06 - Intro to graphics (with ggplot2) (part 2)

ST 597 | Spring 2017
University of Alabama

06-dataviz2.pdf

Contents

Required Packages and Data

```
library(tidyverse)
library(gcookbook)
library(Lahman)    # may need to: install.packages("Lahman")
```

1 Cleveland Dot Plot

William Cleveland wrote a popular book on visualizing data The Elements of Graphing Data that has many useful suggestions. One element he stressed was to reduce the cognitive strain on the view. One way to do this is to use as little ink as possible. The Cleveland dot plot contains the same information as a bar graph, but instead of using all the ink needed for the bar, remove the bar altogether and place a dot at the bar height (using `geom_point()`).

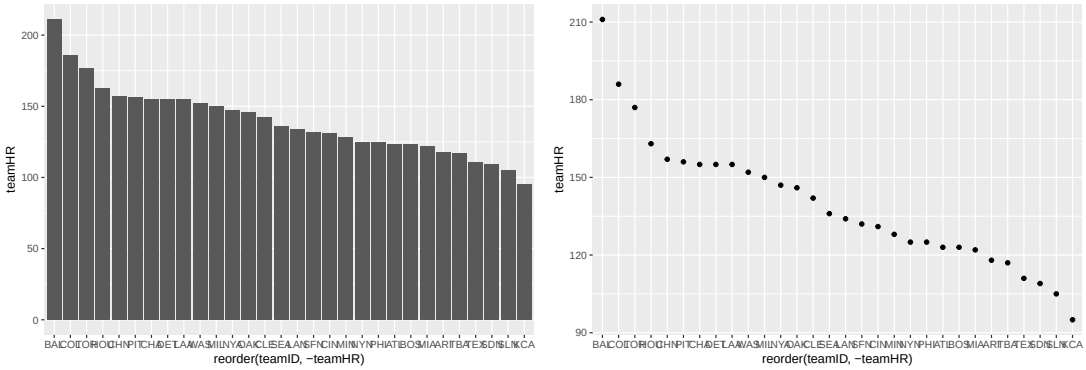
Consider the baseball data `Batting` from the `Lahman` package. We want to display the number of home runs for each team during the 2014 season.

```
library(Lahman)      # load the Lahman package
data(Batting)        # load the Batting data
H = Batting %>%
  filter(yearID == 2014) %>%
  group_by(teamID) %>%
  summarize(teamHR = sum(HR), teamBA=sum(H)/sum(AB), teamR = sum(R))
glimpse(H)
#> Observations: 30
#> Variables: 4
#> $ teamID <fctr> ARI, ATL, BAL, BOS, CHA, CHN, CIN, CLE, COL, DET, HOU,...
#> $ teamHR <int> 118, 123, 211, 123, 155, 157, 131, 142, 186, 155, 163, ...
#> $ teamBA <dbl> 0.2484, 0.2407, 0.2563, 0.2441, 0.2526, 0.2387, 0.2376,...
#> $ teamR  <int> 615, 573, 705, 634, 660, 614, 595, 669, 755, 757, 629, ...
```

I added team batting average (`teamBA`) and runs (`teamR`) to dress up our plot.

Compare the bar graph with the dot plot.

```
#- (left) bar graph
ggplot(H) + geom_col(aes(x=reorder(teamID, -teamHR), y=teamHR))

#- (right) corresponding dot plot
ggplot(H) + geom_point(aes(x=reorder(teamID, -teamHR), y=teamHR))
```



1.1 Your Turn

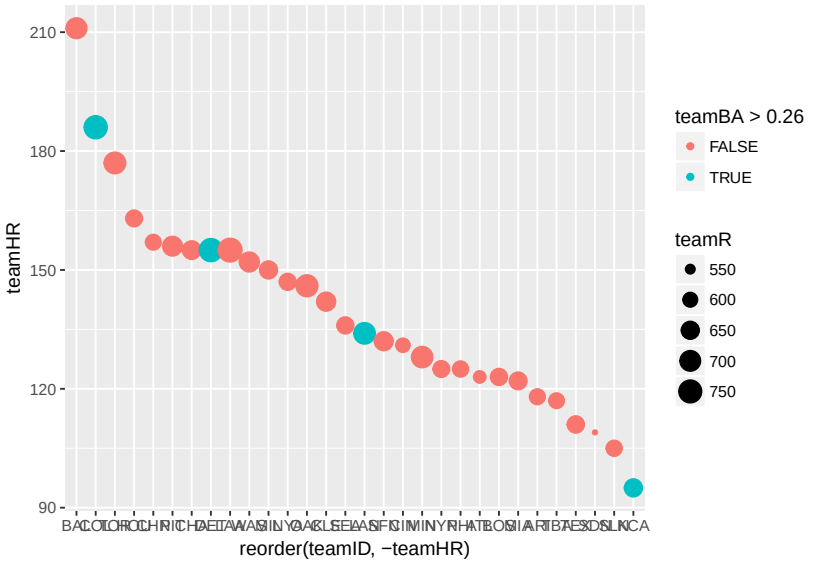
Your Turn #1 : Dot Plot vs. Bar Plot

1. What are the differences between the two plots?
2. What aspects can be improved with the dot plot?

1.2 Cleveland Dot Plot Aesthetics

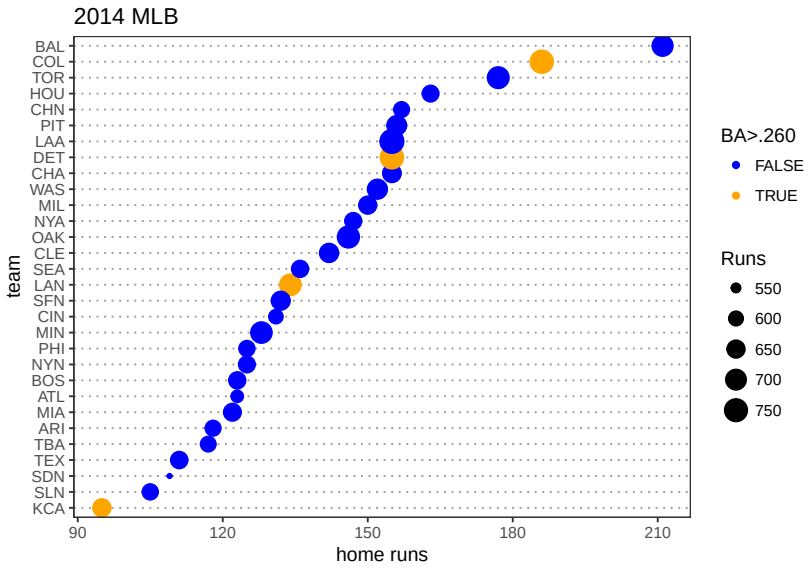
The real strength is in adding additional aesthetics, like size and color

```
ggplot(H) + geom_point(aes(x=reorder(teamID, -teamHR), y=teamHR,
                           size=teamR, color=teamBA>.260))
```



Final touch include putting team on the y-axis, changing the *theme*, and adding a title

```
#- new theme
dot_theme = theme_bw() +
            theme(panel.grid.major.x=element_blank(),
                  panel.grid.minor.x=element_blank(),
                  panel.grid.major.y=element_line(color="grey60",
                                                  linetype="dotted"))

#- Cleveland dot plot
ggplot(H) +
  geom_point(aes(x=teamHR, y=reorder(teamID, teamHR),
                           size=teamR, color=teamBA>.260))  +
  dot_theme +
  labs(title = "2014 MLB", x="home runs", y="team") +
  scale_color_manual(name="BA>.260", values=c("blue", "orange")) +
  scale_size(name="Runs", range=c(1,6))
```



The Cleveland Dot Plot is an alternative to a bar plot. There is also a dot plot (`geom_dotplot()`) that is an alternative to a histogram.

