

Advanced Dogfood Eating

Interactive graphics from R with Shiny and GoogleVis

Josh Laurito

To Dos

- Announcements
- Data visualization in the feedback loop
- Two modules ago homework
- Cover last module's homework
- Moving to interactive graphics
- googleVis
- shiny
- Next module's reading
- Next module's assignment

Announcemments

- We will try to have an NYC meetup: <http://whenisgood.net/e33cqb4>
- Sorry for the slow grading

Two modules ago homework

- Generally very good!
- Common comments:
- 'Better Variable Naming'
- 'Break up lines'
- 'Try to show variability'

Last module's homework

- Thank you all for being great about installing software early
- How did you combine files? I used `csvkit` <http://csvkit.readthedocs.org/>
- load what we need

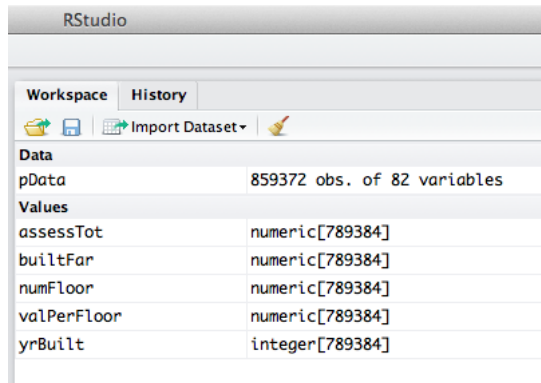
```
library("ggplot2")  
library("plyr")  
library("bigvis")  
  
pData <- read.csv("/Users/josh.laurito/personal/cuny/all_PLUTO_data.csv")
```

Last week's homework - data cleaning

```
builtFar <- pData$BuiltFAR[pData$YearBuilt > 1850 & pData$NumFloors != 0 ]  
  
numFloor <- pData$NumFloors[pData$YearBuilt > 1850 & pData$NumFloors != 0 ]  
  
yrBuilt  <- pData$YearBuilt[pData$YearBuilt > 1850 & pData$NumFloors != 0 ]  
  
assessTot <- pData$AssessTot[pData$YearBuilt > 1850 & pData$NumFloors != 0]  
  
valPerFloor <- assessTot/numFloor
```

Last week's homework - lots of data

- You probably noticed that there was a lot of data
- Slows down analysis: viz becomes more important and harder to do



The screenshot shows the RStudio interface. The top bar indicates the title 'RStudio'. Below it, there are tabs for 'Workspace' and 'History'. The 'Workspace' tab is active, showing a table with the following data:

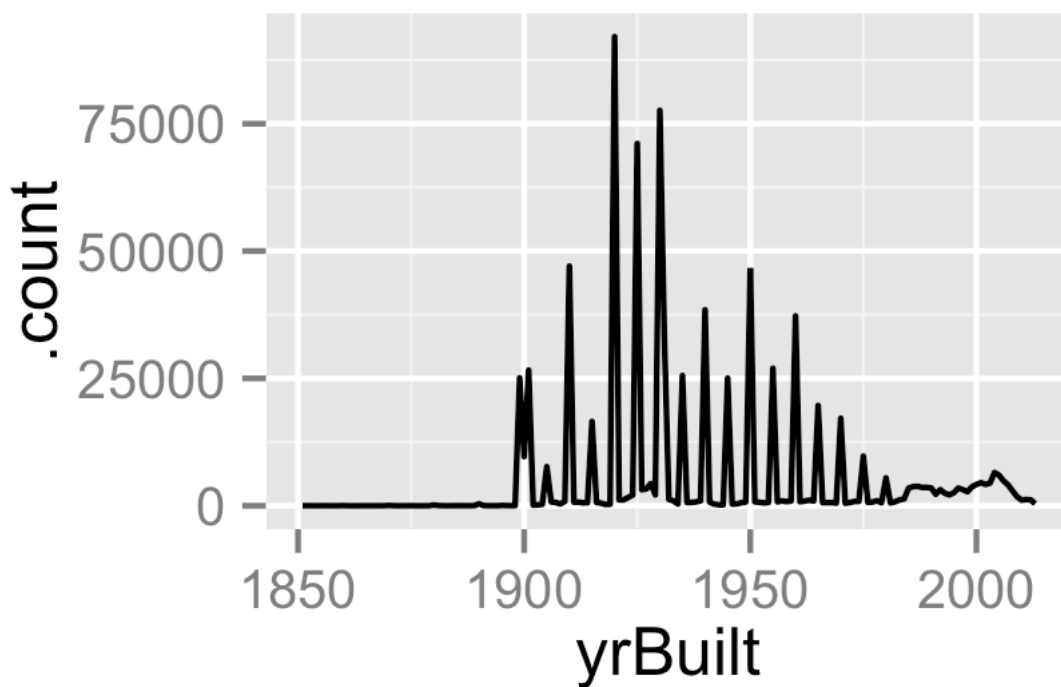
Data	
pData	859372 obs. of 82 variables
Values	
assessTot	numeric[789384]
builtFar	numeric[789384]
numFloor	numeric[789384]
valPerFloor	numeric[789384]
yrBuilt	integer[789384]

