CountyStat PDF Template

Wed December 14, 2022 10:31 AM

Contents

Getting Started

For many of the basics in RMarkdown the RStudio introduction is very worth while. Almost every R visualization output is compatible with it.

For this report standard if other images need to be replaced with others that is fine.

Common Standards

Loading Data

Database Connections

```
require(DBI)

wh_host <- Sys.getenv('WH_HOST')
wh_db <- Sys.getenv('WH_DB')
wh_user <- Sys.getenv('WH_USER')
wh_pass <- Sys.getenv('WH_PASS')

# Connect to the CountyStat DataWarehouse
wh_con <- dbConnect(odbc::odbc(),
                    driver = "{ODBC Driver 17 for SQL Server}",
                    server = wh_host,
                    database = wh_db,
                    UID = wh_user,
                    pwd = wh_pass,
                    Trusted_Connection= "yes")
```

Always keep your credentials for the DataWarehouse in an `.env` file. An example is in this repo, but in normal circumstances this file should be in the `.gitignore` file.

To use DBI check out their documentation, but for the most part you will be using either the `dbGetQuery()` or `dbReadTable()` function.

Reading Flat Files

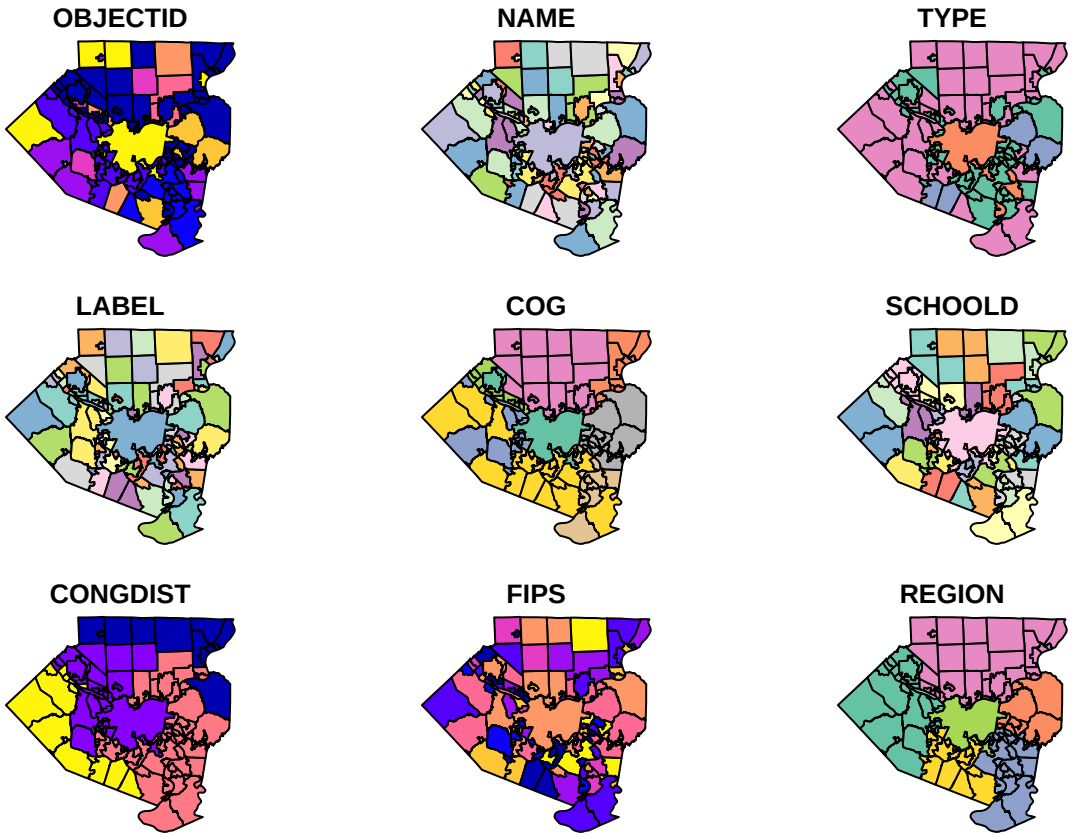
For flat files, `readr` will do well for most CSV's and other basic file formats, `readxl` is best for excel documents, `jsonlite` is ideal for JSON documents, and `xml2` will do if you have the unfortunate circumstance of dealing with that format.

```
require(readr)
require(readxl)
require(jsonlite)
require(xml2)
```

Spatial Data

The easiest method for reading data and mapping in either `ggplot2` or `ggmap` is the sf package. When looking for a spatial dataset, the first option is always what is available from the County ESRI team on their GIS Open Data site. If you can't find what you are looking for, and it is something you think the County should have access to email their team at GISHelp@AlleghenyCounty.US.

```
require(sf)
require(httr)

munis <- read_sf(content(GET('https://opendata.arcgis.com/datasets/9de0e9c07af04e638dbc9cb9070962c2_0.geojso

plot(munis)
```



Census Data

The easiest way to use the Census API in R is with the tidycensus package. First you have to sign-up for an api key, but its a free and easy process.