2 Line Graphs

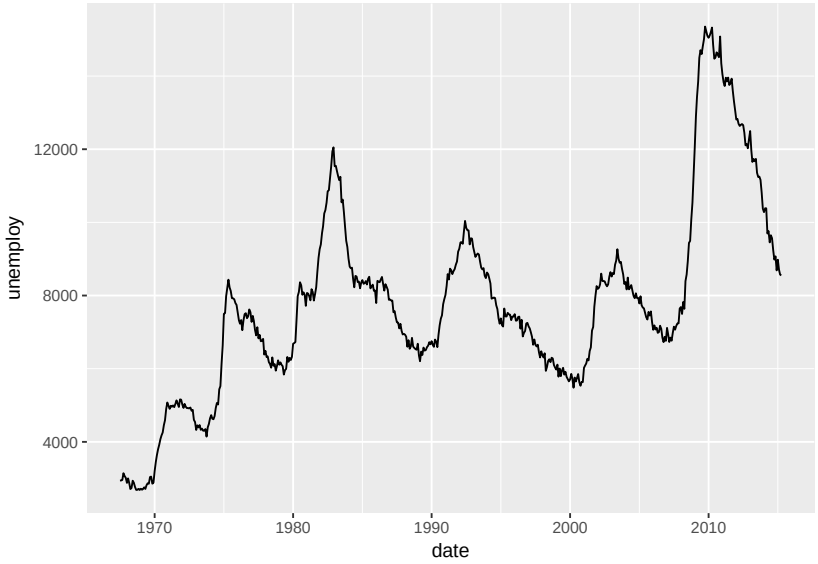
2.1 `economics` data

The `economics` data from the `ggplot2` package contains some economic time series data

```
library(dplyr)
library(ggplot2)
data(economics)
glimpse(economics)
#> Observations: 574
#> Variables: 6
#> $ date     <date> 1967-07-01, 1967-08-01, 1967-09-01, 1967-10-01, 1967...
#> $ pce      <dbl> 507.4, 510.5, 516.3, 512.9, 518.1, 525.8, 531.5, 534....
#> $ pop      <int> 198712, 198911, 199113, 199311, 199498, 199657, 19980...
#> $ psavert  <dbl> 12.5, 12.5, 11.7, 12.5, 12.5, 12.1, 11.7, 12.2, 11.6,...
#> $ uempmed  <dbl> 4.5, 4.7, 4.6, 4.9, 4.7, 4.8, 5.1, 4.5, 4.1, 4.6, 4.4...
#> $ unemploy <int> 2944, 2945, 2958, 3143, 3066, 3018, 2878, 3001, 2877,...
```

We can plot the number of unemployed over time with a line plot (using `geom_line()`)

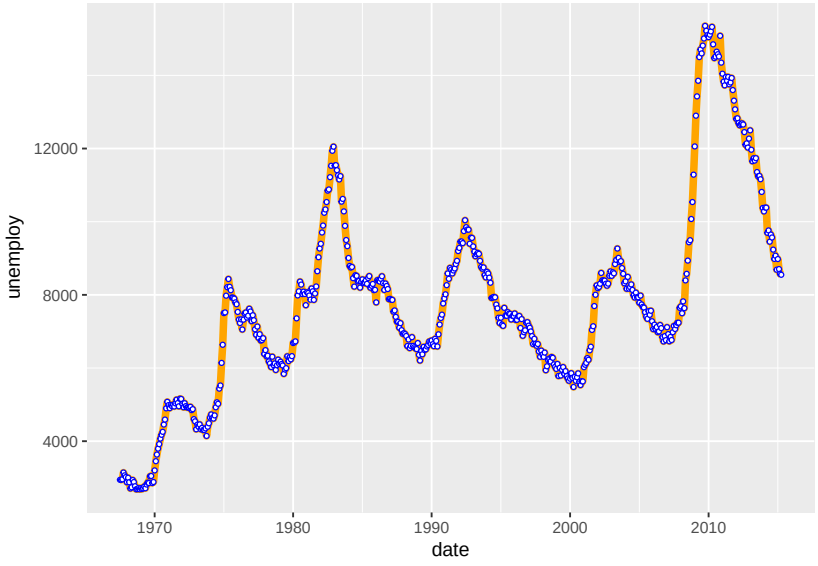
```
ggplot(economics, aes(date, unemploy)) + geom_line()
```



ggplot recognizes the date class and smartly adds yearly tick marks.

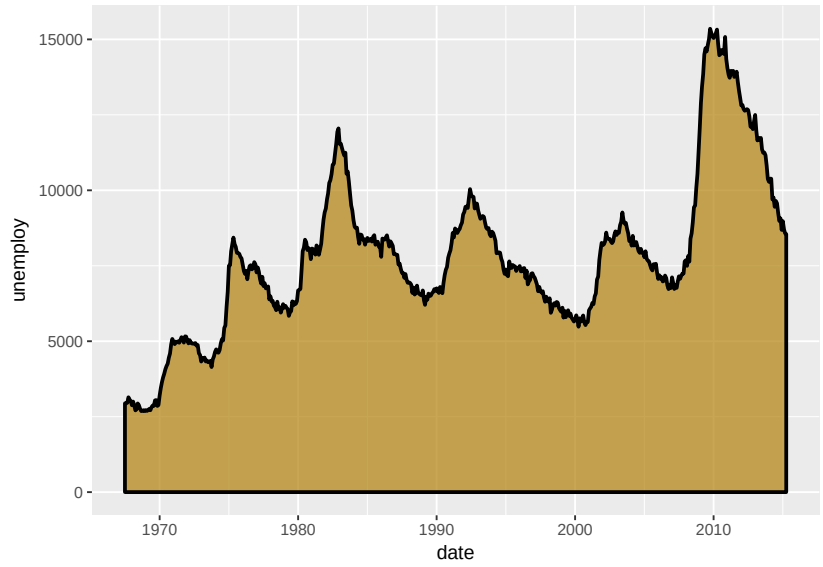
We can fancy it up, maybe add some points

```
ggplot(economics, aes(date, unemploy)) +
  geom_line(size=2, color="orange") +
  geom_point(shape=21, color='blue', fill='white', size= 1)
```



