BADM 372 Applied Analytics

BADM 372

2023-01-10

Contents

Chapter 1

About this course

This website serves as headquarters for **BADM 372 Applied Analytics**.

Content here will be updated with any changes made during the semester, so if at any point you are told there was a change in the schedule or an assignment, you can come here to get the updated version.

Also, this website has benefited greatly from lots of free, readily available resources posted on the web and we leverage these extensively. I would encourage you to review these resources in your analytics journey. Some that we specifically use with great frequency are these (**and I say a loud THANK YOU to the authors!**):

- R for Data Science
- An Introduction to Statistical Learning with Applications in R
- Data Science in a Box
- stackoverflow.com, for example

Chapter 2

Syllabus

Instructor: Tobin Turner

Office Hours: mutually convenient time arranged by email e-mail: jtturner@presby.edu

2.1 Course Objectives and Learning Outcomes

This course is designed to introduce to data science. Students will apply statistical knowledge and techniques to both business and non-business contexts.

At the end of this course students should be able to:

By the end of the course, you will be able to…

- gain insight from data
- gain insight from data, reproducibly
- gain insight from data, reproducibly, using modern programming tools and techniques
- gain insight from data, reproducibly and collaboratively, using modern programming tools and techniques
- gain insight from data, reproducibly (with literate programming and version control) and collaboratively, using modern programming tools and techniques
- communicate results effectively

This course will be focused on both understanding and applying key business analytical concepts. Although the text serves as a useful foundation for the concepts covered in the class, simple memorization of the material in the text will not be sufficient. Class participation, discussion, and application are critical.

2.2 Text and Resources

- This course website (primary resource)
- R for Data Science
- An Introduction to Statistical Learning with Applications in R
- Data Science in a Box
- stackoverflow.com, for example
- Other free, publicly available datasets and publications.

2.3 Performance Evaluation (Grading)

- Labs, Quizzes and Assignments - 40%
- Exam 1 - 20%
- Exam 2 - 20%
- Final Exam - 20%

2.3.1 Missed Exams/Assignments = ZERO

Arrangements for missed or late assignments must be made **PRIOR** to the exam or due date or the student may receive a ZERO grade for the exam or assignment. See attendance and quiz sections below for more details.

- **Missed Quizzes and Assignments cannot be made up later. Be present.**

2.3.2 Exams

Exams will cover assigned chapters in the textbook, other assigned readings, lectures, class exercises, class discussions, videos, and guest speakers. I will typically allocate time prior to each exam to clearly identify the body of knowledge each test will cover and to answer questions about the format and objectives of the exam.

2.3.3 Labs/Quizzes – DON'T MISS CLASS

- The average of all labs, quizzes and assignments will comprise the Quizzes and Assignments - 40% portion of your final grade
- Quizzes and Assignments are designed to prepare you for your exams and to ensure you stay up with the course material
- **Missed Quizzes and Assignments cannot be made up later. Be present.**

Quizzes rule. **LISTEN. - Missed Quizzes and Assignments cannot be made up later. Be present.**

2.3.4 Final Average

- Final Average Grade
  - 90-100 A
  - 88-89 B+
  - 82-87 B+
  - 80-81 B-
  - 78-79 C+
  - 72-77 C+
  - 70-71 C-
  - 60-69 D
  - 59 and below F

2.4 Class Participation:

I will frequently give readings or assignments for you to complete prior to the next class meeting. I expect you to fully engage the material: answer questions, pose questions, provide insightful observations. Keep in mind that quality is an important component in "participation." Periodic cold calls will take place. I will also put students in the "hot seat" on occasion. In these class sessions, I may select a random group of students to lead us in the discussion and debate. Because the selection of participants will not be announced until class begins, everyone will be expected to prepare for the discussion. Reading the assigned chapters and articles are the best way to prepare for the discussion. If you have concerns about being called on in class, please see me to discuss. The purpose of the "hot seat" is not to stress or embarrass students, but to encourage students to actively engage the material.

2.5 Phones

Phones are not allowed to be used in class without the instructor's prior consent. If you have a need of a phone during class please let me know before class. Unauthorized use of electronic devices may result in the lowering of the grade or dismissal from the class. **I mean this.**

The phone thing? I mean this.

2.6 Attendance

You are expected to be regular and punctual in your class attendance. Students are responsible for all the material missed and homework assignments made. If class is missed, notes/homework should be obtained from another student. If I am more than 15 minutes late, class is considered cancelled. No more than 4 absences are allowed during a semester. Exceeding the absence policy may result in receiving an F for the course. The professors roll is the official roll and students not present when roll is taken will be counted as absent. If a student must miss an exam, she or he must work out an agreeable time with the instructor to take the test prior to the exam being given. If a student misses a test due to an emergency, the student must inform the instructor as soon as is possible. In special cases, the instructor may allow the student to take a make-up exam.

2.7 Accommodations

Presbyterian College is committed to providing reasonable accommodations for all students with documented disabilities. If you are seeking academic accommodations under the Americans with Disabilities Act, you must register with the Academic Success Office, located on 5th Avenue (beside Campus Police). To receive these accommodations, please obtain the proper Accommodations Approval Form from that office, and then meet with me at the beginning of the semester to discuss how we may deliver your approved accommodations. I especially encourage you to meet with me well in advance of the actual accommodations being provided, as it may not be feasible to offer immediate accommodations without sufficient advance notice (such as in the case of tests). I can assure you that all discussions will remain confidential. Disability Services information is located at this link http://bit.ly/PCdisabilityservices

Additionally, it is the student's responsibility to give the instructor one week's notice prior to each instance where accommodation will be required.

2.8 Honor Code and Plagiarism:

All assignments/exams must be your own work. Any copying or use of unauthorized assistance will be treated as a violation of PC's Honor Code. If you are unsure of what resources are allowed, please ask. Please note that all text longer than 7 words taken from ANY other source must be placed in quotations and cited. Also, summarizing ANY other source must also be cited. Using ANY other source and showing work to be your own is a violation of plagiarism and the honor code.

2.9 First-Generation Version:

I am a Presby First+ Advocate. I am here to support our current first-generation students. At Presbyterian College, first-generation students are those in which neither parent nor legal guardian graduated from a four-year higher education institution with a bachelor's degree. If you are a first-generation college student, please contact me. For more information about support for first-generation college students on our campus visit our Presby First+ webpage.

2.10 Continuing Advocate Version

I am a Presby First+ Advocate. I am committed to supporting first-generation students at Presbyterian College. At Presbyterian College, first-generation students are those in which neither parent nor legal guardian graduated from a four-year higher education institution with a bachelor's degree. If you are a first-generation college student, please contact me anytime or visit me during my office hours. For more information about support for first-generation college students on our campus visit our Presby First+ webpage.

Chapter 3

Schedule

This is a tentative schedule, and it will change. **BUT** I will do my very best to review this often so that we all stay on the same page and so that you may plan accordingly!

Spring 2023

Date	Topic	Due
Monday, January 9, 2023	Course Intro, R4DS, A1 review; Lab 1	
Wednesday, January 11, 2023	Rmarkdown; ioslides	
Friday, January 13, 2023	Lab 2: Rmarkdown	**Lab 1 Due**
Monday, January 16, 2023	MLK Holiday	
Wednesday, January 18, 2023	ggplot	
Friday, January 20, 2023	Lab 2: ggplot	
Monday, January 23, 2023	EDA & ggplot;	**Lab 2 Due**
Wednesday, January 25, 2023	EDA & ggplot	
Friday, January 27, 2023	Lab 2: EDA & ggplot	
Monday, January 30, 2023	Dates and Times	**Lab 3 Due**
Wednesday, February 1, 2023	Dates and Times	
Friday, February 3, 2023	Relational Data	
Monday, February 6, 2023	Review	**Lab 4 Due**
Wednesday, February 8, 2023	** Exam 1**	

Date	Topic	Due
Friday, February 10, 2023	Day of Celebtation	
Monday, February 13, 2023	Tidy data from R4DS; Lab 5	
Wednesday, February 15, 2023	SEDSI	
Friday, February 17, 2023	SEDSI	
Monday, February 20, 2023	Functions; Lab 6	**Lab 5 Due**
Wednesday, February 22, 2023	Functions	
Friday, February 24, 2023	Iteration	
Monday, February 27, 2023	Regression	**Lab 6 Due**
Wednesday, March 1, 2023	Regression (Stepwise)	
Friday, March 3, 2023	** Exam 2**	** Exam 2**
Monday, March 6, 2023	Logistic regression	
Wednesday, March 8, 2023	Logistic regression	
Friday, March 10, 2023	Quiz	**QUIZ**
Monday, March 13, 2023	SPRING BREAK	
Wednesday, March 15, 2023	SPRING BREAK	
Friday, March 17, 2023	SPRING BREAK	**QUIZ**
Monday, March 20, 2023	Tree-based methods	
Wednesday, March 22, 2023	Tree-based methods	
Friday, March 24, 2023		**QUIZ**
Monday, March 27, 2023	Clusters	
Wednesday, March 29, 2023	Clusters	
Friday, March 31, 2023	LAUNCH PROJECT	**QUIZ**
Monday, April 3, 2023	INDEPENDENT PROJECT	
Wednesday, April 5, 2023	INDEPENDENT PROJECT	
Friday, April 7, 2023	Easter Holidays	
Monday, April 10, 2023	Easter Holidays	
Wednesday, April 12, 2023	INDEPENDENT PROJECT	
Friday, April 14, 2023	INDEPENDENT PROJECT	
Monday, April 17, 2023	INDEPENDENT PROJECT	
Wednesday, April 19, 2023	INDEPENDENT PROJECT	

Date	Topic	Due
Friday, April 21, 2023	INDEPENDENT PROJECT	
Monday, April 24, 2023	PRESENTATIONS	
Wednesday, April 26, 2023	PRESENTATIONS	
Friday, April 28, 2023	**LAST DAY OF CLASSES**	

Chapter 4

Lab 1 Excercises

Let's make sure we feel good about BADM 371 material.

All open notes/internet/R4DS/etc., **but all work must be your own**.

Use the starwars data (dplyr package) to answer/do:

1. Who is the tallest individual? Shortest?
2. How many homeworlds are there?
3. Which homeworld has the most individuals? Fewest? Average # of idividuals per homeworld?
4. Make a plot of all individuals with mass on the x axis and height on the y axis.
5. Put a best fit line on this plot.
6. Who is the biggest outlier in this dataset?
7. Calculate BMI for all these individuals. What is the average BMI for all individuals?
8. What is the average BMI for each homeworld?
9. Which homeworlds have the greatest percentage of individuals with BMI's greater than the average you found in #8 above?
10. How many individuals have no missing data? Which variables have the most missing data?

