R_workshop_Feb_2021

Alin Morariu

25/02/2021

Welcome!

This is a 2 part workshop focusing on key coding skills in R that will be useful in a research setting. Topics covered will include

- R objects
- Basic operations
- Introduction to knitr
- Introduction to the tidyverse
- Data wrangling (via dplyr)
- Data visualization (via ggplot2)
- Export our results to a PDF via an RMarkdown file

What is R?

R is a statistical program language that gives you a means of computing key values and plots to analyze your data. We will be using an IDE overlayed onto R called R Studio which provies a clean, intuitive environment for you to work in. R Studio has 4 main panes - Source file, Console, Environment, and Files.

R Objects

```
x0 <- 42
class(x0)
```

```
## [1] "numeric"
```

```
# can store multiple values in the same memory slot
v0 <- c(41, x0, 43)
v0
```

```
## [1] 41 42 43
```

```
# acccessing specific parts of the vector
v0[2]
```

```
## [1] 42
```

```r
v0[1:3]
```

```
## [1] 41 42 43
```

```r
# can store multiple vectors
df0 <- data.frame(v0, 2*v0, v0/2)
df0
```

```
##   v0 X2...v0 v0.2
## 1 41      82 20.5
## 2 42      84 21.0
## 3 43      86 21.5
```

R operations

```r
# add and substract
x0 + x0
```

```
## [1] 84
```

```r
x0-3
```

```
## [1] 39
```

```r
# mult + div
x0*4
```

```
## [1] 168
```

```r
x0/4
```

```
## [1] 10.5
```

```r
v0*4
```

```
## [1] 164 168 172
```

```r
# mult with dataframes
2*df0
```

```
##   v0 X2...v0 v0.2
## 1 82     164   41
## 2 84     168   42
## 3 86     172   43
```

```r
# define a 3x3 matrix
mt0 <- as.matrix(df0)
mt0
```

```
##      v0 X2...v0 v0.2
## [1,] 41      82 20.5
## [2,] 42      84 21.0
## [3,] 43      86 21.5
```

```r
t(mt0)
```

```
##         [,1] [,2] [,3]
## v0      41.0   42 43.0
## X2...v0 82.0   84 86.0
## v0.2    20.5   21 21.5
```

```r
x0^2
```

```
## [1] 1764
```

```r
# vector multiplication
v0 * v0 # element wise
```

```
## [1] 1681 1764 1849
```

```r
v0 %*% v0 # dot product
```

```
##      [,1]
## [1,] 5294
```

```r
v0 %*% t(v0) # vector product
```

```
##      [,1] [,2] [,3]
## [1,] 1681 1722 1763
## [2,] 1722 1764 1806
## [3,] 1763 1806 1849
```

```r
mt0 %*% mt0 # matrix product
```

```
##          v0 X2...v0    v0.2
## [1,] 6006.5   12013 3003.25
## [2,] 6153.0   12306 3076.50
## [3,] 6299.5   12599 3149.75
```

Functions

Sometimes we want to do a defined set of operations and this is where user functions come in. R contains a set of built in functions which come in handy, but we can also define our own functions.

```r
sum(1,2,3)
```

```
## [1] 6
```

```r
sum(v0)
```

```
## [1] 126
```

```r
my_sum <- function(vector){
     # sum the first and last entry of a vector
     x <- vector[1] + vector[length(vector)]

     return(x)
}

my_sum(v0)
```

```
## [1] 84
```

```r
my_sum(c())
```

```
## integer(0)
```

Packages

```r
install.packages('tidyverse')
install.packages('knitr')
```

```r
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------ tidyv
```

```
## v ggplot2 3.3.0     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts --------------------------------------------------------------------------- tidyverse_c
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(knitr)
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

Today we will focus on the usage of two key R packages - tidyverse and knitr.

Tidyverse

The tidyverse is a conglometration of smaller packages created with the goal of simplifying many of R's data science capabilities as well as taking advantage of modern computing power through highly efficient back end code (you do NOT need to worry about this and this is why having it in a package is so convenient). Within the tidyverse, we will focus on 2 main sub-packages. Namely, dplyr (a package used for data wrangling and cleaning), and ggplot2 (a packaged used for data visualization and creation of plots). Both of these packages have wonderful cheatsheets online which I strongly recommend using (I open these up every time I do anything in R).

- dplyr: https://rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf
- ggplot2: https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

Knitr

Knitr will be particularly useful to us since it will allow us to export our work from these .Rmd files into usable, and sometimes publishable, documents. The RMarkdown documents can be exported to HTML, PDF, or Word documents making them very useful when working on a project. The alternative is to write all of your code in an R script (a .R file), and work with the outputted objects individually (for example, you can save any of your plots as .jpeg or .png files which you can then slot into your figures in another document). I recommend using the RMarkdown setting due to it's ability to handle inline code via the code chunks you've been seeing so far.

Playing with Covid-19

Let's load in some data which we downloaded from the Johns Hopkins Covid-19 dashboard. The most recent files are available for download here.

```
# need to set where we get our data from -CHANGE THIS FOR YOUR COMPUTER
setwd("~/Documents/GitHub/R-Workshop-for-STEM/2021")

# many ways to import data
covid_global <- read_csv("time_series_covid19_confirmed_global.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   `Province/State` = col_character(),
##   `Country/Region` = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
covid_US <- read_csv("time_series_covid19_confirmed_US.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   iso2 = col_character(),
##   iso3 = col_character(),
##   Admin2 = col_character(),
##   Province_State = col_character(),
##   Country_Region = col_character(),
##   Combined_Key = col_character()
## )
## See spec(...) for full column specifications.
```

Now we need to view our data and decide on what to do with it!

