ASRM195_Group5_Final

Jess Krummick (jk54), Sabrina Altynbayeva (saltyn2),
Anna Schulz (aschulz4), Naomi Logan (naomiml2)

12/19/2020

Contents

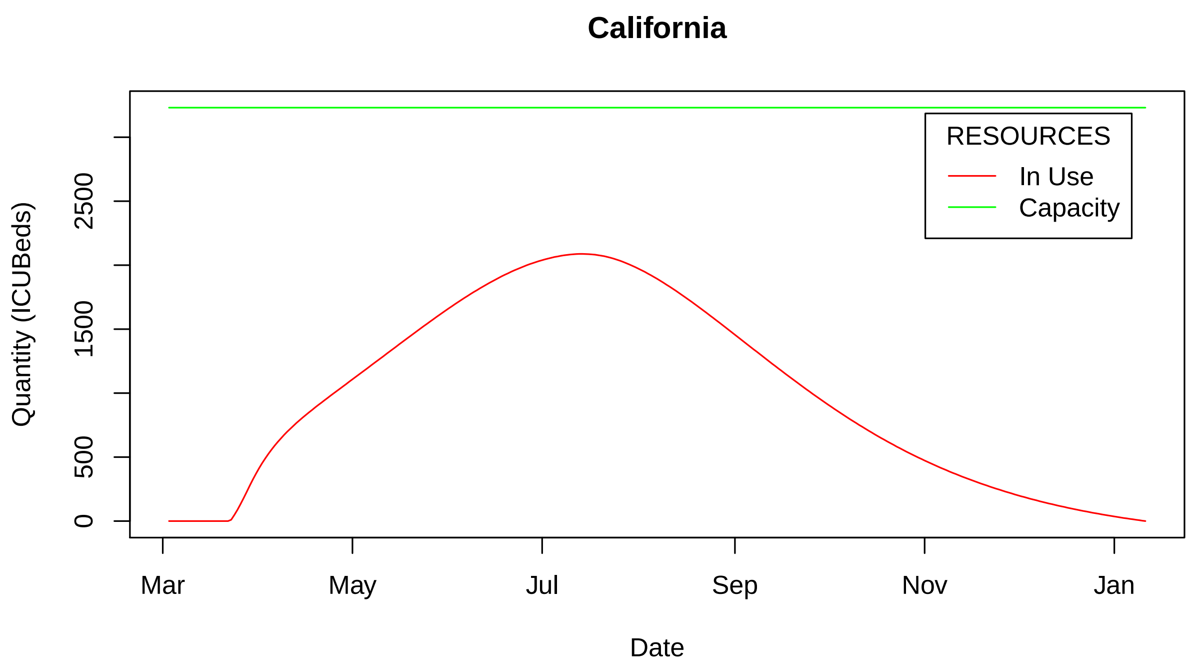
1 Importing Data

```r
covid_data_1013 <- read.csv(
  "~/Downloads/Datasets and Project Details-20201214 (1)/covid_data_1013.csv")
covid_data_1013 <- covid_data_1013[,1:16]
covid_data_1013 <- covid_data_1013[,-c(8:15)]
covid_data_1014 <- read.csv(
  "~/Downloads/Datasets and Project Details-20201214 (1)/covid_data_1014.csv")
covid_data_1014 <- covid_data_1014[,1:16]
covid_data_1014 <- covid_data_1014[,-c(8:15)]
```