Chapter 5

Lab 2 in Rmarkdown

5.1 R Markdown

```
library(dplyr)
```

1. Who is the tallest individual? Shortest?

```
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>    66.0   167.0   180.0   174.4   191.0   264.0       6
```

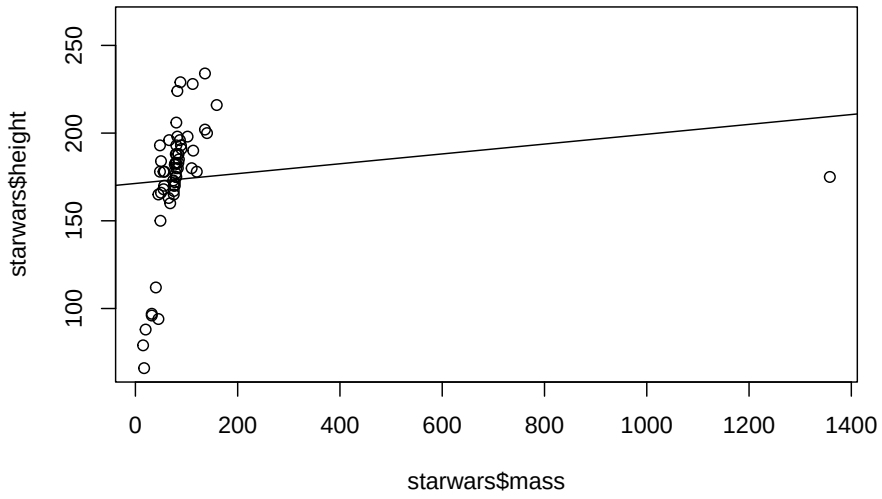
2. How many homeworlds are there?

```
#> # A tibble: 49 x 1
#>    homeworld
#>    <chr>
#>  1 Tatooine
#>  2 Naboo
#>  3 Alderaan
#>  4 Stewjon
#>  5 Eriadu
#>  6 Kashyyyk
#>  7 Corellia
#>  8 Rodia
#>  9 Nal Hutta
#> 10 Bestine IV
#> # ... with 39 more rows
```

3. Which homeworld has the most individuals? Fewest? Average # of individuals per homeworld?

```
#> # A tibble: 49 x 2
#>    homeworld     n
#>    <chr>     <int>
#>  1 Naboo        11
#>  2 Tatooine     10
#>  3 <NA>         10
#>  4 Alderaan      3
#>  5 Coruscant     3
#>  6 Kamino        3
#>  7 Corellia      2
#>  8 Kashyyyk      2
#>  9 Mirial        2
#> 10 Ryloth        2
#> # ... with 39 more rows
#> # A tibble: 49 x 2
#>    homeworld          n
#>    <chr>          <int>
#>  1 Aleen Minor        1
#>  2 Bespin             1
#>  3 Bestine IV         1
#>  4 Cato Neimoidia     1
#>  5 Cerea              1
#>  6 Champala           1
#>  7 Chandrila          1
#>  8 Concord Dawn       1
#>  9 Dathomir           1
#> 10 Dorin              1
#> # ... with 39 more rows
```

4. Make a plot of all individuals with mass on the x axis and height on the y axis. Put a best fit line on this plot. Who is the biggest outlier in this dataset?




```
#> # A tibble: 1 x 3
#>   name                   mass height
#>   <chr>                 <dbl>  <int>
#> 1 Jabba Desilijic Tiure  1358    175
```

5. Make something like this:

5.2 Why I like this class

5.2.1 Its awesome

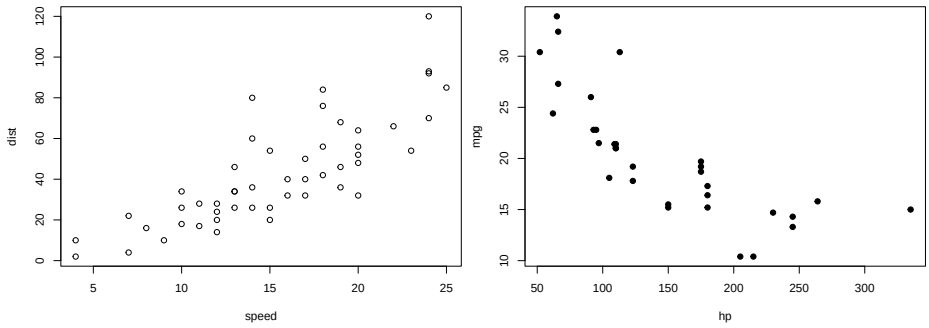
5.2.1.1 Really Awesome

5.2.1.1.1 Super Duper Awesome

5.2.1.1.1.1 Reason 5

6. . Make plots like these:

```
par(mar = c(4, 4, .1, .1))
plot(cars)
plot(mpg ~ hp, data = mtcars, pch = 19)
```



7. Calculate BMI for all these individuals. What is the average BMI for all individuals?

   Via google: With the metric system, the formula for BMI is weight in kilograms divided by height in meters squared. Since height is commonly measured in centimeters, an alternate calculation formula, dividing the weight in kilograms by the height in centimeters squared, and then multiplying the result by 10,000, can be used

```
#> # A tibble: 59 x 4
#>    name                 BMI height  mass
#>    <chr>              <dbl>  <int> <dbl>
#>  1 Luke Skywalker      26.0    172    77
#>  2 C-3PO               26.9    167    75
#>  3 R2-D2               34.7     96    32
#>  4 Darth Vader         33.3    202   136
#>  5 Leia Organa         21.8    150    49
#>  6 Owen Lars           37.9    178   120
#>  7 Beru Whitesun lars  27.5    165    75
#>  8 R5-D4               34.0     97    32
#>  9 Biggs Darklighter   25.1    183    84
#> 10 Obi-Wan Kenobi      23.2    182    77
#> # ... with 49 more rows
#> # A tibble: 1 x 1
#>   `mean(BMI)`
#>         <dbl>
#> 1        32.0
```

8. What is the average BMI for each homeworld?

```
#> # A tibble: 40 x 2
#>    homeworld  avg.BMI
#>    <chr>        <dbl>
```

```
#>  1 Nal Hutta    443.
#>  2 Vulpter       50.9
#>  3 Kalee         34.1
#>  4 Bestine IV    34.0
#>  5 <NA>          32.6
#>  6 Malastare     31.9
#>  7 Trandosha     31.3
#>  8 Tatooine      29.3
#>  9 Sullust       26.6
#> 10 Dathomir      26.1
#> # ... with 30 more rows
```

9. Which homeworlds have the greatest percentage of individuals with BMI's greater than the average you found in #8 above? How many individuals have no missing data? Which variables have the most missing data?

```
#> # A tibble: 5 x 2
#>   homeworld  avg.BMI
#>   <chr>        <dbl>
#> 1 Nal Hutta    443.
#> 2 Vulpter       50.9
#> 3 Kalee         34.1
#> 4 Bestine IV    34.0
#> 5 <NA>          32.6
```

10. How many individuals have no missing data?

    Via google: https://stackoverflow.com/questions/22353633/filter-for-complete-cases-in-data-frame-using-dplyr-case-wise-deletion

Single Variable Considered:

```
#> # A tibble: 29 x 14
#>    name         height  mass hair_color skin_color eye_color
#>    <chr>         <int> <dbl> <chr>      <chr>      <chr>
#>  1 Luke Skywal~    172    77 blond      fair       blue
#>  2 Darth Vader     202   136 none       white      yellow
#>  3 Leia Organa     150    49 brown      light      brown
#>  4 Owen Lars       178   120 brown, gr~ light      blue
#>  5 Beru Whites~    165    75 brown      light      blue
#>  6 Biggs Darkl~    183    84 black      light      brown
#>  7 Obi-Wan Ken~    182    77 auburn, w~ fair       blue-gray
#>  8 Anakin Skyw~    188    84 blond      fair       blue
#>  9 Chewbacca       228   112 brown      unknown    blue
```

```
#> 10 Han Solo        180    80 brown      fair       brown
#> # ... with 19 more rows, and 8 more variables:
#> #   birth_year <dbl>, sex <chr>, gender <chr>,
#> #   homeworld <chr>, species <chr>, films <list>,
#> #   vehicles <list>, starships <list>
```

Which variables have the most missing data?

> Via google: https://stackoverflow.com/questions/26273663/r-how-to-total-the-number-of-na-in-each-col-of-data-frame

Chapter 6

All Variables Considered:

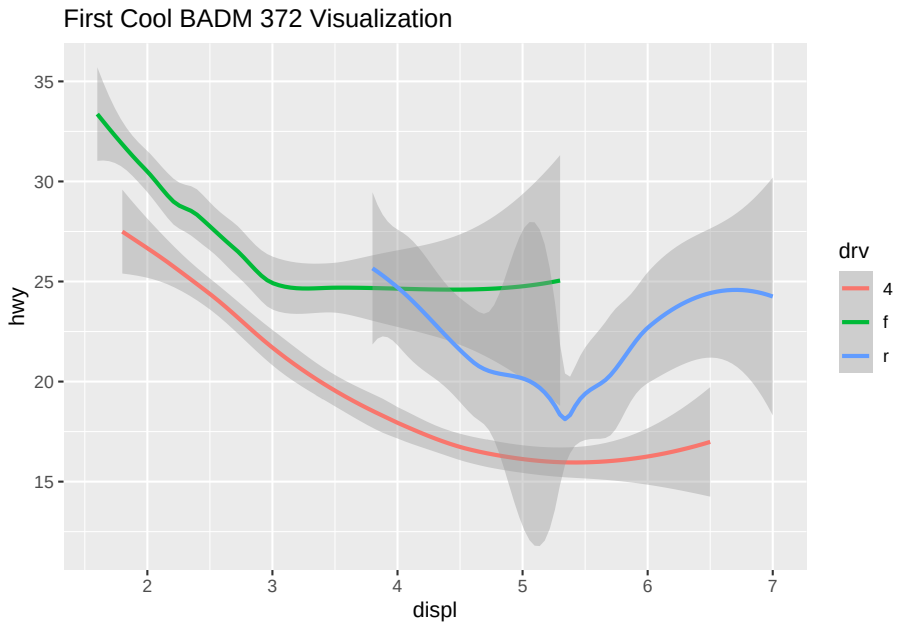
```
#> # A tibble: 1 x 14
#>    name height  mass hair_color skin_color eye_color
#>   <int>  <int> <int>      <int>      <int>     <int>
#> 1     0      6    28          5          0         0
#> # ... with 8 more variables: birth_year <int>, sex <int>,
#> #   gender <int>, homeworld <int>, species <int>,
#> #   films <int>, vehicles <int>, starships <int>
```