```
head(covid_US)
```

```
## # A tibble: 6 x 323
##      UID iso2  iso3  code3  FIPS Admin2 Province_State Country_Region   Lat
##    <dbl> <chr> <chr> <dbl> <dbl> <chr>  <chr>          <chr>          <dbl>
## 1 8.40e7 US    USA     840  1001 Autau~ Alabama        US              32.5
## 2 8.40e7 US    USA     840  1003 Baldw~ Alabama        US              30.7
## 3 8.40e7 US    USA     840  1005 Barbo~ Alabama        US              31.9
## 4 8.40e7 US    USA     840  1007 Bibb   Alabama        US              33.0
## 5 8.40e7 US    USA     840  1009 Blount Alabama        US              34.0
## 6 8.40e7 US    USA     840  1011 Bullo~ Alabama        US              32.1
## # ... with 314 more variables: Long_ <dbl>, Combined_Key <chr>,
## #   `1/22/20` <dbl>, `1/23/20` <dbl>, `1/24/20` <dbl>, `1/25/20` <dbl>,
## #   `1/26/20` <dbl>, `1/27/20` <dbl>, `1/28/20` <dbl>, `1/29/20` <dbl>,
## #   `1/30/20` <dbl>, `1/31/20` <dbl>, `2/1/20` <dbl>, `2/2/20` <dbl>,
## #   `2/3/20` <dbl>, `2/4/20` <dbl>, `2/5/20` <dbl>, `2/6/20` <dbl>,
## #   `2/7/20` <dbl>, `2/8/20` <dbl>, `2/9/20` <dbl>, `2/10/20` <dbl>,
## #   `2/11/20` <dbl>, `2/12/20` <dbl>, `2/13/20` <dbl>, `2/14/20` <dbl>,
## #   `2/15/20` <dbl>, `2/16/20` <dbl>, `2/17/20` <dbl>, `2/18/20` <dbl>,
## #   `2/19/20` <dbl>, `2/20/20` <dbl>, `2/21/20` <dbl>, `2/22/20` <dbl>,
## #   `2/23/20` <dbl>, `2/24/20` <dbl>, `2/25/20` <dbl>, `2/26/20` <dbl>,
## #   `2/27/20` <dbl>, `2/28/20` <dbl>, `2/29/20` <dbl>, `3/1/20` <dbl>,
## #   `3/2/20` <dbl>, `3/3/20` <dbl>, `3/4/20` <dbl>, `3/5/20` <dbl>,
## #   `3/6/20` <dbl>, `3/7/20` <dbl>, `3/8/20` <dbl>, `3/9/20` <dbl>,
## #   `3/10/20` <dbl>, `3/11/20` <dbl>, `3/12/20` <dbl>, `3/13/20` <dbl>,
## #   `3/14/20` <dbl>, `3/15/20` <dbl>, `3/16/20` <dbl>, `3/17/20` <dbl>,
## #   `3/18/20` <dbl>, `3/19/20` <dbl>, `3/20/20` <dbl>, `3/21/20` <dbl>,
## #   `3/22/20` <dbl>, `3/23/20` <dbl>, `3/24/20` <dbl>, `3/25/20` <dbl>,
## #   `3/26/20` <dbl>, `3/27/20` <dbl>, `3/28/20` <dbl>, `3/29/20` <dbl>,
## #   `3/30/20` <dbl>, `3/31/20` <dbl>, `4/1/20` <dbl>, `4/2/20` <dbl>,
## #   `4/3/20` <dbl>, `4/4/20` <dbl>, `4/5/20` <dbl>, `4/6/20` <dbl>,
## #   `4/7/20` <dbl>, `4/8/20` <dbl>, `4/9/20` <dbl>, `4/10/20` <dbl>,
## #   `4/11/20` <dbl>, `4/12/20` <dbl>, `4/13/20` <dbl>, `4/14/20` <dbl>,
## #   `4/15/20` <dbl>, `4/16/20` <dbl>, `4/17/20` <dbl>, `4/18/20` <dbl>,
## #   `4/19/20` <dbl>, `4/20/20` <dbl>, `4/21/20` <dbl>, `4/22/20` <dbl>,
```

```
## #   `4/23/20` <dbl>, `4/24/20` <dbl>, `4/25/20` <dbl>, `4/26/20` <dbl>,
## #   `4/27/20` <dbl>, `4/28/20` <dbl>, ...
```

The data contains information about the location and for each location, a time series (data indexed by a time/date) for the number of confirmed Covid-19 cases between January 1st, 2020 and November 28th, 2020. Before we do anything more, we always want to convert the data into a *tidy* format. Tidy data has a unique variable in each column and one observation per row. Our data frame has many measurements for the number of cases across the rows and must therefore be transformed into tidy form. To do so we use the gather function. It is incredibly powerful because it lets us map each column into a row with the use of key-value pairings. The idea is that we want each row to contain the country, province, city, date, *and* the number of confirmed cases. Having data in that format will allow us to group it together based on the different levels and aggregate as needed. The code is shown below

```
infected_data_tidy <- covid_US %>%
     gather(date_stamp, # this is the new column which will take values from covid_US columns
            num_infect, # this is the new value
            '1/22/20':'11/28/20' # the columns we're getting values from
            ) %>%
     rename(Province = `Province_State`, Country = `Country_Region`, City = `Admin2`) %>%
     select(Country, Province, date_stamp, num_infect) %>%
     group_by(Province)
```

We also want to change the dates to be of the data type instead of the strings they currently are. The lubridate package (part of the tidyverse package) helps us do that with the various date formats (month-day-year in this case). Since we only want to change a specific column, we can use the $ operator which subsets the dataframe (column-wise).

```
# change strings to dates
infected_data_tidy$date_stamp <- lubridate::mdy(infected_data_tidy$date_stamp)
```

Now that the data is formatted we can start exploring the data a bit more. This exploratory stage will often provide many answers and insights to various questions we might have about a dataset. A simple first step is to find the total Covid-19 cases in the United States for each day.

