Laura Stoddard VIP Timeline

Matt Clark

3/30/2020

# Schedule

|  |  |  |
| --- | --- | --- |
| Week | Focus | Deliverable |
| 3/30/2020 | Data Cleaning | Cleaned up dataset with projected fees for relevant parks |
| 4/6/2020 | Plotting 1 | Previous smoke effects plot |
| 4/13/2020 | Plotting 2 | Projected smoke effects plot |
| 4/20/2020 | Write-up / Catch-up | Brief write-up of results |
| 4/27/2020 | Lightening talk | Lightening talk |

# Weekly Specifics

## 3/30/2020 - Data Cleaning

This week, it would be great if you could clean up the dataset that you collected and fill in what we think the entry fees may have been for each park for the years we don’t have data on.

To do this, you will need to do 4 main steps.

1. Upload the data
2. Remove rows that we don’t know
3. Make a very simple model
4. Use that model to predict what the unknown fees likely would have been

I’ll give you some guidance on how to do all 4 of these steps, but I do kind of want you to struggle through some of it on your own. Here’s an example of how you would do this for just 2 parks for 2 years. I want you to take these steps and apply them to all parks for all the years we don’t know

library(tidyverse)  
FEdat<-read\_csv("~/SmokeProject/Data/LauraStoddardEntryFeeCsv.csv") #you can call this whatever you want. Keep in mind that your file path will be different than mine!

## Parsed with column specification:  
## cols(  
## Park = col\_character(),  
## Year = col\_double(),  
## SingleVisitorEntryFee = col\_double(),  
## SingleVehicleEntry = col\_double()  
## )

head(FEdat) #make sure the data came through

## # A tibble: 6 x 4  
## Park Year SingleVisitorEntryFee SingleVehicleEntry  
## <chr> <dbl> <dbl> <dbl>  
## 1 Acadia NP 1980 NA NA  
## 2 Acadia NP 1981 NA NA  
## 3 Acadia NP 1982 NA NA  
## 4 Acadia NP 1983 NA NA  
## 5 Acadia NP 1984 NA NA  
## 6 Acadia NP 1985 NA NA

Now that we’ve loaded the data, we want to remove the single vehicle column. We aren’t using it yet and it has rows with NAs that the single visitor column doesn’t. We will filter by NAs later so we want this column gone.

d<-FEdat %>% dplyr::select(Park,Year,SingleVisitorEntryFee) #keep only these columns  
d<-na.omit(d) #get rid of rows with any NA  
head(d) #check it out

## # A tibble: 6 x 3  
## Park Year SingleVisitorEntryFee  
## <chr> <dbl> <dbl>  
## 1 Acadia NP 1997 3  
## 2 Acadia NP 1998 3  
## 3 Acadia NP 1999 5  
## 4 Acadia NP 2000 5  
## 5 Acadia NP 2001 5  
## 6 Acadia NP 2002 5

Now we’re going to make a model based on these data. Think of this as y=mx+b that you learned in algebra. Y is the predicted fee, x is the year, m is the slope, and b is an intercept. In this case we also have a x2 and m2 which are a 1 or 0 for each park and a slope (m2) for each park.

This is what’s called an generalized linear model. The “Poisson” part just tells the computer that we are modeling values that are discrete counts (1,4,7,etc.)

feefit<-stan\_glm(SingleVisitorEntryFee~Year+Park,family="Poisson",data=d,iter=6000)

Now we can use this model to predict the fees for the parks we don’t know.

Here’s an example of that

preddata<-data.frame(Year = c(2012,2016), Park = c("Arches NP", "Acadia NP")) #make a small, 2 row data frame  
  
apply(posterior\_predict(feefit,preddata),2,mean) #predict our fees for thse parks for those years based on our model.

## 1 2   
## 6.861500 7.512083

Notice that this gives predictions for those two parks for those years.

We want predictions for every row that we are missing. We definitely don’t want to type this in by hand.

On your own, subset the original datframe by just the rows which have NA in the fee column. Use this new dataframe in the place of preddat in the example.

#### Hint:

NAdata<-data%>%dplyr::filter(fee == NA)  
  
apply(posterior\_predict(feefit,NAdata),2,mean)

You should then be able to add this output as a column to the dataframe with all nas that you made.

#### Hint:

NAdata$PredictedVis<-apply(posterior\_predict(feefit,NAdata),2,mean)

Ideally, you will then use R to fill in all of the nas in the original data frame with our predicted values. You should also make a new column indicating if the value is real or a projection.

