Final

Jacob Wisbeck

5/14/2019

Landing at San Francisco International Airport Explored and Other abitrary things: The Data Pipeline

1. Setting up R

R Libraries

```
knitr::opts_chunk$set(echo = TRUE)
## Includes stringr, forcats, tibble, purrr, readr, tidyr, dplyr and ggplot2
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────────────────────
──────────────────────────────── tidyverse 1.2.1 ──
```

```
## ✔ ggplot2 3.1.0     ✔ purrr   0.3.0
## ✔ tibble  2.0.1     ✔ dplyr   0.7.8
## ✔ tidyr   0.8.2     ✔ stringr 1.4.0
## ✔ readr   1.3.1     ✔ forcats 0.3.0
```

```
## ── Conflicts ─────────────────────────────────────────────────────────────────
──────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
## Used to improve texts: https://www.r-project.org/nosvn/pandoc/crayon.html
library(crayon)
```

```
##
## Attaching package: 'crayon'
```

```
## The following object is masked from 'package:ggplot2':
##
##     %+%
```

```
## Used for improved piping: https://cran.r-project.org/web/packages/magrittr/vignettes/
magrittr.html
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
```

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

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
## Used for Making date modification easier: https://lubridate.tidyverse.org/
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
## Used in tidying data and manipulating tables: https://cran.r-project.org/web/packages/broom/vignettes/broom.html
library(broom)
## Stringr is for string manip and string functions: https://www.rdocumentation.org/packages/stringr/versions/1.4.0
library(stringr)
## Not needed but useful for map visualizations: https://rstudio.github.io/leaflet/
library(leaflet)
```

2. Ingesting, Cleaning and/or Building A Dataset

Structured Ingestion: An Aside

There are many tools that enable you to take the tables you have and build a table that is more effective in containing relevant information of your data. For example, below I used Tableau Prep Builder to create a table that combined College Rankings and a Federal data set on US colleges. All things done in Tableau can be done here in R using the appropriate libraries, but because Tableau provides a better visualization of the manipulation steps I use this to create the conceptual methodology if one was to build a data set as an example.

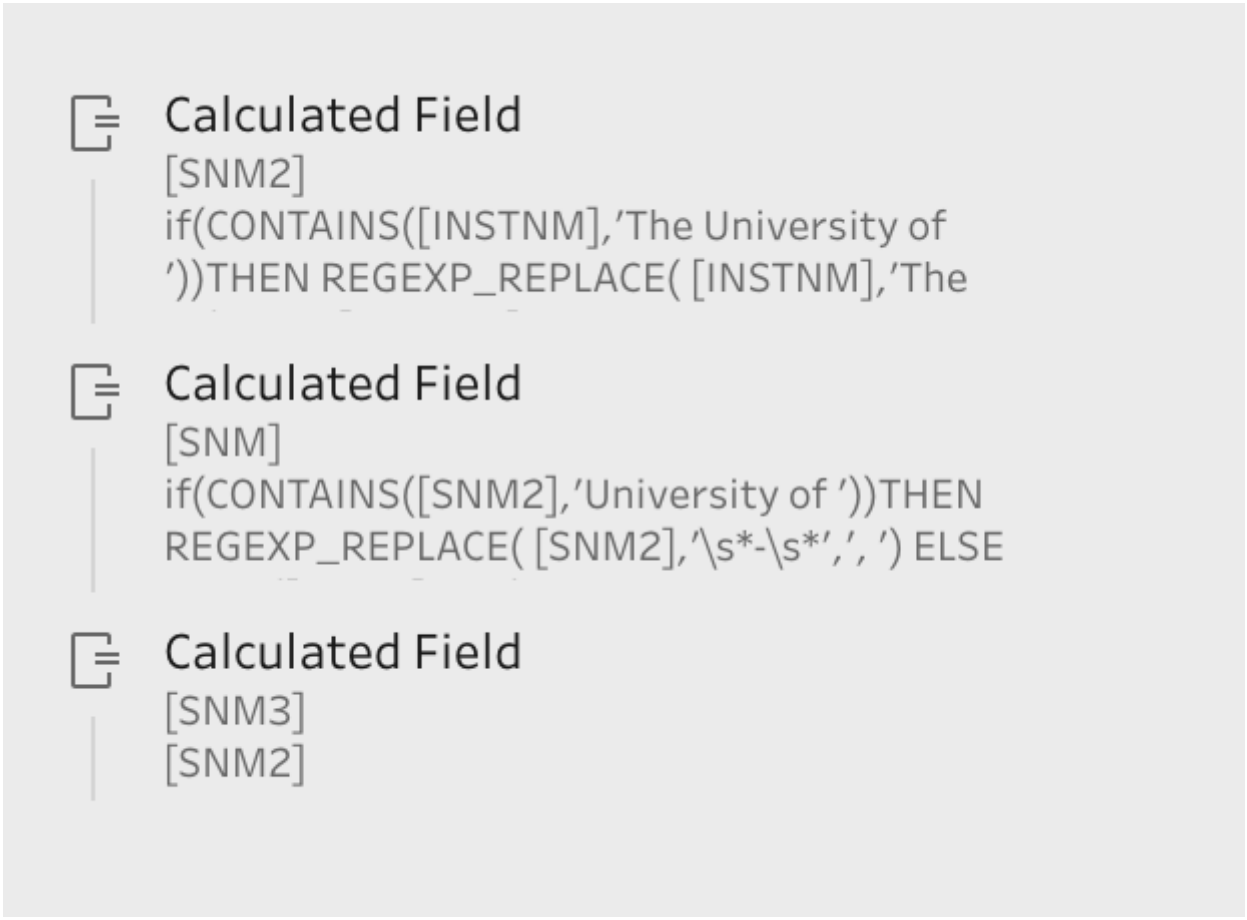
First, I need to state the tables I will be combining and why. Let's say I have 4 tables, cwurData.csv:a table of top 1000 World ranked colleges, hd2017.csv, ic2017.csv, ic2017_ay.csv: 3 tables that are all from the same sourcs, and already have entity resolution on a University ID variable called UNITID, so the biggest issue is that cwurData.csv Does not have this Resolution with the other datasets.

Tidying Data & Entity Resolution

These two different sources of data do have a similarity and that is that the hd2017.csv and cwurData.csv both contain a String variable that contains the name of the universities. And holy schemoly is it messy. If you were to try to have entity resolution between these two lists, you would only get about 160 matches between the two tables. We need to up our game so let's try to see why some school names for the same school are different between the two data sets. What ends up being the case is that the college names for state schools mostly in the cwur data do not contain as long as a name, nor specify which branch of the state school system it has a rank for, we will get back to this as this may end up being a hard problem made easier. Another issue is that there are sometime The's and different and's or at's in the names. So here is where we clean up the data, this is done with regular expressions, or "Regex". To learn more on Regular Expressions go here:

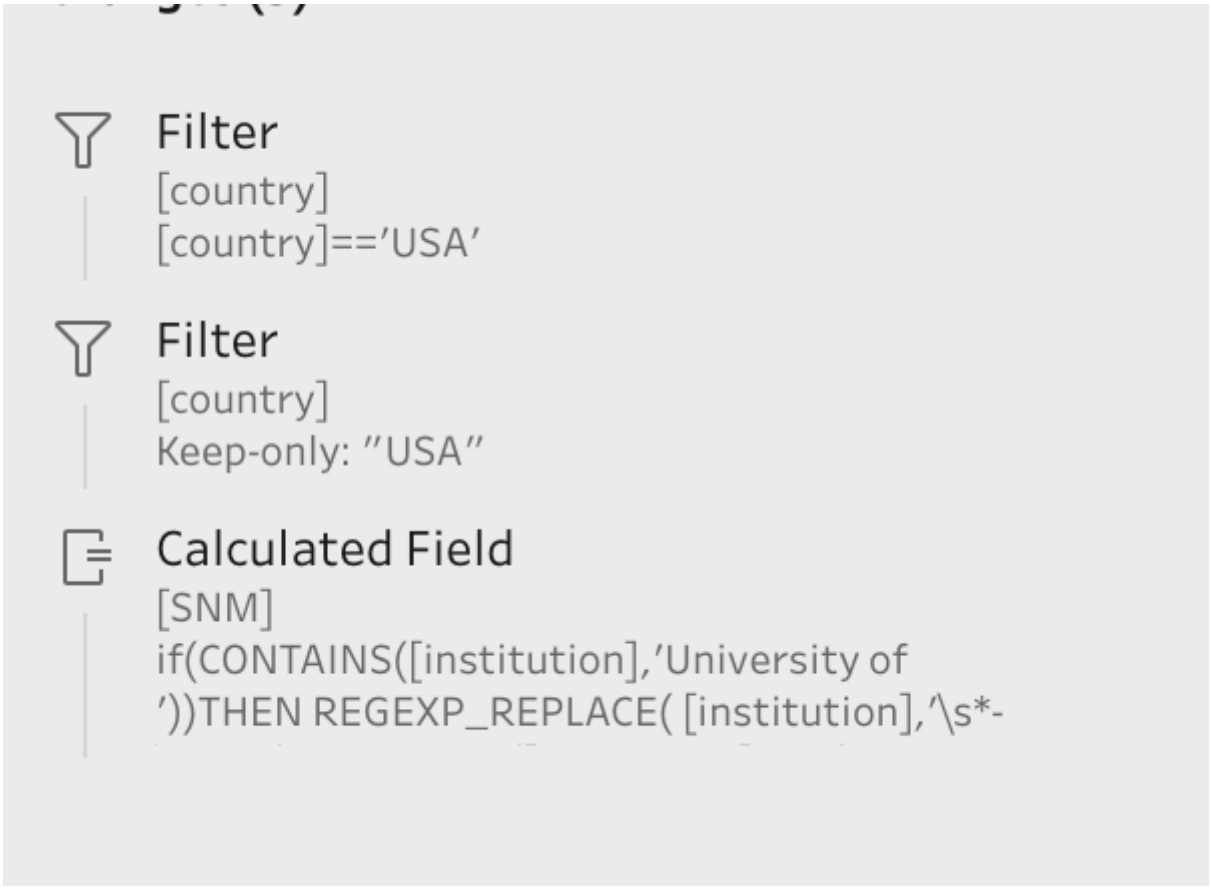
https://www.computerhope.com/jargon/r/regex.htm (https://www.computerhope.com/jargon/r/regex.htm)

In short we are going to create a new column in each table that ignores any The's, commas, at's, and's, of's and other small english modifiers to try and create a unique identifier that has better resolution of values:



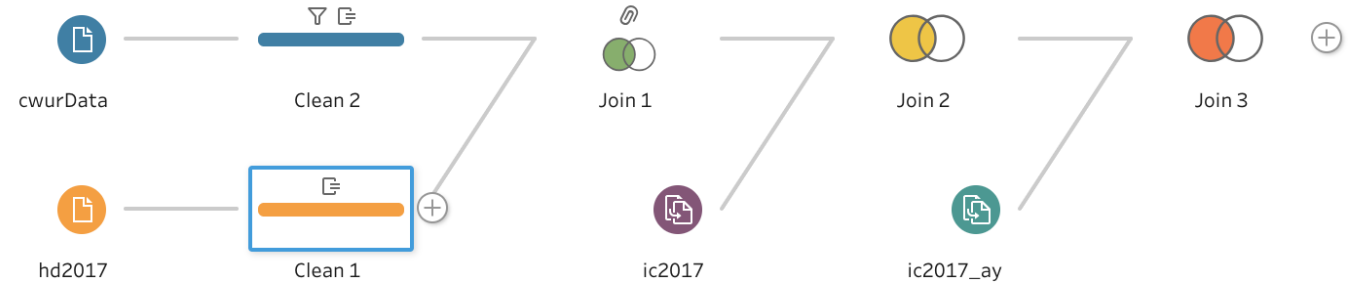
regex

Also to remove the clutter in the cwur dataset which contains international data points we want to remove all the non-US Schools and focus on just american universities:



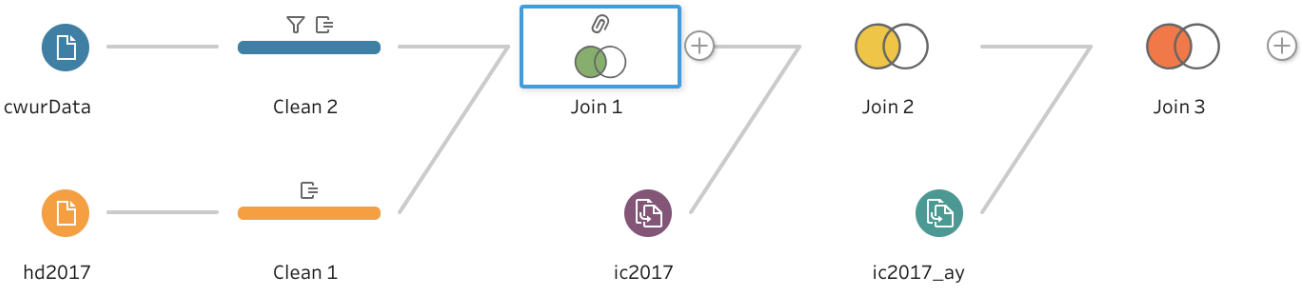
Tidying narrows the cwur down to 573 schools. The new column also sports 48 more matches than before bringing the total 1:1 resolutions to 208 leaving the remaining schools to be 365 in the cwur that need matching and in Tableau that is easy, and so too in R by following the steps of going by hand and searching for the school in the larger data file of hd2017. And this is where we see the expensive portion of data analysis is in the cleaning of data:

https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#32efb3c56f63 (https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#32efb3c56f63)

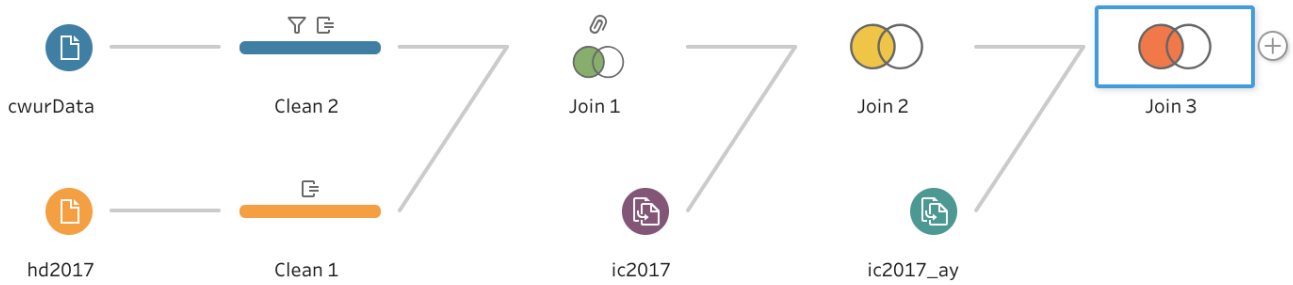


Clean1

Then you can group and replace the remaining values by hand



Joining all the tables together after joining two of the separate sources of data:



NOTE: For the rest of the project, I will be using a premade data set sourced from:

https://www.kaggle.com/san-francisco/sf-air-traffic-passenger-and-landings-statistics/downloads/sf-air-traffic-passenger-and-landings-statistics.zip/68 (https://www.kaggle.com/san-francisco/sf-air-traffic-passenger-and-landings-statistics/downloads/sf-air-traffic-passenger-and-landings-statistics.zip/68)

Loading the Tables

To read in a file that is a .csv you can use the readr library function read_csv. This works with a url to a .csv file as well as a local file.

