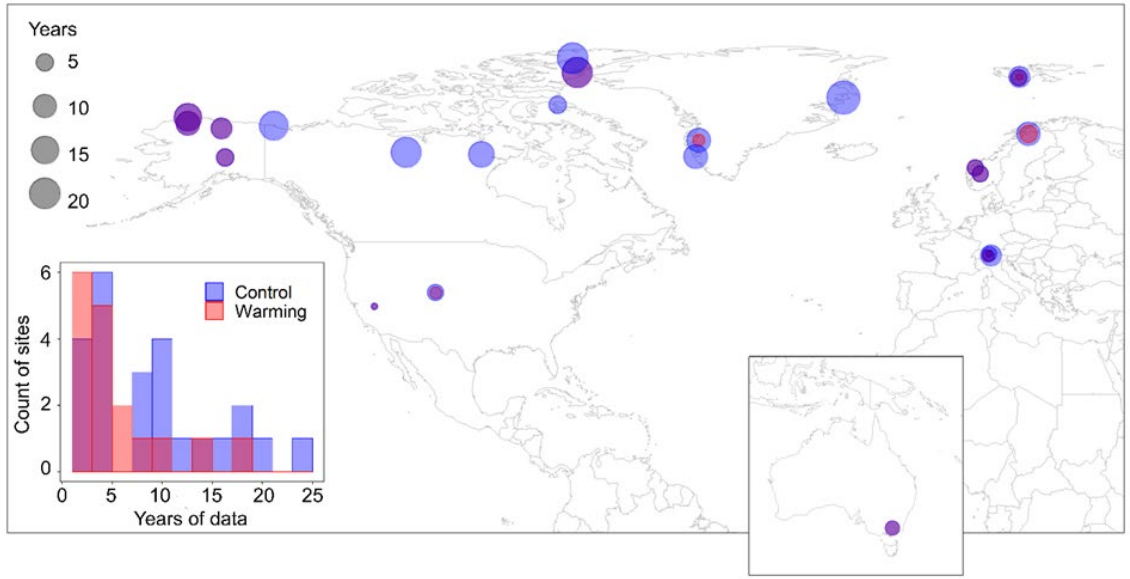
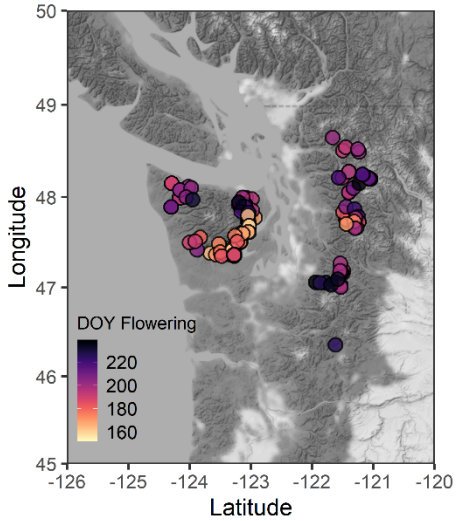
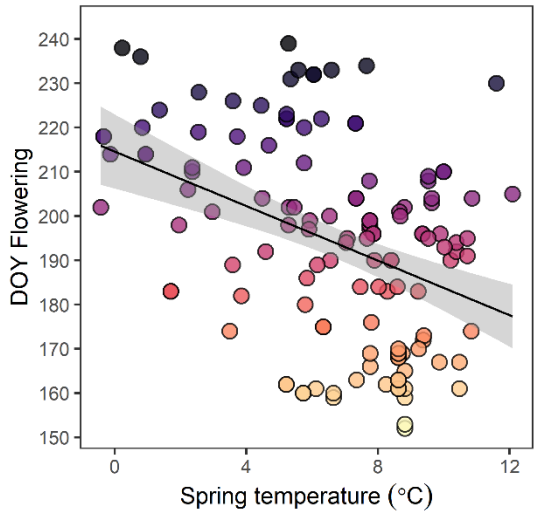
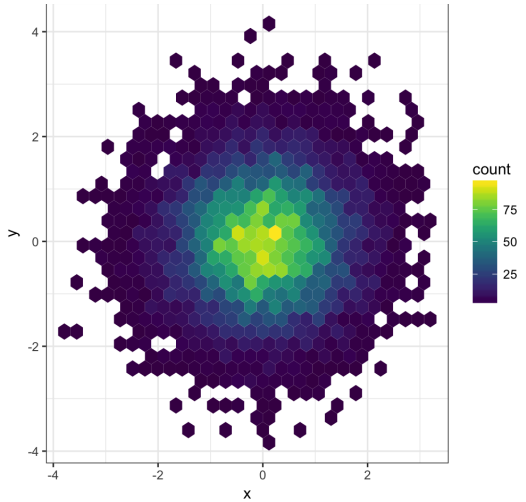
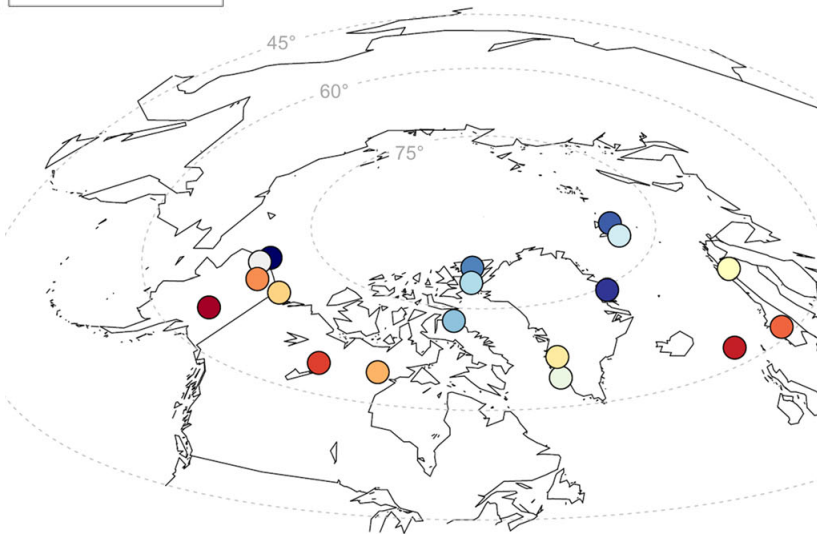
Making beautiful graphics in R

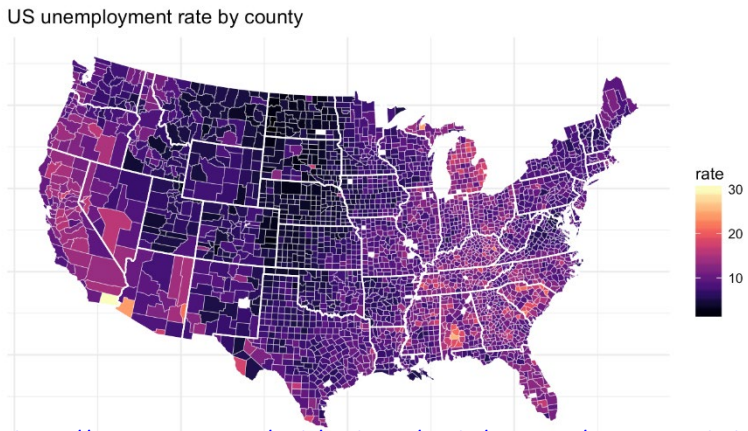












https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html

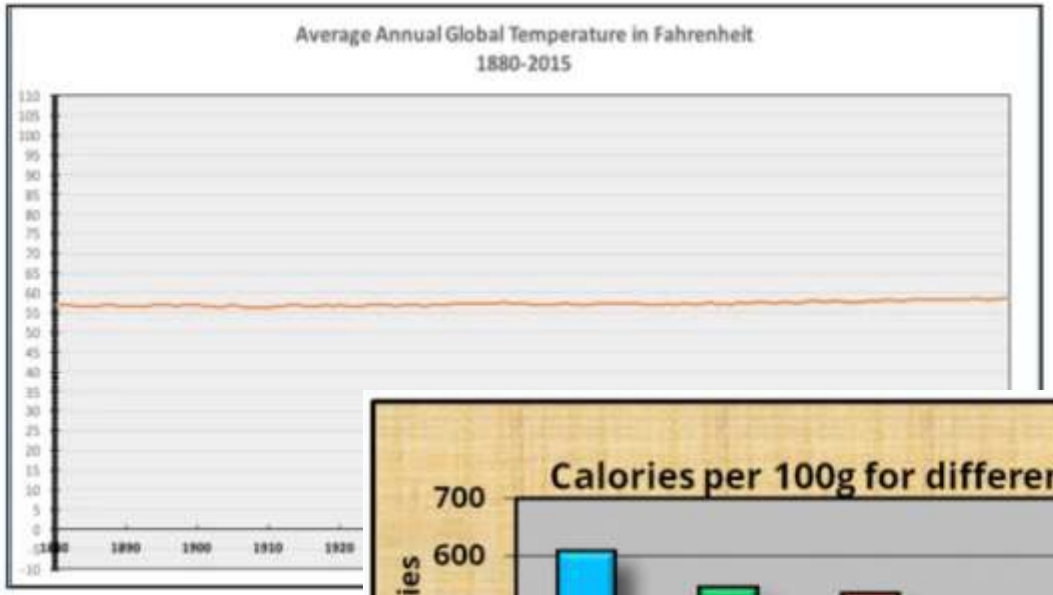
Janet Prevéy – USGS Fort Collins Science Center

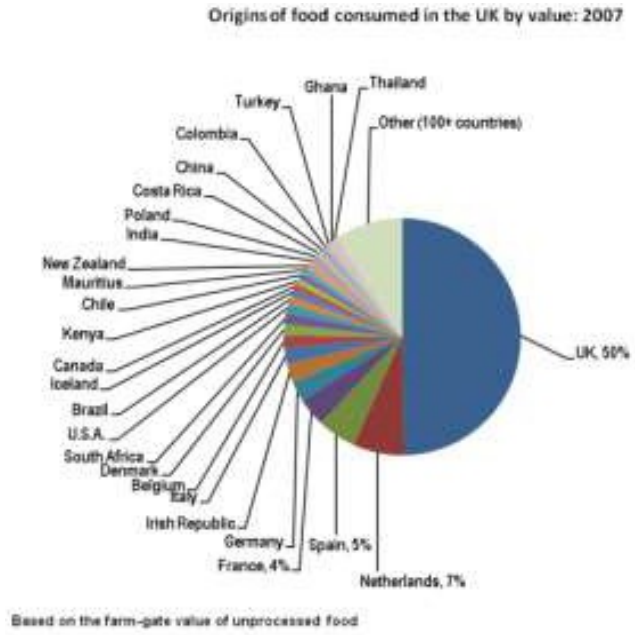
Why?

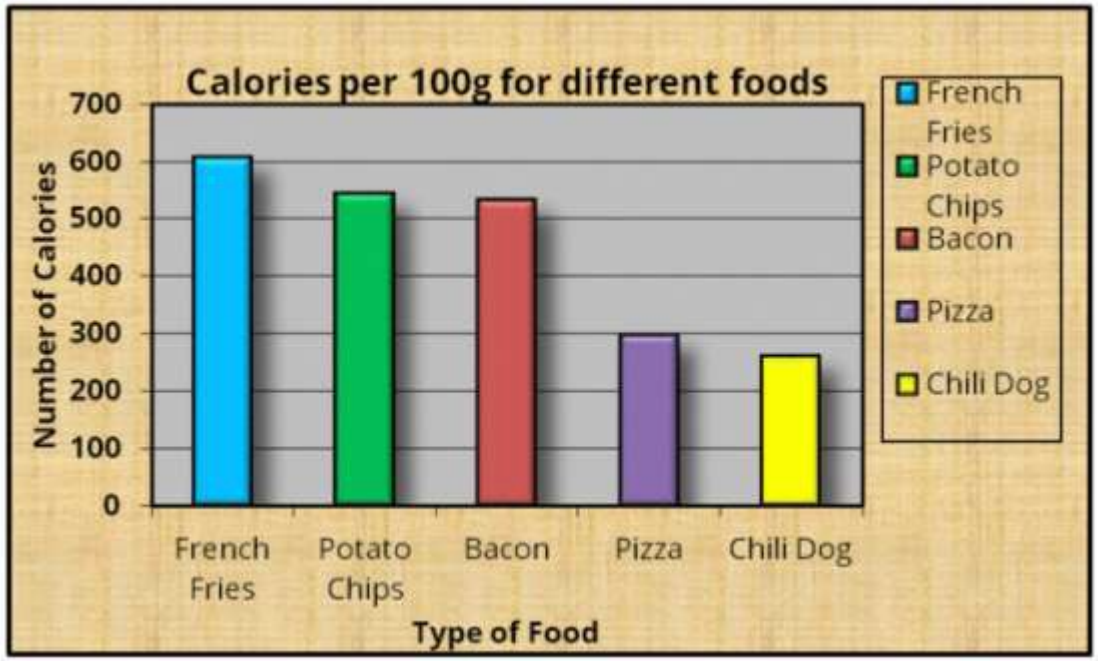
- One graph says a thousand words!
- Stand out when giving presentations
- Clear, consistent graphics and symbols help tell your research story
- Templates can be used again and again to quickly produce quality graphics

Bad graphics

- Itsy bitsy axis labels and numbers
- Incorrect scaling
- Background noise
- Too much clutter

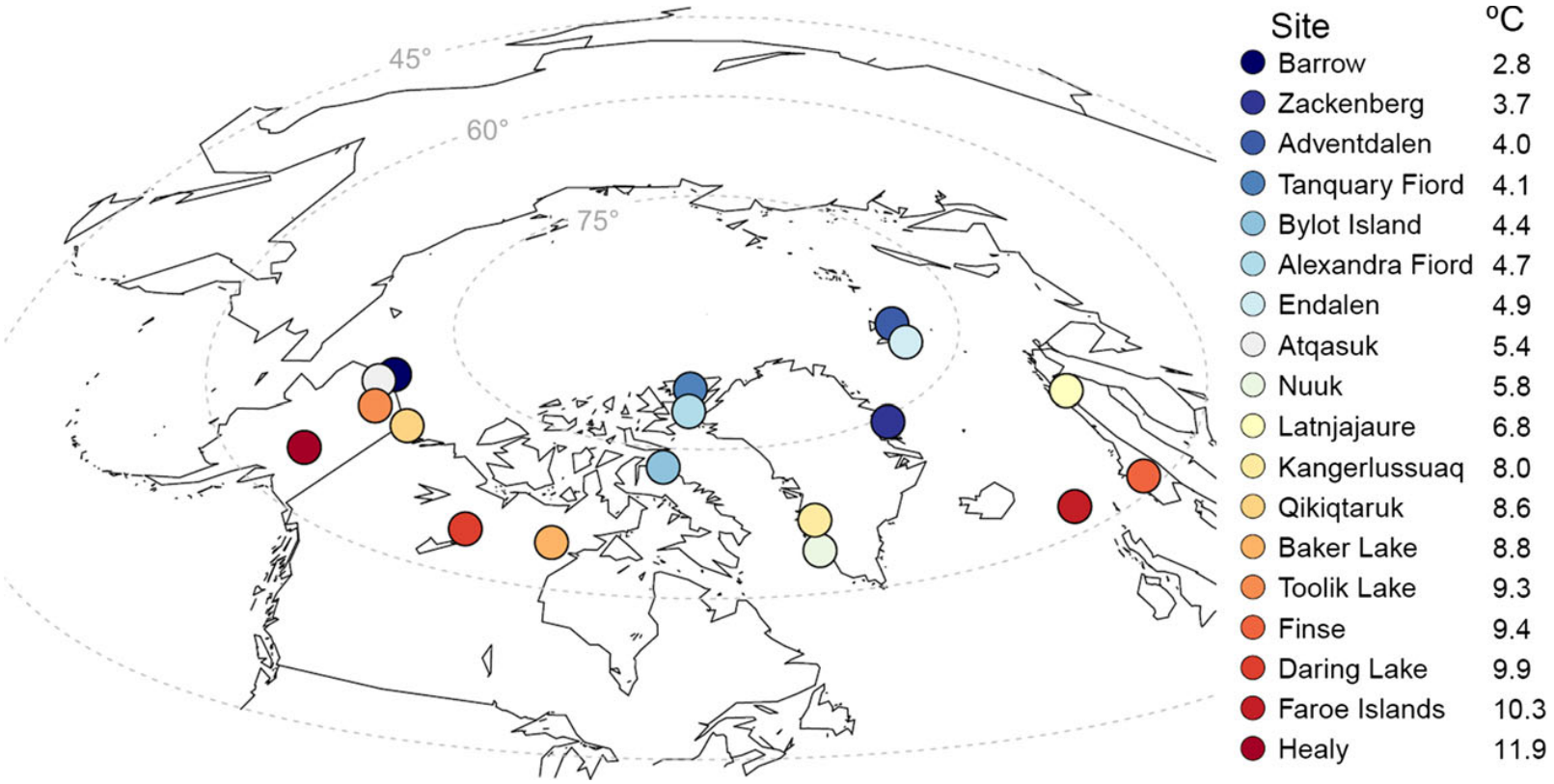






Good graphics

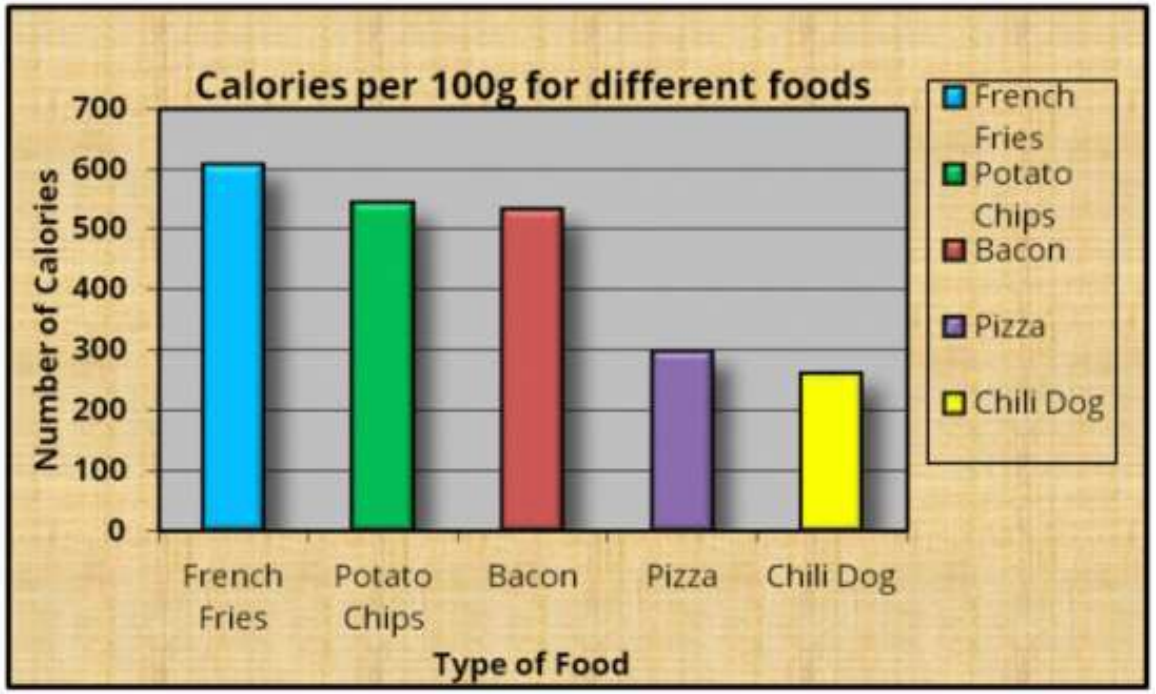
- Good graphs tell a memorable story
- Minimize junk
- Large fonts, effective colors, and minimal background



Prevéy et al. 2017. Global Change Biology 23: 2660–2671, doi: 10.1111/gcb.13619

Good graphics

- Good graphs tell a memorable story
- Minimize junk
- Large fonts, effective colors, and minimal background

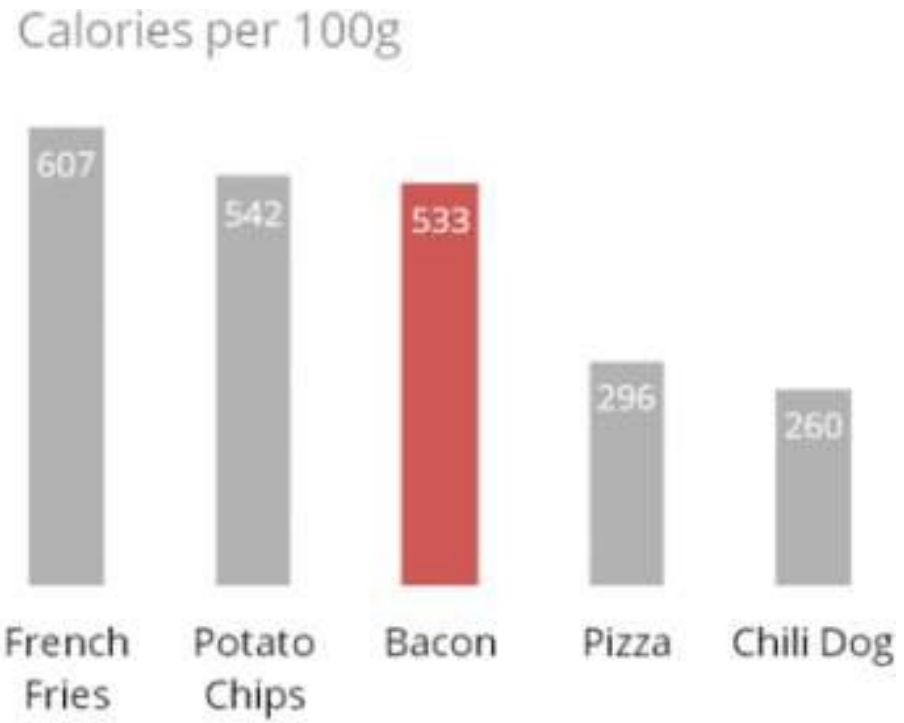


← Yuck!