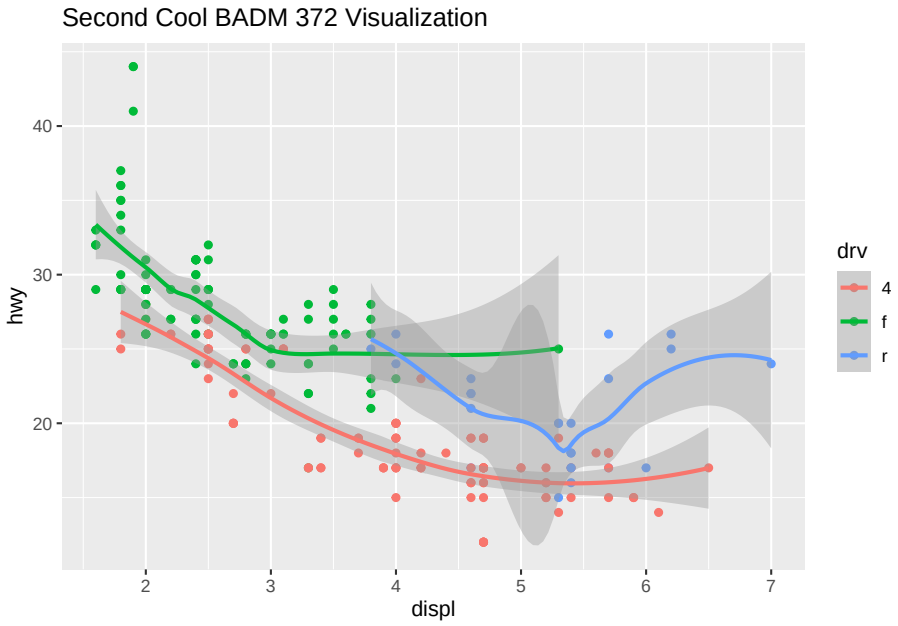
Chapter 7

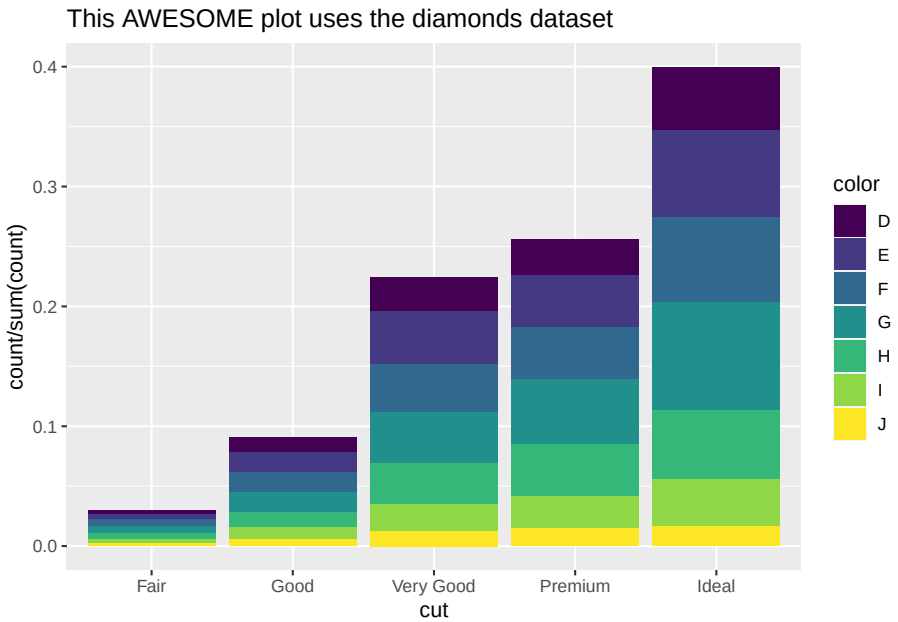
Lab 2 – ggplot without `dsbox`

7.1 Excercises using the data sets `mpg` or `diamonds`

1. Create these figures using the data sets `mpg` or `diamonds` as needed:




Second Cool BADM 372 Visualization
hwy
displ
drv
4
f
r
40
30
20
2
3
4
5
6
7

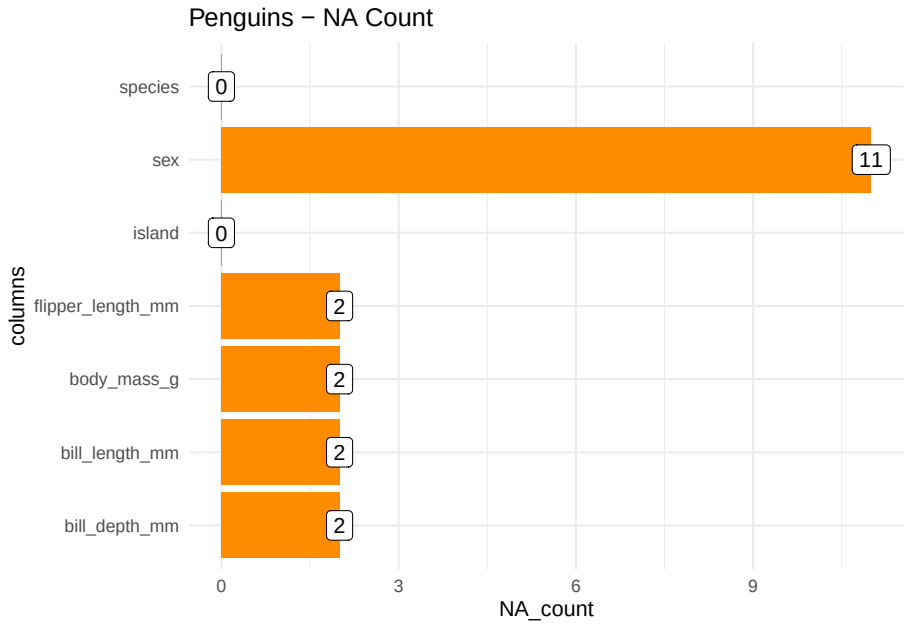

This AWESOME plot uses the diamonds dataset
count/sum(count)
cut
color
D
E
F
G
H
I
J
0.4
0.3
0.2
0.1
0.0
Fair
Good
Very Good
Premium
Ideal

7.2 palmerpenguins

`palmerpenguins` is a realtively new package on CRAN, so you can install it from CRAN instead of Github.

Install it like a normal package. After successful installation, you can find out that there are two datasets attached with the package – penguins and penguins_raw. You can check out their help page (?penguins_raw and ?penguins_raw) to understand more about respective datasets.

1. Please make a well-labeled, meangingful plot that show how many missing variables there are for each variable in the dataset. Your results shoud look something like this:

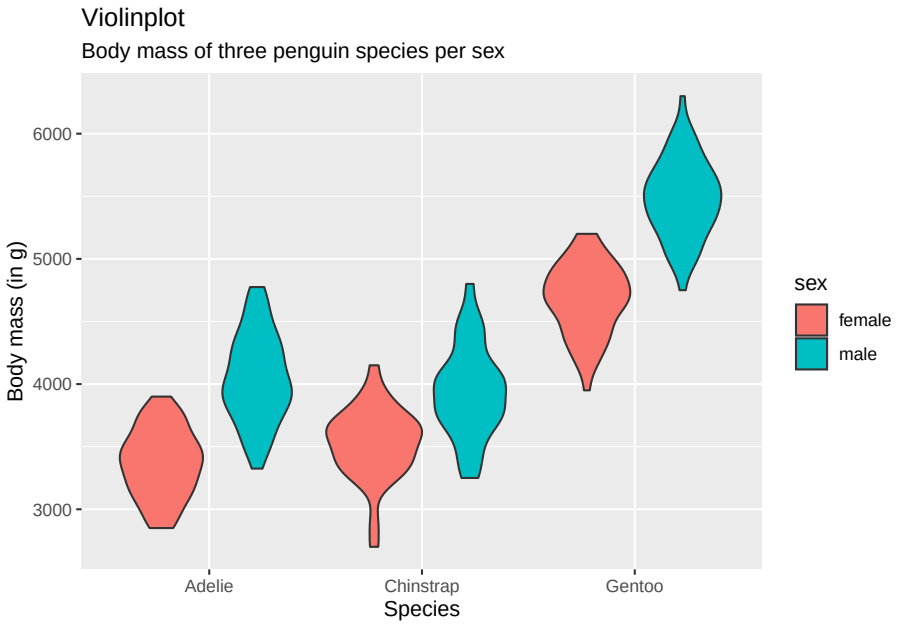


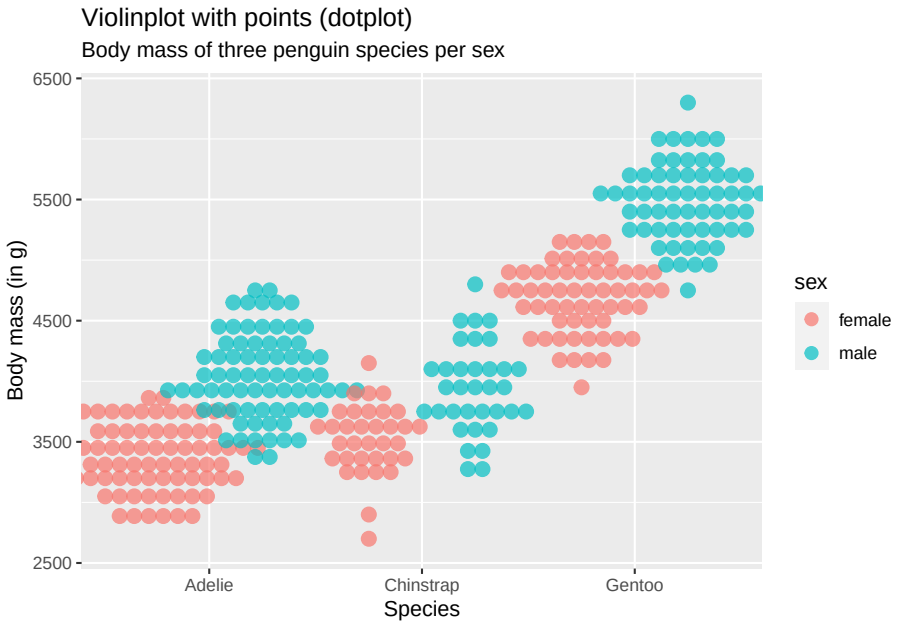
2. Make a plot showing a count of penguins of each species.

3. Create a plot that illustrates the relationship between flipper_length_mm and body_mass_g with respect to each species.

4. Create a plot that illustrates the relationship between flipper_length_mm and body_mass_g with respect to each species for each island.

5. Create a few plots of your own using new/interesting geoms and make sure the plots have meangiful, imprmfative labels, too. For possible examples:


Violinplot
Body mass of three penguin species per sex
Body mass (in g)
6000
5000
4000
3000
Adelie
Chinstrap
Gentoo
Species
sex
female
male


Violinplot with points (dotplot)
Body mass of three penguin species per sex
Body mass (in g)
6500
5500
4500
3500
2500
Adelie
Chinstrap
Gentoo
Species
sex
female
male

Chapter 8

Functions

8.1 Writing Functions

8.1.1 Fahrenheit to Kelvin

$k = ((f - 32) * (5/9)) + 273.15$

```
((32 - 32) * (5 / 9)) + 273.15
#> [1] 273.15
((212 - 32) * (5 / 9)) + 273.15
#> [1] 373.15
((-42 - 32) * (5 / 9)) + 273.15
#> [1] 232.0389
```

```
f_k <- function(f_temp) {
    ((f_temp - 32) * (5 / 9)) + 273.15
}
```

```
f_k(32)
#> [1] 273.15
f_k(212)
#> [1] 373.15
f_k(-42)
#> [1] 232.0389
```

8.1.2 Kelvin to Celsius

```
k_c <- function(temp_k) {
    temp_c <- temp_k - 273.15
    return(temp_c)
}
```

```
k_c(0)
#> [1] -273.15
```

8.1.3 Fahrenheit to Celsius

```
f_c <- function(temp_f) {
    temp_k <- f_k(temp_f)
    temp_c <- k_c(temp_k)
    return(temp_c)
}
```

```
f_c(32)
#> [1] 0
f_c(212)
#> [1] 100
```

8.2 Testing Functions

```
library(testthat)
testthat::expect_equal(f_c(32), 0)
testthat::expect_equal(f_c(212), 100)
```

8.3 Exercise

1. What happens if you use `NA`, `Inf`, `-Inf` in your function?
2. What are some better names to give the functions we wrote?

- How would you name these functions in a package?

8.4 Checking values

Calculating weighted means

```
mean_wt <- function(x, w) {
  sum(x * w) / sum(w)
}
```

```
mean_wt(1:6, 1:6)
#> [1] 4.333333
```

If you expect the lengths to be the same, then you should test for it in the function

```
mean_wt(1:6, 1:3)
#> [1] 7.666667
```

```
mean_wt <- function(x, w) {
  if (length(x) != length(w)) {
    stop("`x` and `w` should be the same length")
  }
  sum(x * w) / sum(w)
}
```

```
mean_wt(1:6, 1:3)
#> Error in mean_wt(1:6, 1:3): `x` and `w` should be the same length
```

8.5 dot-dot-dot ...

Use it to pass on arguments to another function inside.

But you can also use it to force named arguments in your function.

```
sum_3 <- function(x, y, z) {
  return(x + y + z)
}
```

```
sum_3(1, 2, 3)
#> [1] 6
```

```
sum_3 <- function(x, y, ..., z) {
  return(x + y + z)
}
```

```
sum_3(1, 2, z = 3)
#> [1] 6
```

```
sum_3(1, 2, z = 3)
#> [1] 6
```

8.6 Conditionals

8.7 if statements

```
# make a modification to this function
k_c <- function(temp_k) {
    if (temp_k < 0) {
        warning('you passed in a negative Kelvin number')
        # stop()
        return(NA)
    }
    temp_c <- temp_k - 273.15
    return(temp_c)
}
```

```
k_c(-9)
#> Warning in k_c(-9): you passed in a negative Kelvin number
#> [1] NA
```

Our current function does not deal with missing numbers

```
k_c(NA)
```

```
Error in if (temp_k < 0) { : missing value where TRUE/FALSE needed
```

```
k_c(0)
#> [1] -273.15
```

8.8 If else statements

```
k_c <- function(temp_k) {
    if (temp_k < 0) {
        warning('you passed in a negative Kelvin number')
        # stop()
        return(NA)
    } else {
        temp_c <- temp_k - 273.15
        return(temp_c)
    }
}
```

```
k_c(-9)
#> Warning in k_c(-9): you passed in a negative Kelvin number
#> [1] NA
```

Our current function does not deal with missing numbers

```
k_c(NA)
```

```
k_c(0)
#> [1] -273.15
```

8.9 Dealing with NA

Re-write our function to work with missing values.