```
landing<-read_csv("./data/air-traffic-landings-statistics.csv")%>%
  as.data.frame()
```

```
## Parsed with column specification:
## cols(
##   `Activity Period` = col_double(),
##   `Operating Airline` = col_character(),
##   `Operating Airline IATA Code` = col_character(),
##   `Published Airline` = col_character(),
##   `Published Airline IATA Code` = col_character(),
##   `GEO Summary` = col_character(),
##   `GEO Region` = col_character(),
##   `Landing Aircraft Type` = col_character(),
##   `Aircraft Body Type` = col_character(),
##   `Aircraft Manufacturer` = col_character(),
##   `Aircraft Model` = col_character(),
##   `Aircraft Version` = col_character(),
##   `Landing Count` = col_double(),
##   `Total Landed Weight` = col_double()
## )
```

```
passengers<-read_csv("./data/air-traffic-passenger-statistics.csv")%>%
  as.data.frame()
```

```
## Parsed with column specification:
## cols(
##   `Activity Period` = col_double(),
##   `Operating Airline` = col_character(),
##   `Operating Airline IATA Code` = col_character(),
##   `Published Airline` = col_character(),
##   `Published Airline IATA Code` = col_character(),
##   `GEO Summary` = col_character(),
##   `GEO Region` = col_character(),
##   `Activity Type Code` = col_character(),
##   `Price Category Code` = col_character(),
##   Terminal = col_character(),
##   `Boarding Area` = col_character(),
##   `Passenger Count` = col_double()
## )
```

Showing a short preview of the Data

Instead of making the program print the max amount of rows of a table, or eve the whole thing, you can use head to show the first limited number of rows. This is helpful in being quicker and cleaner than the whole table.

```
head(passengers)
```

```
##   Activity Period Operating Airline Operating Airline IATA Code
## 1          200507      ATA Airlines                         TZ
## 2          200507      ATA Airlines                         TZ
## 3          200507      ATA Airlines                         TZ
## 4          200507        Air Canada                         AC
## 5          200507        Air Canada                         AC
## 6          200507         Air China                         CA
##   Published Airline Published Airline IATA Code   GEO Summary GEO Region
## 1      ATA Airlines                         TZ      Domestic         US
## 2      ATA Airlines                         TZ      Domestic         US
## 3      ATA Airlines                         TZ      Domestic         US
## 4        Air Canada                         AC International     Canada
## 5        Air Canada                         AC International     Canada
## 6         Air China                         CA International       Asia
##   Activity Type Code Price Category Code      Terminal Boarding Area
## 1           Deplaned            Low Fare    Terminal 1             B
## 2           Enplaned            Low Fare    Terminal 1             B
## 3     Thru / Transit            Low Fare    Terminal 1             B
## 4           Deplaned               Other    Terminal 1             B
## 5           Enplaned               Other    Terminal 1             B
## 6           Deplaned               Other International            G
##   Passenger Count
## 1           27271
## 2           29131
## 3            5415
## 4           35156
## 5           34090
## 6            6263
```

```
head(landing)
```

```
##   Activity Period                      Operating Airline
## 1          200204                           ATA Airlines
## 2          200204                           ATA Airlines
## 3          200204                           ATA Airlines
## 4          200204 Aeroflot Russian International Airlines
## 5          200204                             Air Canada
## 6          200204                             Air Canada
##   Operating Airline IATA Code                    Published Airline
## 1                          TZ                         ATA Airlines
## 2                          TZ                         ATA Airlines
## 3                          TZ                         ATA Airlines
## 4                       <NA> Aeroflot Russian International Airlines
## 5                          AC                           Air Canada
## 6                          AC                           Air Canada
##   Published Airline IATA Code   GEO Summary GEO Region
## 1                          TZ      Domestic         US
## 2                          TZ      Domestic         US
## 3                          TZ      Domestic         US
## 4                       <NA> International     Europe
## 5                          AC International     Canada
## 6                          AC International     Canada
##   Landing Aircraft Type Aircraft Body Type Aircraft Manufacturer
## 1             Passenger        Narrow Body                Boeing
## 2             Passenger        Narrow Body                Boeing
## 3             Passenger          Wide Body              Lockheed
## 4             Passenger          Wide Body                Boeing
## 5             Passenger        Narrow Body                Boeing
## 6             Passenger        Narrow Body                Boeing
##   Aircraft Model Aircraft Version Landing Count Total Landed Weight
## 1            757              200            83             16434000
## 2            757              300             3               672000
## 3          L1011                0            27              9666000
## 4            777                0             9              4139946
## 5            737              200             5               525000
## 6            737             <NA>            15              1605000
```

Renaming Entries In the Table

So in looking at our tables(We will focus on just landing for this project) I notice that there are column names with spaces. Now you can use `` to surround these names, but some times R just does not like invalid names, so we have to get rid of these spaces using a Regex just like we used a regex to modify the table, only this time I am modifying the table names. I use “.” just because when I am typing these names, it is faster for me to use the . than a _ or - or somthing else like camel case.

```
names(passengers)<-str_replace_all(names(passengers), c(" " = "." , "," = "" ))
```

If you notice the c() modifier. This is just combining the attributes inside into a vector to use when applying the Regex.

