Adamek's Complete R Collection

John F. Adamek

2023-02-01

Contents

1 The Layout **5**

2 General Content **7**

2.1 Base R 7

2.2 Tidy 18

2.3 Dplyr 30

2.4 Ggplot2 34

2.5 Functions 55

2.6 If Statements 57

2.7 For() Loops 58

2.8 While Loops 59

3 Lab Stuff **63**

3.1 ETC 63

3.2 EPPL 63

4 Courses **65**

4.1 STAT 420 65

4.2 EPSY 582 65

4.3 PSYC 594 65

4.4 EPSY 590 65

4.5 PSYC 581 65

5 UIUC Projects **67**

5.1 Simulation Project (Mid-Term: Stat420) 67

5.2 EPSY590 Final Baseball Project 67

5.3 Best Research Practices stroop final project 67

5.4 NHANES (ACSM Abstract: 2021) 67

5.5 Yoga (ACSM Abstract: 2022) 67

6 Personal Projects **69**

6.1 Betting Models 69

6.2 My Functions 69

7 Categories **71**

7.1 NYY Part 1 71

8 Statistics **89**

9 Machine Learning **91**

Chapter 1

The Layout

The purpose of this was to have a single location of all my past personal projects for easy access. This would include basic things such a base R, ggplot, tidy, and use of functions as well as content from courses I took as part of the doctoral program at the University of Illinois at Urbana-Champaign. A major aspect that helped me learn R quickly was when I was 'playing' around with R on personal projects. Therefore, this will also include those projects consisting of baseball and sports modeling code and others.

This book is broken down by:

Categories	Description
General	Basic R codes
Lab	Related to ETC and ExPPL Stuff created
Classes	Code from classes not in the General Category
UIUC Projects	Mid-terms, Final projects, abstracts
Personal Projects	All my personal projects
Statistics	Hypothesis tests (ANOVA, Regression)
Predictive Models	K-NN, etc
Others	

General

- Dplyr Basics
- For Loops
  - Nested For Loops
- While Loops
  - Nested While Loops

- If Statements
- ifelse statements
- Functions
- with for loops
- with while loops
- with if statements

Lab

- ETC
  - Visits & QR
  - Fitbit
  - QUestionnaires (ESSQ, HADS)
  - Cognitive
- ExPPL
  - Class cognitive codebook

Classes

(in order)

- STAT 420:
- EPSY 582: Advanced Statistics Methods (Kern)
- PSYC 594: Multivariate Analysis (Yan)
- ESPY 590: Int to R Data Comp (Kern)
- PSYC 581: Applied Regression Analysis (Yan)

UIUC Projects

- Simulation Project (stat420)
- Final and Midterms

Chapter 2

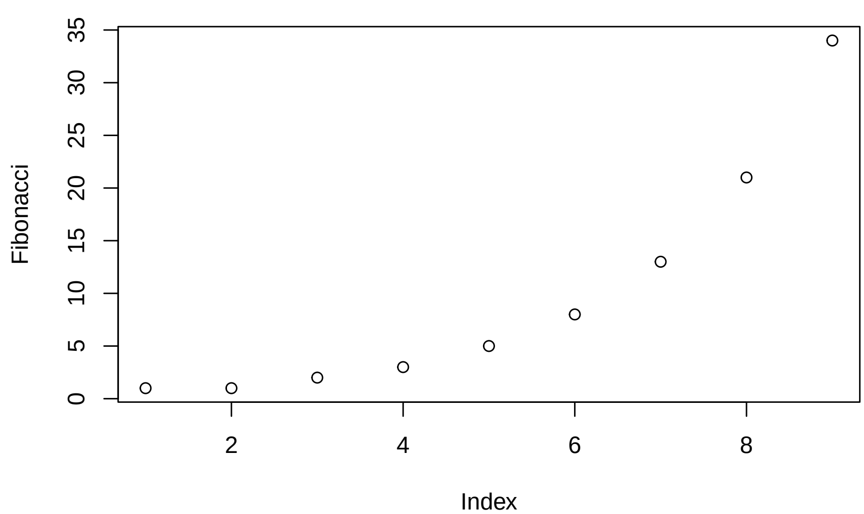
General Content

2.1 Base R

2.1.1 plot()

Simple plot

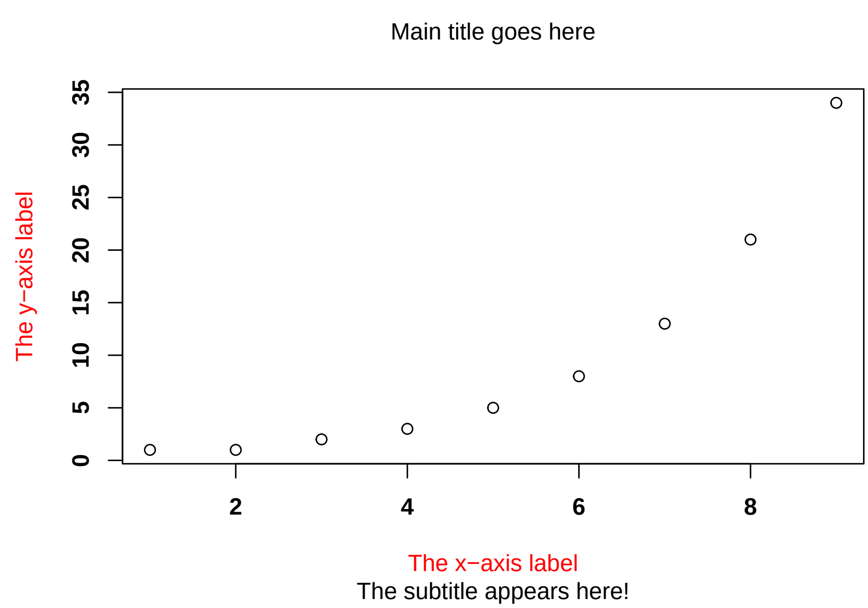
```
Fibonacci <- c(1, 1, 2, 3, 5, 8, 13, 21, 34)
plot(Fibonacci)
```



Labels

- main: A character string containing the title.
- sub: A character string containing the subtitle.
- xlab: A character string containing the x-axis label.
- ylab: A character string containing the y-axis label.
- Font styles: font.main, font.sub, font.lab, font.axis
- Font colours: col.main, col.sub, col.lab, col.axis
- Font size: cex.main, cex.sub, cex.lab, cex.axis

```
plot(Fibonacci,
   main = "Main title goes here",
   sub = "The subtitle appears here!",
   xlab = "The x-axis label",
   ylab = "The y-axis label",
   font.main = 1,            # plain text for title
   cex.main = 1,             # normal size for title
   font.axis = 2,            # bold text for numbering
   col.lab = "red"           # red colour for axis labels
   )
```

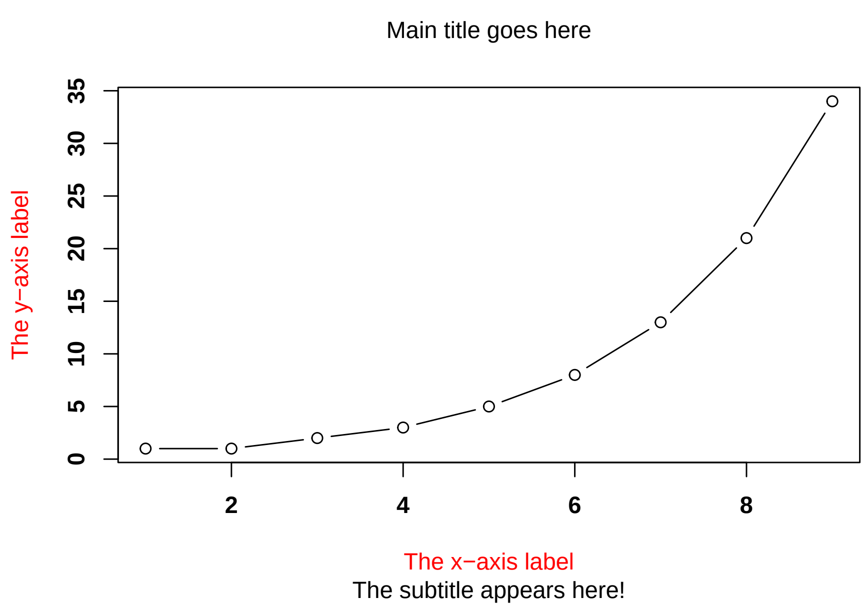


Plot Type

- type = “p”. Draw the points only.

- type = "l". Draw a line through the points.
- type = "o". Draw the line over the top of the points.
- type = "b". Draw both points and lines, but don't over plot.
- type = "h". Draw "histogram-like" vertical bars.
- type = "s". Draw a staircase, going horizontally then vertically.
- type = "S". Draw a Staircase, going vertically then horizontally.
- type = "c". Draw only the connecting lines from the "b" version.
- type = "n". Draw nothing.

```
plot(Fibonacci,
   type = "b",
   main = "Main title goes here",
   sub = "The subtitle appears here!",
   xlab = "The x-axis label",
   ylab = "The y-axis label",
   font.main = 1,           # plain text for title
   cex.main = 1,            # normal size for title
   font.axis = 2,           # bold text for numbering
   col.lab = "red"          # red colour for axis labels
   )
```

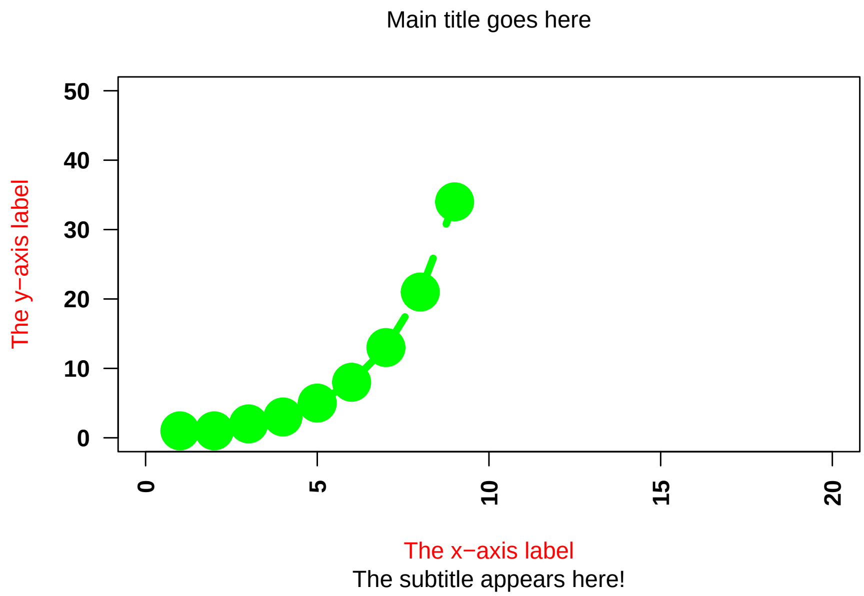


Other customisable Features

- Colour of the plot: the col parameter will do this for you e.g. col="blue"

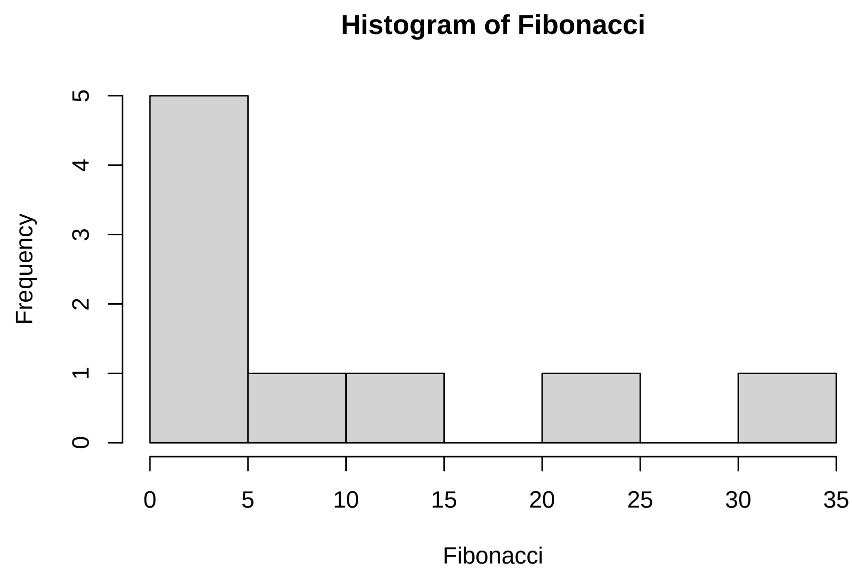
- Character of plot points: the pch parameter. This tells R what symbol to use to draw the points on the plot e.g. pch=12
- Plot size: the cex parameter is used to change the size of your symbols e.g. cex=2
- Line type: the lty parameter is used to tell R which type of line to draw. You can either specify using a number between 0 and 1, or by using a meaningful character string e.g. blank, dashed, solid, dotdash
- Line width: the lwd parameter can be used to specify the width of a line e.g. lwd=3
- Change plot axis scales: the xlim and ylim parameters will do this for you e.g. xlim=c(0, 10)
- Suppress labeling: the ann parameter can be used if you don't want R to label any axis, just set it to false e.g. ann=FALSE
- Suppress axis drawing: the axes parameter is used when you don't want R to draw any axes e.g. axes=FALSE. You can suppress the axes individually using the xant and yant arguments e.g. xant = “n”.
- Label orientation: the las parameter allows you to customise the orientation of the text used to label the individual tick marks e.g. las=2

```
plot(Fibonacci,
   type = "b",
   main = "Main title goes here",
   sub = "The subtitle appears here!",
   xlab = "The x-axis label",
   ylab = "The y-axis label",
   font.main = 1,            # plain text for title
   cex.main = 1,             # normal size for title
   font.axis = 2,            # bold text for numbering
   col.lab = "red",          # red colour for axis labels

   col = "green",            # green colour
   pch = 19,                 # plot character is a solid circle
   cex = 3,                  # plot is 2x normal size
   lty = 2,                  # dashed line
   lwd = 5,                  # line width 5x normal width

   xlim = c(0, 20),          # x-axis min and max values
   ylim = c(0, 50),          # y-axis min and max values
   las = 2                   # changed orientation of text
   )
```