Last week's homework - building 'cut-off date'

- Poorly specified problem
- General concept: what does it mean to be an 'old building'
- We should start by checking when buildings were built

```
summary(yrBuilt)
yr <- condense(bin(yrBuilt, 1))
autoplot(yr)
ggsave('assets/img/yrBuilt.png', height=2, width = 3)
```


Last week's homework - building 'cut-off date'

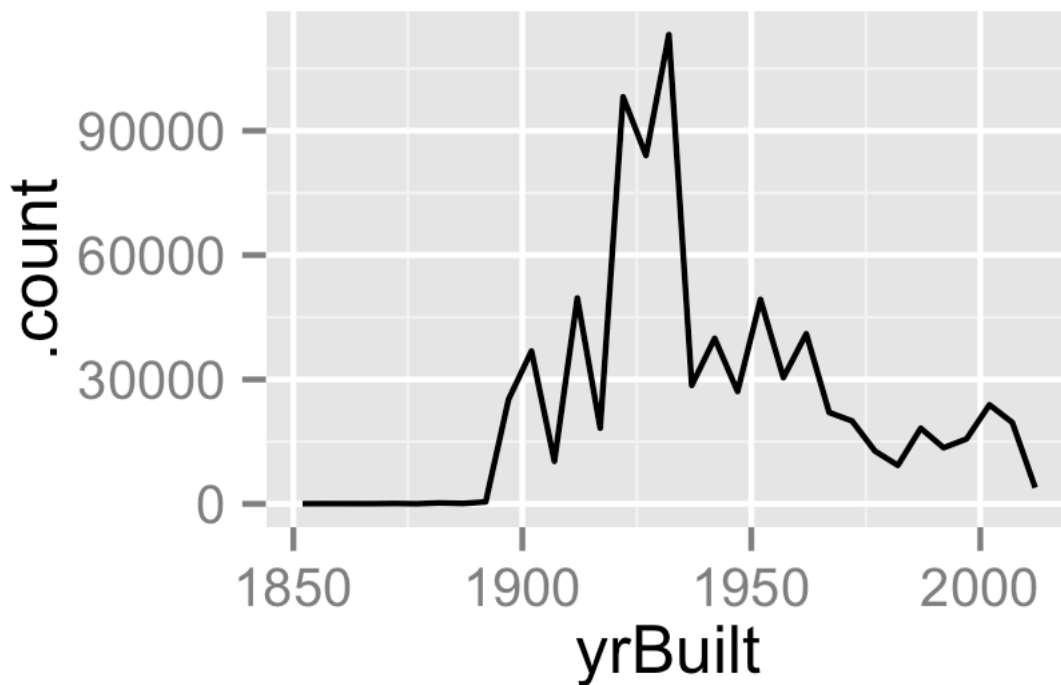


Last week's homework - building 'cut-off date'