```
names(landing)<-str_replace_all(names(landing), c(" " = "." , "," = "" ))
head(landing)
```

```
##   Activity.Period                      Operating.Airline
## 1          200204                           ATA Airlines
## 2          200204                           ATA Airlines
## 3          200204                           ATA Airlines
## 4          200204 Aeroflot Russian International Airlines
## 5          200204                             Air Canada
## 6          200204                             Air Canada
##   Operating.Airline.IATA.Code                      Published.Airline
## 1                          TZ                           ATA Airlines
## 2                          TZ                           ATA Airlines
## 3                          TZ                           ATA Airlines
## 4                       <NA> Aeroflot Russian International Airlines
## 5                          AC                             Air Canada
## 6                          AC                             Air Canada
##   Published.Airline.IATA.Code   GEO.Summary GEO.Region
## 1                          TZ      Domestic         US
## 2                          TZ      Domestic         US
## 3                          TZ      Domestic         US
## 4                       <NA> International     Europe
## 5                          AC International     Canada
## 6                          AC International     Canada
##   Landing.Aircraft.Type Aircraft.Body.Type Aircraft.Manufacturer
## 1             Passenger        Narrow Body                Boeing
## 2             Passenger        Narrow Body                Boeing
## 3             Passenger          Wide Body              Lockheed
## 4             Passenger          Wide Body                Boeing
## 5             Passenger        Narrow Body                Boeing
## 6             Passenger        Narrow Body                Boeing
##   Aircraft.Model Aircraft.Version Landing.Count Total.Landed.Weight
## 1            757              200            83             16434000
## 2            757              300             3               672000
## 3          L1011                0            27              9666000
## 4            777                0             9              4139946
## 5            737              200             5               525000
## 6            737             <NA>            15              1605000
```

3. Exploring the Data

Setting up new data columns

Ok so here is the fun part. If I ever need to create a new column to save myself time from recalculating every time I need to access a certain variable I set up a section in R that I use to efficiently run at the begining so all the columns I could ever want are already made. That Or I create a small new table of the Entities that I would like.

Also take note of the %>% modifier. This is a shortcut to pass data argument into a function and passing that functions return to the next int the flow of functions so that

returnval<-function(arg1 DATA, arg2 …..) endgoal<-function2(arg1 returnval, arg2….)

can be written as endgoal<- DATA%>%function(arg2 …..) %>% function2(arg2,….)

```
landing$Avg.Weight<-landing$Total.Landed.Weight/landing$Landing.Count

landing$activity.date=as.Date(paste0(as.character(landing$Activity.Period), '01'), forma
t='%Y%m%d')

landing$activity.year=format(as.Date(landing$activity.date, format="%d/%m/%Y"),"%Y")

landing$Aircraft=paste(landing$Aircraft.Manufacturer,landing$Aircraft.Model, sep =" ")

landing$GM=paste(landing$Aircraft.Manufacturer,landing$GEO.Summary, sep =" ")

landing=landing%>%filter(activity.year>2004)

head(landing)
```

```
##   Activity.Period Operating.Airline Operating.Airline.IATA.Code
## 1          200507           ABX Air                          GB
## 2          200507           ABX Air                          GB
## 3          200507      ATA Airlines                          TZ
## 4          200507      ATA Airlines                          TZ
## 5          200507        Air Canada                          AC
## 6          200507        Air Canada                          AC
##   Published.Airline Published.Airline.IATA.Code   GEO.Summary GEO.Region
## 1           ABX Air                          GB      Domestic         US
## 2           ABX Air                          GB      Domestic         US
## 3      ATA Airlines                          TZ      Domestic         US
## 4      ATA Airlines                          TZ      Domestic         US
## 5        Air Canada                          AC International     Canada
## 6        Air Canada                          AC International     Canada
##   Landing.Aircraft.Type Aircraft.Body.Type Aircraft.Manufacturer
## 1             Freighter        Narrow Body     McDonnell Douglas
## 2             Freighter        Narrow Body     McDonnell Douglas
## 3             Passenger        Narrow Body                Boeing
## 4             Passenger        Narrow Body                Boeing
## 5             Passenger          Wide Body                Boeing
## 6             Passenger        Narrow Body                Airbus
##   Aircraft.Model Aircraft.Version Landing.Count Total.Landed.Weight
## 1           DC-9               30            40              4066000
## 2           DC-9               41             1               102000
## 3            757              200             2               396000
## 4            757              300           167             37408000
## 5            767              333             1               320000
## 6           A319              114           160             21520000
##   Avg.Weight activity.date activity.year               Aircraft
## 1     101650    2005-07-01          2005 McDonnell Douglas DC-9
## 2     102000    2005-07-01          2005 McDonnell Douglas DC-9
## 3     198000    2005-07-01          2005             Boeing 757
## 4     224000    2005-07-01          2005             Boeing 757
## 5     320000    2005-07-01          2005             Boeing 767
## 6     134500    2005-07-01          2005            Airbus A319
##                          GM
## 1 McDonnell Douglas Domestic
## 2 McDonnell Douglas Domestic
## 3            Boeing Domestic
## 4            Boeing Domestic
## 5       Boeing International
## 6       Airbus International
```

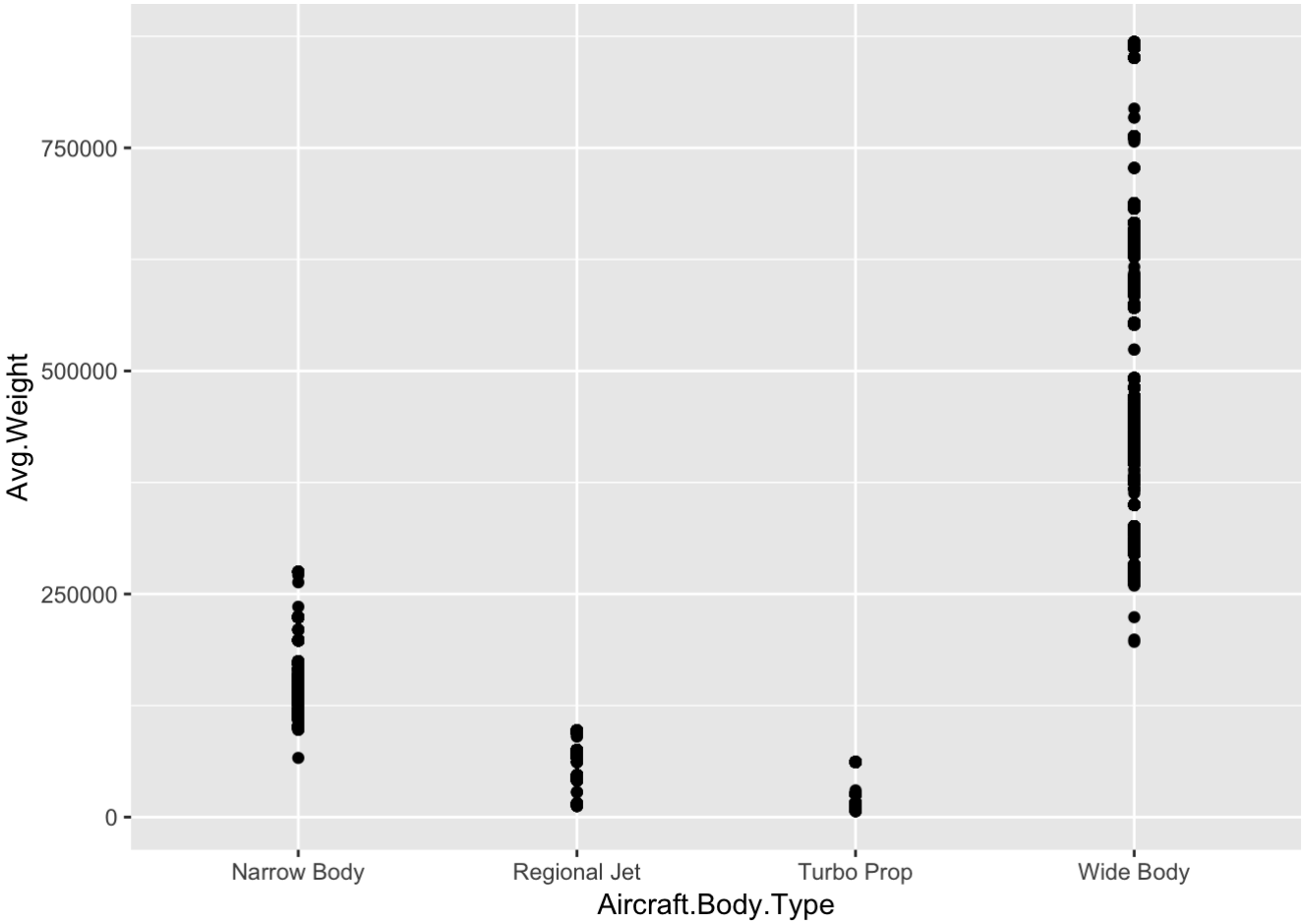
Vizualizing Data Types

Using ggplot2: The Basics

Plotting points using ggplot2 is easy. Using the Pipe to flow the data and the + modifier specified for the ggplot() function, we create an “aesthetic” mapping with aes(x,y,…). This allows us to also change the mapping for future plot types and also different data sets can be used in the same plot.

We can see in this code snippet below:

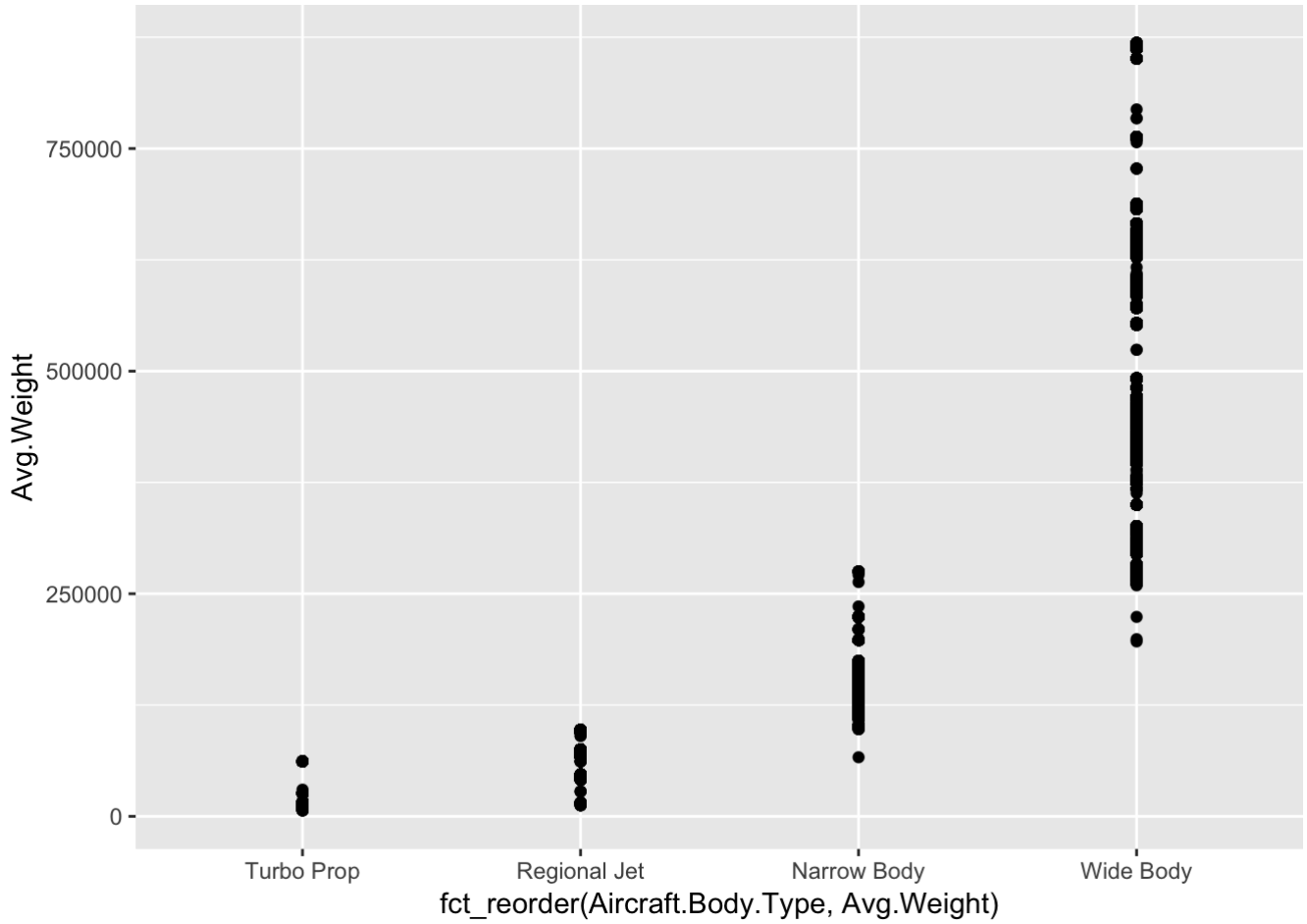
```
#plotting a scatter based on the average weight per body type