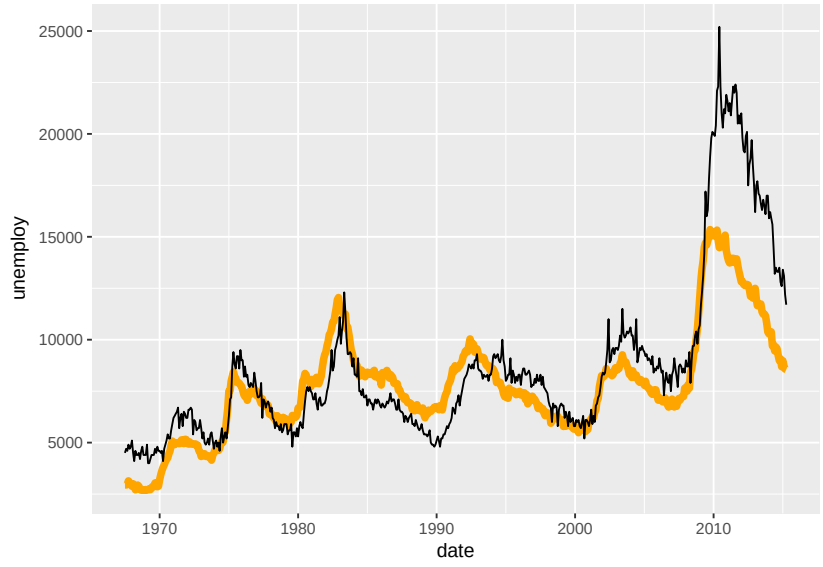
We can shade the region under the line with `geom_area()`

```
ggplot(economics, aes(date, unemploy)) +
  geom_area(color='black', fill='#B1810B', alpha=.7, size=1)
```



Multiple lines (using another aesthetic mapping for second line)

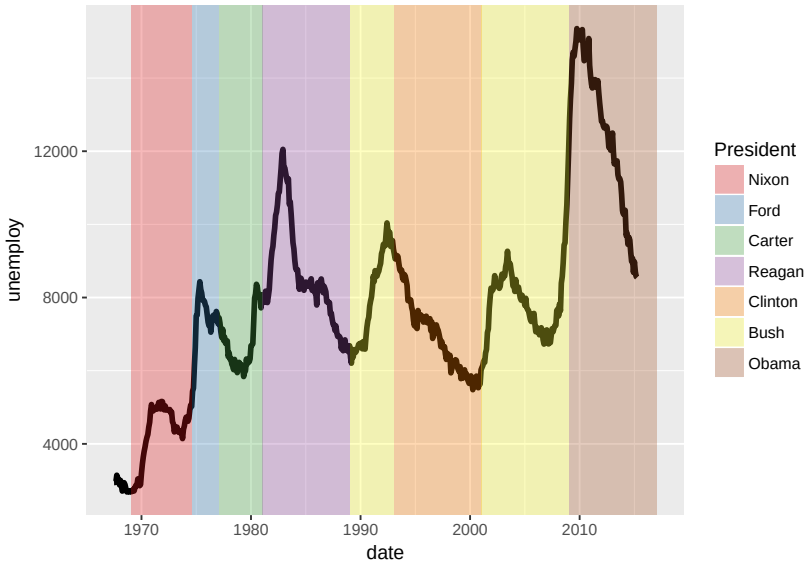
```
ggplot(economics, aes(date, unemploy)) +
  geom_line(size=2, color="orange") +
  geom_line(aes(date, uempmed*1000 ))
```



How did the economy do for the presidents? Let's use the `presidential` data from `ggplot2` and use `geom_rect()` to shade in the time period for each president

```
ggplot(economics) +
  geom_line(aes(date, unemploy),size=1.5, color="black") +
  geom_rect(data=filter(presidential, start>as.Date("1969-01-01")),
            aes(xmin=start, xmax=end, ymin=-Inf, ymax=Inf, fill=reorder(name, start)),
            alpha=.3) +
```

```
scale_fill_brewer(palette="Set1", name="President")
```



2.2 Your Turn: Stock Price

Your Turn #2 : Stock Price

This exercise will walk you through a simple way to plot stock data.

1. Check out the yahoo webpage of historical stock price data for Netflix (NFLX).
2. Notice the url from the `Download Data` link. There is a pattern in requesting dates (a,d are month-1). Here is the link I used from 1/1/16-2/4/17:

```
url = "http://chart.finance.yahoo.com/table.csv?
s=NFLX&a=00&b=01&c=2016&d=01&e=04&f=2017&g=d&ignore=.csv"
```

3. Read the data into R as a tibble:

```
NFLX = read_csv(url)
```

4. Examine the data, then create a line plot of the `Close` price by `Date`. Color the line darkgreen.
5. Use `geom_area()` to fill the area below the line with lightgreen.

3 Estimating Distributions

We need to estimate **distributions** to understand how likely (or unlikely) a certain outcome is. For many problems, an optimal decision can be formulated if we know the **distribution** of the random variable. Often, only certain properties of the distribution (expected value, variance, quantiles) are needed to make decisions. Much of statistics is involved with estimation of the distributions or their properties.

Let X be a random variable of interest. The **cumulative distribution function (cdf)** is $F(x) =$

$\Pr(X \leq x)$. For *discrete* random variables, the **probability mass function (pmf)** is $f(x) = \Pr(X = x)$. For *continuous* random variables, the **probability density function (pdf)** is $f(x) = \frac{d}{dx}F(x)$.

A **parametric** distribution is one that is fully characterized by a set of parameters. Examples include Normal/Gaussian (mean-μ,standard deviation-σ), Poisson (rate-λ), Binomial (size-n, probability-p). There are also multivariate versions: Gaussian $N(\mu, \Sigma)$. If we can model (assume) the random variable follows a specific parametric distribution, then we only need to estimate the parameter(s) to have the entire distribution characterized. The parameters are often of direct interest themselves (mean, standard deviation).

3.1 Discrete Data

Discrete data can be numeric or categorical. Categorical (or Qualitative) Data can take values in a (finite) fixed set that indicate a group or category, while Discrete Numeric data may take a countably infinite number of values.

- Numeric (ordered)
  - *crimes per year*: {0,1,...}
  - *number of home runs in next 10 at bats*: {0,1,...,10}
- Ordinal (ordered)
  - *rate your boss*: {horrible, bad, fair, good, great}
- Nominal (no order)
  - *get to work*: {car, walk, bus, fly}

Relative frequency (proportion of times the value occurs in a sample) is the most basic way to estimate a discrete probability mass function. Bar graphs are the basic graphical device.

3.2 Continuous Data

Continuous data can take an uncountably infinite number of values (if it was possible to measure with enough resolution). Continuous data are usually measurements and take values on intervals of the real numbers. For example,

- time
- weight
- speed
- temperature

Because we can never actually measure or store something to infinite precision, we will always technically be working with discrete data. The distinction is more related to what methods are used to analyze the data. For example, we can treat discrete random variables, like counts, as continuous for regression modelling or histograms. Or we can bin continuous data for use in frequency tables or cluster/segmentation analysis (e.g., $\mathbb{1}(\text{heightIn} > 60)$).

In R, continuous data is represented with numeric vectors. Sometimes discrete data (like counts) can be treated as continuous data for graphical purposes and sometimes treated like categorical data.

3.2.1 Density Estimation

There are two flavors of density estimation: parametric and non-parametric.

Parametric Density Estimation involves two steps:

1. Choose a parametric distribution
2. Estimate the parameter(s)

Of course, these are not trivial tasks. There is a huge number of parametric families and many ways to estimate parameters (e.g., MLE, MOM, Bayesian approaches).