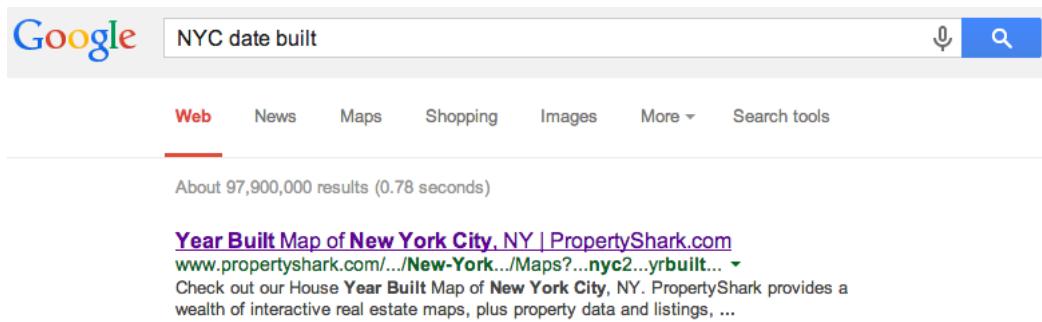
- Oh no! Our data stinks!
- Clearly estimated when year \leq 1980
- Can we proceed? Depends on context of questions

```
yr <- condense(bin(yrBuilt, 5))  
autoplot(yr)  
ggsave('assets/img/yrBuilt5.png', height=2, width = 3)
```

Last week's homework - building 'cut-off date'

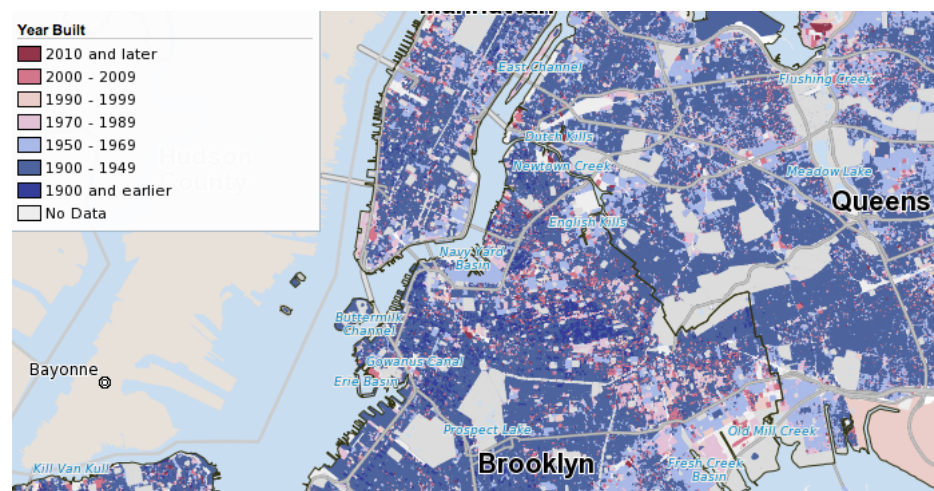


Last week's homework - double check suspicious data



A screenshot of a Google search interface. The search bar contains the text "NYC date built". Below the search bar, the "Web" tab is selected and highlighted with a red underline. Other tabs include "News", "Maps", "Shopping", "Images", "More", and "Search tools". Below the tabs, it says "About 97,900,000 results (0.78 seconds)". The first search result is titled "Year Built Map of New York City, NY | PropertyShark.com" and includes the URL "www.propertyshark.com/.../New-York.../Maps?...nyc2...yrbuilt...". The description below the link reads: "Check out our House Year Built Map of New York City, NY. PropertyShark provides a wealth of interactive real estate maps, plus property data and listings, ...".