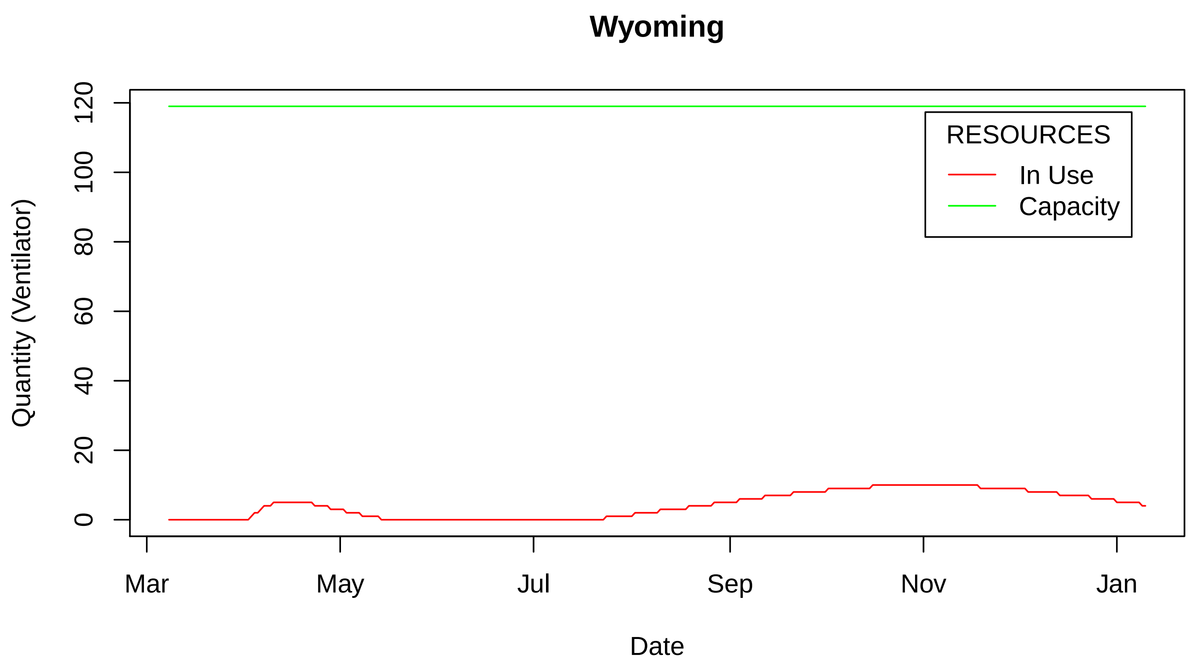
2 Function 1: plotResourceAdequacy

2.1 Code

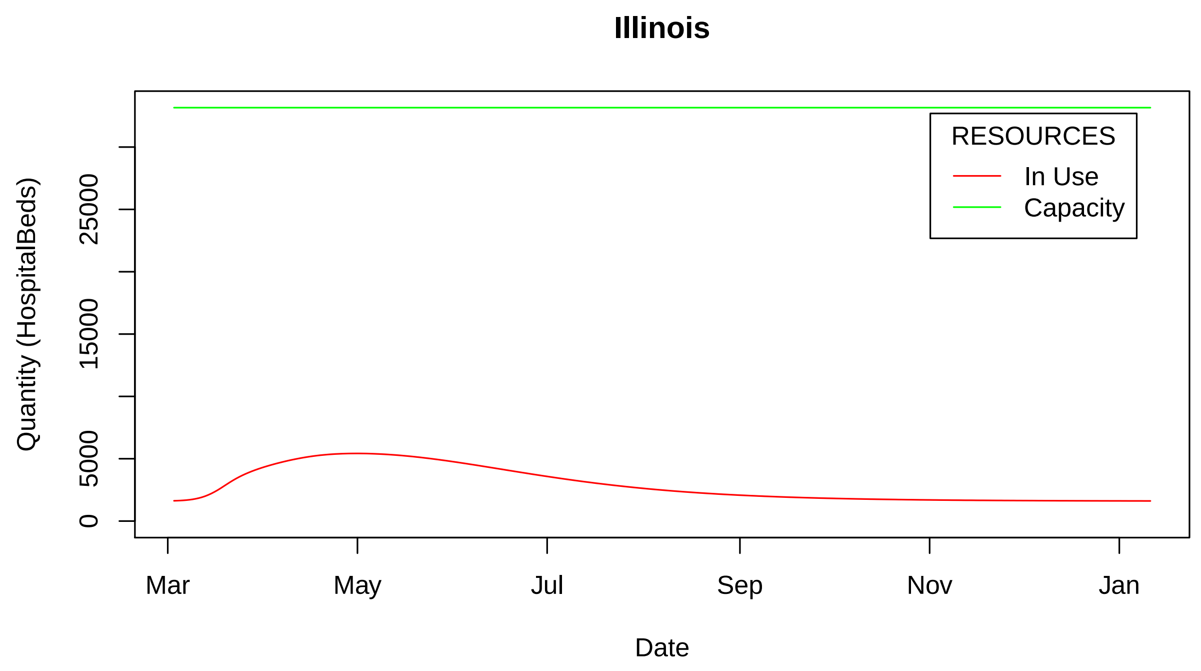
```r
covid_data13 <- covid_data_1013
covid_data14 <- covid_data_1014

plotResourceAdequacy <- function(state, resource, dataset){

  if (dataset == "covid_data_1013"){
    state_name <- covid_data13[covid_data_1013$stateName == state,]
  }else if(dataset == "covid_data_1014"){
    state_name <- covid_data14[covid_data_1014$stateName == state,]
  }else{
    stop("ERROR: This function cannot proceed
         forward with the invalid input. Please check dataset spelling.")}

  states <- c(covid_data_1013$stateName)
  if (!state %in% states){
    stop("ERROR: This function cannot proceed
         forward with the invalid input. Please check state spelling.")
  }else{
    state <- state}

  date <- as.Date(state_name[,1])

  if (resource == "Ventilator"){

    ventilator <- state_name[,6:7]
    vcapacity <- ventilator[,2]
    vin_use <- ventilator[,1]

    plot(date,
         vcapacity,
         type="l",
         main=state,
         xlab="Date",
         ylab="Quantity (Ventilator)",
         ylim=c(0,max(vcapacity,vin_use)),
         col="green")
    lines(date, vin_use, col="red")
    legend('topright',
           inset= 0.05,
           c("In Use", "Capacity"),
           lty=1,
           col = c("red","green"),