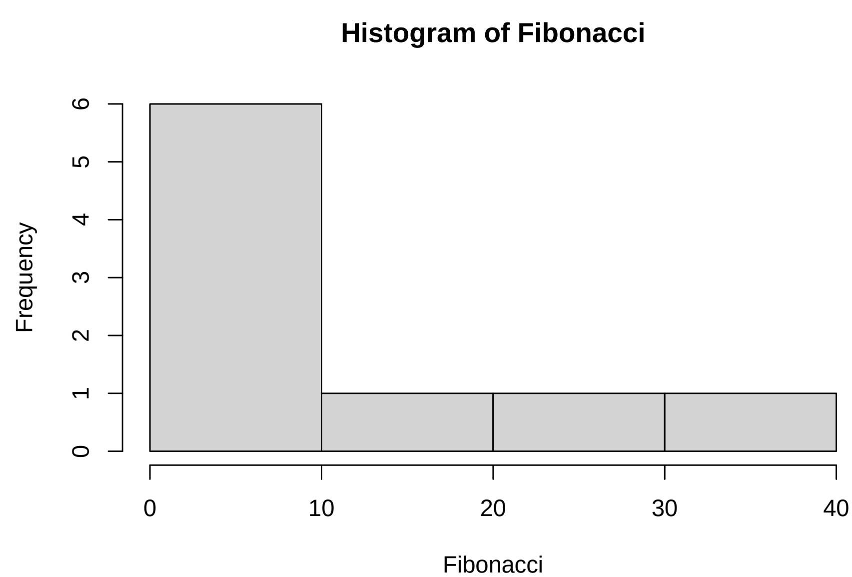


2.1.2 hist()

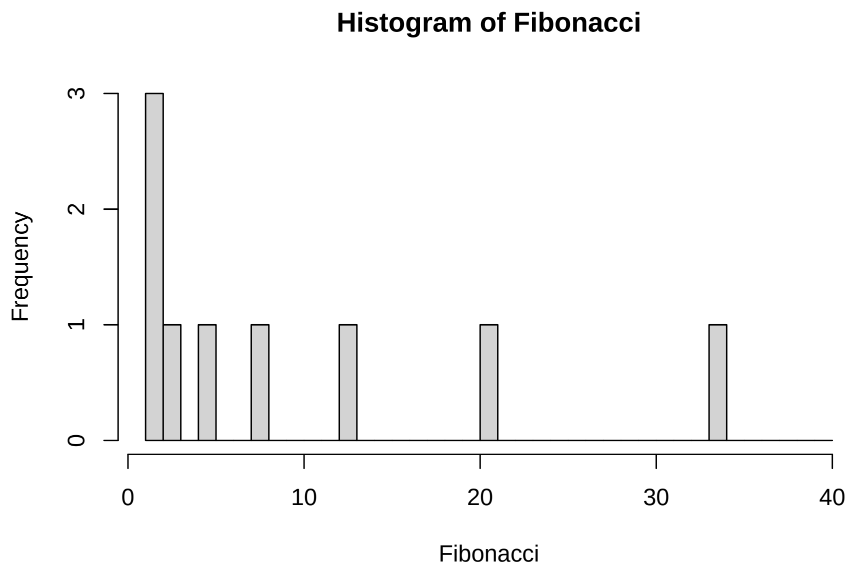
```
hist(Fibonacci)
```




```
hist(Fibonacci, breaks = 3) #add in number of bins
```



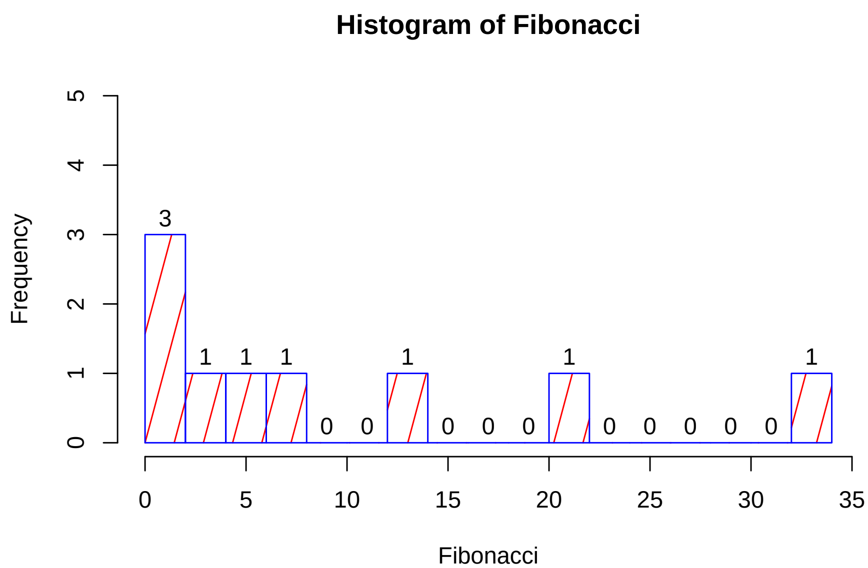
```
hist(Fibonacci, breaks = 1:40) #add in where breaks should start and finish
```



Customising

- Shading lines: the density and angle arguments will allow you to: (1) add diagonal lines to your bars, and (2) indicate the angle of the lines e.g. density = 1 and angle = 45
- Colors: like plot(), you can use the col parameter to change the colour of the shading of the interiors of the bars. You can also use the border argument to set the colour of the bar borders
- Labels: you can label each of the bars in your histogram using the labels argument. You can either write labels = TRUE, or you can choose the labels yourself, by giving a vector of strings e.g. lables = c("Label 1", "Label 2")

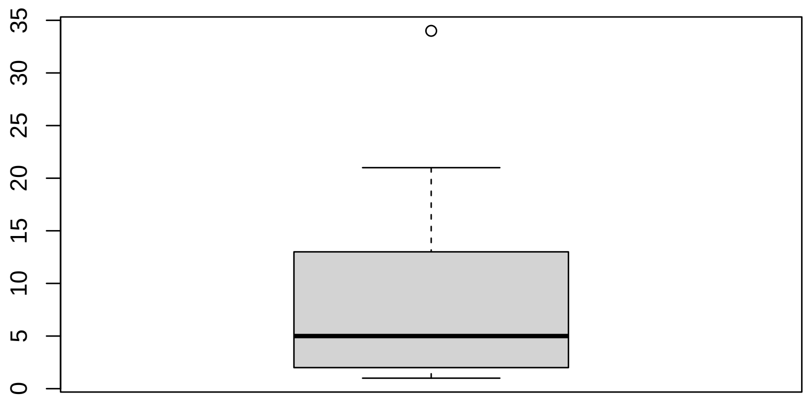
```
hist(Fibonacci,
      breaks = 15,       #add in number of bins
     density = 5,        #5 shading lines per inch
     angle = 75,         #angle of 75 degrees for shading lines
     border = "blue",    #set border colour to blue
     col = "red",        #set colour of shading lines
     labels = TRUE,      #add frequency labels to the bars
     ylim = c(0,5)       #y-axis min and max
     )
```



2.1.3 boxplot()

Boxplots are another useful way of visualising interval or ratio data. They provide you with the information that you see when using the summary() function, and are useful for providing a visual illustration of the median, interquartile range, and the overall range of data.

```
boxplot(Fibonacci)
```

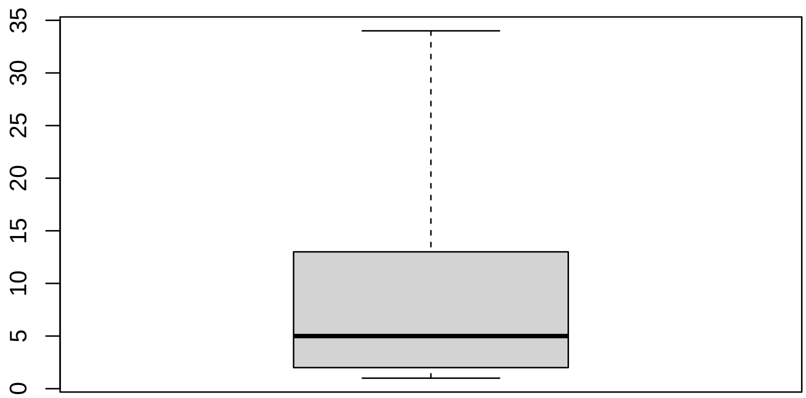



```
summary(Fibonacci)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.000   2.000   5.000   9.778  13.000  34.000
```

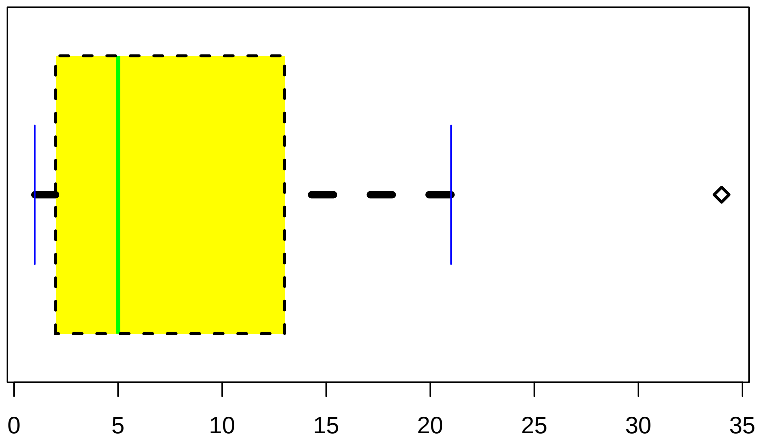
Adjusting the range.By default, R highlights cases that are 1.5x the interquartle range. That means that the top whisker is pulled back to the next highest value that is within the range.

```
boxplot(Fibonacci, range = 2)
```



Customising boxplots

```
boxplot(Fibonacci,
       boxlwd = 2,           # box line width 2
       whisklwd = 5,         # whisker line width 5
       outlwd = 2,           # outlier line width 2
       boxlty = 2,           # dashed box line
       staplecol = "blue",
       medcol = "green",
       boxfill = "yellow",
       outpch = 5,
       varwidth = T,
       horizontal = T
       )
```



2.1.4 Scatterplot with plot()

Scatterplot of the relationship between age and sex

```
plot(mydata$age, mydata$score)
```

Scatterplot with titles and labels

```
plot(mydata$age, mydata$score,
     main = "Relationship between Age and Score",
     xlab = "Age",
     ylab = "Score",
     col = "green",
     pch = 12
     )
```

You can get R to draw a clean straight line to highlight the trajectory using the scatterplot function in the car package

```
library(car)
scatterplot(mydata$age, mydata$score, smooth = F) #smooth stops R from drawing a fancy "smoothed"
```

2.2 Tidy

- pivot_longer()
- pivot_wider()
- separate()
- unite()
- Missing Values

```
library(tidyr)
library(knitr)
library(kableExtra)
```

Pivot Longer

Ex 1.

```
knitr::kable(table4a)
```

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

```
table4a %>%
  pivot_longer(c('1999', '2000'), names_to = 'year', values_to = 'cases')
```

```
## # A tibble: 6 x 3
##   country     year   cases
##   <chr>       <chr>  <int>
## 1 Afghanistan 1999     745
## 2 Afghanistan 2000    2666
## 3 Brazil      1999   37737
## 4 Brazil      2000   80488
## 5 China       1999  212258
## 6 China       2000  213766
```

Ex 2.

```
knitr::kable(relig_income)
```

religion	<$10k	$10-20k	$20-30k	$30-40k	$40-50k	$50-75k	$75-100k	$100-150k	>1
Agnostic	27	34	60	81	76	137	122	109	
Atheist	12	27	37	52	35	70	73	59	
Buddhist	27	21	30	34	33	58	62	39	
Catholic	418	617	732	670	638	1116	949	792	
Don't know/refused	15	14	15	11	10	35	21	17	
Evangelical Prot	575	869	1064	982	881	1486	949	723	
Hindu	1	9	7	9	11	34	47	48	
Historically Black Prot	228	244	236	238	197	223	131	81	
Jehovah's Witness	20	27	24	24	21	30	15	11	
Jewish	19	19	25	25	30	95	69	87	
Mainline Prot	289	495	619	655	651	1107	939	753	
Mormon	29	40	48	51	56	112	85	49	
Muslim	6	7	9	10	9	23	16	8	
Orthodox	13	17	23	32	32	47	38	42	
Other Christian	9	7	11	13	13	14	18	14	
Other Faiths	20	33	40	46	49	63	46	40	
Other World Religions	5	2	3	4	2	7	3	4	
Unaffiliated	217	299	374	365	341	528	407	321	

```
relig_income %>%
  pivot_longer(!religion, names_to = 'income', values_to = 'count')
```

```
## # A tibble: 180 x 3
##    religion income             count
##    <chr>    <chr>              <dbl>
##  1 Agnostic <$10k                 27
##  2 Agnostic $10-20k               34
##  3 Agnostic $20-30k               60
##  4 Agnostic $30-40k               81
##  5 Agnostic $40-50k               76
##  6 Agnostic $50-75k              137
##  7 Agnostic $75-100k             122
##  8 Agnostic $100-150k            109
##  9 Agnostic >150k                 84
## 10 Agnostic Don't know/refused    96
## # ... with 170 more rows
```