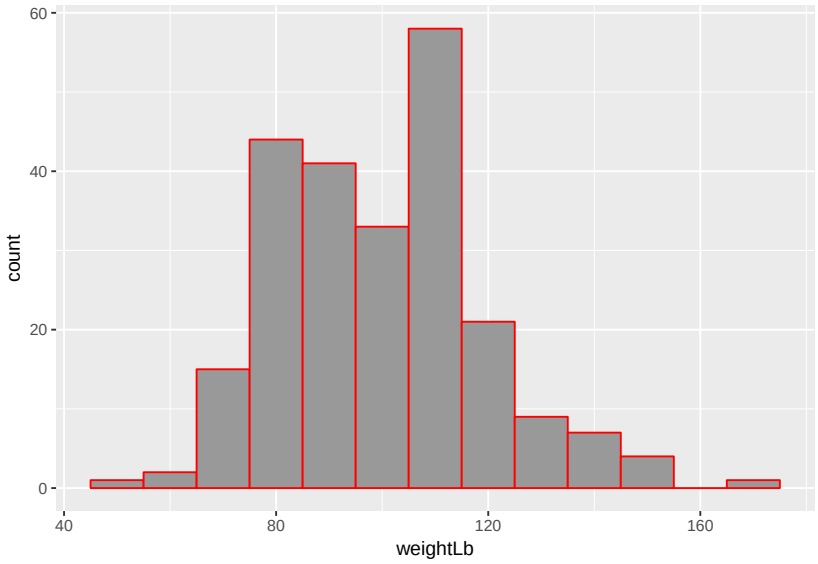
Non-parametric Density Estimation attempts to estimate the density with minimal assumptions on the distribution. Non-parametric density estimation is a great exploratory tool. You are familiar with one approach already: histograms.

4 Histograms

4.1 `geom_hist()`

A histogram still requires some parameters (or settings); namely the the binning. Here we use a 10lb bin width.

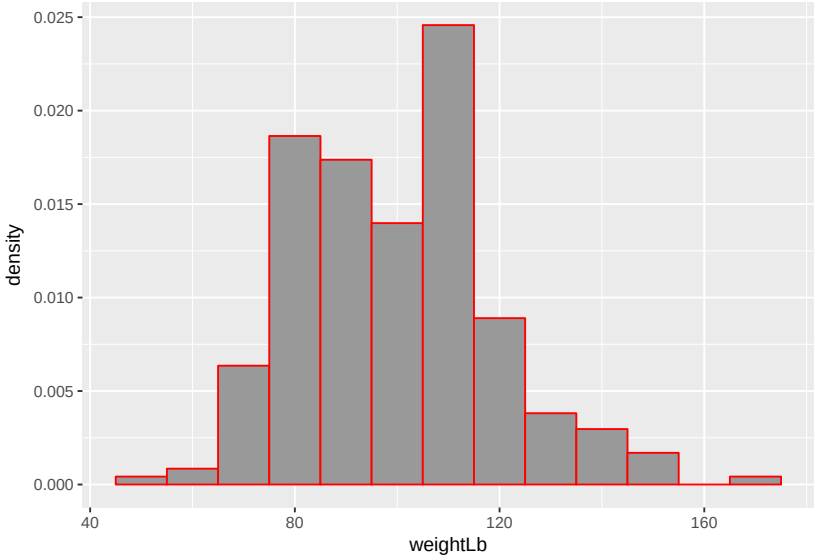
```
library(gcookbook)
data(heightweight)
summary(heightweight)
#>  sex        ageYear        ageMonth      heightIn       weightLb
#>  f:111   Min.   :11.6   Min.   :139   Min.   :50.5   Min.   : 50.5
#>  m:125   1st Qu.:12.3   1st Qu.:148   1st Qu.:58.7   1st Qu.: 85.0
#>          Median :13.6   Median :163   Median :61.5   Median :100.5
#>          Mean   :13.7   Mean   :164   Mean   :61.3   Mean   :101.0
#>          3rd Qu.:14.8   3rd Qu.:178   3rd Qu.:64.3   3rd Qu.:112.0
#>          Max.   :17.5   Max.   :210   Max.   :72.0   Max.   :171.5

g = ggplot(heightweight, aes(x=weightLb))
g + geom_histogram(binwidth=10, fill="grey60", color="red")
```



But these histograms are not actually a valid density estimation. They are actually bar graphs without space between the bars. Recall that that a proper density will have the properties: $\hat{f}(x) \geq 0$ and $\int \hat{f}(x)\, dx = 1$.

So, to be a proper density the area of the bars must sum to one. In ggplot2, this can be accomplished by specifying the y-axis aesthetic to be `..density...`

```
hist.density = geom_histogram(aes(y=..density..), binwidth=10,
                              fill="grey60", color="red")
g + hist.density
```



Relative frequency bar heights can be obtained with `aes(y=..count../sum(..count..))`. The default value is `aes(y=..count..)`

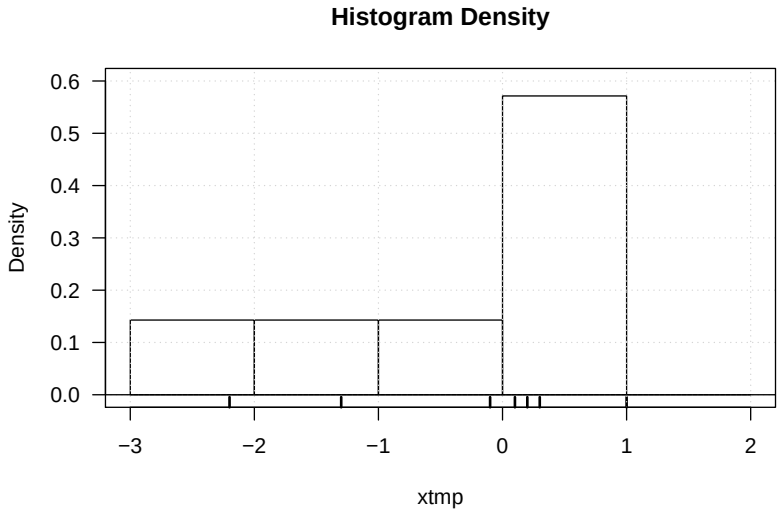
The authors recommend *density scaled* frequency polygons `geom_freqpoly(aes(y=..density..))`

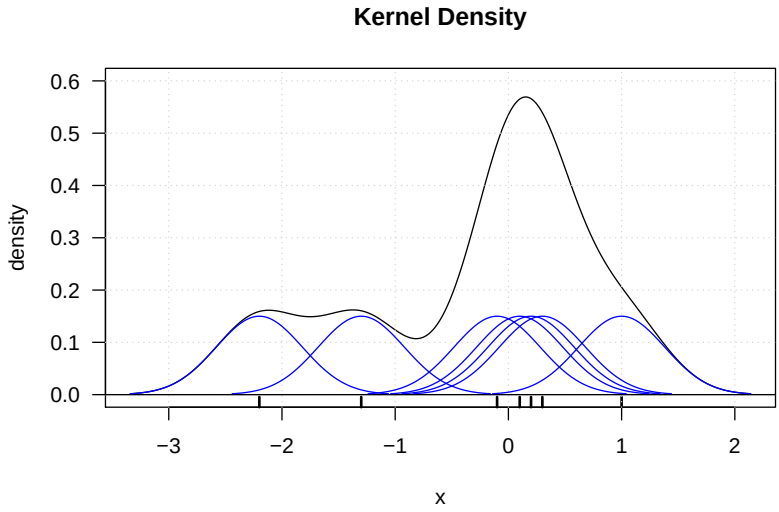
5 Kernel Density Estimation

Density histograms are only one non-parametric density estimation technique. A better one is kernel density estimation (KDE). This estimates the density by summing a set of **kernels** which are centered on each observation

