, values_from=TotEffort, values_fill=0) -> e

e$Contributor <- gsub("Alan Turing Institute","ATI",e$Contributor)
e$Contributor <- gsub("University of Exeter","UoE",e$Contributor)
e$Contributor <- gsub("University of Leeds","UoL",e$Contributor)
e$Contributor <- gsub("University of Cambridge","UoC",e$Contributor)

e$Contributor <- factor(e$Contributor)

tasks <- names(e)[4:ncol(e)]

e <- add_column(e,lon2=e$lon,.after=3)
e <- add_column(e,lat2=e$lat,.after=4)

offset <- 0.25

e[e$Contributor == "ATI","lon2"] <- e[e$Contributor == "ATI","lon2"] + offset
e[e$Contributor == "ATI","lat2"] <- e[e$Contributor == "ATI","lat2"] + offset

e[e$Contributor == "UCL","lon2"] <- e[e$Contributor == "UCL","lon2"] - offset
e[e$Contributor == "UCL","lat2"] <- e[e$Contributor == "UCL","lat2"] - offset

e[e$Contributor == "UCLH","lon2"] <- e[e$Contributor == "UCLH","lon2"] - offset
e[e$Contributor == "UCLH","lat2"] <- e[e$Contributor == "UCLH","lat2"] + offset

e[e$Contributor == "REG","lon2"] <- e[e$Contributor == "REG","lon2"] + offset
e[e$Contributor == "REG","lat2"] <- e[e$Contributor == "REG","lat2"] - offset

e[e$Contributor == "UoC","lon2"] <- e[e$Contributor == "UoC","lon2"] + offset
e[e$Contributor == "UoC","lat2"] <- e[e$Contributor == "UoC","lat2"] + offset

e[e$Contributor == "UoE","lon2"] <- e[e$Contributor == "UoE","lon2"] - offset
e[e$Contributor == "UoE","lat2"] <- e[e$Contributor == "UoE","lat2"] + offset

e[e$Contributor == "UHB","lon2"] <- e[e$Contributor == "UHB","lon2"] - offset
e[e$Contributor == "UHB","lat2"] <- e[e$Contributor == "UHB","lat2"] - offset

e[e$Contributor == "MRC Harwell","lon2"] <- e[e$Contributor == "MRC Harwell","lon2"] - offset
e[e$Contributor == "MRC Harwell","lat2"] <- e[e$Contributor == "MRC Harwell","lat2"] - offset

e[e$Contributor == "UoL","lon2"] <- e[e$Contributor == "UoL","lon2"] - offset
e[e$Contributor == "UoL","lat2"] <- e[e$Contributor == "UoL","lat2"] - offset

ggplot(data=world, aes(x=long, y=lat, group=group)) +
   geom_polygon(data=world, aes(x=long, y=lat), fill="darkseagreen", color="black") +
coord_quickmap(xlim=c(-5.0, 1.68), ylim=c(50,54)) +
ylab("Latitude") + xlab("Longitude") +
geom_point(aes(x=lon,y=lat),data=e, inherit.aes = FALSE) +
geom_scatterpie(data = e,aes(x=lon2, y=lat2, group=Contributor), cols=tasks ,pie_scale = 3,
                legend_name ="Tasks",sorted_by_radius = TRUE) +
geom_label_repel(aes(x=lon2,y=lat2,label= Contributor), data=e,
                 size=2, inherit.aes = FALSE, force=5, alpha=0.5,segment.size = 0) +
theme(
   panel.background = element_rect(fill="lightsteelblue2"),
   panel.grid.minor = element_line(colour="grey90", size=0.5),
   panel.grid.major = element_line(colour="grey90", size=0.5),
   legend.position = "top")
```



Shortening partner names

As some of the partners names are too long to fit on the geographic plot without taking up too much space, they need to be shortened. To do this use the following code, but replace 'Alan Turing Institute' and 'ATI' with the name of the new partner and what you want to shorten it to.

```
e$Contributor <- gsub("Alan Turing Institute","ATI",e$Contributor)
```

Manually giving the co-ords an offset

So the pie charts are not on top of one another, you need to set an offset. This can be doing using the following code, but the 'ATI' needs to be replaced with the name of the new partner. If you have abbreviated the partners name earlier, the abbreviated version needs to be used here.

```
e[e$Contributor == "ATI","lon2"] <- e[e$Contributor == "ATI","lon2"] + offset
e[e$Contributor == "ATI","lat2"] <- e[e$Contributor == "ATI","lat2"] + offset
```