MATH 3050 – Predictive Analytics

UNC CHARLOTTE

Topic 1: The R Programming Language – Part 3

- Package dplr
- Exploratory Data Analysis Checklist
- Principles of Analytic Graphics
- GGPLOT

1

Objectives of this Lesson:

By the end of this lesson you should be able to:

- Use the package dplyr to subset data
- Apply exploratory techniques to data analysis
- Apply principles of analytics graphics to data analysis
- Create graphics layers using ggplot versus plot()
- Create smooth function plots using:
  - Generalized Additive Models (GAMs)
  - Locally Weighted Scatterplot Smoothing (LOWESS)
  - Local Regressions (LOESS)
  - Cubic Polynomials

2

The "dplyr" package

Developed by Hadley Wickham of Rstudio.
A disciple of Leland Wilkinson.

Operations, like filtering, re-ordering, and collapsing, can often be tedious operations in R whose syntax is not very intuitive.

The **dplyr** package is designed to mitigate a lot of these problems and to provide a highly optimized set of routines specifically for dealing with data frames

The **dplyr** package does not provide any "new" functionality to R per se, in the sense that everything **dplyr** does could already be done with base R, but **dplyr** *greatly* simplifies existing functionality in R.

Another useful contribution is that the **dplyr** functions are **very** fast, as many key operations are coded in C++.

3

The "dplyr" package

```
install.packages("dplyr")
library(dplyr)
```

```
Attaching package: 'dplyr'

The following object is masked from 'package:stats':

    filter

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

You may get some warnings when the package is loaded because there are functions in the dplyr package that have the same name as functions in other packages. For now you can ignore the warnings or use the format dplyr::objectname to avoid masking warnings.

4

The "dplyr" package

dplyr Grammar

Some of the key "verbs" provided by the **dplyr** package are

1. select: return a subset of the columns of a data frame, using a flexible notation
2. filter: extract a subset of rows from a data frame based on logical conditions
3. arrange: reorder rows of a data frame
4. rename: rename variables in a data frame
5. mutate: add new variables/columns or transform existing variables
6. summarise / summarize: generate summary statistics of different variables in the data frame, possibly within strata
7. %>%: the "pipe" operator is used to connect multiple verb actions together into a pipeline

5

The "dplyr" package

select(): You can use ":" with names as well as column numbers. Need to import chicago.csv.

To select columns

```
Chicago <- read.csv("C:/RData/ChicagoData/chicago.csv", stringsAsFactors=FALSE)
attach(Chicago)
subset <- select(Chicago, 1:3)
subset <- select(Chicago, city:dptp)
```

"chicago" dataset contains air pollution and temperature data for the city of Chicago.

Select () works regardless of masking!

To drop columns as well

```
select(Chicago, -(city:dptp))
```

Notice the use of the "-" to drop columns.

select(): You can column names based on patterns

```
'data.frame': 6940 obs. of 4 variables:
$ pm25tmean2: num NA NA NA NA NA NA NA NA NA NA ...
$ pm10tmean2: num 34 NA 34.2 47 NA ...
$ o3tmean2 : num 4.25 3.3 3.33 4.38 4.75 ...
$ no2tmean2 : num 20 23.2 23.8 30.4 30.3 ...
```


6

The "dplyr" package

fliter(): Used to extract subsets of rows from a data frame. Similar to subset() but faster.

```
chic.f <- filter(Chicago, pm25tmean2 > 30)
str(chic.f)
```

```
'data.frame': 194 obs. of 8 variables:
$ city : chr "chic" "chic" "chic" "chic" ...
$ tmpd : num 23 28 55 59 57 57 75 61 73 78 ...
$ dptp : num 21.9 25.8 51.3 53.7 52 56 65.8 59 60.3 67.1 ...
$ date : Date, format: "1998-01-17" "1998-01-23" ...
$ pm25tmean2: num 38.1 34 39.4 35.4 33.3 ...
$ pm10tmean2: num 32.5 38.7 34 28.5 35 ...
$ o3tmean2 : num 3.18 1.75 10.79 14.3 20.66 ...
$ no2tmean2 : num 25.3 29.4 25.3 31.4 26.8 ...
```


The "dplyr" package

Data Dictionary for chicago data set

This data set contains air quality data for the city of Chicago. It is from a larger multi-city, governmental data set.

city : This is populated with 'chic' for Chicago
tmpd : Daily average temparature in Fahrenheit
dptp : This is the dew point temparature
date : Date of collection of air quality data
pm25tmean2 : Particulate Matter of 2.5 micrometers in diameter.
pm10tmean2 : Particulate Matter of 10 micrometers in diameter
o3tmean2 : Mean daily Ozone (o3) level
no2tmean2 : Mean daily Nitrogen Dioxide (no2) level


HEI
RESEARCH REPORT
HEALTH EFFECTS INSTITUTE
Number 94, Part II
June 2000
Final Version
The National Morbidity, Mortality, and Air Pollution Study
Part II: Morbidity and Mortality from Air Pollution in the United States
Jonathan M Samet, Scott L Zeger, Francesca Dominici, Frank Curriero, Ivan Coursac, Douglas W Dockery, Joel Schwartz, and Antonella Zanobetti

The "dplyr" package

fliter(): Can use logical expressions to filter.

```
chic.f <- filter(Chicago, pm25tmean2 > 30 & tmpd > 80)
select(chic.f, date, tmpd, pm25tmean2)
```

```
date tmpd pm25tmean2
1 1998-08-23 81 39.60000
2 1998-09-06 81 31.50000
3 2001-07-20 82 32.30000
4 2001-08-01 84 43.70000
5 2001-08-08 85 38.83750
6 2001-08-09 84 38.20000
7 2002-06-20 82 33.00000
8 2002-06-23 82 42.50000
9 2002-07-08 81 33.10000
```


The "dplyr" package

arrange(): used to reorder rows of a data frame according to one or more of the variables/columns.

```
Chicago <- arrange(Chicago, date)
Chicago

Chicago <- arrange(Chicago, desc(date))
Chicago

Chicago <- arrange(Chicago, tmpd, desc(date))
Chicago
```

The "dplyr" package

rename(): Use to easily rename a variable in a data frame.

```
Chicago <- rename(Chicago, dewpoint = dptp, pm25 = pm25tmean2)
head(Chicago[, 1:5], 3)
```

```
city tmpd dewpoint date pm25
1 chic 35 30.1 2005-12-31 15.00000
2 chic 36 31.0 2005-12-30 15.05714
3 chic 35 29.4 2005-12-29 7.45000
```

There are two new variables:
1. **dewpoint**
2. **pm25**

The syntax inside the **rename()** function is to have the new name on the left-hand side of the **=** sign and the old name on the right-hand side.

The "dplyr" package

mutate(): Used to compute transformations of variables in a data frame. It is used to provide a clean interface to create new variables that are derived from existing variables.

```
Chicago <- mutate(Chicago, pm25detrend = pm25 - mean(pm25, na.rm = TRUE))
head(Chicago)
```

```
  city tmpd dewpoint       date     pm25 pm10tmean2  o3tmean2 no2tmean2 pm25detrend
1 chic   35     30.1 2005-12-31 15.00000       23.5  2.531250  13.25000   -1.230958
2 chic   36     31.0 2005-12-30 15.05714       19.2  3.034420  22.80556   -1.173815
3 chic   35     29.4 2005-12-29  7.45000       23.5  6.794837  19.97222   -8.780958
4 chic   37     34.5 2005-12-28 17.75000       27.5  3.260417  19.28563    1.519042
5 chic   40     33.6 2005-12-27 23.56000       27.0  4.468750  23.50000    7.329042
6 chic   35     29.6 2005-12-26  8.40000        8.5 14.041667  16.81944   -7.830958
```

There is a new variables:
1. **Pm25detrend**
2. **pm25**

The "dplyr" package

transmutate(): which does the same thing as mutate() but then drops all non-transformed variables.

```
Chicago <- arrange(Chicago, desc(date))
head(transmute(Chicago, pm10detrend = pm10tmean2 - mean(pm10tmean2, na.rm = TRUE),
                        o3detrend = o3tmean2 - mean(o3tmean2, na.rm = TRUE)))
```

```
  pm10detrend  o3detrend
1  -10.395206 -16.904263
2  -14.695206 -16.401093
3  -10.395206 -12.640676
4   -6.395206 -16.175096
5   -6.895206 -14.966763
6  -25.395206  -5.393846
```


The "dplyr" package

group_by(): Used to generate summary statistics from the data frame within strata defined by a variable. It is often used with the summarize() function.

First, we can create a year varible using as.POSIXlt().