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} K(x - x_i)$$

The kernels themselves are usually taken to be proper density function (e.g., Gaussian) that are centered at zero. The level of overall smoothing in controlled by the **bandwidth** (which is proportional to the standard deviation for a Gaussian kernel). Consider the data `xtmp = c(-2.2,-1.3,-0.1,0.1,0.2,0.3,1)`



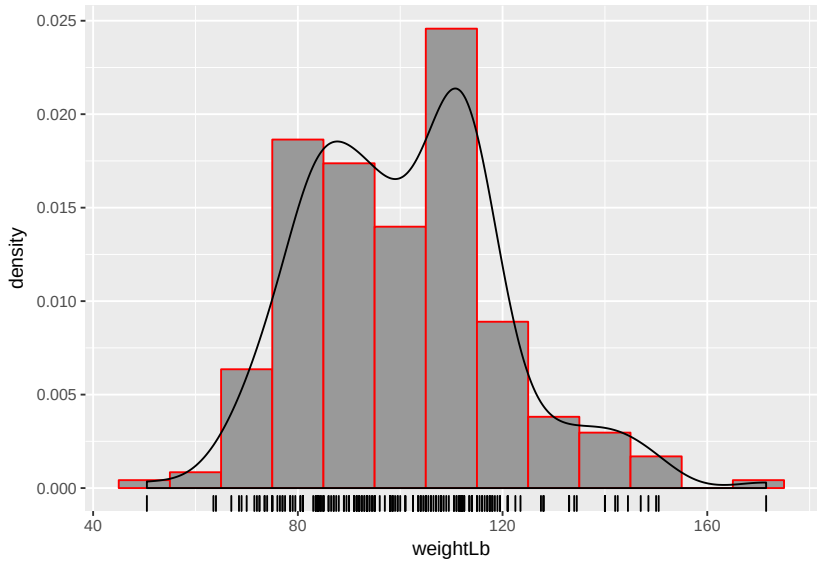


The histogram can give a very different solution if the origin for the bin is adjusted. Is it really good to assume a 0 density for values greater than 1? The kernel density, alternatively, eliminates the binning origin issue.

5.1 `geom_density()`

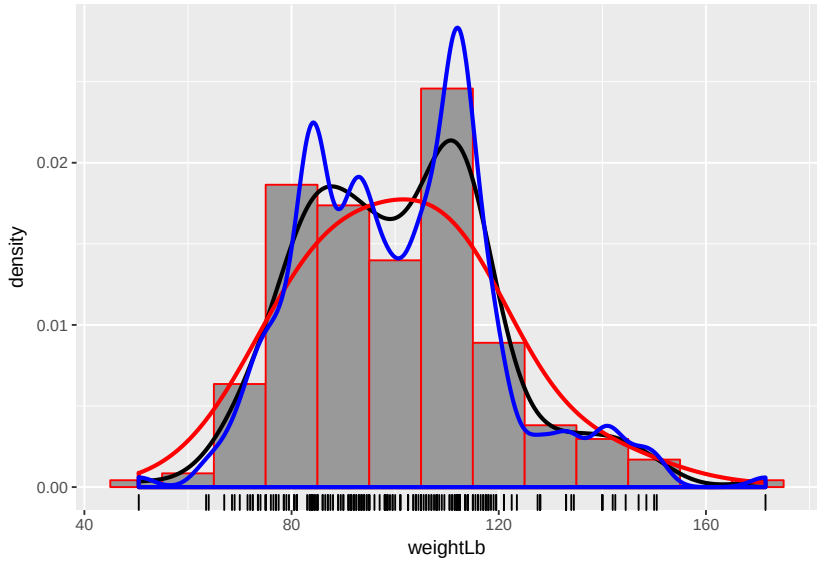
Add this to the duration data, we can see how the kernel density compares to the histogram

```
g + hist.density + geom_density() + geom_rug()
```



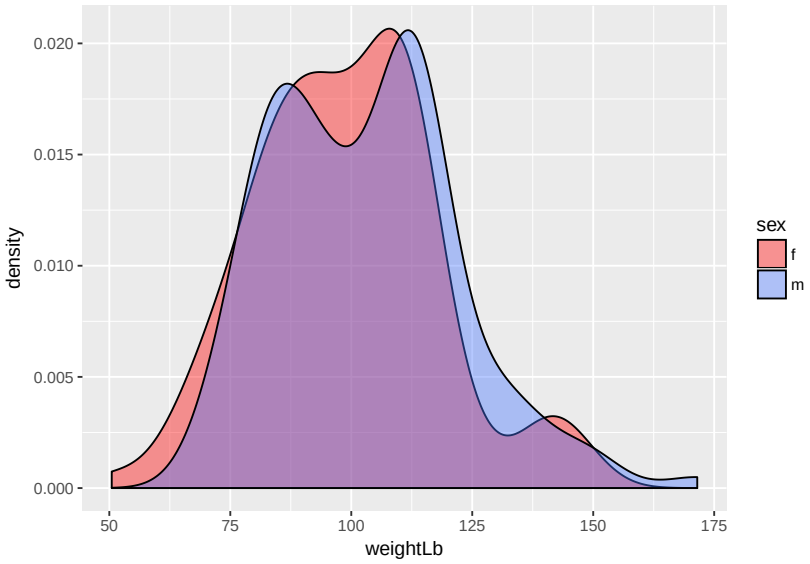
While it is possible to change the kernel (gaussian, biweight, etc.) the biggest difference will occur by adjusting the bandwidth. This is done with the `adjust=` argument.

```
base = g + hist.density + geom_rug()

base + geom_density(color="black", size=1.1) +
  geom_density(adjust=2, color="red", size=1.1) +
  geom_density(adjust=1/2, color="blue", size=1.1)
```



Compare the density of weight between the boys and girls

```
g + geom_density(aes(fill=sex), alpha=.4) +
  scale_fill_manual(values = c(f="red", m="royalblue1")) # modify default colors
```

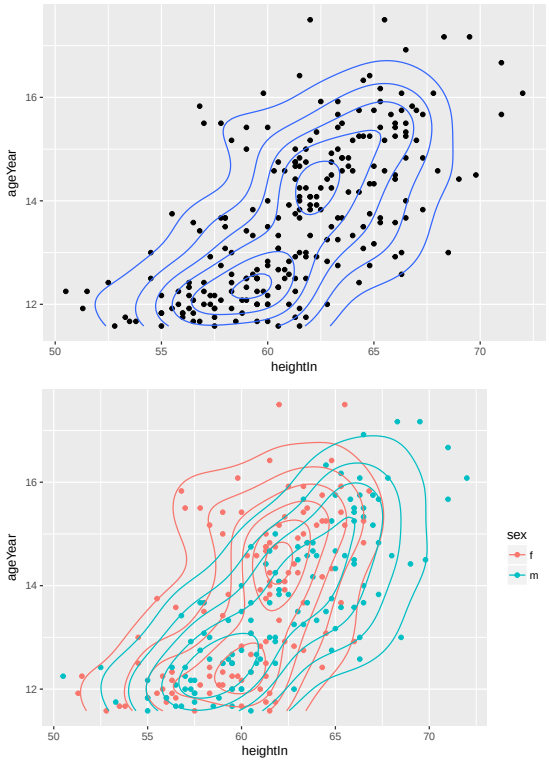


More information and examples of density curves, including using multiple curves for grouped data, can be found in RGraphics Chapter 6.