```
# we want to aggregate data on a country scale for summary purposes
infected_summary <- infected_data_tidy %>%
     group_by(date_stamp) %>%                          # each locale has a date stamp and we want to s
     summarise(total = sum(num_infect))                # aggregating function

tail(infected_summary) # a variant of head
```

```
## # A tibble: 6 x 2
##   date_stamp    total
##   <date>        <dbl>
## 1 2020-11-23 12418228
## 2 2020-11-24 12591163
## 3 2020-11-25 12772653
## 4 2020-11-26 12883264
## 5 2020-11-27 13088821
## 6 2020-11-28 13244417
```

Someone mentioned that we should find the date with the highest number of cases. Since the data is now sorted by the days, this can be a very simple tax of just using the built-in R function max. There are a

few ways to do this, but one good way is to use the base R function summary which provides a 6 number summary (minimun, first quartile, median, third quartile, maximum)

```
max(infected_data_tidy$num_infect) # we use the subset operation $ to select the one column we want thes
```

```
## [1] 390891
```

```
print('-------------------------------------------------------------------------')
```

```
## [1] "-------------------------------------------------------------------------"
```

```
summary(infected_data_tidy$num_infect)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0      51    1120     425  390891
```

```
print('-------------------------------------------------------------------------')
```

```
## [1] "-------------------------------------------------------------------------"
```

```
# get max from summary (note that the output is a list)
summary(infected_data_tidy$num_infect)[6]
```

```
##   Max.
## 390891
```

```
print('-------------------------------------------------------------------------')
```

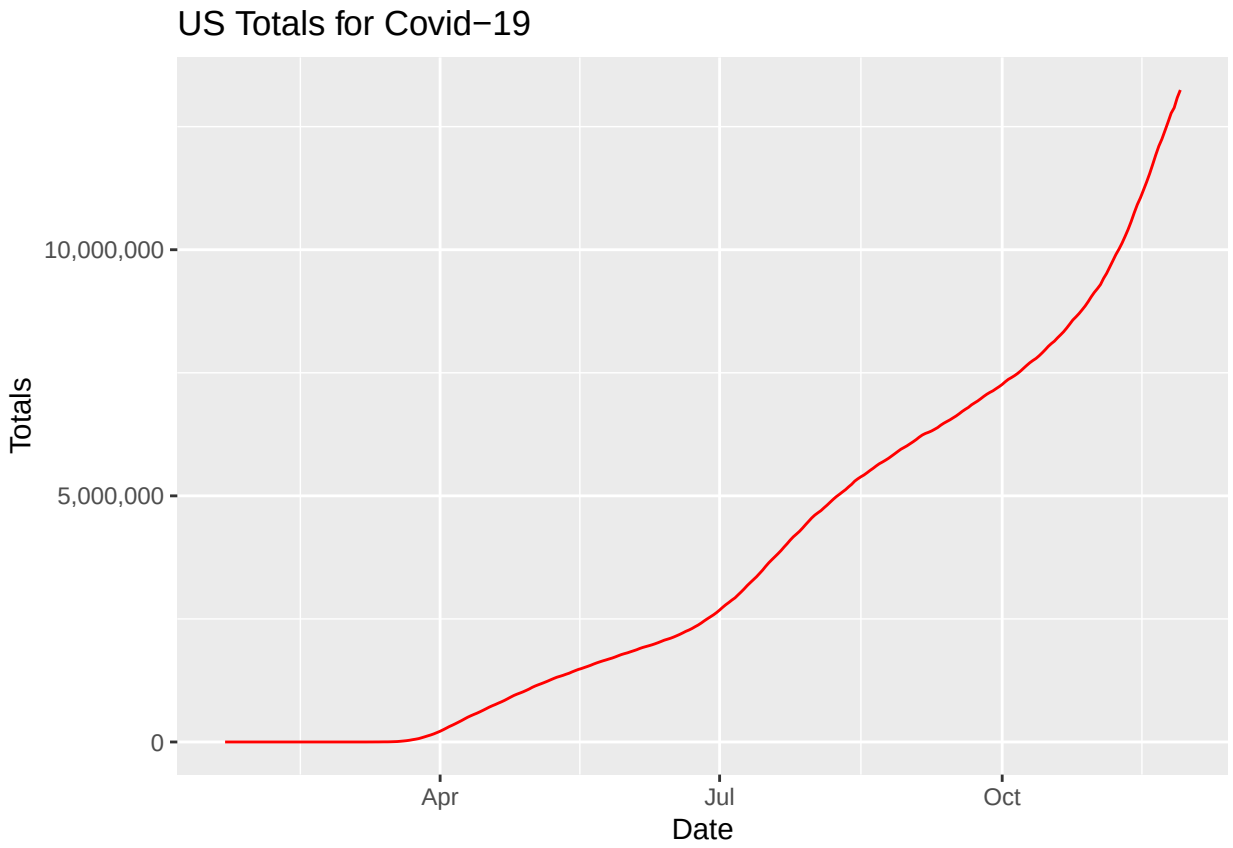
```
## [1] "-------------------------------------------------------------------------"
```

```
# check that it is same as above
summary(infected_data_tidy$num_infect)[6] == max(infected_data_tidy$num_infect)
```

```
## Max.
## TRUE
```

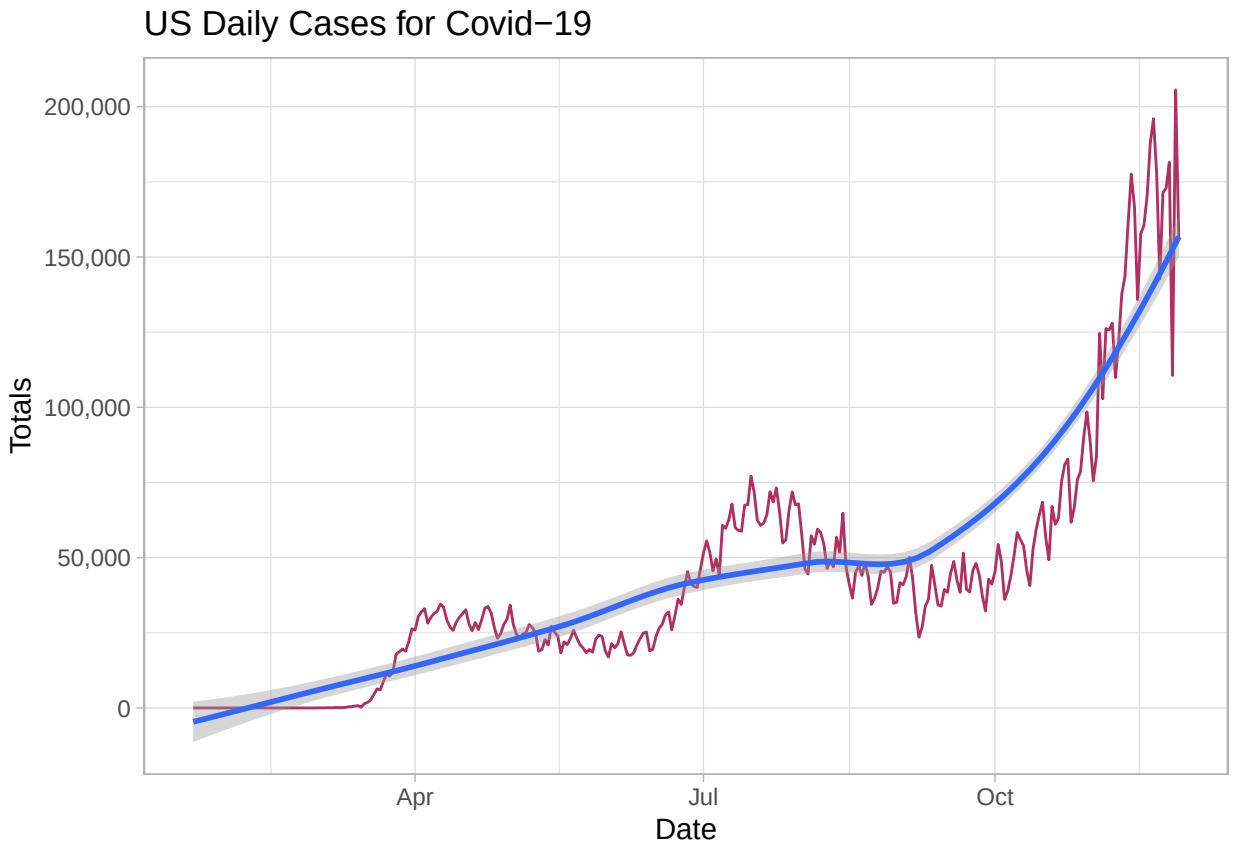
Now for some plots! We begin with an obvious one and that is the total number of infected people. In the graph below we see the