```
Chicago <- mutate(Chicago, year = as.POSIXlt(date)$year + 1900) #We added a year variable to dataset
```

Now we can create a separate data frame that splits the original data frame by year.

```
years <- group_by(Chicago, year)
```

Compute summary statistics for each year in the data frame with the summarize() function.

```
summarize(years, pm25 = mean(pm25, na.rm = TRUE),  o3 = max(o3tmean2, na.rm = TRUE),
                                                   no2 = median(no2tmean2, na.rm = TRUE))
```

Parameter for handling missing values:

```
na.rm = TRUE #This will ignore missing values for any calculations
na.rm = FALSE #This will produce an NA for any calculations using missing data
```

The "dplyr" package

group_by(): Used to generate summary statistics from the data frame within strata defined by a variable. It is often used with the summarize() function.

summarize() returns a data frame with year as the first column, and then the annual averages of pm25, o3, and no2.

	year	pm25	o3	no2
1	1987	NaN	62.96966	23.49369
2	1988	NaN	61.67708	24.52296
3	1989	NaN	59.72727	26.14062
4	1990	NaN	52.22917	22.59583
5	1991	NaN	63.10417	21.38194
6	1992	NaN	50.82870	24.78921
7	1993	NaN	44.30093	25.76993
8	1994	NaN	52.17844	28.47500
9	1995	NaN	66.58750	27.26042
10	1996	NaN	58.39583	26.38715
11	1997	NaN	56.54167	25.48143
12	1998	18.26467	50.66250	24.58649
13	1999	18.49646	57.48864	24.66667
14	2000	16.93806	55.76103	23.46082
15	2001	16.92632	51.81984	25.06522
16	2002	15.27335	54.88043	22.73750
17	2003	15.23183	56.16608	24.62500
18	2004	14.62864	44.48240	23.39130
19	2005	16.18556	58.84126	22.62387

Note:
- NaN ("Not a Number") means 0/0
- NA ("Not Available") is generally interpreted as a missing value and has various forms – NA_integer_, NA_real_, etc.

The "dplyr" package

group_by() & summarize(): What are the average levels of ozone (o3) and nitrogen dioxide (no2) within quintiles of pm25?

First, we create a categorical variable of pm25 divided into quintiles.

```
qq <- quantile(Chicago$pm25, seq(0, 1, 0.2), na.rm = TRUE)
Chicago <- mutate(Chicago, pm25.quint = cut(pm25, qq))
```

Two new functions:
1. **quantile()**
2. **cut()**

Now we can group the data frame by the pm25.quint variable.

```
quint <- group_by(Chicago, pm25.quint)
```

Finally, we can compute the mean of o3 and no2 within quintiles of pm25.

```
> summarize(quint, o3 = mean(o3tmean2, na.rm = TRUE),  no2 = mean(no2tmean2, na.rm = TRUE))
```

The "dplyr" package

group_by() & summarize(): **What are the average levels of ozone (o3) and nitrogen dioxide (no2) within quintiles of pm25?**

```
summarize(quint, o3 = mean(o3tmean2, na.rm = TRUE),  no2 = mean(no2tmean2, na.rm = TRUE))
```

	pm25.quint	o3	no2
1	(1.7,8.7]	21.66401	17.99129
2	(8.7,12.4]	20.38248	22.13004
3	(12.4,16.7]	20.66160	24.35708
4	(16.7,22.6]	19.88122	27.27132
5	(22.6,61.5]	20.31775	29.64427
6	NA	18.79044	25.77585

Observations:
1. There isn't a strong relationship between pm25 and o3.
2. There appears to be a slightly positive correlation between pm25 and no2.

The "dplyr" package

%>%(): **Very handy for stringing together multiple dplyr functions in a sequence of operations. Avoids the use of nesting to improve readability of code.**

```
#The nested line of code below
third(second(first(x)))

#becomes
 first(x) %>% second %>% third
```

The "dplyr" package

%>%(): Very handy for stringing together multiple dplyr functions in a sequence of operations. Avoids the use of nesting to improve readability of code.

The last example can be rewritten as:

```
mutate(Chicago, pm25.quint = cut(pm25, qq)) %>% group_by(pm25.quint)  %>% summarize(o3 =
mean(o3tmean2, na.rm = TRUE), no2 = mean(no2tmean2, na.rm = TRUE))
```

```
   pm25.quint       o3      no2
1   (1.7,8.7] 21.66401 17.99129
2  (8.7,12.4] 20.38248 22.13004
3 (12.4,16.7] 20.66160 24.35708
4 (16.7,22.6] 19.88122 27.27132
5 (22.6,61.5] 20.31775 29.64427
6          NA 18.79044 25.77585
```

- We don't have to create a set of temporary variables
- The chicago data frame is passed only once
- The first argument of the next function is taken to be the output of the previous element in the pipeline.

The "dplyr" package

Homework:

1. Read the chicago_data into the variable "chicago".
2. Use histograms to examine the completeness of the data on ther following variables:
   a) tmpd
   b) dptp
   c) pm25tmeans2
   d) pm10tmeans2
   e) o3means2
   f) no2mean2
3. What is your opinion as to how the missing date might bias any analysis?
4. Using group_by and summarize, create an analysis to determine the seasonality pattern in o3 and no2 by temperature deciles.
5. Using group_by and summarize, create an analysis to determine the seasonality pattern in o3 and no2 by dew point deciles.
6. What, if anything, can you conclude from the analyses?

Exploratory Data Analysis Checklist

A "checklist" of things to do when embarking on an exploratory data analysis.

1. Understand the question(s) the data was designed to answer.
2. Determine structure of applicable data set using:
   a. str()
   b. head()
   c. tail()
   d. summary()
   e. dim()
   f. nrow()
   g. ncol()
3. Determine the "messiness" of your data with respect to:
   a. Determine how much is missing by examining counts
   b. Variable data consistent with data type
   c. Amount of garbage data in records
   d. Determine number of duplicate rows
4. Identify reconciliation totals to verify data counts

Exploratory Data Analysis Checklist

A "checklist" of things to do when embarking on an exploratory data analysis.

5. Make sure you have a data dictionary for the data set.
6. Determine the unique values on all categorical data to make sure they are consistent with variable definition. You can use **unique(variablename)** function.
7. Use basic plots functions to understand your data, such as:
   a. Histograms & Barplots
   b. Scatterplots
   c. Pairs() & Coplot()
   d. Boxplots
   e. Density plot
8. Validate your dataset with an external source.
9. Challenge the results you are seeing for reasonableness and biases. Are there unreasonable distributions across categoric, discrete, and continuous variables?
10. Determine any follow-up questions regarding data set.
11. Do you need to collect more data?

Exploratory Data Analysis Checklist

The hourly_44201_2014.csv data set can be found at: https://aqs.epa.gov/aqsweb/airdata/download_files.html