Note you need to make the `NA` check first.

```
k_c <- function(temp_k) {
    if (is.na(temp_k)) {
        return(NA)
    } else if (temp_k < 0) {
        warning('you passed in a negative Kelvin number')
        # stop()
        return(NA)
    } else {
        temp_c <- temp_k - 273.15
        return(temp_c)
    }
}
```

```
k_c(-9)
#> Warning in k_c(-9): you passed in a negative Kelvin number
#> [1] NA
```

```
k_c(NA)
#> [1] NA
```

```
k_c(0)
#> [1] -273.15
```

use `&&` and `||` to short-circuit the boolean comparisons. This will also guarantee a value of length `1L`. `==` is also vectorized, should use `identical()` or `all.equal()`.

`identical` is very strict. Doesn't corece types.

```
identical(0L, 0)
#> [1] FALSE
```

`all.equal` has ability to set tolerances.

`all.equal`: compare R objects x and y testing 'near equality'. If they are different, comparison is still made to some extent, and a report of the differences is returned. Do not use all.equal directly in if expressions—either use isTRUE(all.equal(....)) or identical if appropriate.

```
all.equal(0L, 0)
#> [1] TRUE
```

```
if (isTRUE(all.equal(0L, 0))) {print("Hello")}
#> [1] "Hello"
```

8.10 Fizzbuzz

```
fizzbuzz <- function(x) {
  # these two lines check that x is a valid input
  stopifnot(length(x) == 1)
  stopifnot(is.numeric(x))
  if (!(x %% 3) && !(x %% 5)) {
    "fizzbuzz"
  } else if (!(x %% 3)) {
    "fizz"
```

```
  } else if (!(x %% 5)) {
    "buzz"
  } else {
    # ensure that the function returns a character vector
    as.character(x)
  }
}
```

```
fizzbuzz(6)
#> [1] "fizz"
```

Check modulo 3 only once

```
fizzbuzz2 <- function(x) {
  # these two lines check that x is a valid input
  stopifnot(length(x) == 1)
  stopifnot(is.numeric(x))
  if (!(x %% 3)) {
    if (!(x %% 5)) {
      "fizzbuzz"
    } else {
      "fizz"
    }
  } else if (!(x %% 5)) {
    "buzz"
  } else {
    # ensure that the function returns a character vector
    as.character(x)
  }
}
```

```
fizzbuzz(6)
#> [1] "fizz"
```

8.10.1 Vectorized conditionals

```
library(dplyr)
#>
#> Attaching package: 'dplyr'
#> The following object is masked from 'package:testthat':
#>
#>     matches
```

```
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union
fizzbuzz_vec <- function(x) {
  dplyr::case_when(
    !(x %% 3) & !(x %% 5) ~ "fizzbuzz",
    !(x %% 3) ~ "fizz",
    !(x %% 5) ~ "buzz",
    TRUE ~ as.character(x)
  )
}
```

```
fizzbuzz(1:10)
#> Error in fizzbuzz(1:10): length(x) == 1 is not TRUE
```

```
fizzbuzz_vec(1:10)
#>  [1] "1"    "2"    "fizz" "4"    "buzz" "fizz" "7"    "8"
#>  [9] "fizz" "buzz"
```

8.10.2 Multiple conditions

```
if (this) {
  # do that
} else if (that) {
  # do something else
} else {
  #
}
```

Chapter 9

Linear Regression

Your resource for this is ISLR chapter 3: linear regression.

9.1 Exercises

1. Make sure you can define the terms below **outloud**, in your own words, so that they make sense both to you and to someone else (me?). Actually practice saying and defining these terms **outloud** until your answers make sense:

- least squares approach
- confidence interval
- p-value
- R^2
- Adjusted R^2
- qualitative predictor
- collinearity
- KNN
- Residual standard error
- F-statistic
- Explain the point of Figure 3.1

2. In `m1`, below, which variables are significant predictors of Balance? How do you know?

```
library("ISLR")
data(Credit)
attach(Credit)
```

```
head(Credit)
#>   ID  Income Limit Rating Cards Age Education Gender
#> 1  1  14.891  3606    283     2  34        11   Male
#> 2  2 106.025  6645    483     3  82        15 Female
#> 3  3 104.593  7075    514     4  71        11   Male
#> 4  4 148.924  9504    681     3  36        11 Female
#> 5  5  55.882  4897    357     2  68        16   Male
#> 6  6  80.180  8047    569     4  77        10   Male
#>   Student Married Ethnicity Balance
#> 1      No     Yes Caucasian     333
#> 2     Yes     Yes     Asian     903
#> 3      No      No     Asian     580
#> 4      No      No     Asian     964
#> 5      No     Yes Caucasian     331
#> 6      No      No Caucasian    1151
m1 <- lm(Balance ~ Age + Income + Education, data = Credit)
summary(m1)
#>
#> Call:
#> lm(formula = Balance ~ Age + Income + Education, data = Credit)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -867.14 -343.14  -49.44  316.55 1080.56
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 348.8115   112.6895   3.095  0.00211 **
#> Age          -2.1863     1.2004  -1.821  0.06930 .
#> Income        6.2380     0.5877  10.614  < 2e-16 ***
#> Education     0.8058     6.5254   0.123  0.90179
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 407.2 on 396 degrees of freedom
#> Multiple R-squared:  0.2215, Adjusted R-squared:  0.2156
#> F-statistic: 37.56 on 3 and 396 DF,  p-value: < 2.2e-16
```

3. How "good" is the model created in `m1`? How do you know?
4. Add more `Credit` variables to model `m1`. Can you find two other variables that have extremely high collinearity? What are they? How do you know that they have high collinearity? Why does this make sense given what each of the variables mean?
5. Based on the model below, what would you predict the balance to be for

an individual who is 40, has an income of $100,000, 16 years of education, is Asian and not a student?

```
m2 <- lm(Balance ~ Age + Income + Education + Ethnicity + Student, data = Credit)
summary(m2)
#>
#> Call:
#> lm(formula = Balance ~ Age + Income + Education + Ethnicity +
#>     Student, data = Credit)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -818.77 -322.14  -54.52  315.67  781.45
#>
#> Coefficients:
#>                    Estimate Std. Error t value Pr(>|t|)
#> (Intercept)        336.6241   115.6311   2.911  0.00381 **
#> Age                 -1.9756     1.1595  -1.704  0.08922 .
#> Income               6.1491     0.5666  10.853  < 2e-16 ***
#> Education           -1.7606     6.3060  -0.279  0.78024
#> EthnicityAsian     -14.2547    55.5240  -0.257  0.79752
#> EthnicityCaucasian   8.8839    48.3276   0.184  0.85424
#> StudentYes         382.0498    65.6854   5.816 1.25e-08 ***
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 392.2 on 393 degrees of freedom
#> Multiple R-squared:  0.2833, Adjusted R-squared:  0.2723
#> F-statistic: 25.89 on 6 and 393 DF,  p-value: < 2.2e-16
```

6. Interpret this model and its output, especially the coefficients `Income:Education  0.3149`:

```
m3 <- lm(Balance ~ Income*Education, data = Credit)
summary(m3)
#>
#> Call:
#> lm(formula = Balance ~ Income * Education, data = Credit)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -858.07 -349.99  -56.12  304.51 1083.93
#>
#> Coefficients:
```

```
#>                  Estimate Std. Error t value Pr(>|t|)
#> (Intercept)      435.4599   147.1000   2.960  0.00326 **
#> Income             1.8168     2.4727   0.735  0.46294
#> Education        -13.9887    10.5931  -1.321  0.18741
#> Income:Education   0.3149     0.1788   1.761  0.07902 .
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 407.3 on 396 degrees of freedom
#> Multiple R-squared:  0.2211, Adjusted R-squared:  0.2152
#> F-statistic: 37.47 on 3 and 396 DF,  p-value: < 2.2e-16
```

7. What's going on here?

```
options(scipen=999)
options(digits=3)
library(ISLR)
data("Credit")
attach(Credit)
#> The following objects are masked from Credit (pos = 3):
#>
#>     Age, Balance, Cards, Education, Ethnicity,
#>     Gender, ID, Income, Limit, Married, Rating,
#>     Student
?Credit
#> starting httpd help server ...
#>  done
m1 <- lm(Balance~Income+Education)
summary(m1)
#>
#> Call:
#> lm(formula = Balance ~ Income + Education)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -806.2 -349.7  -53.4  330.4 1103.4
#>
#> Coefficients:
#>             Estimate Std. Error t value            Pr(>|t|)
#> (Intercept)  236.975     94.767    2.50               0.013
#> Income         6.050      0.580   10.43 <0.0000000000000002
#> Education      0.703      6.544    0.11               0.914
#>
#> (Intercept) *
```

```
#> Income      ***
#> Education
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 408 on 397 degrees of freedom
#> Multiple R-squared:  0.215,  Adjusted R-squared:  0.211
#> F-statistic: 54.4 on 2 and 397 DF,  p-value: <0.0000000000000002
236.975+(6.050 *100)+(0.703*12)
#> [1] 850
predict(m1, newdata = list(Income = 100, Education = 12))
#>   1
#> 850

m2 <- lm(Balance~Income*Education)
summary(m2)
#>
#> Call:
#> lm(formula = Balance ~ Income * Education)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -858.1 -350.0  -56.1  304.5 1083.9
#>
#> Coefficients:
#>                  Estimate Std. Error t value Pr(>|t|)
#> (Intercept)       435.460    147.100    2.96   0.0033 **
#> Income              1.817      2.473    0.73   0.4629
#> Education         -13.989     10.593   -1.32   0.1874
#> Income:Education    0.315      0.179    1.76   0.0790 .
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 407 on 396 degrees of freedom
#> Multiple R-squared:  0.221,  Adjusted R-squared:  0.215
#> F-statistic: 37.5 on 3 and 396 DF,  p-value: <0.0000000000000002

435.460+(1.817*100)+(-13.989*12)+(0.315*100*12)
#> [1] 827
predict(m2, newdata = list(Income = 100, Education = 12))
#>   1
#> 827
```

Chapter 10

Logistic Regression

Your resource for this is ISLR Chapter 4: Classification.

Chapter 11

Tree-Based Methods

Your resource for this is ISLR Chapter 8: Tree-Based Methods.