Graphs from: https://blog.hubspot.com/marketing/data-graph-design-powerpoint-tips-ht

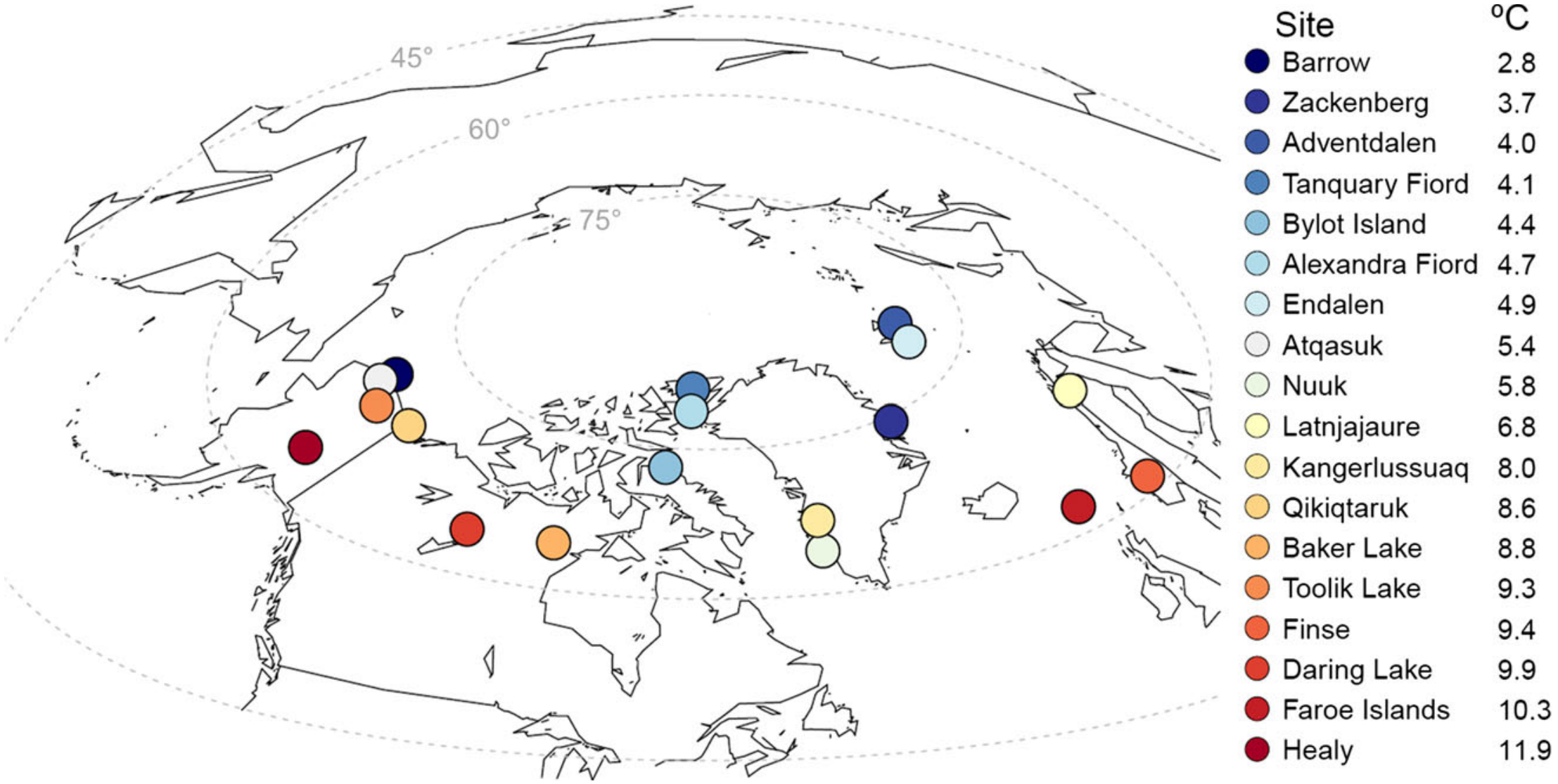
Good graphics

- Good graphs tell a memorable story
- Minimize junk
- Large fonts, effective colors, and minimal background

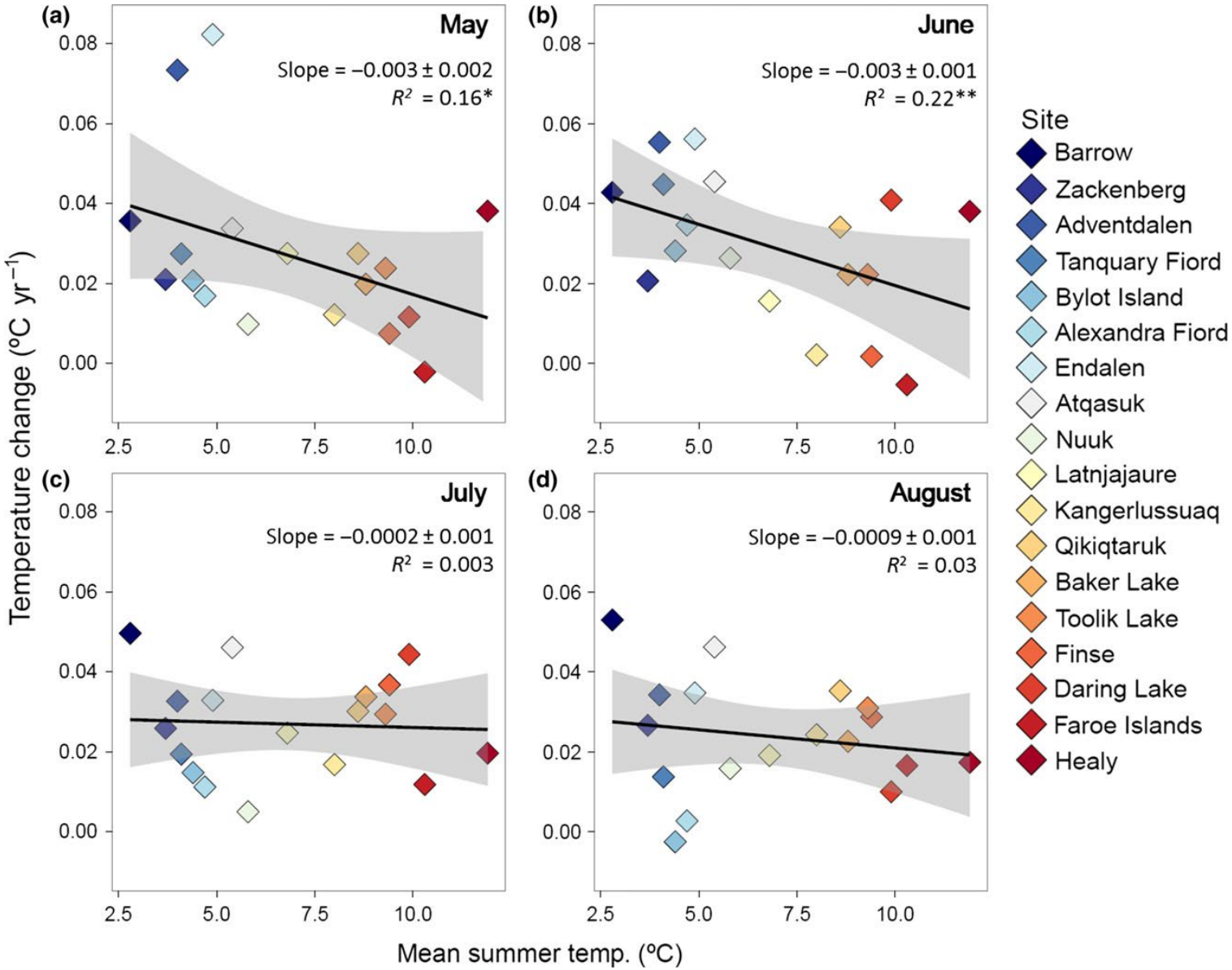


← Better!

Graphs from: https://blog.hubspot.com/marketing/data-graph-design-powerpoint-tips-ht



Use color to help tell your research story…..



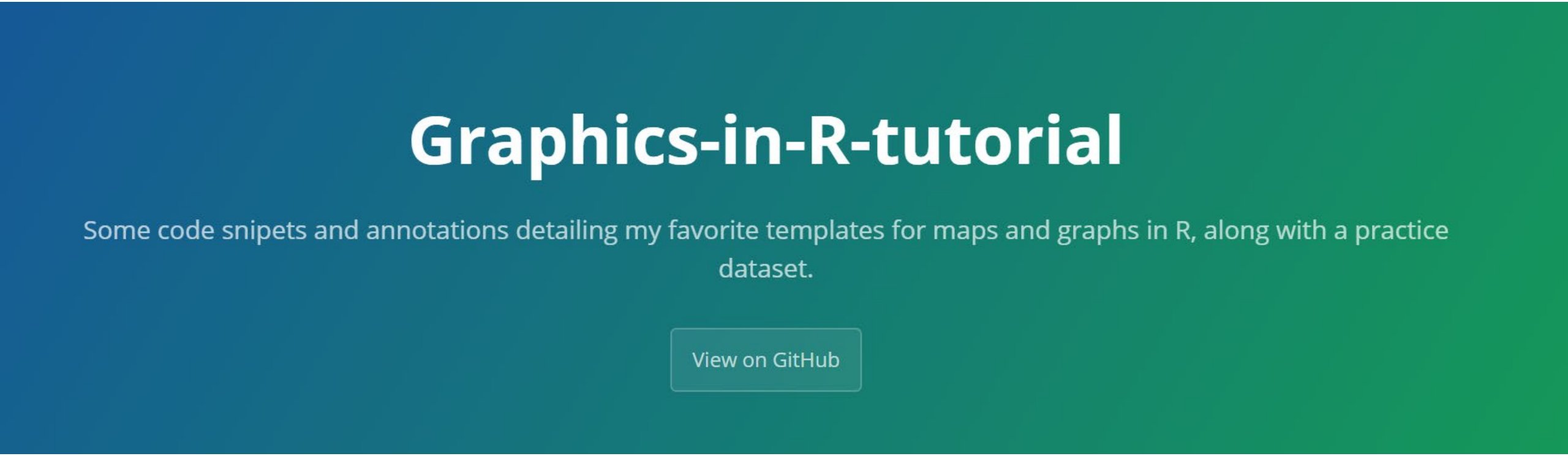
Use color consistently to create continuity between graphics…..

Outline

- Walk through a few examples of maps and graphs using R studio and sample data
  - Play along or sit back and watch!
- Best practices for saving and exporting images generated in R
- Rasters and next steps for R graphics aficionados…
- Questions / trouble-shooting / discussion

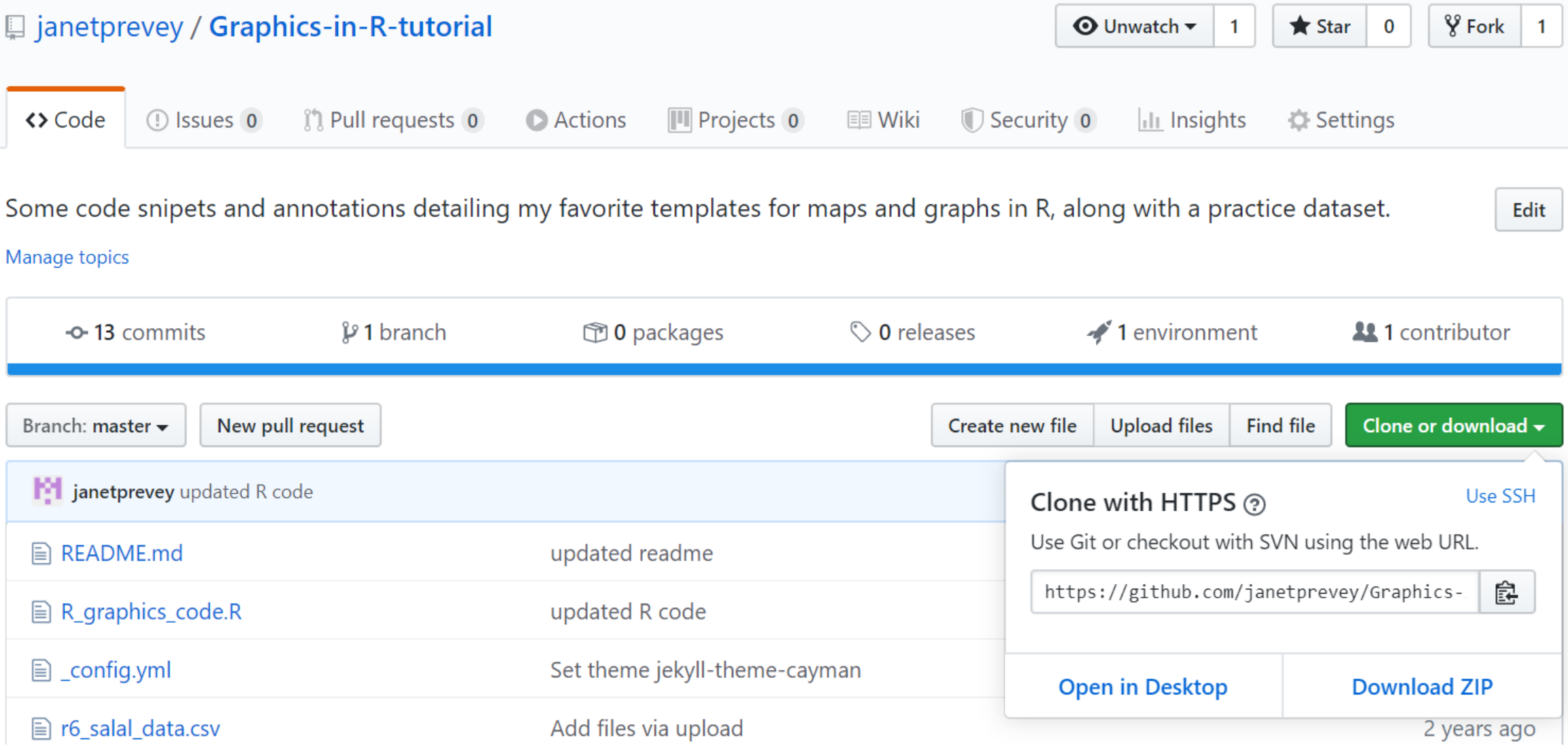
Accessing the code and data

https://janetprevey.github.io/Graphics-in-R-tutorial/



Accessing the code and data

https://janetprevey.github.io/Graphics-in-R-tutorial/



Accessing the code and data

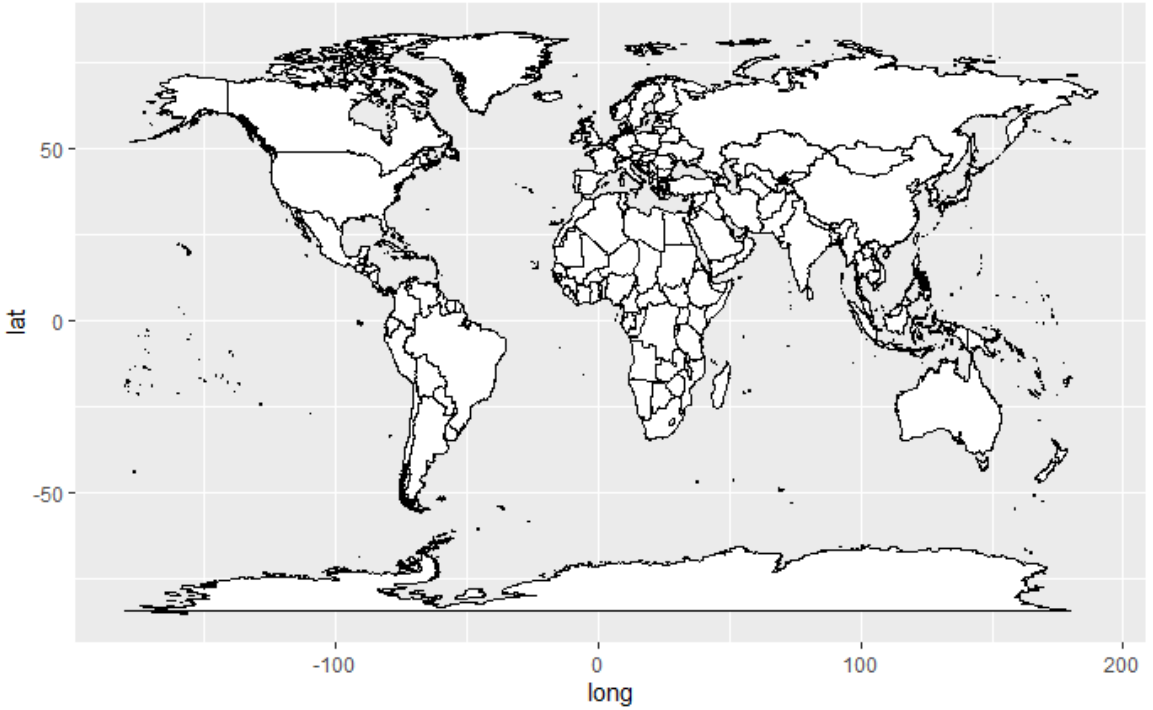
```
#if(!requireNamespace("devtools")) install.packages("devtools")
#devtools::install_github("dkahle/ggmap")

install.packages('maps')
install.packages('maptools')
install.packages('ggplot2') #one of the most popular graphics packages
install.packages('ggmap') # ggmap draws maps that work seamlessly w/ ggplot2
install.packages('RColorBrewer') #some nice color palette options, website: http://colorbrewer2.org/#type=sequer
install.packages('viridis') #other beautiful color options - website: https://cran.r-project.org/web/packages/vi
install.packages('wesanderson') #color palettes inspired by Wes Anderson movies: http://blog.revolutionanalytics
install.packages('colorRamps')#nice rainbow color palettes: https://cran.r-project.org/web/packages/colorRamps/c
install.packages('gridExtra') #helps to place graphs together on the same page

#### Load the packages we just installed

library(maps)
library(maptools)
library(ggplot2)
library(ggmap)
library(RColorBrewer)
library(viridis)
library(wesanderson)
library(colorRamps)
library(gridExtra)
```

green notes with hashtags are not read by R – they are just annotations describing the code

Error codes? May need to install viridisLite, or ggmap from the github development page……