           title = "RESOURCES")

  }else if (resource == "HospitalBeds"){

    hospitalbed <- state_name[,2:3]
    hbcapacity <- hospitalbed[,2]
    hbin_use <- hospitalbed[,1]

    plot(date,
         hbcapacity,
         type="l",
         main=state,
         xlab="Date",
         ylab="Quantity (HospitalBeds)",
         ylim=c(0,max(hbcapacity,hbin_use)),
         col="green")
    lines(date, hbin_use, col="red")
    legend('topright',
           inset= 0.05,
           c("In Use", "Capacity"),
           lty=1,
           col = c("red","green"),
           title = "RESOURCES")

  }else if (resource == "ICUBeds"){

    icubed <- state_name[,4:5]
    ibcapacity <- icubed[,2]
    ibin_use <- icubed[,1]

    plot(date,
         ibcapacity,
         type="l",
         main=state,
         xlab="Date",
         ylab="Quantity (ICUBeds)",
         ylim=c(0,max(ibcapacity,ibin_use)),
         col="green")
    lines(date, ibin_use, col="red")
    legend('topright',
           inset= 0.05,
           c("In Use", "Capacity"),
           lty=1,
           col = c("red","green"),
           title = "RESOURCES")

    }else{
      stop("ERROR: This function cannot proceed
           forward with the invalid input. Please check resource spelling.")}
}

plotResourceAdequacy("California", "ICUBeds", "covid_data_1014")
```