Ex 3.

```
knitr::kable(billboard)
```

artist	track	date.entered	wk1	wk2	wk3	wk4
2 Pac	Baby Don't Cry (Keep...	2000-02-26	87	82	72	77
2Ge+her	The Hardest Part Of ...	2000-09-02	91	87	92	NA
3 Doors Down	Kryptonite	2000-04-08	81	70	68	67
3 Doors Down	Loser	2000-10-21	76	76	72	69
504 Boyz	Wobble Wobble	2000-04-15	57	34	25	17
98^0	Give Me Just One Nig...	2000-08-19	51	39	34	26
A*Teens	Dancing Queen	2000-07-08	97	97	96	95
Aaliyah	I Don't Wanna	2000-01-29	84	62	51	41
Aaliyah	Try Again	2000-03-18	59	53	38	28
Adams, Yolanda	Open My Heart	2000-08-26	76	76	74	69
Adkins, Trace	More	2000-04-29	84	84	75	73
Aguilera, Christina	Come On Over Baby (A...	2000-08-05	57	47	45	29
Aguilera, Christina	I Turn To You	2000-04-15	50	39	30	28
Aguilera, Christina	What A Girl Wants	1999-11-27	71	51	28	18
Alice Deejay	Better Off Alone	2000-04-08	79	65	53	48
Allan, Gary	Smoke Rings In The D...	2000-01-22	80	78	76	77
Amber	Sexual	1999-07-17	99	99	96	96
Anastacia	I'm Outta Love	2000-04-01	92	NA	NA	95
Anthony, Marc	My Baby You	2000-09-16	82	76	76	70
Anthony, Marc	You Sang To Me	2000-02-26	77	54	50	43
Avant	My First Love	2000-11-04	70	62	56	43
Avant	Separated	2000-04-29	62	32	30	23
BBMak	Back Here	2000-04-29	99	86	60	52
Backstreet Boys, The	Shape Of My Heart	2000-10-14	39	25	24	15
Backstreet Boys, The	Show Me The Meaning ...	2000-01-01	74	62	55	25
Backstreet Boys, The	The One	2000-05-27	58	50	43	37
Badu, Erkyah	Bag Lady	2000-08-19	67	53	42	41
Baha Men	Who Let The Dogs Out	2000-07-22	99	92	85	76
Barenaked Ladies	Pinch Me	2000-09-09	77	76	69	45
Beenie Man	Girls Dem Sugar	2000-10-21	72	72	63	56
Before Dark	Monica	2000-05-20	95	87	80	80
Bega, Lou	Tricky Tricky	2000-01-29	75	74	87	NA
Big Punisher	It's So Hard	2000-04-22	96	87	75	79
Black Rob	Whoa!	2000-03-04	78	59	53	52
Black, Clint	Been There	2000-02-19	87	73	62	58
Blaque	Bring It All To Me	1999-10-23	73	63	50	42
Blige, Mary J.	Deep Inside	1999-11-13	83	80	80	75
Blige, Mary J.	Give Me You	2000-04-15	97	94	77	76
Blink-182	All The Small Things	1999-12-04	89	76	69	59
Bloodhound Gang	The Bad Touch	2000-03-18	70	62	55	55
Bon Jovi	It's My Life	2000-08-12	64	58	51	51
Braxton, Toni	He Wasn't Man Enough	2000-03-18	63	55	48	39
Braxton, Toni	Just Be A Man About ...	2000-07-29	76	69	51	42
Braxton, Toni	Spanish Guitar	2000-12-02	98	98	98	NA
Brock, Chad	A Country Boy Can Su...	2000-01-01	93	75	92	NA
Brock, Chad	Yes!	2000-04-08	90	77	66	61
Brooks & Dunn	You'll Always Be Lov...	2000-06-10	95	85	85	85
Brooks, Garth	Do What You Gotta Do	2000-02-19	86	81	72	70
Byrd, Tracy	Put Your Hand In Min...	2000-01-29	81	77	76	76
Cagle, Chris	My Love Goes On And ...	2000-10-21	99	94	94	87
Cam'ron	What Means The World...	2000-10-14	94	94	96	91
Carey, Mariah	Crybaby	2000-06-24	28	34	48	62
Carey, Mariah	Thank God I Found Yo...	1999-12-11	82	68	50	50

```
billboard %>%
  pivot_longer(c('wk1', 'wk2'), names_to = 'week', values_to = 'rank')
```

```
## # A tibble: 634 x 79
## artist track date.ent~1 wk3 wk4 wk5 wk6 wk7 wk8 wk9 wk10 wk11
## <chr> <chr> <date> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 2 Pac Baby~ 2000-02-26 72 77 87 94 99 NA NA NA NA
## 2 2 Pac Baby~ 2000-02-26 72 77 87 94 99 NA NA NA NA
## 3 2Ge+h~ The ~ 2000-09-02 92 NA NA NA NA NA NA NA NA
## 4 2Ge+h~ The ~ 2000-09-02 92 NA NA NA NA NA NA NA NA
## 5 3 Doo~ Kryp~ 2000-04-08 68 67 66 57 54 53 51 51 51
## 6 3 Doo~ Kryp~ 2000-04-08 68 67 66 57 54 53 51 51 51
## 7 3 Doo~ Loser 2000-10-21 72 69 67 65 55 59 62 61 61
## 8 3 Doo~ Loser 2000-10-21 72 69 67 65 55 59 62 61 61
## 9 504 B~ Wobb~ 2000-04-15 25 17 17 31 36 49 53 57 64
## 10 504 B~ Wobb~ 2000-04-15 25 17 17 31 36 49 53 57 64
## # ... with 624 more rows, 67 more variables: wk12 <dbl>, wk13 <dbl>,
## # wk14 <dbl>, wk15 <dbl>, wk16 <dbl>, wk17 <dbl>, wk18 <dbl>, wk19 <dbl>,
## # wk20 <dbl>, wk21 <dbl>, wk22 <dbl>, wk23 <dbl>, wk24 <dbl>, wk25 <dbl>,
## # wk26 <dbl>, wk27 <dbl>, wk28 <dbl>, wk29 <dbl>, wk30 <dbl>, wk31 <dbl>,
## # wk32 <dbl>, wk33 <dbl>, wk34 <dbl>, wk35 <dbl>, wk36 <dbl>, wk37 <dbl>,
## # wk38 <dbl>, wk39 <dbl>, wk40 <dbl>, wk41 <dbl>, wk42 <dbl>, wk43 <dbl>,
## # wk44 <dbl>, wk45 <dbl>, wk46 <dbl>, wk47 <dbl>, wk48 <dbl>, wk49 <dbl>,
...
```

```
billboard %>%
  pivot_longer(starts_with('wk'), names_to = 'week', values_to = 'rank',
               values_drop_na = TRUE, names_prefix = 'wk')
```

```
## # A tibble: 5,307 x 5
##    artist  track                   date.entered week   rank
##    <chr>   <chr>                   <date>       <chr> <dbl>
##  1 2 Pac   Baby Don't Cry (Keep... 2000-02-26   1        87
##  2 2 Pac   Baby Don't Cry (Keep... 2000-02-26   2        82
##  3 2 Pac   Baby Don't Cry (Keep... 2000-02-26   3        72
##  4 2 Pac   Baby Don't Cry (Keep... 2000-02-26   4        77
##  5 2 Pac   Baby Don't Cry (Keep... 2000-02-26   5        87
##  6 2 Pac   Baby Don't Cry (Keep... 2000-02-26   6        94
##  7 2 Pac   Baby Don't Cry (Keep... 2000-02-26   7        99
##  8 2Ge+her The Hardest Part Of ... 2000-09-02   1        91
##  9 2Ge+her The Hardest Part Of ... 2000-09-02   2        87
## 10 2Ge+her The Hardest Part Of ... 2000-09-02   3        92
## # ... with 5,297 more rows
```

Pivot Wider

(opposite of pivot_longer)

Ex 1.

```
knitr::kable(table2)
```

country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

```
table2 %>%
  pivot_wider(names_from = type, values_from = count)
```

```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>       <int>  <int>      <int>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3 Brazil       1999  37737  172006362
## 4 Brazil       2000  80488  174504898
## 5 China        1999 212258 1272915272
## 6 China        2000 213766 1280428583
```

Ex 2.

```
knitr::kable(us_rent_income)
```

GEOID	NAME	variable	estimate	moe
01	Alabama	income	24476	136
01	Alabama	rent	747	3
02	Alaska	income	32940	508
02	Alaska	rent	1200	13
04	Arizona	income	27517	148
04	Arizona	rent	972	4
05	Arkansas	income	23789	165
05	Arkansas	rent	709	5
06	California	income	29454	109
06	California	rent	1358	3
08	Colorado	income	32401	109
08	Colorado	rent	1125	5
09	Connecticut	income	35326	195
09	Connecticut	rent	1123	5
10	Delaware	income	31560	247
10	Delaware	rent	1076	10
11	District of Columbia	income	43198	681
11	District of Columbia	rent	1424	17
12	Florida	income	25952	70
12	Florida	rent	1077	3
13	Georgia	income	27024	106
13	Georgia	rent	927	3
15	Hawaii	income	32453	218
15	Hawaii	rent	1507	18
16	Idaho	income	25298	208
16	Idaho	rent	792	7
17	Illinois	income	30684	83
17	Illinois	rent	952	3
18	Indiana	income	27247	117
18	Indiana	rent	782	3
19	Iowa	income	30002	143
19	Iowa	rent	740	4
20	Kansas	income	29126	208
20	Kansas	rent	801	5
21	Kentucky	income	24702	159
21	Kentucky	rent	713	4
22	Louisiana	income	25086	155
22	Louisiana	rent	825	4
23	Maine	income	26841	187
23	Maine	rent	808	7
24	Maryland	income	37147	152
24	Maryland	rent	1311	5
25	Massachusetts	income	34498	199
25	Massachusetts	rent	1173	5
26	Michigan	income	26987	82
26	Michigan	rent	824	3
27	Minnesota	income	32734	189
27	Minnesota	rent	906	4
28	Mississippi	income	22766	194
28	Mississippi	rent	740	5
29	Missouri	income	26999	113
29	Missouri	rent	784	4
30	Montana	income	26249	206

```
us_rent_income %>%
  pivot_wider(names_from = variable, values_from = c(estimate, moe))
```

```
## # A tibble: 52 x 6
## GEOID NAME estimate_income estimate_rent moe_income moe_rent
## <chr> <chr> <dbl> <dbl> <dbl> <dbl>
## 1 01 Alabama 24476 747 136 3
## 2 02 Alaska 32940 1200 508 13
## 3 04 Arizona 27517 972 148 4
## 4 05 Arkansas 23789 709 165 5
## 5 06 California 29454 1358 109 3
## 6 08 Colorado 32401 1125 109 5
## 7 09 Connecticut 35326 1123 195 5
## 8 10 Delaware 31560 1076 247 10
## 9 11 District of Columbia 43198 1424 681 17
## 10 12 Florida 25952 1077 70 3
## # ... with 42 more rows
```

```
us_rent_income %>%
  pivot_wider(names_from = variable, names_sep = '.',
              values_from = c(estimate, moe))
```

```
## # A tibble: 52 x 6
## GEOID NAME estimate.income estimate.rent moe.income moe.rent
## <chr> <chr> <dbl> <dbl> <dbl> <dbl>
## 1 01 Alabama 24476 747 136 3
## 2 02 Alaska 32940 1200 508 13
## 3 04 Arizona 27517 972 148 4
## 4 05 Arkansas 23789 709 165 5
## 5 06 California 29454 1358 109 3
## 6 08 Colorado 32401 1125 109 5
## 7 09 Connecticut 35326 1123 195 5
## 8 10 Delaware 31560 1076 247 10
## 9 11 District of Columbia 43198 1424 681 17
## 10 12 Florida 25952 1077 70 3
## # ... with 42 more rows
```

Separate

Ex 1.

```
knitr::kable(table3)
```

country	year	rate
Afghanistan	1999	745/19987071
Afghanistan	2000	2666/20595360
Brazil	1999	37737/172006362
Brazil	2000	80488/174504898
China	1999	212258/1272915272
China	2000	213766/1280428583

```
table3 %>%
  separate(rate, into = c('cases', 'population'), sep = '/')
```

```
## # A tibble: 6 x 4
##   country      year cases  population
##   <chr>       <int> <chr>  <chr>
## 1 Afghanistan  1999 745    19987071
## 2 Afghanistan  2000 2666   20595360
## 3 Brazil       1999 37737  172006362
## 4 Brazil       2000 80488  174504898
## 5 China        1999 212258 1272915272
## 6 China        2000 213766 1280428583
```

```
table3 %>%
  separate(rate, into = c('cases', 'population'),
           sep = '/', convert = TRUE)
```

```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>       <int>  <int>      <int>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3 Brazil       1999  37737  172006362
## 4 Brazil       2000  80488  174504898
## 5 China        1999 212258 1272915272
## 6 China        2000 213766 1280428583
```

```
table3 %>%
  separate(year, into = c('century', 'year'), sep = 2)
```

```
## # A tibble: 6 x 4
##   country     century year  rate
##   <chr>       <chr>   <chr> <chr>
## 1 Afghanistan 19      99    745/19987071
## 2 Afghanistan 20      00    2666/20595360
## 3 Brazil      19      99    37737/172006362
```

```
## 4 Brazil      20      00    80488/174504898
## 5 China       19      99    212258/1272915272
## 6 China       20      00    213766/1280428583
```