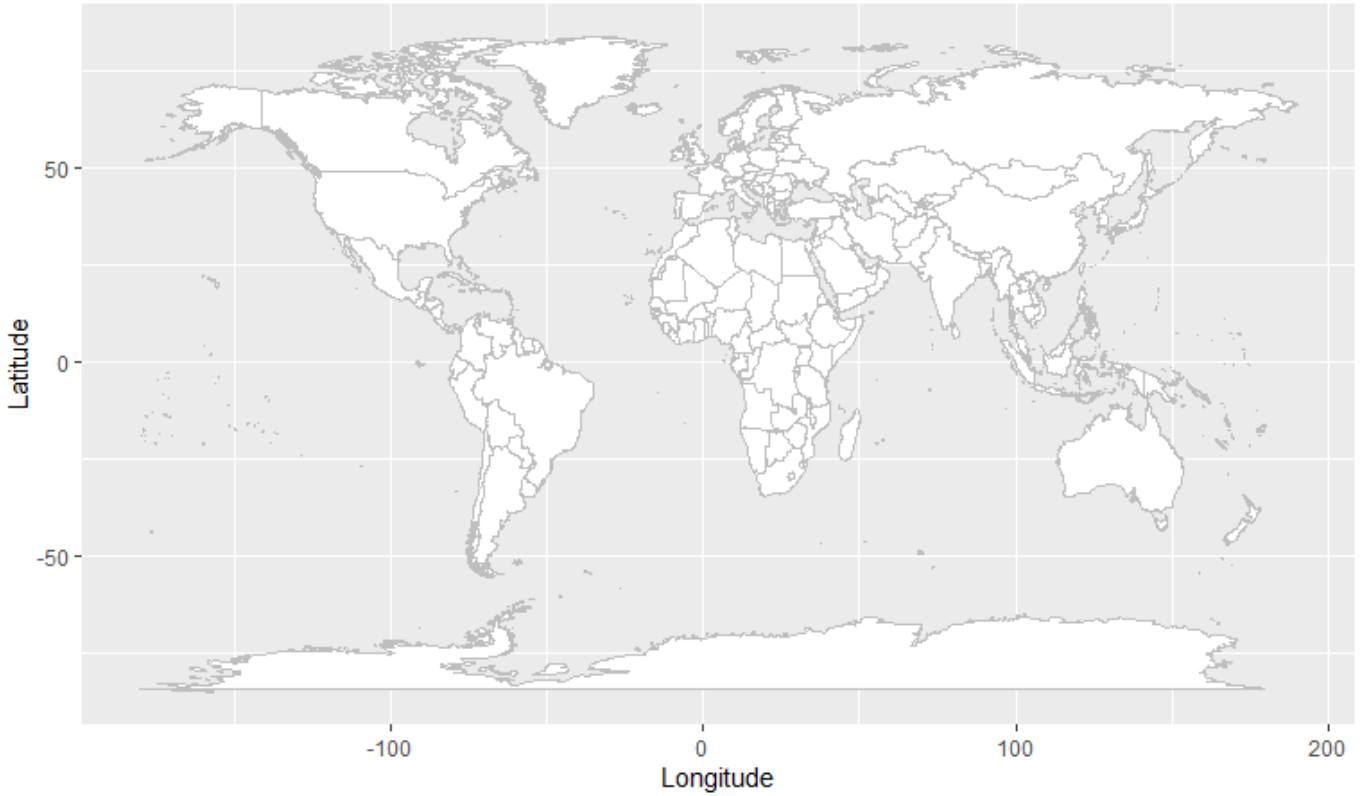
Make a world map

```
# First, lets make a simple world map using ggplot
# This code grabs a layer of country borders for the world

mapWorld <- borders("world", colour="black", fill="white", size = 0.2)

# Now we add our world map outline to a ggplot object

mp <- ggplot() +  mapWorld

# And typing the name of our object prints our map out in 'Plots'

mp
```



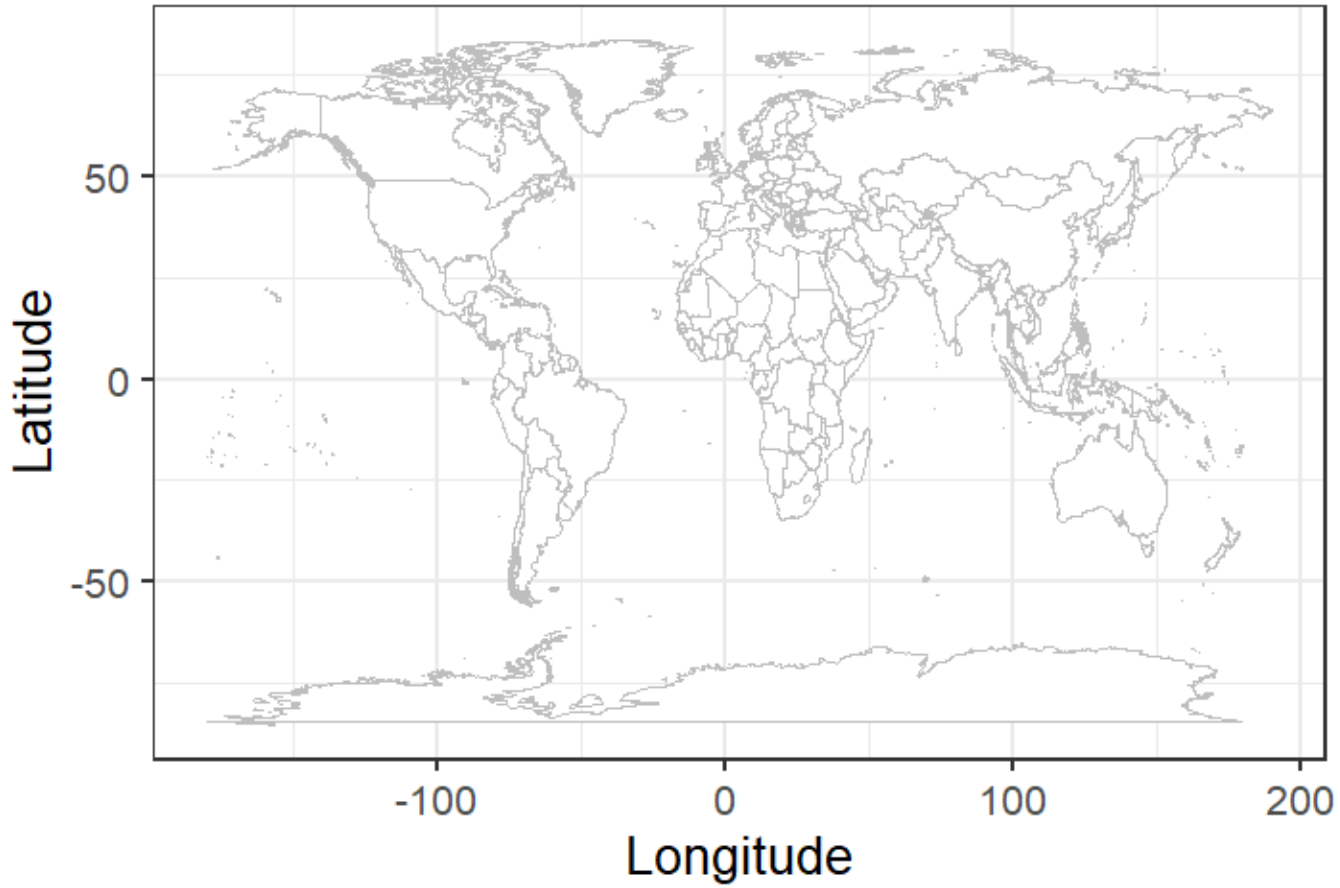
Make a world map

```
# We can custimize the world map in a variety of ways - 'colour', 'fill', and 'size'
# let us specify different style options, we customize x and y axis labels the same
# as any ggplot graph

mapWorld <- borders("world", colour="gray", fill="white", size = 0.1)
mp <- ggplot() +  mapWorld + xlab("Longitude") + ylab("Latitude")
mp
```



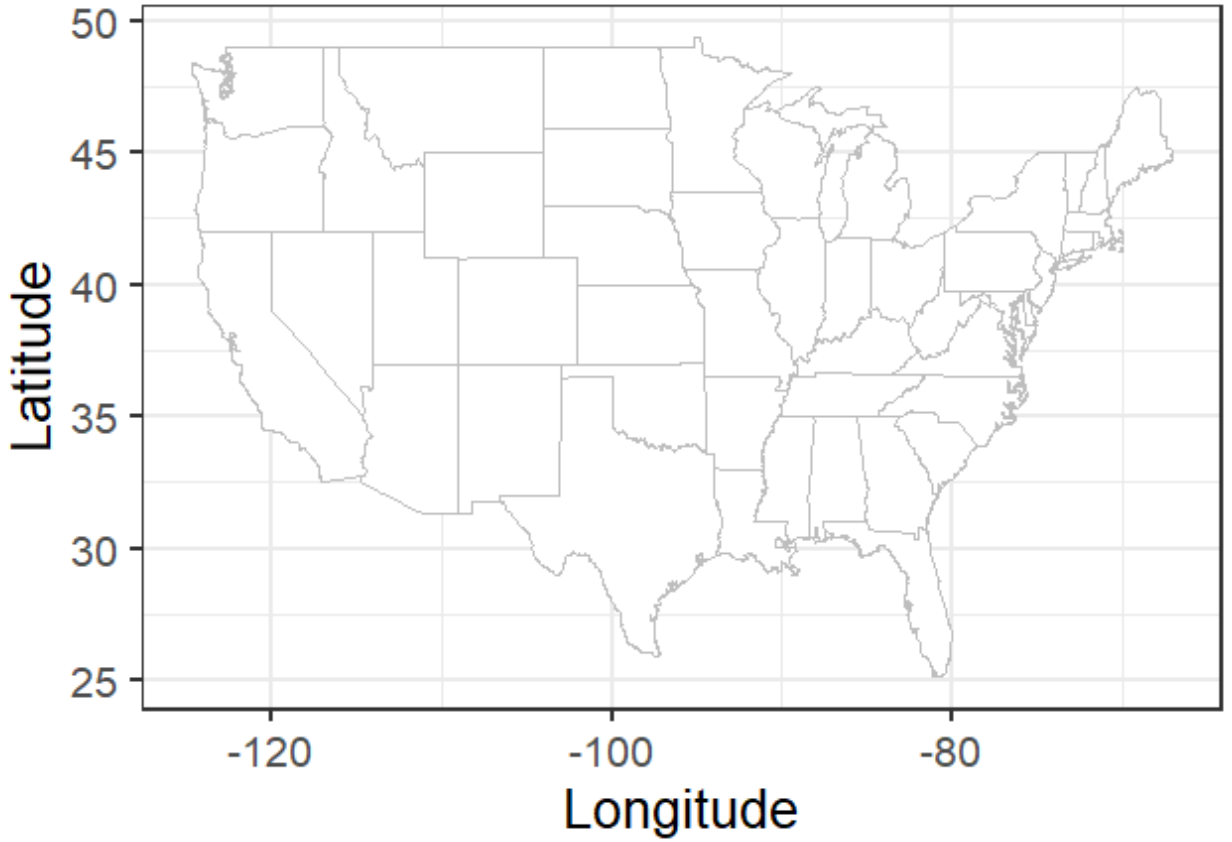
Make a world map

```
# we can use ggplot themes to change global graph settings for all ggplot graphics
# lets set a theme for all the graphs and maps in ggplot and ggmaps below to make su
# we have white backgrounds and larger than normal axes labels
# 'theme_bw' converts the normal grey background to white, and 'base_size=...'
# indicates the size of labels for all graphs

theme_set(theme_bw(base_size=20))

# check out how our map looks with the new theme:

mp
```



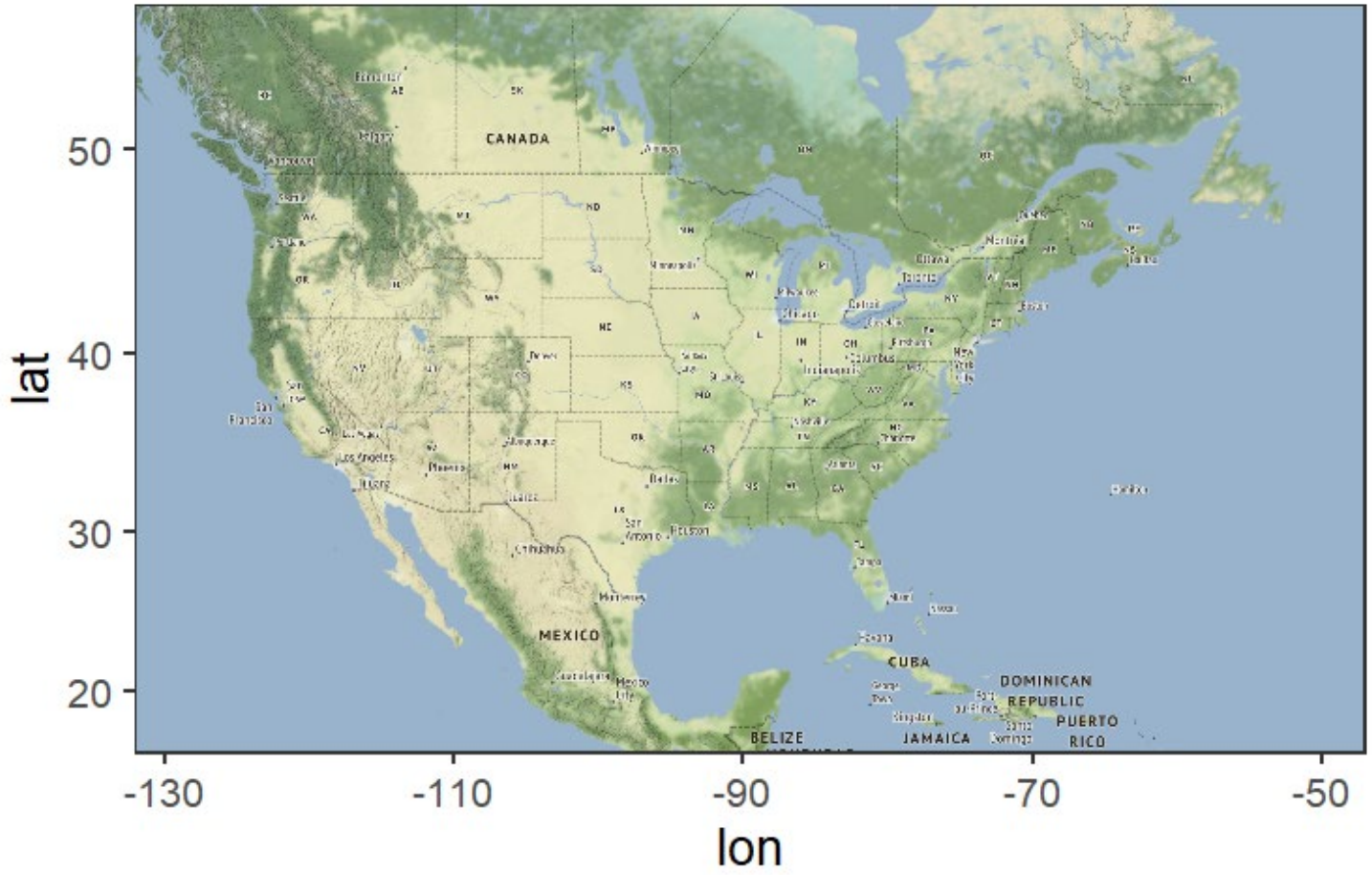
Make a world map

```
# We can also zoom in to the continental USA if we want...

mapUSA <- borders("state", colour="gray", fill="white", size = 0.1)
mp <- ggplot() +  mapUSA + xlab("Longitude") + ylab("Latitude")
mp
```



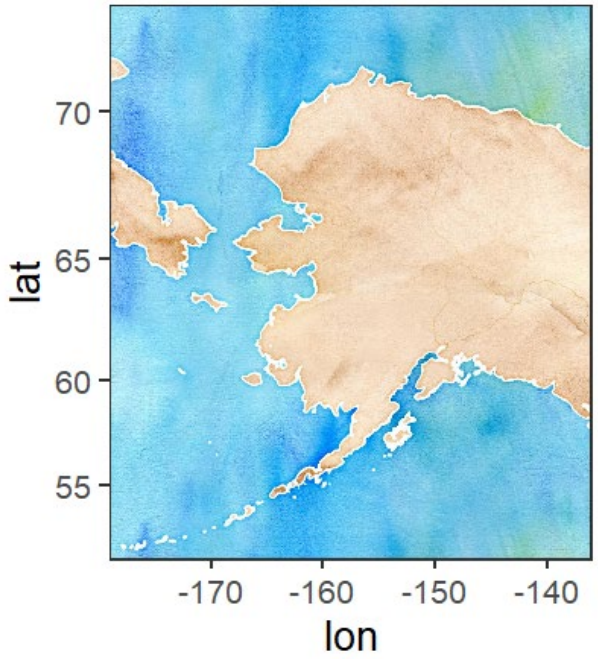
Making maps with ggmap

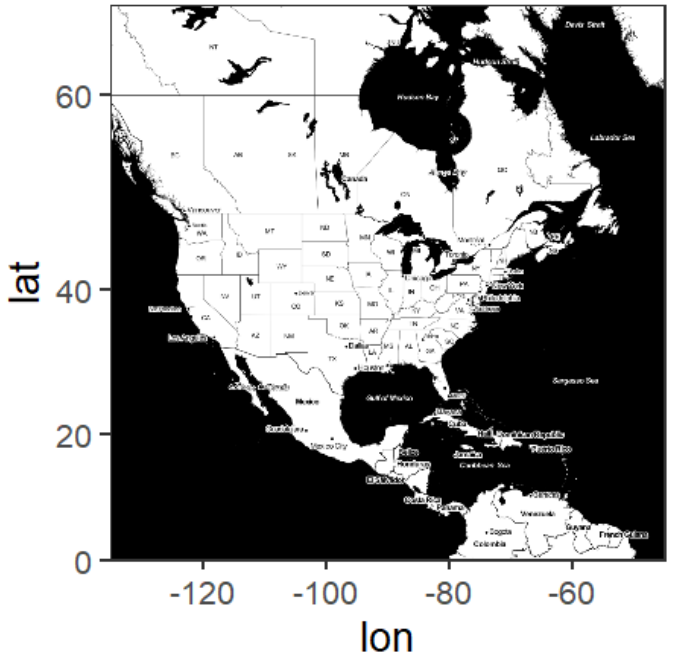
```
# Specify the bounding box of your map with:
#'location <- c(left longitude, bottom latitude, right longitude, upper latitude)

myMap <- get_stamenmap(location<- c(-132,16,-47,56), zoom = 5, maptype="terrain",
          crop=TRUE) ## smaller zoom values require fewer tiles but show less detail

# view the map you just made
ggmap(myMap)
```



Making maps with ggmap

```
### Watercolor map of Alaska

myMap <- get_stamenmap(location<- c(-179,51,-136,73), zoom = 5, maptype="watercolor",crop=TRUE)
ggmap(myMap)
```

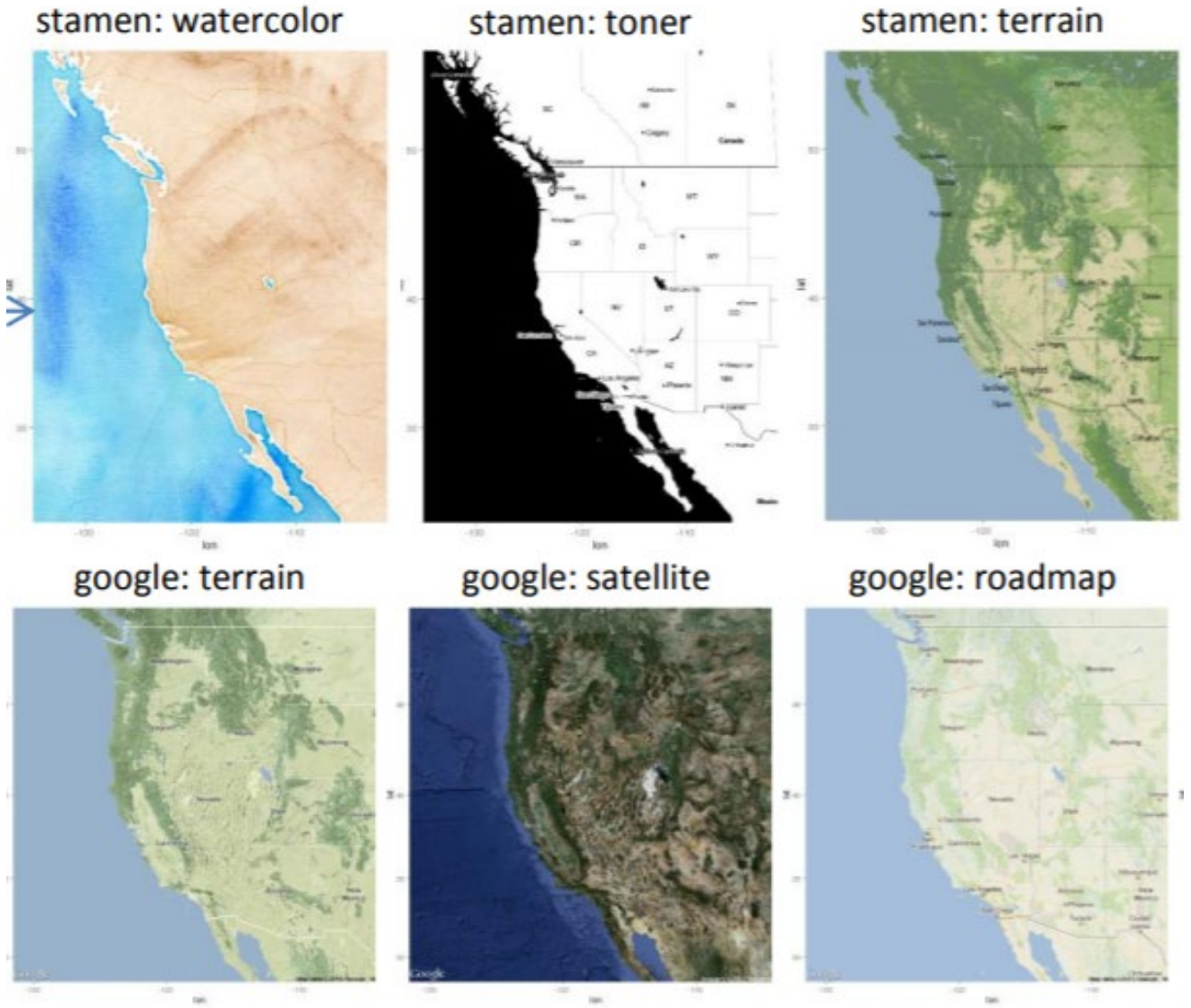





```
### Black and white map

myMap <- get_stamenmap(location<- c(-132,16,-47,56), zoom = 4, maptype="toner", crop=FALSE)
ggmap(myMap)
```

Making maps with ggmap

- https://www.nceas.ucsb.edu/sites/default/files/2020-04/ggmapCheatsheet.pdf
- https://cran.r-_project.org/web/packages/ggmap/ggmap.pdf
- https://www.rdocumentation.org/packages/ggmap/versions/3.0.0



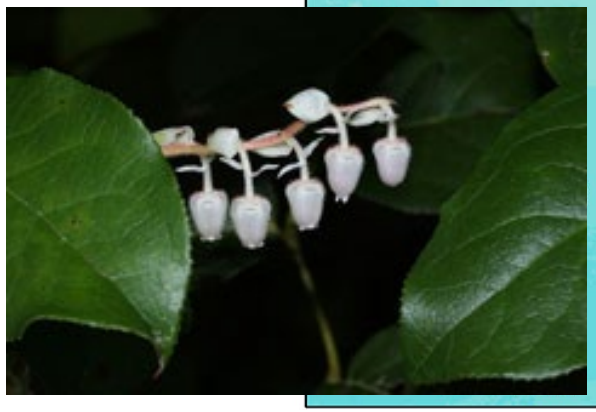
Note the new rules about accessing googlemaps through ggmap…

Background on the data file

- Measurements of % cover and phenological stage of salal (*Gaultheria shallon*) in Olympic and Mt. Baker Snoqualmie National forests from long-term monitoring plots: 1980 - 2003

- Monthly temperature data from DAYMET https://daymet.ornl.gov/

- **NOTE**: data are ***preliminary*** and for practice only – if you are interested in these types of data, contact me later.