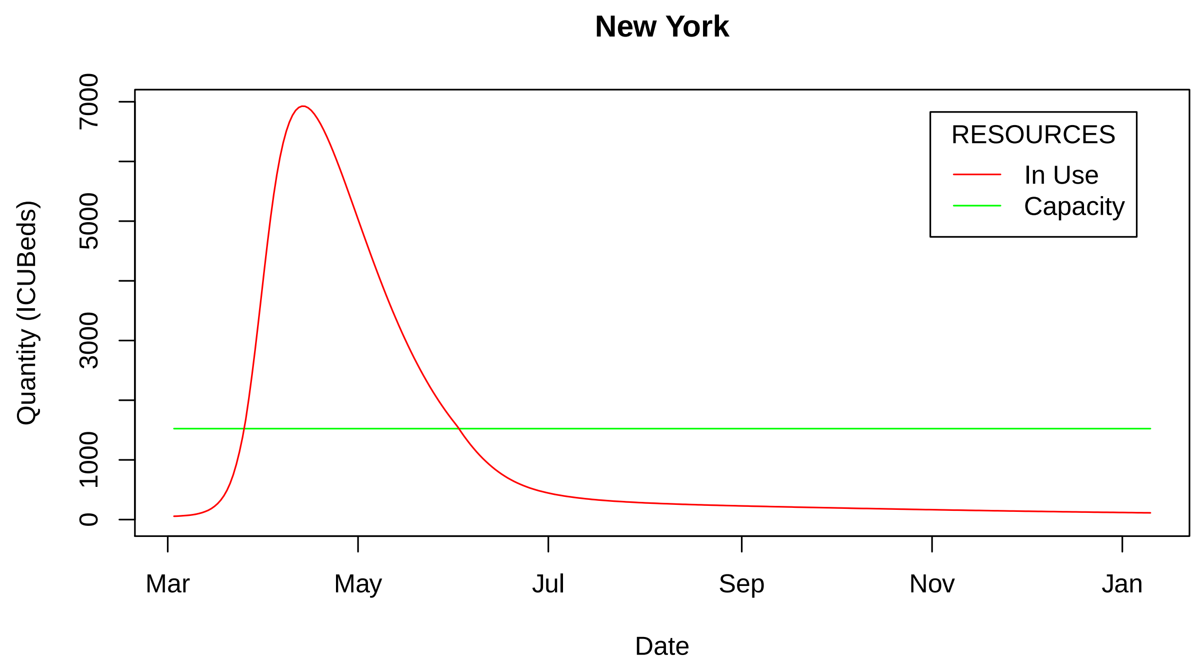
```r
plotResourceAdequacy("Wyoming", "Ventilator", "covid_data_1013")
```



```r
plotResourceAdequacy("Illinois", "HospitalBeds", "covid_data_1014")
```



```r
plotResourceAdequacy("New York", "ICUBeds", "covid_data_1013")
```



2.2 Write Up

We began this project by laying out how we wanted to begin each part of the questions in order to get working, interchangeable, functions at the end. The following is a summary of how we worked together to complete these problems in the most effective way possible while leaving no possible error to occur.

Before we started writing the code for the functions, we had to import the covid data into R Markdown. After having done that, we took out the columns that we didn't need by using a "-c[8:15]." In order to make this first function "plotResourceAdequacy", we had to have three arguments, (state, resource, and dataset), that were all changeable. We started by making the dataset interchangeable between the 1014 covid data and the 1013 data. We made an If and Else If statement to have the data we use for the state name come from the specific dataset you choose to use in the function.

With this same statement, we had it give us the rows for the state that we chose using a logical function ("=="). Then, we made the plot change based on which resource we chose. We made three If statements that were almost identical aside from the parts that are specific for that resource chosen. If resource is set as "Ventilator," then our variable "ventilator" is the columns 6 and 7 for our given state name. Our next variable, "vcapacity" is set as the second column in "ventilator," because that column corresponds to the capacity for ventilators. Then, we have "vin_use" to be the first column of "ventilator" as that column corresponds to the amount of ventilators in use.

After we made these variables, we wrote the code for actually plotting the line graph: date is the x-axis, vcapcity is the y-axis, and the type equaling "l" makes it a line graph. Then we have the titles for the whole graph, the x-axis, and the y-axis, and we made the y limit to go from zero to the larger of the two maximums of the capacity and in use data. After this we have what the other line is – ventilators in use (and again, the date is the x-axis) – and finally we put in a legend that shows what the colors of the lines mean. This was all for the resource "Ventilator," so we had 3 of these for the three different variable names all doing practically the same thing, aside from which original resource data is used. We finally ran our function four times so that we would have four different line graphs with different inputs.