Ex 2.

```
df <- data.frame(x = c('a', 'a b', 'a b c', NA))
df
```

```
##       x
## 1     a
## 2   a b
## 3 a b c
## 4  <NA>
```

```
df %>% separate(x, into = c('a', 'b'), extra = 'drop', fill = 'right')
```

```
##      a    b
## 1    a <NA>
## 2    a    b
## 3    a    b
## 4 <NA> <NA>
```

```
df %>% separate(x, into = c('a', 'b'), extra = 'merge', fill = 'right')
```

```
##      a    b
## 1    a <NA>
## 2    a    b
## 3    a  b c
## 4 <NA> <NA>
```

```
df %>% separate(x, into = c('a', 'b', 'c'), fill = 'right')
```

```
##      a    b    c
## 1    a <NA> <NA>
## 2    a    b <NA>
## 3    a    b    c
## 4 <NA> <NA> <NA>
```

Unite

(opposite of separate)

```r
df <- expand_grid(x = c('a', NA), y = c('b', NA))
df
```

```
## # A tibble: 4 x 2
##   x     y
##   <chr> <chr>
## 1 a     b
## 2 a     <NA>
## 3 <NA>  b
## 4 <NA>  <NA>
```

```r
df %>% unite('z', x:y, remove = FALSE)
```

```
## # A tibble: 4 x 3
##   z     x     y
##   <chr> <chr> <chr>
## 1 a_b   a     b
## 2 a_NA  a     <NA>
## 3 NA_b  <NA>  b
## 4 NA_NA <NA>  <NA>
```

```r
# Removes NA from the original string in col z
df %>% unite('z', x:y, na.rm = TRUE, remove = FALSE)
```

```
## # A tibble: 4 x 3
##   z     x     y
##   <chr> <chr> <chr>
## 1 "a_b" a     b
## 2 "a"   a     <NA>
## 3 "b"   <NA>  b
## 4 ""    <NA>  <NA>
```

All together

```r
df
```

```
## # A tibble: 4 x 2
##   x     y
##   <chr> <chr>
## 1 a     b
## 2 a     <NA>
## 3 <NA>  b
## 4 <NA>  <NA>
```

-

```
df %>%
  unite('xy', x:y)
```

```
## # A tibble: 4 x 1
##   xy
##   <chr>
## 1 a_b
## 2 a_NA
## 3 NA_b
## 4 NA_NA
```

=

```
df %>%
  unite('xy', x:y) %>%
  separate(xy, into = c('x', 'y'))
```

```
## # A tibble: 4 x 2
##   x     y
##   <chr> <chr>
## 1 a     b
## 2 a     NA
## 3 NA    b
## 4 NA    NA
```

Missing Values

```
stocks <- tibble(
  year   = c(2015, 2015, 2015, 2015, 2016, 2016, 2016),
  qtr    = c(   1,    2,    3,    4,    2,    3,    4),
  return = c(1.88, 0.59, 0.35,   NA, 0.92, 0.17, 2.66)
)
stocks
```

```
## # A tibble: 7 x 3
##    year   qtr return
##   <dbl> <dbl>  <dbl>
## 1  2015     1   1.88
## 2  2015     2   0.59
## 3  2015     3   0.35
## 4  2015     4  NA
```

```
## 5  2016     2   0.92
## 6  2016     3   0.17
## 7  2016     4   2.66
```

The completes the 'qtr' column which should show quarter 1, 2, 3, and 4

```
stocks %>%
  complete(year, qtr)
```

```
## # A tibble: 8 x 3
##    year   qtr return
##   <dbl> <dbl>  <dbl>
## 1  2015     1   1.88
## 2  2015     2   0.59
## 3  2015     3   0.35
## 4  2015     4  NA
## 5  2016     1  NA
## 6  2016     2   0.92
## 7  2016     3   0.17
## 8  2016     4   2.66
```

Important code: This fill's in the NA rows by person

```
treatment <- tibble(
  person = c('Derrick Whitmore', NA, NA, 'Katherine Burke'),
  treatment = c(1, 2, 3, 1),
  response = c(7, 10, 9, 4)
)
treatment
```

```
## # A tibble: 4 x 3
##   person           treatment response
##   <chr>                <dbl>    <dbl>
## 1 Derrick Whitmore         1        7
## 2 <NA>                     2       10
## 3 <NA>                     3        9
## 4 Katherine Burke          1        4
```

```
treatment %>%
  fill(person)
```

```
## # A tibble: 4 x 3
##   person           treatment response
```

```
##   <chr>                 <dbl>    <dbl>
## 1 Derrick Whitmore         1        7
## 2 Derrick Whitmore         2       10
## 3 Derrick Whitmore         3        9
## 4 Katherine Burke          1        4
```

2.3 Dplyr

- filter(): Pick observations by their values
- arrange(): Reorder the rows
- select(): Pick variables by their names
- mutate(): Create new variables with functions of existing variables
- summarise(): Collapse many values down to a single summary

filter()

filter() allows you to subset observations based on their values. The first argument is the name of the data frame. The second and subsequent arguments are the expressions that filter the data frame.

Examples:

```
# Selecting all flights on January 1st
jan1 <- filter(flights, month == 1, day == 1)

# Find all flights that departed in November and December
nov_dec <- filter(flights, month %in% c(11, 12))

# Find flights that weren't delayed (on arrival or departure) by more than two hours
filter(flights, !(arr_delay > 120 | dep_delay > 120))
filter(flights, arr_delay <= 120, dep_delay <= 120)
```

arrange()

arrange() works similarly to filter() except that instead of selecting rows, it changes their order. It takes a data frame and a set of column names (or more complicated expressions) to order by. If you provide more than one column name, each additional column will be used to break ties in the values of preceding columns.

```
arrange(flights, year, month, day)
```

Use desc() to re-order by a column in descending order:

```
arrange(flights, desc(dep_delay))
```

select()

select() allows you to rapidly zoom in on a useful subset using operations based on the names of the variables.

```
# Select columns by name
select(flights, year, month, day)

# Select all columns between year and day (inclusive)
select(flights, year:day)

# Select all columns except those from year to day (inclusive)
select(flights, -(year:day))
```

Functions to use within **select()**:

```
**starts_with("abc")**: matches names that begin with "abc".

**ends_with("xyz")**: matches names that end with "xyz".

**contains("ijk")**: matches names that contain "ijk".

**matches("(.)\\1")**: selects variables that match a regular expression. This one matches any va

**num_range("x", 1:3)**: matches x1, x2 and x3.

**everything()**: This is useful if you have a handful of variables you'd like to move to the sta
```

```
flights_sml <- select(flights,
  year:day,
  ends_with("delay"),
  distance,
  air_time
)
```

```
select(flights, time_hour, air_time, everything())
```

mutate()

mutate() always adds new columns at the end of your dataset

```
# Creates GAIN and SPEED variables in the flights_sml dataframe
mutate(flights_sml,
  gain = dep_delay - arr_delay,
  speed = distance / air_time * 60
)
```

summarise()

This collapses a data frame to a single row

```
summarise(flights, delay = mean(dep_delay, na.rm = TRUE))
```

summarise() is not terribly useful unless we pair it with group_by()

```
by_day <- group_by(flights, year, month, day)
summarise(by_day, delay = mean(dep_delay, na.rm = TRUE))
```

Grouping by multiple variables and Grouped Mutates (and filters)

```
# Grouping by multiple variables
daily <- group_by(flights, year, month, day)
per_day <- summarise(daily, flights = n())
per_month <- summarise(per_day, flights = sum(flights))
per_year  <- summarise(per_month, flights = sum(flights))

# Find the worst members of each group
flights_sml %>%
  group_by(year, month, day) %>%
  filter(rank(desc(arr_delay)) < 10)

# Find all groups bigger than a threshold:
popular_dests <- flights %>%
  group_by(dest) %>%
  filter(n() > 365)

# Standardise to compute per group metrics:
popular_dests %>%
  filter(arr_delay > 0) %>%
  mutate(prop_delay = arr_delay / sum(arr_delay)) %>%
  select(year:day, dest, arr_delay, prop_delay)
```

Using pipe to combine operations: %>%

A good way to think of %>% is reading it as "then".

Groups flights by destination. Summarise to compute distance, average delay, and number of flights [count=n()].Filter to remove noisy points and Honolulu (“HNL”) airport.

```
delays <- flights %>%
  group_by(dest) %>%
  summarise(
    count = n(),
    dist = mean(distance, na.rm = TRUE),
    delay = mean(arr_delay, na.rm = TRUE)
  ) %>%
  filter(count > 20, dest != "HNL")
```

Other Example:

```
flights %>%
  group_by(year, month, day) %>%
  summarise(mean = mean(dep_delay, na.rm = TRUE))

not_cancelled <- flights %>%
  filter(!is.na(dep_delay), !is.na(arr_delay))

not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(mean = mean(dep_delay))

# Using TIDYVERSE with GGPLOT2
delays <- not_cancelled %>%
  group_by(tailnum) %>%
  summarise(
    delay = mean(arr_delay, na.rm = TRUE),
    n = n()
  )
ggplot(data = delays, mapping = aes(x = n, y = delay)) +
  geom_point(alpha = 1/10)

delays %>%
  filter(n > 25) %>%
  ggplot(mapping = aes(x = n, y = delay)) +
    geom_point(alpha = 1/10)
```

Useful summary functions:

Measures of location: we’ve used mean(x), but median(x)

Measures of spread: sd(x), IQR(x), mad(x) -median absolute deviation

Measures of rank: min(x), quantile(x, 0.25), max(x)

Measures of position: first(x), nth(x, 2), last(x).

Counts and proportions of logical values: sum(x > 10), mean(y == 0).

2.4 Ggplot2

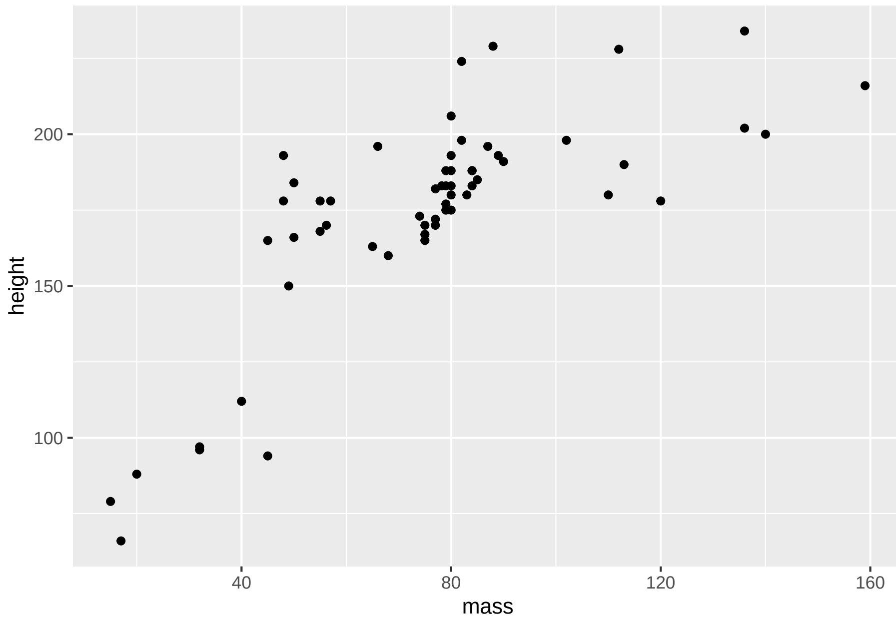
- Simple graph
- … with color aesthetic
- Wrap Faceting
- Grid Faceting
- Smooth Geom
- Bar Geom/Graph
- Histogram Geom
- Modifications

```
library(ggplot2)
library(dplyr)
library(knitr)
library(kableExtra)
```

Simple Graphic

Height by Mass

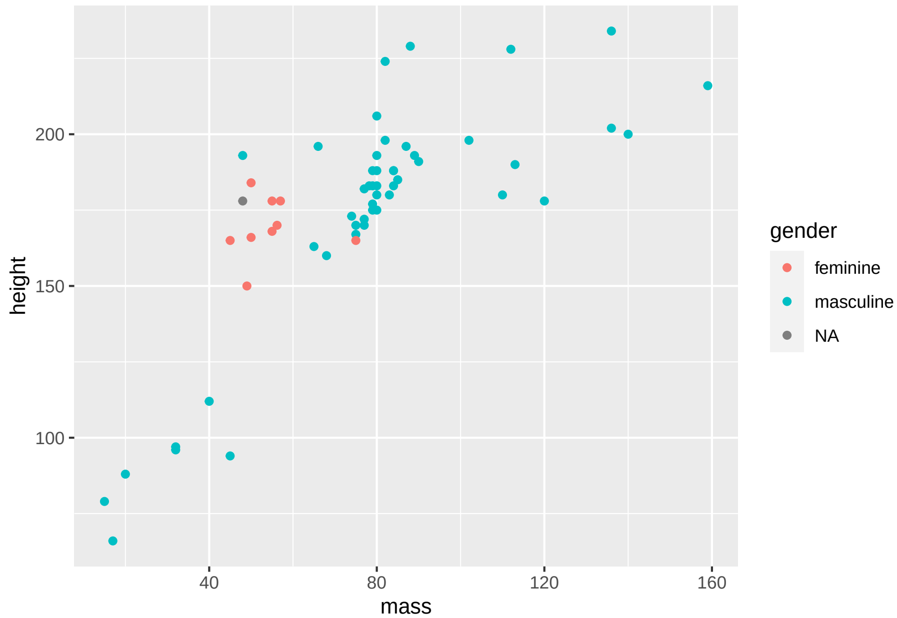
```
starwars_filtered <- starwars %>%
  filter(mass < 500)

ggplot(starwars_filtered) +
  geom_point(aes(x = mass, y = height))
```