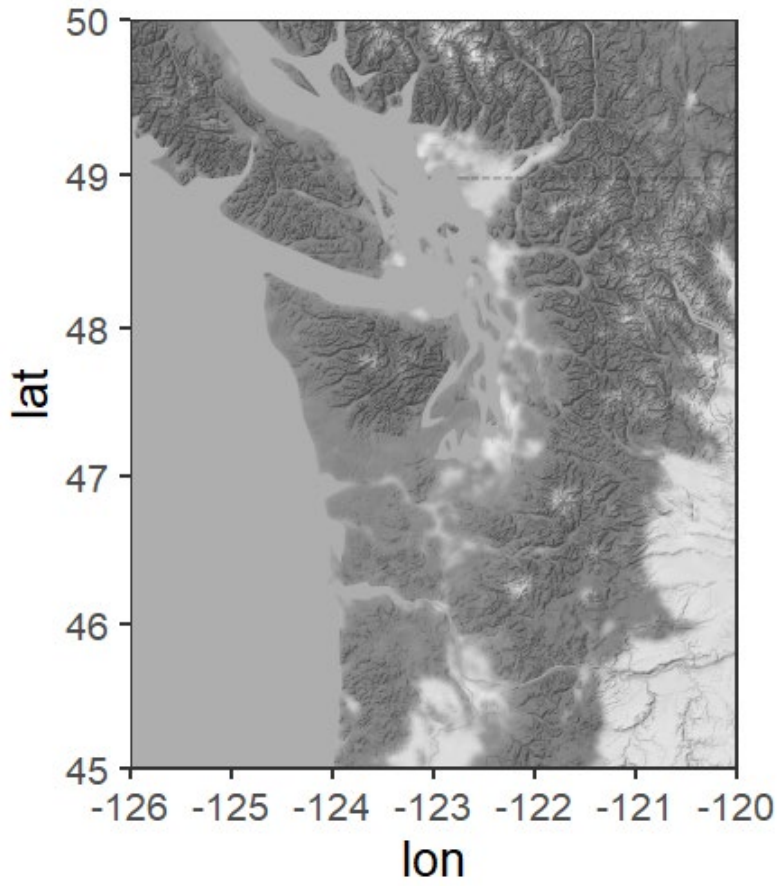
Import the data file in R studio

```
setwd("C:/Users/jprevey/Desktop/Graphics-in-R-tutorial-master")

dat <- read.csv("r6_salal_data.csv", header = TRUE, sep = ",")

# Now we have a new data frame called 'dat' loaded into our workspace with the raw data

#Lets look at a summary of our variables

summary(dat)
```

```
     Plot_ID                   species     Phenophase        year          Month      
0609-1834:  5   Gaultheria shallon:351   Flower:141   Min.   :1980   Min.   :6.000  
0609-1835:  5                            Fruit :210   1st Qu.:1982   1st Qu.:7.000  
0609-1836:  5                                         Median :1985   Median :8.000  
0609-1579:  4                                         Mean   :1987   Mean   :7.553  
0609-803 :  4                                         3rd Qu.:1992   3rd Qu.:8.000  
0609-820 :  4                                         Max.   :2003   Max.   :9.000  
(Other)  :324                                                                       
      DOY             cover             lat             long              jan         
 Min.   :152.0   Min.   : 0.50   Min.   :46.36   Min.   :-124.3   Min.   :-8.5887  
 1st Qu.:195.5   1st Qu.: 5.00   1st Qu.:47.38   1st Qu.:-123.5   1st Qu.:-2.1250  
 Median :214.0   Median :25.00   Median :47.60   Median :-123.2   Median : 1.4435  
 Mean   :214.4   Mean   :36.15   Mean   :47.69   Mean   :-122.8   Mean   : 0.8808  
 3rd Qu.:239.0   3rd Qu.:70.00   3rd Qu.:47.98   3rd Qu.:-121.5   3rd Qu.: 3.5323  
```

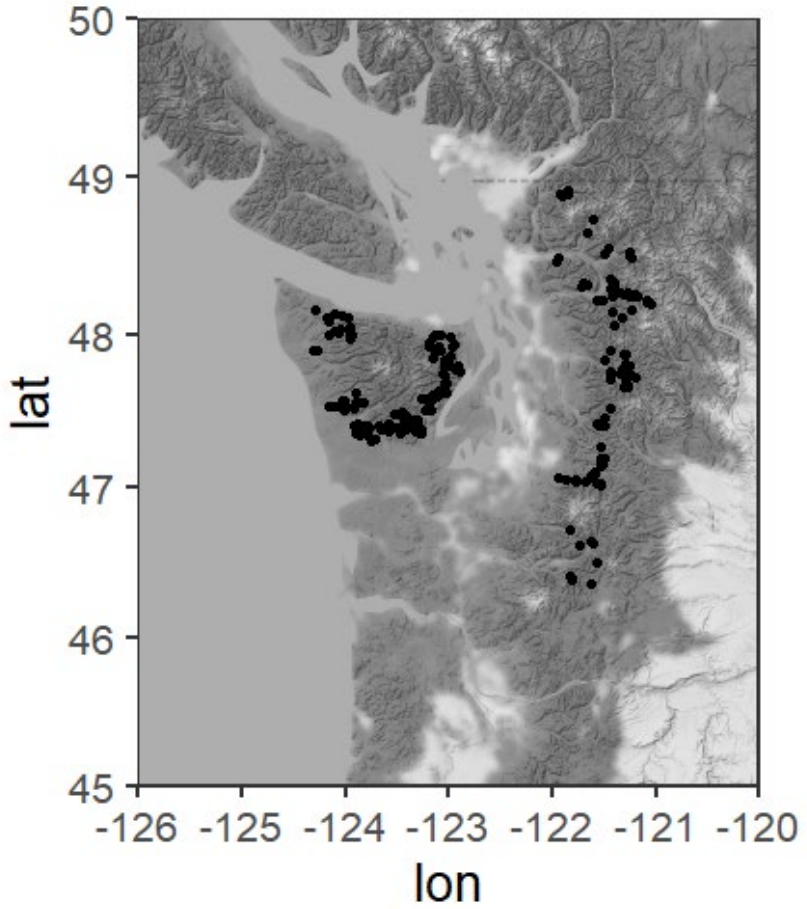
Making a map of salal plot locations

```
## Lets move forward with a black and white terrain map, and lets zoom in on western Washington:
## Play around with the zoom levels to get your desired level of detail

myMap <- get_stamenmap(location<- c(-126,45,-120,50), zoom = 7,
          maptype="terrain-background",color = "bw", crop=TRUE)

ggmap(myMap)
```



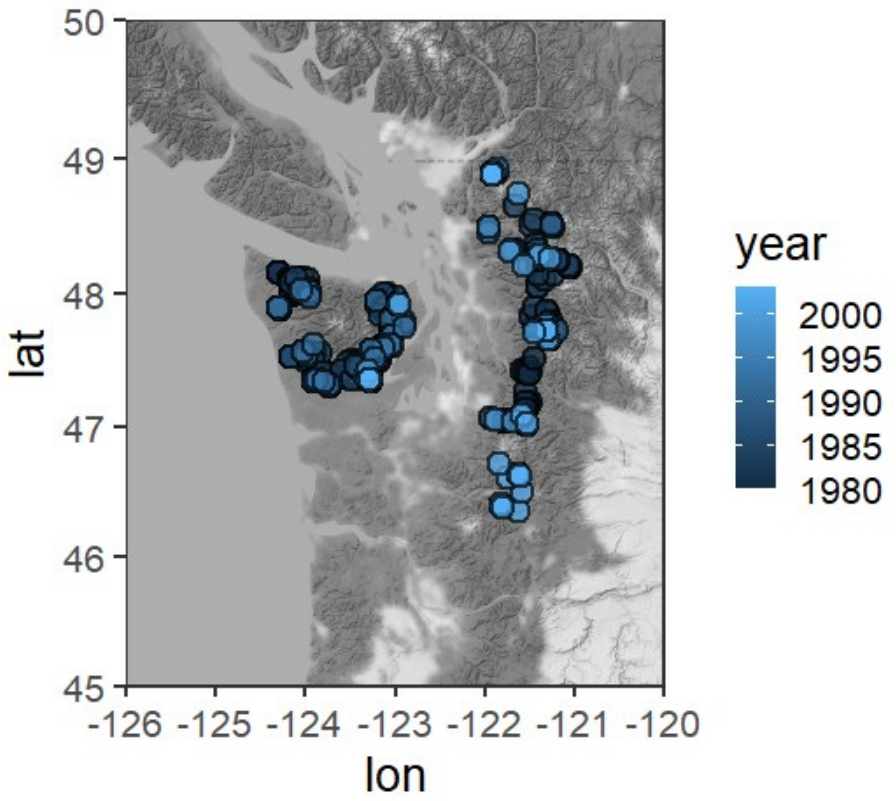
Making a map of salal plot locations

```
# and now we add the plot locations from the .csv of data to our map with the geom_point
ptmap <- ggmap(myMap) + geom_point(data=dat, aes(x=long, y=lat))
# view our new map with points
ptmap
```



Making a map of salal plot locations

```
# So the points are small and hard to see - lets fix that.
# We can increase the size of the points ('size = ..'), and color the plots by the year
# they were sampled to provide more information (using "fill = year" in the asthetics
# mapping. We can also make the points slighty transparent to show overlapping locations

ptmap <- ggmap(myMap) + geom_point(data=dat, aes(x=long, y=lat, fill = year),
          pch = 21, color = "black", stroke = 1.1, size = 4.2, alpha = 0.7)
ptmap
```



Meeehhhhhhh...

A note on colors….

“Color is the place where our brain and the universe meet.” – Paul Klee

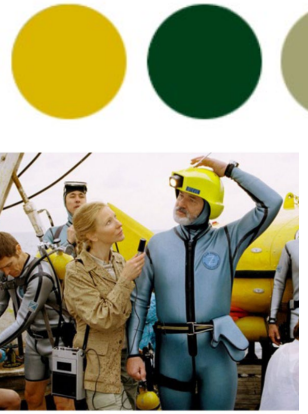
Image from: http://blog.visme.co/color-combinations/

There are some AMAZING R packages and color palettes out there…..

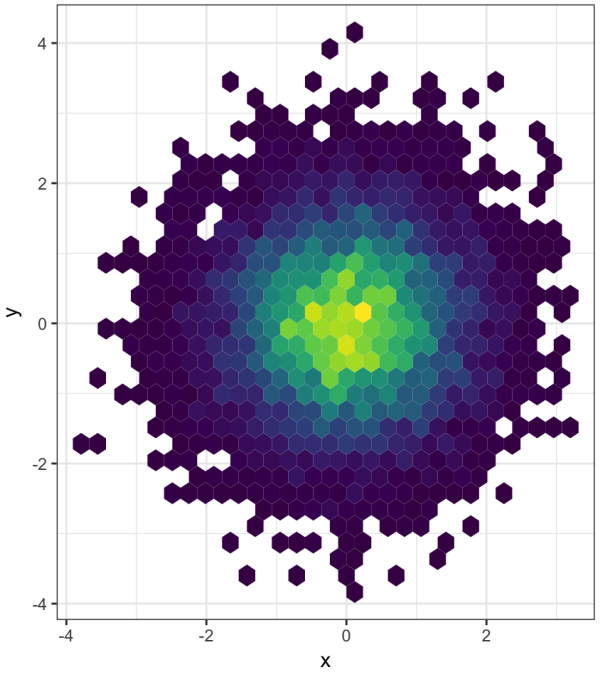
Color palettes for pros:

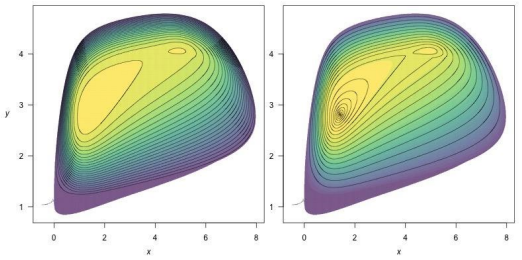
Give your R charts that Wes Anderson style

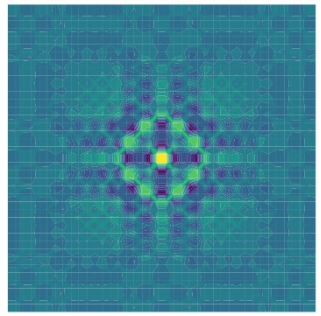
I'm a big fan of Wes Anderson's movies. I love the quirky characters and stories, the distinctive cinematography, and the unique visual style. Now you can bring some of that style to your own R charts, by making use of these Wes Anderson inspired palettes. Just choose your favourite Wes Anderson film or short:

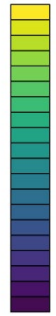



```
library(wesanderson)
```

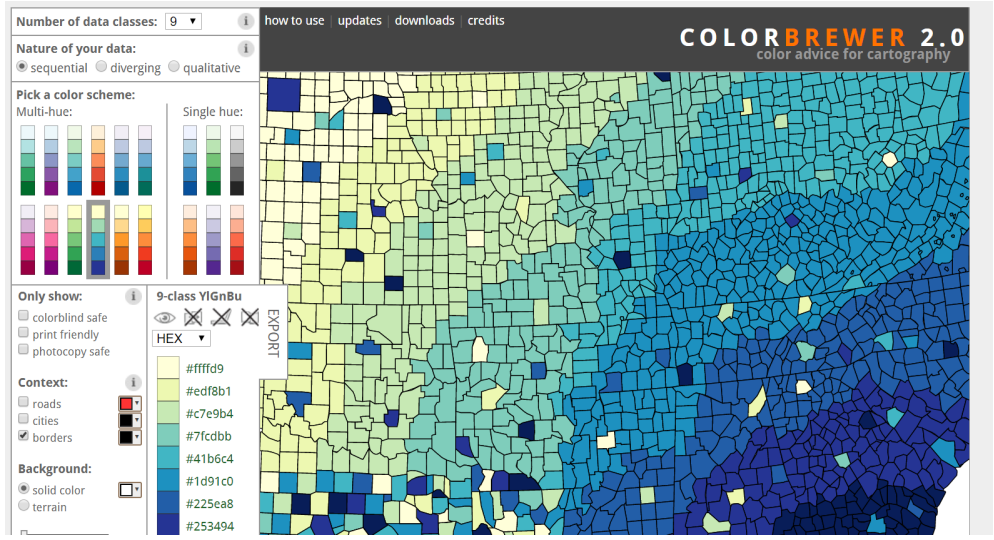








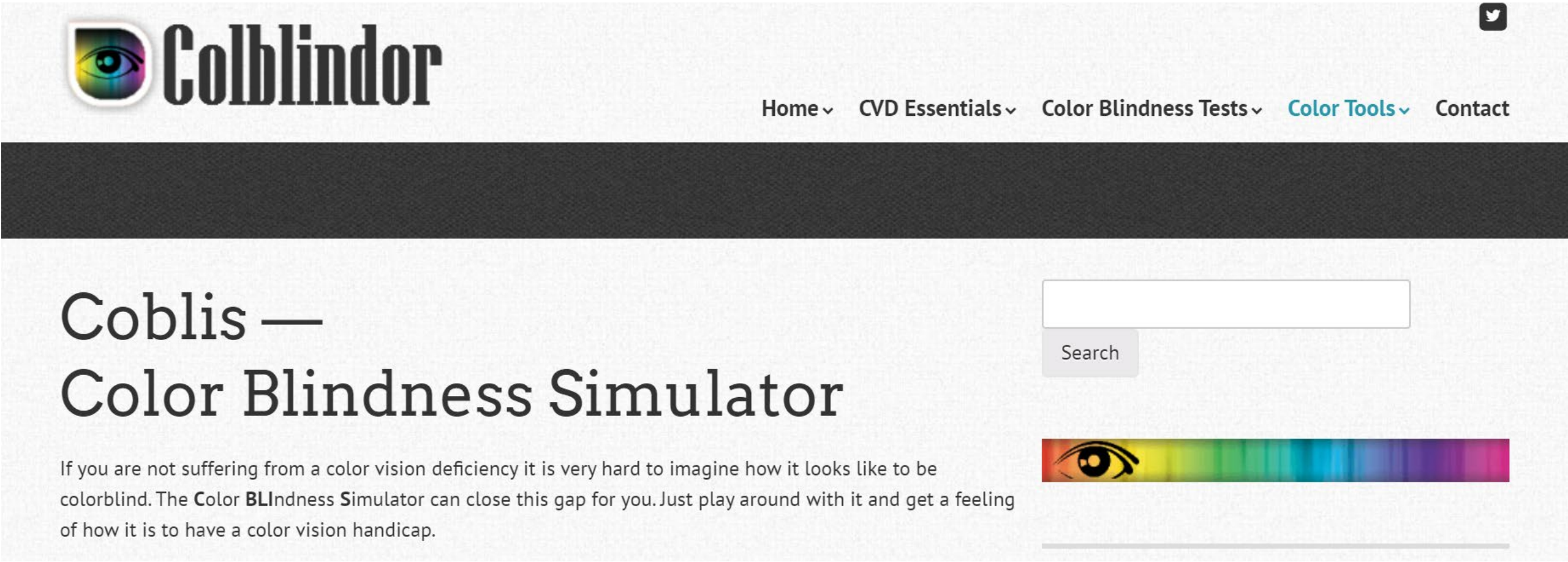
```
library(viridis)
```




```
library(RColorBrewer)
```