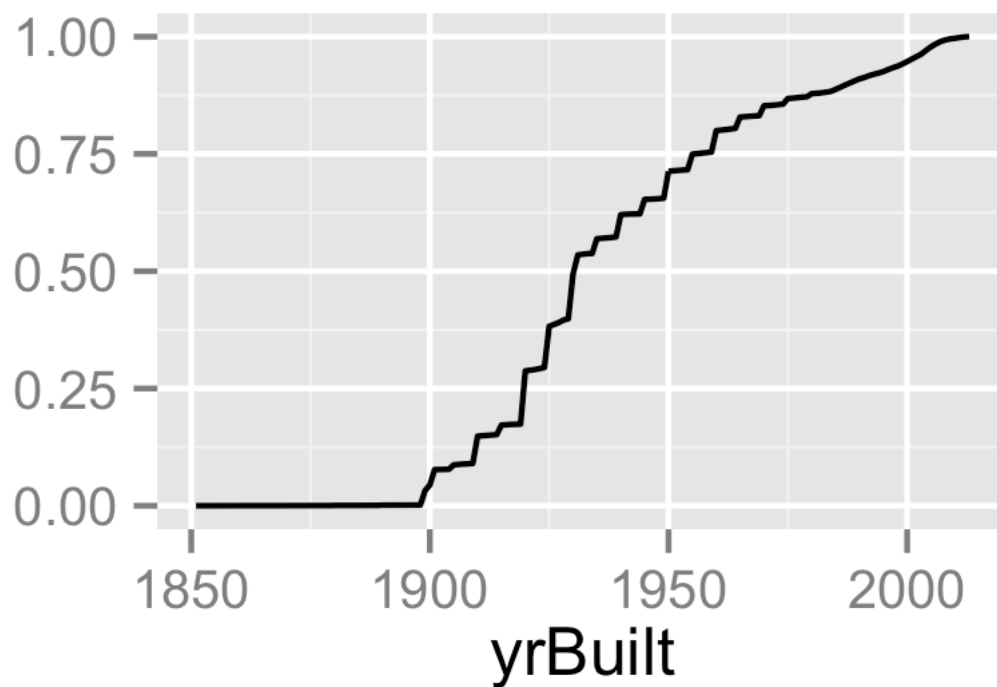
Last week's homework - double check suspicious data



Last week's homework - finding cut-offs

```
yr2 <- condense(bin(yrBuilt, 1))
total <- sum(yr2$.count)
yr2$perc_built <- cumsum(yr2$.count)/total
ggplot(yr2, aes(x= yrBuilt, y=perc_built)) + geom_line() +ylab('')
ggsave('assets/img/cumcurve.png', height=2, width = 3, dpi = 100)
```

