

# QTM 151

## Week 11 – maps

---

---

Umberto Mignozzetti

# Recap

We've learned:

- Programming methods
- `dplyr *_join` methods: joining data
- `tidyr` methods: reshape datasets
- `forcats` methods: working with categorical variables
- `lubridate` methods: to work with dates
- `qplot` methods: to create quick plots
- `plotly` methods: create beautiful interactive plots.

**Great job!!** Do you have any questions about these contents?

Reminder: **no class next Friday**. I'll be at a conference.

Today we will learn how to make **maps in R**. Our GitHub page is:

<https://github.com/umbertomig/qtm151>

# Getting Started

---

# Getting Started: loading packages

```
# Loading tidyverse
```

```
library(tidyverse)
```

```
## — Attaching packages ————— tidyverse_2023.09.03
```

```
## ✓ ggplot2 3.3.5      ✓ purrr 0.3.4
```

```
## ✓ tibble 3.1.6       ✓ dplyr 1.0.8
```

```
## ✓ tidyr 1.2.0        ✓ stringr 1.4.0
```

```
## ✓ readr 2.1.2        ✓ forcats 0.5.1
```

```
## Warning: package 'tidyr' was built under R version 4.1.2
```

```
## Warning: package 'readr' was built under R version 4.1.2
```

```
## Warning: package 'dplyr' was built under R version 4.1.2
```

```
## — Conflicts ————— tidyverse_2023.09.03
```

```
## x dplyr::filter() masks stats::filter()
```

# Getting Started: loading data

```
GAdat←read.csv("https://raw.githubusercontent.com/umbertomig/qtm1  
GAdat$County ← tolower(GAdat$County)  
head(GAdat, 2)
```

```
##      County Population  
## 1  appling      18236  
## 2 atkinson      8375
```

# ggmap

---

# ggmap package

The package makes it easy to retrieve raster map tiles from popular online mapping services like Stamen Maps and Google Maps

It is easy to plot the info using the ggplot2 framework.

Suppose we want to plot the average arrival delay of NY flights.

```
data1←flights %>%  
  drop_na() %>%  
  group_by(dest) %>%  
  summarise(average=mean(arr_delay)) %>%  
  left_join(airports, by=c("dest"="faa"))  
head(data1, 2)
```

# ggmap package

```
us ← c(left = -125, bottom = 25.75, right = -67, top = 49)
map ← get_stamenmap(us, zoom = 5, maptype = "toner-lite")
ggmap(map) # create a layer US map
```

```
ggmap(map) + geom_point(data=data1,
  aes(x=lon, y=lat, color=average, size=average), na.rm = T) +
  scale_color_gradient(low = "green", high="darkblue")
```

**Your turn:** do the same graph, but with the variable `distance`, instead of `arr_delay`. You should learn smt trivial!



# ggmap package

Let's plot the number of flights from each destination.

```
data2<-flights %>%  
  drop_na() %>%  
  group_by(dest) %>%  
  summarise(sum=n()) %>%  
  left_join(airports, by=c("dest"="faa"))  
head(data2, 2)
```

# ggmap package

```
us ← c(left = -125, bottom = 25.75, right = -67, top = 49)
map ← get_stamenmap(us, zoom = 5, maptype = "toner-lite")
ggmap(map)
```

```
ggmap(map) + geom_point(data=data2,
                        aes(x=lon, y=lat, color=sum, size=sum), na.rm = T) +
  scale_color_gradient(low = "blue", high="red")
```

**Your turn:** Plot the number of flights for the *Delta* (DL) carrier only.

maps package

---

# maps package

It allows us to turn data from the maps into a data frame suitable for plotting with ggplot.

The structure of the data needed:

- `long`: longitude.
- `lat`: latitude.
- `order`: shows in which order ggplot should “connect the dots”
- `region` and `subregion`: tell what region or subregion a set of points surrounds.
- `group`: ggplot2’s functions can take a group argument, which controls whether lines should connect adjacent points.
  - If they are in the same group, they get connected, but if they are in different groups, they don't.

# maps package

Plot the USA map using *geom\_polygon()*.