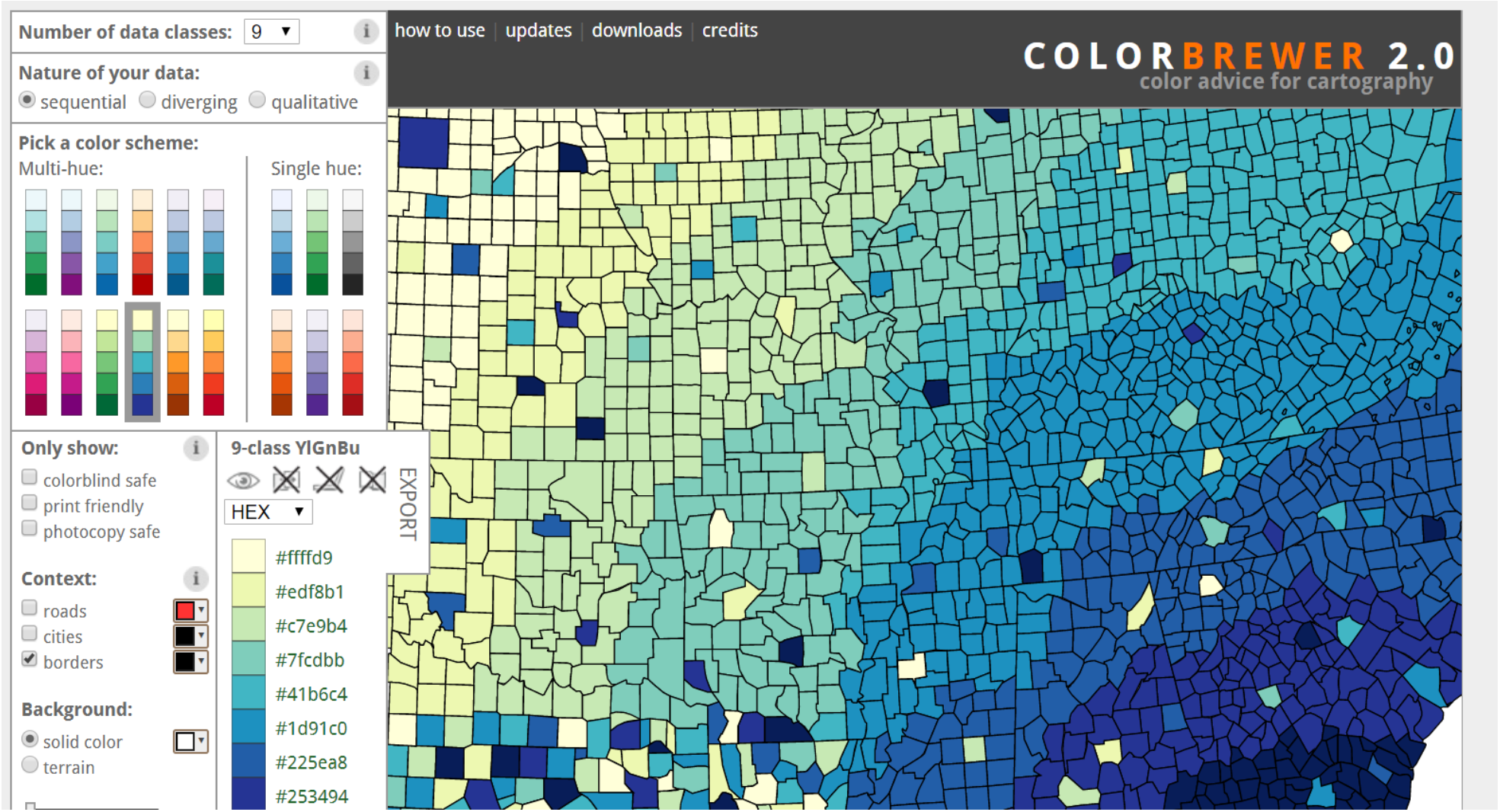
Choosing color-blind friendly palettes…

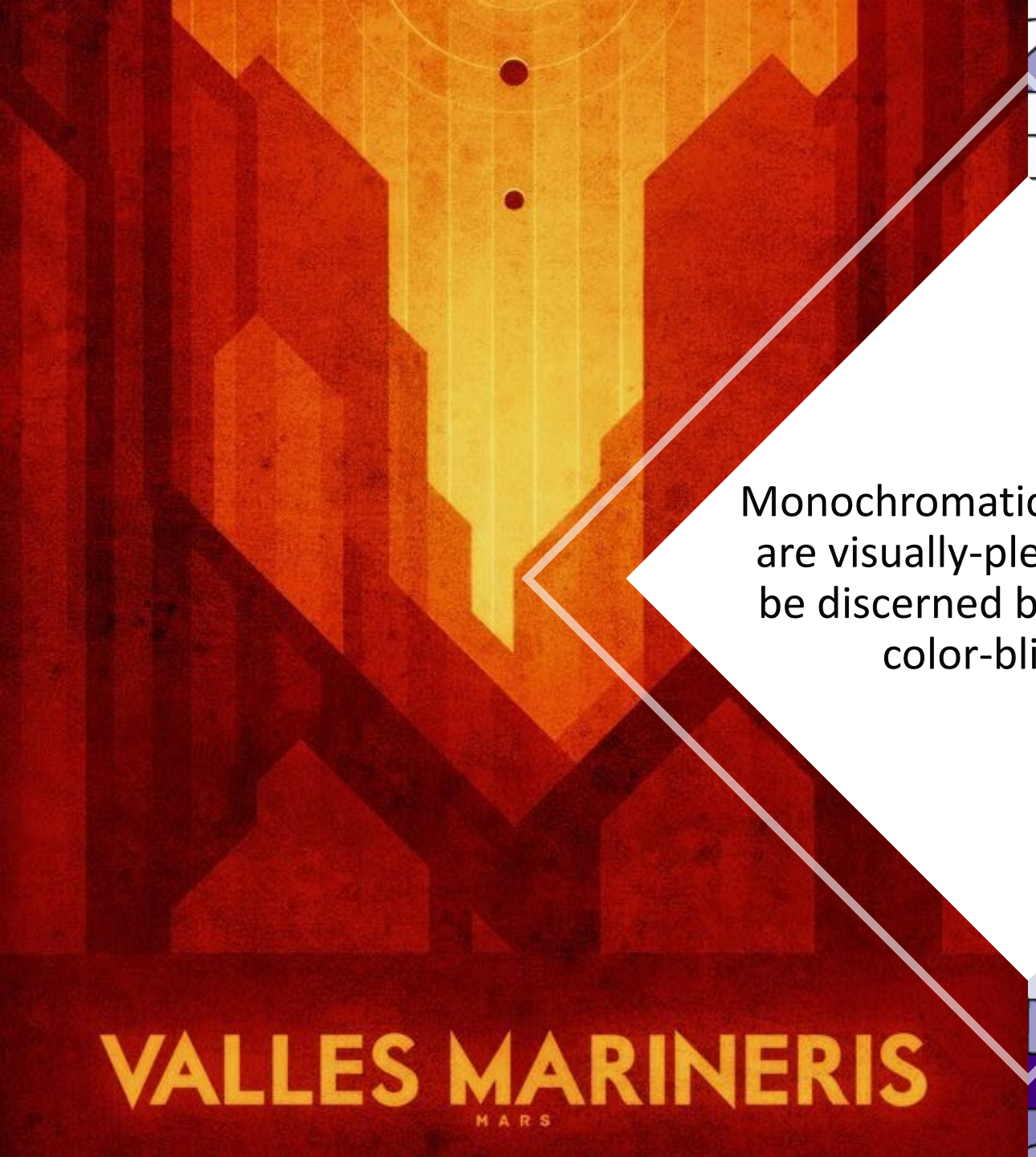
https://www.color-blindness.com/coblis-color-blindness-simulator/



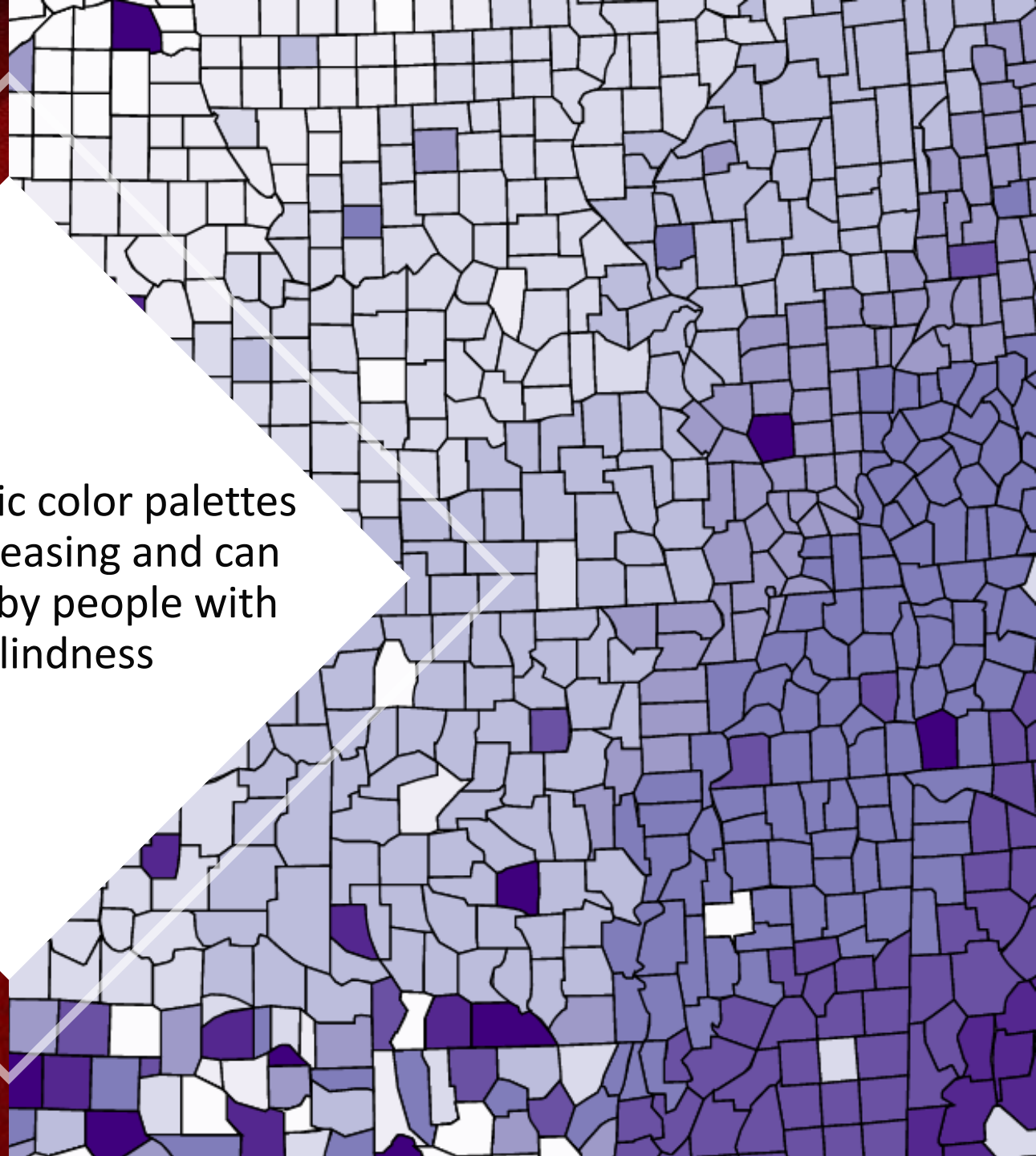
Choosing color-blind friendly palettes…

https://colorbrewer2.org/



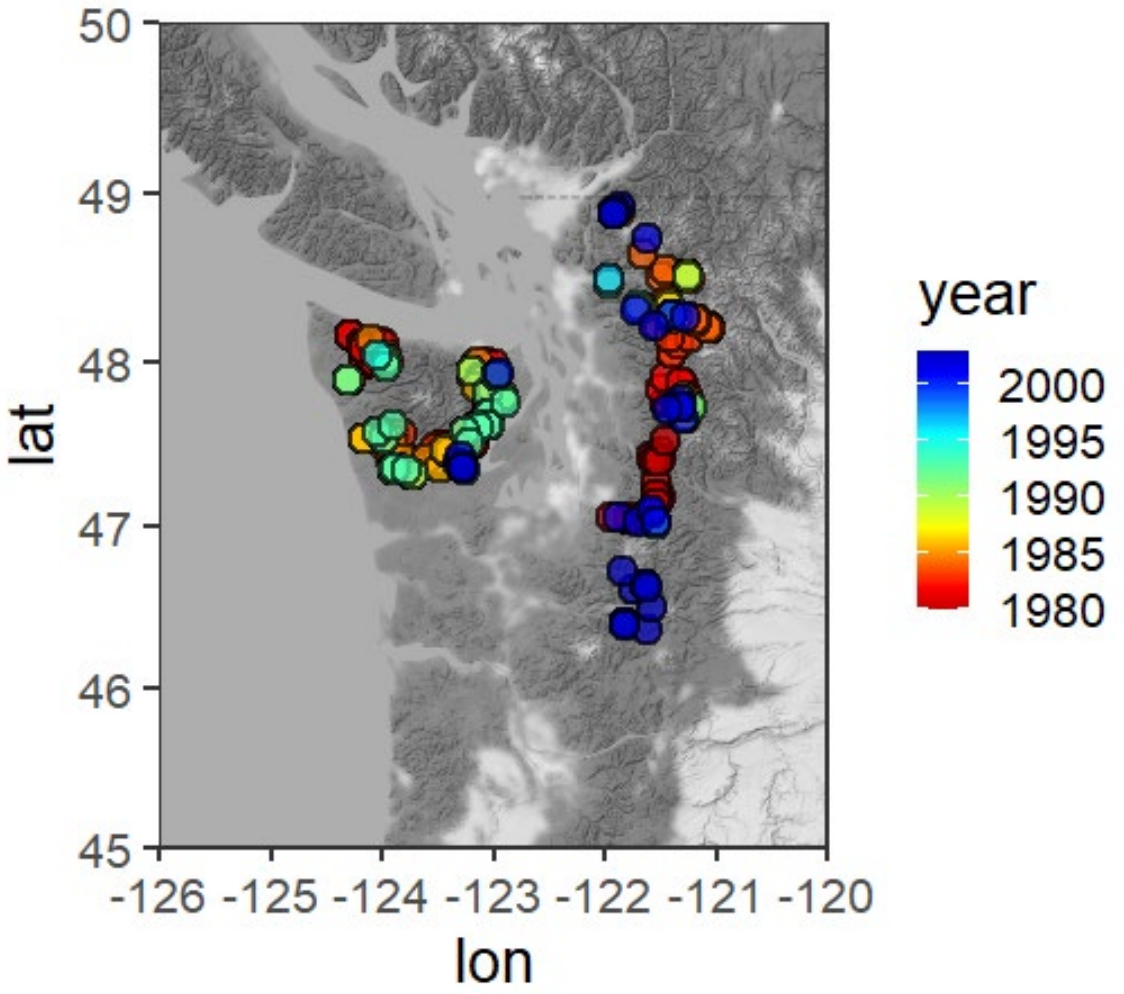


Monochromatic color palettes are visually-pleasing and can be discerned by people with color-blindness



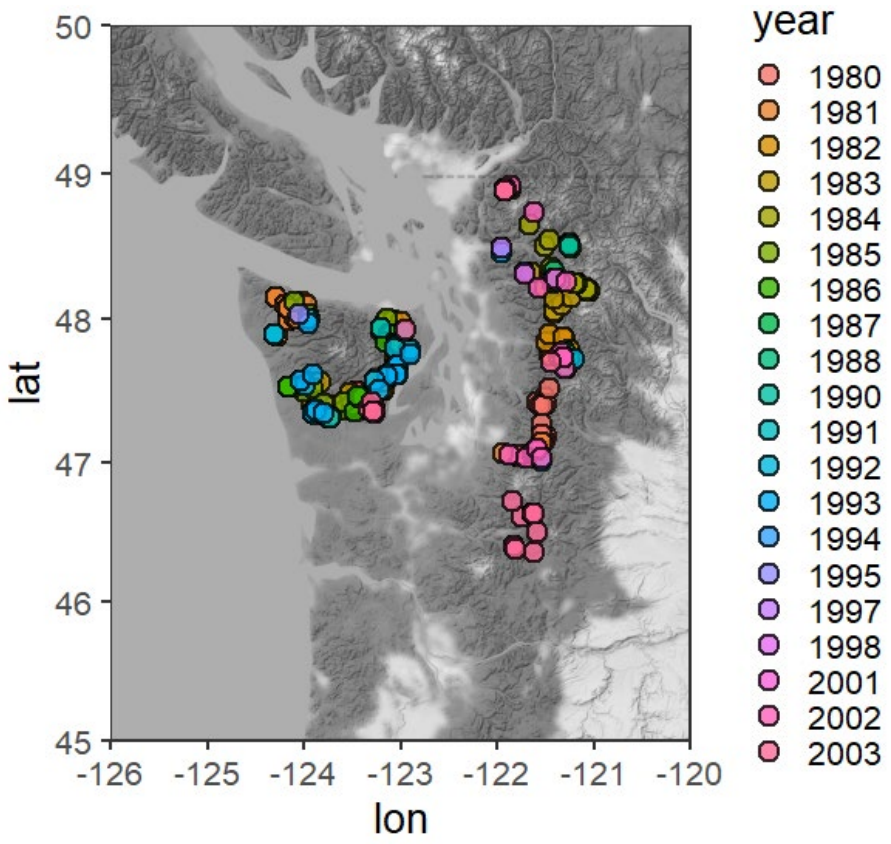
Making a map of salal plot locations

```
# Using colors from the viridis package

newmap <- ptmap + scale_fill_viridis()
newmap


# Using colors from the colorRamps Package:

newmap <- ptmap  +  scale_fill_gradientn(colours= (blue2green2red(23)))
newmap

# Use 'rev' to reverse the scale

newmap <- ptmap  +  scale_fill_gradientn(colours= rev(blue2green2red(23)))
newmap
```



Making a map of salal plot locations

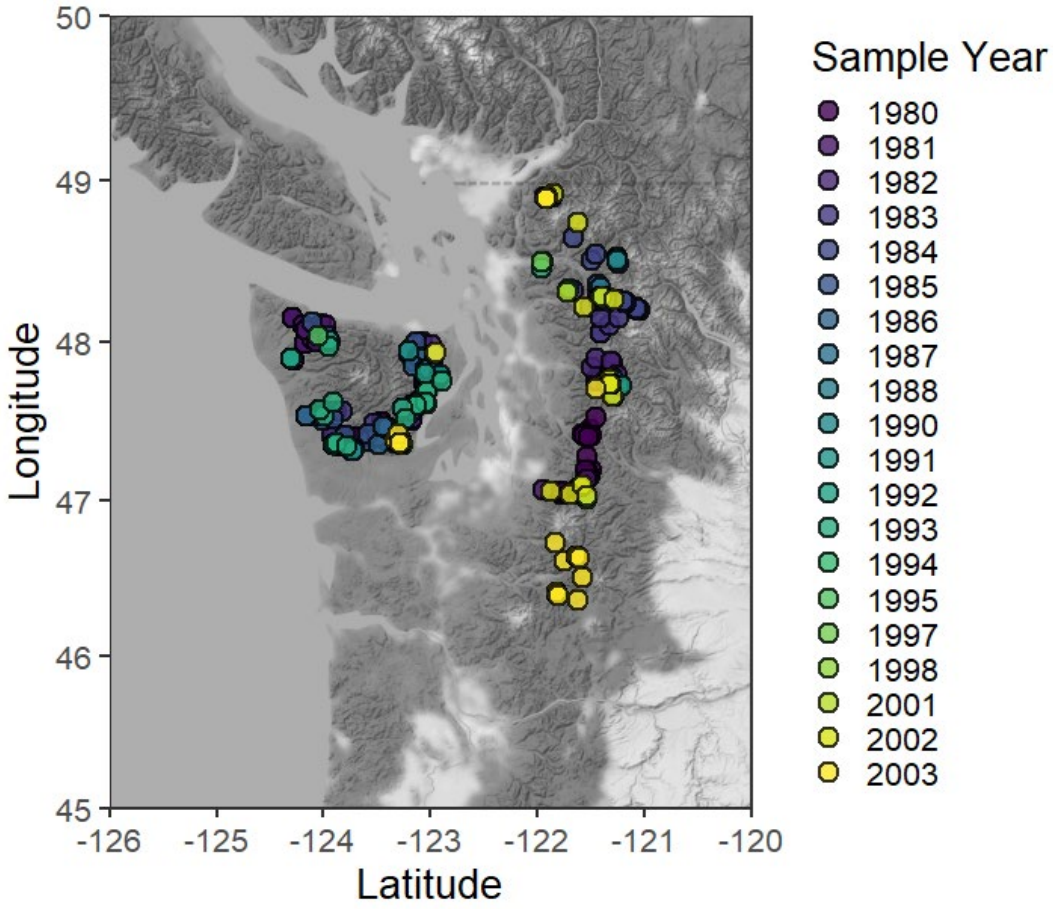
```
# We can also change year to a discrete variable and see if that helps:
# Designate year as a factor instead of number

dat$year <-factor(dat$year)

# Try again with year as a discrete variable

ptmap <- ggmap(myMap) + geom_point(data=dat, aes(x=long, y=lat, fill = year),
                                   pch = 21, color = "black", stroke = 1.1,
                                   size = 4.2, alpha = 0.8)
ptmap
```