Last week's homework - finding cut-offs

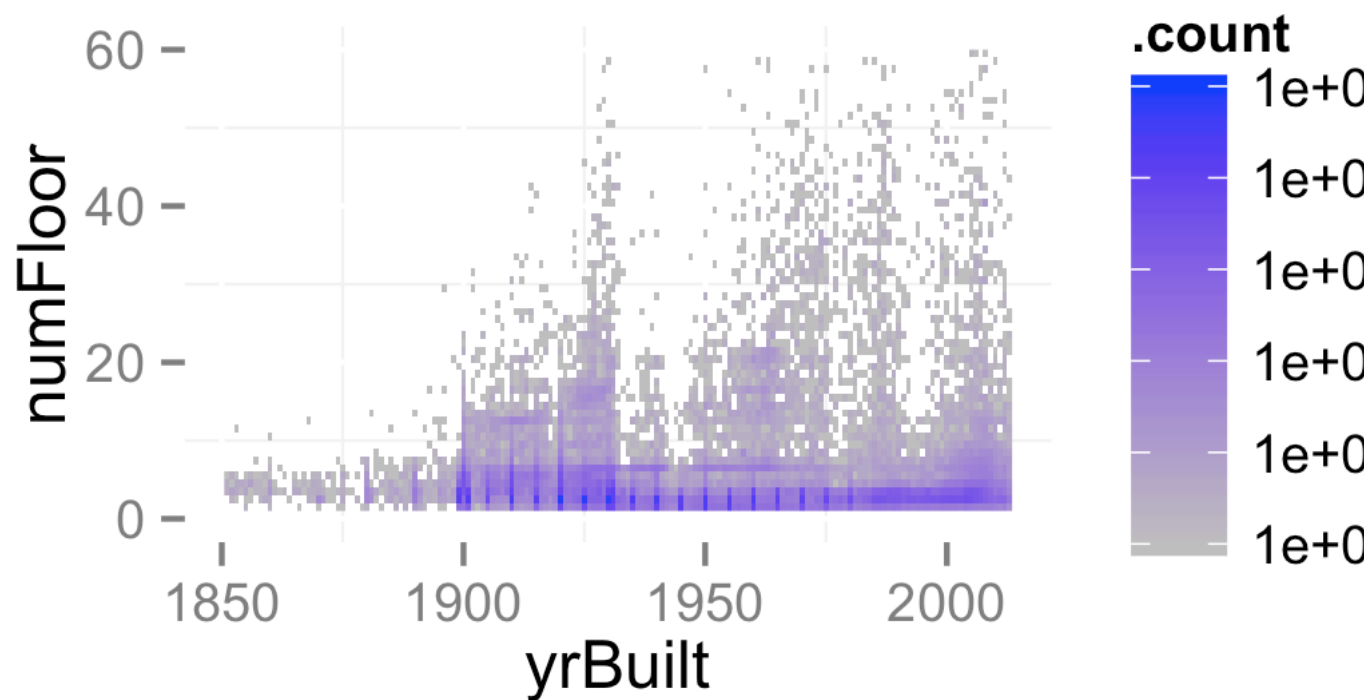


Last week's homework - cut-offs by group

- Similar question to previous, only with groups

```
flrVsYr <- condense(bin(yrBuilt, 1), bin(numFloor, 1))
p <- autoplot(flRvsYr) + theme(panel.background=element_rect(fill='white')) + ylim(0,60)
p + scale_fill_gradient(limits= c(1,100000),
                        low='grey',
                        high='blue',
                        trans="log",
                        breaks=myBreaks)
ggsave('assets/img/yr_v_flr.png', height=2, width = 4, dpi = 300)
```