#### Hint:

merge(FEdata,NAdata, by=1:2)

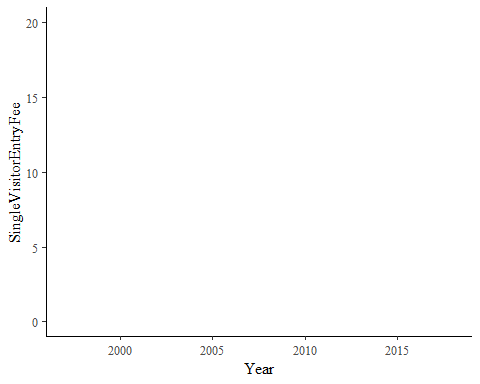
## 4/6/2020 Plotting 1

We will use the package ggplot2 to plot the data that you have collected/extrapolated.

If you remember from before, the ggplot() function follows the [grammar of graphics](https://towardsdatascience.com/a-comprehensive-guide-to-the-grammar-of-graphics-for-effective-visualization-of-multi-dimensional-1f92b4ed4149), a theory for how to visualize data that relies on stacking layers on top of each other.

When you create ggplot() code, the top most code will be the first layer created. For our purposed, we will always want this to be the axes of the figure. We will do this by specifying the data and the x and y axes. Remember that the x and y axes need to be wrapped in an aesthetic function aes(). This just tells ggplot to look for these things in the dataset that we specified.

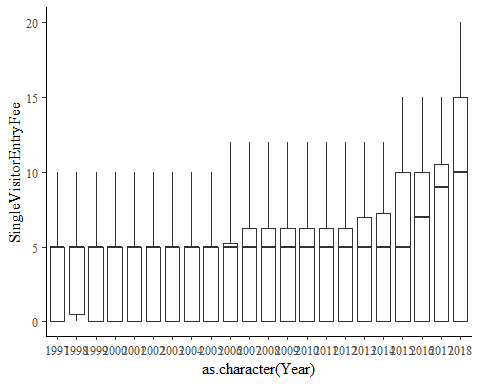
library(ggplot2) #load the package  
  
ggplot(data= d, aes(x=Year,y=SingleVisitorEntryFee))



You’ll see that this just “sets the stage” for the rest of the plot. We haven’t told it what kind of plot we want yet.

Next, we will layer the data on top of this background.

ggplot(data= d, aes(x=as.character(Year),y=SingleVisitorEntryFee))+  
 geom\_boxplot()



Is this the best way to visualize this simple dataset? probably not. Can you think of better way?

There is a ton of flexibility in ggplot2. See if you can make this plot better.

Once you do that, see if you can plot your data that you created by extrapolating from the fee data we had.

I like to hand draw my plots first so I know what I’m aiming for.

If you want help and inspiration check out the [Cookbook for ggplot](http://www.cookbook-r.com/Graphs/) and the [ggplot gallery](https://www.r-graph-gallery.com/).

## 4/13/2020 Plotting 2

Now that you have some experience with the basics of plotting, let’s work on cleaning up our figures so they look good and could go into a publication.

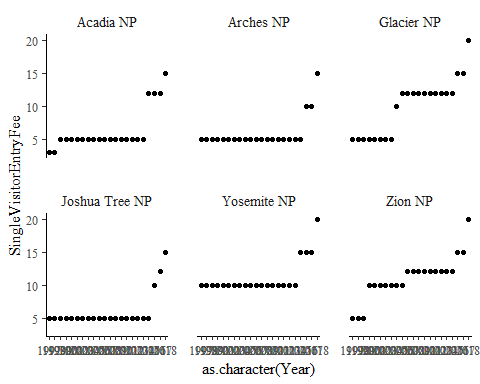
When we’re dealing with grouped data like this park data, we often want to facet the data so it’s in small multiples, or repeated figures showing the same data for different groups.