5.1.1 2D Density

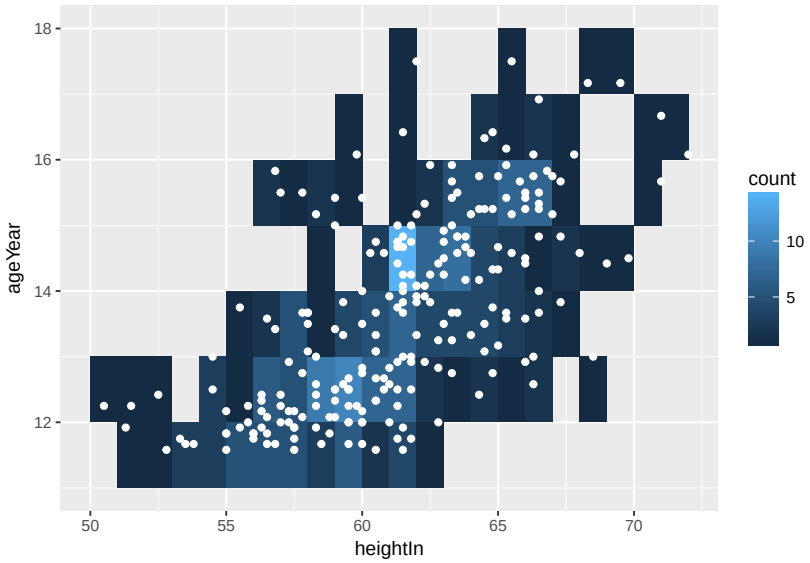
There are also functions for 2D density, for example `geom_density2d()`:

```
#- (left) basic
ggplot(heightweight, aes(x = heightIn, y = ageYear)) + geom_point() +
  geom_density2d()

#- (right) separate by sex
ggplot(heightweight, aes(x = heightIn, y = ageYear, color=sex)) + geom_point() +
  geom_density2d()
```



The 2D histogram to show 2D covariation is called by `geom_bin2d`

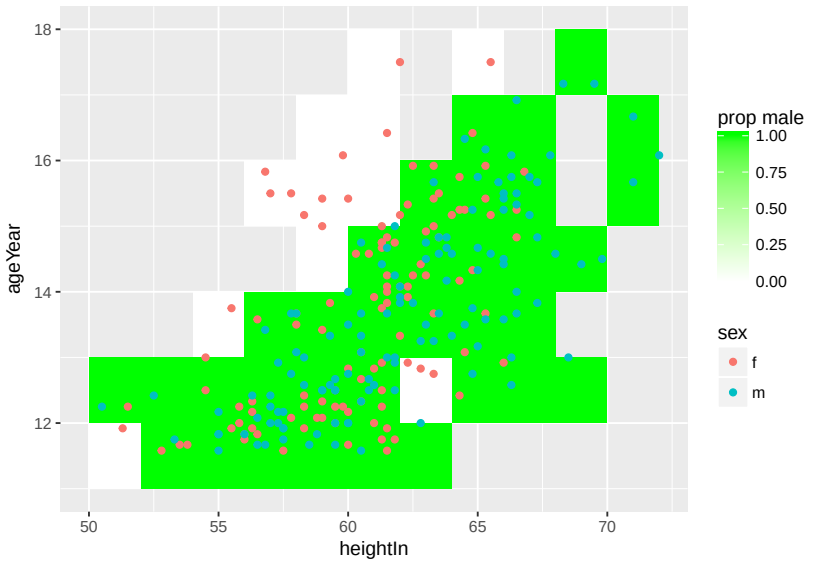
```
ggplot(heightweight, aes(x = heightIn, y = ageYear)) +
  geom_bin2d(binwidth=c(1,1)) +                # explicitly set binwidths
  geom_point(color="white")                    # add overlay of points
```



It can be useful to fill the color of the 2d bins with other measures besides the count. Here will set the fill color to the proportion of males:

```
ggplot(heightweight, aes(x = heightIn, y = ageYear, z=sex)) +
  stat_summary_2d(fun=function(z) mean(z=="m"), binwidth=c(2,1)) +
```

```
  geom_point(aes(color=sex)) +
  scale_fill_gradient(low="white", high="green", name="prop male")
```



Notice the use of `z=` aesthetic in `ggplot()` and the `fun=` argument in `stat_summary_2d()`.

5.2 Some Useful Settings: `geom_histogram`, `geom_density`

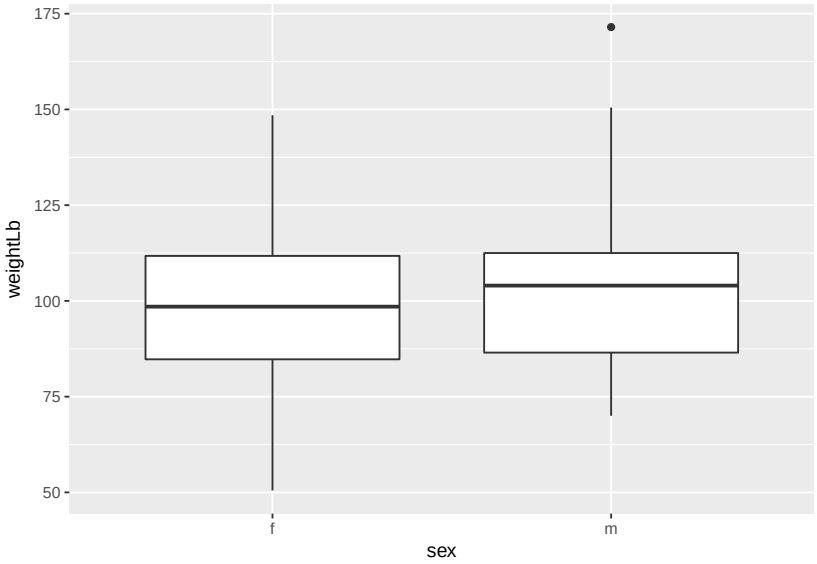
- `geom_histogram` has options: (see `?stat_bin` for details)
  - binwidth: width of the bins
  - origin: the start point for the first bin (default is min(data))
  - breaks: the specific breakpoints. Overrides binwidth and origin if provided
  - right: FALSE (default) right open, left closed
- `geom_density` has options: (see `?stat_density` for details)
  - adjust: the bandwidth used is actually adjust*bw. This makes it easy to specify values like *half the default* bandwidth.
  - kernel: a character string giving the smoothing kernel to be used. See `?density` for options. Default is `'gaussian'`.
  - trim: if TRUE, the default, densities are trimmed to the actual range of the data.
  - na.rm: If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
- They both can accept aesthetics: (see `?geom_histogram` and `?geom_density`)
  - color: line color
  - fill: fill color
  - alpha: transparency [0,1]
  - linetype: "blank", "solid", "dashed", "dotted", "dotdash", "longdash", and "twodash"
  - size: size of lines
  - weight: maybe weighted observations?

6 Boxplot and Violin Plot

6.1 Boxplot

The old boxplot's strength is comparing multiple distributions without a parametric assumption.