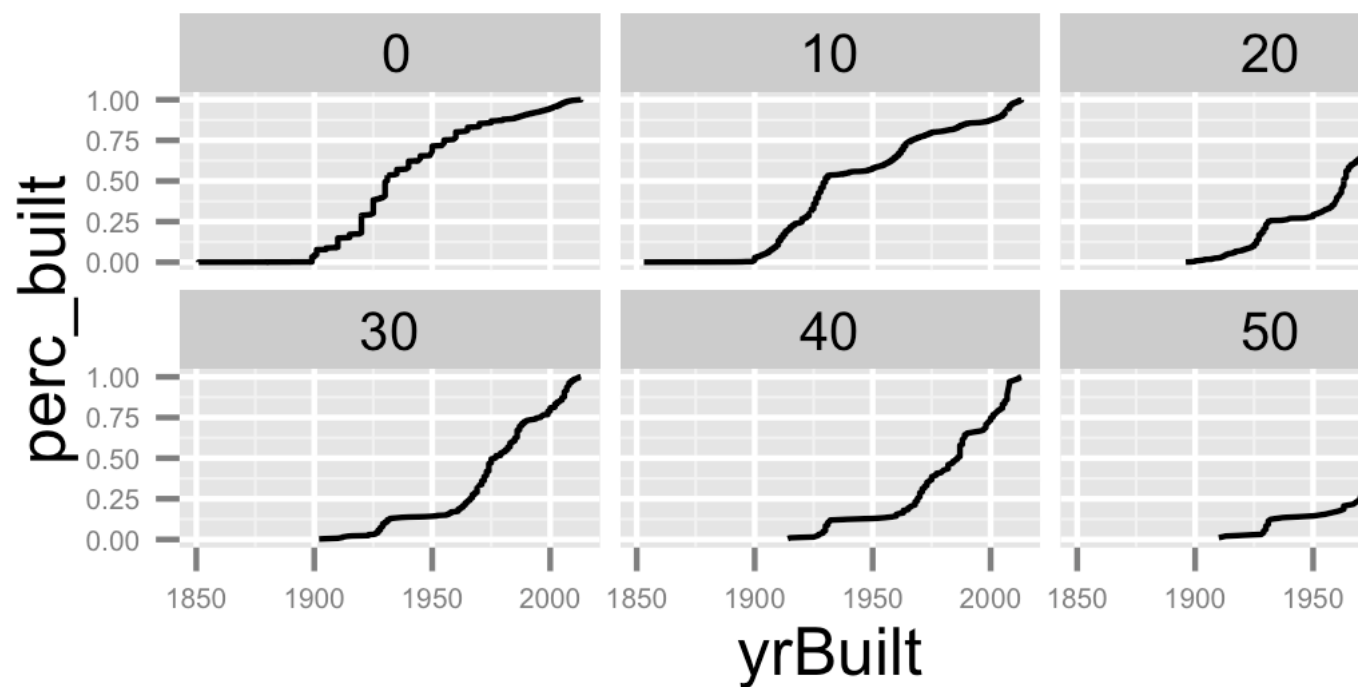

Last week's homework - cut-offs by group



Last week's homework - cut-offs by group

```
flrVsYr$stories <- 10*trunc(flrVsYr$numFloor/10)
flrVsYr$stories <- sapply(flrVsYr$stories, min, 50)
flrVsYr$num_built <- ave(flrVsYr$count, flrVsYr$stories, FUN=cumsum)
flrVsYr$perc_built <- flrVsYr$num_built / ave(flrVsYr$count, flrVsYr$stories, FUN=sum)
ggplot(flrVsYr, aes(x=yrBuilt, y=perc_built, group=stories)) + geom_line() +
facet_wrap(~ stories) + theme(axis.text=element_text(size=5))
ggsave('assets/img/stories.png', height=2, width = 4, dpi = 300)
```

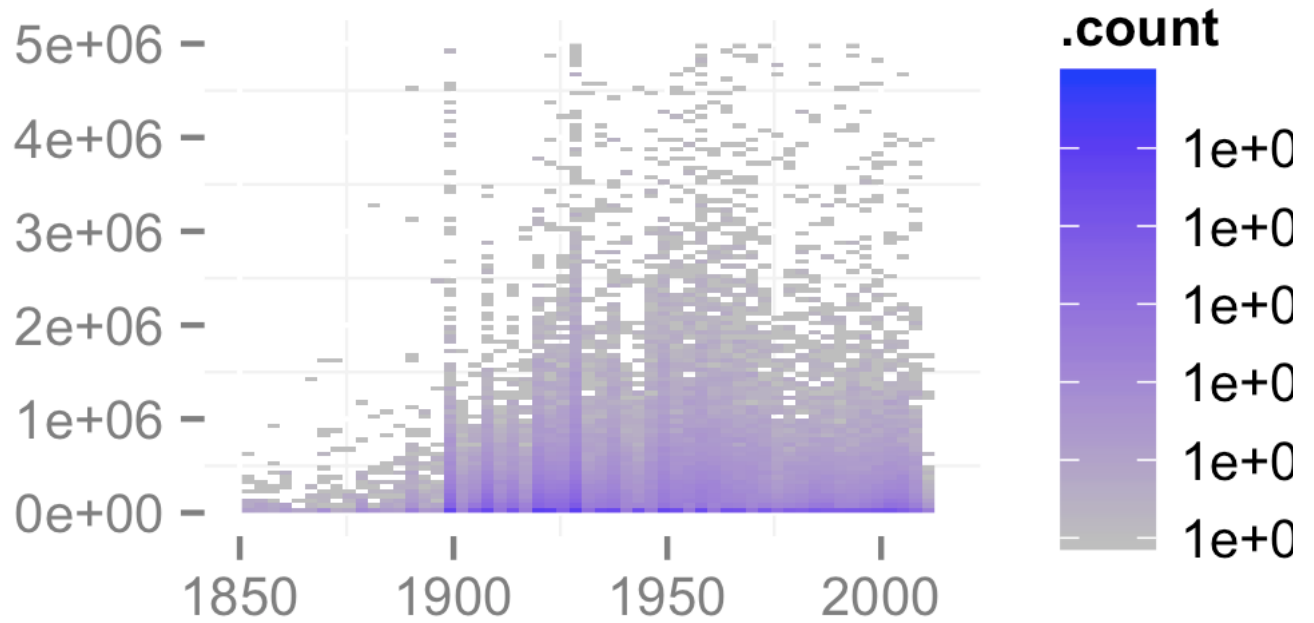
Last week's homework - cut-offs by group



Last week's homework - cut-offs by group

```
assessVsYr <- condense(bin(yrBuilt[assessTot < 5000000],3), bin(valPerFloor[assessTot < 5000000],3))
p <- autoplot(assessVsYr) + theme(panel.background=element_rect(fill='white')) + xlim(1850,2015)
p + scale_fill_gradient(limits= c(1,1000000),
                        low='grey',
                        high='blue',
                        trans="log",
                        breaks=myBreaks)
ggsave('assets/img/assess.png', height=2, width = 4, dpi = 300)
```