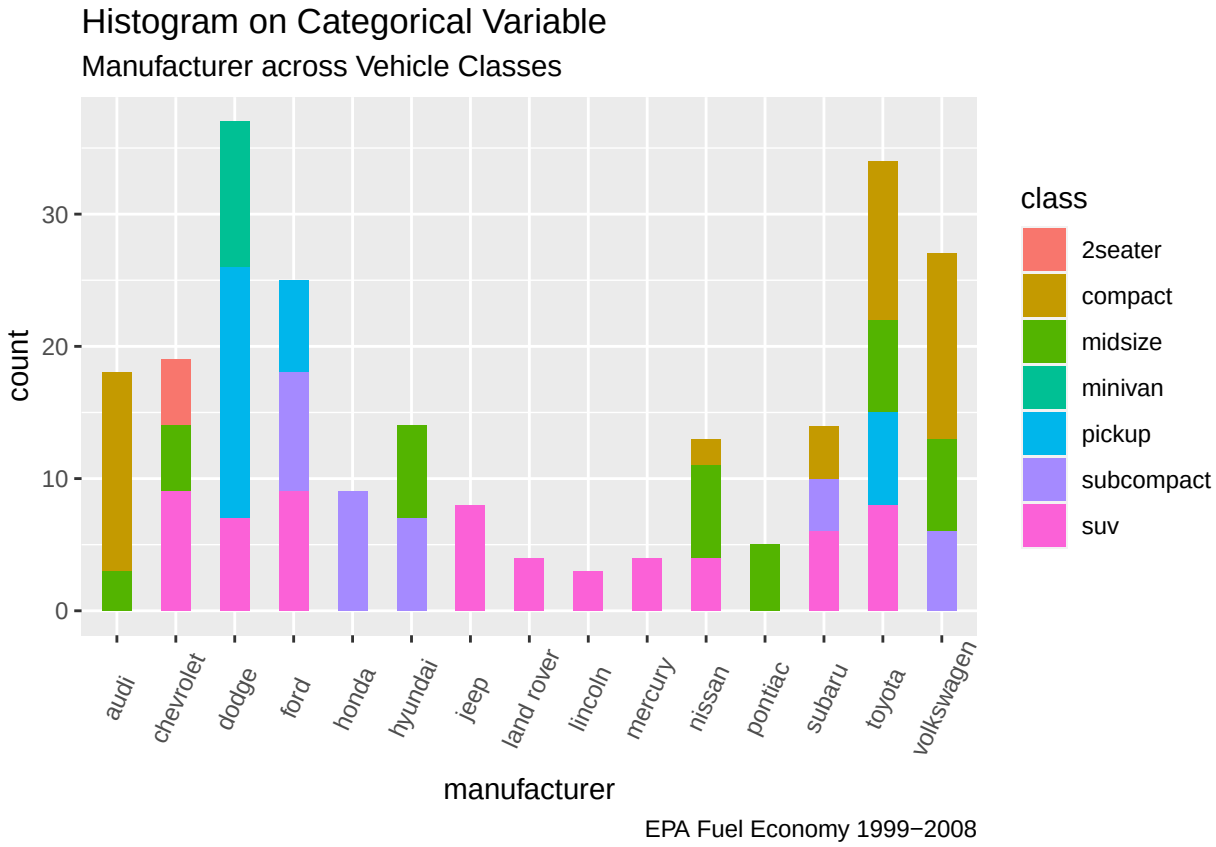
```
require(tidycensus)

census_api_key(Sys.getenv("census"))

v19 <- load_variables(2019, "acs5", cache = TRUE)

alco_muni_pop <- get_acs("county subdivision",
                         state = 'PA',
                         county = 'Allegheny',
                         year = 2019,
                         variables = 'B01003_001',
                         geometry = T) # Includes geometry column for mapping
```

```
##   |                                                                              |
```

Charts and Graphs

The easiest package for quickly creating charts and graphs in R is ggplot2. Once of the best resources for ggplot2 visualizations is the Topic 50 ggplot2 Visualizations from Selva Prabhakaran. This includes full code and examples using you can access directly from R. This site also has a good introduction to ggplot2 if you find the one from RStudio insufficient.