Meeehhhhhhh……

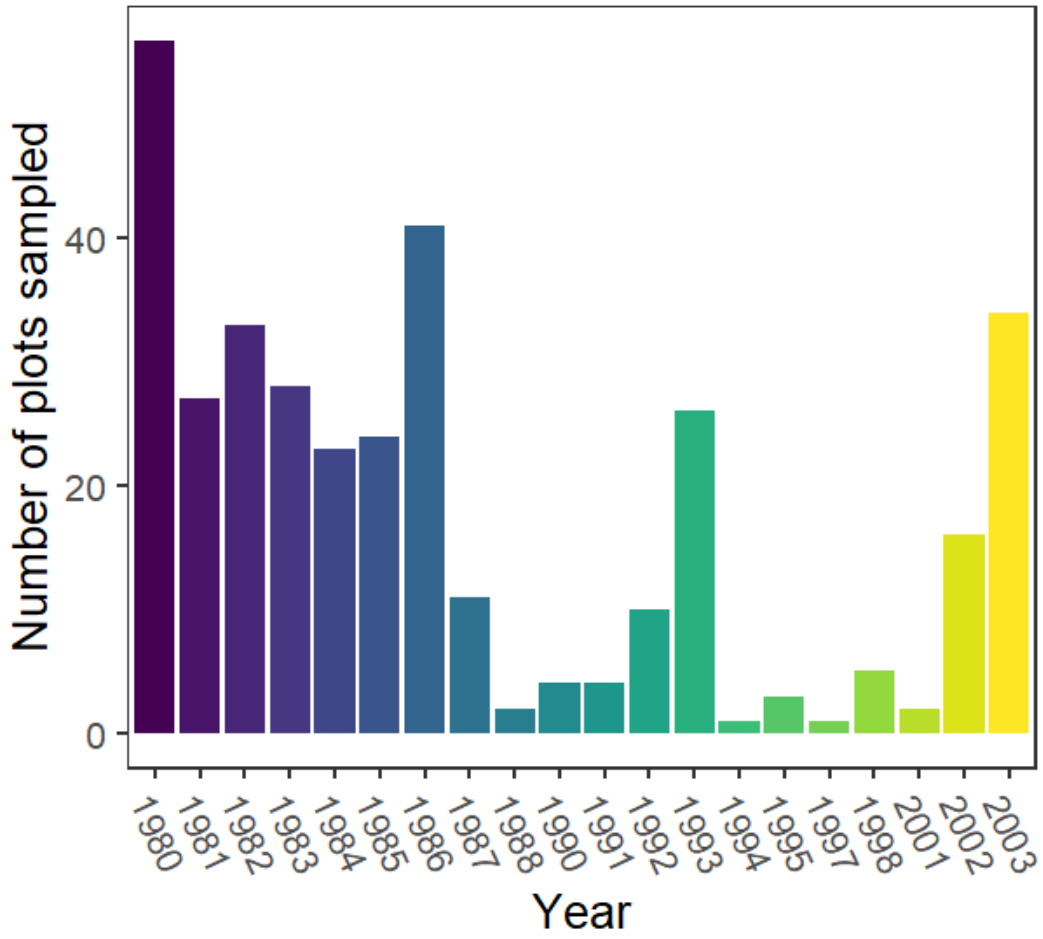
Making a map of salal plot locations

```
newmap <- ptmap + scale_fill_viridis(discrete=TRUE)

###Now Lets add better labels:

scdmap <- newmap + labs(x="Latitude",y= "Longitude", fill = "Sample Year")
scdmap
```



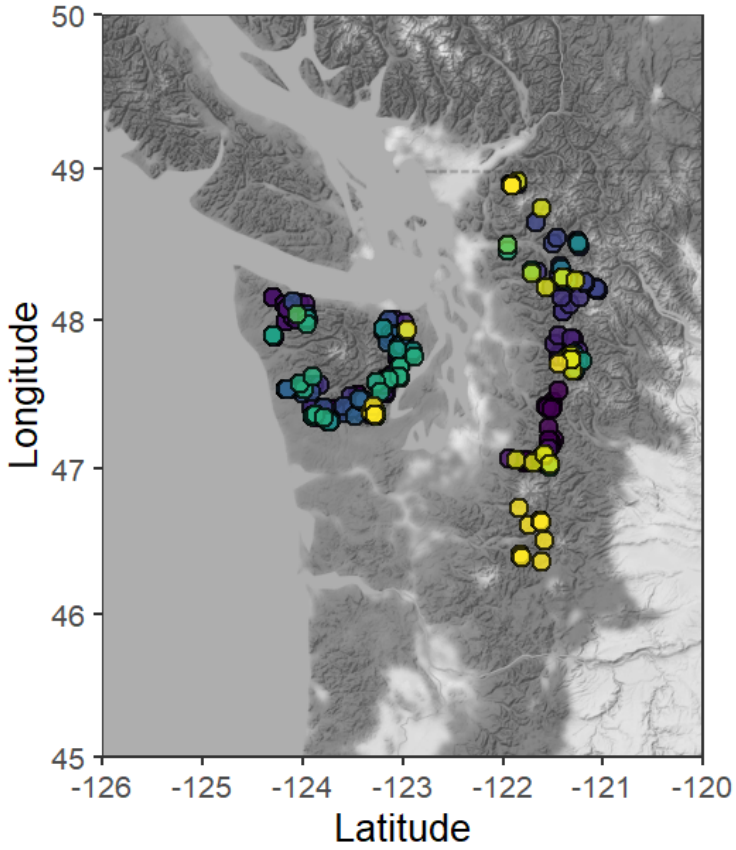
Making a map of salal plot locations

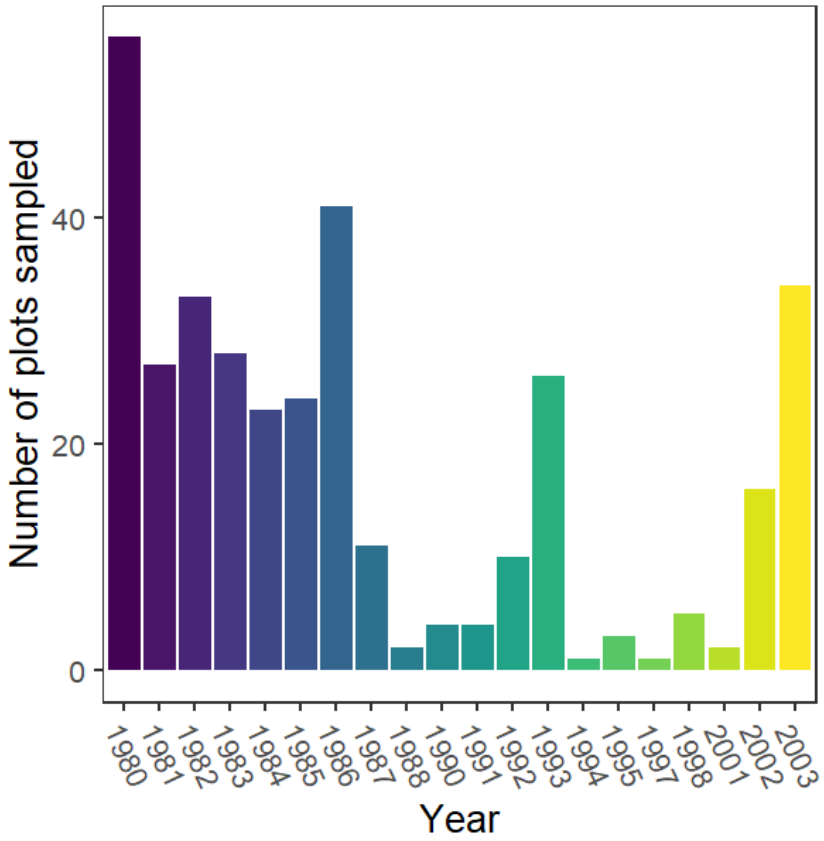
```
# It's hard to get a clear picture of how many plots were sampled per year
# To make it more clear, we can also show a histogram using the same colors
# that shows the number of sampled plots per year

# Make a bar graph showing the count of plots per year by specifying that you want
# year on the x axis (x=year)

histo <- ggplot(data=dat, aes(x=year, fill=year)) + geom_bar()
histo

# quickly change to the same color scheme as above

bhisto <- histo + scale_fill_viridis(discrete=TRUE, guide=FALSE)
bhisto

# Add better labels, remove the background grid, and change the x axis scale
# so the numbers are visible

chisto <- bhisto + labs(x="Year",y= "Number of plots sampled") +
          theme(panel.grid.major=element_blank(),panel.grid.minor=element_blank())+
          theme(axis.text.x=element_text(angle = -65, hjust = 0))
chisto
```



Making a map of salal plot locations

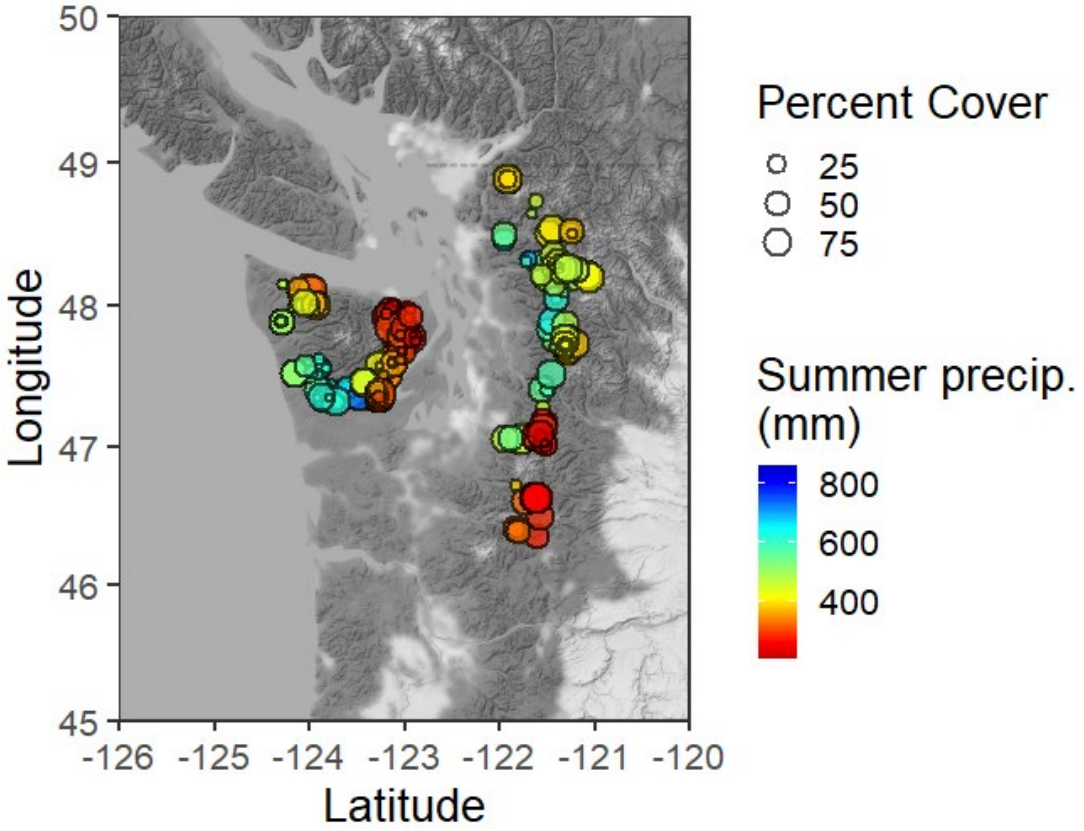
```
### And now we can place our map and histogram next to each other using the gridExtra

grid.arrange(scdmap,chisto, ncol=2)

### And now we really don't need the legend on the map, since the colors for each year

smap <- scdmap + scale_fill_viridis(discrete=TRUE, guide=FALSE)

## Voila ###

grid.arrange(smap,chisto, ncol=2)
```





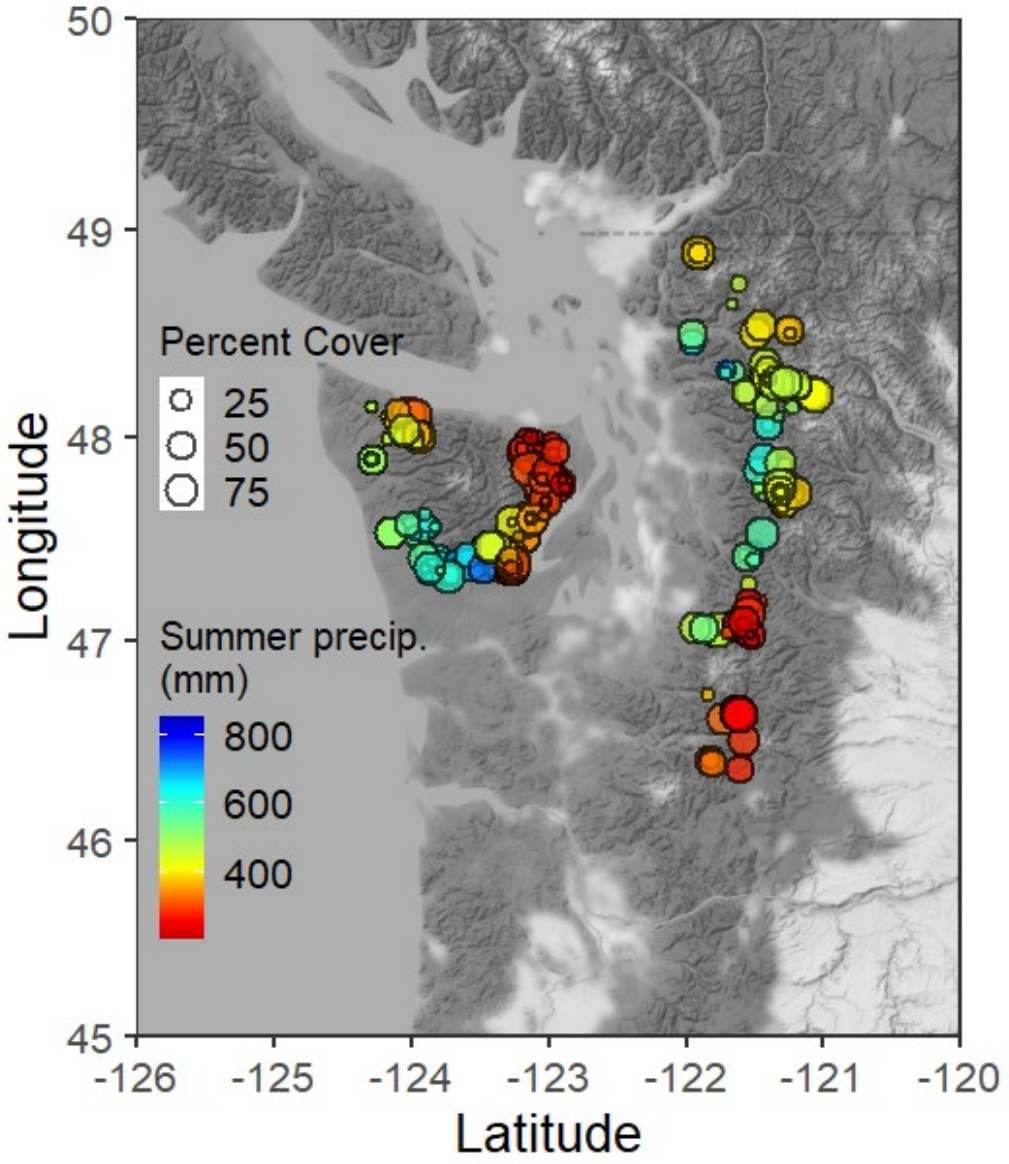
Making a map of salal plot locations

```
## Perhaps we would like to show how the % cover of Salal varies over plot locations
## and mean summer precipitation. We can color the plots by mean summer temperature,
## but map the size of the points to the cover variable

ptmap <- ggmap(myMap) + geom_point(data=dat, aes(x=long, y=lat, fill = MSP_av,
         size = cover), pch = 21, color = "black", stroke = 1.1, alpha = 0.7)
ptmap

## Still hard to distinguish colors - reds to blues are good for precipitation data

newmap <- ptmap  +  scale_fill_gradientn(colours= rev(blue2green2red(23)))
newmap

## Add correct labels

scdmap <- newmap + labs(x="Latitude",y= "Longitude", fill = "Summer precip.\n(mm)",
                        size = "Percent Cover")
scdmap
```



Making a map of salal plot locations

```
## Tweak the legend positions to suit our preference
## the coordinates for legend.position are x- and y- offsets from the bottom-left
## of the plot, ranging from 0 - 1

thrdmap <- scdmap + theme(strip.text.x = element_blank(),
                strip.background = element_rect(colour="none", fill="none"),
                legend.position=c(.19,.4))

thrdmap

## We can also remove the white background of the legends, and change the size of
## the legend titles

frthmap <- thrdmap + theme(legend.background = element_rect(fill=NA, size=0.5)) +
           theme(legend.title = element_text(size=15))

frthmap
```



Flowering of salal and seasonal temperatures

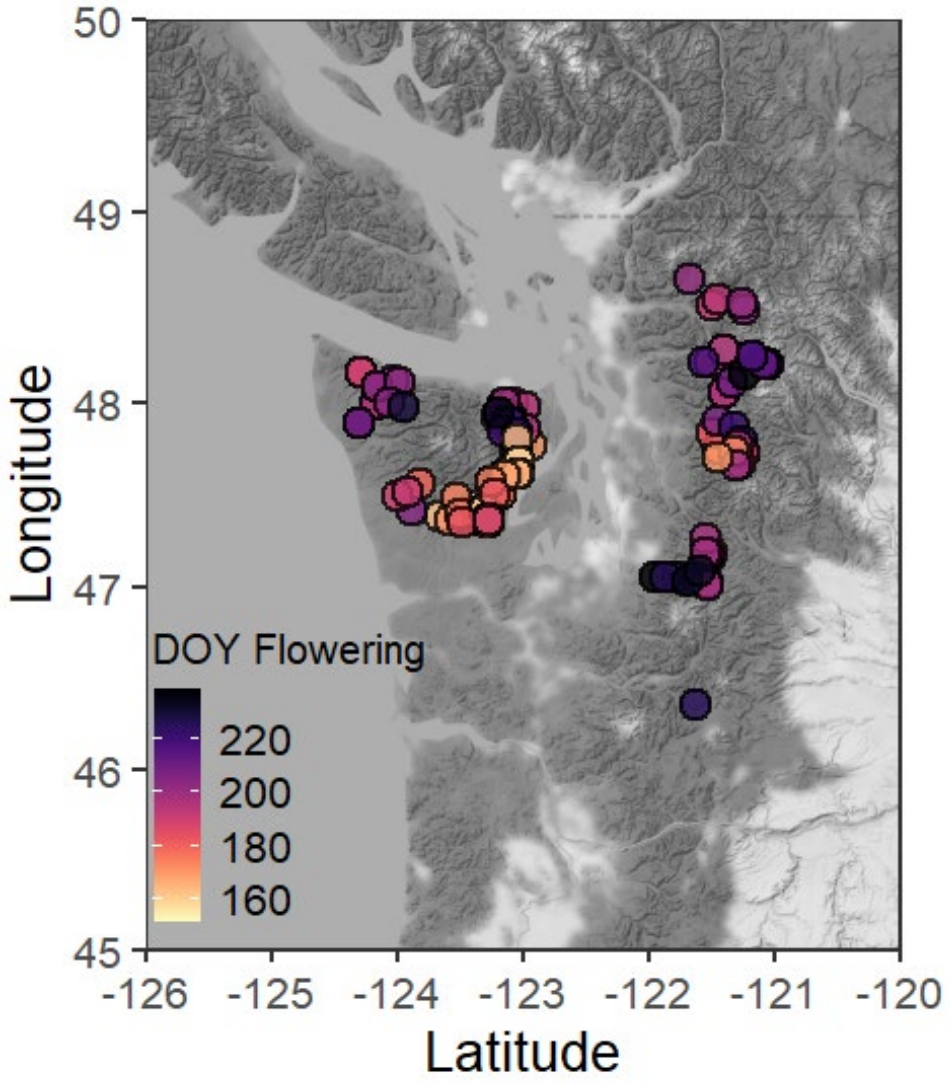
Use maps and graphs to compare day of year (DOY) of flowering observations to mean seasonal temperatures in that year.