landing%>%
  ggplot(aes(x=Aircraft.Body.Type,y=Avg.Weight))+
  geom_point()
```



So there is a visual difference in height. Let's clean this up with the r factor order functions: fct_reorder and fct_inorder. Because the data is categorical and not continuous we can do so using these functions.

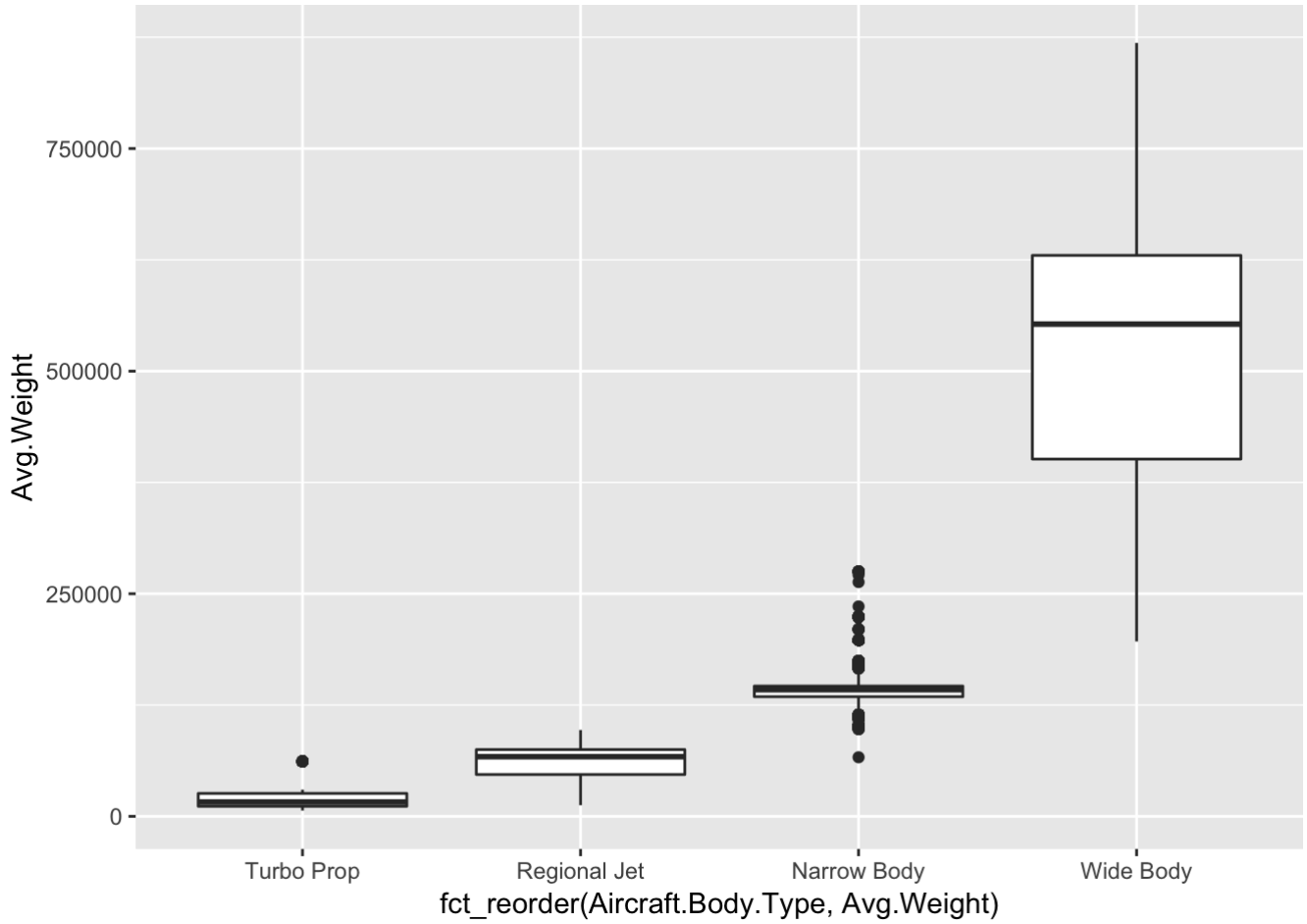
Let's order these categorical data points:

```
#plotting a scatter based on the average weight per body type

landing%>%
  ggplot(aes(x=fct_reorder(Aircraft.Body.Type,Avg.Weight),group=GEO.Summary,y=Avg.Weigh
t))+
  geom_point()
```



Drawing boxplots with ggplot2 is simple. One can just set up the aes like a scatter and then call geom_boxplot.

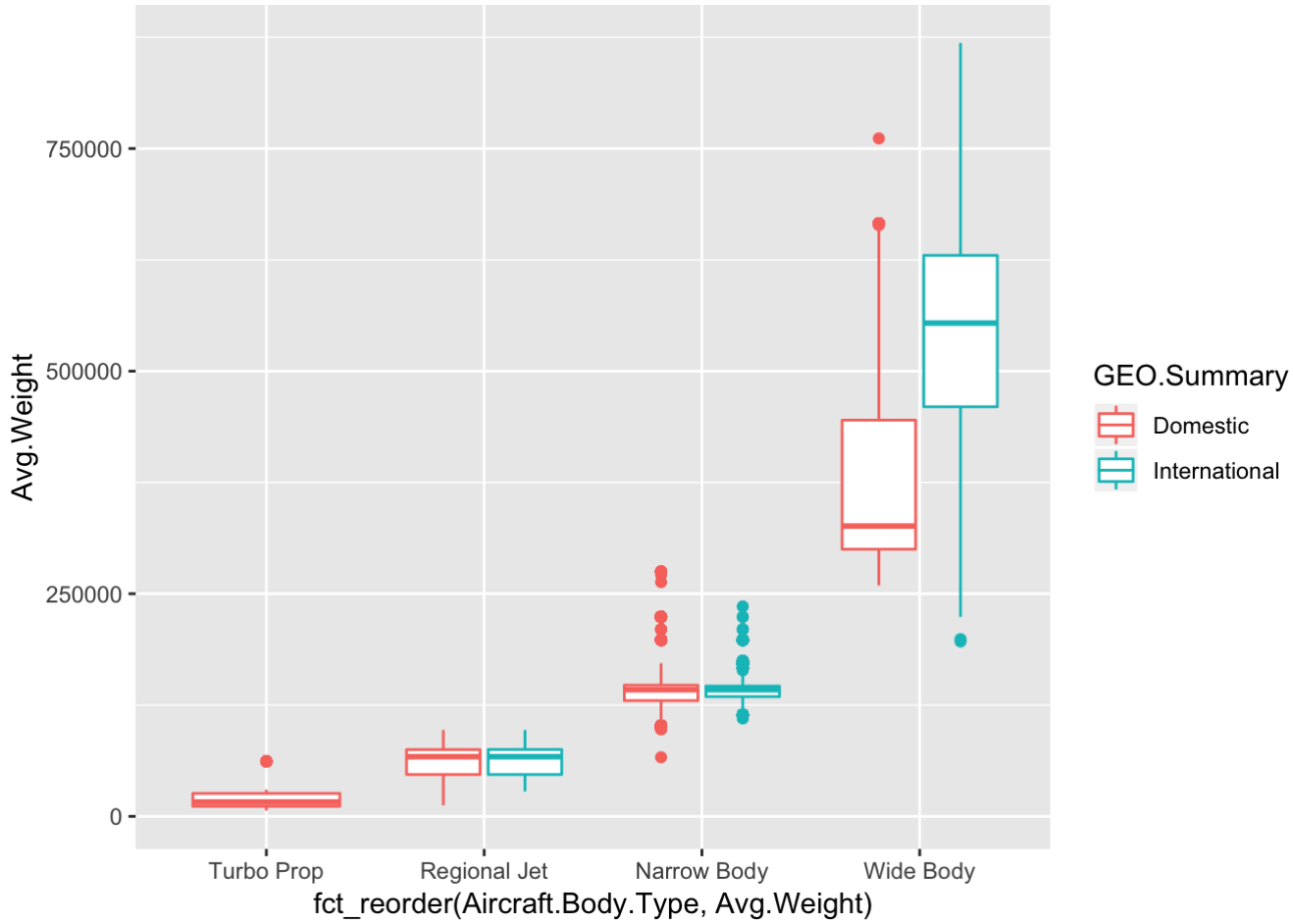
```
#plotting a scatter based on the average weight per body type