Scatterplot with color aesthetic

Height by Mass, pointing out gender

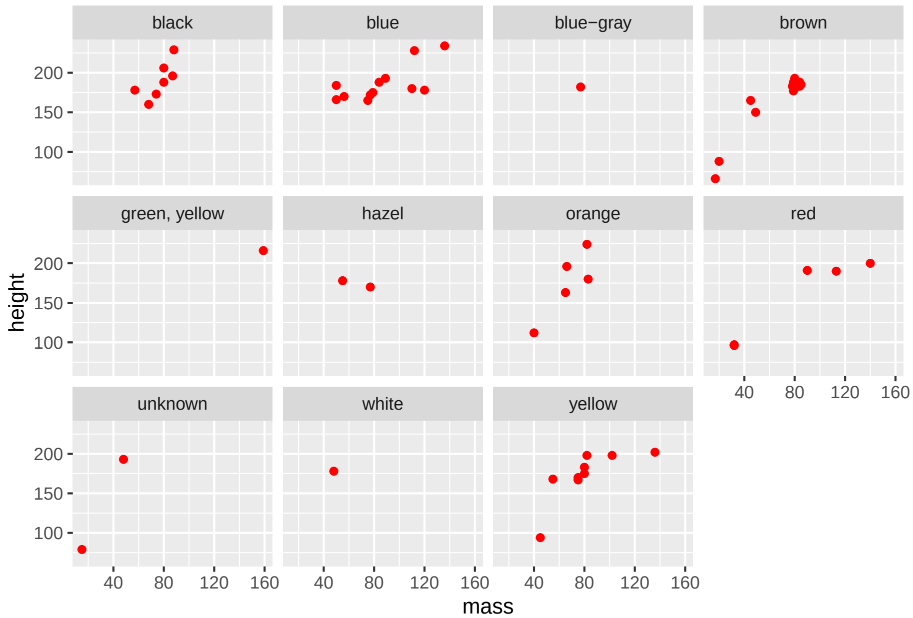
```
ggplot(starwars_filtered, aes(mass, height)) +
  geom_point(aes(color = gender))
```



Wrap Faceting

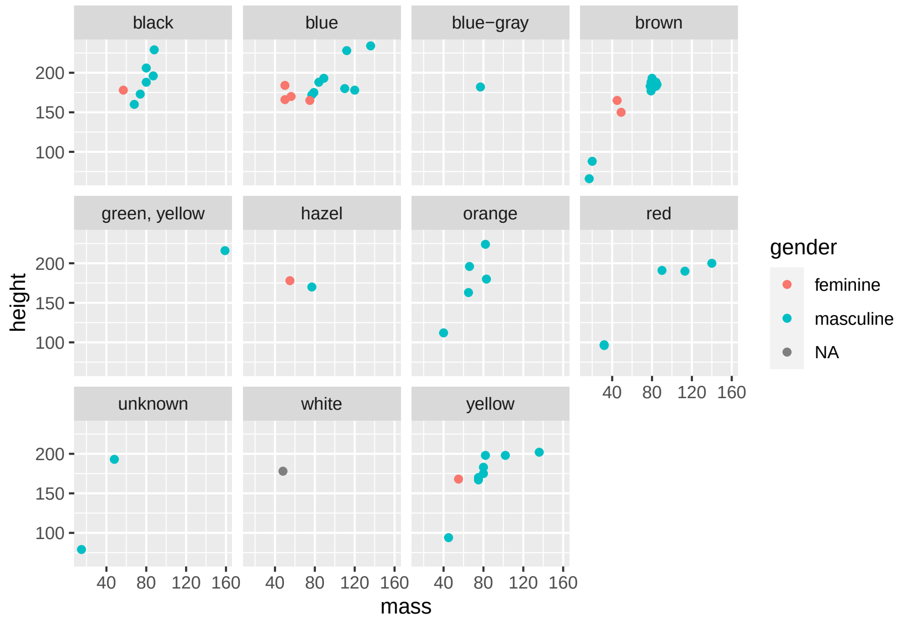
Height by Mass wrapped with eye color

```
# Basic
ggplot(starwars_filtered, aes(mass, height)) +
  geom_point(color = 'red') +
  facet_wrap(~ eye_color)
```



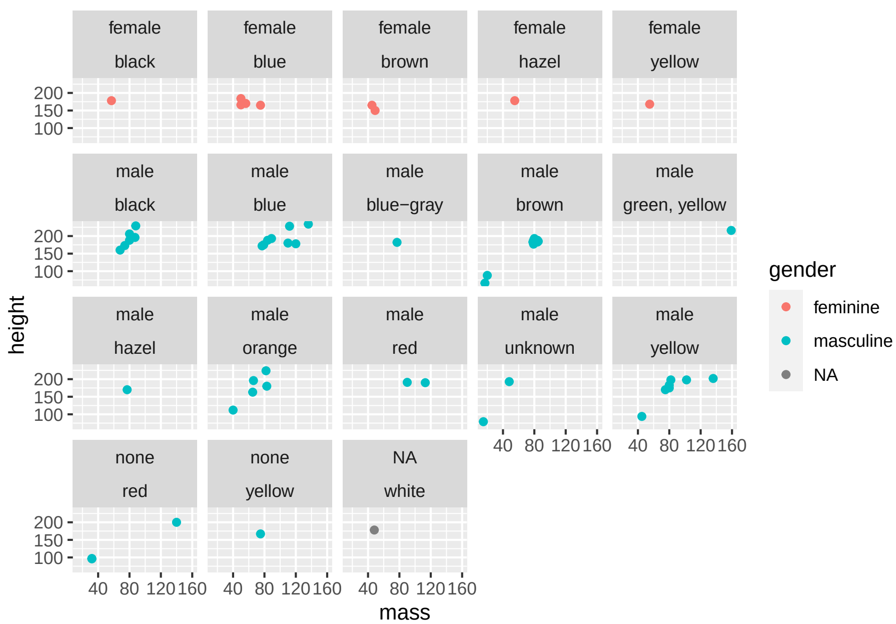
Height by Mass, pointing out gender, wrapped with eye color

```
# with color aes
ggplot(starwars_filtered, aes(mass, height)) +
  geom_point(aes(color = gender)) +
  facet_wrap(~ eye_color)
```



Height by Mass, pointing out gender, wrapped with sex and eye color

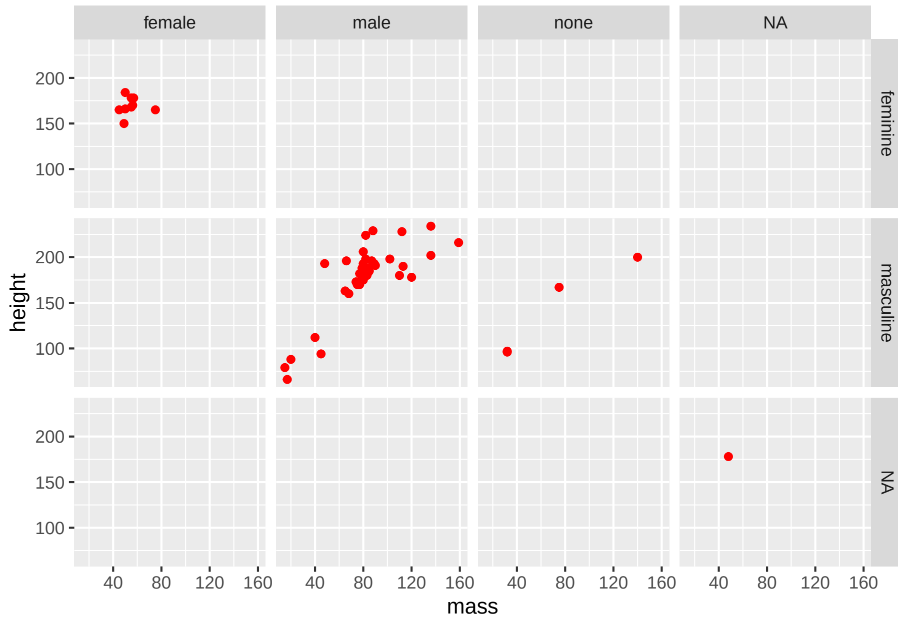
```
# broken down by sex ~ eye_color
ggplot(starwars_filtered, aes(mass, height)) +
  geom_point(aes(color = gender)) +
  facet_wrap(sex ~ eye_color)
```



Grid Faceting

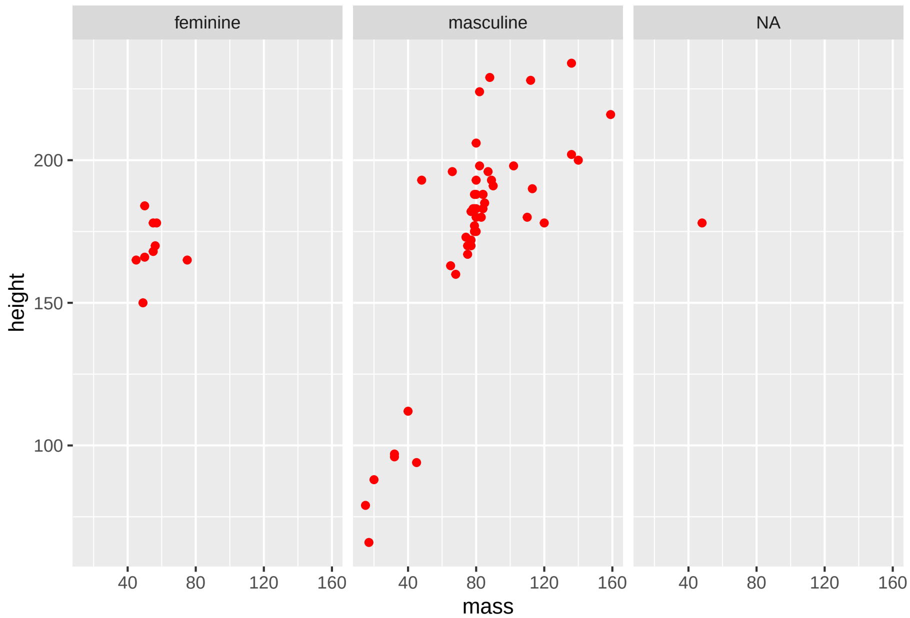
Height by Mass, grid with gender and sex

```
ggplot(starwars_filtered, aes(mass, height)) +
  geom_point(color = 'red') +
  facet_grid(gender ~ sex)
```



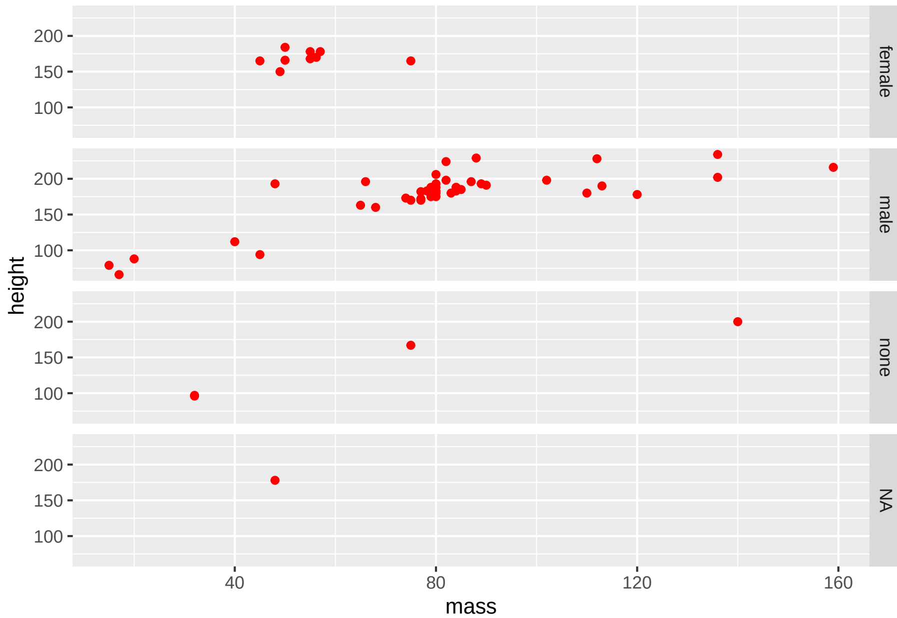
Height by Mass, grid with everything by gender

```
ggplot(starwars_filtered, aes(mass, height)) +
  geom_point(color = 'red') +
  facet_grid(. ~ gender)
```



