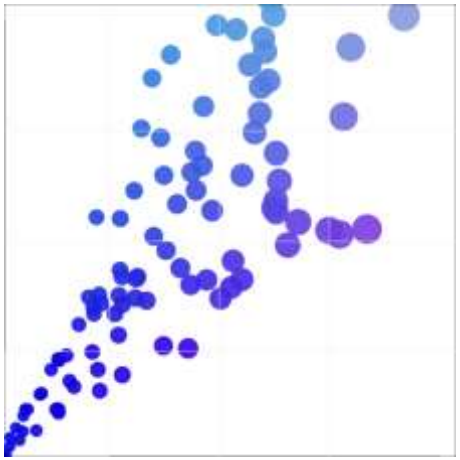
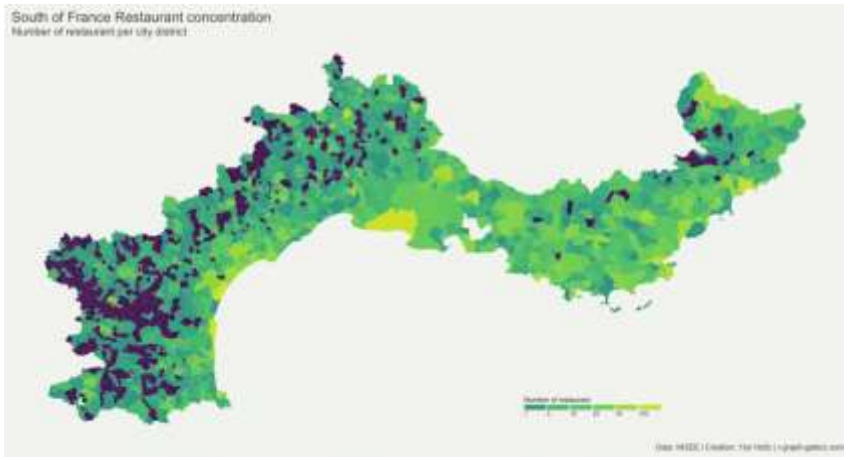
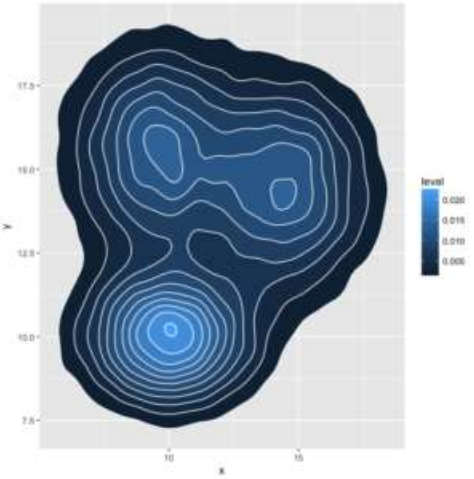
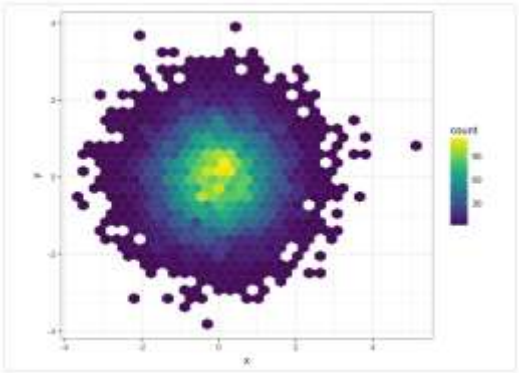
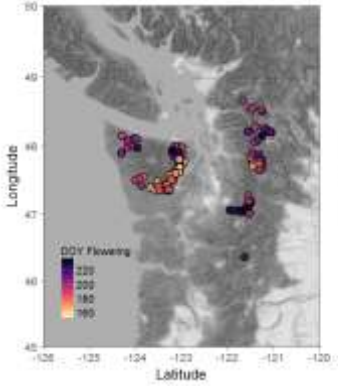
Making beautiful graphics in R

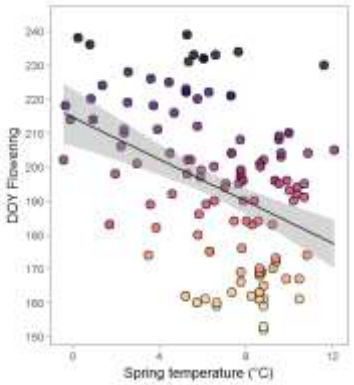












Above images from: www.r-graph-gallery.com

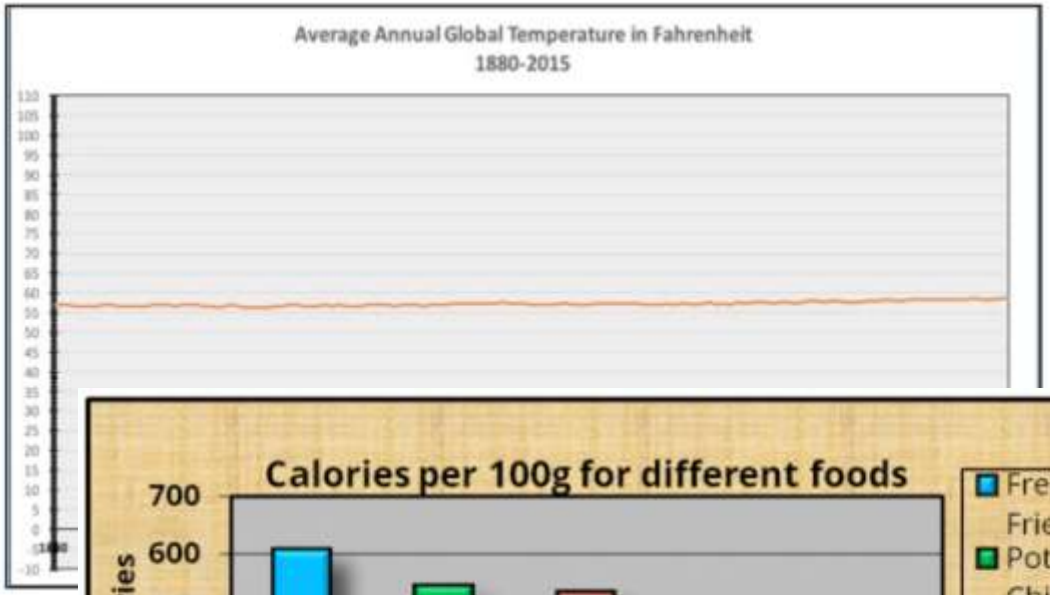
Janet Prevéy - USDA Forest Service, Pacific Northwest Research Station, Olympia, WA

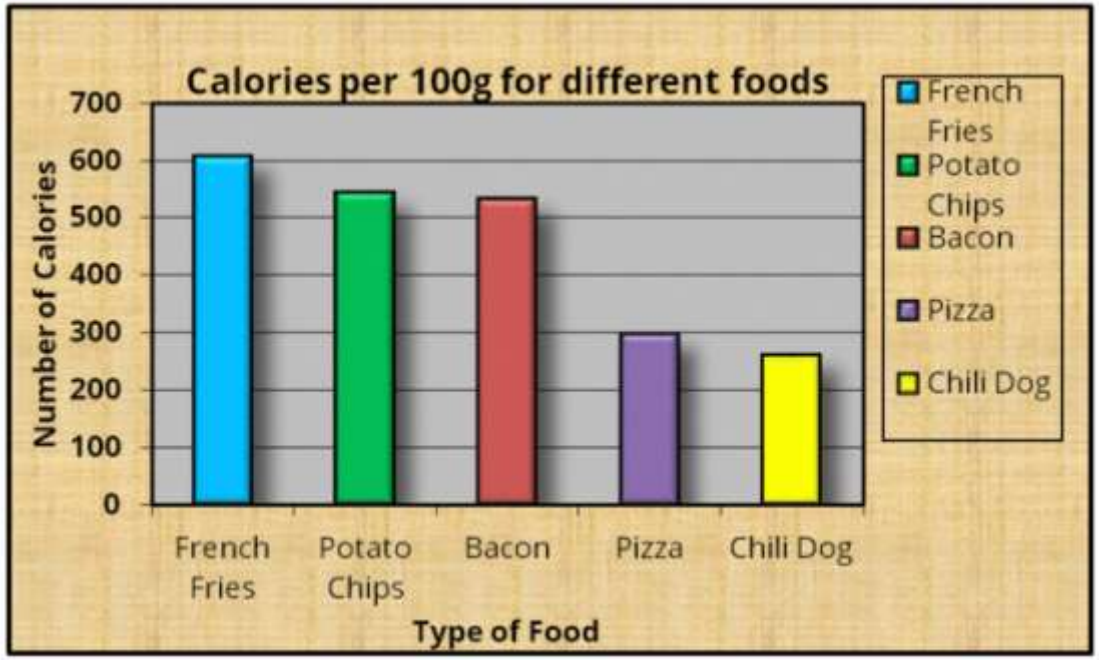
Why?

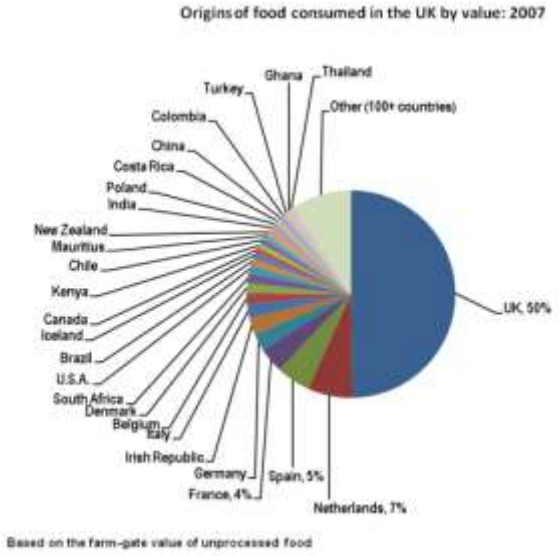
- One graph says a thousand words!
- Stand out when giving presentations
- Clear, consistent graphics and symbols help tell your research story
- Templates can be used again and again to quickly produce quality graphics

Bad graphics:

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

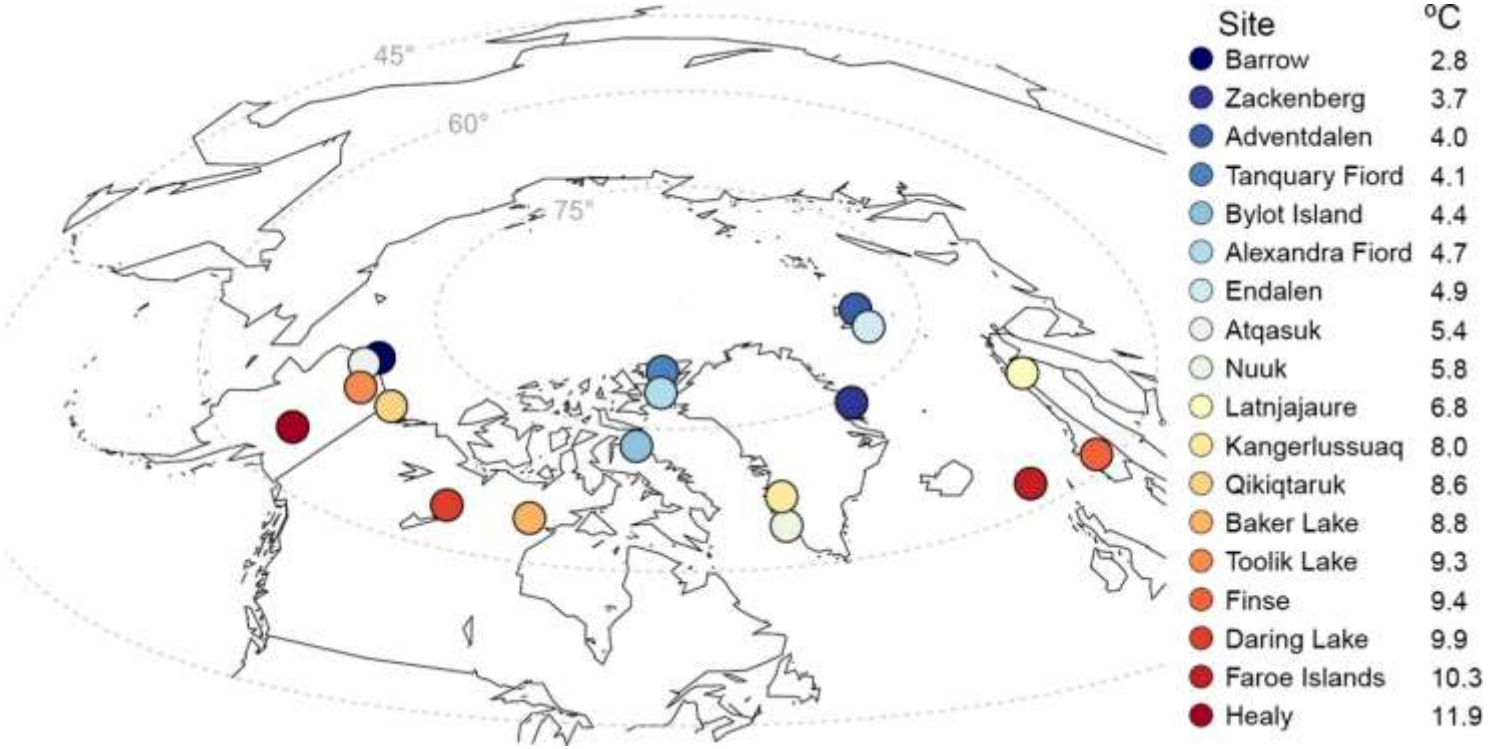






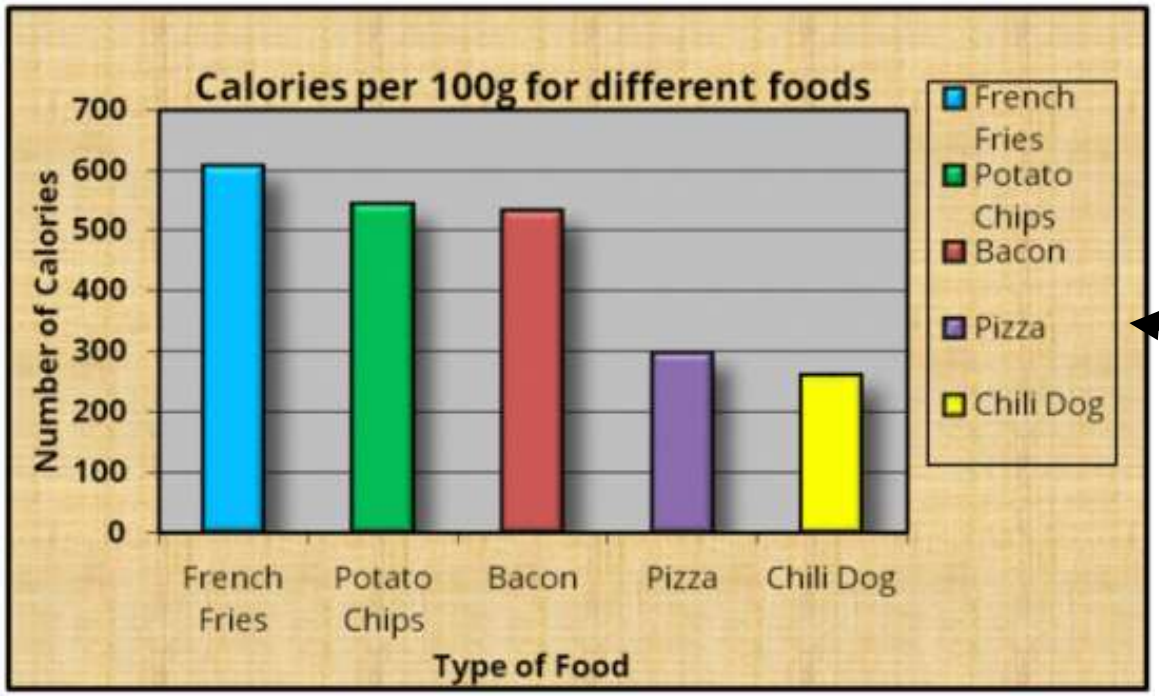
Keys for effective graphics:

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



Keys for effective graphics:

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

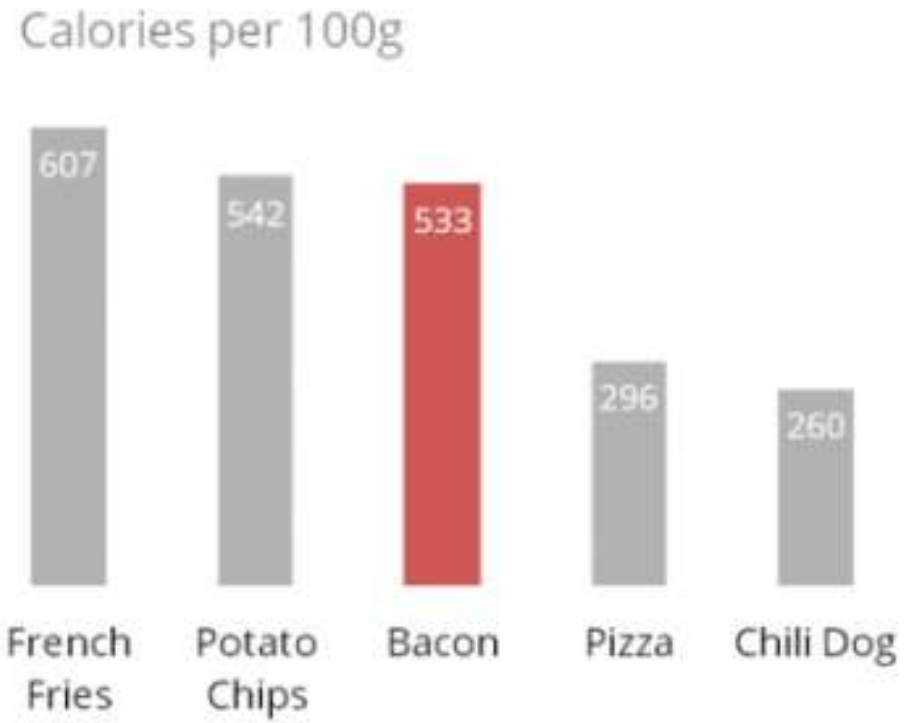


Yuck!

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

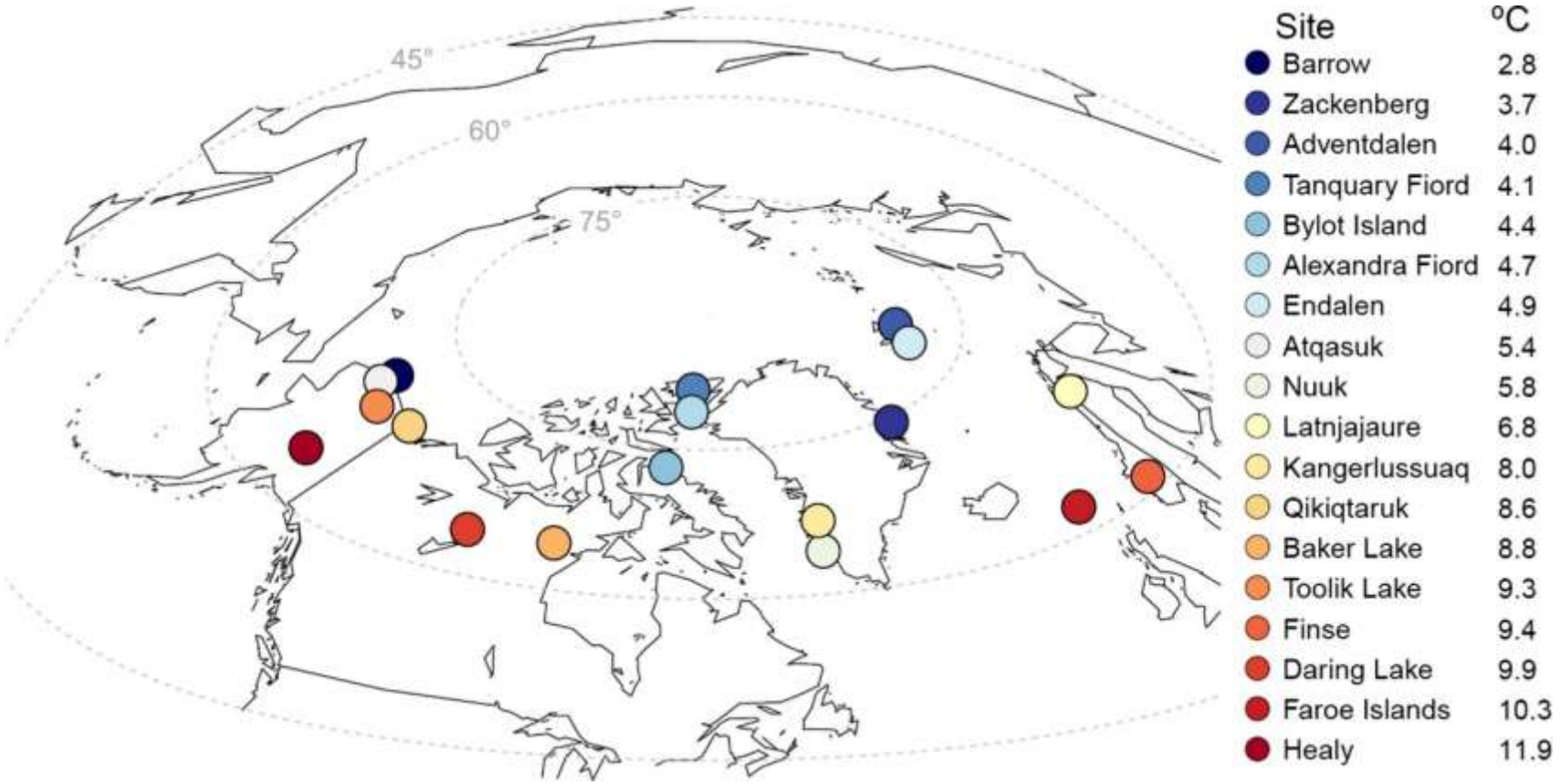
Keys for effective graphics:

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

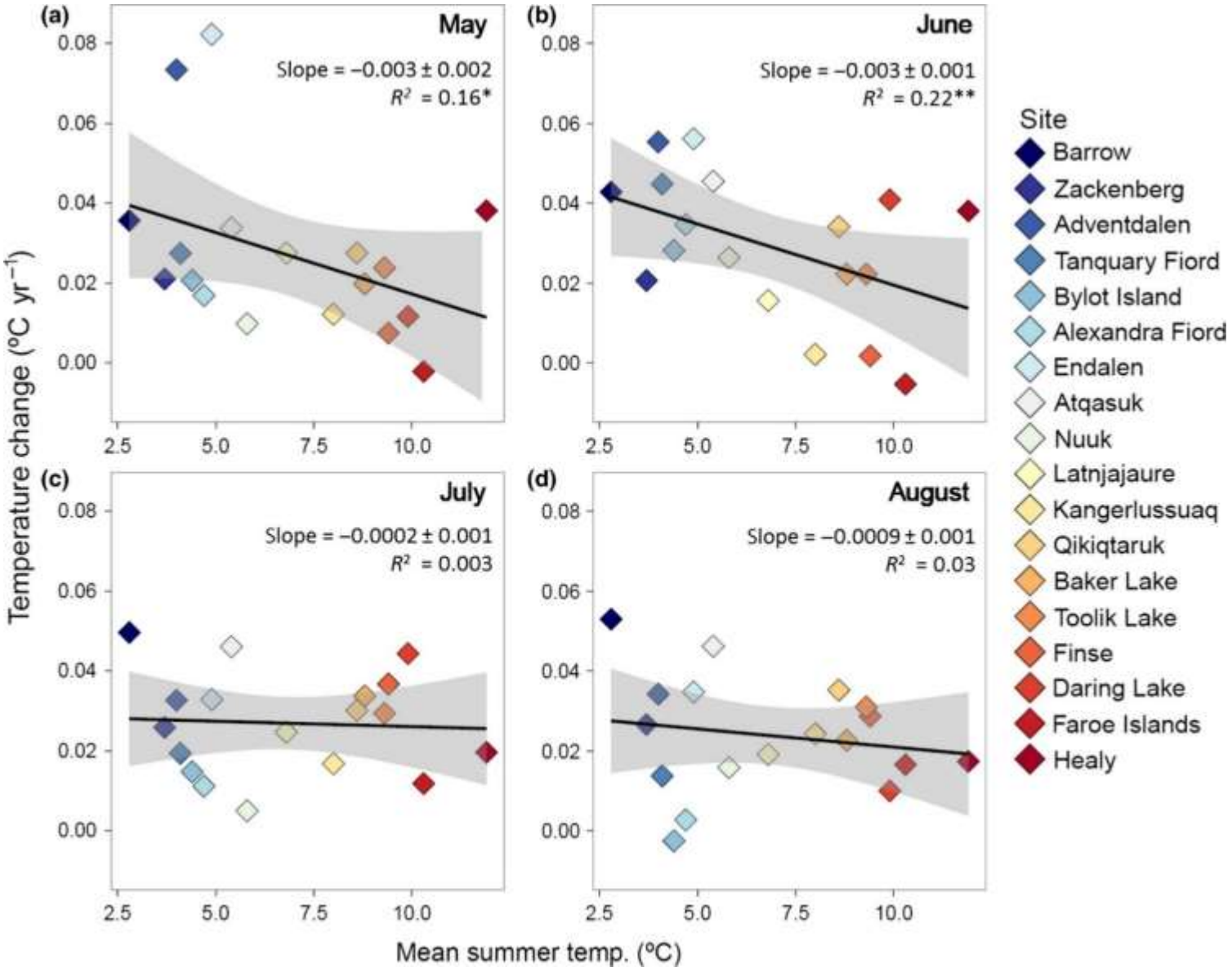


Better!