```
library(data.table)
ozone<-fread("C:/Rdata/hourly_44201_2014.csv")
attach(ozone)
str(ozone)
head(ozone)
tail(ozone)
names(ozone)
names(ozone)[22]<-paste("StateName")
names(ozone)[14]<-paste("SampleMeasurement")
names(ozone)[23]<-paste("CountyName")
attach(ozone)
table(ozone$"Time Local")
library(dplyr)
unique(ozone$"State Name")
select(ozone, "State Name") %>% unique %>% nrow
quantile(ozone$ "Sample Measurement ", seq(0, 1, 0.1))
ranking <- group_by(ozone, StateName, CountyName)%>%summarize(ozone = mean(SampleMeasurement),
n=n())#%>% as.data.frame%>%arrange(desc(ozone1))
```

Note:

1. The data set has over 9 million records.
2. There are 24 variables
3. It covers 53 states (???)
4. Study how the ranking line of code was written.

The "dplyr" package

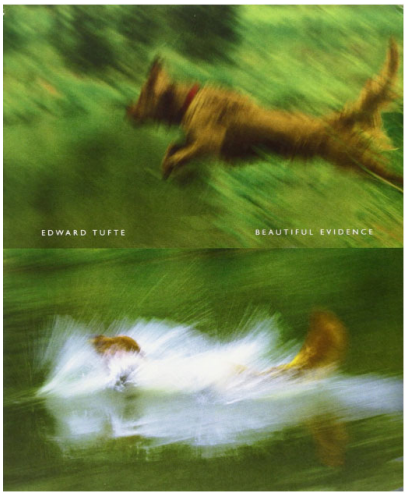
Homework:

1. Attempt to read in the data set in Slide 19 and run the code snippet.
2. Was your system able to load the data set and run the code?

Principles of Analytic Graphics

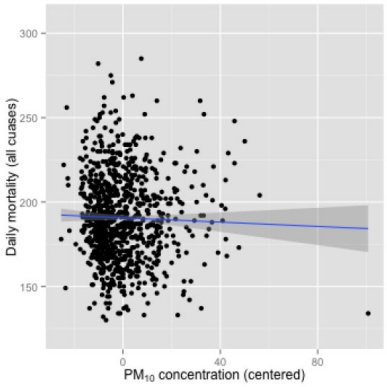
Six principles to creating informative and useful graphics

1. Show comparisons
2. Show causality, mechanism, explanation, systematic structure
3. Show multivariate data
4. Integrate evidence
5. Describe and document the evidence
6. Content, Content, Content

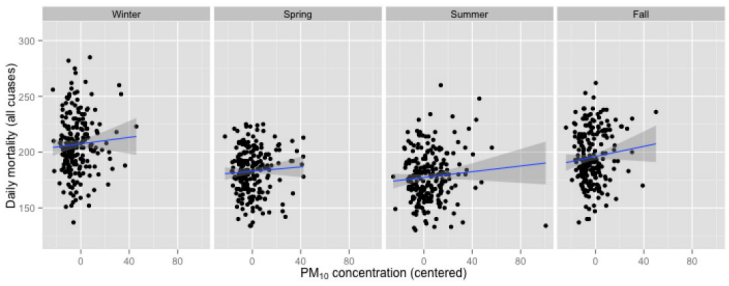


www.edwardtufte.com

Principles of Analytic Graphics



PM10 and mortality in New York City



PM10 and mortality in New York City by season

Simpson's Paradox: A phenomenon in probability and statistics, in which a trend appears in several different groups of data but disappears or reverses when these groups are combined.

Principles of Analytics Graphics

Homework:

1. Watch the following videos:
   a. https://www.youtube.com/watch?v=6lOvA_y7p7w&feature=youtu.be
   b. https://www.youtube.com/watch?v=ebEkn-BiW5k (Ignore the last 50 seconds)

   You may have to copy the link into a browser if clicking on the link does not work.

Getting Started with GGPlot2

Download packages with the install() command

```
install.packages("ggplot2")
install.packages("ggthemes")
```

Make packages available with library()

```
library(ggplot2)
library(ggthemes)
```

Get help with ?

```
?ggplot
```

Special Note:
>data(package = .packages(all.available = TRUE)) #Will list all the data available across all installed packages.

Getting Started with GGPlot2

Grammar of Graphics - GGPlot2

What is Grammar of Graphics?

The grammar tells us:

- A description of the deep features that underlie all statistical graphics.
- A statistical graphic is a mapping from data to aesthetic attributes (color, shape, size) of geometric objects (points, lines, bars).
- The plot may also contain statistical transformations of the data and is drawn on a specific coordinate system.
- Faceting can be used to generate the same plot for different subsets of the dataset.

It is the combination of these independent components that make up a graphic.

Getting Started with GGPlot2

Grammar of Graphics: Basic Plot Components

All plots are made up of

1. Data: Elements to plot
2. Aesthetic Mappings: Mapping data to perceivable data attributes
3. Layers: Geometric objects (**geoms**) and statistical transformations (**stats**),
4. Scales: Legends, axes, gridlines, etc.
5. Coordinate System: Cartesian, polar, map projections, etc.
6. Facet Specifications: Latticing, trellising to subset data to display more effectively
7. Theme: Font size, background color, etc.

Getting Started with GGPlot2

Grammar of Graphics - GGPlot2

What is Grammar of Graphics?

The Three Key Components

1. Data
2. A set of aesthetic mappings between variables in the data and visual properties
3. At least one layer which describes how to render each observation. Layers are usually created with a geom function.

Getting Started with GGPlot2

The MPG Dataset: It is included with GGPlot2

```
>str(ggplot2::mpg)
Classes 'tbl_df', 'tbl' and 'data.frame':       234 obs. of  11 variables:
 $ manufacturer: chr  "audi" "audi" "audi" "audi" ...
 $ model       : chr  "a4" "a4" "a4" "a4" ...
 $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
 $ year        : int  1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
 $ cyl         : int  4 4 4 4 6 6 6 4 4 4 ...
 $ trans       : chr  "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
 $ drv         : chr  "f" "f" "f" "f" ...
 $ cty         : int  18 21 20 21 16 18 18 18 16 20 ...
 $ hwy         : int  29 29 31 30 26 26 27 26 25 28 ...
 $ fl          : chr  "p" "p" "p" "p" ...
 $ class       : chr  "compact" "compact" "compact" "compact" ...
```

Getting Started with GGPlot2

The MPG Data Dictionary

1. cty and hwy record miles per gallon (mpg) for city and highway driving.
2. displ is the engine displacement in litres.
3. drv is the drivetrain: front wheel (f), rear wheel (r) or four wheel (4).
4. model is the model of car. There are 38 models, selected because they had a new edition every year between 1999 and 2008.
5. class is a categorical variable describing the "type" of car: two-seater, SUV, compact, etc.
6. trans is the type of transmission.
7. manufacturer is the maker of the car
8. fl is the fuel type
9. year is model year
10. cyl is number of cylinders

```
Classes 'tbl_df', 'tbl' and 'data.frame':       234 obs. of  11 variables:
 $ manufacturer: chr  "audi" "audi" "audi" "audi" ...
 $ model       : chr  "a4" "a4" "a4" "a4" ...
 $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
 $ year        : int  1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
 $ cyl         : int  4 4 4 4 6 6 6 4 4 4 ...
 $ trans       : chr  "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
 $ drv         : chr  "f" "f" "f" "f" ...
 $ cty         : int  18 21 20 21 16 18 18 18 16 20 ...
 $ hwy         : int  29 29 31 30 26 26 27 26 25 28 ...
 $ fl          : chr  "p" "p" "p" "p" ...
 $ class       : chr  "compact" "compact" "compact" "compact" ...
```


Getting Started with GGPlot2

The MPG Dataset: It is included with GGPlot2

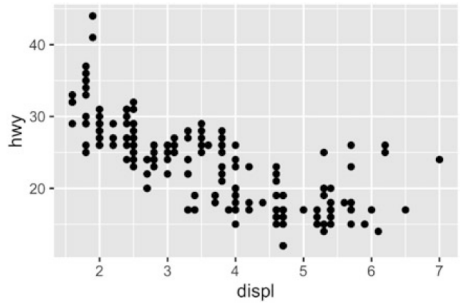
Here's a simple example:

```
mpg<-ggplot2::mpg
attach(mpg)
ggplot(mpg, aes(x = displ, y = hwy)) + geom_point()
```

Pattern of Function Call: ggplot + elements to layer on top

Notice the three elements:

1. The data: mpg
2. The aesthetics: x is mapped to engine size and y to highway fuel economy
3. Layers: Points are layered on the basic default GG plot region



Question: What is graph telling us about engine size & fuel economy?

Getting Started with GGPlot2

GGThemes

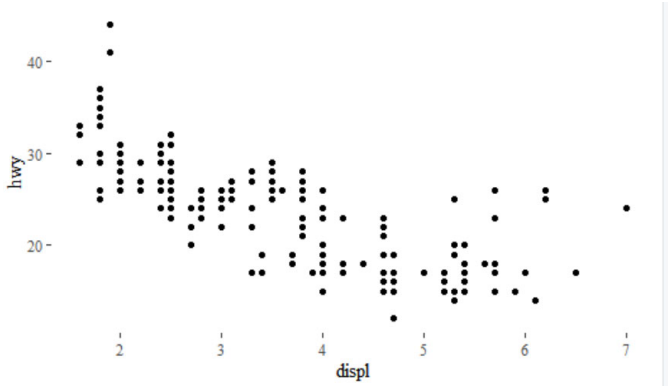
theme_base: a theme resembling the default base graphics in R. See also theme_par.
theme_calc: a theme based on LibreOffice Calc.
theme_economist: a theme based on the plots in the The Economist magazine.
theme_excel: a theme replicating the classic ugly gray charts in Excel
theme_few: theme from Stephen Few's "Practical Rules for Using Color in Charts".
theme_fivethirtyeight: a theme based on the plots at fivethirtyeight.com.
theme_gdocs: a theme based on Google Docs.
theme_hc: a theme based on Highcharts JS.
theme_par: a theme that uses the current values of the base graphics parameters in par.
theme_pander: a theme to use with the pander package.
theme_solarized: a theme using the solarized color palette.
theme_stata: themes based on Stata graph schemes.
theme_tufte: a minimal ink theme based on Tufte's The Visual Display of Quantitative Information.
theme_wsj: a theme based on the plots in the The Wall Street Journal.

35

Getting Started with GGPlot2

GGTheme – Edward Tufte