Make sure you can exlain the terms/ideas/figures below **outloud**, in your own words, so that they make sense both to you and to someone else (me?). Actually practice exlaining the terms/ideas/figures **outloud** until your answers make sense:

- Regression vs. classification trees
- Understand Figure 8.1, Figure 8.2, Algorithm 8.1, Figure 8.3, Figure 8.4, Figure 8.5, Figure 8.6, Figure 8.7
- Be able to explain the +'s and -'s of trees (see section 8.1.4)
- Understand "combining a large number of trees can often result in dramatic improvements in prediction accuracy, at the expense of some loss in interpretation."
- top-down, greedy approach (aka recursive binary splitting)
- tree pruning and subtrees
- cost complexity pruning (aka weakest link pruning)
- bagging
- random forests
- boosting
- Bayesian additive regression trees

Chapter 12

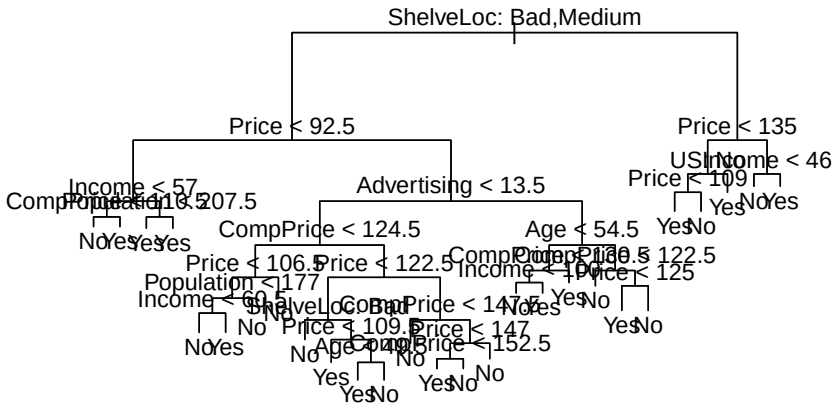
Chapter 8 Lab: Decision Trees

**Check out this Video: StatsLearning Lect10 R trees A 111213

Or Click Here: Here!!!

```
## Fitting Classification Trees

###
library(tree)
###
library(ISLR2)
attach(Carseats)
High <- factor(ifelse(Sales <= 8, "No", "Yes"))
###
Carseats <- data.frame(Carseats, High)
###
tree.carseats <- tree(High ~ . - Sales, Carseats)
###
summary(tree.carseats)
#>
#> Classification tree:
#> tree(formula = High ~ . - Sales, data = Carseats)
#> Variables actually used in tree construction:
#> [1] "ShelveLoc"   "Price"       "Income"      "CompPrice"
#> [5] "Population"  "Advertising" "Age"         "US"
#> Number of terminal nodes:  27
#> Residual mean deviance:  0.4575 = 170.7 / 373
#> Misclassification error rate: 0.09 = 36 / 400
```

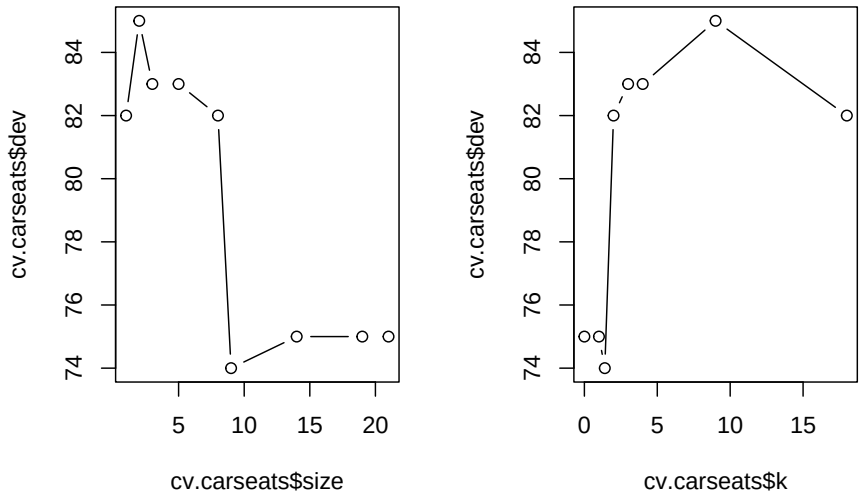
```
###
plot(tree.carseats)
text(tree.carseats, pretty = 0)
```



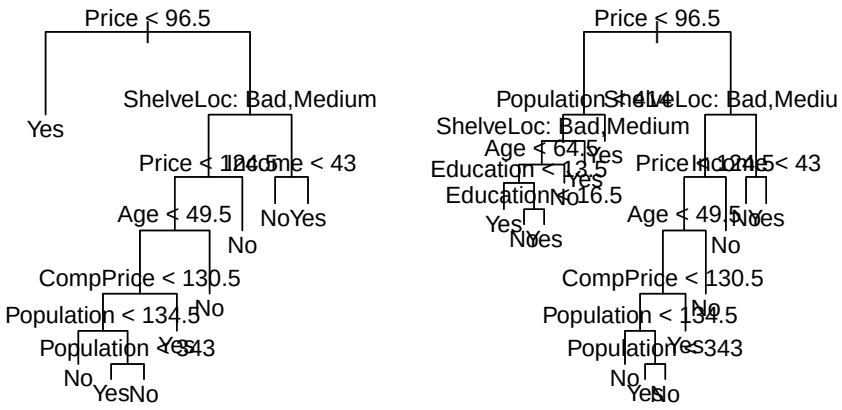

```
###
tree.carseats
#> node), split, n, deviance, yval, (yprob)
#>       * denotes terminal node
#>
#>   1) root 400 541.500 No ( 0.59000 0.41000 )
#>     2) ShelveLoc: Bad,Medium 315 390.600 No ( 0.68889 0.31111 )
#>       4) Price < 92.5 46  56.530 Yes ( 0.30435 0.69565 )
#>         8) Income < 57 10  12.220 No ( 0.70000 0.30000 )
#>          16) CompPrice < 110.5 5   0.000 No ( 1.00000 0.00000 ) *
#>          17) CompPrice > 110.5 5   6.730 Yes ( 0.40000 0.60000 ) *
#>         9) Income > 57 36  35.470 Yes ( 0.19444 0.80556 )
#>          18) Population < 207.5 16  21.170 Yes ( 0.37500 0.62500 ) *
#>          19) Population > 207.5 20   7.941 Yes ( 0.05000 0.95000 ) *
#>       5) Price > 92.5 269 299.800 No ( 0.75465 0.24535 )
#>        10) Advertising < 13.5 224 213.200 No ( 0.81696 0.18304 )
#>          20) CompPrice < 124.5 96  44.890 No ( 0.93750 0.06250 )
#>            40) Price < 106.5 38  33.150 No ( 0.84211 0.15789 )
#>              80) Population < 177 12  16.300 No ( 0.58333 0.41667 )
#>               160) Income < 60.5 6   0.000 No ( 1.00000 0.00000 ) *
```

```
#>                 161) Income > 60.5 6   5.407 Yes ( 0.16667 0.83333 ) *
#>               81) Population > 177 26   8.477 No ( 0.96154 0.03846 ) *
#>             41) Price > 106.5 58   0.000 No ( 1.00000 0.00000 ) *
#>           21) CompPrice > 124.5 128 150.200 No ( 0.72656 0.27344 )
#>             42) Price < 122.5 51  70.680 Yes ( 0.49020 0.50980 )
#>               84) ShelveLoc: Bad 11   6.702 No ( 0.90909 0.09091 ) *
#>               85) ShelveLoc: Medium 40  52.930 Yes ( 0.37500 0.62500 )
#>                 170) Price < 109.5 16   7.481 Yes ( 0.06250 0.93750 ) *
#>                 171) Price > 109.5 24  32.600 No ( 0.58333 0.41667 )
#>                   342) Age < 49.5 13  16.050 Yes ( 0.30769 0.69231 ) *
#>                   343) Age > 49.5 11   6.702 No ( 0.90909 0.09091 ) *
#>             43) Price > 122.5 77  55.540 No ( 0.88312 0.11688 )
#>               86) CompPrice < 147.5 58  17.400 No ( 0.96552 0.03448 ) *
#>               87) CompPrice > 147.5 19  25.010 No ( 0.63158 0.36842 )
#>                 174) Price < 147 12  16.300 Yes ( 0.41667 0.58333 )
#>                   348) CompPrice < 152.5 7   5.742 Yes ( 0.14286 0.85714 ) *
#>                   349) CompPrice > 152.5 5   5.004 No ( 0.80000 0.20000 ) *
#>                 175) Price > 147 7   0.000 No ( 1.00000 0.00000 ) *
#>         11) Advertising > 13.5 45  61.830 Yes ( 0.44444 0.55556 )
#>           22) Age < 54.5 25  25.020 Yes ( 0.20000 0.80000 )
#>             44) CompPrice < 130.5 14  18.250 Yes ( 0.35714 0.64286 )
#>               88) Income < 100 9  12.370 No ( 0.55556 0.44444 ) *
#>               89) Income > 100 5   0.000 Yes ( 0.00000 1.00000 ) *
#>             45) CompPrice > 130.5 11   0.000 Yes ( 0.00000 1.00000 ) *
#>           23) Age > 54.5 20  22.490 No ( 0.75000 0.25000 )
#>             46) CompPrice < 122.5 10   0.000 No ( 1.00000 0.00000 ) *
#>             47) CompPrice > 122.5 10  13.860 No ( 0.50000 0.50000 )
#>               94) Price < 125 5   0.000 Yes ( 0.00000 1.00000 ) *
#>               95) Price > 125 5   0.000 No ( 1.00000 0.00000 ) *
#>      3) ShelveLoc: Good 85  90.330 Yes ( 0.22353 0.77647 )
#>        6) Price < 135 68  49.260 Yes ( 0.11765 0.88235 )
#>         12) US: No 17  22.070 Yes ( 0.35294 0.64706 )
#>           24) Price < 109 8   0.000 Yes ( 0.00000 1.00000 ) *
#>           25) Price > 109 9  11.460 No ( 0.66667 0.33333 ) *
#>         13) US: Yes 51  16.880 Yes ( 0.03922 0.96078 ) *
#>        7) Price > 135 17  22.070 No ( 0.64706 0.35294 )
#>         14) Income < 46 6   0.000 No ( 1.00000 0.00000 ) *
#>         15) Income > 46 11  15.160 Yes ( 0.45455 0.54545 ) *
###
set.seed(2)
train <- sample(1:nrow(Carseats), 200)
Carseats.test <- Carseats[-train, ]
High.test <- High[-train]
tree.carseats <- tree(High ~ . - Sales, Carseats,
                      subset = train)
```

```
tree.pred <- predict(tree.carseats, Carseats.test,
                     type = "class")
table(tree.pred, High.test)
#>           High.test
#> tree.pred  No Yes
#>       No  104  33
#>       Yes  13  50
(104 + 50) / 200
#> [1] 0.77
###
set.seed(7)
cv.carseats <- cv.tree(tree.carseats, FUN = prune.misclass)
names(cv.carseats)
#> [1] "size"   "dev"    "k"      "method"
cv.carseats
#> $size
#> [1] 21 19 14  9  8  5  3  2  1
#>
#> $dev
#> [1] 75 75 75 74 82 83 83 85 82
#>
#> $k
#> [1] -Inf  0.0  1.0  1.4  2.0  3.0  4.0  9.0 18.0
#>
#> $method
#> [1] "misclass"
#>
#> attr(,"class")
#> [1] "prune"         "tree.sequence"
###
par(mfrow = c(1, 2))
plot(cv.carseats$size, cv.carseats$dev, type = "b")
plot(cv.carseats$k, cv.carseats$dev, type = "b")
```

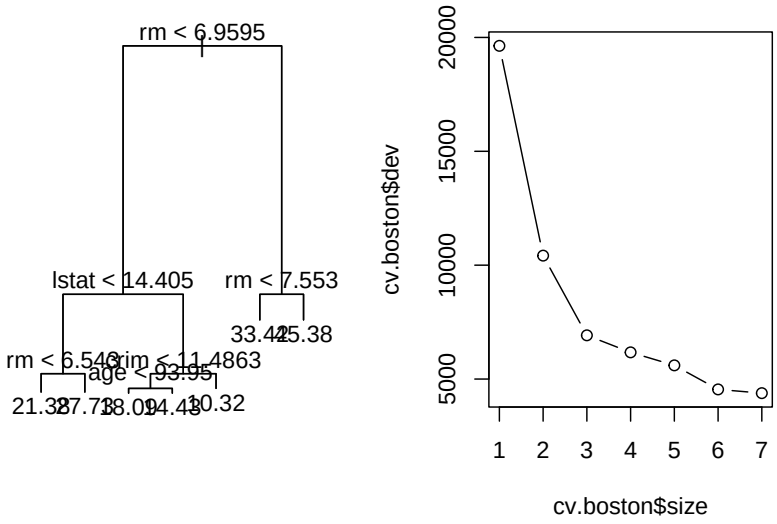



```
###
prune.carseats <- prune.misclass(tree.carseats, best = 9)
plot(prune.carseats)
text(prune.carseats, pretty = 0)
###
tree.pred <- predict(prune.carseats, Carseats.test,
                     type = "class")
table(tree.pred, High.test)
#>           High.test
#> tree.pred No Yes
#>       No  97  25
#>       Yes 20  58
(97 + 58) / 200
#> [1] 0.775
###
prune.carseats <- prune.misclass(tree.carseats, best = 14)
plot(prune.carseats)
text(prune.carseats, pretty = 0)
```