```
infected_summary %>% ggplot(aes(x = date_stamp, y = total)) +
        geom_line(colour = 'red')+
        scale_y_continuous(labels = comma) +
        labs(title = "US Totals for Covid-19",
             x = "Date",
             y = "Totals") +
        theme_grey()
```



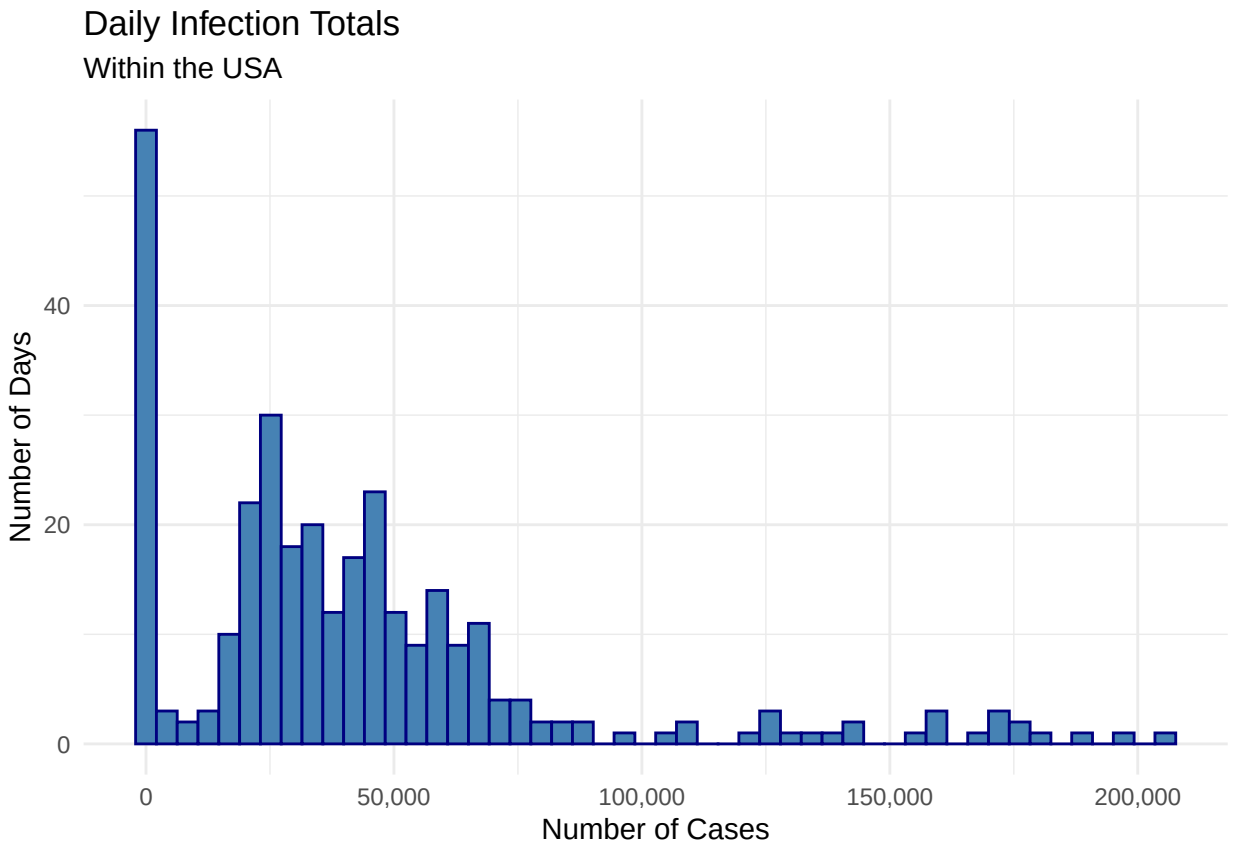

```
infected_summary %>% ggplot(aes(x = date_stamp, y = c(0,diff(total)))) +
        geom_line(colour = 'maroon') +
        geom_smooth(method = 'loess', se = TRUE) +
        scale_y_continuous(labels = comma) +
        labs(title = "US Daily Cases for Covid-19",
             x = "Date",
             y = "Totals") +
        theme_light()
```