landing%>%
  ggplot(aes(x=fct_reorder(Aircraft.Body.Type,Avg.Weight),y=Avg.Weight))+
  geom_boxplot()
```



This is cool but lets say we want to encode a 3 axis in this data set. We have already used the 2D position and 3D visualizations are not the best so the another channel we can use is colour. ggplot2 has a great colouring system already in place to automatically define the scale for either categorical variables or even continuous. If those don suit your fancy you can create your own scale: cof <- colorFactor(c(list of colors), domain=c(the categories of the domain))

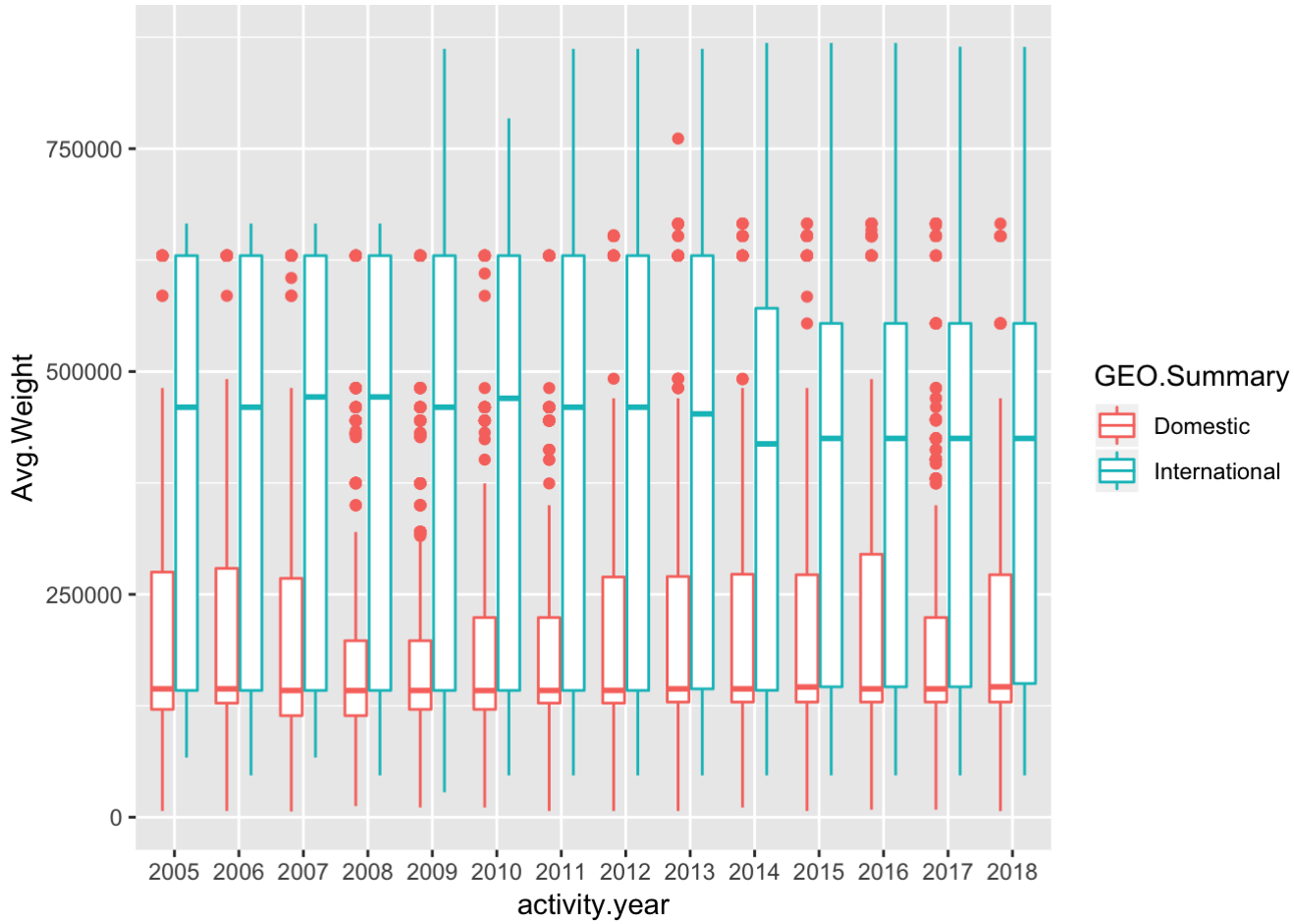
however we do not worry about that here:

```
#plotting a box based on the average weight per body type and then separating the boxes
 by color over Domestic Vs International.

landing%>%
  ggplot(aes(x=fct_reorder(Aircraft.Body.Type,Avg.Weight),y=Avg.Weight,colour=GEO.Summar
y))+
  geom_boxplot()
```



we can do the same over the year. Some times you may need to factor an x variable to create buckets so that multiple boxplots show up one for each “bucket” using the factor(variable to be factored) functions.

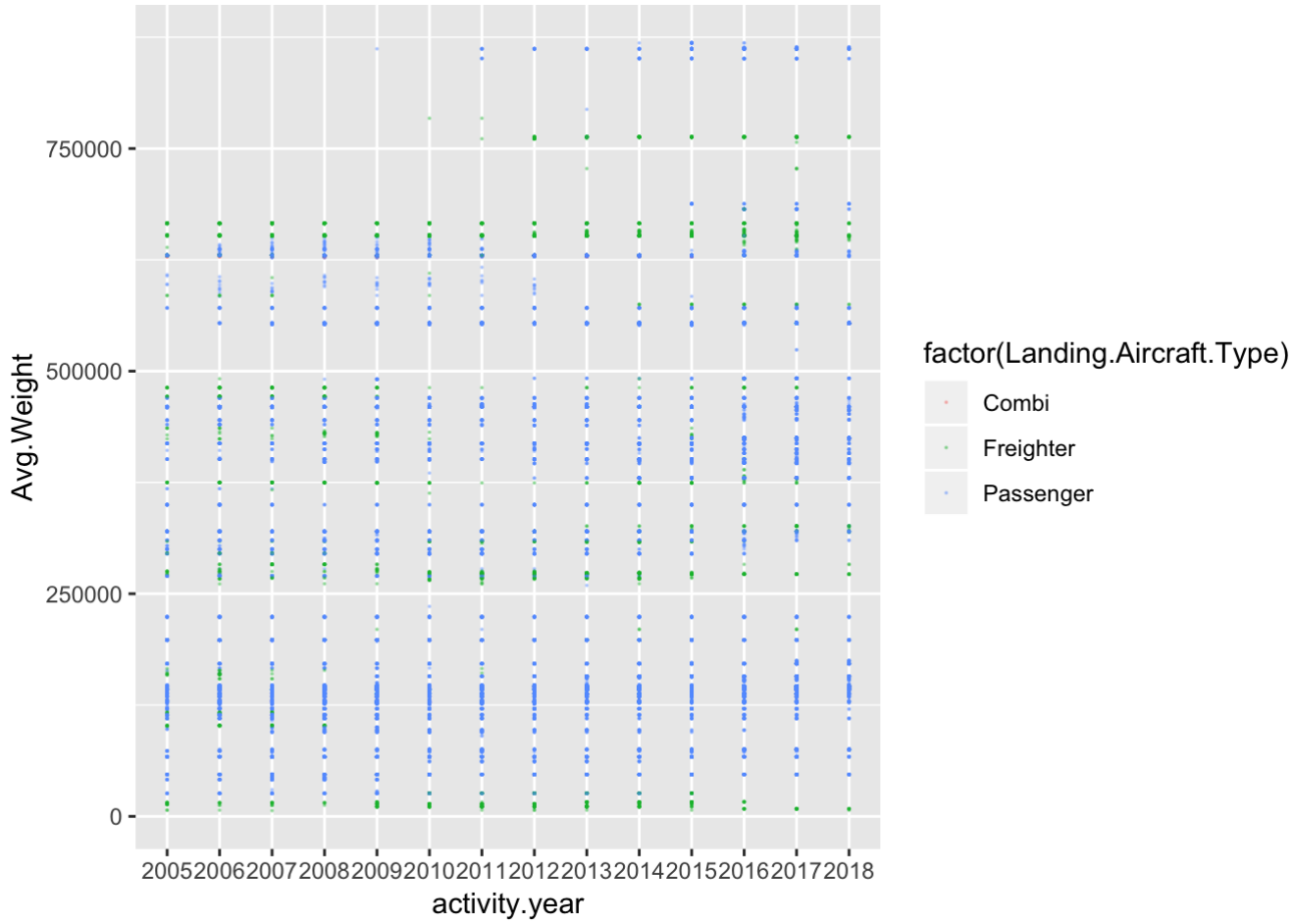
```
#plotting a box based on the average weight per body type
landing%>%
  ggplot(aes(x=activity.year,y=Avg.Weight,colour=GEO.Summary))+
  geom_boxplot()
```



Using ggplot2: More Advanced

changing the dot size for more legibility and adding a fade, and remember when adding different plots to a graph you should add them in the order of the layers they should appear of the map, where the last one is the layer on top and the first is on the bottom:

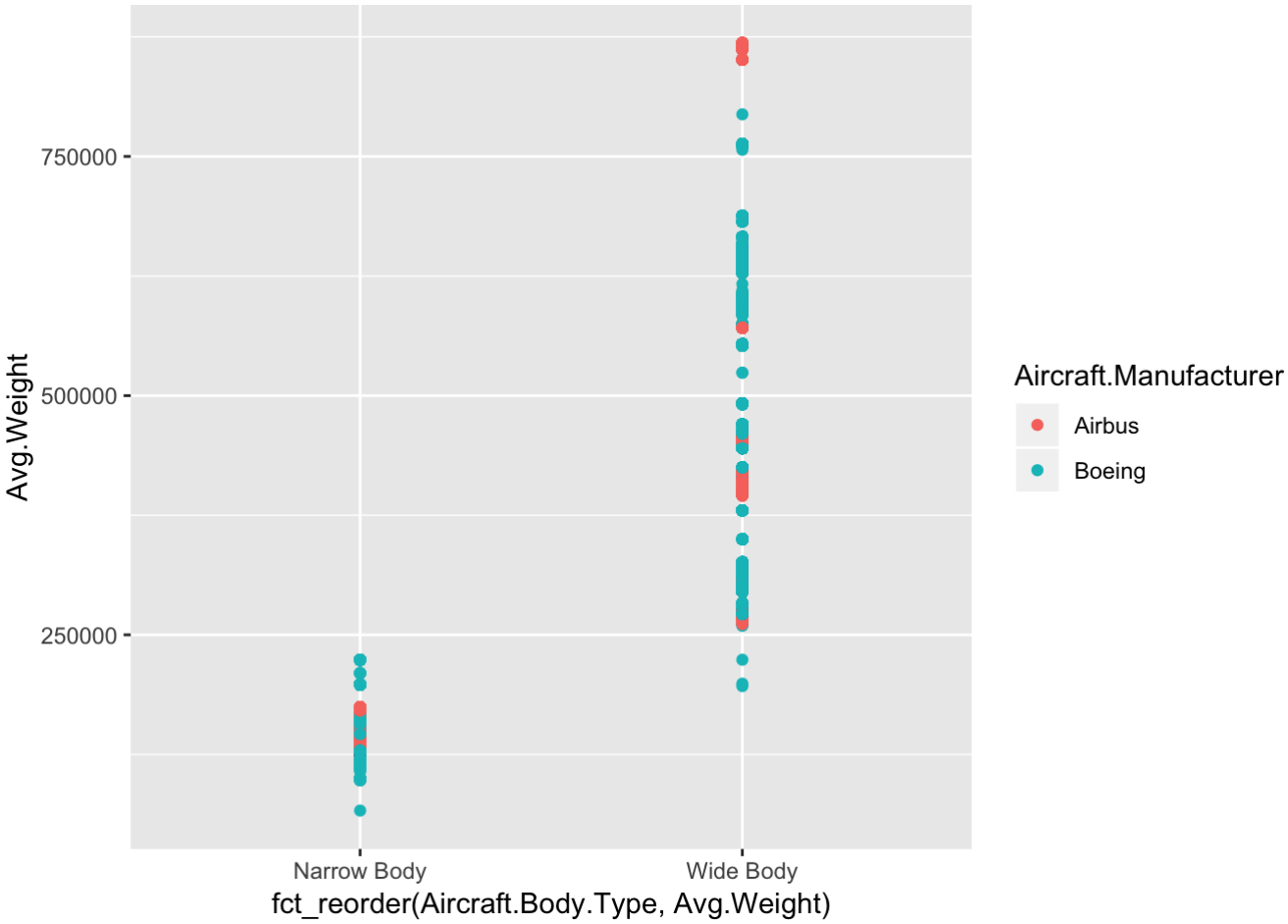
```
#plotting a scatter based on the average weight per body type
landing%>%
  ggplot(aes(x=activity.year,colour=factor(Landing.Aircraft.Type),y=Avg.Weight))+
  geom_point(size=.03,alpha=1/3)
```



Filtering and Adding “Jitter”

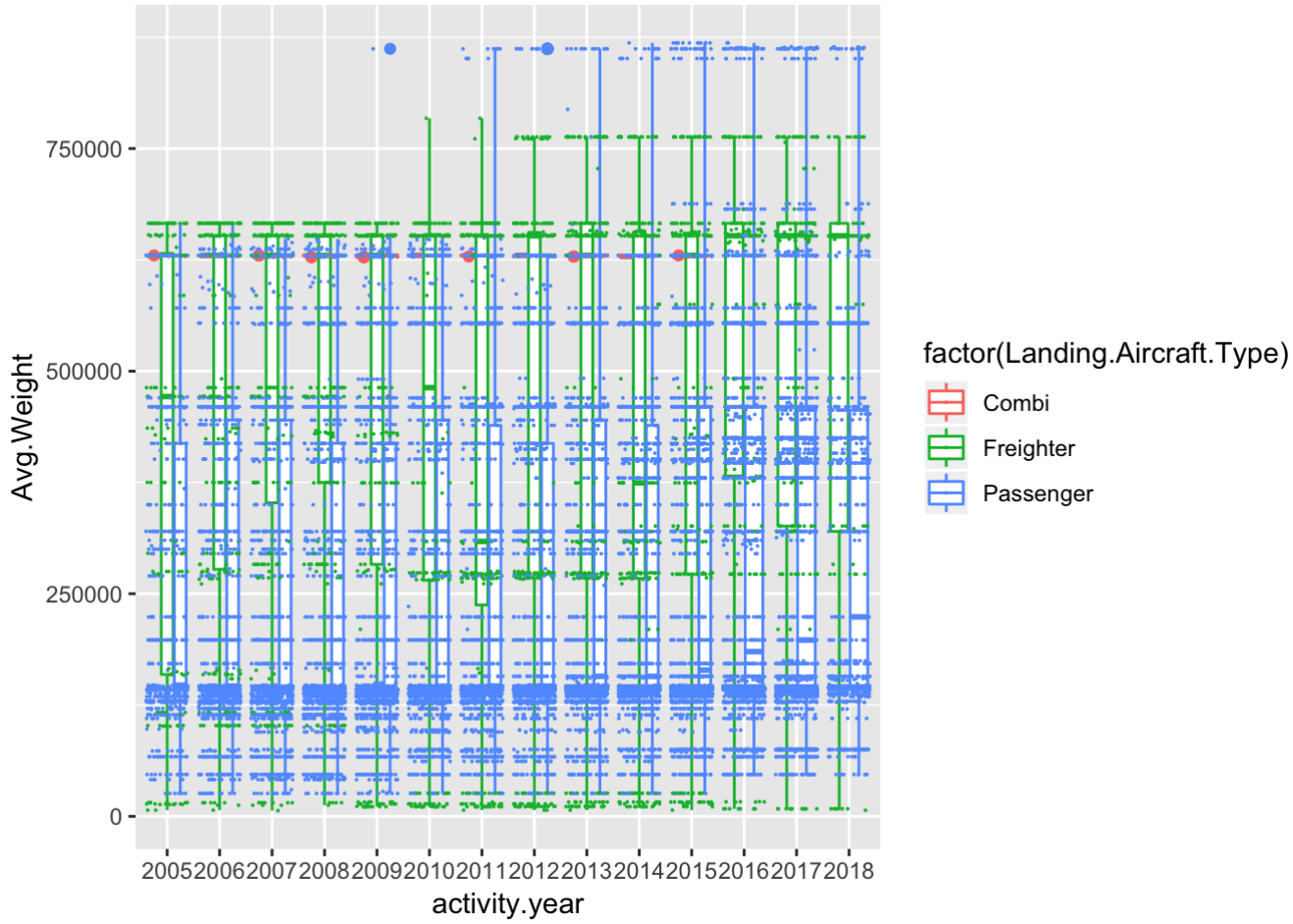
Filtering allows for further analyzing a dataset by allowing one to single out specific relationships. This is done with the Filter functions.