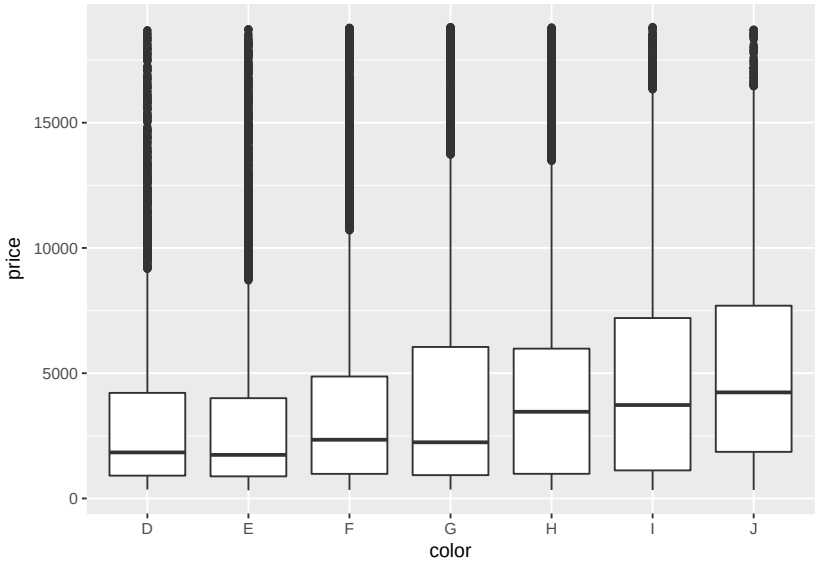
```
g2 = ggplot(heightweight, aes(x=sex, y=weightLb))
g2 + geom_boxplot()
```



Remember the box indicates the 1st and 3rd quartile (with horizontal line the median). The *whiskers* extend 1.5(Q3-Q1) units (or min/max) from box and outliers are represented by points.

The boxplot is usually better than kde/histogram when comparing the distribution of several groups. For example, the distribution of diamond `price` by `color`

```
ggplot(diamonds, aes(color, price)) + geom_boxplot()
```

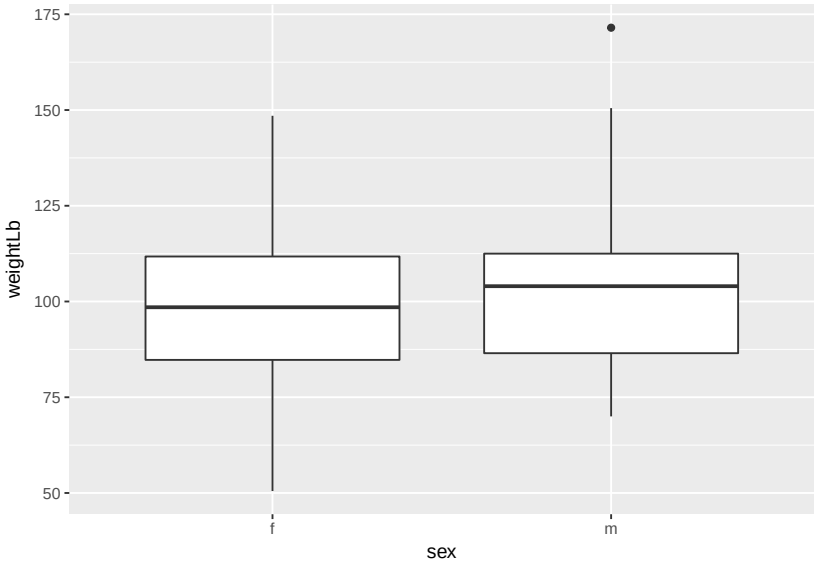


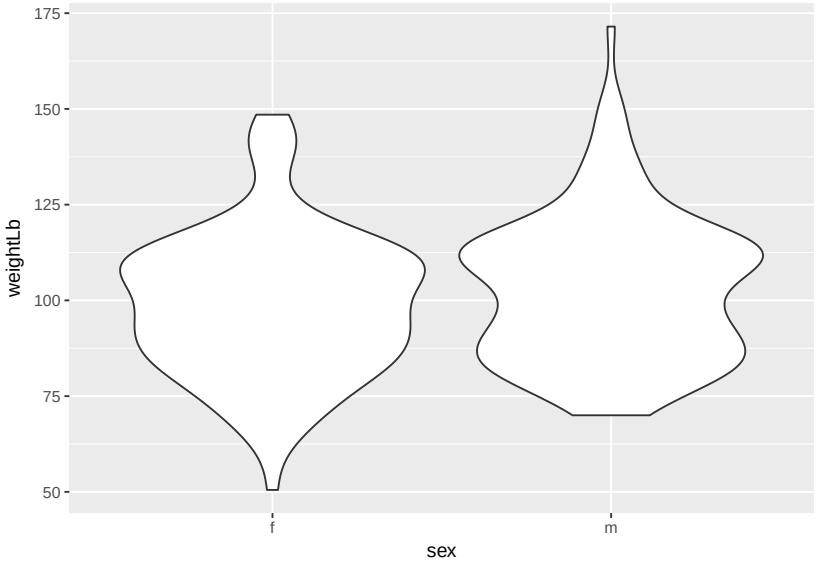
This reveals, at a glance, the five number summary for each color group.

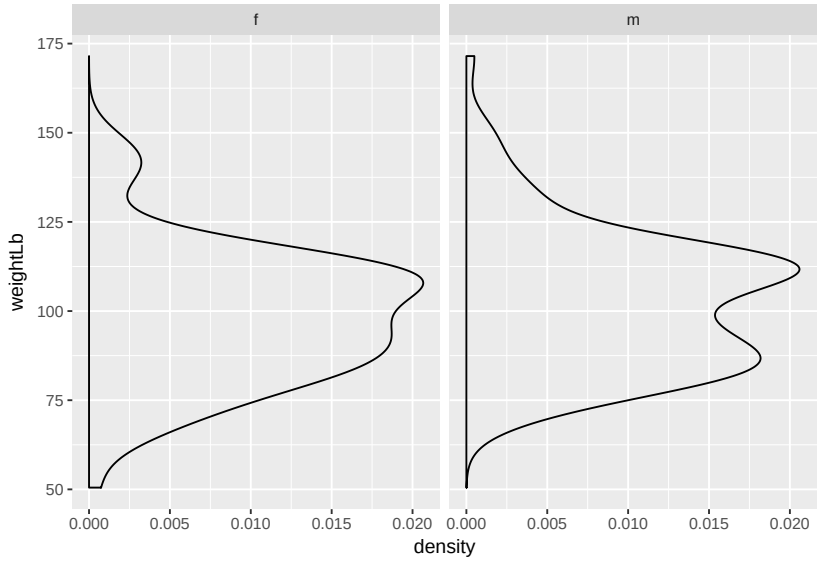
6.2 Violin Plot

A problem with the boxplot is that it does not reveal the shape of the distribution (e.g., modes). A *violin plot* is a better alternative. It replaces the box with a kernel density estimate.

```
g2 + geom_boxplot()

g2 + geom_violin()

ggplot(heightweight, aes(x=weightLb)) + geom_density() +
  facet_wrap(~sex) + coord_flip()
```