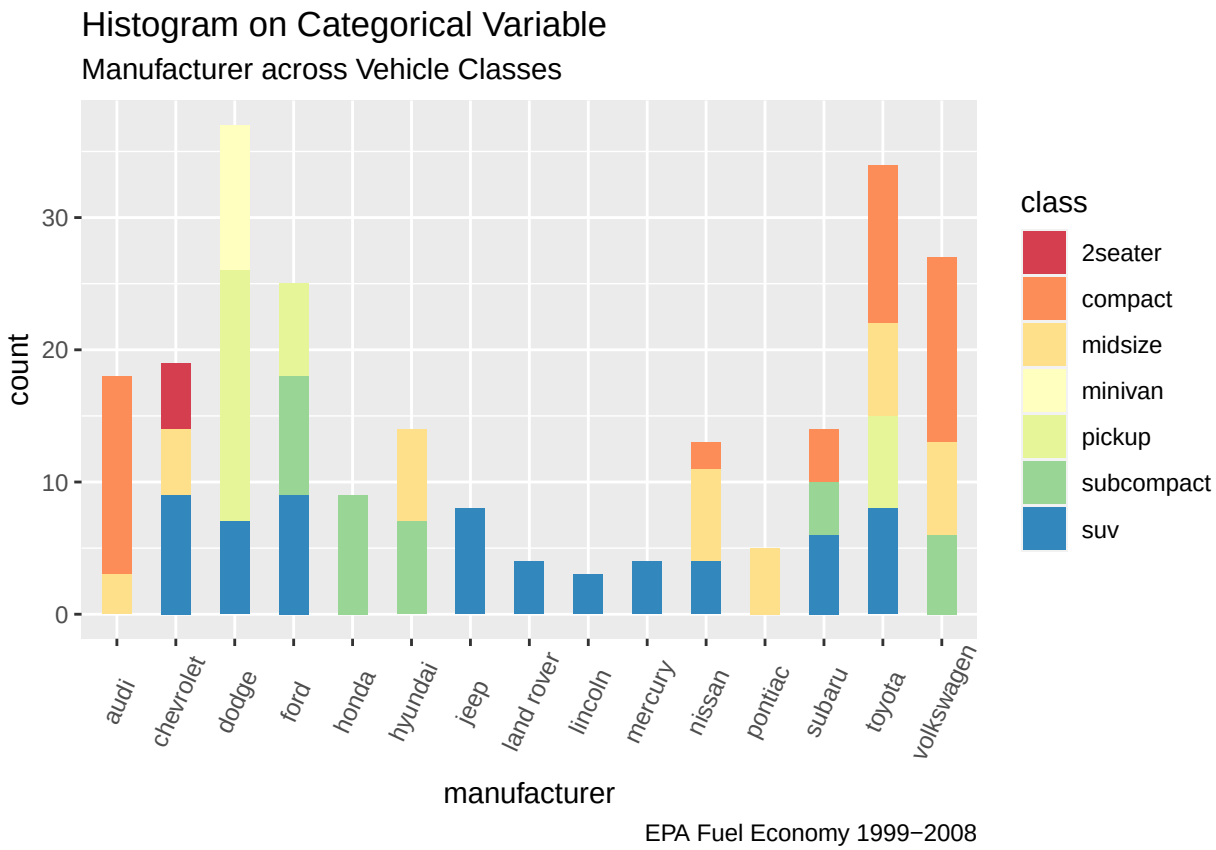
```
require(ggplot2)

g <- ggplot(mpg, aes(manufacturer)) + geom_bar(aes(fill=class), width = 0.5) +
  theme(axis.text.x = element_text(angle=65, vjust=0.6)) +
  labs(title="Histogram on Categorical Variable",
       subtitle="Manufacturer across Vehicle Classes",
       caption = "EPA Fuel Economy 1999-2008")

g
```



Colors

A great resource for color palettes is colorbrewer, which is easy to implement in ggplot2.

```
g + scale_fill_brewer(palette = "Spectral")
```