```
## `geom_smooth()` using formula 'y ~ x'
```

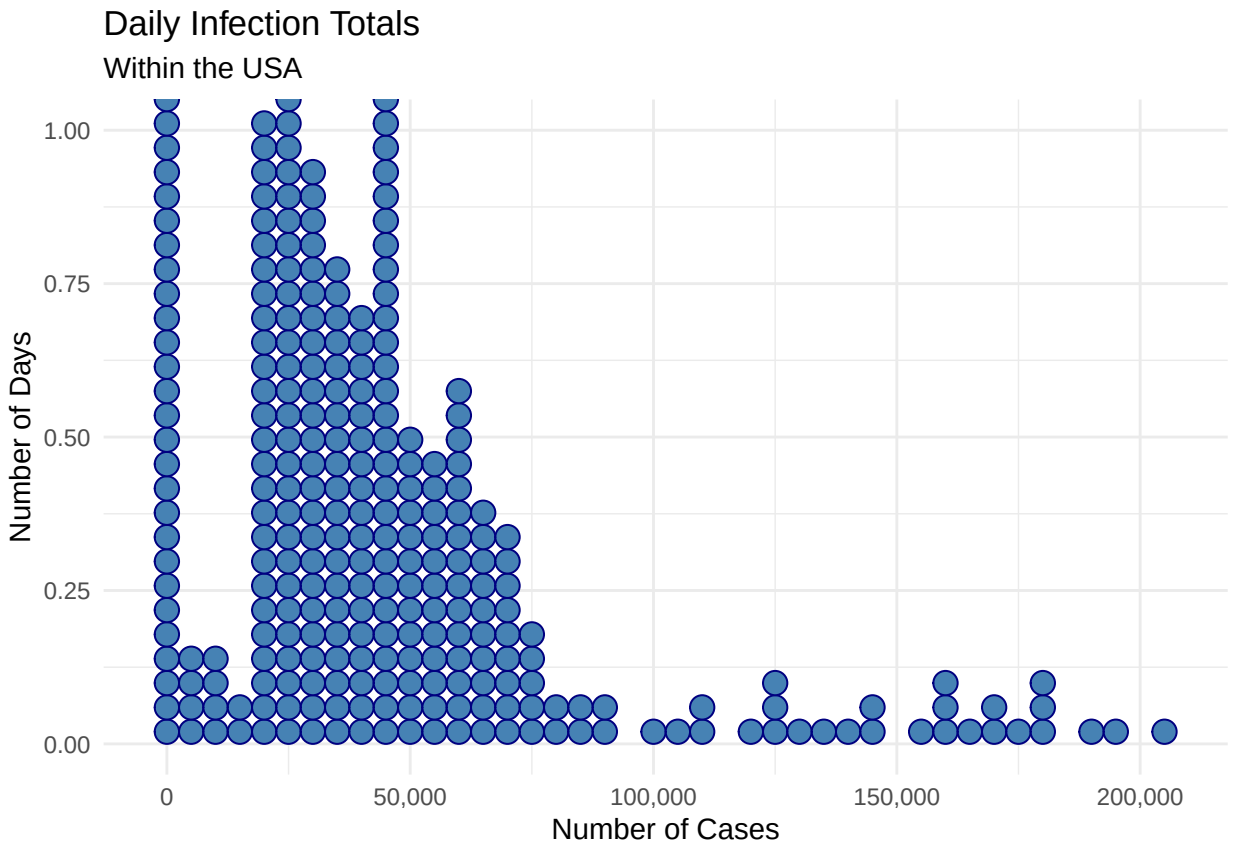


A common question is finding the distribution of a single variable. In our case, the distribution of the number of daily infections could be a very useful piece of information for modeling spread. Let's examine this distribution with a histogram

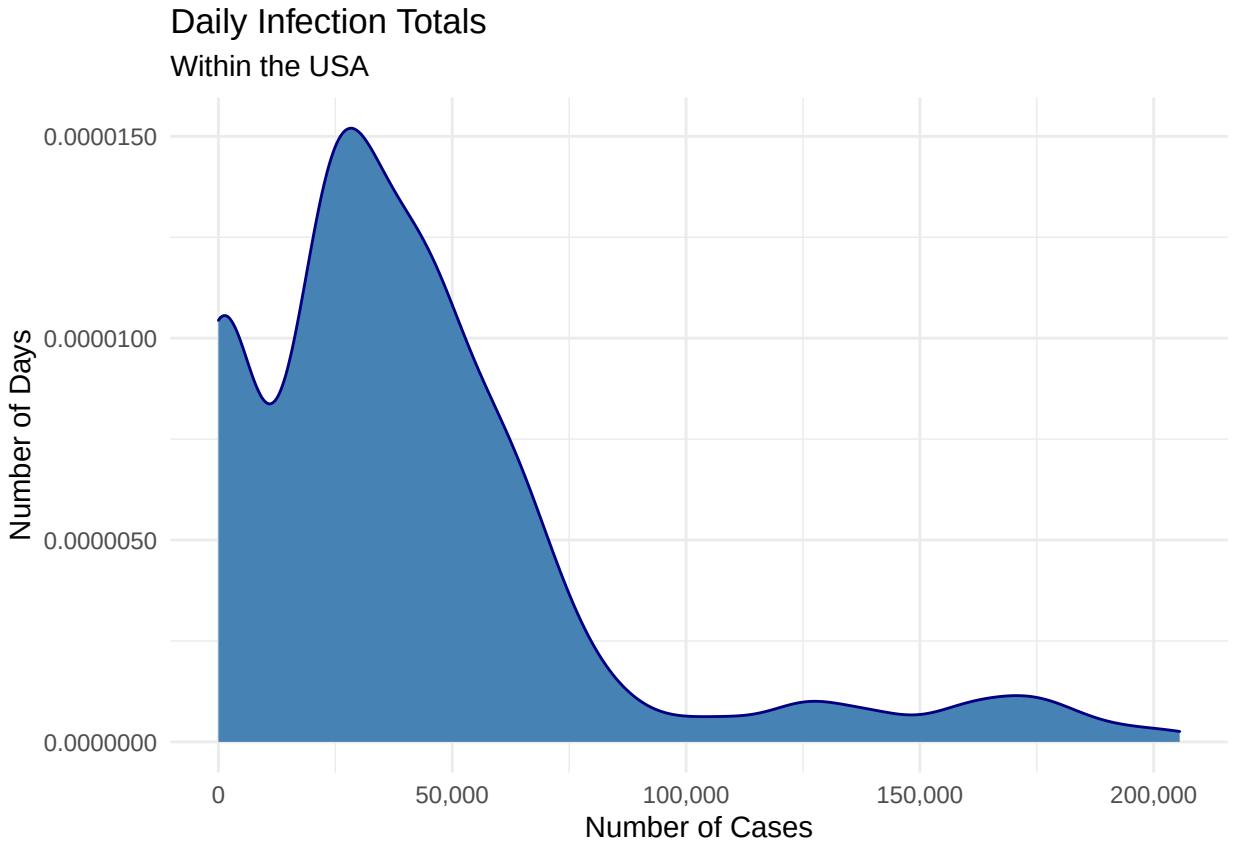
```
# option 1: Histogram
infected_summary %>% ggplot(aes(c(0,diff(total)))) +
        geom_histogram(bins = 50, fill = 'steelblue', colour = 'navy')+
        labs(title = 'Daily Infection Totals',
             subtitle = 'Within the USA',
             x = 'Number of Cases',
             y = 'Number of Days') +
        scale_x_continuous(labels = comma)+
        theme_minimal()
```