```
ggplot(mpg, aes(x = displ, y = hwy)) + theme_tufte() + geom_point()
```

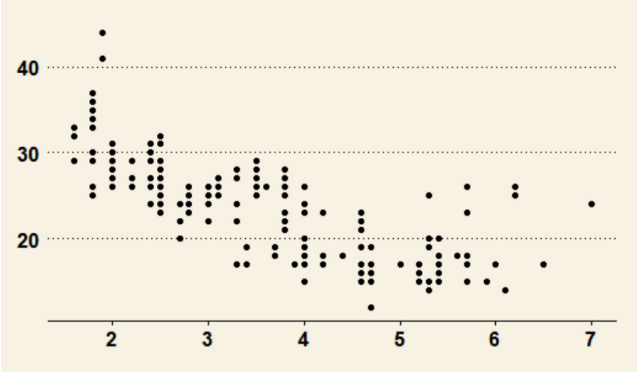


36

Getting Started with GGPlot2

GGTheme – Wall Street Journal

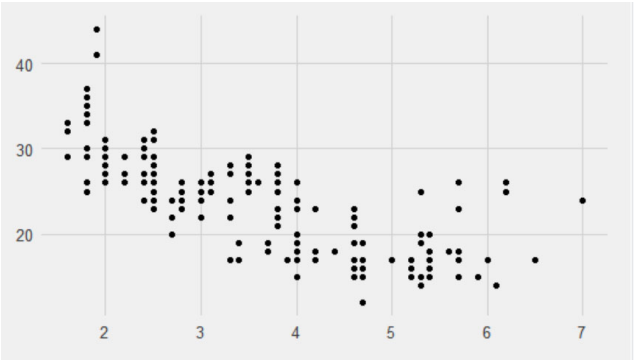
```
ggplot(mpg, aes(x = displ, y = hwy)) + theme_wsj() + geom_point()
```



Getting Started with GGPlot2

GGTheme

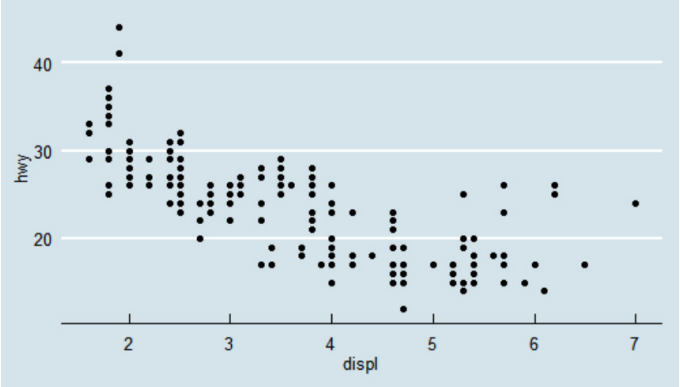
```
ggplot(mpg, aes(x = displ, y = hwy)) + theme_ fivethirtyeight() + geom_point()
```



Getting Started with GGPlot2

GGTheme – The Economist

```
ggplot(mpg, aes(x = displ, y = hwy)) + theme_ economist() + geom_point()
```



Getting Started with GGPlot2

Homework:
Data sets: mpg, diamonds, economics – Included in GGPLOT2

1. How would you describe the relationship between cty and hwy? Do you have any concerns about drawing conclusions from that plot?
2. What does ggplot(mpg, aes(model, manufacturer)) + geom point() show? Is it useful? How could you modify the data to make it more informative?
3. Describe the data, aesthetic mappings and layers used for each of the following plots. See if you can predict what the plot will look like before running the code.
   - ggplot(mpg, aes(cty, hwy)) + geom point()
   - ggplot(diamonds, aes(carat, price)) + geom point()
   - ggplot(economics, aes(date, unemploy)) + geom line()
   - ggplot(mpg, aes(cty)) + geom histogram()

Getting Started with GGPlot2

Color, Size, Shape and Other Aesthetic Attributes

To add additional variables to a plot, we can use other aesthetics like color, shape, and size. These work in the same way as the x and y aesthetics, and are added into the aes() function:

- aes(displ, hwy, color = class)
- aes(displ, hwy, shape = drv)
- aes(displ, hwy, size = cyl)

```
unique(mpg$class)
[1] "compact"    "midsize"    "suv"        "2seater"    "minivan"    "pickup"     "subcompact"

unique(mpg$drv)
[1] "f" "4" "r"

unique(mpg$cyl)
[1] 4 6 8 5
```

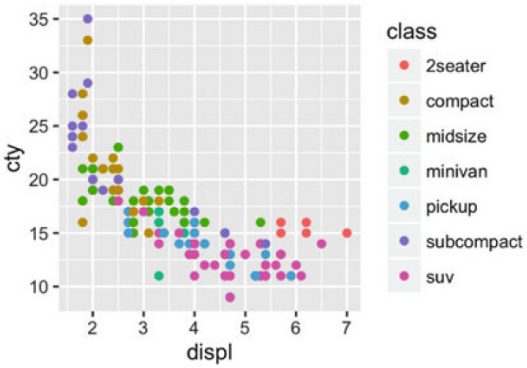

Getting Started with GGPlot2

Let's try some

```
ggplot(mpg, aes(displ, cty, color = class)) + geom_point()
```

Note: We did not make an election for he coloring of each point. GGPlot makes the election. We can override the default selections.

The legend allows us to read data values from the color, showing us that the group of cars with unusually high fuel economy for their engine size are two seaters: cars with big engines, but lightweight bodies.

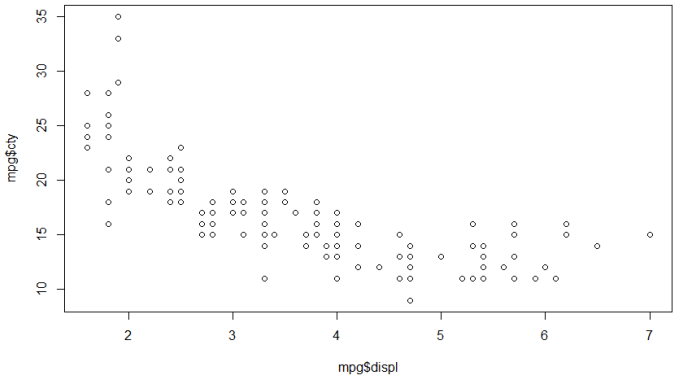


Getting Started with GGPlot2

Let's compare to plot()

```
plot(mpg$displ,mpg$cty)
```

There is a big difference in quality between plot() and ggplot() in terms of the appearance of the two graphs.



Getting Started with GGPlot2

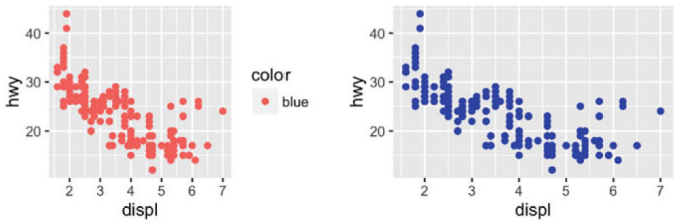
The aes() function can be an argument of the geom() function

If you want to set an aesthetic to a fixed value, without scaling it, do so in the individual layer outside of aes(). Compare the following two plots:

```
ggplot(mpg, aes(displ, hwy)) + geom_point(aes(color = "blue"))
ggplot(mpg, aes(displ, hwy)) + geom_point(color = "blue")
```

When color is an argument of aes(), R thinks you want to plot the attribute blue and will give it a default color.

Use the color parameter outside the aes() function for the desired result.



Getting Started with GGPlot2

Homework

1. How is drive train related to fuel economy? How is drive train related to engine size and class? Create plots to support your answers.

Getting Started with GGPlot2

Facetting

Facetting creates tables of graphics by splitting the data into subsets and displaying the same graph for each subset.