Let’s first subset the parks data so that there’s only 6 parks.

d<-d%>%filter(Park %in% c("Acadia NP", "Yosemite NP", "Glacier NP","Zion NP", "Arches NP","Joshua Tree NP"))

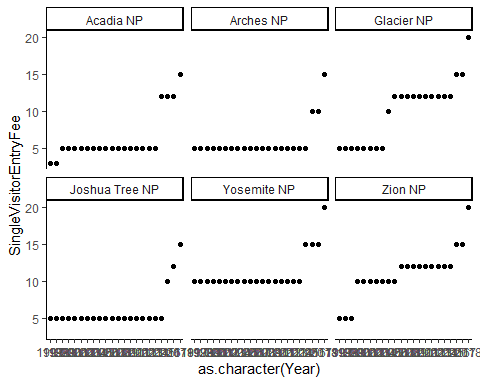
Now we might want to look at the entry fees for each of these parks in their own figure. It would be a pain to code those figures for each park. Luckily we can use faceting.

ggplot(d,aes(x=as.character(Year),y=SingleVisitorEntryFee))+  
 geom\_point()+  
 facet\_wrap(~Park) #this is the part that does the faceting

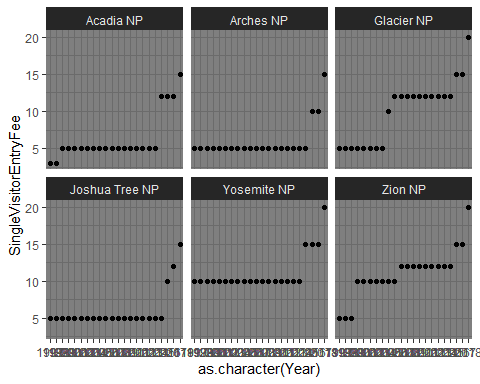


An easy thing we can do to make our plots look better is just to apply the build in ggplot themes to them. Here’s some examples

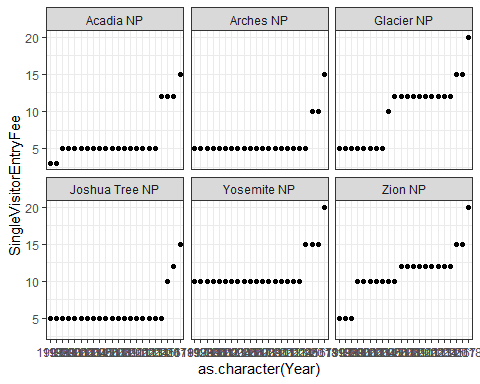
ggplot(d,aes(x=as.character(Year),y=SingleVisitorEntryFee))+  
 geom\_point()+  
 facet\_wrap(~Park)+theme\_classic()



ggplot(d,aes(x=as.character(Year),y=SingleVisitorEntryFee))+  
 geom\_point()+  
 facet\_wrap(~Park)+theme\_dark()



ggplot(d,aes(x=as.character(Year),y=SingleVisitorEntryFee))+  
 geom\_point()+  
 facet\_wrap(~Park)+theme\_bw()



These themes are great, but usually, we want to customize them a bit more.

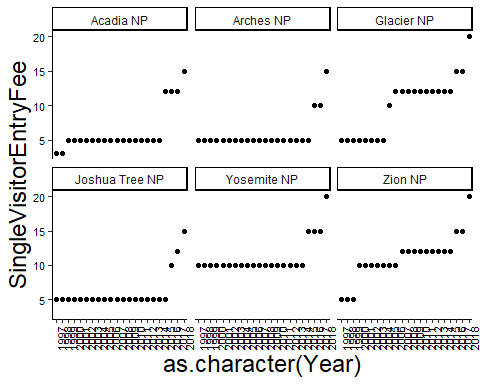
We can do that by creating our own themes and stacking them on top.

There’s a TON of flexibility here. No one has all these options memorized. A good rule is that if you have an idea of what you want to do in ggplot it’s almost definitely possible and someone has probably thought of it before you. Googling things is usually a good place to start.

mytheme<- theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(),  
 plot.title = element\_text( size=18, color="black",face="bold"),  
 axis.title.x = element\_text( size=18),  
 axis.title.y = element\_text( size=18),  
 axis.text=(element\_text(color="black", size=8)),  
 axis.text.x = element\_text(angle = 90, hjust = 1),  
 legend.title = element\_text(colour="black", size=18),  
 legend.text = element\_text( size = 14))

Now that I have that saved, I can just add it to the end of my plot to make the changes I want

ggplot(d,aes(x=as.character(Year),y=SingleVisitorEntryFee))+  
 geom\_point()+  
 facet\_wrap(~Park)+theme\_classic()+  
 mytheme



We’ve really just scratched the surface here, but hopefully this is enough to get you started. Don’t hesitate to reach out when you have questions!