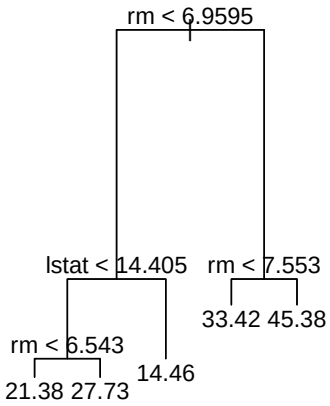


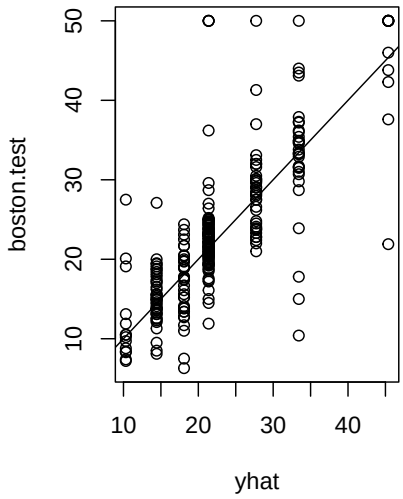

```
tree.pred <- predict(prune.carseats, Carseats.test,
                     type = "class")
table(tree.pred, High.test)
#>          High.test
#> tree.pred  No Yes
#>       No  102  31
#>       Yes  15  52
(102 + 52) / 200
#> [1] 0.77

## Fitting Regression Trees

###
set.seed(1)
train <- sample(1:nrow(Boston), nrow(Boston) / 2)
tree.boston <- tree(medv ~ ., Boston, subset = train)
summary(tree.boston)
#>
#> Regression tree:
#> tree(formula = medv ~ ., data = Boston, subset = train)
#> Variables actually used in tree construction:
#> [1] "rm"    "lstat" "crim"  "age"
#> Number of terminal nodes:  7
#> Residual mean deviance:  10.38 = 2555 / 246
```

```
#> Distribution of residuals:
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> -10.1800  -1.7770  -0.1775   0.0000   1.9230  16.5800
###
plot(tree.boston)
text(tree.boston, pretty = 0)
###
cv.boston <- cv.tree(tree.boston)
plot(cv.boston$size, cv.boston$dev, type = "b")
```



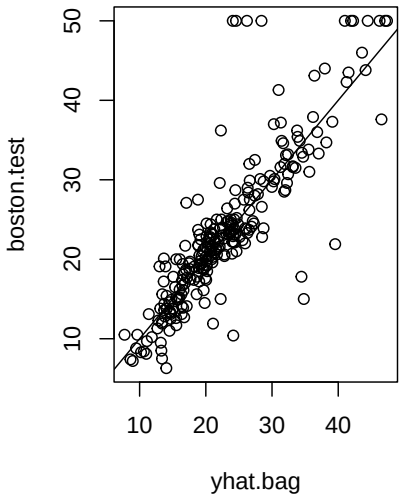

```
###
prune.boston <- prune.tree(tree.boston, best = 5)
plot(prune.boston)
text(prune.boston, pretty = 0)
###
yhat <- predict(tree.boston, newdata = Boston[-train, ])
boston.test <- Boston[-train, "medv"]
plot(yhat, boston.test)
abline(0, 1)
```



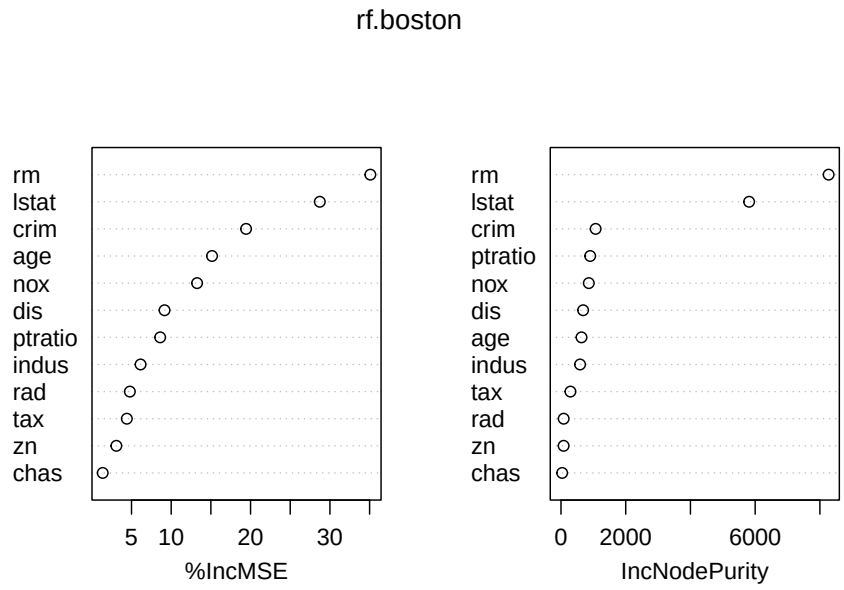



```
mean((yhat - boston.test)^2)
#> [1] 35.28688

## Bagging and Random Forests

###
library(randomForest)
#> randomForest 4.7-1
#> Type rfNews() to see new features/changes/bug fixes.
set.seed(1)
bag.boston <- randomForest(medv ~ ., data = Boston,
                           subset = train, mtry = 12, importance = TRUE)
bag.boston
#>
#> Call:
#>  randomForest(formula = medv ~ ., data = Boston, mtry = 12, importance = TRUE,
#>                Type of random forest: regression
#>                      Number of trees: 500
#> No. of variables tried at each split: 12
#>
#>           Mean of squared residuals: 11.40162
#>                     % Var explained: 85.17
###
yhat.bag <- predict(bag.boston, newdata = Boston[-train, ])
```

```
plot(yhat.bag, boston.test)
abline(0, 1)
mean((yhat.bag - boston.test)^2)
#> [1] 23.41916
###
bag.boston <- randomForest(medv ~ ., data = Boston,
                           subset = train, mtry = 12, ntree = 25)
yhat.bag <- predict(bag.boston, newdata = Boston[-train, ])
mean((yhat.bag - boston.test)^2)
#> [1] 25.75055
###
set.seed(1)
rf.boston <- randomForest(medv ~ ., data = Boston,
                          subset = train, mtry = 6, importance = TRUE)
yhat.rf <- predict(rf.boston, newdata = Boston[-train, ])
mean((yhat.rf - boston.test)^2)
#> [1] 20.06644
###
importance(rf.boston)
#>           %IncMSE IncNodePurity
#> crim    19.435587    1070.42307
#> zn       3.091630      82.19257
#> indus    6.140529     590.09536
#> chas     1.370310      36.70356
#> nox     13.263466     859.97091
#> rm      35.094741    8270.33906
#> age     15.144821     634.31220
#> dis      9.163776     684.87953
#> rad      4.793720      83.18719
#> tax      4.410714     292.20949
#> ptratio  8.612780     902.20190
#> lstat   28.725343    5813.04833
###
varImpPlot(rf.boston)
```



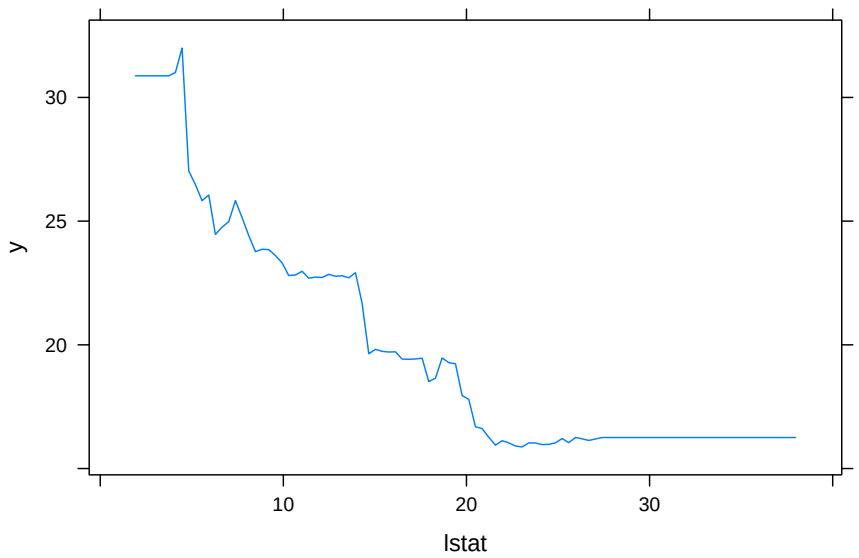

```
## Boosting

###
library(gbm)
#> Loaded gbm 2.1.8
```




```
set.seed(1)
boost.boston <- gbm(medv ~ ., data = Boston[train, ],
                    distribution = "gaussian", n.trees = 5000,
                    interaction.depth = 4)
###
summary(boost.boston)
#>             var     rel.inf
#> rm           rm 44.48249588
#> lstat     lstat 32.70281223
#> crim       crim  4.85109954
#> dis         dis  4.48693083
#> nox         nox  3.75222394
#> age         age  3.19769210
#> ptratio ptratio  2.81354826
#> tax         tax  1.54417603
#> indus     indus  1.03384666
#> rad         rad  0.87625748
#> zn           zn  0.16220479
#> chas       chas  0.09671228
###
plot(boost.boston, i = "rm")
plot(boost.boston, i = "lstat")
###
yhat.boost <- predict(boost.boston,
```

```
                      newdata = Boston[-train, ], n.trees = 5000)
mean((yhat.boost - boston.test)^2)
#> [1] 18.39057
###
boost.boston <- gbm(medv ~ ., data = Boston[train, ],
                    distribution = "gaussian", n.trees = 5000,
                    interaction.depth = 4, shrinkage = 0.2, verbose = F)
yhat.boost <- predict(boost.boston,
                      newdata = Boston[-train, ], n.trees = 5000)
mean((yhat.boost - boston.test)^2)
#> [1] 16.54778

## Bayesian Additive Regression Trees

###
library(BART)
#> Loading required package: nlme
#> Loading required package: nnet
#> Loading required package: survival
x <- Boston[, 1:12]
y <- Boston[, "medv"]
xtrain <- x[train, ]
ytrain <- y[train]
xtest <- x[-train, ]
ytest <- y[-train]
set.seed(1)
bartfit <- gbart(xtrain, ytrain, x.test = xtest)
#> *****Calling gbart: type=1
#> *****Data:
#> data:n,p,np: 253, 12, 253
#> y1,yn: 0.213439, -5.486561
#> x1,x[n*p]: 0.109590, 20.080000
#> xp1,xp[np*p]: 0.027310, 7.880000
#> *****Number of Trees: 200
#> *****Number of Cut Points: 100 ... 100
#> *****burn,nd,thin: 100,1000,1
#> *****Prior:beta,alpha,tau,nu,lambda,offset: 2,0.95,0.795495,3,3.71636,21.7866
#> *****sigma: 4.367914
#> *****w (weights): 1.000000 ... 1.000000
#> *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,12,0
#> *****printevery: 100
#>
#> MCMC
#> done 0 (out of 1100)
#> done 100 (out of 1100)
```

```
#> done 200 (out of 1100)
#> done 300 (out of 1100)
#> done 400 (out of 1100)
#> done 500 (out of 1100)
#> done 600 (out of 1100)
#> done 700 (out of 1100)
#> done 800 (out of 1100)
#> done 900 (out of 1100)
#> done 1000 (out of 1100)
#> time: 4s
#> trcnt,tecnt: 1000,1000
###
yhat.bart <- bartfit$yhat.test.mean
mean((ytest - yhat.bart)^2)
#> [1] 15.94718
###
ord <- order(bartfit$varcount.mean, decreasing = T)
bartfit$varcount.mean[ord]
#>     nox   lstat     tax     rad      rm   indus    chas
#>  22.952  21.329  21.250  20.781  19.890  19.825  19.051
#> ptratio     age      zn     dis    crim
#>  18.976  18.274  15.952  14.457  11.007
###
```