```
#plotting a scatter based on the average weight per body type
landing_select <- landing %>%
   filter(Aircraft.Manufacturer == "Boeing" | Aircraft.Manufacturer == "Airbus")
landing_select%>%
  ggplot(aes(x=fct_reorder(Aircraft.Body.Type,Avg.Weight),y=Avg.Weight,colour=Aircraft.M
anufacturer))+
  geom_point()
```



Adding a Jitter is easy and there are many different specifications for a jitter. Jitters are for specifically trying to show areas of more events or dots so that instead of a straight line of points on the map you have points distributed randomly over a short range of width to unhide the density and in a way adds volume to your mapping.

```
#plotting a jitter based on the average weight per body type
landing%>%
  ggplot(aes(x=activity.year,colour=factor(Landing.Aircraft.Type),y=Avg.Weight))+
  geom_boxplot()+
  geom_jitter(size=.01)
```



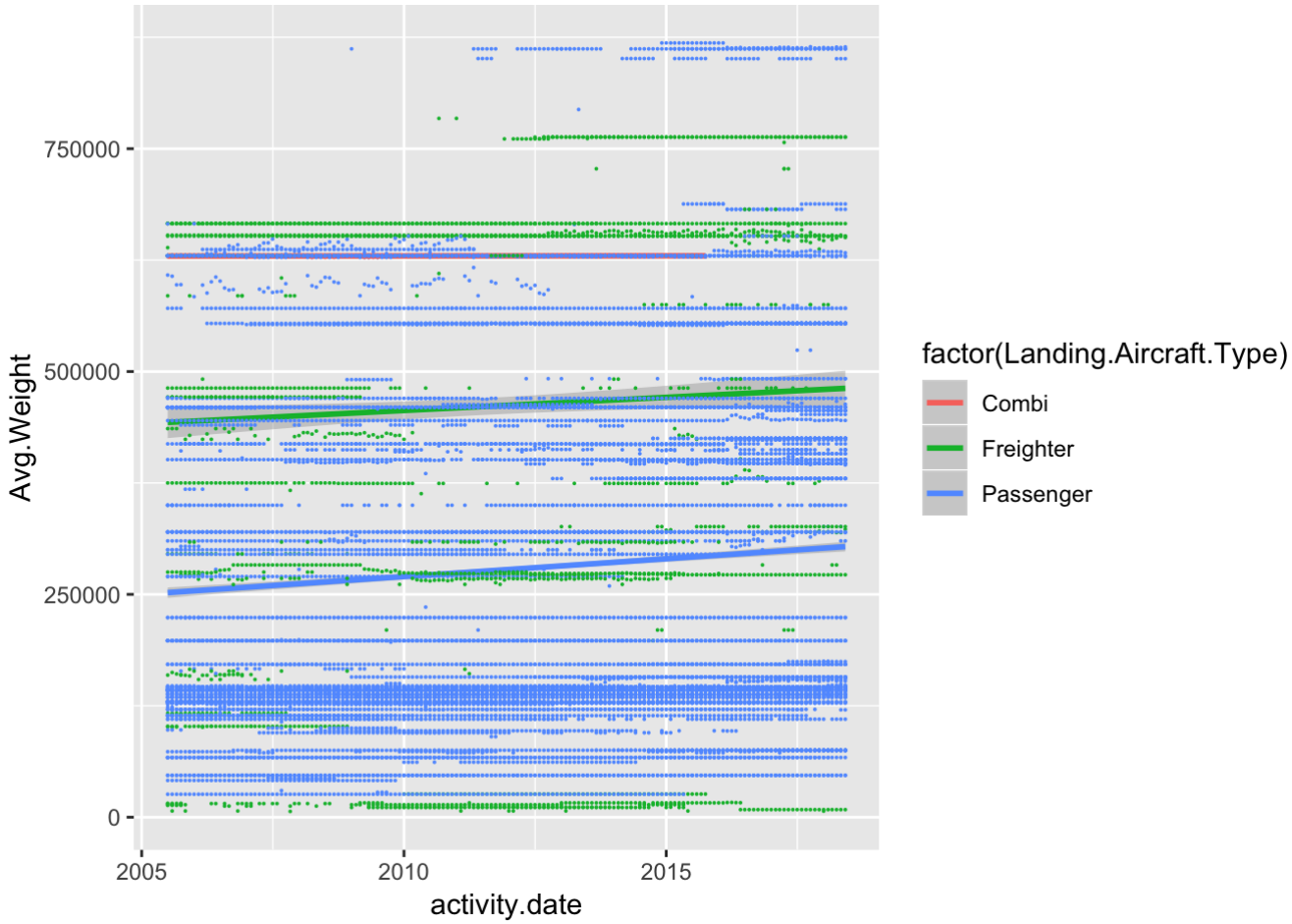
4. Analyzing Your Data Exploration

Something you may want to look at is the story you are building up to.

"Are Planes Landing heavier on average?"

Regressions

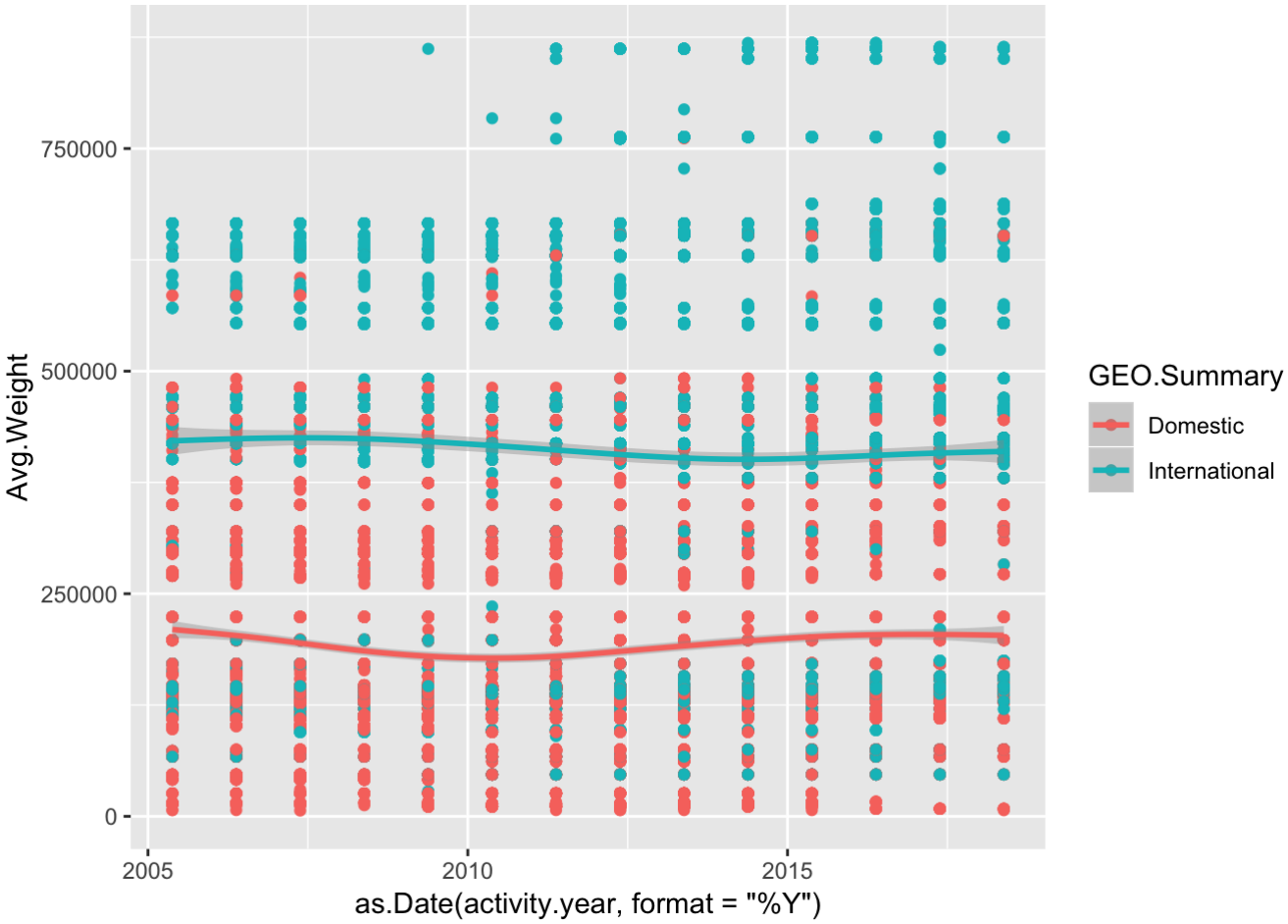
```
#plotting a scatter based on the average weight per body type
landing%>%
  ggplot(aes(x=activity.date,colour=factor(Landing.Aircraft.Type),y=Avg.Weight))+