Height by Mass, grid with sex by everything

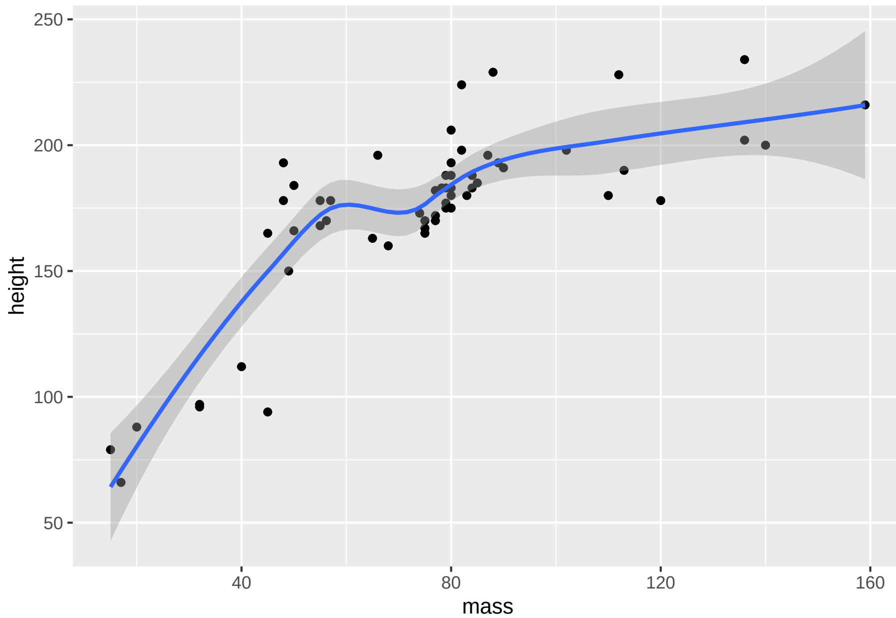
```
ggplot(starwars_filtered, aes(mass, height)) +
  geom_point(color = 'red') +
  facet_grid(sex ~ .)
```



Smooth Geom

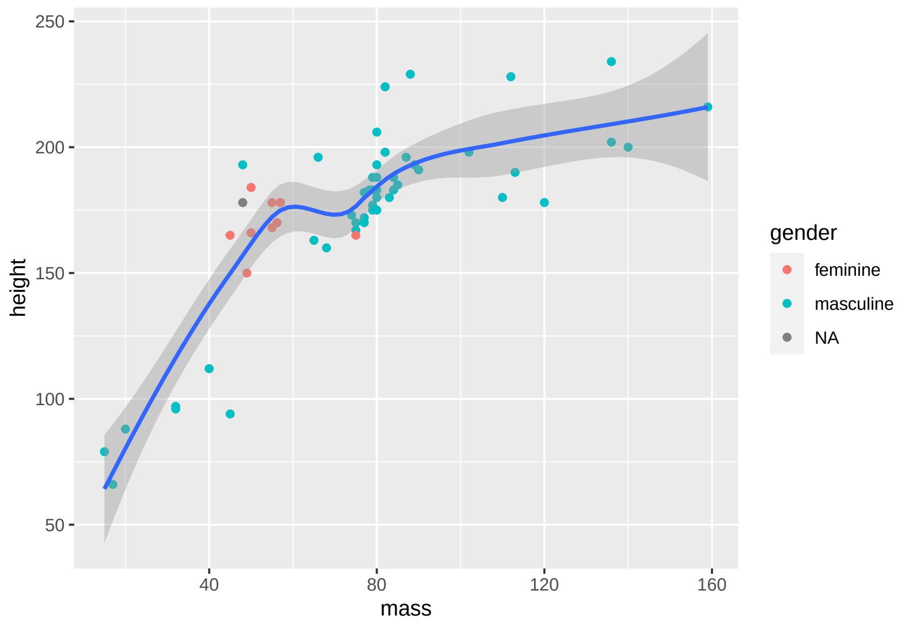
Graphs line with grey area width associated with disperse data

```
ggplot(starwars_filtered, aes(mass, height)) +
  geom_point() +
  geom_smooth()
```



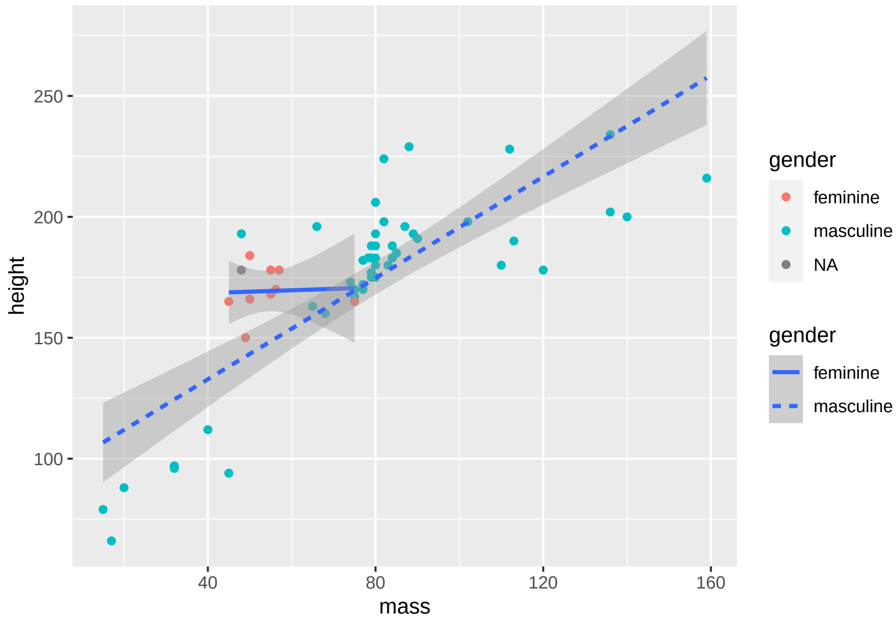
Height by Mass, color coordinated with gender

```
ggplot(starwars_filtered, aes(mass, height)) +
  geom_point(aes(color = gender)) +
  geom_smooth()
```



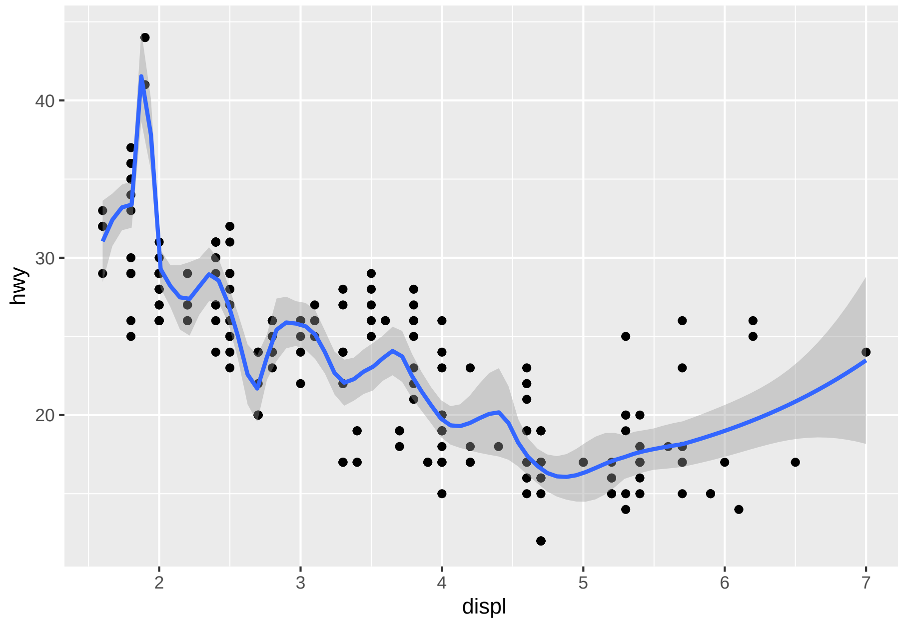
Graphs with a linear regression line (doing this without method = “lm” creates a weird line)

```
ggplot(starwars_filtered, aes(mass, height)) +
  geom_point(aes(color = gender)) +
  geom_smooth(aes(linetype = gender), method = 'lm')
```

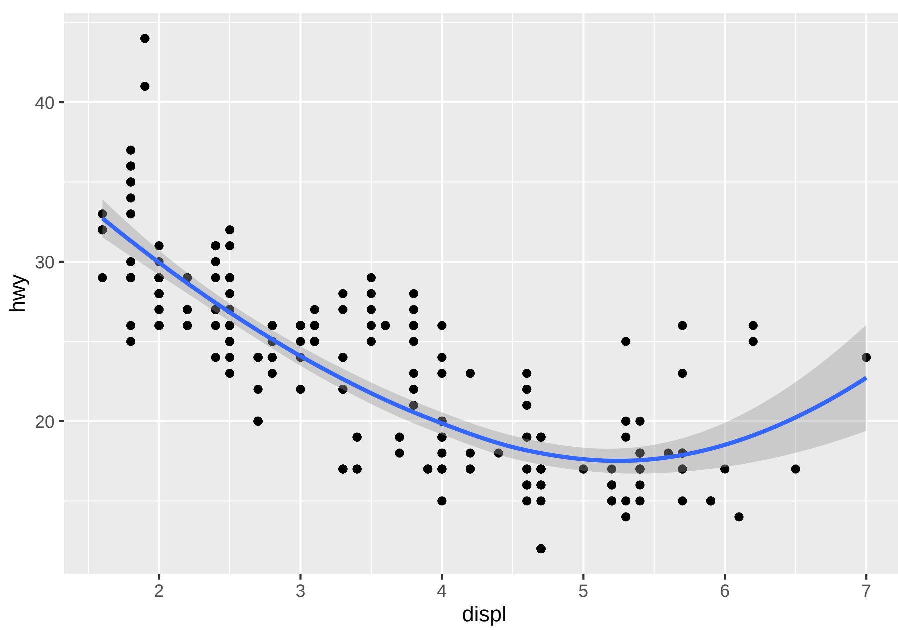


span = controls the amount of smoothing. Larger = smoother lines

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



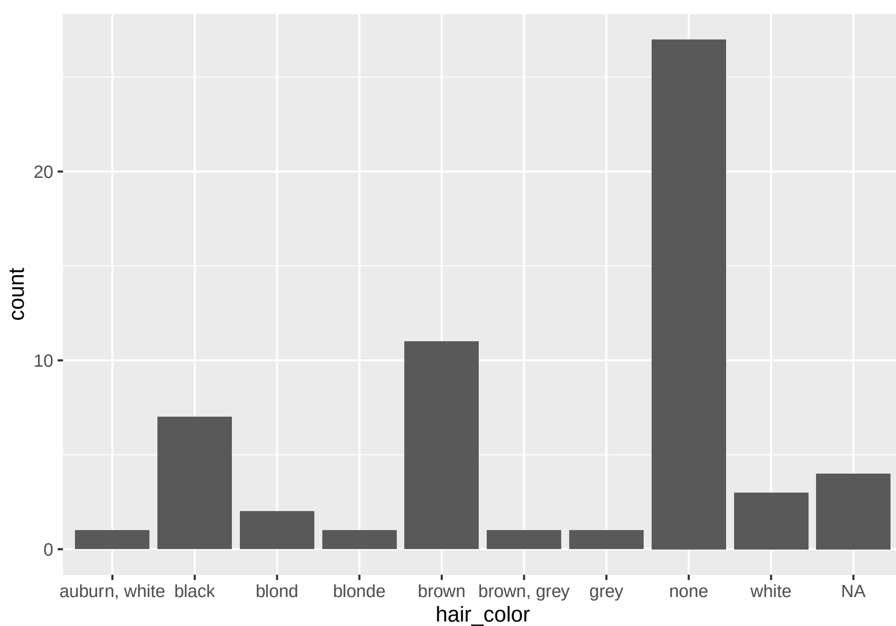

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_smooth(span = 1)
```



Bar Graph

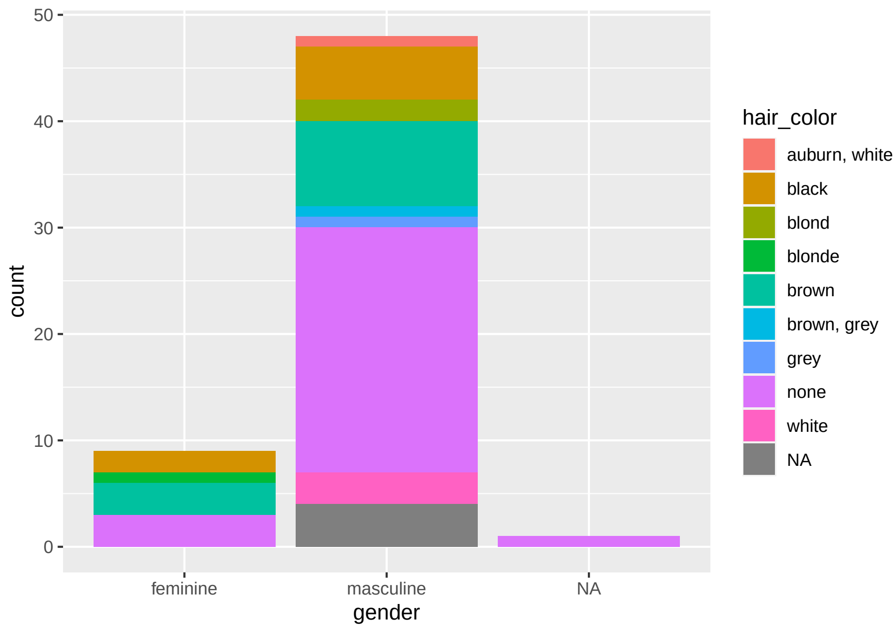
Bar graph of hair color