Tables

The best function for PDF table `kable()`. CountyStat has a standard table format to start with. The kableExtra package greatly expands the options you have to format your `kable()` tables. Below is the standard CountyStat format, but Analysts should feel free to use the wide variety of features the `kableExtra` library provides to improve readability and convey the message you are trying to deliver to your audience.

```
require(kableExtra)
library(palmerpenguins)

select(penguins, 1:5) %>%
  sample_n(15) %>% # Select 15 random rows
  kbl(caption = "Palmer Archipelago") %>%
  kable_styling(font_size = 12, latex_options = "HOLD_position") %>%
  row_spec(0, bold = T, background = "#008080", color = "white") # Special formatting for the top row of the
```

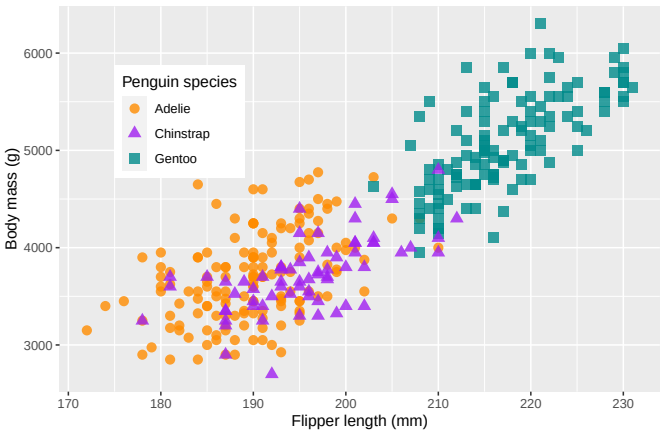
Table 1: Palmer Archipelago

species	island	bill_length_mm	bill_depth_mm	flipper_length_mm
Gentoo	Biscoe	43.3	14.0	208
Adelie	Biscoe	42.0	19.5	200
Gentoo	Biscoe	45.3	13.7	210
Adelie	Dream	40.2	20.1	200
Adelie	Biscoe	40.6	18.8	193
Gentoo	Biscoe	46.8	14.3	215
Adelie	Biscoe	36.5	16.6	181
Adelie	Dream	38.1	17.6	187
Gentoo	Biscoe	50.0	15.3	220
Adelie	Biscoe	45.6	20.3	191
Adelie	Dream	43.2	18.5	192
Adelie	Biscoe	41.6	18.0	192
Adelie	Biscoe	35.0	17.9	190
Chinstrap	Dream	42.5	16.7	187
Gentoo	Biscoe	46.8	16.1	215

Two Column Content

Sometimes you may want to include content next to other content. For example, say you want an explanation of the chart on the right. You could put that text between `\btwocol` and a `\columnbreak` for the content that goes on the right. To go back to normal page setups add a line with `\etwocol`. You can put anything on either side of the content. The code for the plot on the right is echoed to make the PDF render more true to form.