  geom_smooth(method = "lm")+
  geom_point(size=.01)
```



Well maybe they are, so let draw some lines to find out. Using ggplot2 we can call geom_smooth() to aid in our line drawing skills. Infact it takes over for us. I colour based on whether the plane was domestic or international.

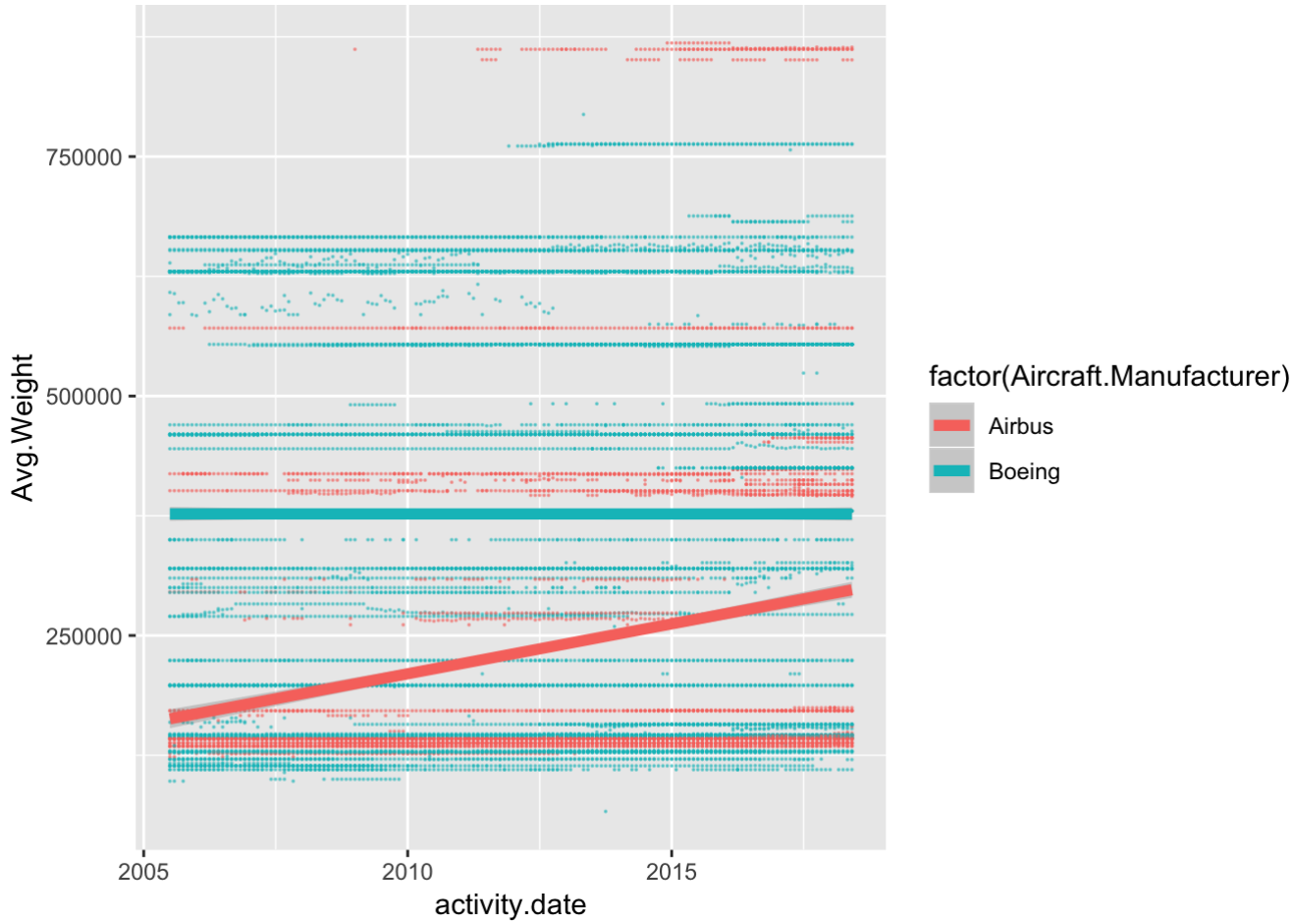
```
#plotting a scatter based on the average weight per body type


landing%>%
  ggplot(aes(x=as.Date(activity.year,format="%Y"),y=Avg.Weight,colour=GEO.Summary))+
  geom_point()+
  geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



To force the type of regression out of the Default Loess Regression, one may specify any of the multiple options. Here I use “lm”. I also make the line a thicker line for visibility purposes and I again add an alpha to try and declutter the numerous points and sort of create a density.

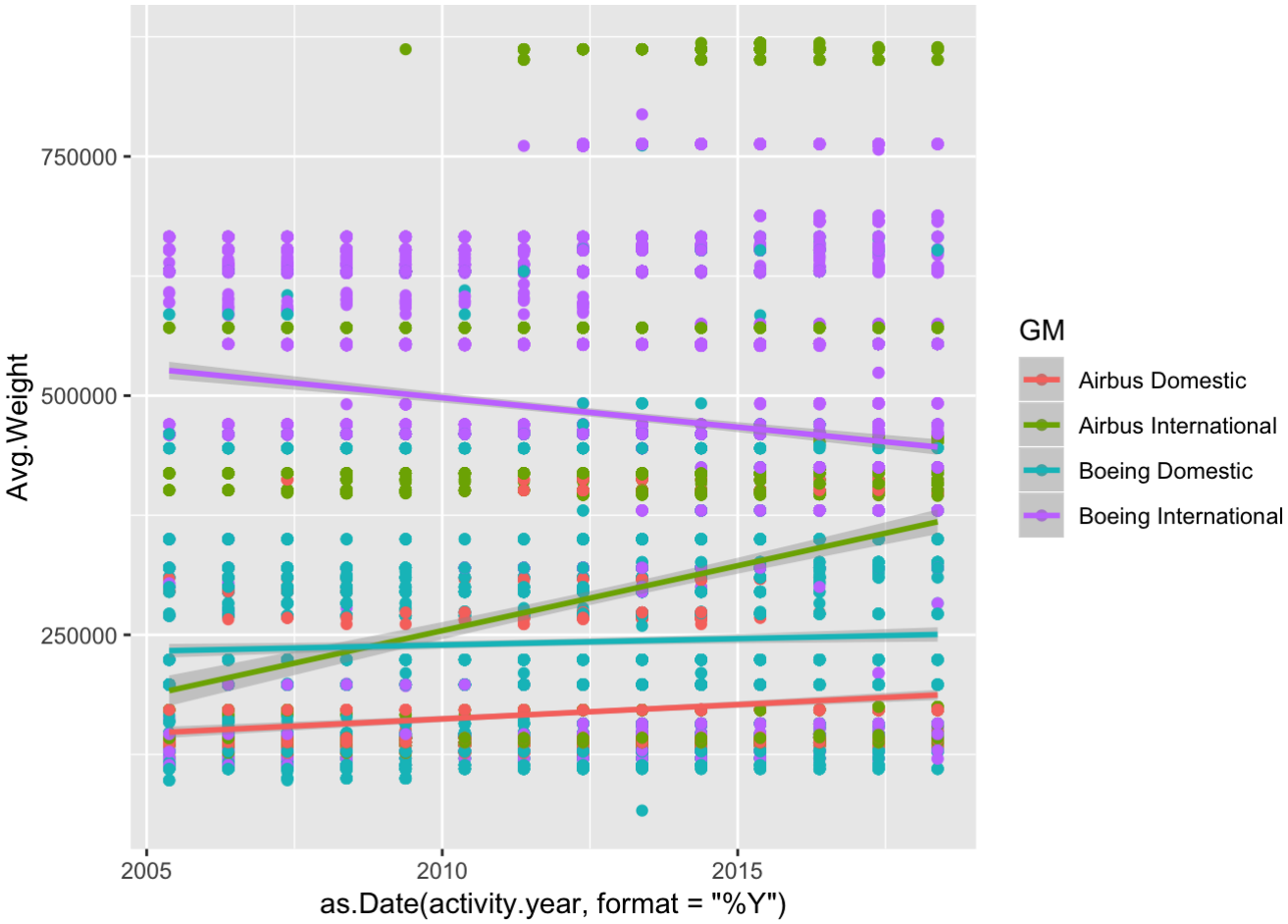
```
#plotting a scatter based on the average weight per body type
landing%>%
  filter(Aircraft.Manufacturer == "Boeing" | Aircraft.Manufacturer == "Airbus")%>%
  filter(activity.year>2004)%>%
  ggplot(aes(x=activity.date,colour=factor(Aircraft.Manufacturer),y=Avg.Weight))+
  geom_point(size=.01,alpha=1/2)+
  geom_smooth(method = "lm",size=2)
```