*geom\_polygon()* draws lines between points and “closes them up” (i.e. draws a line from the last point back to the first point).

You have to map the group aesthetic to the group column.

```
# map_data function in maps package
states←map_data("state")
head(states)
qplot(long, lat, data=states)
qplot(long, lat, data=states, geom="path")
```

- And they look ugly...

# maps package

To make them look better, we need to group!

```
qplot(long, lat, data=states, geom="path", group=group)
```

```
qplot(long, lat, data=states, geom="polygon", group=group)
```

```
# color for boarder lines
```

```
ggplot(states)+
```

```
  geom_polygon(aes(x=long, y=lat, group=group), color="red")
```

```
# fill for inside color
```

```
qplot(long, lat, data=states, geom="polygon",  
  group=group, fill=long, color="red")
```

- And they look much better.

# maps package

We can also turn off the color legend:

```
states ← map_data("state")
ggplot(data = states) +
  geom_polygon(aes(x = long, y = lat, fill = region, group = group),
    coord_fixed(1.3) +
    guides(fill=FALSE)
```

# maps package

We can also plot a subregion:

```
west_coast ← subset(states, region %in% c("california", "oregon", "wa")  
  
ggplot(data = west_coast) +  
  geom_polygon(aes(x = long, y = lat), fill = "palegreen",  
    color = "black")
```



# maps package

But we can do better:

- `group()`
- `coord_fixed()`: it fixes the relationship between one unit in the y-direction and one in the x-direction.
  - Every y unit was 1.3 times longer than an x unit. The plot came out looking good.

```
ggplot(data = west_coast) +  
  geom_polygon(aes(x = long, y = lat, group = group),  
    fill = "palegreen", color = "black") +  
  coord_fixed(1.3)
```

**Your Turn:** Do the same plot for a different set of states of your choice.

# maps package

And to plot only one state, we need to filter this state out of the dataset:

```
states <- map_data("state")
```

```
ga_df <- states %>%  
  filter(region == "georgia")
```

```
ggplot(data = ga_df) +  
  geom_polygon(aes(x = long, y = lat), fill = "palegreen", color = "black")
```

```
ggplot(data = ga_df) +  
  geom_polygon(aes(x = long, y = lat, group = group), fill = "palegreen",  
  coord_quickmap())
```

**Your Turn:** Do the same plot for a different US state.

# maps package

Using `theme_void()`: you can make a plot with no background.

```
ga_base <- ggplot(data = ga_df, mapping = aes(x = long, y = lat, group = group)) +  
  coord_quickmap() +  
  geom_polygon(color = "black", fill = "gray")  
  
ga_base + theme_void()
```

# maps package

The package maps also have subregions and subdivisions within a given country.

This is helpful to plot counties in Georgia:

```
county_df ← map_data("county") %>% filter(region = "georgia")
state_df ← map_data("state") %>% filter(region = "georgia")

ga_base + theme_void() +
  geom_polygon(data = county_df, fill = NA, color = "white") +
  geom_polygon(color = "black", fill = NA)
```

# maps package

And in the GAdat, we have the population by county. We can plot this in the Georgia map:

```
county_df$subregion ← replace(county_df$subregion, county_df$subregion  
mapdat ← left_join(GAdat, county_df, by = c("County" = "subregion"))  
  
p ← ggplot(mapdat, aes(long, lat, group = group)) +  
  geom_polygon(aes(fill = Population), color = "yellow") +  
  scale_fill_gradient(low = "blue", high = "red") +  
  geom_polygon(data = state_df, colour = "black", fill = NA) +  
  theme_void() +  
  coord_fixed(1.2)
```

p

# maps package

And for an alternative map:

```
ggplot(mapdat, aes(long, lat, group = group)) +  
  geom_polygon(aes(fill = Population, color="yellow"),  
    colour = alpha("red", 1/2)) +  
  geom_polygon(data = state_df, colour = "black", fill = NA) +  
  theme_void() + coord_fixed(1.2) +  
  scale_fill_gradientn(colours = rev(rainbow(7)),  
    breaks = c(2, 4, 10, 100, 1000, 10000), trans = "log10")
```

Questions?

---

Have a great weekend!

---