```
# option 2: dot plot - doesn't work well IN THIS CASE but shown for completeness
infected_summary %>% ggplot(aes(c(0,diff(total)))) +
        geom_dotplot(method = 'histodot', binwidth = 5000, fill = 'steelblue', colour = 'navy')+
        labs(title = 'Daily Infection Totals',
             subtitle = 'Within the USA',
             x = 'Number of Cases',
             y = 'Number of Days') +
        scale_x_continuous(labels = comma)+
        theme_minimal()
```



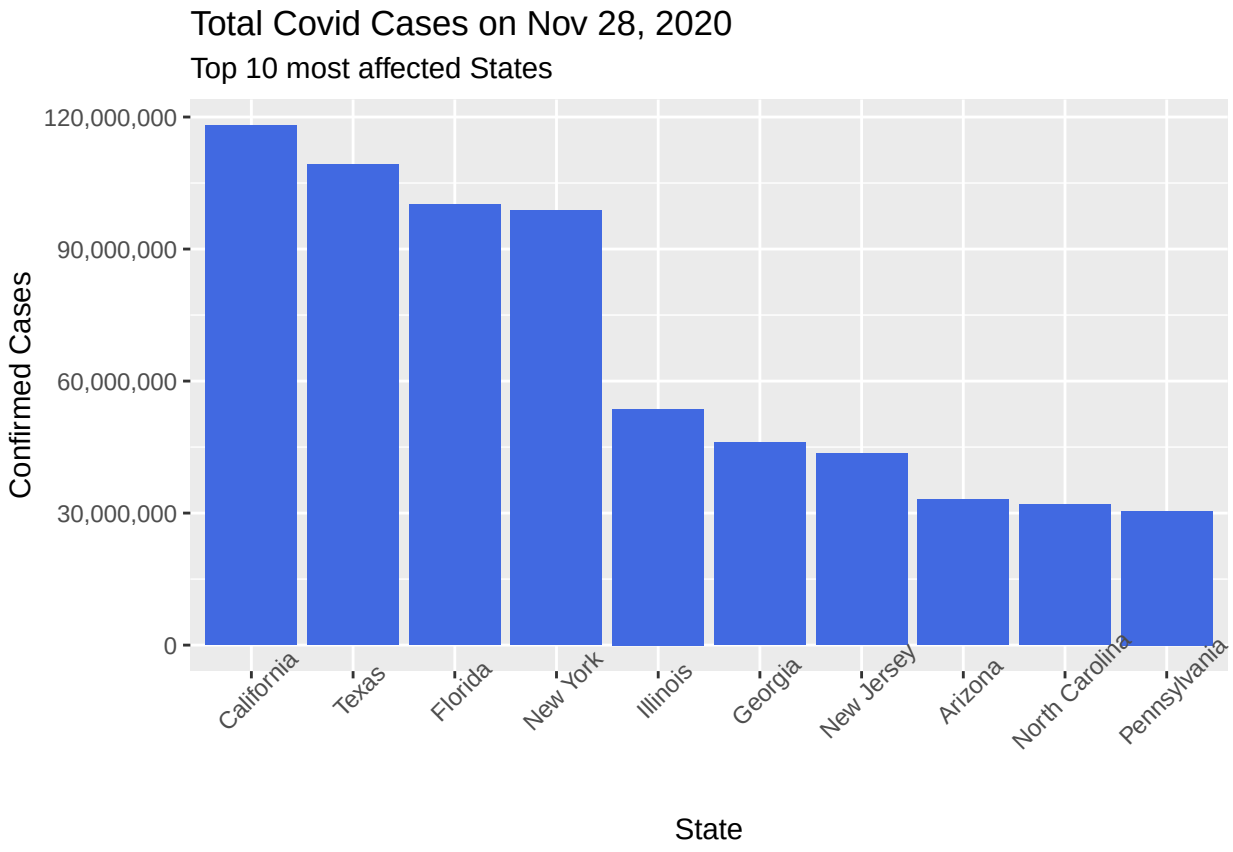