Boeing vs Airbus yearly

Here I am adding another regression but now only over Airbus and Boeing and splitting the two companies back into domestic and International landings.

```
#plotting a scatter based on the average weight per body type

landing%>%
  filter(Aircraft.Manufacturer == "Boeing" | Aircraft.Manufacturer == "Airbus")%>%
  ggplot(aes(x=as.Date(activity.year,format="%Y"),y=Avg.Weight,colour=GM))+
  geom_point()+
  geom_smooth(method="lm")
```



Plotting a Residual Plot

You want to make sure that your X is numeric, note the year change between fit2 and fit3. Year is a numeric. These equations fit a linear regression to the data plot. To analyze the fit of the plot one can examine the coefficients printed by r:

```
gap_fit2 <- lm(activity.year~Avg.Weight*GEO.Summary*Aircraft.Manufacturer, data=landing_
select)
gap_fit2
```

```
##
## Call:
## lm(formula = activity.year ~ Avg.Weight * GEO.Summary * Aircraft.Manufacturer,
##     data = landing_select)
##
## Coefficients:
##                                                          (Intercept)
##                                                            2.011e+03
##                                                           Avg.Weight
##                                                            7.983e-06
##                                          GEO.SummaryInternational
##                                                            5.913e-01
##                                       Aircraft.ManufacturerBoeing
##                                                            4.984e-01
##                                Avg.Weight:GEO.SummaryInternational
##                                                           -3.554e-06
##                             Avg.Weight:Aircraft.ManufacturerBoeing
##                                                           -7.044e-06
##            GEO.SummaryInternational:Aircraft.ManufacturerBoeing
##                                                            1.562e+00
## Avg.Weight:GEO.SummaryInternational:Aircraft.ManufacturerBoeing
##                                                            1.240e-07
```

```
gap_fit3 <- lm(activity.year~Avg.Weight, data=landing_select)
gap_fit3
```

```
##
## Call:
## lm(formula = activity.year ~ Avg.Weight, data = landing_select)
##
## Coefficients:
## (Intercept)   Avg.Weight
##   2.012e+03    6.151e-07
```

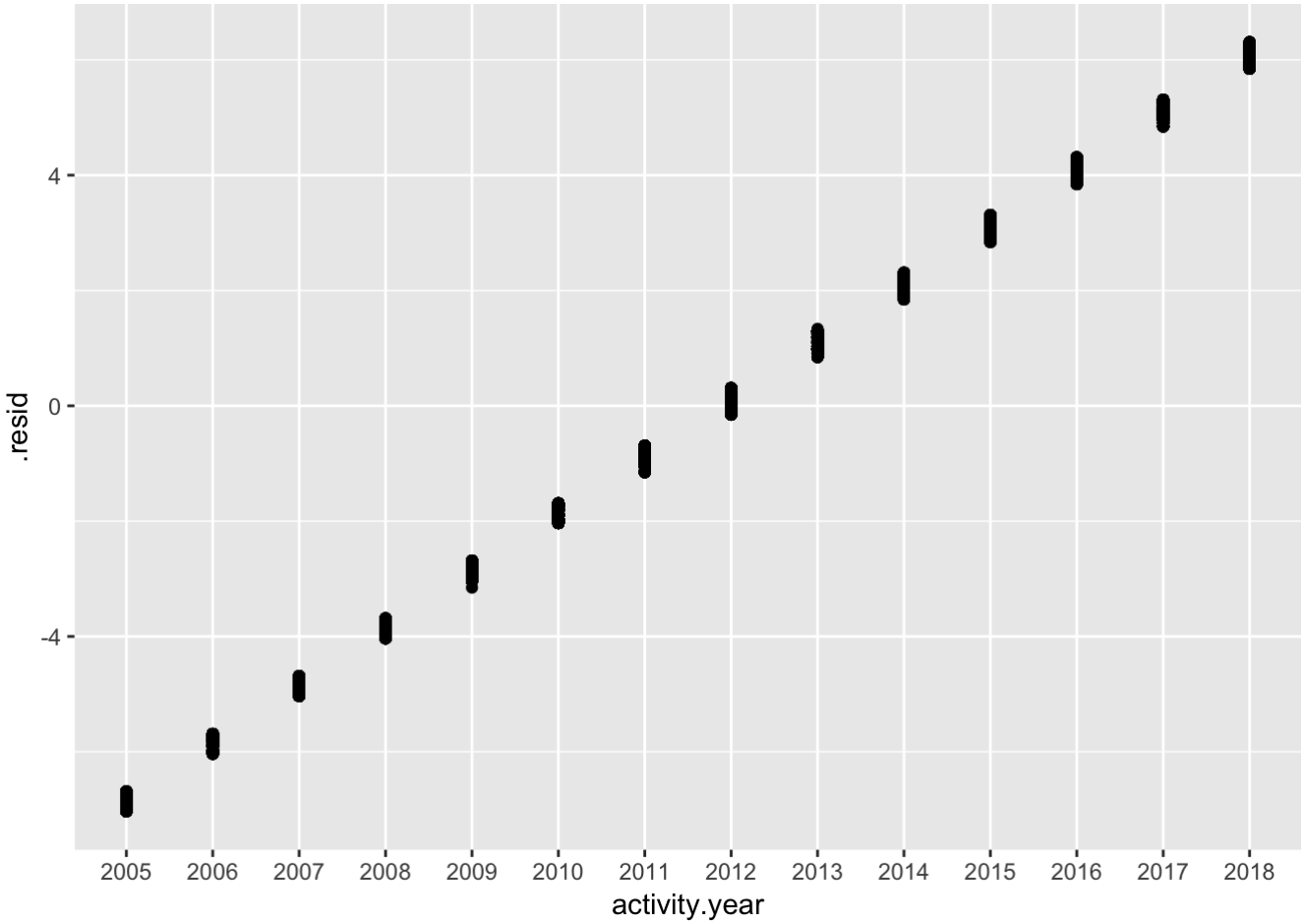
To compare the two linear regressions you may use anova:

```
anova(gap_fit2,gap_fit3)
```

```
## Analysis of Variance Table
##
## Model 1: activity.year ~ Avg.Weight * GEO.Summary * Aircraft.Manufacturer
## Model 2: activity.year ~ Avg.Weight
##   Res.Df    RSS Df Sum of Sq     F    Pr(>F)
## 1  18468 260262
## 2  18474 268206 -6     -7944 93.95 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

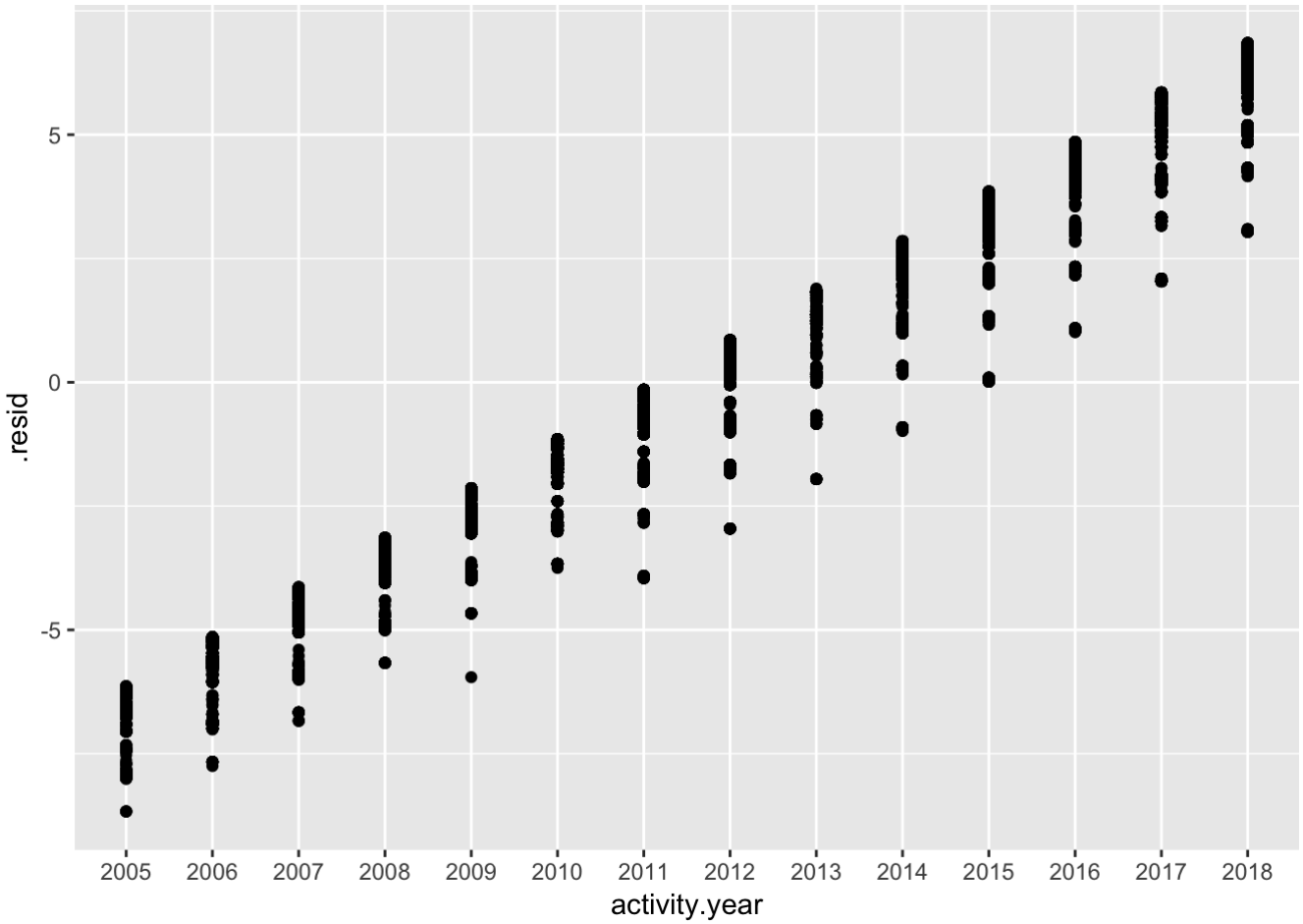
And to utilize these linear regressions with ggplot, you can use .fitted for a fitted x or any other relative x variable, (I chose year as numeric), and to access such data like residuals of the linear regression is simple, just use ".resid".

```
ggplot(gap_fit3) +
  geom_point(aes(x=activity.year, y=.resid))
```



And again.

```
ggplot(gap_fit2) +
  geom_point(aes(x=activity.year, y=.resid))
```


5
0
-5
.resid
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
activity.year