```
ggplot(mpg, aes(displ, hwy)) + geom_point() + facet_wrap(~class)
```

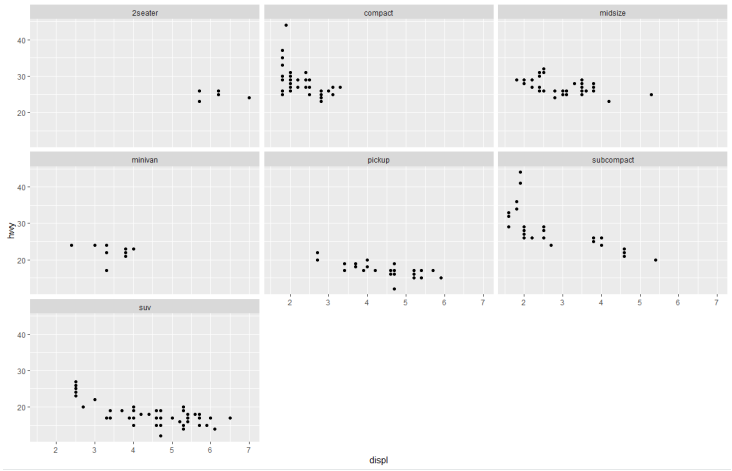
There will be a graph for each level of class.

Recall the levels of class are:
"compact"
"midsize"
"suv"
"2seater"
"minivan"
"pickup"
"subcompact"

Getting Started with GGPlot2

Facetting

```
>ggplot(mpg, aes(displ, hwy)) + geom_point() +    facet_wrap(~class)
```



Getting Started with GGPlot2

Homework

1. What happens if you try to facet by a continuous variable like hwy? What about cyl? What's the key difference?
2. Use facetting to explore the three-way relationship between fuel economy, engine size, and number of cylinders. How does facetting by number of cylinders change your assessment of the relationship between engine size and fuel economy?
3. Read the documentation for facet wrap(). What arguments can you use to control how many rows and columns appear in the output?
4. What does the scales argument to facet wrap() do? When might you use it?

Getting Started with GGPlot2

Geom Functions

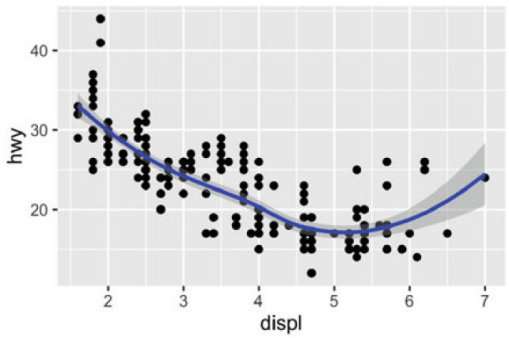
1. geom_smooth() fits a smoother to the data and displays the smooth LINE and its standard error.
2. geom_boxplot() produces a box-and-whisker plot to summarize the distribution of a set of points.
3. geom_histogram() and geom_freqpoly() show the distribution of continuous variables.
4. geom_bar() shows the distribution of categorical variables.
5. geom_path() and geom_line() draw lines between the data points. A line plot is constrained to produce lines that travel from left to right, while paths can go in any direction. Lines are typically used to explore how things change over time.

Getting Started with GGPlot2

Adding a Smoother to a Plot

If you have a scatterplot with a lot of noise, it can be hard to see the dominant pattern. In this case it's useful to add a smoothed line to the plot with geom_smooth():

ggplot(mpg, aes(displ, hwy)) + geom_point() + geom_smooth()



This overlays the scatterplot with a smooth curve, including an assessment of uncertainty in the form of point-wise confidence intervals shown in grey. If you're not interested in the confidence interval, turn it off with

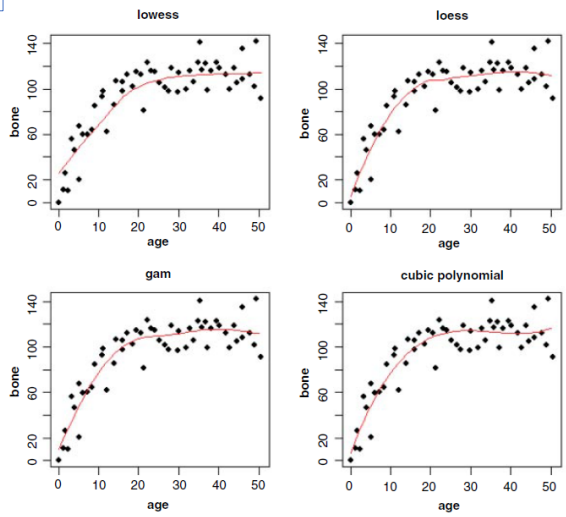
>geom_smooth(se = FALSE).

Stands for "Standard Error"

Getting Started with GGPlot2

Arguments to the geo_smooth() Function: Method

1. Lowess stands for "Locally Weighted Scatterplot Smoothing" and creates a smooth curve through a scatter of points.
2. Loess stands for "Local Regression" and is the most common method used to smooth a volatile time series. It is a non-parametric method where least squares regression is performed in localized subsets, which makes it a suitable candidate for smoothing any numerical vector.
3. GAM stands for "Generalized Additive Model." They are used to fit "wiggly" data and are basically linear models with a smoothing term. Uses splines.
4. Cubic Polynomial – A third degree polynomial.



51

Getting Started with GGPlot2

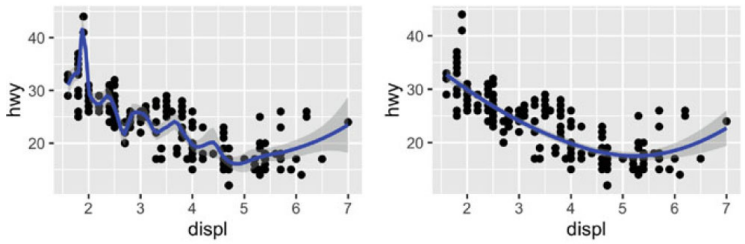
Arguments to the geo_smooth() Function: Method

method = "loess", the default for small n, uses a smooth local regression (as described in ?loess).

The wiggliness of the line is controlled by the span parameter, which ranges from 0 (exceedingly wiggly) to 1 (not so wiggly).

```
ggplot(mpg, aes(displ, hwy)) + geom_point() + geom_smooth(span = 0.2)
ggplot(mpg, aes(displ, hwy)) + geom_point() + geom_smooth(span = 1)
```

Loess does not work well for large datasets, so an alternative smoothing algorithm is used when *n* is greater than 1000.



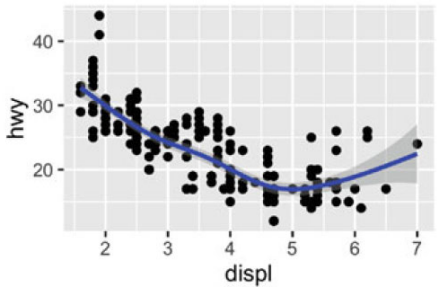
52

Getting Started with GGPlot2

method = "gam"

Fits a generalized additive model provided by the **mgcv** package. You need to first load mgcv, then use a formula like formula = y ~ s(x) or y ~ s(x, bs = "cs") (for large data).

```
library(mgcv)
ggplot(mpg, aes(displ, hwy)) + geom_point() + geom_smooth(method = "gam", formula = y ~ s(x))
```

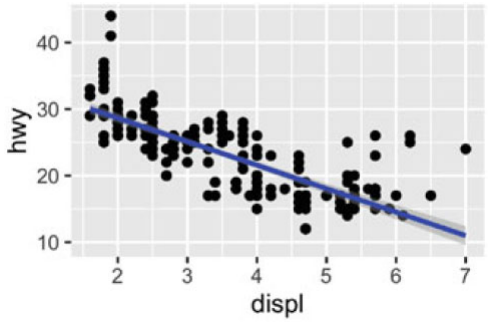


Getting Started with GGPlot2

method = "lm"

Fits a linear model, giving the line of best fit

```
ggplot(mpg, aes(displ, hwy)) + geom_point() + geom_smooth(method = lm)
```

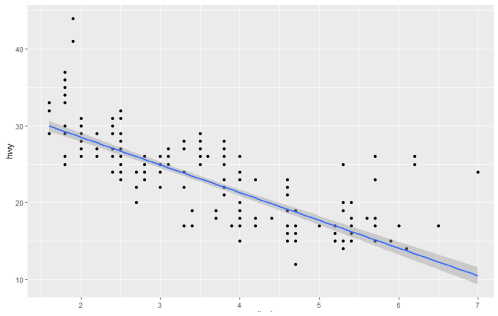


Getting Started with GGPlot2

method ="rlm"

Uses a robust fitting algorithm so that outliers don't affect the fit as much. It's part of the MASS package, so remember to load that first.