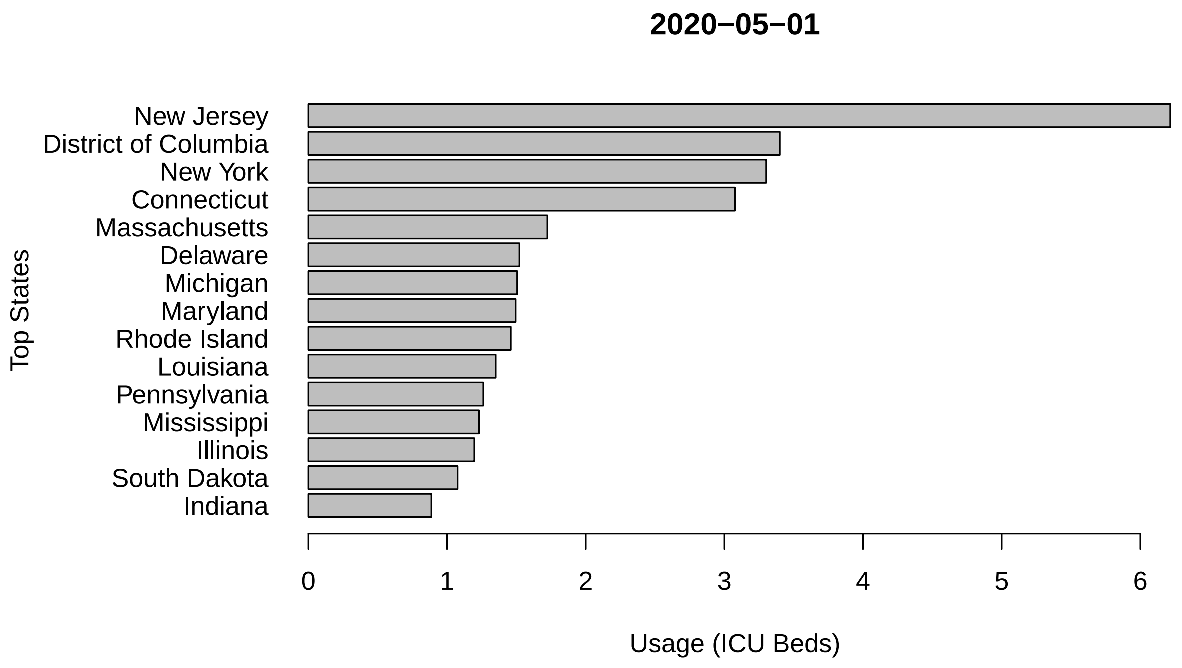
3 Function 2: plotTopUsage

3.1 Code

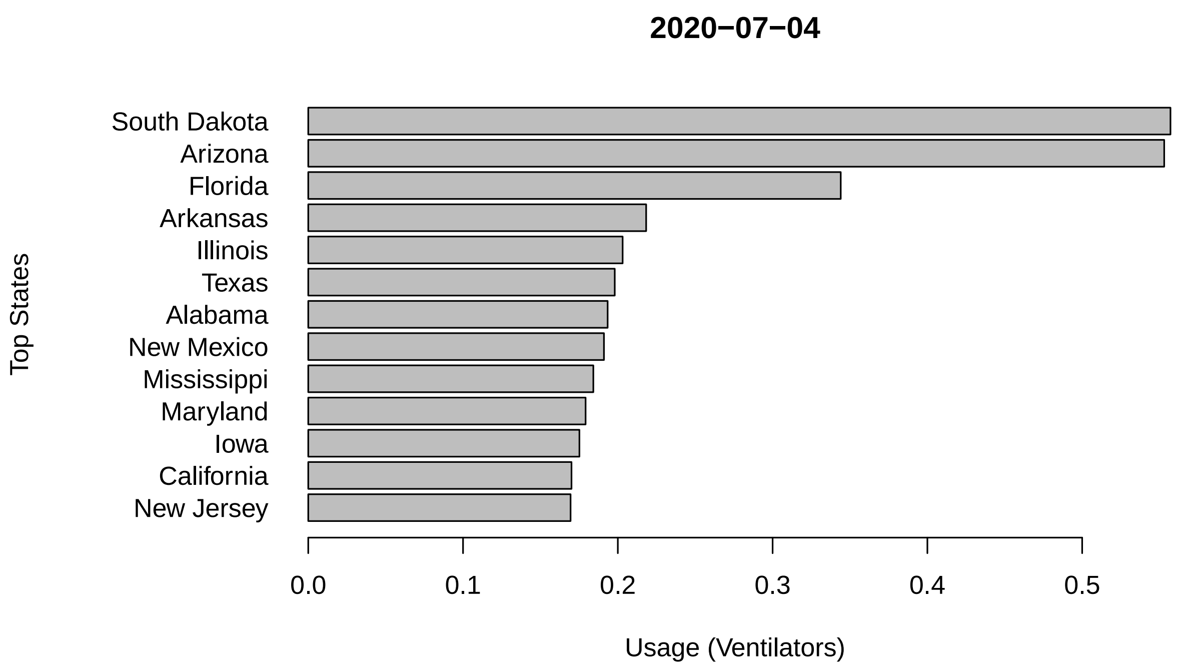
```r
covid_data13 <- covid_data_1013
covid_data14 <- covid_data_1014

plotTopUsage <- function(date,resource,dataset,top_num){

  if (dataset == "covid_data_1013"){
    date_name <- covid_data13[covid_data13$date == as.Date(date),]
  }else if(dataset == "covid_data_1014"){
    date_name <- covid_data14[covid_data14$date == as.Date(date),]
  }else{
    stop("ERROR: This function cannot proceed
         forward with the invalid input. Please check dataset spelling.")}

  dates <- c(covid_data_1013$date,covid_data_1014$date)
  if (!date %in% dates){
    stop("ERROR: This function cannot proceed
         forward with the invalid input. Please check date.")
```

```r
}else{
  date <- date}

if (resource == "Ventilator"){

  ventilator <- date_name[,6:7]
  vcapacity <- ventilator[,2]
  vin_use <- ventilator[,1]

  vusage <- c(vin_use/vcapacity)
  date_name$VentilatorUsage=vusage