```
ggplot(starwars_filtered) +
  geom_bar(aes(x = hair_color))
```



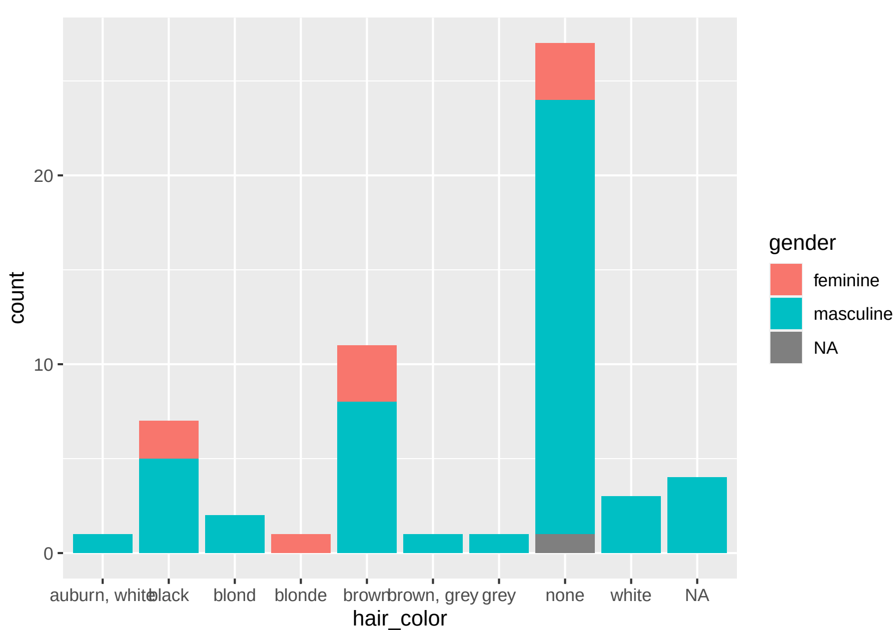
Bar graph of gender which is broken down by hair color

```
ggplot(starwars_filtered) +
  geom_bar(aes(x = gender, fill = hair_color))
```



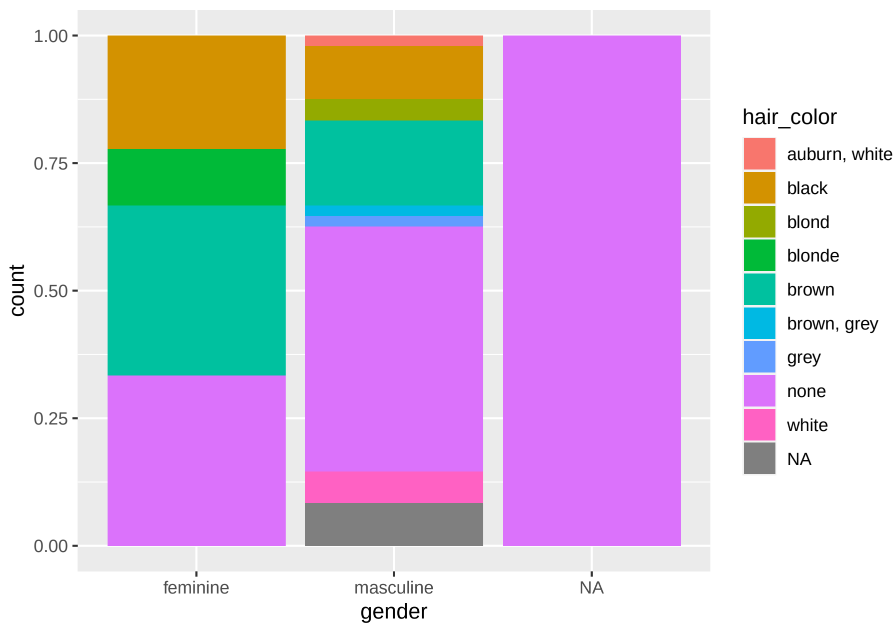
Bar graph of hair color which is broken down by gender

```
ggplot(starwars_filtered) +
  geom_bar(aes(x = hair_color, fill = gender))
```



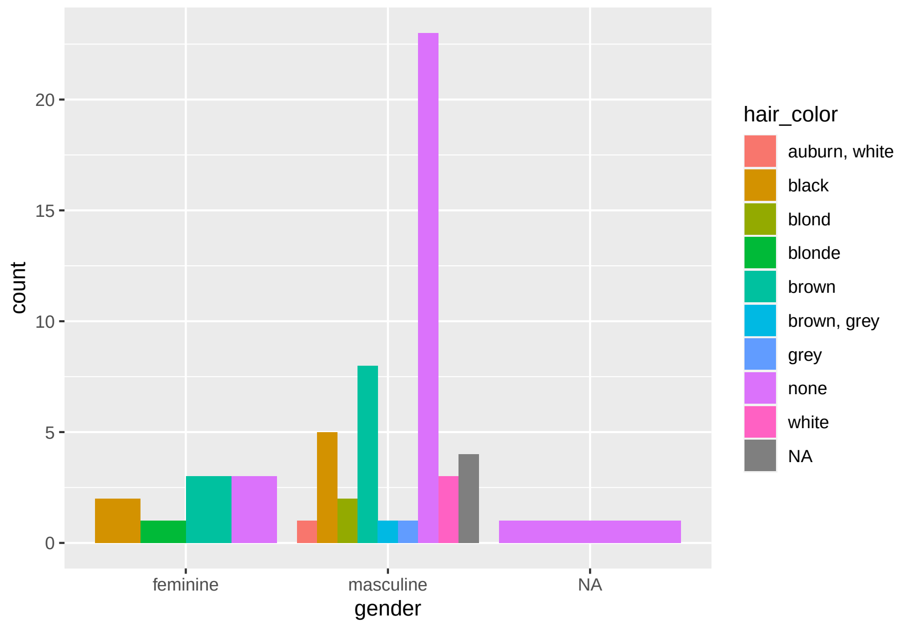
position = ‘fill’ shows what percentage of gender make up a certain hair color

```
ggplot(starwars_filtered, aes(x = gender, fill = hair_color)) +
  geom_bar(position = 'fill')
```



position = ‘dodge’ shows the breakdown of certain hair color for gender

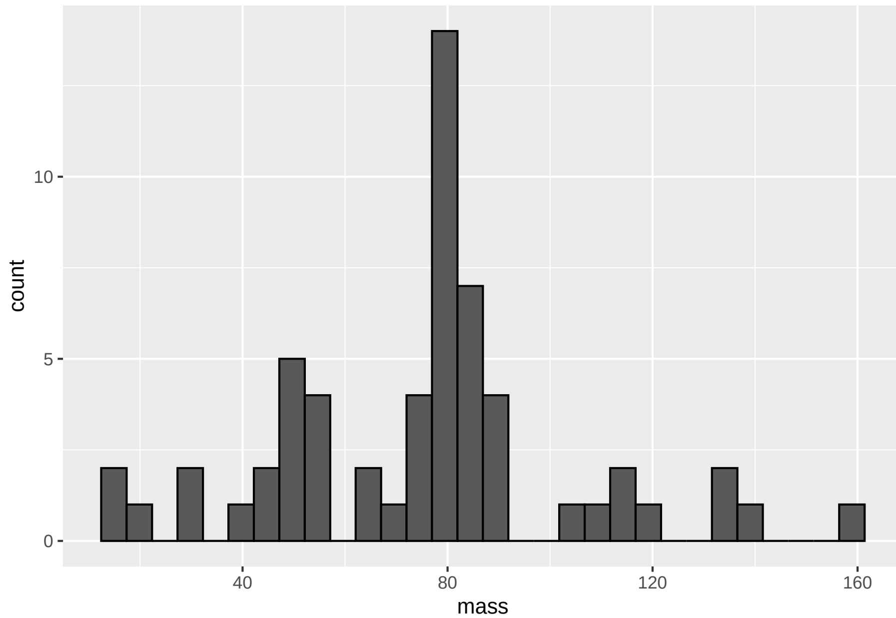
```
ggplot(starwars_filtered, aes(x = gender, fill = hair_color)) +
  geom_bar(position = 'dodge')
```



Histogram Geom

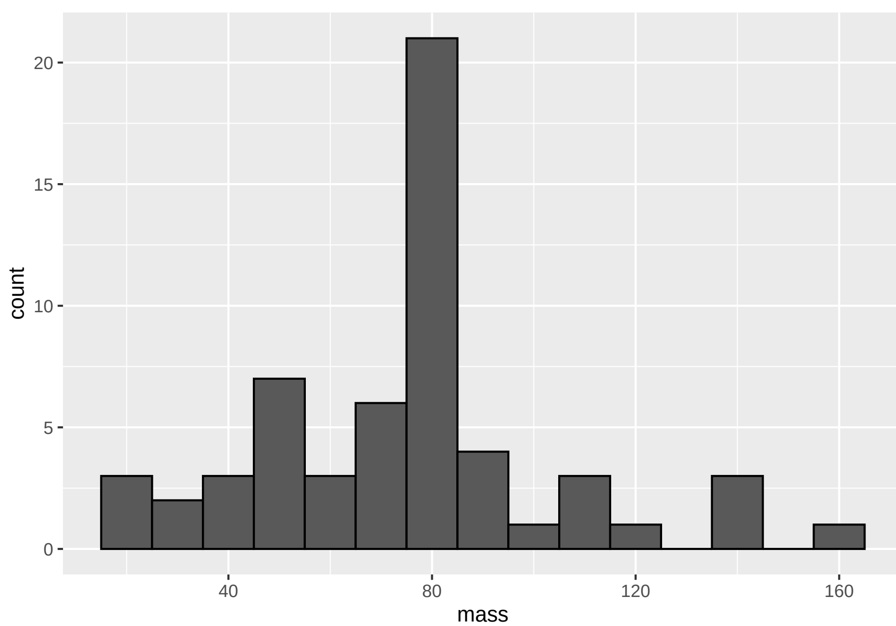
Basic Histogram

```
ggplot(starwars_filtered) +
  geom_histogram(aes(x = mass), color = 'black')
```



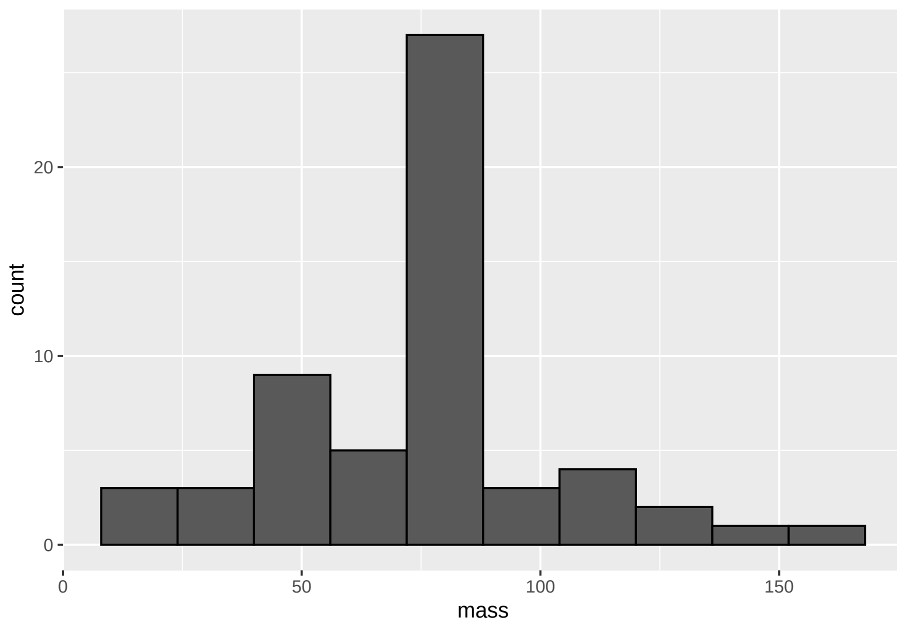
Changing the binwidth

```
ggplot(starwars_filtered) +
  geom_histogram(aes(x = mass), color = 'black', binwidth = 10)
```



Changing the amount of bins to 10 'fills' in the spaces

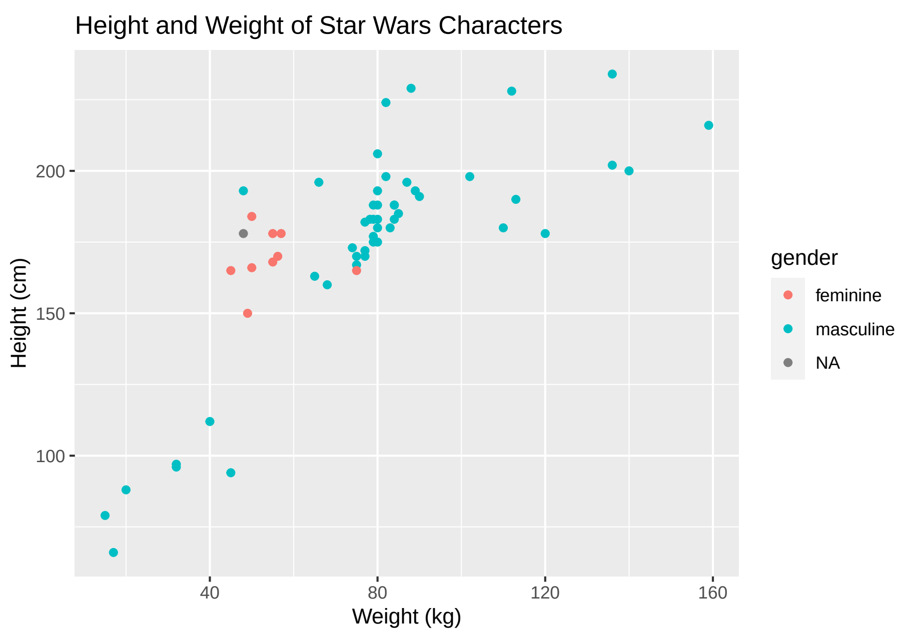
```
ggplot(starwars_filtered) +
  geom_histogram(aes(x = mass), color = 'black', bins = 10)
```



Modifications

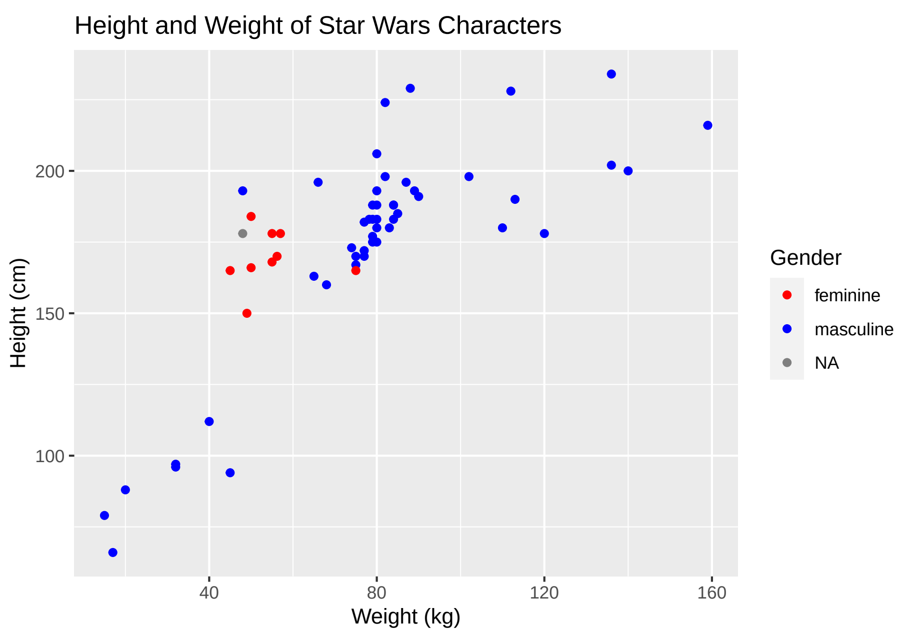
Labeling the axis and title