```
# option 3:
infected_summary %>% ggplot(aes(c(0,diff(total)))) +
        geom_density(kernel = 'gaussian', fill = 'steelblue', colour = 'navy')+
        labs(title = 'Daily Infection Totals',
             subtitle = 'Within the USA',
             x = 'Number of Cases',
             y = 'Number of Days') +
        scale_x_continuous(labels = comma) +
        scale_y_continuous(labels = comma) +
        theme_minimal()
```



Since our data has state level data it would be interesting to see how the different states have been handling the pandemic. Let's try to create some plots to compare the most and least affected states. A simple way to do this is going to be with bar graphs since we can display totals

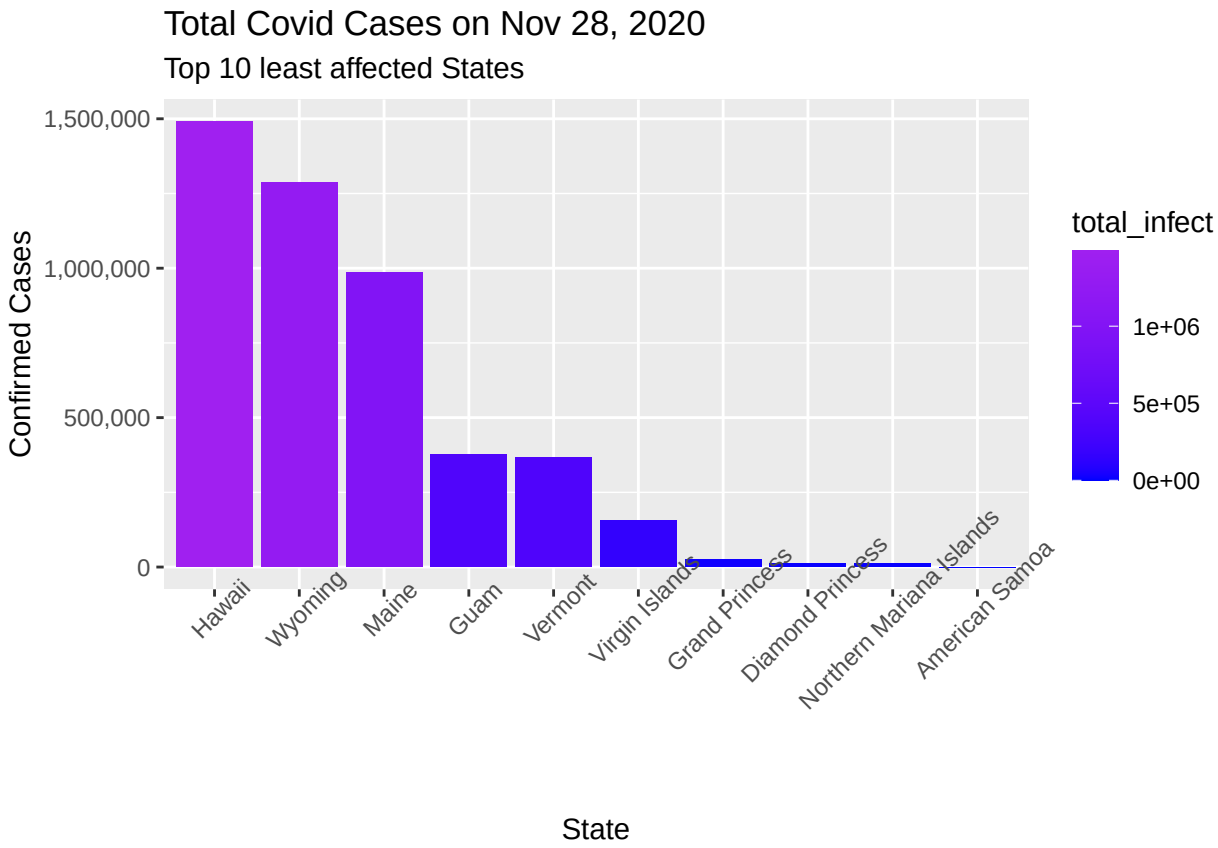
```
infected_data_tidy %>%
        group_by(Province) %>%
        summarise(total_infect = sum(num_infect)) %>%
        arrange(desc(total_infect)) %>%
        top_n(10) %>%
        ggplot(aes(x = reorder(Province,-total_infect), y = total_infect)) +
        geom_bar(stat = 'identity', fill = 'royalblue') +
        labs(title = 'Total Covid Cases on Nov 28, 2020',
             subtitle = 'Top 10 most affected States',
             x = 'State',
             y = 'Confirmed Cases') +
        scale_y_continuous(labels = comma) +
        theme(axis.text.x = element_text(angle = 45))
```