  vbar <- date_name[order(date_name$VentilatorUsage,decreasing = TRUE),]

  vbarplot <- vbar[top_num:1,c(8:9)]
  par(mar=c(5.1,10,4.1,2.1))
  barplot(
    vbarplot$VentilatorUsage,
    horiz=TRUE,
    names.arg=vbarplot$stateName,
    las=1,
    xlab="Usage (Ventilators)",
    main=date)
  mtext(side=2, text="Top States",line=8.5)

}else if (resource == "HospitalBeds"){

  hospitalbed <- date_name[,2:3]
  hbcapacity <- hospitalbed[,2]
  hbin_use <- hospitalbed[,1]

  hbusage <- c(hbin_use/hbcapacity)
  date_name$HospitalBedUsage=hbusage

  hbbar <- date_name[order(date_name$HospitalBedUsage,decreasing = TRUE),]

  hbbarplot <- hbbar[top_num:1,c(8:9)]
  par(mar=c(5.1,10,4.1,2.1))
  barplot(
    hbbarplot$HospitalBedUsage,
    horiz=TRUE,
    names.arg=hbbarplot$stateName,
    las=1,
    xlab="Usage (Hospital Beds)",
    main=date)
  mtext(side=2, text="Top States",line=8.5)

}else if (resource == "ICUBeds"){

  icubed <- date_name[,4:5]
  ibcapacity <- icubed[,2]
  ibin_use <- icubed[,1]
```