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
- Next steps for R graphics aficionados…
- Questions / trouble-shooting

Background on the sample data file: r6_salal_data.csv

- 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

- I matched this data to temperature data from DAYMET

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

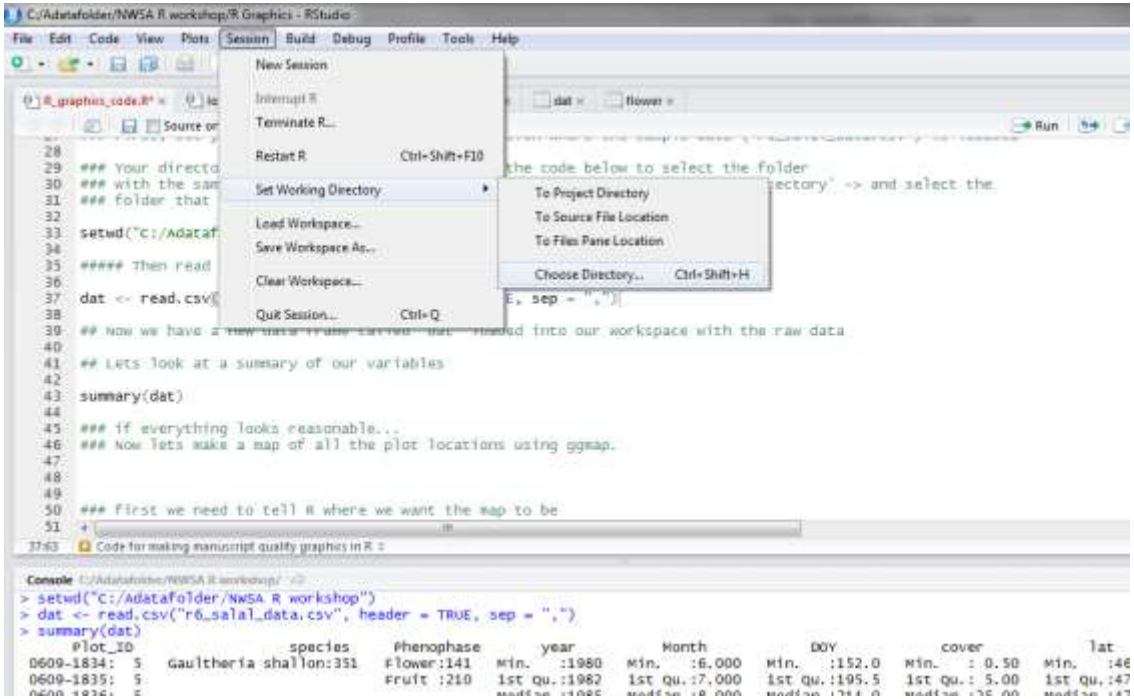
Install and load packages…

```
install.packages('ggplot2') #one of the most popular graphics packages

if(!requireNamespace("devtools")) install.packages("devtools")
devtools::install_github("dkahle/ggmap")  # ggmap draws maps that work seamlessly w
install.packages('RColorBrewer') #some nice color palette options, website: http://
install.packages('viridis') #other beautiful color options - website: https://cran.
install.packages('wesanderson') #color palettes inspired by Wes Anderson movies: ht
install.packages('colorRamps')#nice rainbow color palettes: https://cran.r-project.
install.packages('gridExtra') #helps to place graphs together on the same page


#### Load the packages we just installed

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……

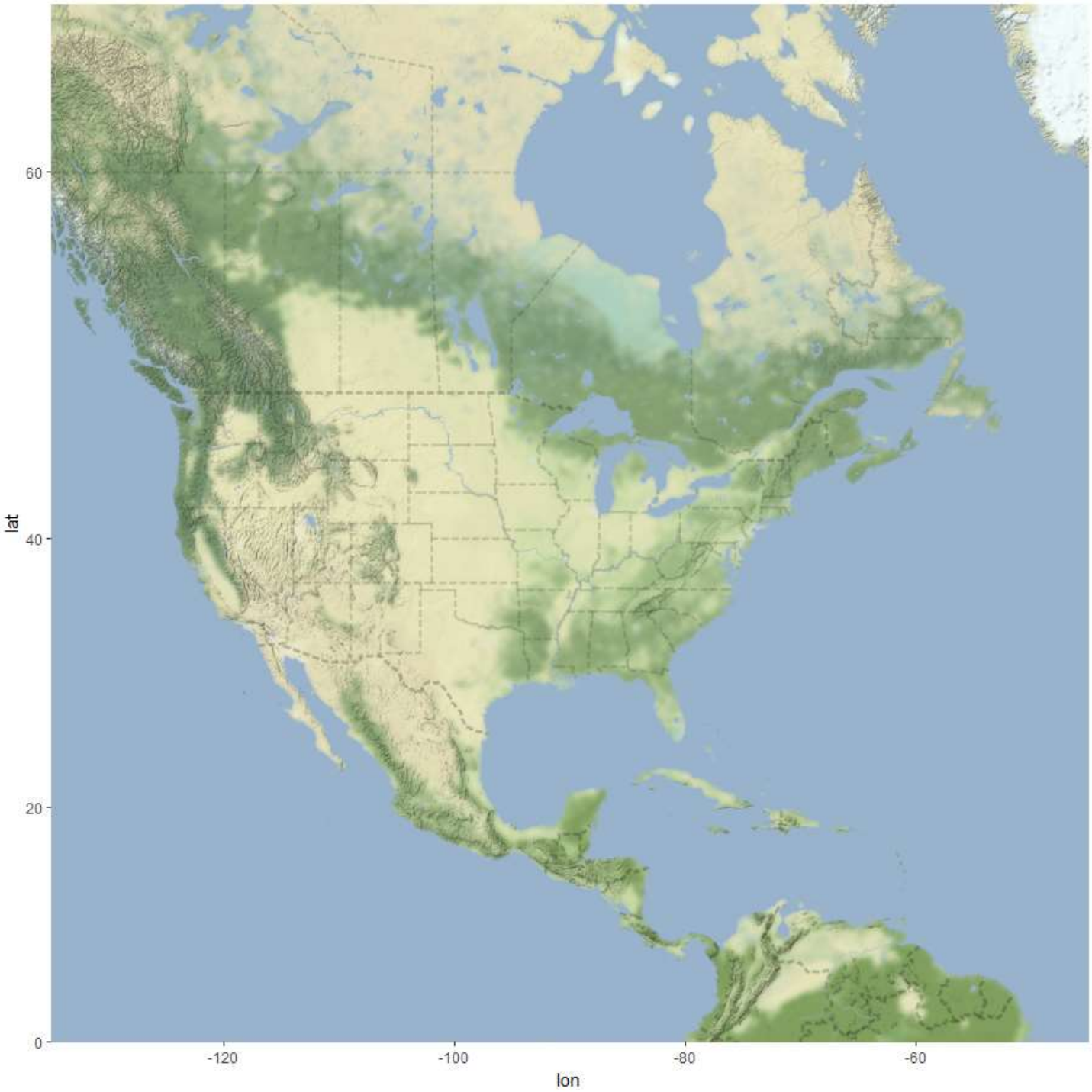
Read in the data and check it out….




```
##### Then read in the data we will use to create maps and graphs:

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)
```

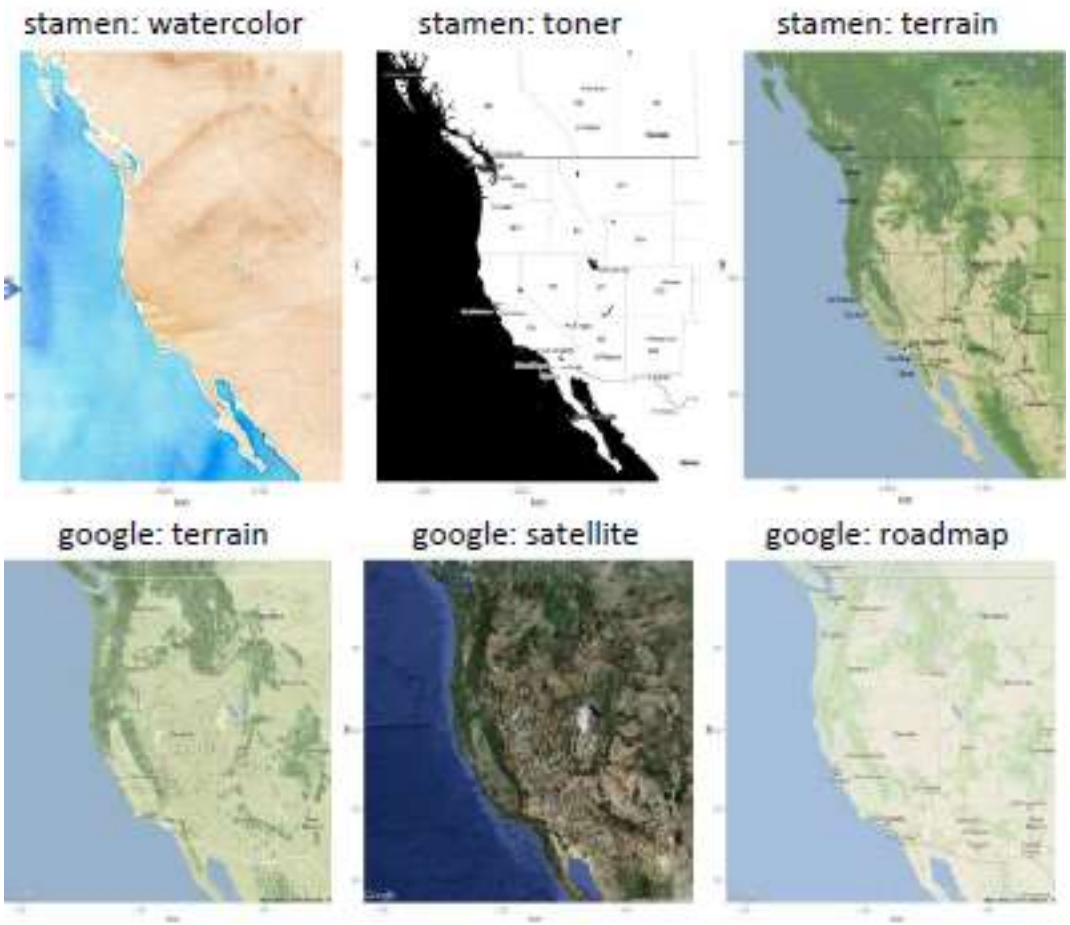
Let's make a map!

```
### Now lets make a map of all the plot locations using ggmap.