```
library(MASS)
ggplot(mpg, aes(displ, hwy)) + geom_point() + geom_smooth(method = rlm)
```



Getting Started with GGPlot2

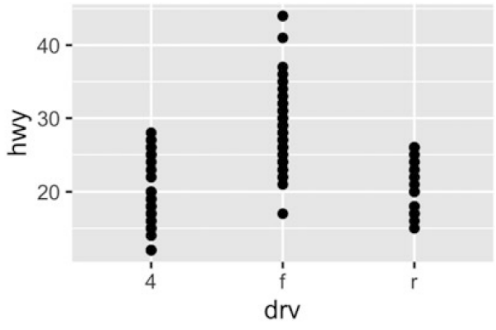
Boxplots and Jittered Points

When a set of data includes a categorical variable and one or more continuous variables, you will probably be interested to know how the values of the continuous variables vary with the levels of the categorical variable.

We might be interested in seeing how fuel economy varies within drv class. Start with a scatterplot like this:

```
ggplot(mpg, aes(drv, hwy)) + geom_point()
```

Many points are plotted in the same location, and it's difficult to see the distribution.



Getting Started with GGPlot2

Three techniques to deal with overplotting

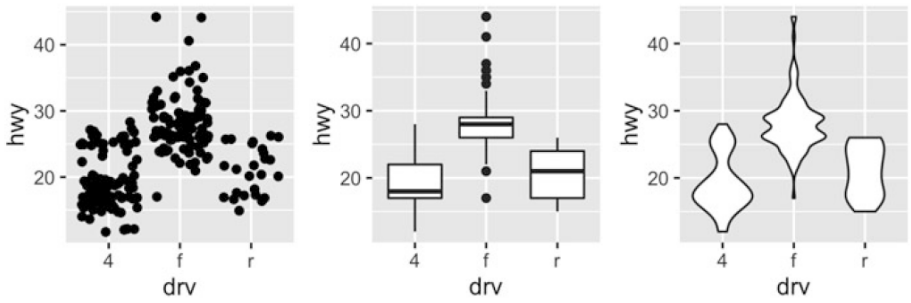
- **Jittering**, geom_ jitter(), adds a little random noise to the data which can help avoid overplotting.
- **Boxplots**, geom_boxplot(), summarize the shape of the distribution with a handful of summary statistics.
- **Violin plots**, geom_violin(), show a compact representation of the "density" of the distribution, highlighting the areas where more points are found.

Getting Started with GGPlot2

Three techniques to deal with overplotting

```
ggplot(mpg, aes(drv, hwy)) + geom_jitter()
ggplot(mpg, aes(drv, hwy)) + geom_boxplot()
ggplot(mpg, aes(drv, hwy)) + geom_violin()
```

For jittered points, geom_ jitter() offers the same control over aesthetics as geom_point(): size, colour, and shape. For geom_boxplot() and geom_violin(), you can control the outline colour or the internal fill colour.

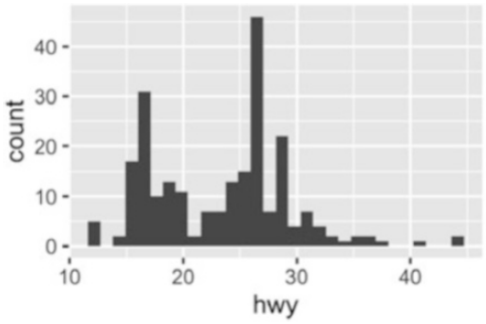


Getting Started with GGPlot2

Histograms and Frequency Polygons

Histograms and frequency polygons show the distribution of a single numeric variable. They provide more information about the distribution of a single group than boxplots do, at the expense of needing more space.

```
ggplot(mpg, aes(hwy)) + geom_histogram()
#stat_bin() using bins = 30. Pick better value with
#binwidth.
```

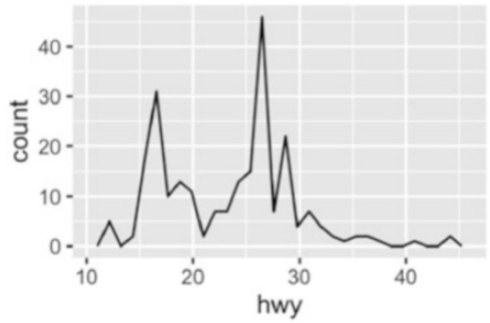


Getting Started with GGPlot2

Histograms and Frequency Polygons

Histograms and frequency polygons show the distribution of a single numeric variable. They provide more information about the distribution of a single group than boxplots do, at the expense of needing more space.

```
ggplot(mpg, aes(hwy)) + geom_freqpoly()
#stat_bin() using bins = 30. Pick better value with
#binwidth.
```



Getting Started with GGPlot2

Histograms and Frequency Polygons

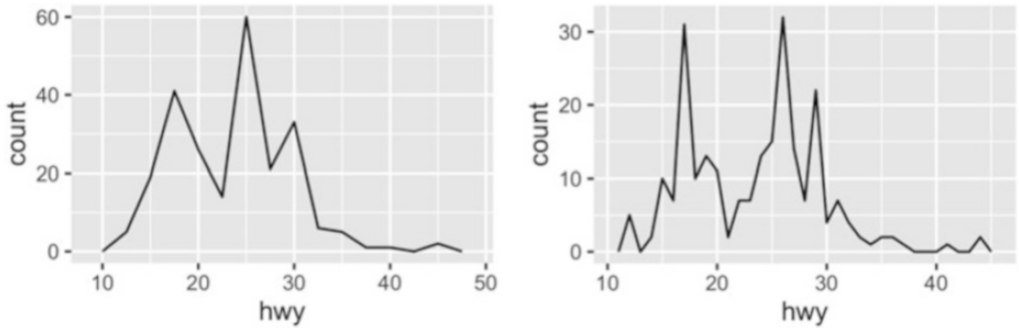
Histograms and frequency polygons work in the same way: they **bin** the data, then count the number of observations in each bin.

You can control the width of the bins with the binwidth argument (if you don't want evenly spaced bins you can use the breaks argument). It is **very important** to experiment with the bin width. The default just splits your data into 30 bins, which is unlikely to be the best choice.

Getting Started with GGPlot2

Histograms and Frequency Polygons

```
ggplot(mpg, aes(hwy)) + geom_freqpoly(binwidth = 2.5)
ggplot(mpg, aes(hwy)) + geom_freqpoly(binwidth = 1)
```



Getting Started with GGPlot2

An Alternative Function: geom_density()

Complexities with density plots:

- They are harder to interpret since the underlying computations are more complex.
- They also make assumptions that are not true for all data, namely that the underlying distribution is continuous, unbounded, and smooth.

Getting Started with GGPlot2

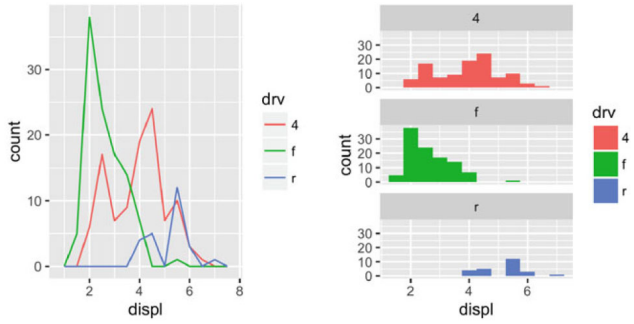
An Alternative Function: geom_density()

```
library(ggplot2)
m <- matrix(data=cbind(rnorm(10000, 0), rnorm(10000, 2),
        rnorm(10000, 5)), nrow=30, ncol=3)
colnames(m) <- c('method1', 'method2', 'method3')
head(m)
df <- as.data.frame(m)
dfs <- stack(df)
is.factor(dfs[,2])
ggplot(dfs, aes(x=values)) +
  geom_density(aes(group=ind, color=ind,
fill=ind),alpha=0.3)
```

Getting Started with GGPlot2

Using Facetting