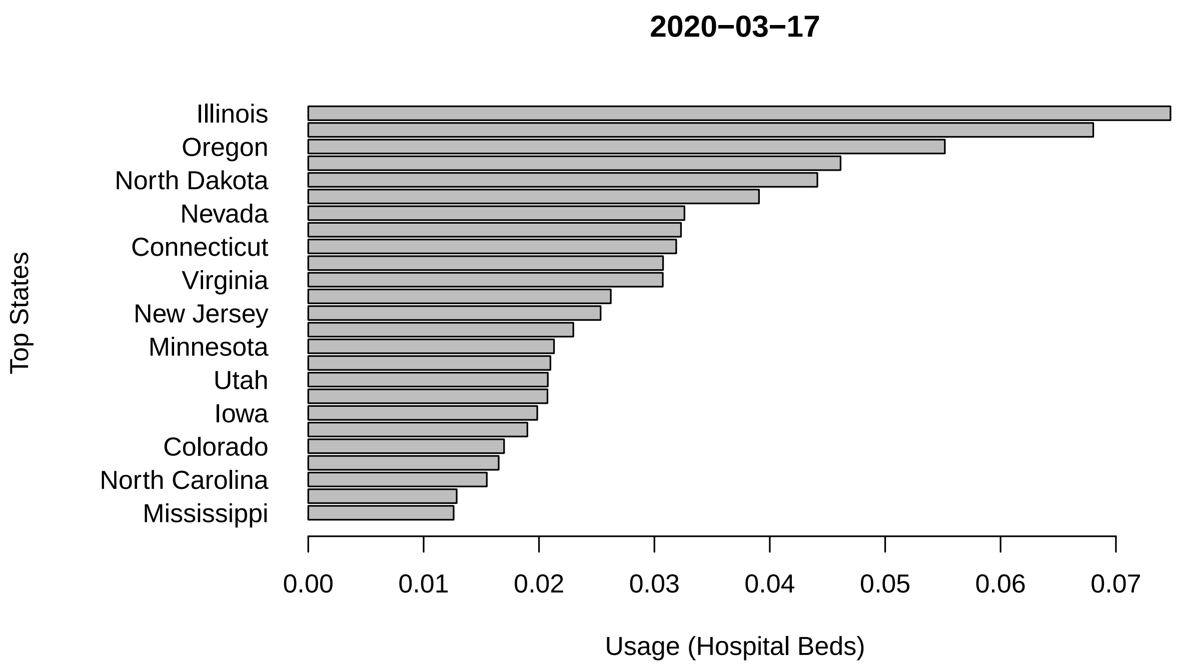
```
    ibusage <- c(ibin_use/ibcapacity)
    date_name$ICUBedUsage=ibusage

    ibbar <- date_name[order(date_name$ICUBedUsage,decreasing = TRUE),]

    ibbarplot <- ibbar[top_num:1,c(8:9)]
    par(mar=c(5.1,10,4.1,2.1))
    barplot(
      ibbarplot$ICUBedUsage,
      horiz=TRUE,
      names.arg=ibbarplot$stateName,
      las=1,
      xlab="Usage (ICU Beds)",
      main=date)
    mtext(side=2, text="Top States",line=8.5)

  }else{
      stop("ERROR: This function cannot proceed
           forward with the invalid input. Please check resource spelling.")}
}

plotTopUsage("2020-05-01", "ICUBeds","covid_data_1014",15)
```