### first we need to tell R where we want the map to be -
## we can define a bounding box for our map here, called 'myLocation'
## the lats and longs are ordered thus:

## myLocation <- c(left long., bottom lat., right long., upper lat.)

myLocation <- c(-132,16,-47,56)

## then use the 'get_map' command to grab rasters from the internet to make your map

myMap <- get_map(location=myLocation,
                 source="stamen", maptype="terrain-background",crop=FALSE)

##view the map you just made!

ggmap(myMap)
```

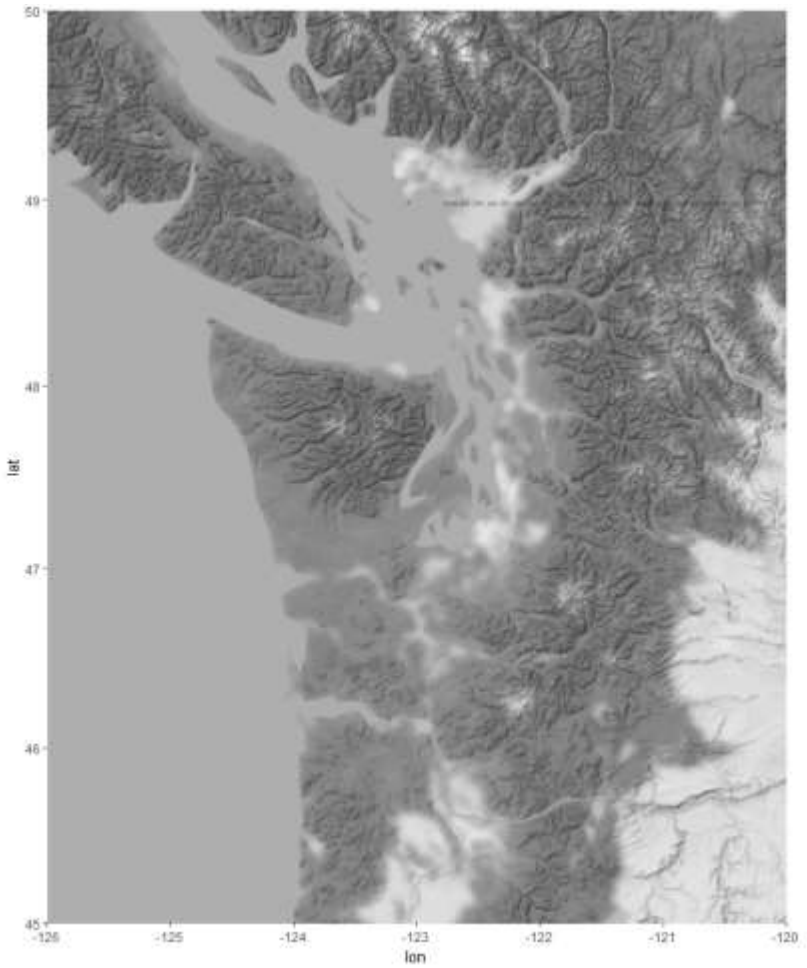

60
40
20
0
lat
-120
-100
-80
-60
lon

ggMap options….

- Googlemaps and Stamen maps give TONS of options
- https://cran.r-project.org/web/packages/ggmap/ggmap.pdf
- http://maps.stamen.com/#toner/12/37.7706/-122.3782

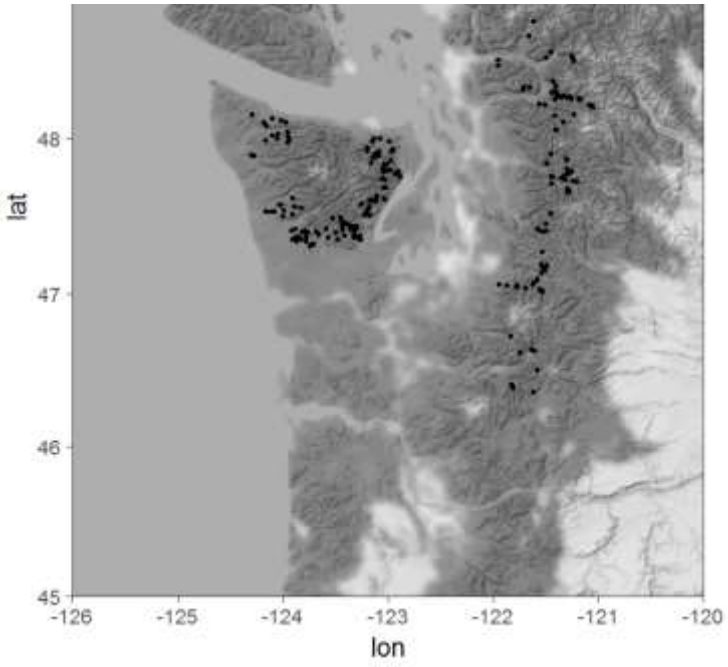


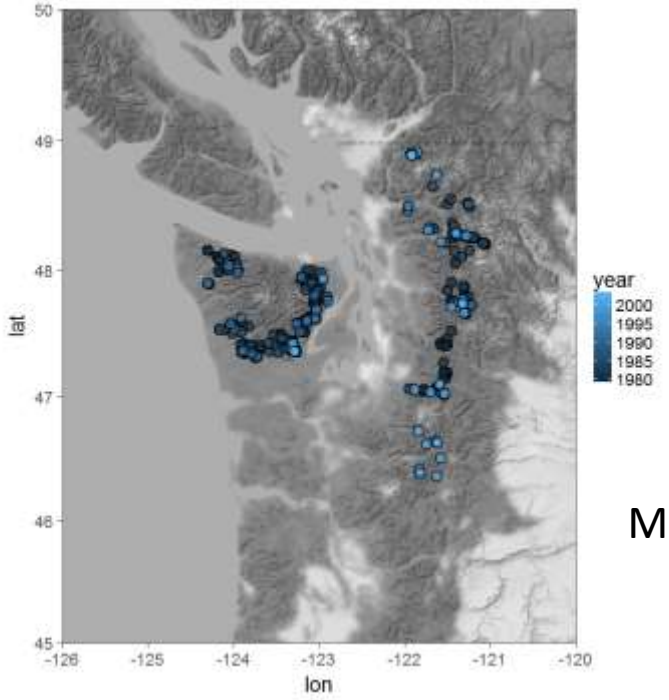
```r
### And all maps can be made into black-and-white maps by specifying color = "bw"

### Lets move forward with a black and white map for now - and lets zoom in on western Washington:

myLocation <- c(-126,45,-120,50)

myMap <- get_map(location=myLocation,
                 source="stamen", maptype="terrain-background",color = "bw", crop=TRUE)

ggmap(myMap)
```



```
### We will set a theme for all the graphs and maps in ggplot and ggmaps below to make sure we have white backg
### and larger than normal axes labels
## 'theme_bw' converts the normal grey background to white, and 'base_size=...' indicates the size of labels

theme_set(theme_bw(base_size=20))

## and now we add the plot locations from the .csv of data to our map with ggplot

ptmap <- ggmap(myMap) + geom_point(data=dat, aes(x=long, y=lat))

## now we've created the graphic called 'ptmap', and we can type the name below to view the image

ptmap
```

```
### 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 p
### 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 =
ptmap
```





Meeehhhhhhh...

A note on colors….

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

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

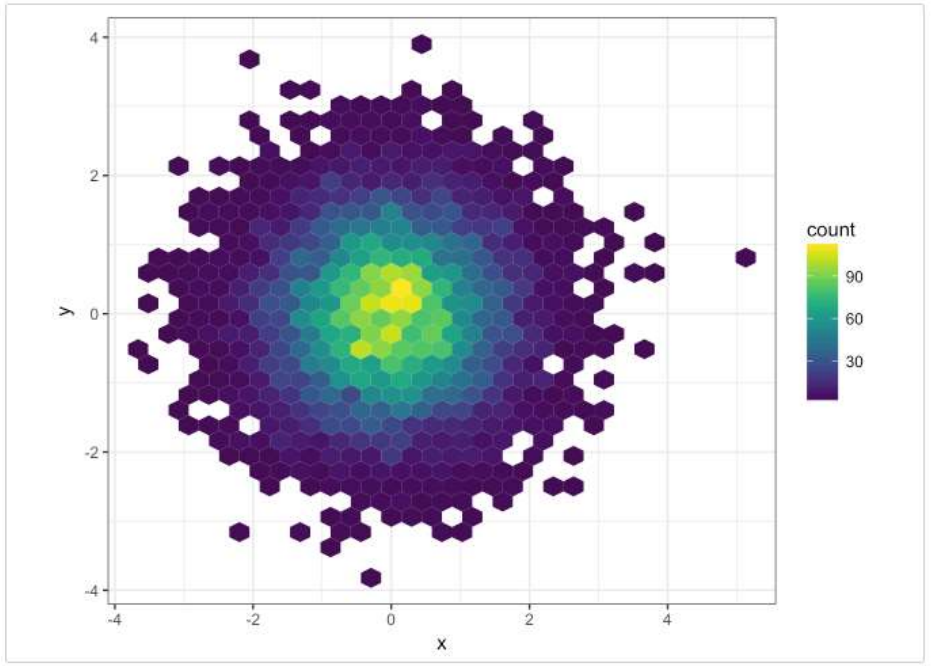
Image from: http://blog.visme.co/color-combinations/

Color palettes for pros…

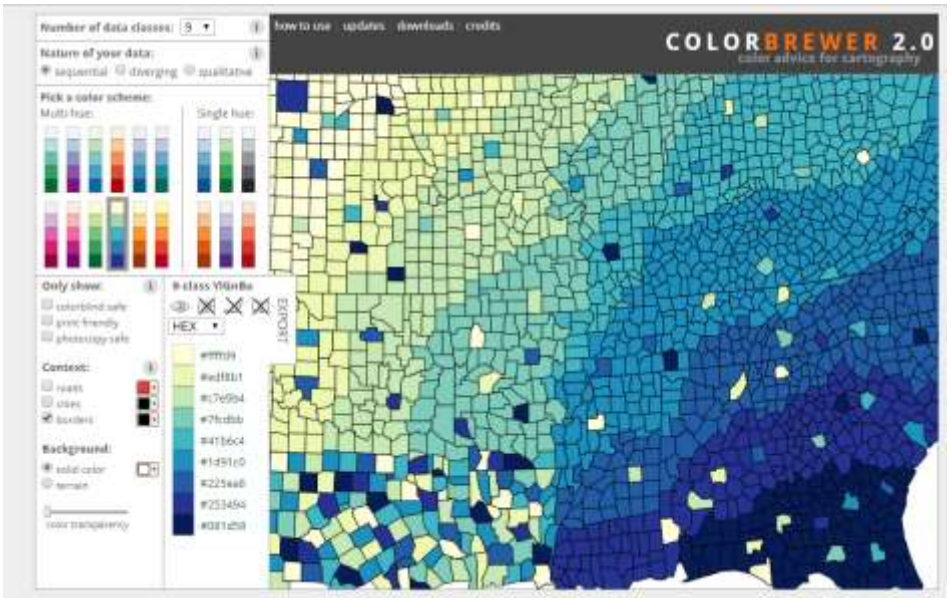
```
library(wesanderson)
```

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:

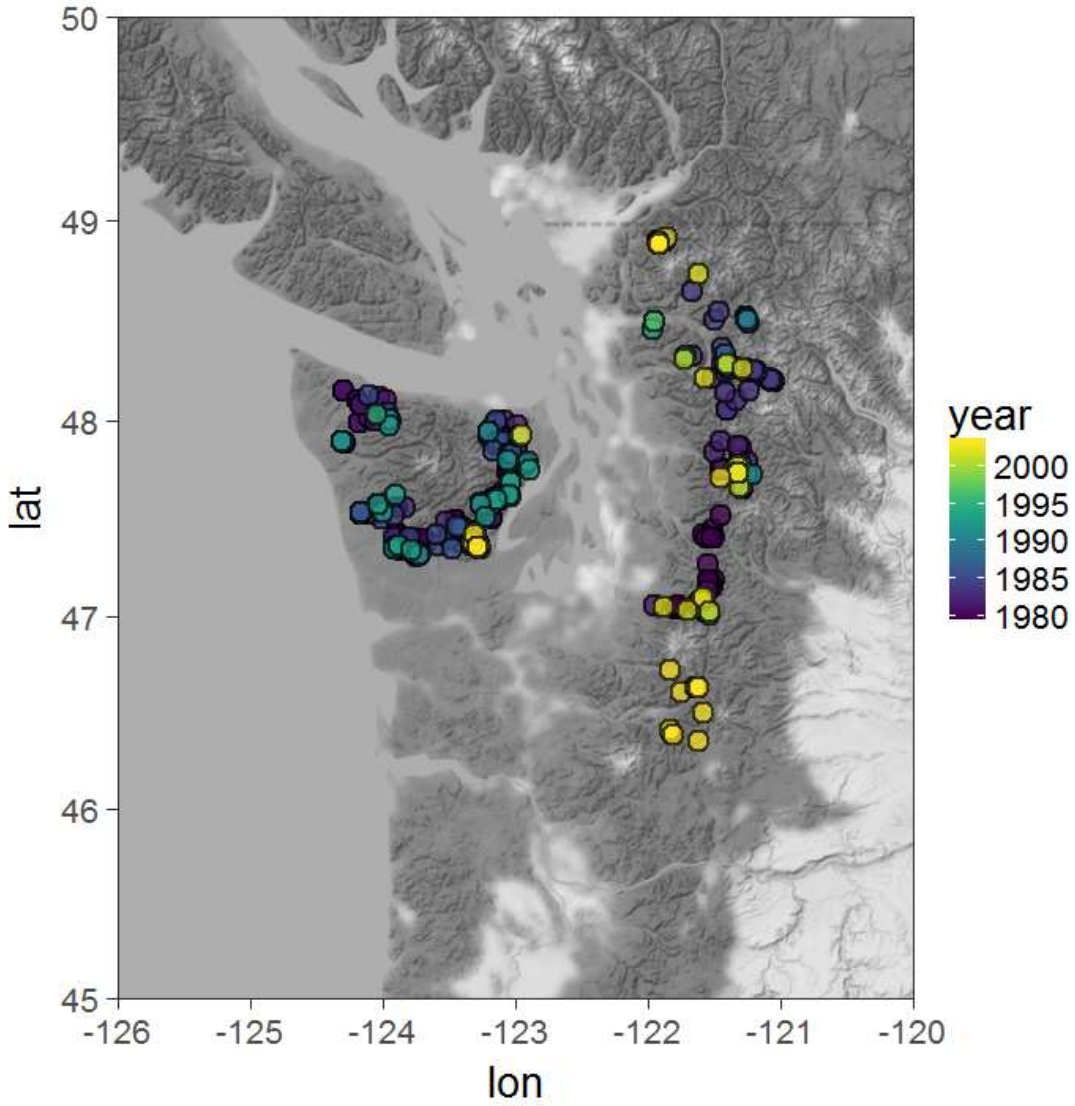