Penguin size, Palmer Station LTER



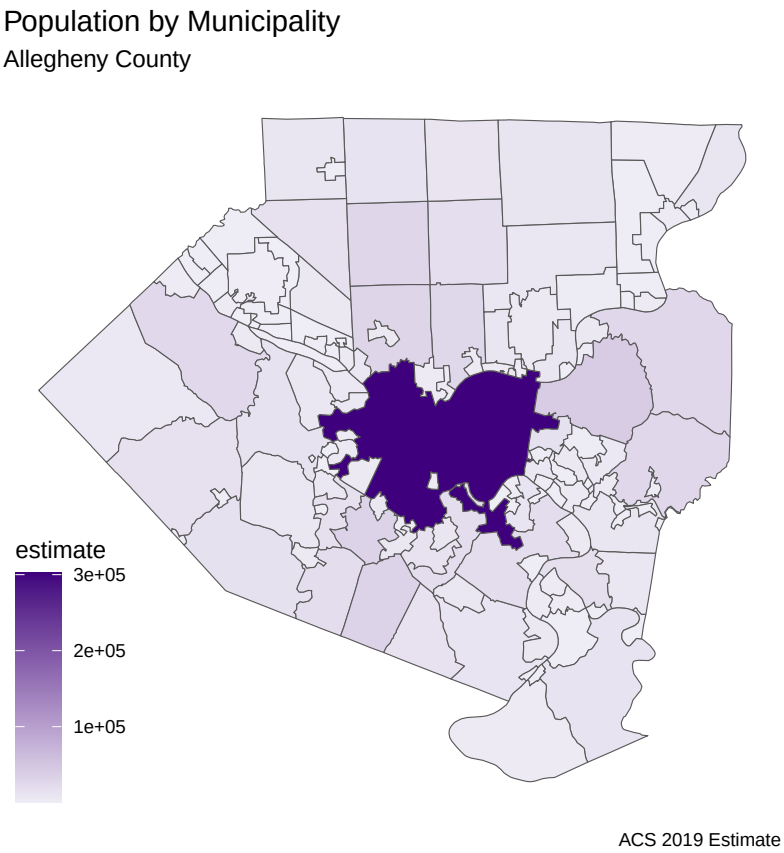
Mapping

ggplot2 map

If you don't need a base map because the shapefile contains enough information for your audience about location you can just use `ggplot2`.

```
require(ggthemes)

ggplot(data = alco_muni_pop, aes(fill = estimate)) +
  geom_sf() +
  theme_map() +
  labs(title = "Population by Municipality",
       subtitle = "Allegheny County",
```

```
       caption = "ACS 2019 Estimate") +
  scale_fill_gradient(low = '#efedf5', high = '#3f007d')
```



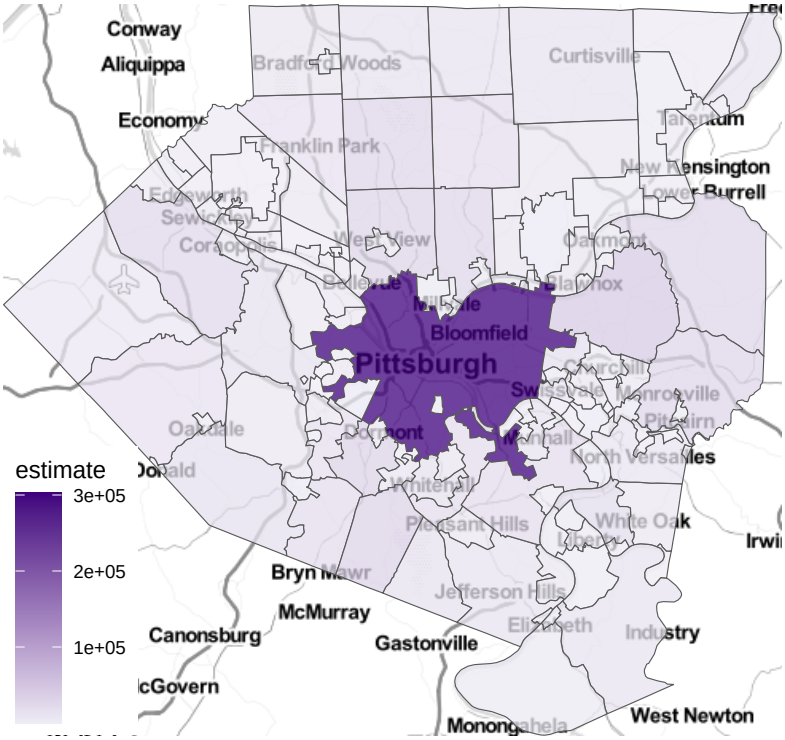
ggmap

ggmap works if you need to include a base map for context. Like leaflet the `Stamen.Toner` base map is suggested.

```
require(ggmap)
require(ggthemes)

bbox <- st_bbox(munis)
bbox_trans <- c(left = bbox[[1]], bottom = bbox[[2]], right = bbox[[3]], top = bbox[[4]])

get_stamenmap(bbox_trans, maptype = "toner-lite") %>%
  ggmap() +
  theme_map() +
  labs(title = "Population by Municipality",
       subtitle = "Allegheny County",
       caption = "ACS 2019 Estimate") +
  geom_sf(data = alco_muni_pop, aes(fill = estimate), inherit.aes = FALSE, alpha = .75) +
  scale_fill_gradient(low = '#efedf5', high = '#3f007d')
```

Population by Municipality

Allegheny County



ACS 2019 Estimate

Landscaped Pages

Sometimes a horizontal page is insufficient for the type of plot you are trying to provide to users. In these instances using the `lscape` Latex package allows you to flip the pages between the `\blandscape` and `\elandscape` commands. You will want to play around with the `fig.width` and `fig.height` for these figures to get the best settings for your plot.

This command is best for when you need to switch the orientation back and forth during the report. If your report will be in a landscape format throughout the document just add `classoption: landscape` in the YAML header.

```
ggplot(mpg, aes(x = cty, y = hwy, colour = manufacturer)) +
  geom_point() +
  labs(title = "Highway vs City MPG by Manufacturer",
       x = "City MPG",
       y = "Highway MPG",
       caption = "EPA Fuel Economy 1999-2008",
       colour = "Manufacturer") +
  theme_bw()
```