PSA *** Normally these would be **based on hypotheses** and accompanied by **statistical tests of significance******

But that is a topic for different powerpoint...

Flowering of salal and seasonal temperatures

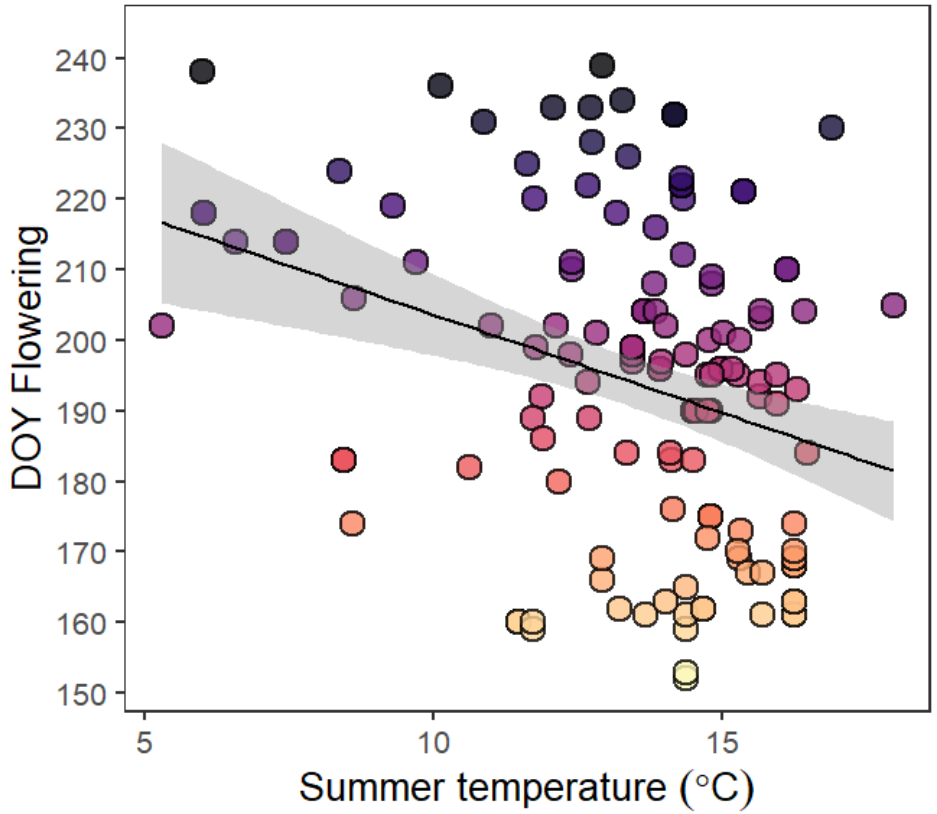
```
## create a new dataframe with only plots with flower observations

flower <- dat[dat$Phenophase=='Flower', ]

ptmap <- ggmap(myMap) + geom_point(data=flower, aes(x=long, y=lat, fill = DOY), pch = 21,
                                   color = "black", stroke = 1.1, size = 5, alpha = 0.8)
ptmap

### Let's create a color scheme that uses lighter colors for earlier flowering observations
## "direction = - 1" reverses the order of the color palette

newmap <- ptmap  + scale_fill_viridis(option="magma",  direction = - 1)
newmap

## Add correct labels and move legend

scdmap <- newmap + labs(x="Latitude",y="Longitude", fill = "DOY Flowering")
scdmap

thrdmap <- scdmap + theme(strip.text.x = element_blank(), strip.background =
           element_rect(colour="none", fill="none"),
           legend.position=c(.19,.19))
thrdmap

fthmap <- thrdmap  + theme(legend.background = element_rect(fill=NA, size=0.5)) +
          theme(legend.title = element_text(size=15))
fthmap
```



Flowering of salal and seasonal temperatures

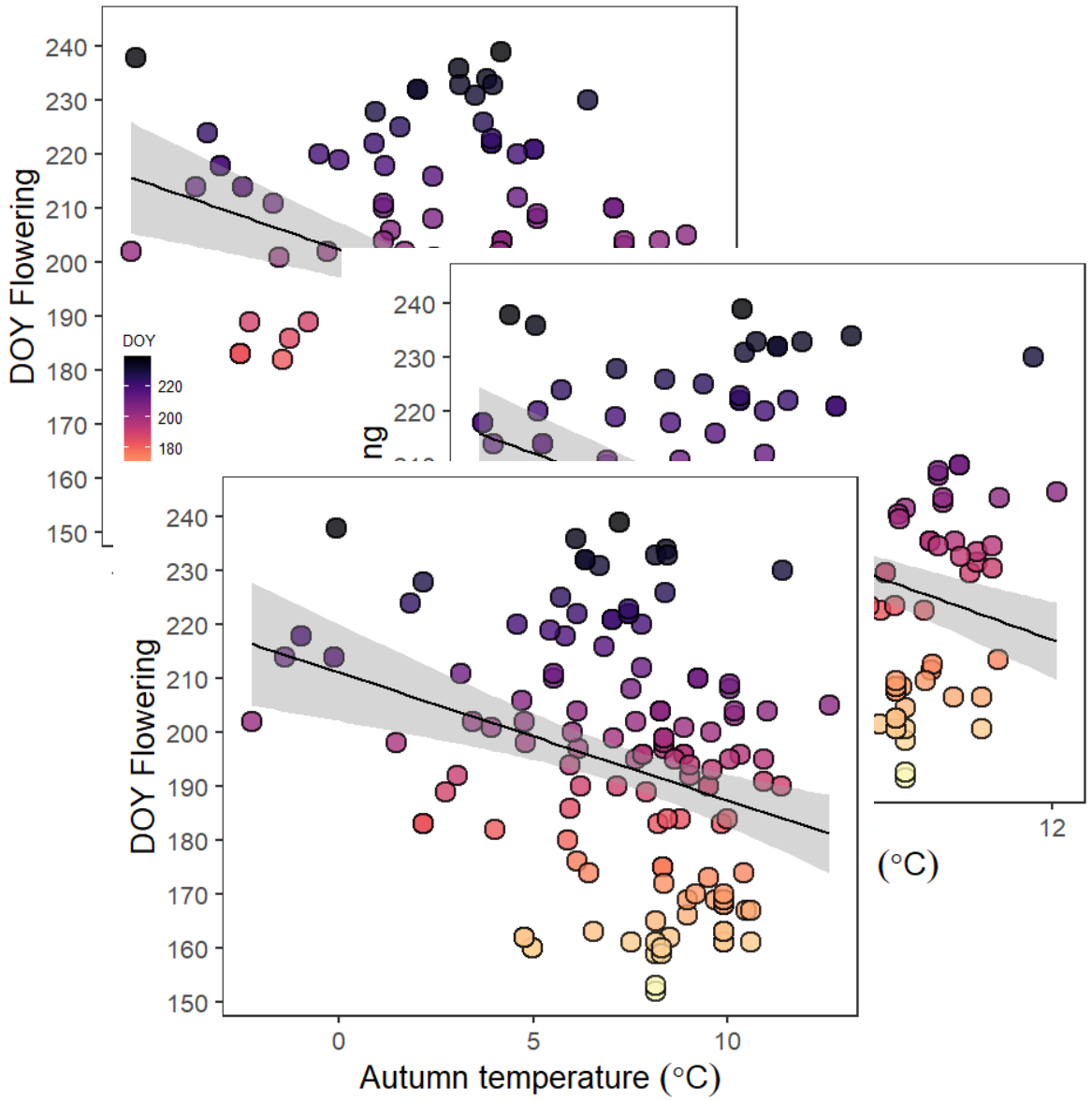
```
scat <- ggplot(data=flower, aes(x=summer, y=DOY)) + geom_point(data=flower,
        aes(x=summer, y=DOY, fill = DOY), pch = 21, color = "black", stroke = 1.1,
        size = 5.5, alpha = 0.8)
scat

scat2 <- scat  + scale_fill_viridis(option="magma",  direction = - 1, guide=FALSE)
scat2

## We can use geom_smooth to plot the linear relationship between summer temperature
## and the DOY that flowers were observed in plots. The grey band denotes the 95%
## confidence intervals of the linear relationship between summer temp. and DOY

scat3 <- scat2 + geom_smooth(method = "lm",color = "black")
scat3

## We can also alter the breaks and limits for the axes...
## Below we can change the y axis to start near the beginning of June (DOY 152)
## and go through the end of August (DOY 243)
## we can also use the breaks argument to designate how often the axes are labeled
## (every 10 units here)

scat4 <- scat3 + scale_y_continuous(breaks=seq(150,243,by=10), limits=c(152,243)) +
         theme(legend.position="none")

scat4
```

```
## Add better labels, remove the background grid

sum <- scat4 + labs(x=expression("Summer temperature "( degree*C)),
       y="DOY Flowering") + theme(panel.grid.major=element_blank(),
       panel.grid.minor=element_blank())
sum
```



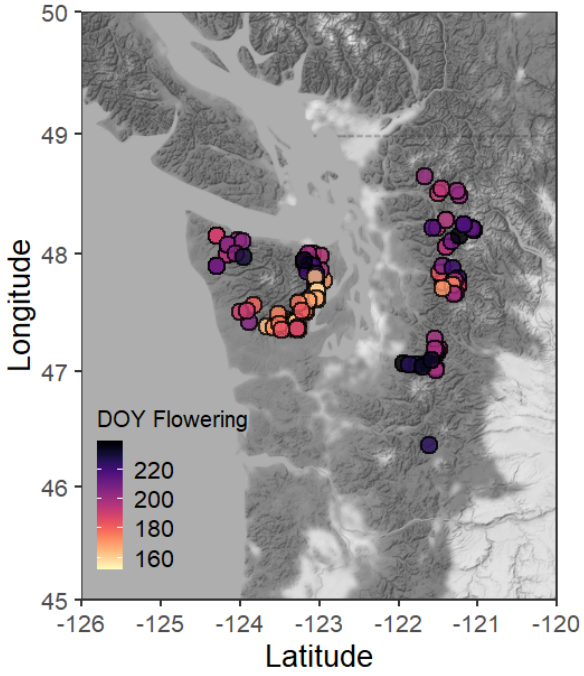
Flowering of salal and seasonal temperatures

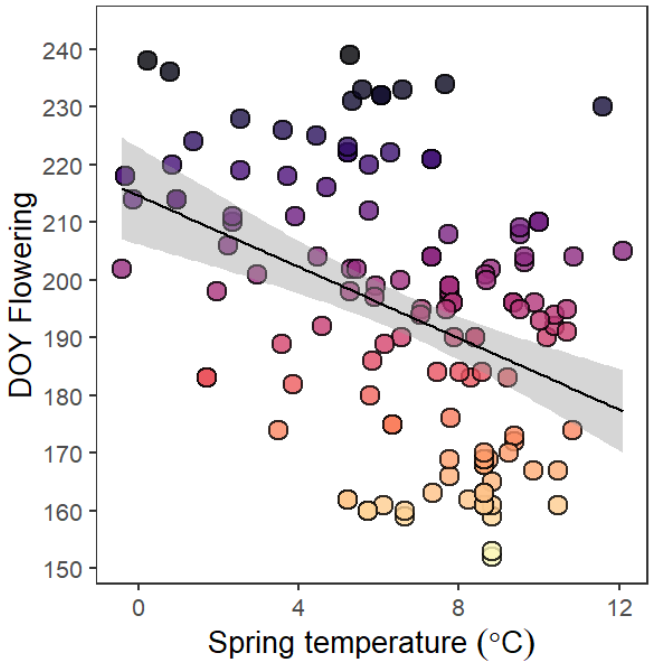
```
#Winter - let's put the legend back in for this one

scat <- ggplot(data=flower, aes(x=winter, y=DOY)) + geom_point(data=flower,
              aes(x=winter, y=DOY, fill = DOY), pch = 21, color = "black",
              stroke = 1.1,size = 5.5, alpha = 0.8)
scat2 <- scat + scale_fill_viridis(option="magma",  direction = - 1)
scat3 <- scat2 + geom_smooth(method = "lm",color = "black")
scat4 <- scat3 + scale_y_continuous(breaks=seq(150,243,by=10), limits=c(152,243))+
         theme(legend.position="none")
scat5 <- scat4 + labs(x=expression("Winter temperature "( degree*C)),
         y= "DOY Flowering") + theme(panel.grid.major=element_blank(),
         panel.grid.minor=element_blank())
winter<- scat5 + theme(strip.text.x = element_blank(),
         strip.background = element_rect(colour="none", fill="none"),
         legend.position=c(.08,.25))  + theme(legend.title = element_text(size=10)) +
         theme(legend.text = element_text(size=10))
winter

## Spring

scat <- ggplot(data=flower, aes(x=spring, y=DOY)) + geom_point(data=flower,
        aes(x=spring, y=DOY, fill = DOY), pch = 21, color = "black", stroke = 1.1,
        size = 5.5, alpha = 0.8)
scat2 <- scat  + scale_fill_viridis(option="magma",  direction = - 1, guide=FALSE)
scat3 <- scat2 + geom_smooth(method = "lm",color = "black")
scat4 <- scat3 + scale_y_continuous(breaks=seq(150,243,by=10), limits=c(152,243)) +
        theme(legend.position="none")
sprg <- scat4 + labs(x=expression("Spring temperature "( degree*C)),
        y= "DOY Flowering") + theme(panel.grid.major=element_blank(),
        panel.grid.minor=element_blank())
sprg

## Fall

scat <- ggplot(data=flower, aes(x=fall, y=DOY)) + geom_point(data=flower, aes(x=fall,
        y=DOY, fill = DOY), pch = 21, color = "black", stroke = 1.1, size = 5.5,
        alpha = 0.8)
scat2 <- scat  + scale_fill_viridis(option="magma",  direction = - 1, guide=FALSE)
scat3 <- scat2 + geom_smooth(method = "lm",color = "black")
scat4 <- scat3 + scale_y_continuous(breaks=seq(150,243,by=10), limits=c(152,243)) +
         theme(legend.position="none")
fll <- scat4 + labs(x=expression("Autumn temperature "( degree*C)),y= "DOY Flowering") +
       theme(panel.grid.major=element_blank(),panel.grid.minor=element_blank())
fll
```

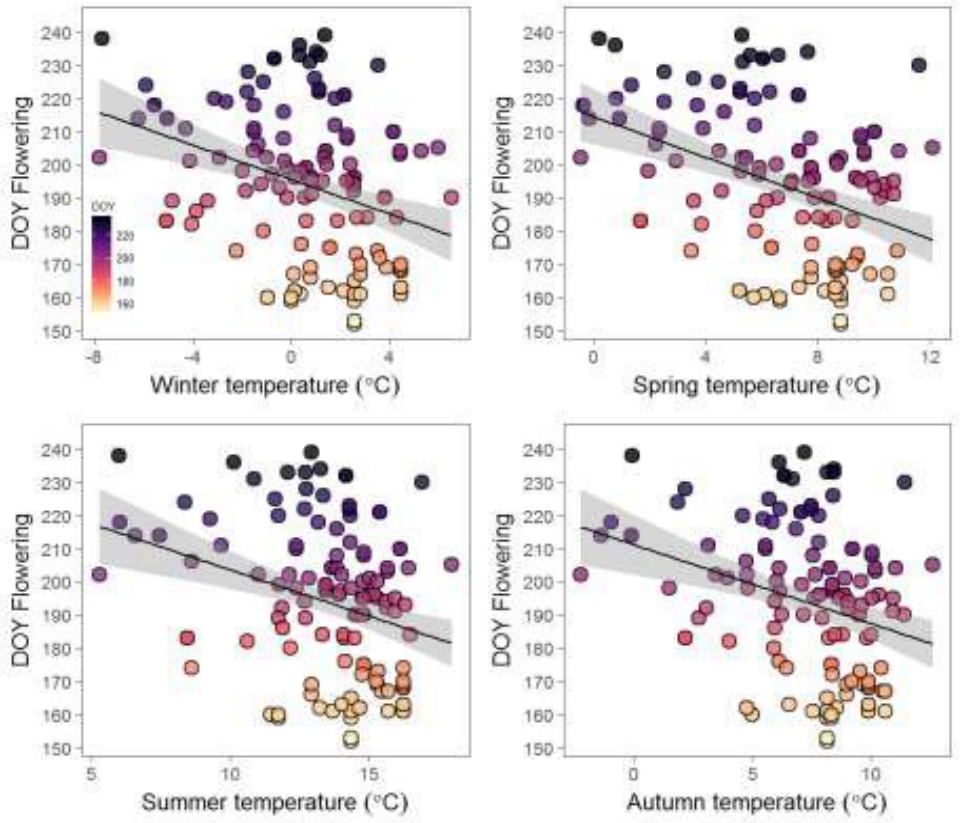


Flowering of salal and seasonal temperatures

```
## And we can place our map and a graph together using the gridExtra package

grid.arrange(fthmap,sprg,ncol = 2)

## Or perhaps we would like to show the relationship between flowering and ALL four
## seasons on a four panel graph....

grid.arrange(winter, sprg, sum, fll, nrow=2, ncol = 2)
```

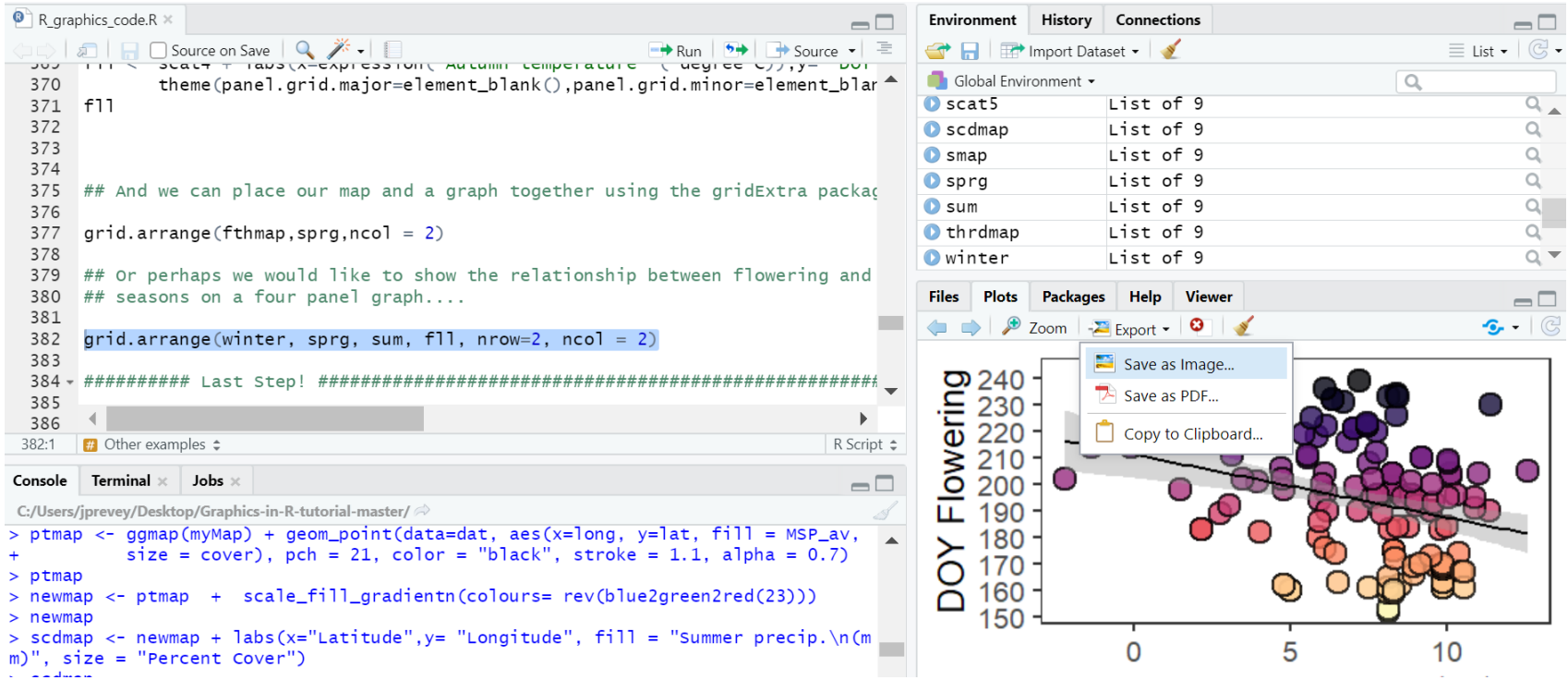






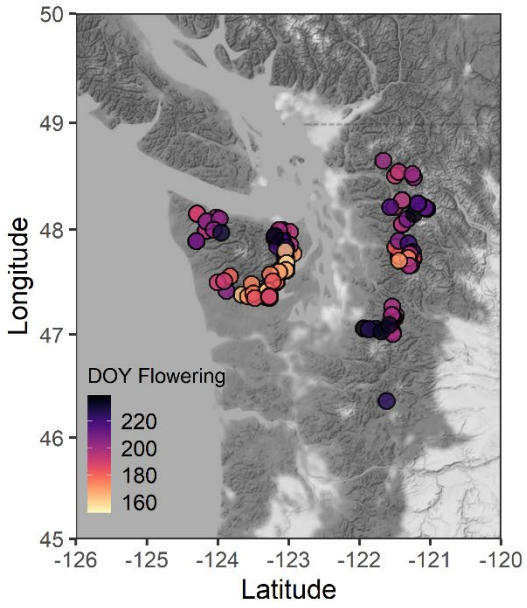
Exporting your images: best practices

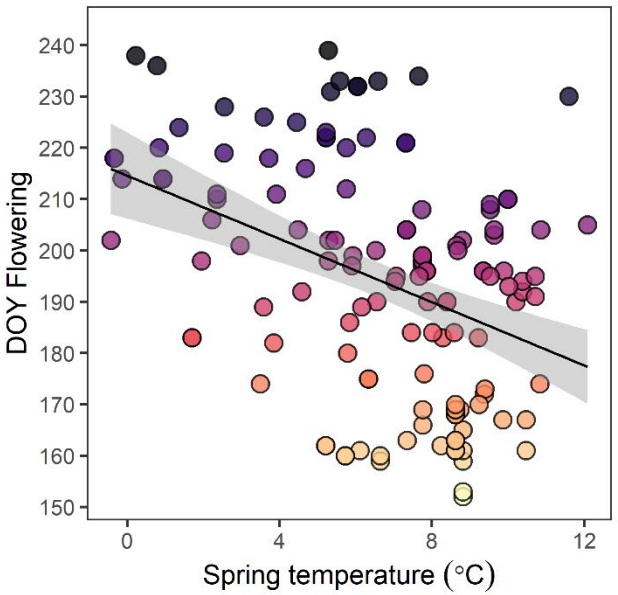
- .jpgs and .pngs are low resolution, but easy to save and open in a variety of programs - sufficient for online or presentations
- However, many journals require image resolution of 300 ppi and above…
- **High resolution .tiff files are good for raster images, and .pdfs are good for vector-based images**
- Metafiles are also great for vector-based images, but they can be buggy…….