```
library(viridis)
```

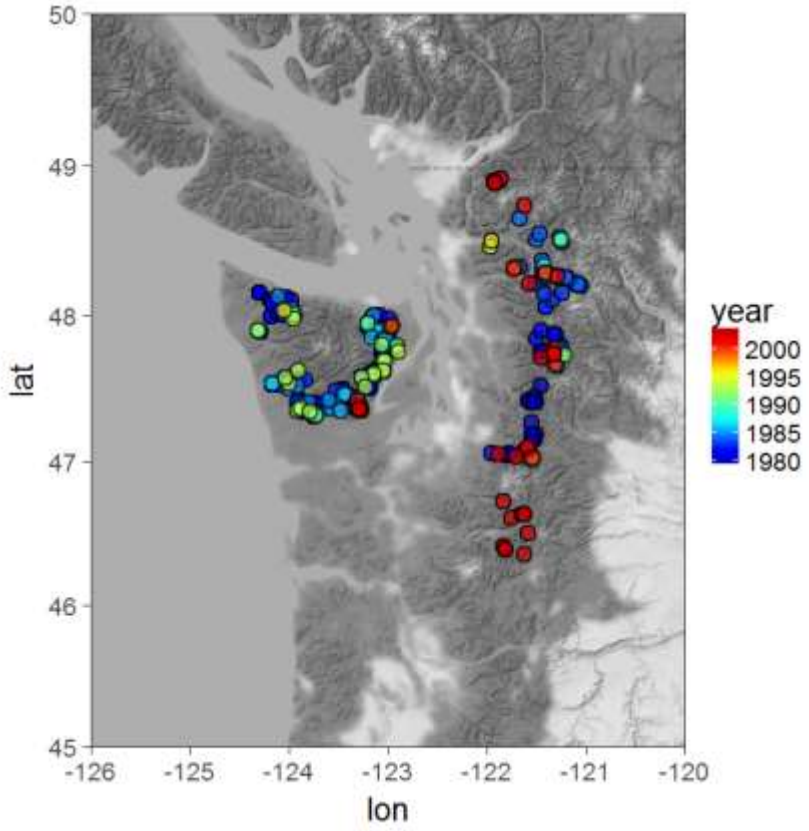


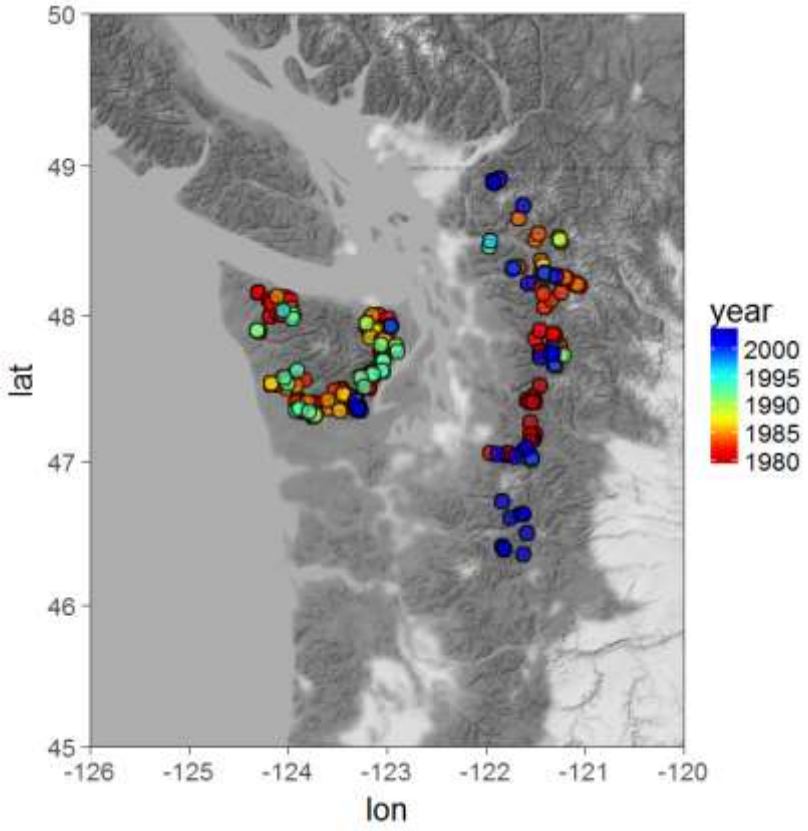

```
library(RColorBrewer)
```

```
### The continous blue colors make it hard to distinguish between years.. Lets customize!

### Using colors from the viridis package

newmap <- ptmap + scale_fill_viridis()
newmap
```

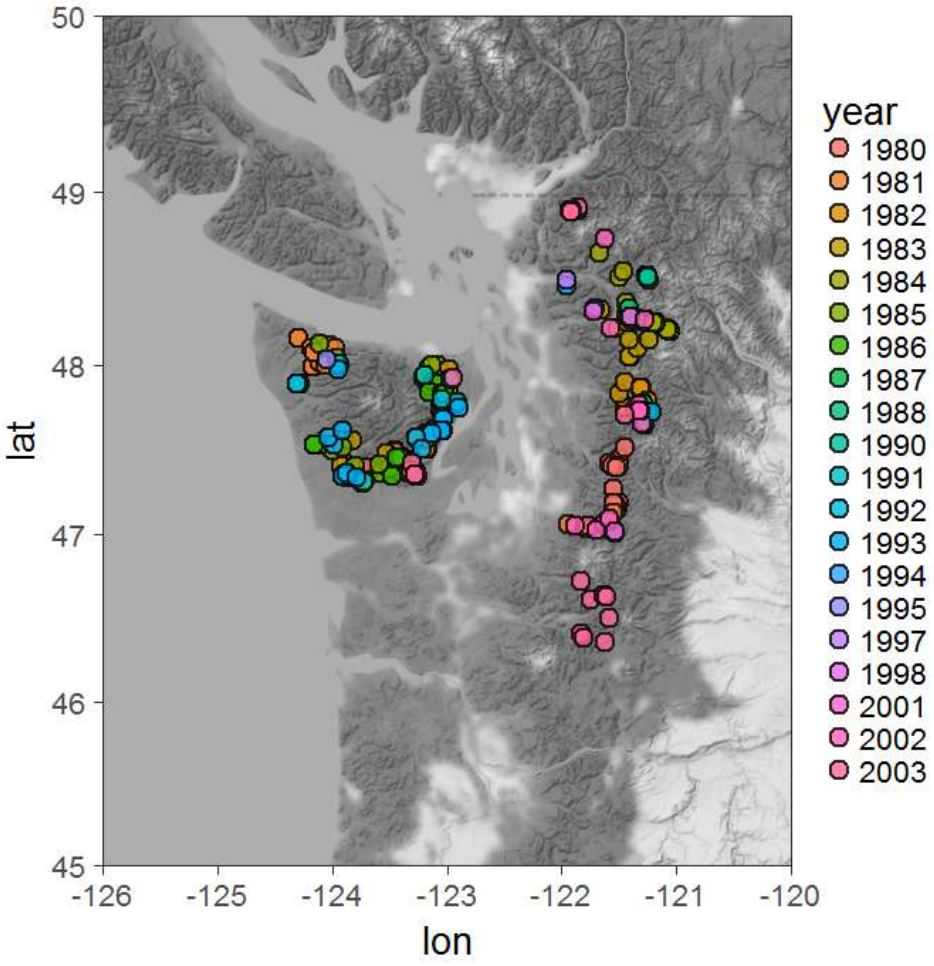


```
### Using colors from the colorRamps Package:

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
```



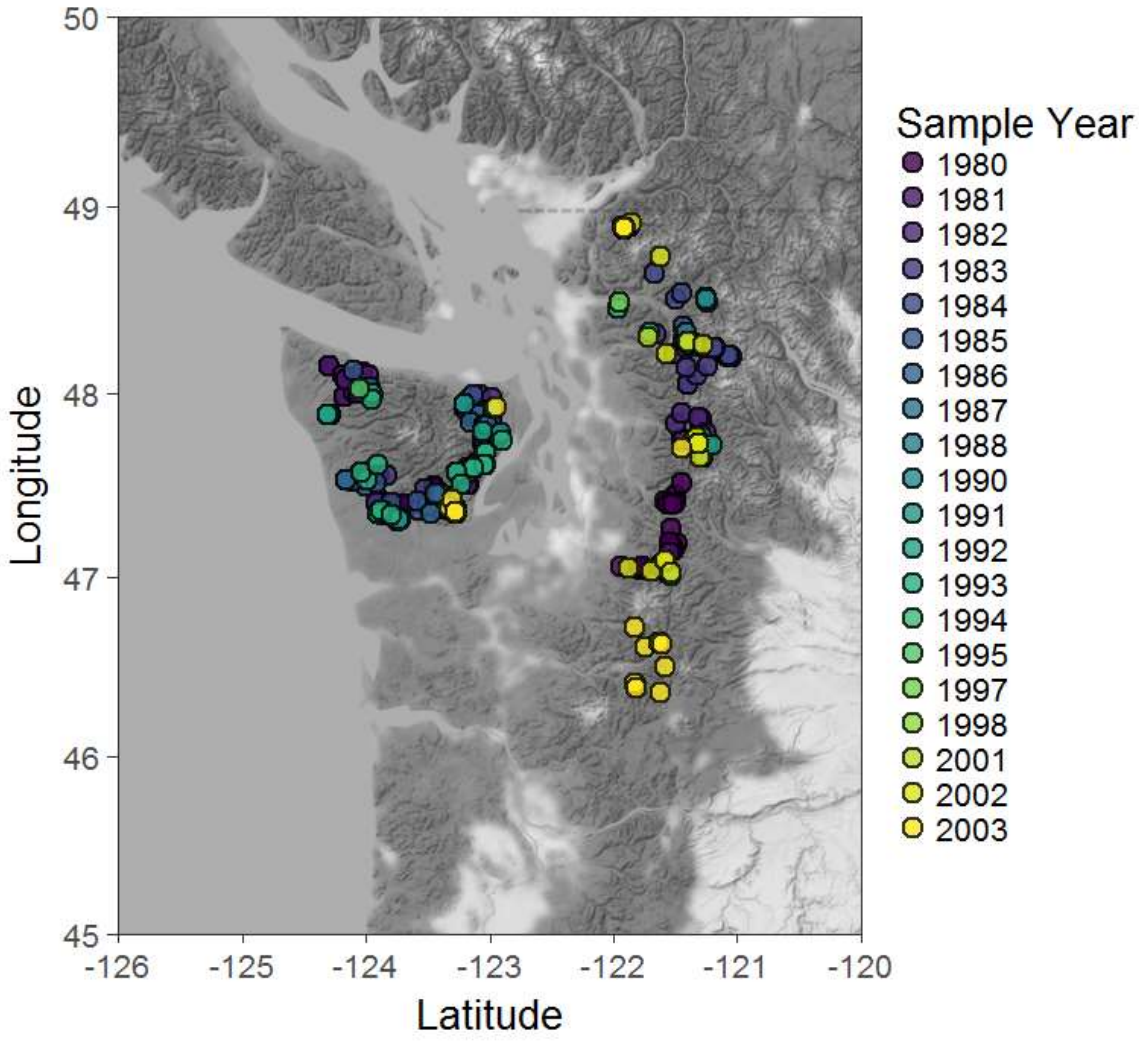