Chapter 13

Trees #9 The OJ Dataset

This problem involves the OJ data set which is part of the ISLR package.

```
#>
#> Attaching package: 'dplyr'
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union
```

```
dplyr::glimpse(OJ)
#> Rows: 1,070
#> Columns: 18
#> $ Purchase       <fct> CH, CH, CH, MM, CH, CH, CH, CH, CH,~
#> $ WeekofPurchase <dbl> 237, 239, 245, 227, 228, 230, 232, ~
#> $ StoreID        <dbl> 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 7,~
#> $ PriceCH        <dbl> 1.75, 1.75, 1.86, 1.69, 1.69, 1.69,~
#> $ PriceMM        <dbl> 1.99, 1.99, 2.09, 1.69, 1.69, 1.99,~
#> $ DiscCH         <dbl> 0.00, 0.00, 0.17, 0.00, 0.00, 0.00,~
#> $ DiscMM         <dbl> 0.00, 0.30, 0.00, 0.00, 0.00, 0.00,~
#> $ SpecialCH      <dbl> 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,~
#> $ SpecialMM      <dbl> 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,~
#> $ LoyalCH        <dbl> 0.500000, 0.600000, 0.680000, 0.400~
#> $ SalePriceMM    <dbl> 1.99, 1.69, 2.09, 1.69, 1.69, 1.99,~
#> $ SalePriceCH    <dbl> 1.75, 1.75, 1.69, 1.69, 1.69, 1.69,~
#> $ PriceDiff      <dbl> 0.24, -0.06, 0.40, 0.00, 0.00, 0.30~
#> $ Store7         <fct> No, No, No, No, Yes, Yes, Yes, Yes,~
#> $ PctDiscMM      <dbl> 0.000000, 0.150754, 0.000000, 0.000~
```

```
#> $ PctDiscCH      <dbl> 0.000000, 0.000000, 0.091398, 0.000~
#> $ ListPriceDiff  <dbl> 0.24, 0.24, 0.23, 0.00, 0.00, 0.30,~
#> $ STORE          <dbl> 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,~
```

(a) train/test Split

Q: Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

A: Since this is my first time seeing this dataset, here is my quick overview: The OJ dataset contains 1,070 purchases of two brands of orange juice ('Citrus Hill' or 'Minute Maid'), captured in the values of the Purchase variable (CH or MM). The remaining 17 predictors are characteristics of the customer, product, store, etc. Throughout this question we are basically tasked with predicting which orange juice the customer purchased based on these statistics.

I create train and test below.

```
set.seed(5)
train_index <- sample(1:nrow(OJ), 800)
train <- OJ[train_index, ]
test <- OJ[-train_index, ]
```

(b) Classification Tree

Q: Fit a tree to the training data, with Purchase as the response and the other variables as predictors. Use the summary() function to produce summary statistics about the tree, and describe the results obtained. What is the training error rate? How many terminal nodes does the tree have?

A: The classification tree has 7 terminal nodes and a training error rate of 18.38%.

```
tree_model <- tree(Purchase ~ ., train)
summary(tree_model)
#>
#> Classification tree:
#> tree(formula = Purchase ~ ., data = train)
#> Variables actually used in tree construction:
#> [1] "LoyalCH"       "PriceDiff"     "ListPriceDiff"
#> Number of terminal nodes:  9
#> Residual mean deviance:  0.7347 = 581.1 / 791
#> Misclassification error rate: 0.1662 = 133 / 800
```

Despite there being 17 predictors in the dataset, only three were used in splits. These were:

```
LoyalCH - Customer brand loyalty for CH
PriceDiff - Sale price of MM less sale price of CH
DiscCH - Discount offered for CH
```

(c) tree() - Text Interpretation

Q: Type in the name of the tree object in order to get a detailed text output. Pick one of the terminal nodes, and interpret the information displayed.

A: I print the text output below.

```
tree_model
#> node), split, n, deviance, yval, (yprob)
#>       * denotes terminal node
#>
#>  1) root 800 1068.00 CH ( 0.61250 0.38750 )
#>    2) LoyalCH < 0.5036 346  412.40 MM ( 0.28324 0.71676 )
#>      4) LoyalCH < 0.280875 164  125.50 MM ( 0.12805 0.87195 )
#>        8) LoyalCH < 0.0356415 56   10.03 MM ( 0.01786 0.98214 ) *
#>        9) LoyalCH > 0.0356415 108  103.50 MM ( 0.18519 0.81481 ) *
#>      5) LoyalCH > 0.280875 182  248.00 MM ( 0.42308 0.57692 )
#>       10) PriceDiff < 0.05 71   67.60 MM ( 0.18310 0.81690 ) *
#>       11) PriceDiff > 0.05 111  151.30 CH ( 0.57658 0.42342 ) *
#>    3) LoyalCH > 0.5036 454  362.00 CH ( 0.86344 0.13656 )
#>      6) PriceDiff < -0.39 31   40.32 MM ( 0.35484 0.64516 )
#>       12) LoyalCH < 0.638841 10    0.00 MM ( 0.00000 1.00000 ) *
#>       13) LoyalCH > 0.638841 21   29.06 CH ( 0.52381 0.47619 ) *
#>      7) PriceDiff > -0.39 423  273.70 CH ( 0.90071 0.09929 )
#>       14) LoyalCH < 0.705326 135  143.00 CH ( 0.77778 0.22222 )
#>         28) ListPriceDiff < 0.255 67   89.49 CH ( 0.61194 0.38806 ) *
#>         29) ListPriceDiff > 0.255 68   30.43 CH ( 0.94118 0.05882 ) *
#>       15) LoyalCH > 0.705326 288   99.77 CH ( 0.95833 0.04167 ) *
```

Choosing node 11), which is a terminal node as it is marked by a *:

First the root node: 1) root 800 1064.00 CH (0.61750 0.38250)

This means that, at the root node, there are 800 observations, the deviance is 1064.00, the overall prediction is CH and the split is 61.75% CH vs 38.25% MM.

We can see that, from the root node, three splits take place to produce the terminal node labelled by 11):

```
A split at LoyalCH = 0.5036
A split at LoyalCH = 0.142213
A split at PriceDiff = 0.235
```

```
 1) root 800 1064.00 CH ( 0.61750 0.38250 )
   2) LoyalCH < 0.5036 354  435.50 MM ( 0.30508 0.69492 )
     4) LoyalCH < 0.142213 100   45.39 MM ( 0.06000 0.94000 ) *
     5) LoyalCH > 0.142213 254  342.20 MM ( 0.40157 0.59843 )
      10) PriceDiff < 0.235 136  153.00 MM ( 0.25000 0.75000 ) *
      11) PriceDiff > 0.235 118  160.80 CH ( 0.57627 0.42373 ) *
```

Node 11) is therefore the subset of purchases where 0.142213 $<$ LoyalCH $<$ 0.5036 and PriceDiff $>$ 0.235. The overall prediction is CH, and the node seems quite impure with 57.627% CH vs 42.373% MM.

There are 118 observations in the node, and from the percentages above we know that 68/118 are CH and 50/118 are MM (demonstrated below).

```
train %>%
  filter(LoyalCH < 0.5036,
         LoyalCH > 0.142213,
         PriceDiff > 0.235) %>%
  select(Purchase) %>%
  table()
#> Purchase
#> CH MM
#> 57 54
```

Based on the formula on page 325 for the overall deviance of a classification tree

$$(-2 \sum m \sum knmklog(p^m k))$$

where the overall deviance sum is over m regions (terminal nodes). We calculate can the deviance of node 11) only using the code below:

```
-2 * (68 * log(68/118) + 50 * log(50/118))
#> [1] 160.8262
```

tree() reports the number as 160.80, and testing with other nodes revealed this is because it's rounding the result to 4 significant figures.

(d) Plotting

Q: Create a plot of the tree, and interpret the results.

A: LoyalCH is certainly the most important variable (the top 3 nodes all split on this variable), followed by PriceDiff and DiscCH. We can see node 11) is the third terminal node (from left $\rightarrow$ right).

LoyalCH ranges from 0 to 1, so the first split sends those less loyal to Citrus Hill (CH) orange juice to the left and those more loyal to the right:

plot(tree_model) text(tree_model, pretty = 0, cex = 0.7)

Those that scored lowest in Citrus Hill loyalty (LoyalCH < 0.142213) were predicted to buy Minute Maid (MM), which isn't surprising. Those that were slightly more loyal to CH ($0.142213 <$ LoyalCH < 0.5036) would still buy MM if it wasn't too much more expensive (PriceDiff < 0.235), but if the price difference is large enough (CH was much cheaper) they could end up purchasing CH.

Those on the far-right terminal node are the most loyal to CH (LoyalCH > 0.705699), so it should be unsurprising that this is their predicted purchase. Those with slightly lower brand loyalty ($0.5036 <$ LoyalCH < 0.705699) would still purchase CH if it was much cheaper (PriceDiff > 0.25), or if it wasn't but was sufficiently discounted (PriceDiff < 0.25 & DiscCH > 0.15). For those cases where CH wasn't much cheaper (PriceDiff < 0.25) and wasn't sufficiently discounted (DiscCH < 0.15), the predicted purchase actually ended up being MM.

This was a much more detailed explanation, but this could be summarized at a much higher level in the following way: people go with the brand they are more loyal towards, but there are some edge cases (based on discounts and the prices relative to one another) that can sway people against their usual brand loyalties.

(e) Test Error

Q: Predict the response on the test data, and produce a confusion matrix comparing the test labels to the predicted test labels. What is the test error rate?

A:

Here is the confusion matrix for the unpruned regression tree:

```
test_pred <- predict(tree_model, test, type = "class")
table(test_pred, test_actual = test$Purchase)
#>           test_actual
#> test_pred  CH  MM
#>        CH 148  32
#>        MM  15  75
```

13.1 test_actual

13.2 test_pred CH MM

13.3 CH 125 32

13.4 MM 34 79

The test error rate corresponding to it:

1 - mean(test_pred == test$Purchase)

13.5 [1] 0.2444444

CH was the most common orange juice in train so, for comparison, a baseline classifier (that predicted CH for all observations in test) would have the following error rate:

1 - mean(test$Purchase == "CH")

13.6 [1] 0.4111111

(f) Cost-Complexity Pruning

Q: Apply the cv.tree() function to the training set in order to determine the optimal tree size.