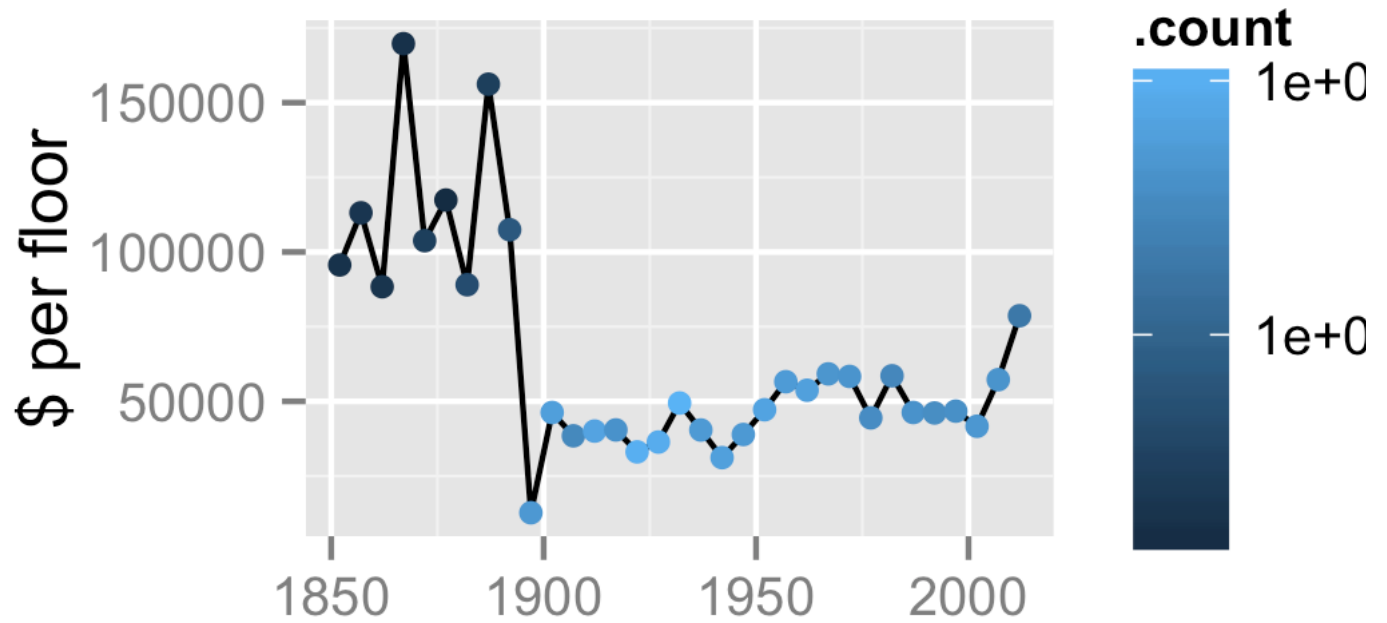
Last week's homework - cut-offs by group



Last week's homework - wartime

```
flrVal <- condense(bin(yrBuilt[assessTot < 5000000],5), z =valPerFloor[assessTot < 5000000] )  
autoplot(flVal) +xlab('') + ylab('$ per floor')  
ggsave('assets/img/flrVal.png', height=2, width = 4, dpi = 300)
```

Last week's homework - cut-offs by group



Moving to interactive graphics

- Painful to create multiple different charts for investigation
- We are moving away from statistics here and towards design
- `ggplot2` is doing this too- moving to `ggvis`

Moving to interactive graphics - terms

- 'Server-side': what happens on the server (back-end, database)
- 'Client-side': what happens on the user's computer (Browser, JS)
- Scalable: how a site handles many visits at once

googleVis

- Often known as 'Hans Robling style charts'
- Interface to Google Chart API
- Compiles to HTML/Javascript
- Great demo by running `demo('googleVis')`

googleVis - advantages

- Fast and Easy
- Output is interactive and in-browser
- Very simple to combine charts

googleVis - drawbacks

- Warning: not all google charts are secure
- Need web access to run
- Risky: Google has history of deprecating projects

shiny

- Sponsored by RStudio: similar to googleVis but has a server-side component
- Can be integrated with googleVis
- Open Source

Next module's reading

- All on Blackboard

Next module's assignment

- CDC Wonder Data <http://wonder.cdc.gov/controller/datarequest/D76>