```
### We can also change year to a discrete variable and see if that helps:
## Designate year as a factor instead of number

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
```

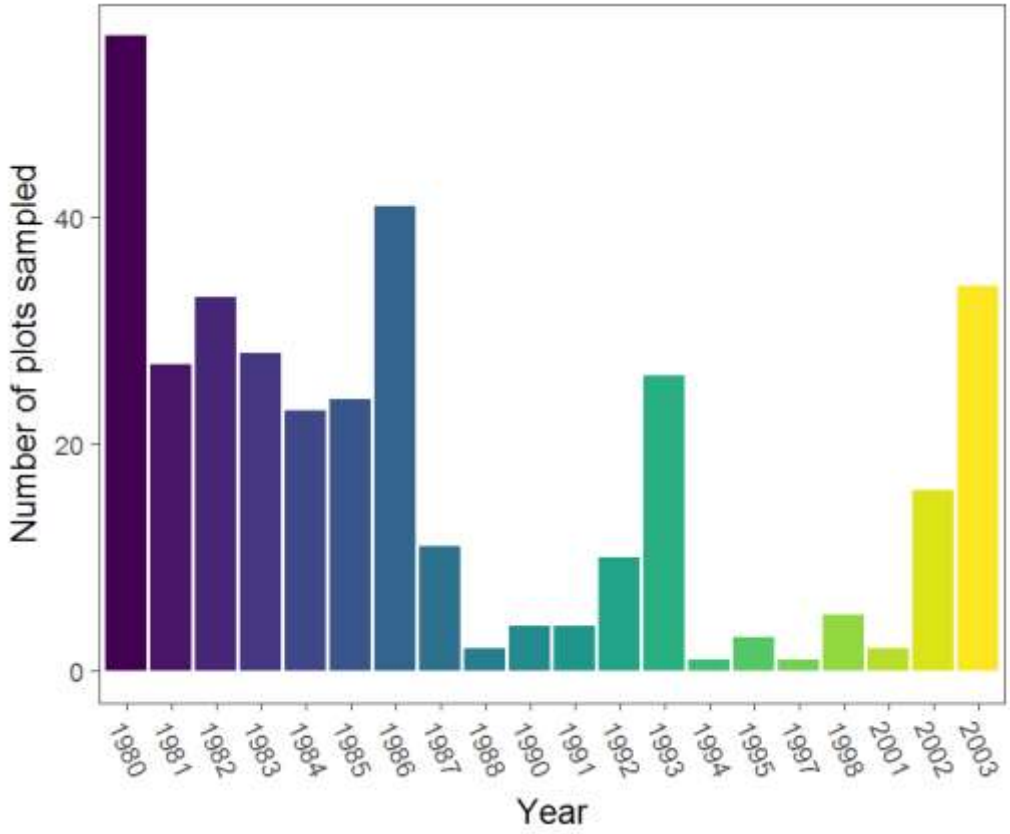


Meeehhhhhhh……

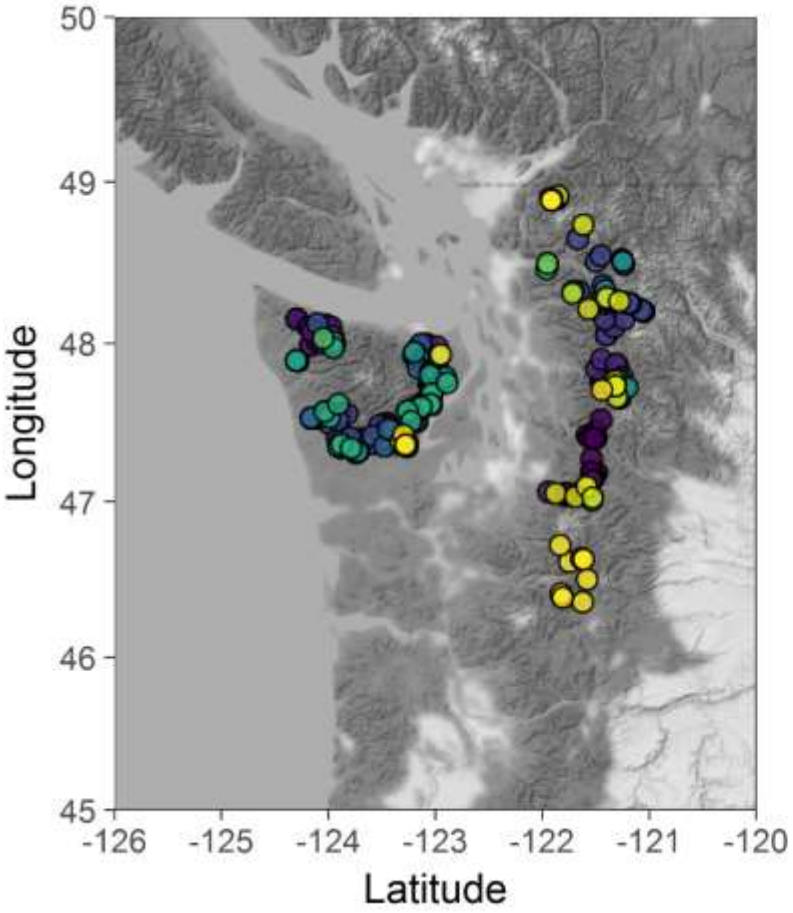
```
### That looks bad......lets customize the colors!!!!

newmap <- ptmap + scale_fill_viridis(discrete=TRUE)


###Now Lets add better labels:

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

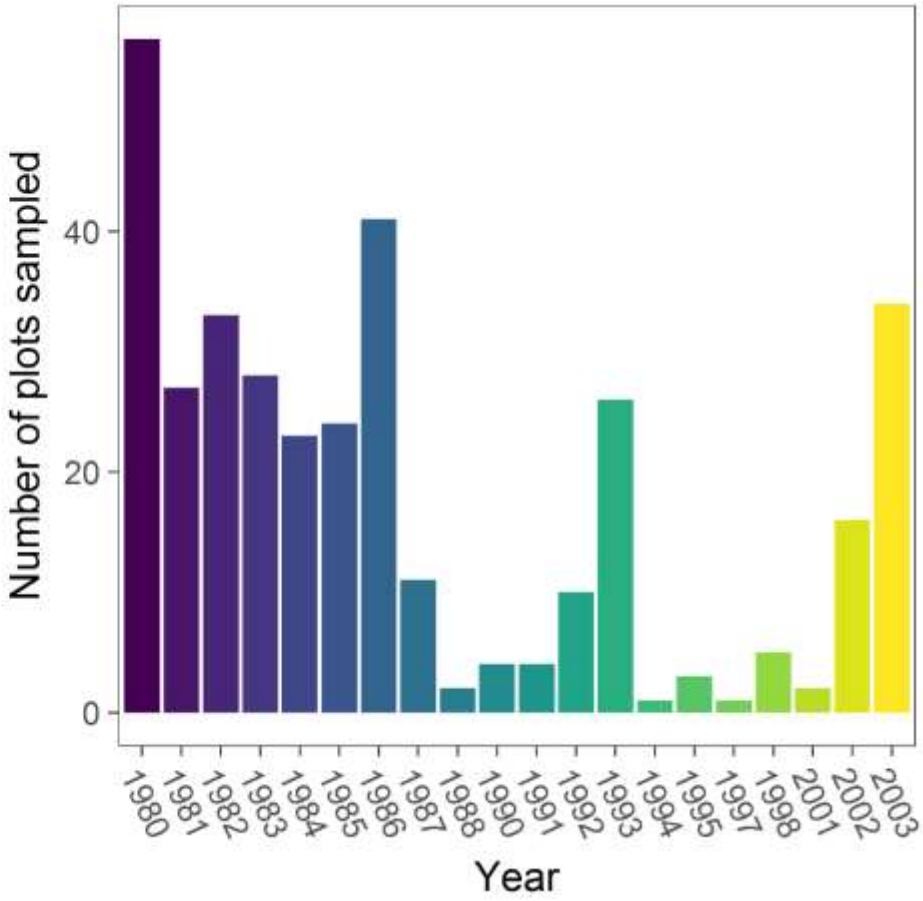


```r
### It's hard to get a clear picture of how many plots were sampled per year
### To make it more clear, we can also show a histogram using the same colors, that shows the number of sampled plots per year

## 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 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
```

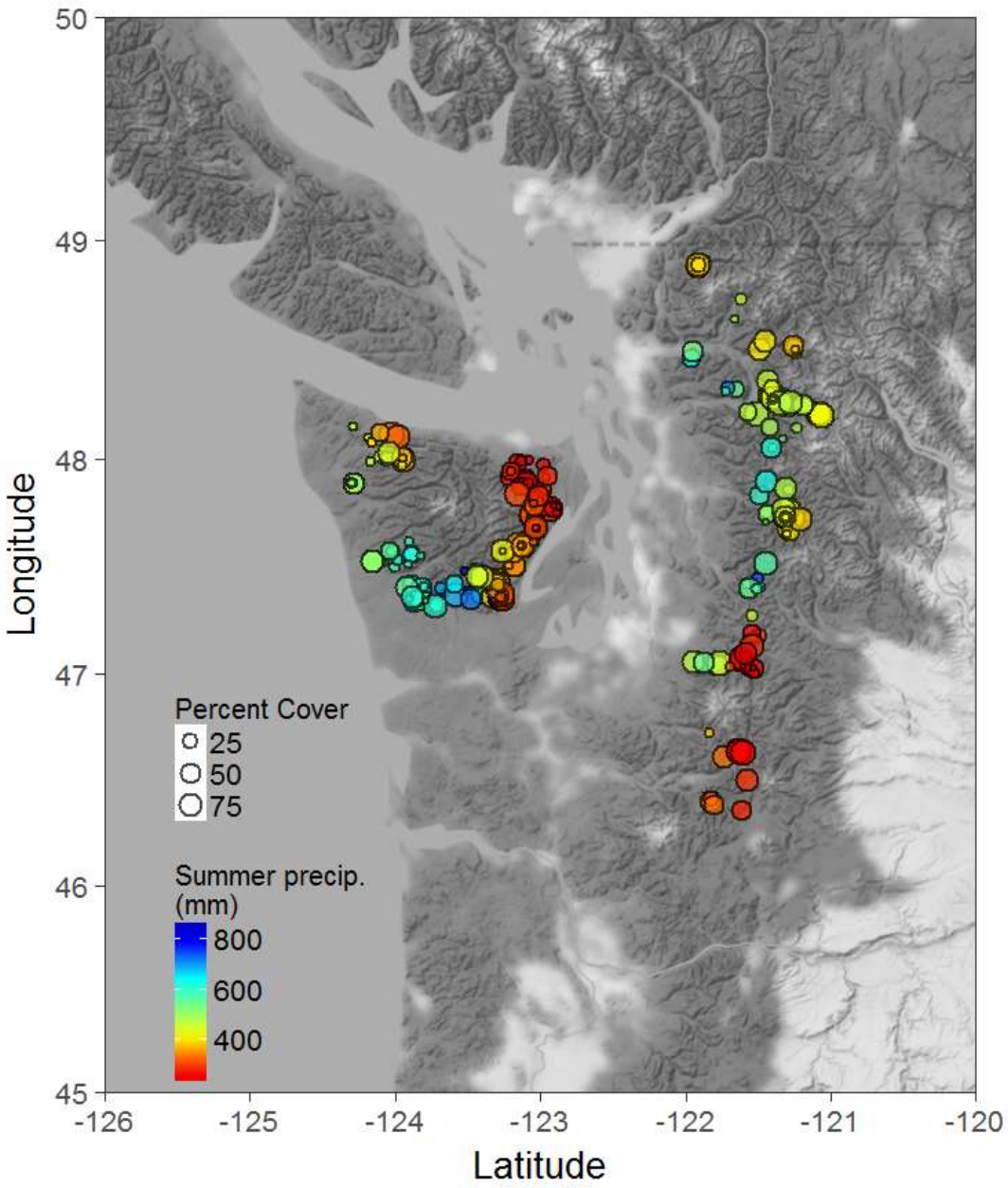


```
### And we can place our map and histogram next to each other using the gridExtra package
grid.arrange(scdmap,chisto, ncol=2)
### ANd now we really don't need the legend on the map, since the colors for each year are already shown in the bar graph
smap <- scdmap + scale_fill_viridis(discrete=TRUE, guide=FALSE)
## Voila ###
grid.arrange(smap,chisto, ncol=2)
```





```
################################### Other examples ###############################################

## 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

newmap <- ptmap  +  scale_fill_gradientn(colours= rev
newmap

## Add correct labels

scdmap <- newmap + labs(x="Latitude",y= "Longitude",
scdmap

### Change the position of legends

### Tweak the legend positions to suit our preference
### the coordinates for legend.position are x- and y-

thrdmap <- scdmap + theme(strip.text.x = element_blan
                strip.background = element_rect(colour
                legend.position=c(.19,.19))

thrdmap

### We can also remove the white background of the le

frthmap <- thrdmap + theme(legend.background = elemen

frthmap
```