```
ggplot(mpg, aes(displ, color = drv)) + geom_freqpoly(binwidth = 0.5)
ggplot(mpg, aes(displ, fill = drv)) + geom_histogram(binwidth = 0.5) + facet_wrap(~drv, ncol = 1)
```

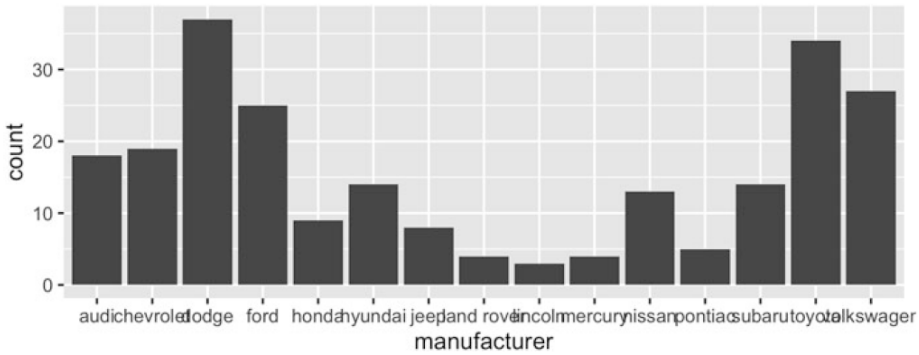


65

Getting Started with GGPlot2

Bar Charts

The discrete analogue of the histogram is the bar chart, geom_bar(). It's easy to use:



66

Getting Started with GGPlot2

Two Data Organizations for Bar Charts

1. Un-Summarized Data: Raw data elements that require binning. geo_bar() will run default stats and counts on the data
2. Summarized Data: Is already summarized and geo_bar() needs to be told not the run default stats and counts.

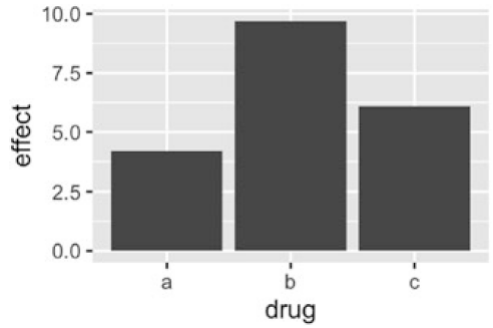
For example, you might have three drugs with their average effect:

```
drugs <- data.frame(drug = c("a", "b", "c"), effect = c(4.2, 9.7, 6.1))
ggplot(drugs, aes(drug, effect)) + geom_bar(stat = "identity")
ggplot(drugs, aes(drug, effect)) + geom_point()
```

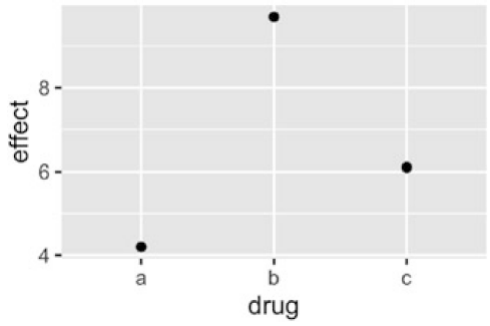

Getting Started with GGPlot2

Two Data Organizations for Bar Charts

>ggplot(drugs, aes(drug, effect)) + geom_bar(stat = "identity")



Note: Needs to start at zero.



>ggplot(drugs, aes(drug, effect)) + geom_point()

Note: Using geom_point() uses less space and does not need to start at zero.

Getting Started with GGPlot2

Time Series with Line and Path Plots

Line and path plots are typically used for time series data. Line plots join the points from left to right. Path plots join them in the order that they appear in the dataset.

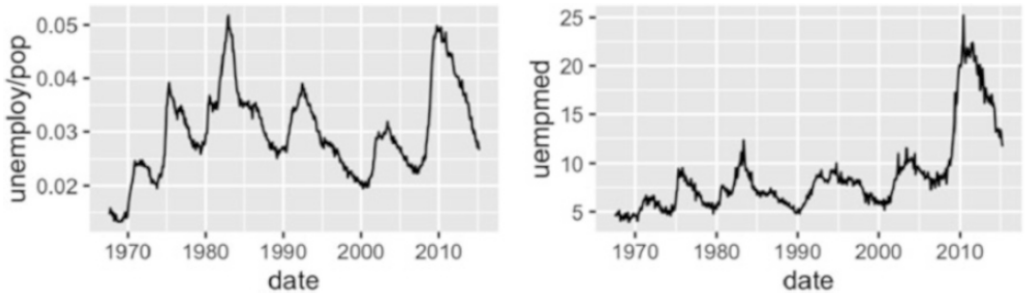
Line plots usually have time on the x-axis, showing how a single variable has changed over time.

Path plots show how two variables have simultaneously changed over time, with time encoded in the way that observations are connected.

Getting Started with GGPlot2

Example

```
ggplot(economics, aes(date, unemploy / pop)) + geom_line()
ggplot(economics, aes(date, uempmed)) + geom_line()
```



Getting Started with GGPlot2

Plotting on the same graph

The dataset is called economics it is in the ggplot2 package

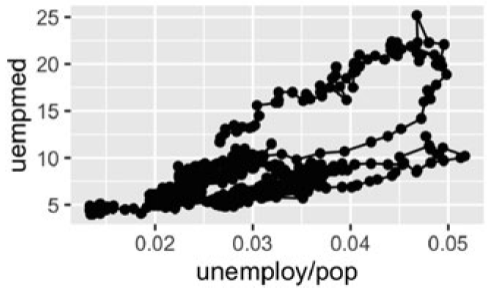
```
ggplot(economics, aes(unemploy / pop, uempmed)) + geom_path() + geom_point()

year <- function(x) as.POSIXlt(x)$year + 1900

ggplot(economics, aes(unemploy / pop, uempmed)) + geom_path(colour = "grey50") +
geom_point(aes(colour = year(date)))

Unemployment Rate = unemploy / pop
Median Number of Weeks Unemployed = uempmed
```

Note: The as.POSIXlt(x) function can extract parts of a date or time.

Getting Started with GGPlot2

Plotting on the same graph

```
ggplot(economics, aes(unemploy / pop, uempmed)) + geom_path() + geom_point()
```

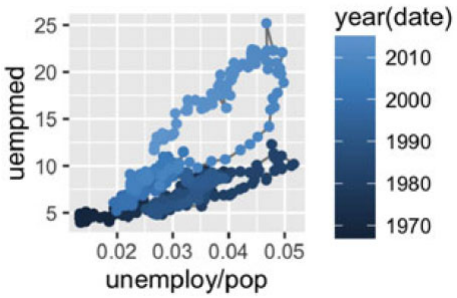


Getting Started with GGPlot2

Plotting on the same graph

```
year <- function(x) as.POSIXlt(x)$year + 1900

ggplot(economics, aes(unemploy / pop, uempmed)) + geom_path(color = "grey50") +
geom_point(aes(color = year(date)))
```

We can see that unemployment rate and length of unemployment are highly correlated, but in recent years the length of unemployment has been increasing relative to the unemployment rate.



Getting Started with GGPlot2

Homework

1. Use the data set AutoClaims.csv to study claims in the first quarter of 2015 as a time series. You are trying to analyze the effect of month on the time series. Create a graph the shows a path plot with points and colors data for each month a different data. Analyze as a total_claim_amount as a function policy_annual_premium.

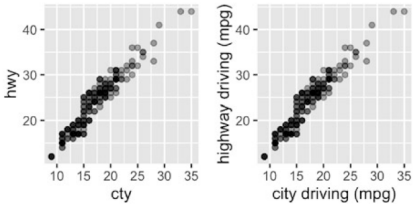
We will be using this data set for other analyses. It a data set from the Kaggle website.

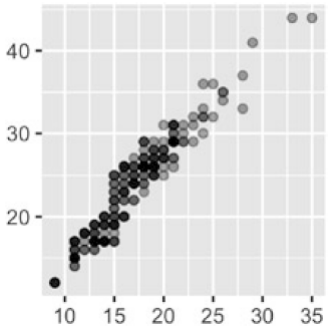
Getting Started with GGPlot2

Modifying the Axes

Two families of useful helpers let you make the most common modifications. xlab() and ylab() modify the x- and y-axis labels:

```
ggplot(mpg, aes(cty, hwy)) + geom_point(alpha = 1 / 3)
ggplot(mpg, aes(cty, hwy)) + geom_point(alpha = 1 / 3) + xlab("city driving (mpg)") + ylab("highway driving (mpg)")

# Remove the axis labels with NULL
ggplot(mpg, aes(cty, hwy)) + geom_point(alpha = 1 / 3) + xlab(NULL) + ylab(NULL)
```



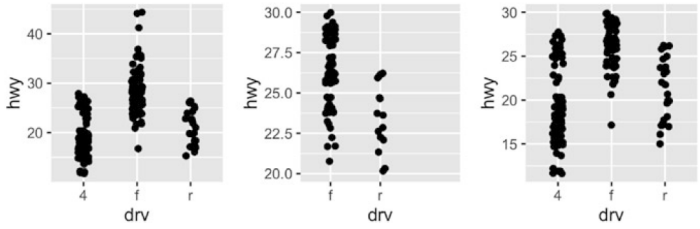


Getting Started with GGPlot2

Modifying the Axes

xlim() and ylim() modify the limits of axes:

```
ggplot(mpg, aes(drv, hwy)) + geom_jitter(width = 0.25)
ggplot(mpg, aes(drv, hwy)) + geom_jitter(width = 0.25) + xlim("f", "r") + ylim(20, 30)

#Warning: Removed 138 rows containing missing values (geom_point).
#For continuous scales, use NA to set only one limit

ggplot(mpg, aes(drv, hwy)) + geom_jitter(width = 0.25, na.rm = TRUE) + ylim(NA, 30)
```



Changing the axes limits sets values outside the range to NA.

na.rm = TRUE: Suppresses the warning

Getting Started with GGPlot2

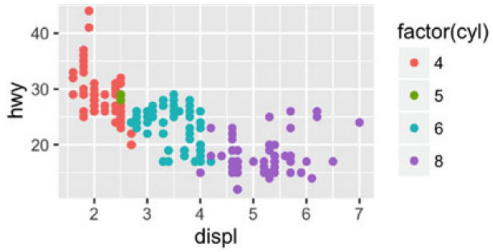
Saving the plot to a variable

```
p <- ggplot(mpg, aes(displ, hwy, color = factor(cyl))) + geom_point()
```

Once you have a plot object, there are a few things you can do with it:

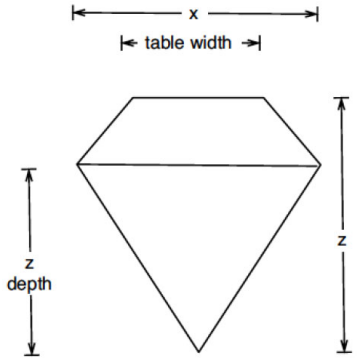
1. Render it on screen with print(). Great if you want to put plots in a loop and display them. Remember the animated bubble plot?

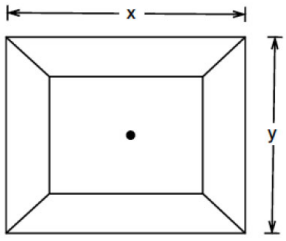
```
print(p)
```



Getting Started with GGPlot2

The Diamonds Dataset





depth = z depth / z * 100
table = table width / x * 100

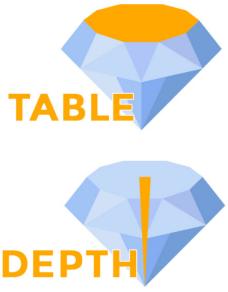
Getting Started with GGPlot2

When a gemologist determines the value of a diamond, he or she considers a number of different factors:

The 4C's

How large is the diamond? (Carat)
Was the stone cut well? (Cut)
Is it colorless or does it have a slight hue of color? (Color)
Are there any visible inclusions in the diamond? (Clarity)

TWO FACTORS THAT AFFECT CUT:



79

Getting Started with GGPlot2

A Diamond's Table

Table percentage is calculated by dividing the width of the table by the overall width of the diamond.

The ideal table percentage will vary based on the shape of your diamond.

The ideal table for round-cut diamonds is between 54-60 percent, while the ideal dimensions for emerald-cut diamonds puts table percentage between 66-72 percent.

80

Getting Started with GGPlot2

A Diamond's Depth

The depth of a diamond might also be called the "height": it is the distance from the table to the culet (the pointed tip) of the diamond.

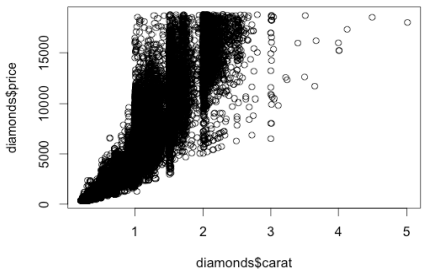
Jewelers grade a diamond's depth based on its depth percentage. Depth percentage is the diamond's depth divided by the width of the diamond. This percentage dictates the overall proportions of the diamond, which in turn **directly impact how light reflects off the facets in the stone**.

The ideal depth will also vary based on shape. Look for a diamond that allows light to bounce around the facets, creating that gorgeous, eye-catching sparkle.

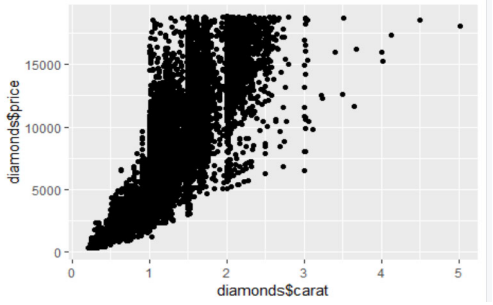
Getting Started with GGPlot2

The Diamond Dataset: Plot() vs. QPlot()

```
plot(diamonds$carat,diamonds$price)
```




```
qplot(diamonds$carat,diamonds$price)
```



Unless otherwise specified, qplot() tries to pick a sensible geometry and statistic based on the arguments provided. For example, if you give qplot() x and y variables, it'll create a scatterplot. If you just give it an x, it'll create a histogram or bar chart depending on the type of variable.

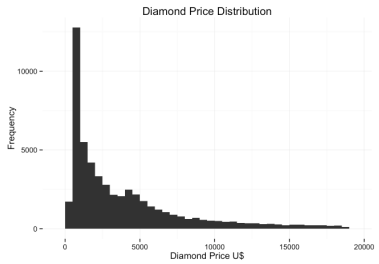
Getting Started with GGPlot2

Homework: The Diamonds Dataset

The Diamonds data set come with ggplot2

1. Display the first 6 rows of the dataset
2. Display the last 6 rows of the dataset
3. Display a summary of the dataset
4. Display the structure of the dataset
5. Display the dimension of the data set

6. Plot a basic histogram of the prices of the dataset using ggplot
7. Replot with a binwidth of 500
   a. The title “Diamond Price Distribution”
   b. X axis label “Diamond Price U$”
   c. Y axis label Frequency



Getting Started with GGPlot2

Homework

8. Plot a basic histogram of the prices of the dataset using ggplot with the following features:

   a. A binwidth of 100
   b. The title “Diamond Price Distribution”
   c. X axis label “Diamond Price U$”
   d. Y axis label Frequency
   e. Facet based on cut

Getting Started with GGPlot2

Homework

9. Calculate the following:
   a. Mean of prices
   b. Median of prices
   c. Maximum price
   d. Minimum price
   e. Count where cut = “Fair”
   f. Count where cut = “Good”
   g. Count where cut = “Very Good”
   h. Count where cut = “Ideal”
   i. Contingency Table for cut

85

Getting Started with GGPlot2

Homework

10. Answer the following questions:
    a. How many cost <= $500
    b. How many cost <= $250
    c. How many cost at least $15,000

11. Display the records for the highest priced diamond
12. Display records for the lowest priced diamonds
13. Display records for which cut = “Fair”
14. Calculate the median prices for each cut of diamonds
15. Create a boxplot of diamond prices based on cut

86

Getting Started with GGPlot2

Homework

16. What is the price range for the middle 50% of diamonds with color D (best color)?
17. What is the price range for the middle 50% of diamonds with color J (worst color)?
18. What is the IQR for diamonds with the best color (color D)?
19. What is the IQR for diamonds with the worst color (color J)?