A:

Since our goal appears to be low test error, I specify FUN = prune.misclass. This indicates that we want the classification error rate to guide the cross-validation and pruning process, rather than the default for the cv.tree() function, which is deviance.

```
set.seed(2)
cv_tree_model <- cv.tree(tree_model, K = 10, FUN = prune.misclass)
cv_tree_model
#> $size
#> [1] 9 6 5 3 2 1
#>
#> $dev
```

```
#> [1] 149 149 149 173 172 310
#>
#> $k
#> [1]  -Inf   0.0   1.0   8.5   9.0 150.0
#>
#> $method
#> [1] "misclass"
#>
#> attr(,"class")
#> [1] "prune"         "tree.sequence"
```

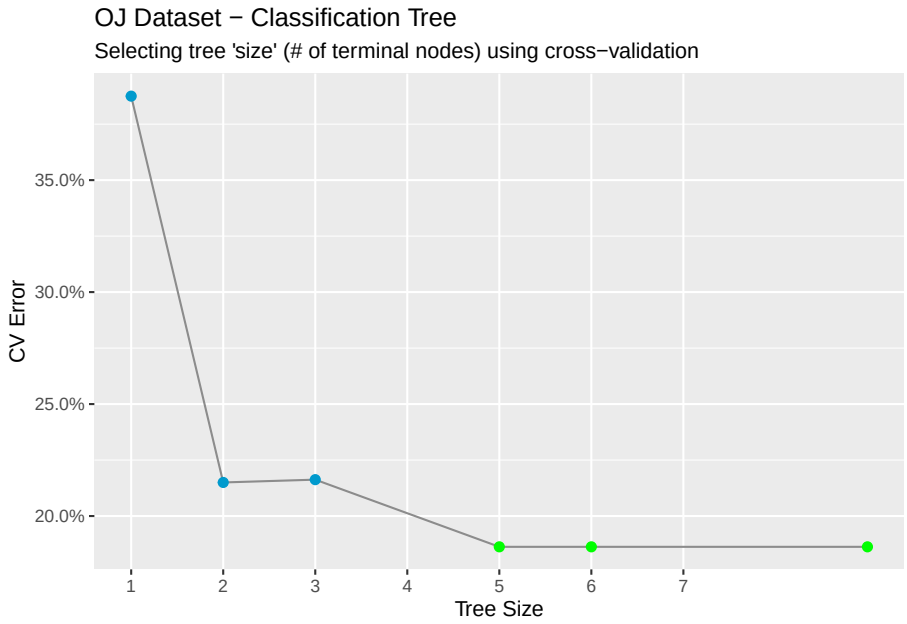
(g) CV Error Plot

Q: Produce a plot with tree size on the x-axis and cross-validated classification error rate on the y-axis.

A:

The plot is below. Note that cv_tree_model$size is the number of terminal nodes (so 1 means it is just the root node with no
gives the total number of errors made from the out-of-fold predictions during cross-validation (only because we specified FUN = prune.misclass - omitting this would mean this reports the deviance). From this we can obtain the cross-validation error rate.

```
data.frame(size = cv_tree_model$size, CV_Error = cv_tree_model$dev / nrow(train)) %>%
  mutate(min_CV_Error = as.numeric(min(CV_Error) == CV_Error)) %>%
  ggplot(aes(x = size, y = CV_Error)) +
  geom_line(col = "grey55") +
  geom_point(size = 2, aes(col = factor(min_CV_Error))) +
  scale_x_continuous(breaks = seq(1, 7), minor_breaks = NULL) +
  scale_y_continuous(labels = scales::percent_format()) +
  scale_color_manual(values = c("deepskyblue3", "green")) +
  theme(legend.position = "none") +
  labs(title = "OJ Dataset - Classification Tree",
       subtitle = "Selecting tree 'size' (# of terminal nodes) using cross-validation",
       x = "Tree Size",
       y = "CV Error")
```



(h) Best Tree - CV Error

Q: Which tree size corresponds to the lowest cross-validated classification error rate?

A:

Of the sequence of trees generated, trees of sizes 4 and 7 have the same cross-validation error. It makes sense to select the more parsimonious model here with 4 terminal nodes.

(i) Best Tree - Selecting

Q: Produce a pruned tree corresponding to the optimal tree size obtained using cross-validation. If cross-validation does not lead to selection of a pruned tree, then create a pruned tree with five terminal nodes.

A:

I produce the tree with 4 terminal nodes. Interestingly we have the same cross-validation error as the full tree with 7 terminal nodes, but have only split on LoyalCH to achieve this. This would have the added benefit of simplifying the interpretation in part (d).

```
pruned_tree_model <- prune.tree(tree_model, best = 4)
pruned_tree_model
#> node), split, n, deviance, yval, (yprob)
#>       * denotes terminal node
#>
#> 1) root 800 1068.00 CH ( 0.61250 0.38750 )
#>   2) LoyalCH < 0.5036 346  412.40 MM ( 0.28324 0.71676 )
#>     4) LoyalCH < 0.280875 164  125.50 MM ( 0.12805 0.87195 ) *
#>     5) LoyalCH > 0.280875 182  248.00 MM ( 0.42308 0.57692 ) *
#>   3) LoyalCH > 0.5036 454  362.00 CH ( 0.86344 0.13656 )
#>     6) PriceDiff < -0.39 31   40.32 MM ( 0.35484 0.64516 ) *
#>     7) PriceDiff > -0.39 423  273.70 CH ( 0.90071 0.09929 ) *
```

(j) Training Error Comparison

Q: Compare the training error rates between the pruned and unpruned trees. Which is higher?

A: Here is the training error for the unpruned tree (7 terminal nodes):

```
mean(predict(tree_model, type = "class") != train$Purchase)
#> [1] 0.16625
```

The same for the pruned tree (4 terminal nodes):

```
mean(predict(pruned_tree_model, type = "class") != train$Purchase)
#> [1] 0.18875
```

The training error for the pruned tree is higher. This isn't surprising - we would expect the training error of a tree to monotonically decrease as its flexibility (number of splits) increases.

(k) Test Error Comparison

Q: Compare the test error rates between the pruned and unpruned trees. Which is higher?

A:

The test error for the unpruned tree:

```
mean(predict(tree_model, type = "class", newdata = test) != test$Purchase)
#> [1] 0.1740741
```

The same for the pruned tree:

```
mean(predict(pruned_tree_model, type = "class", newdata = test) != test$Purchase)
#> [1] 0.2
```

Now the order has reversed and the error is higher for the unpruned tree.

It is interesting that the cross-validation errors were in fact equal but the test error is noticeably lower for the simpler tree. A lot of this probably comes from random variability when working with a small dataset; using a different random state for the CV folds and train/test split would likely change all of these results (particularly because decision trees are such high-variance approaches).

Chapter 14

Project (E1)

14.1 A project to call your own

Pick a dataset, any dataset…

…and do something with it. That is your first Analytics 2 project. Make us both proud, in a nutshell. More details below.

14.2 May be too long, but please do read

This project for this class will consist of analysis on a dataset of your own choosing. **Please make sure I am ok with your choice.** The dataset may already exist, or you may collect your own data using a survey or by conducting an experiment. You can choose the data based on your interests or based on work in other courses or research projects. The goal of this project is for you to demonstrate proficiency in the techniques we have covered in this class (and beyond, if you like) and apply them to a novel dataset in a meaningful way.

14.3 Data

In order for you to have the greatest chance of success with this project it is important that you choose a manageable dataset. This means that the data should be readily accessible and large enough that multiple relationships can be explored. As such, your dataset must have at least 50 observations and between 10 to 20 variables (exceptions can be made but you must speak with me first). The dataset's variables should include categorical variables, discrete numerical variables, and continuous numerical variables.

All analyses must be done in RStudio. If you are using a dataset that comes in a format that we haven't encountered in class, make sure that you are able to load it into RStudio as this can be tricky depending on the source. If you are having trouble ask for help before it is too late.

Reusing datasets from class: Do not reuse datasets used in examples / homework in the class.

14.4 Components

14.4.1 Project proposal

This is a draft of the introduction section of your project as well as a data analysis plan and your dataset. Each section should be no more than 1 page (excluding figures). You can check a print preview to confirm length.

> Your write up and all analysis including visuals must be done using R Markdown.

14.4.1.1 Section 1 - Introduction:

The introduction should introduce your general research question and your data (where it came from, how it was collected, what are the cases, what are the variables, etc.).

14.4.1.2 Section 2 - Data analysis plan:

The data analysis plan should include:

- The outcome (dependent, response, Y) and predictor (independent, explanatory, X) variables you will use to answer your question.
- The comparison groups you will use, if applicable.
- Very preliminary exploratory data analysis, including some summary statistics and visualizations, along with some explanation on how they help you learn more about your data. (You can add to these later as you work on your project..)
- The statistical method(s) that you believe will be useful in answering your question(s). (You can update these later as you work on your project.)
- Ideally you will use at least two out of these options: tree methods, linear regression, and classification (like logistic regression).
- What results from these specific statistical methods are needed to support your hypothesized answer?

14.4.1.3 Section 3 - Data:

In yuor write up, include enough details that I understand what your raw data looked like and included.

14.4.2 Project

14.4.2.1 Write up

After providing the description of your dataset and research question in the introduction use the remainder of your write up to showcase how you have arrived at an answer / answers to your question using any techniques we have learned in this class (and some beyond, if you're feeling adventurous). The goal is not to do an exhaustive data analysis i.e., do not calculate every statistic and procedure you have learned for every variable, but rather let me know that you are proficient at asking meaningful questions and answering them with results of data analysis, that you are proficient in using R, and that you are proficient at interpreting and presenting the results. Focus on methods that help you begin to answer your research questions. You do not have to apply every statistical procedure we learned. Also pay attention to your presentation. Neatness, coherency, and clarity will count.

Your write up must also include a one to two page conclusion and discussion. This will require a summary of what you have learned about your research question along with statistical arguments supporting your conclusions. Also critique your own methods and provide suggestions for improving your analysis. Issues pertaining to the reliability and validity of your data, and appropriateness of the statistical analysis should be discussed here. A paragraph on what you would do differently if you were able to start over with the project or what you would do next if you were going to continue work on the project should also be included.

> The project is very open ended. You should create some kind of compelling visualization(s) of this data in R.

There is no limit on what tools or packages you may use, but sticking to packages we learned in class (ISLR and R4DS) is required. You do not need to visualize all of the data at once. A single high quality visualization will receive a much higher grade than a large number of poor quality visualizations.

Before you finalize your write up, make sure your chunks are turned off with `echo = FALSE`. **Exception:** If you want to highlight something specific about a piece of code, you're welcomed to show that portion. [See below: I will also want a copy of the raw .Rmd file not just the html output.]

You can add sections as you see fit to the project but make sure you have a section called Introduction at the beginning and a section called Conclusion at the end. The rest is up to you!

14.4.2.2 Presentation

10 minutes maximum.

You can use any software you like for your final presentation, including R Markdown to create your slides. There isn't a limit to how many slides you can use, just a time limit (10 minutes total). Perhaps try `ioslides` or `beamer`. Your presentation should not just be an account of everything you tried ("then we did this, then we did this, etc."), instead it should convey what choices you made, and why, and what you found.

14.4.2.3 Delivarables

Your submission should include

- RMarkdown file (formatted to clearly present all of your code and results)
- HTML file
- Dataset(s) (in csv or RData format, in a `/data` folder)
- Presentation (if using Keynote/PowerPoint/Google Slides, export to PDF and put in repo, in a `/presentation` folder)

Style and format does count for this assignment, so please take the time to make sure everything looks good and your data and code are properly formated.

14.5 Grading

Total	100 pts
Introduction	20 pts
Data analysis plan	20 pts
Data Methods and code quality	50 pts
Organization	10 pts

Grading of the project will take into account the following:

- Content - What is the quality of research and/or policy question and relevancy of data to those questions?

- Correctness - Are statistical procedures carried out and explained correctly?
- Writing and Presentation - What is the quality of the statistical presentation, writing, and explanations?
- Creativity and Critical Thought - Is the project carefully thought out? Are the limitations carefully considered? Does it appear that time and effort went into the planning and implementation of the project?

A general breakdown of scoring is as follows:

- 90%-100% - Outstanding effort. Student understands how to apply all statistical concepts, can put the results into a cogent argument, can identify weaknesses in the argument, and can clearly communicate the results to others.
- 80%-89% - Good effort. Student understands most of the concepts, puts together an adequate argument, identifies some weaknesses of their argument, and communicates most results clearly to others.
- 70%-79% - Passing effort. Student has misunderstanding of concepts in several areas, has some trouble putting results together in a cogent argument, and communication of results is sometimes unclear.
- 60%-69% - Struggling effort. Student is making some effort, but has misunderstanding of many concepts and is unable to put together a cogent argument. Communication of results is unclear.
- Below 60% - Student is not making a sufficient effort.

Late penalty:

- Late, but within 24 hours of due date/time: -20% (only applies to written portion, there is no option to do your presentation later)
- Any later: no credit