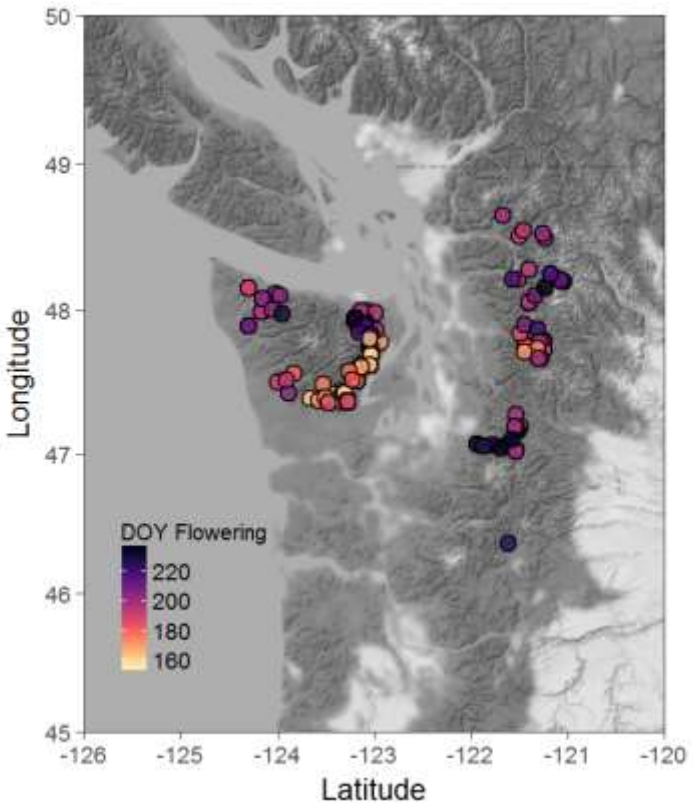
Is the timing of flowering of salal influenced by seasonal temperatures?

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

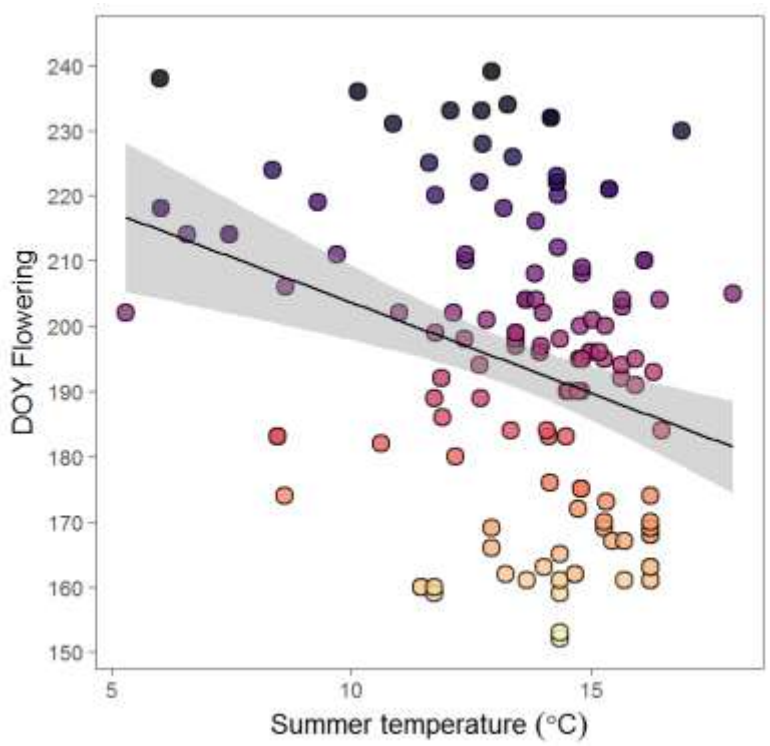
PSA *** Normally these would be **based on hypotheses** and accompanied by **statistical tests of significance******

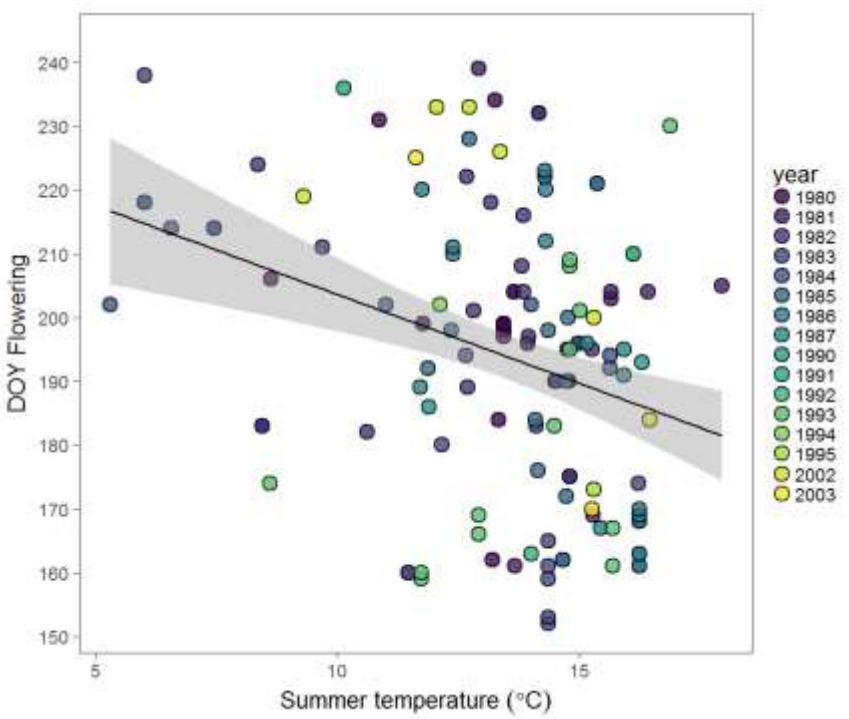
But that is a topic for a whole other powerpoint……

```
########### And for the last example - let's look at how the day of year when open Salal flowers were observed on a plot relates to Mean summer temperature

### 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
```

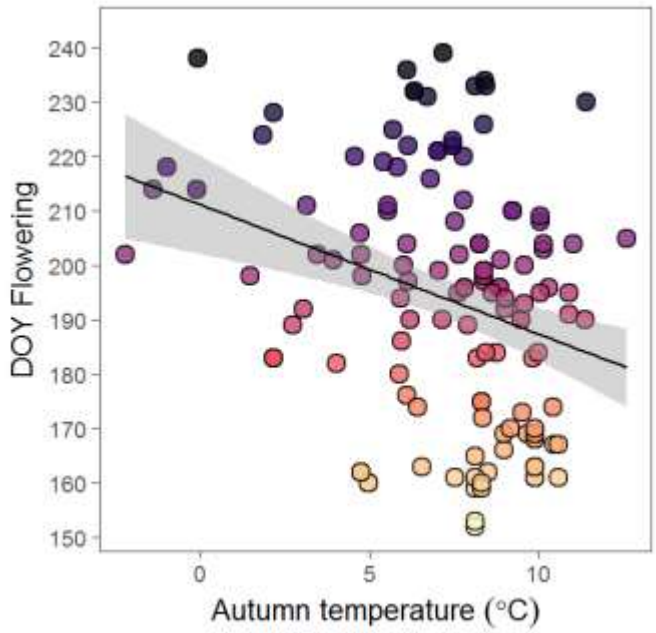


```r
## Now lets make a scatterplot to visualize if summer temperature is related to flowering date of salal

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

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 Augu
########### we can also use the breaks argument to dsignate 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, and change the x axis scale so numbers are visible

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

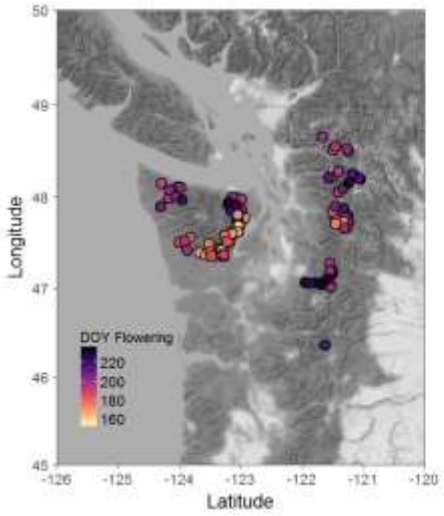


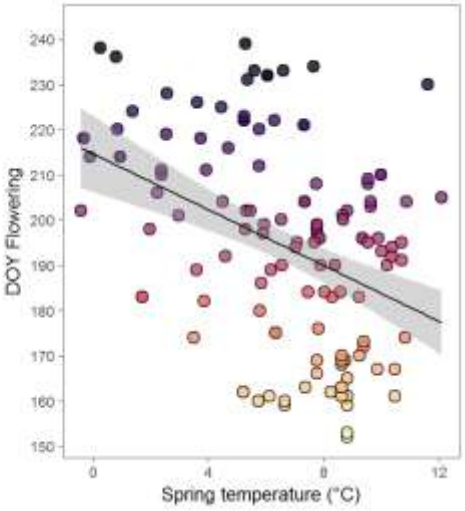


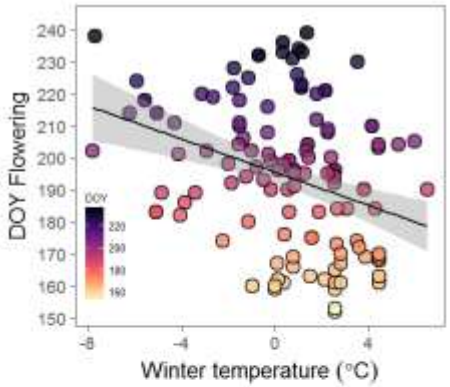
```
###########  We could also look at the relationships between other seasonal temperatures as well....