Exporting your images: best practices

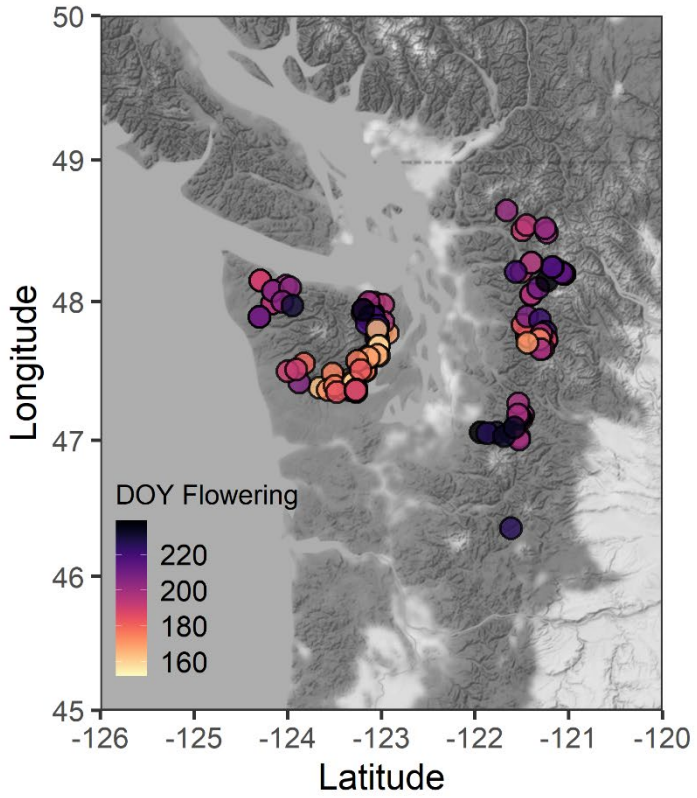
```
########## Last Step! ############################################################################

## Save high resolution copies of your images, after figuring out the correct sizing
## using the 'Export' option under 'Plots'
## 300 dpi or greater is good for publication quality graphs

## place the below line of the code before the line of code that displays the graphics you
## would like to save

tiff("Example.tiff", width = 12.9, height = 6.30, pointsize = 12, units = 'in', res = 300)

grid.arrange(fthmap,sprg,ncol = 2) ## then the line of code that displays the graphs

dev.off()  ## now the displayed graphics are saved to a file with the above name file
```

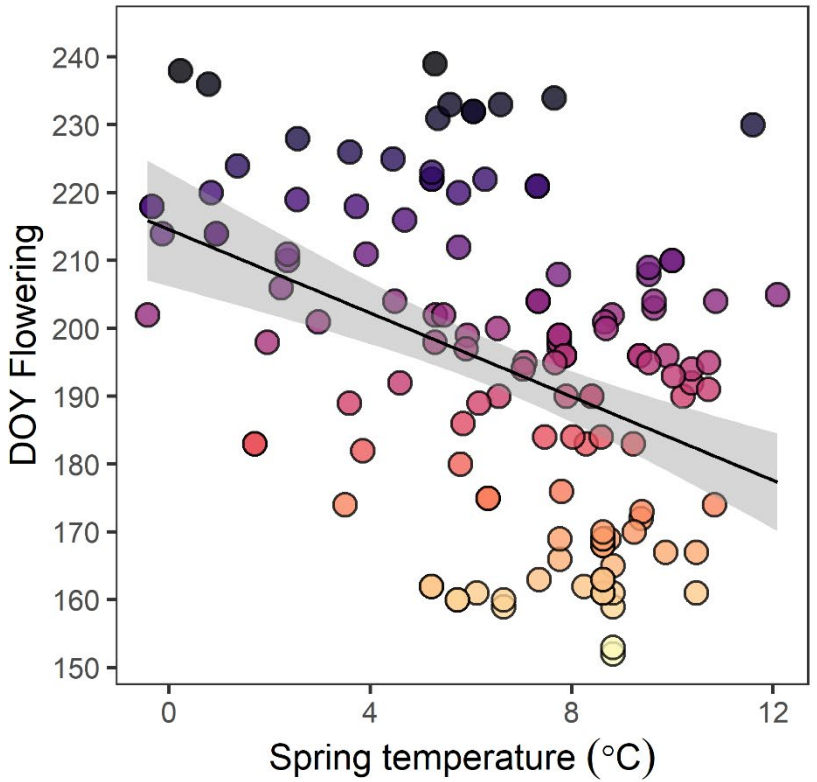




Take home messages

- Use color and shape to tell compelling stories with your graphs
- Use consistent colors across multiple figures to create continuity
- Less is more





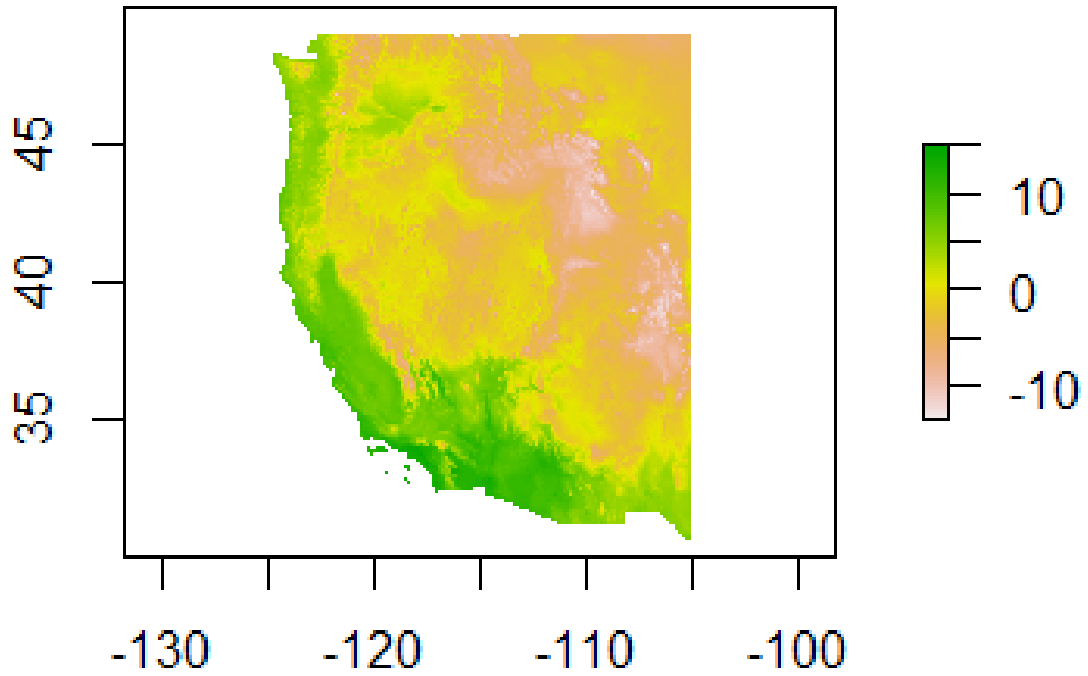
Visualizing Raster data in R

Adapted from a tutorial by Scott Anderson, USGS Hydrologist (and R pro!)

```
## A very brief introduction to visualizing raster data in R
## Adapted from code by Scott W. Anderson - USGS Washington Water Science Center

install.packages(c("ggspatial", "prism", "raster"))

library(prism)
library(raster)
library(ggspatial)

## We can download temperature data from PRISM at a 4km data using the prism package
## More information about the prism package here: https://cran.r-project.org/web/packages/prism/prism.pdf

## Tell prism where to store the rasters
options(prism.path = "C:\\Users\\jprevey\\Desktop\\Prism")
```

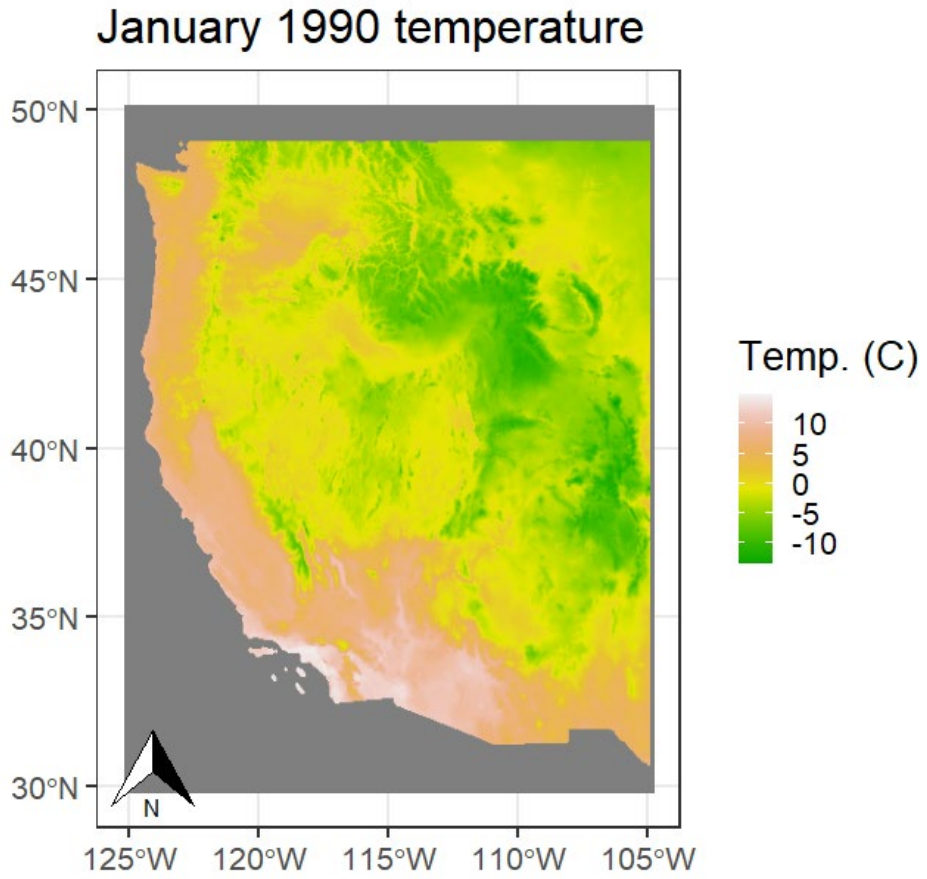
Visualizing Raster data in R

```
## Tell prism where to store the rasters
options(prism.path = "C:\\Users\\jprevey\\Desktop\\Prism")

## Download January mean temperature for 1990 through 2000
get_prism_monthlys(type="tmean", years = 1990:2000, mon = 1, keepZip=FALSE)

## Show R where to get the .bil files to display as a rasters
prism1990 = "C:\\Users\\jprevey\\Desktop\\Prism\\PRISM_tmean_stable_4kmM3_199001_bil\\PRISM_tmean_stable_4kmM3_199001_bil.bil"
prism2000 = "C:\\Users\\jprevey\\Desktop\\Prism\\PRISM_tmean_stable_4kmM3_200001_bil\\PRISM_tmean_stable_4kmM3_200001_bil.bil"

# Read in 1990 data

tRas90 = raster(prism1990)
tRas90NW = crop(tRas90, extent(c(-125, -105, 30, 52)))

# Graph the raster quickly using base r graphics

plot(tRas90NW)
```



Visualizing Raster data in R

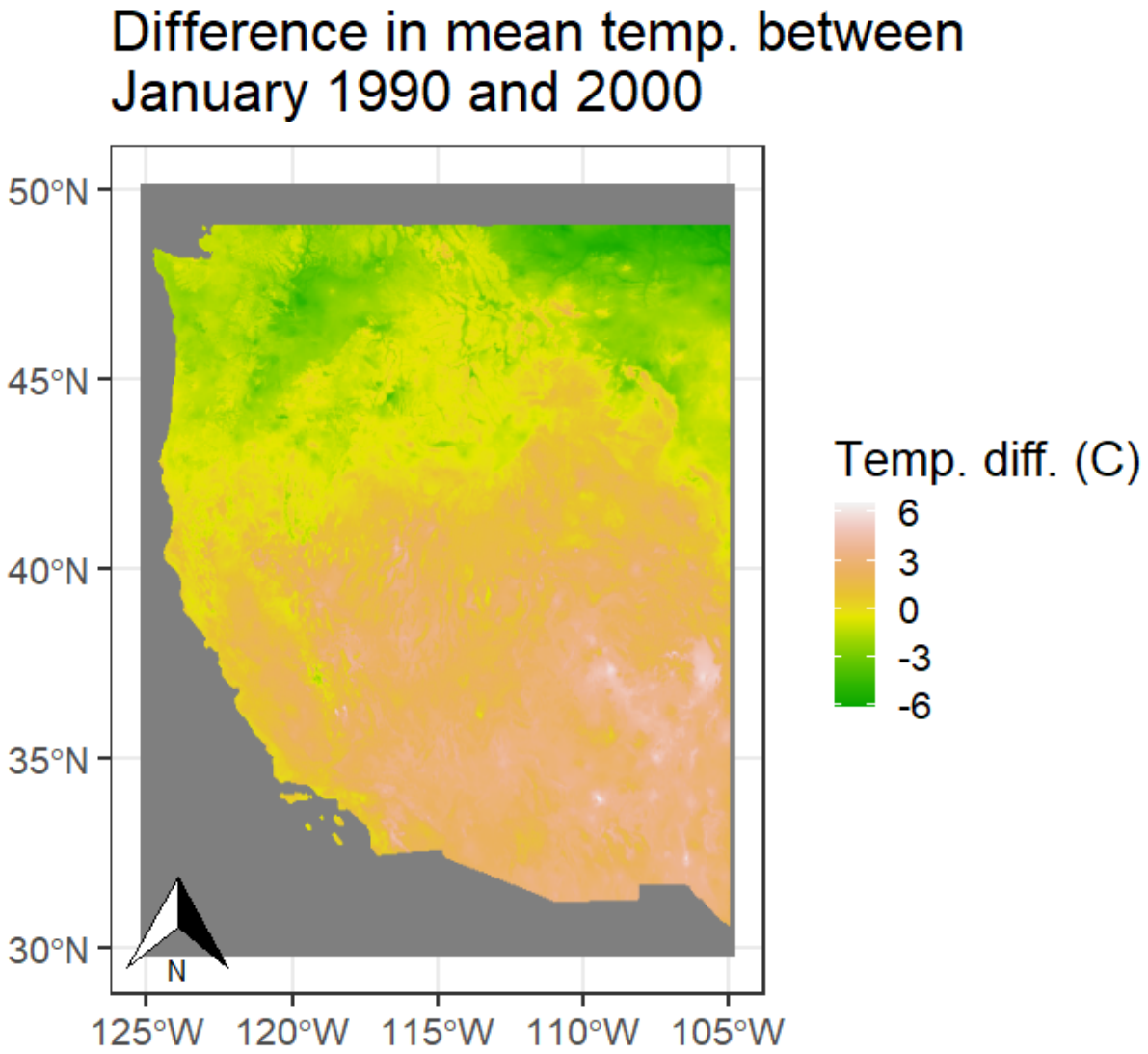
```
# Graph the data in ggplot, adding a north arrow

ggplot() + layer_spatial(tRas90NW) +
  scale_fill_gradientn(colors = terrain.colors(10), name = 'Temp. (C)') +
  annotation_north_arrow() +
  labs(title = 'January 1990 temperature')
```



Visualizing Raster data in R

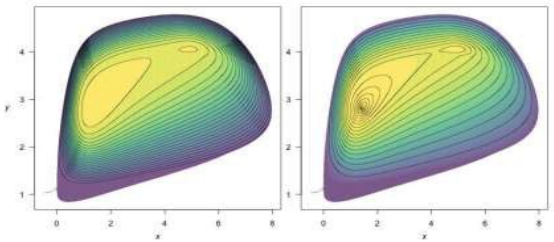
```
## Now suppose you want to compare January temperatures across the decade

tRas00 = raster(prism2000)
tRas00NW = crop(tRas00, extent(c(-125, -105, 30, 52)))

# Graph the 2000 raster quickly using base r graphics

plot(tRas00NW)

# Use basic r commands to get the difference in monthly mean temperatures
# between 2000 and 1990

tRasdiff <- tRas00NW - tRas90NW

# Graph the data in ggplot, adding a north arrow

ggplot() + layer_spatial(tRasdiff) +
  scale_fill_gradientn(colors = terrain.colors(10), name = 'Temp. diff. (C)') +
  annotation_north_arrow() +
  labs(title = 'Difference in mean temp. between \nJanuary 1990 and 2000')
```

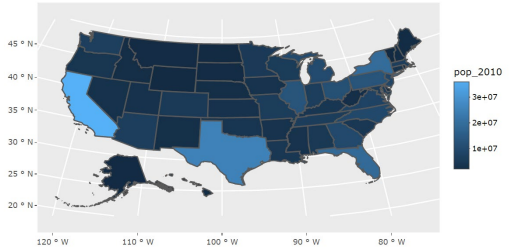
Visualizing Raster data in R



But can we make this graph more beautiful???

Going further…





- More R graphing websites:
  - http://zevross.com/blog/2014/08/04/beautiful-plotting-in-r-a-ggplot2-cheatsheet-3/
    - Some good info for customizing ggplots
  - https://www.r-graph-gallery.com/
    - Tons of examples of beautiful images and graphs of all types, with reproducible codes
    - Learn how to create interactive charts
  - https://bhaskarvk.github.io/user2017.geodataviz/notebooks/03-Interactive-Maps.nb.html
    - Several methods for creating interactive maps
- Visualizing raster data in R
  - https://cran.r-project.org/web/packages/raster/raster.pdf
  - https://oscarperpinan.github.io/rastervis/

- Remember - Google is your friend!

Citations

- D. Kahle and H. Wickham. ggmap: Spatial Visualization with ggplot2. The R Journal, 5(1), 144-161. URL: http://journal.r-project.org/archive/2013-1/kahle-wickham.pdf
- H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer- Verlag New York, 2009.
- Simon Garnier (2018). viridis: Default Color Maps from 'matplotlib'. R package version 0.5.0. https://CRAN.R-project.org/package=viridis
- Tim Keitt (2012). colorRamps: Builds color tables. R package version 2.3. https://CRAN.R-project.org/package=colorRamps
- Erich Neuwirth (2014). RColorBrewer: ColorBrewer Palettes. R package version 1.1-2. https://CRAN.R-project.org/package=RColorBrewer