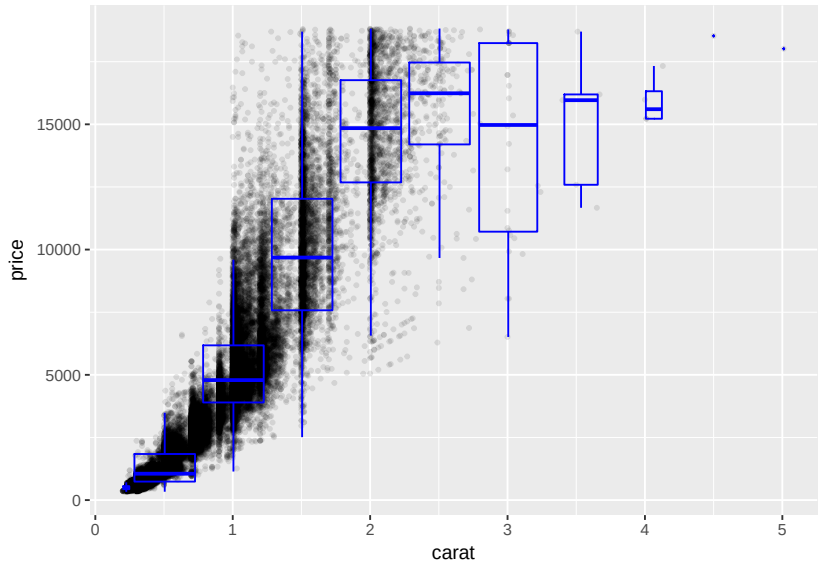




6.3 Discretizing continuous variables for boxplot

If we want to see the conditional distribution of `price` (y-axis) according to `carat` (x-axis) we can bin the x variable (using `cut_width()`) and make boxplots:

```
ggplot(diamonds, aes(carat, price)) +
  geom_point(alpha=.1, shape=20) +
  geom_boxplot(aes(group=cut_width(carat, .5)), fill=NA, color="blue", outlier.alpha=0)
```



Notice I used `cut_width(x, width)` to make the groups. This divides `x`, a vector of numeric data, into bins of width equal to `width`. Here are some similar functions. All take a *numeric* vector `x`:

- `cut_width(x, width)` bins of width `width`
- `cut_number(x, n)` finds `n` bins of approximately equal number of observations

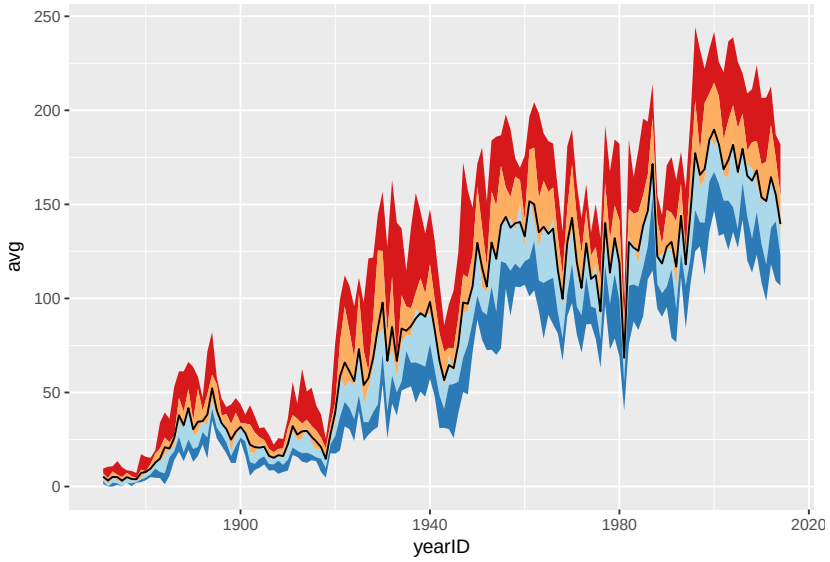
- `cut_interval(x, n)` find `n` bins of equal width
- `cut(x, breaks, ...)` for general purpose. Set the `breaks` argument for ultimate flexibility.

6.4 Sequentile Quantiles (Fanchart, Fanplot)

Instead of a series of separate boxplots, we can connect the quantiles with lines.

For this, we will need to some baseball data and some transformations

```
library(Lahman)

#-- Get the quantiles for team HR's by year
qHR =
  Batting %>%
    group_by(yearID, teamID) %>%
    summarize(HR=sum(HR, na.rm=TRUE)) %>%  # team HR's each year
    summarize(                             # quantiles for each year
      q95=quantile(HR, .95, na.rm=TRUE),
      q75=quantile(HR, .75, na.rm=TRUE),
      q50=quantile(HR, .50, na.rm=TRUE),
      q25=quantile(HR, .25, na.rm=TRUE),
      q05=quantile(HR, .05, na.rm=TRUE),
      avg=mean(HR, na.rm=TRUE))            # always nice to get mean too


#-- Plot with geom_ribbon
ggplot(qHR, aes(x=yearID)) +
  geom_ribbon(aes(ymax=q95, ymin=q75), fill="#d7191c") +
  geom_ribbon(aes(ymax=q75, ymin=q50), fill="#fdae61") +
  geom_ribbon(aes(ymax=q50, ymin=q25), fill="#abd9e9") +
  geom_ribbon(aes(ymax=q25, ymin=q05), fill="#2c7bb6") +
  geom_line(aes(y=avg), color="black")
```



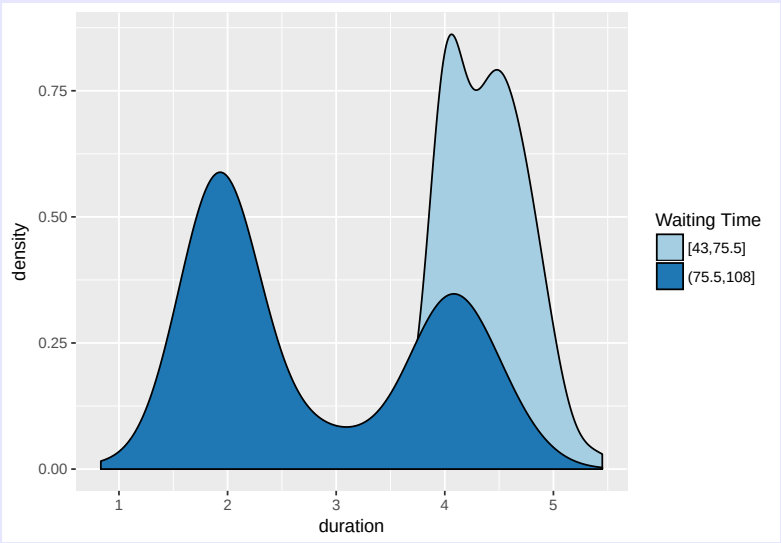
6.5 Your Turn: Old Faithful

{https://upload.wikimedia.org/wikipedia/commons/1/1b/Old_Faithfull-pdPhoto.jpg }

Your Turn #3 : Old Faithful

The `geyser` data from the `MASS` package gives duration and waiting times for The Old Faithful Geyser in Yellowstone National Park, Wyoming. Load the `geyser` data. (Remember `install.packages()` if you don't have `MASS` loaded)

1. Estimate the *density* of `duration` with a histogram.
   - What did you choose for binwidth and/or origin and/or bins?
2. Add a KDE of `duration` to the histogram
   - What did you choose for adjust (i.e., kernel bandwidth)?
3. Estimate the bivariate density of `duration` and `waiting`. Do you see how `duration` can have two groups?
4. Overlay two KDE density curves by creating a discrete variable for `fill=` by splitting the `waiting` time into two equally sized bins with `cut_number(waiting, n=2)`. (Make the plot below (ignore colors):)


0.75
0.50
0.25
0.00
density
1
2
3
4
5
duration
Waiting Time
[43,75.5]
(75.5,108]