####### 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, a
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, a
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
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
```

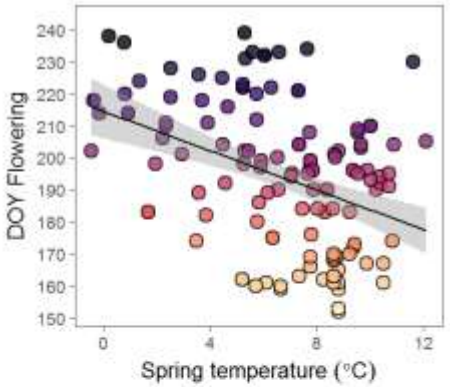


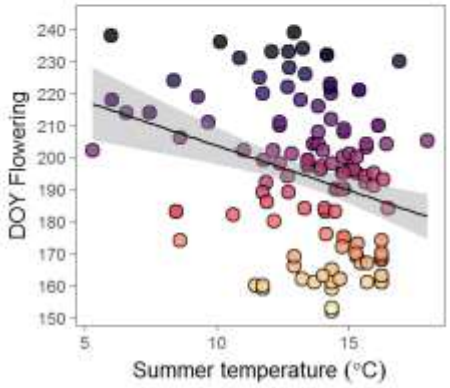
```
### And we can place our map and a graph together using the gridExtra package

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)
```

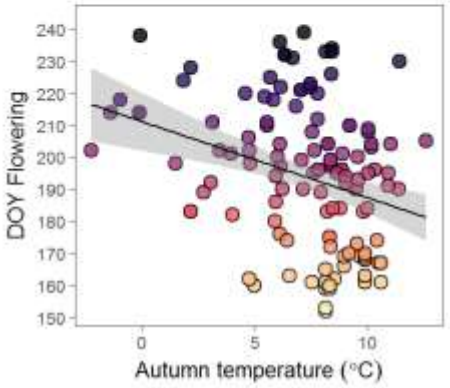






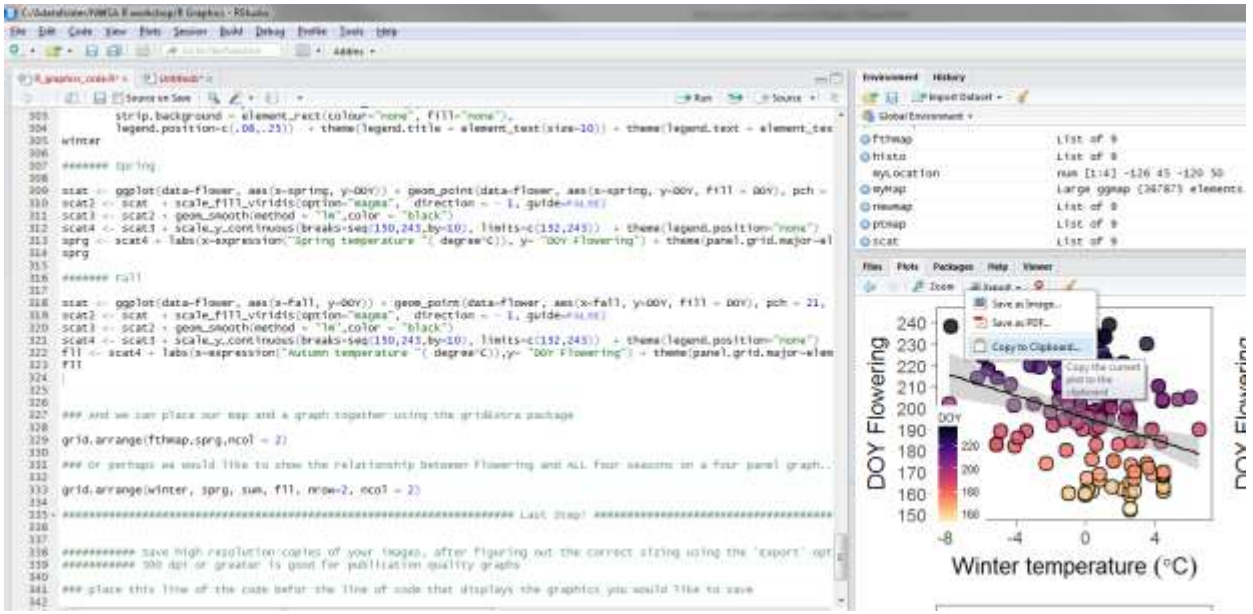




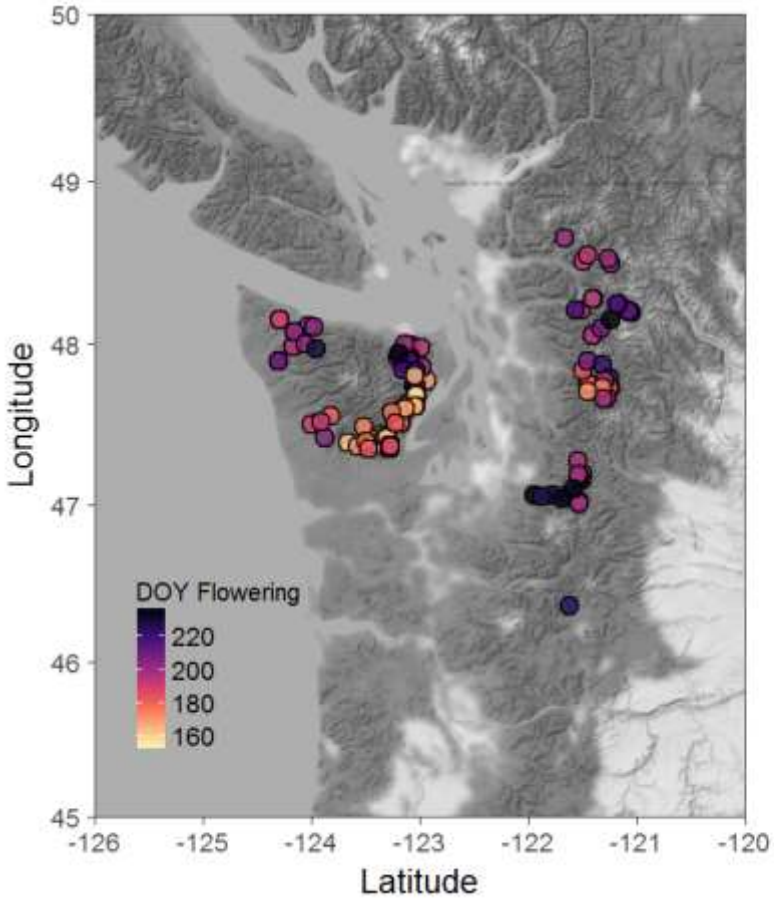


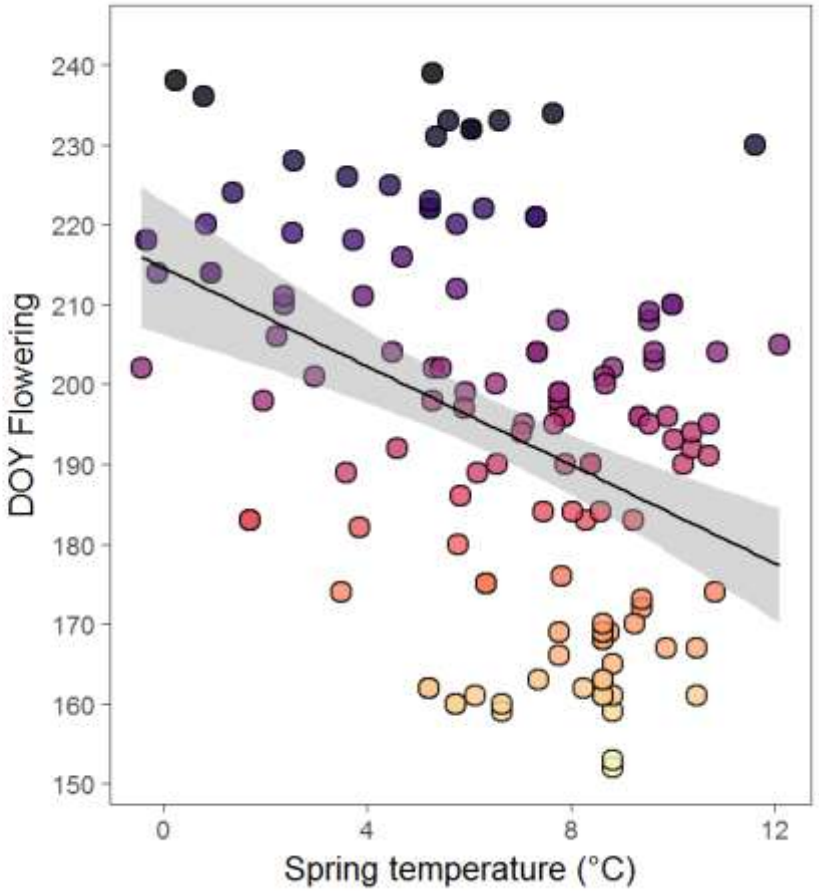
Exporting your images: best practices

- .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 images 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 very are buggy…….



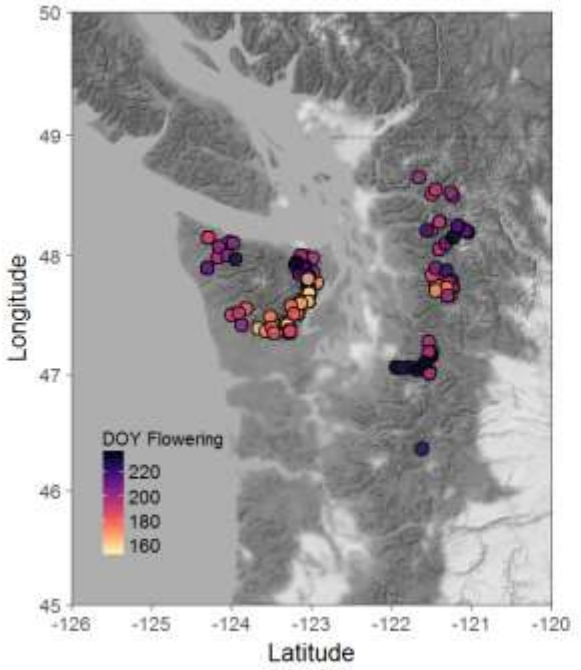
```
################################################################## Last Step! ##########################################################


########## 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 this line of the code befor 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) ##this will save a .tiff file to your working directory

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
```

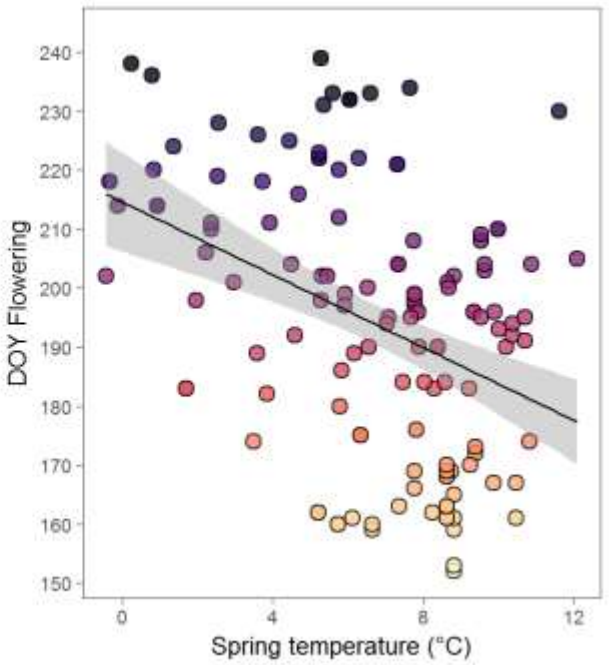




Take home messages:

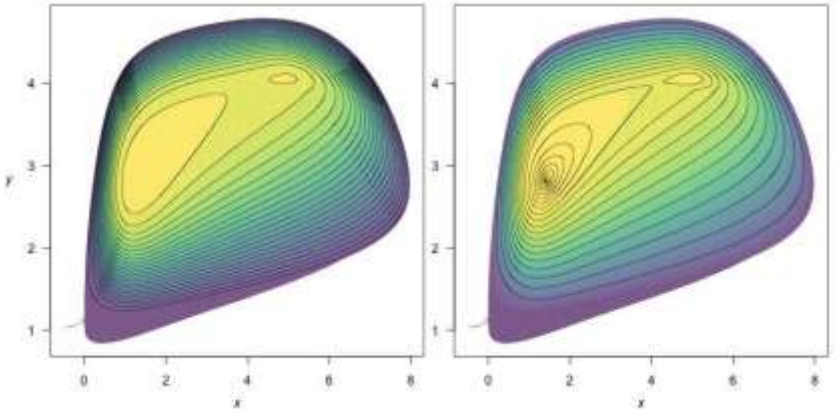
- Use color and shape to tell compelling stories with your graphs
- Use consistent colors across multiple figures to create continuity
- Less is more





All workshop materials are available on github: https://github.com/janetprevey/Graphics-in-R-tutorial

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 beautiful images and graphs of all types, with reproducible codes
- Visualizing raster data in R
  - https://cran.r-project.org/web/packages/raster/raster.pdf
  - http://geog.uoregon.edu/bartlein/courses/geog490/week04-raster.html
  - https://oscarperpinan.github.io/rastervis/
- Remember - Google is your friend!

Acknowledgements

- 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