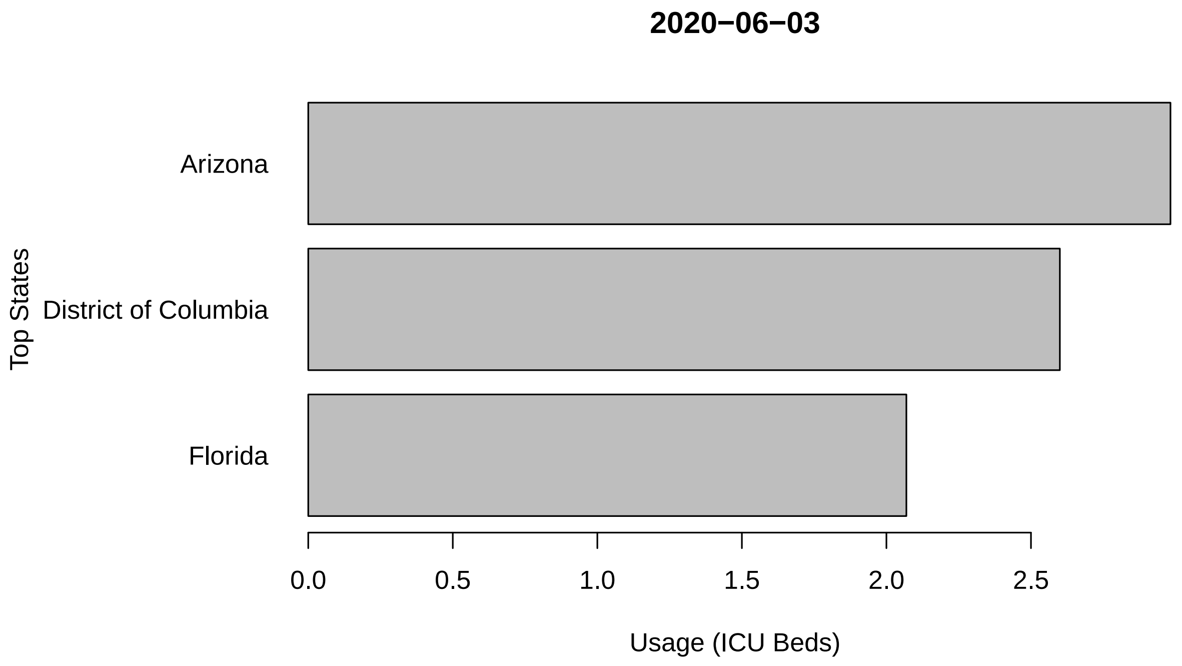
```
plotTopUsage("2020-07-04", "Ventilator","covid_data_1013",13)
```



```r
plotTopUsage("2020-03-17", "HospitalBeds","covid_data_1014",25)
```



```r
plotTopUsage("2020-06-03", "ICUBeds","covid_data_1013",3)
```



3.2 Write Up

We began our second question the same way we started the first function, by renaming the datasets and setting out a function "plotTopUsage" with four arguments, (date, resource, dataset, and top_num). The first variable we worked with was the datasets. We started by making the dataset interchangeable between the 1014 covid data and the 1013 data. We made an If and Else If statement to have the data we use for the date come from the specific dataset you choose to use in the function.

Next, we wanted to focus on all the resources: ventilators, hospital beds, and ICU beds. Although each of these resources are different, their codes were quite similar. We located the two columns that are necessary for the following lines of code and separated these two columns by their two purposes: the resources in use, and the amount of resources available(capacity). Since we were focusing greatly on their usage, we used the equation, usage = in_use/capacity, in order to get the variable we wanted to evaluate. After calculating this, we made each of these three resources usages their own columns and put them into a new dataset, "vbarplot." In doing this, it was easier to create the bar plot because all the information needed to complete these bar plots had their own respective columns in the new dataset.

We ordered the barplot's columns in descending order to make our data reflect the greatest usage to the least amount of usage. After doing this for each of the resources, we extracted the column we needed to make the graph by using "vbarplot$VentilatorUsage." Then, we set the margins, made the graph horizontal, named the different axis', etc. This would return the necessary barplot for each resource entered. We then ran our function four times with different inputs to get our four graphs.

4 Extra Credit

4.1 Write Up

The first thing we took into account was the fact that every variable in the functions could be misspelled or misentered. Because of this, we dealt with each variable separately. We used stop functions at the end of all of the If Statements in case there was an invalid input. The function returns the message: “This function cannot proceed forward with the invalid input. Please check “variable” spelling.” For our code, we specified which variable had an issue being read. For example, if the state entered was not in the state name column (it was misspelled or another error), it would return “This function cannot proceed forward with the invalid input. Please check state spelling.” We included this because we thought it would be beneficial to make this code specific to the actual issue to help the user locate the issue immediately. We did this for every variable in both of the functions, so all inputs could be validated.