```
## Selecting by total_infect
```



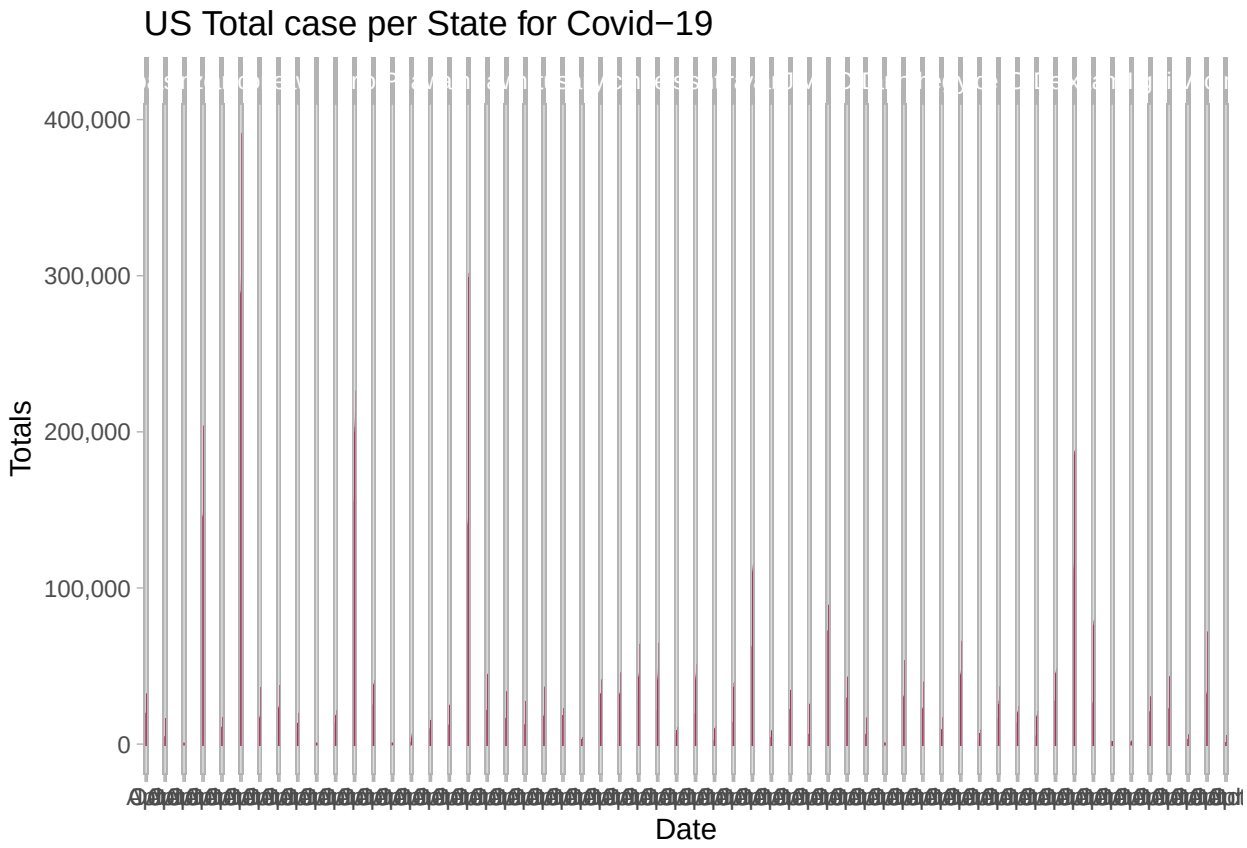

```
infected_data_tidy %>%
        group_by(Province) %>%
        summarise(total_infect = sum(num_infect)) %>%
        arrange(desc(total_infect)) %>%
        top_n(-10) %>%
        ggplot(aes(x = reorder(Province, -total_infect), y = total_infect)) +
        geom_bar(stat = 'identity', aes(fill = total_infect)) +
        labs(title ='Total Covid Cases on Nov 28, 2020',
             subtitle = 'Top 10 least affected States',
             x = 'State',
             y = 'Confirmed Cases') +
        scale_y_continuous(labels = comma) +
        scale_fill_gradient(low = "blue", high = "purple") +
        theme(axis.text.x = element_text(angle = 45))
```

```
## Selecting by total_infect
```

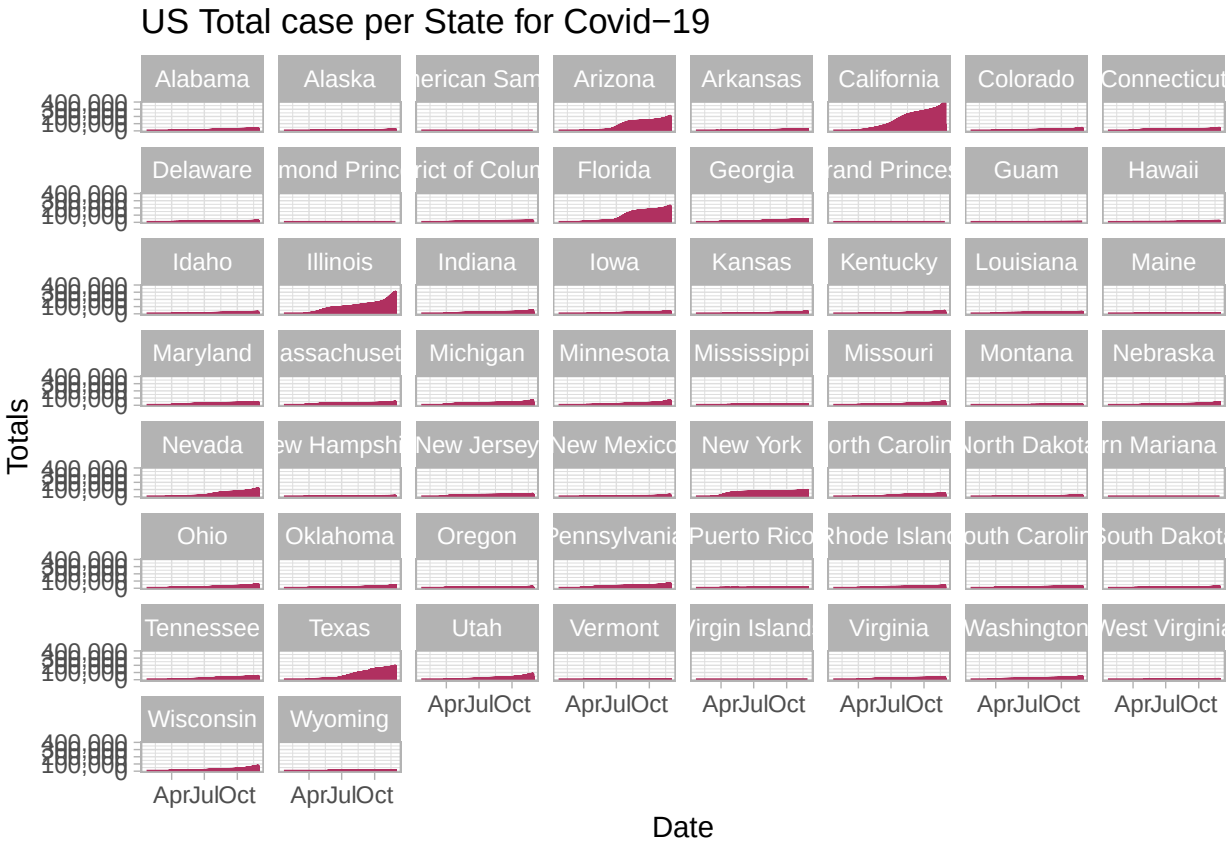


Another way to compare this is with faceting. As per the ggplot2 cheatsheet: Facets divide a plot into subplots based on the values of one or more discrete variables.

```
# option 1: side by side
infected_data_tidy %>% ggplot(aes(x = date_stamp, num_infect)) +
        geom_line(colour = 'maroon') +
        facet_grid(~Province) +
        scale_y_continuous(labels = comma) +
        labs(title = "US Total case per State for Covid-19",
             x = "Date",
             y = "Totals") +
        theme_light()
```




```
# option 2: wraping
infected_data_tidy %>% ggplot(aes(date_stamp, num_infect)) +
        geom_line(colour = 'maroon') +
        facet_wrap(~Province) +
        scale_y_continuous(labels = comma) +
        labs(title = "US Total case per State for Covid-19",
             x = "Date",
             y = "Totals") +
        theme_light()
```


US Total case per State for Covid−19
Totals
Date
Alabama
Alaska
erican Sam
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
mond Princ
rict of Colu
Florida
Georgia
rand Princes
Guam
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
assachuset
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
ew Hampshi
New Jersey
New Mexico
New York
orth Carolin
North Dakota
rn Mariana
Ohio
Oklahoma
Oregon
Pennsylvania
Puerto Rico
Rhode Island
outh Carolin
South Dakot
Tennessee
Texas
Utah
Vermont
Virgin Island
Virginia
Washington
West Virginia
Wisconsin
Wyoming
Apr
Jul
Oct