```
ggplot(starwars_filtered, aes(mass, height, color = gender)) +
  geom_point() +
  xlab('Weight (kg)') +
  ylab('Height (cm)') +
  ggtitle('Height and Weight of Star Wars Characters')
```



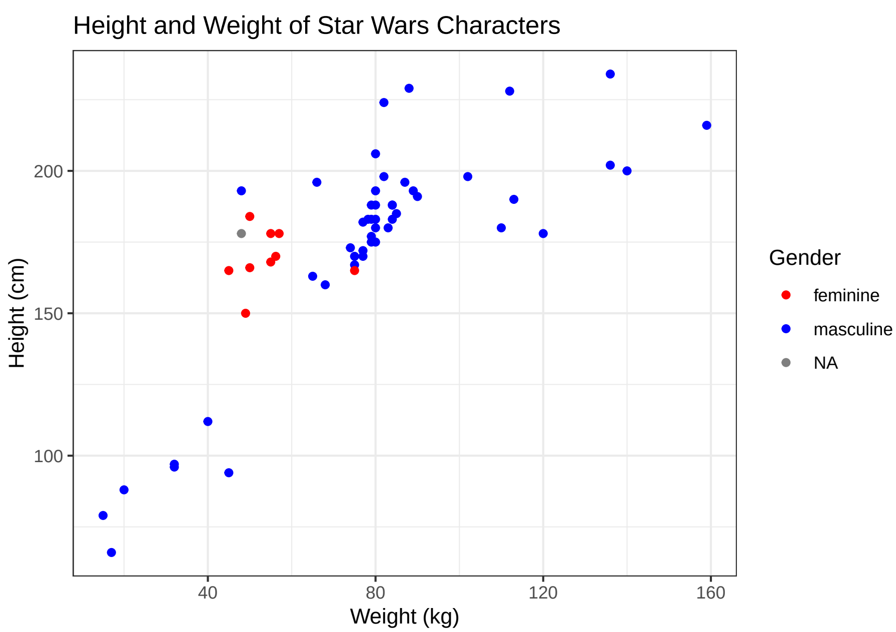
Scale the color of the points

```
ggplot(starwars_filtered, aes(mass, height, color = gender)) +
  geom_point() +
  labs(x = 'Weight (kg)', y = 'Height (cm)',
       title = 'Height and Weight of Star Wars Characters') +
  scale_color_discrete(name = 'Gender', type = c('red', 'blue', 'black'))
```



theme_bw()

```
ggplot(starwars_filtered, aes(mass, height, color = gender)) +
  geom_point() +
  labs(x = 'Weight (kg)', y = 'Height (cm)',
       title = 'Height and Weight of Star Wars Characters') +
  scale_color_discrete(name = 'Gender', type = c('red', 'blue', 'black')) +
  theme_bw()
```



2.5 Functions

- Basic coding
- Using default arguments
- Case Study

2.5.1 Basic Coding style

```
add_two_values <- function(value1, value2){
  the_sum <- value1 + value2
  return(the_sum)
}
add_two_values(5, 10)
```

2.5.2 Having a default value for a argument

```
adder <- function(value1, value2 = 2){
  the_sum <- value1 + value2
  return(the_sum)
```

```
}
adder(10) # 10 + 2 = 12
```

2.5.3 Case Study

Case Study Function Suppose you are considering buying a house. You are interested in seeing an amortization table for a given set of inputs. The table includes, by month, the principal left, the amount of interest paid, the amount of principal paid, the total amount of interest paid, and the total amount paid. You decide to write a function that determines these for a given house price, down payment, monthly payment, and interest rate.

```
amortization_table <- function(price, downPayment, monthlyPayment, interestRate){
  perMonthRate <- interestRate/365*30
  principalLeft <- price - downPayment
  interestPaid <- 0
  principalPaid <- 0
  totalInterestPaid <- 0
  totalAmountPaid <- downPayment
  month <- 0
  amortTable <- data.frame(month = month, principalLeft = principalLeft,
                           interestPaid = interestPaid, principalPaid = principalPaid,
                           totalInterestPaid = totalInterestPaid, totalAmountPaid = to
  while(principalLeft > 0){
    month <- month + 1
    interestPaid <- round(perMonthRate*principalLeft, 2)
    if(interestPaid >= monthlyPayment){
      return('Monthly payment is too low.')
    }
    principalPaid <- monthlyPayment - interestPaid
    principalLeft <- principalLeft - principalPaid
    totalInterestPaid <- interestPaid + totalInterestPaid
    totalAmountPaid <- interestPaid + principalPaid + totalAmountPaid
    amortTable[month + 1,] <- c(month, principalLeft, interestPaid, principalPaid,
                                totalInterestPaid, totalAmountPaid)
  }
  amortTable[month + 1,] <- c(month, 0, interestPaid, amortTable$principalLeft[month],
                              totalInterestPaid, amortTable$totalAmountPaid[month] + i
  return(amortTable)
}

amortization_table(100000, 20000, 1000, .05)
```

2.6 If Statements

if() statements

```
# Ex. 1
a <- 3
mynumber <- 4
if(a <= mynumber){
  a <- a^2
}

# Ex. 2
a <- 5
mynumber <- 4
if(a >= mynumber){
  a <- a - mynumber
}

# Ex. 3
a <- 3
mynumber <- 6
if(a != mynumber){
  a <- a^3 + 3*mynumber
}
```

ifelse statements: short

```
a <- 3
mynumber <- 4
if(a > mynumber){
  a <- a - mynumber
}else{
  a <- a + mynumber
}
```

ifelse statements: longer

```
a <- 3
mynumber <- 4
if(a < mynumber){
  a <- a^2
}else if(a > mynumber){
  a <- 2
}else{
  a <- a*mynumber
}
```

2.7 For() Loops

- Basic for-loops
- Nested for-loops

clean and filter this

2.7.1 Basic for-loops

```
for(i in 1:3){
  3*i + 1
}

for(i in 1:3){
  print(3*i + 1)
}

values <- c(2, 4, 1, -3)
for(i in values){
  print(3*i + 1)
}

values <- c(2, 4, 1, -1)
x <- numeric(length(values))
for(i in 1:length(values)){
  x[i] <- 3*values[i] + 1
}

values <- c(2, 4, 1, -1)
x <- 0
index <- 0
for(i in values){
  index <- index + 1
  x[index] <- 3*i + 1
}

values <- c(2, 4, 1, -1)
x <- NULL
for(i in values){
  x <- c(x, 3*i + 1)
}

system.time({
```

```
  x <- 0
  for(i in 1:1000000){
    x[i] <- 3*i + 1
  }
  x
})

system.time({
  x <- numeric(1000000)
  for(i in 1:length(x)){
    x[i] <- 3*i + 1
  }
  x
})
```

2.7.2 Nested for-loops

```
x <- 1:4
y <- 7:9
out.matrix <- matrix(NA, nrow = length(x), ncol = length(y))
for(i in 1:length(x)){
  for(j in 1:length(y)){
    out.matrix[i, j] <- x[i] + y[j]
  }
}
```

2.8 While Loops

- Basic while-loops
- Nested While Loops
  - Break
  - Next

2.8.1 Basic while-loops

```
value <- 0
while(value < 10){
  value <- 3*value + 1
}
```

```
value <- 0
while(value < 10){
  value <- 3*value + 1
  value
}

value <- 0
while(value < 10){
  value <- 3*value + 1
  print(value)
}

value <- 0
i <- 1
while(value[i] < 10){
  i <- i + 1
  value[i] <- 3*value[i-1] + 1
}

value <- 0
i <- 1
while(value[i] < 10){
  i <- i + 1
  value <- c(value, 3*value[i-1] + 1)
}
```

2.8.2 Nested While Loops

```
# break
value <- 0
i <- 1
maxiters <- 5
while(value[i] < 10000){
  i <- i + 1
  value[i] <- 3*value[i-1] + 1
  if(i == 5){
    break
  }
}
value

value1 <- c(1, 2, 3, 0, 2)
value2 <- c(3, 0, 2, 2, 5)
```

```
result <- rep(NA, length(value1))
for(i in 1:length(value1)){
  temp <- 6*value2[i]/value1[i]
  if(is.finite(temp)){
    result[i] <- temp
  }else{
    break
  }
}

# next
value1 <- c(1, 2, 3, 0, 2)
value2 <- c(3, 0, 2, 2, 5)
result <- rep(NA, length(value1))
for(i in 1:length(value1)){
  temp <- 6*value2[i]/value1[i]
  if(is.finite(temp)){
    result[i] <- temp
  }else{
    next
  }
}
```

Chapter 3

Lab Stuff

3.1 ETC

3.1.1 Visit & QR

3.1.2 Fitbit

3.1.3 Questionnaires

- ESSQ
- HADS
- MMSE

3.1.4 Cognitive

- Sternberg
- Stroop

3.2 EPPL

- Cognitive Codebook

Chapter 4

Courses

4.1 STAT 420

This will consist of the homework assignments

4.2 EPSY 582

4.3 PSYC 594

4.4 EPSY 590

4.5 PSYC 581

Chapter 5

UIUC Projects

This chapter consists of projects I have done as part of my doctoral studies at UIUC such as:

- Final Projects
- Mid-term Projects
- Code for conference abstracts

5.1 Simulation Project (Mid-Term: Stat420)

5.2 EPSY590 Final Baseball Project

5.3 Best Research Practices stroop final project

5.4 NHANES (ACSM Abstract: 2021)

5.5 Yoga (ACSM Abstract: 2022)

Chapter 6

Personal Projects

This will consist of all personal projects that I've done on my own spare time that has no affiliation with any organization.

6.1 Betting Models

- Baseball
- Football
- Nascar
- Tennis
- Roulette
  - Labouchere
  - Martingal
- Monte Carlo

6.2 My Functions

The opposite of the fill function. If I have a column

```
unfill_vec <- function(x) {
  same <- x == dplyr::lag(x)
  ifelse(!is.na(same) & same, "", x)
}

# Example
treatment <- tibble(
```

```
  person = c('Derrick Whitmore', 'Derrick Whitmore', 'Derrick Whitmore', 'Katherine Bu
  treatment = c(1, 2, 3, 1),
  response = c(7, 10, 9, 4)
)

treatment
```

```
## # A tibble: 4 x 3
##   person           treatment response
##   <chr>                <dbl>    <dbl>
## 1 Derrick Whitmore         1        7
## 2 Derrick Whitmore         2       10
## 3 Derrick Whitmore         3        9
## 4 Katherine Burke          1        4
```

Apply function

```
treatment$person <- unfill_vec(treatment$person)
treatment
```

```
## # A tibble: 4 x 3
##   person             treatment response
##   <chr>                  <dbl>    <dbl>
## 1 "Derrick Whitmore"         1        7
## 2 ""                         2       10
## 3 ""                         3        9
## 4 "Katherine Burke"          1        4
```

Chapter 7

Categories

7.1 NYY Part 1

Goal: Give the most likely pitch type for all of the pitches in the test dataset (year 3) using information from the training dataset (years 1-2)

The goal is to predict the type of pitch from the training set by only given a numerical value associated with the pitch type and not the actual name. This will be done through a series of steps:

Step 1: Check and visualize the data.

The first step is to look at and visualize the data. What are the variables in the provided dataset? The basic descriptive means of the independent variables and observations for each pitcher were displayed. Findings show that the pitchers in this dataset are likely to be right handed pitchers due to their release point (initposx) being on the third base side of the pitching rubber (Tables 2 and 4).Additionally, we can see that pitch type 9 and 10 are most likely refer to fastballs due to greater initial speed with pitch type 9 associated with a 2-seam fastball/sinker and pitch type 10 associated with a 4-seam fastball based on greater horizontal movement towards a right-handed hitter (breakx) for type 9 and lesser vertical movements downward (breakz) for type 10. Furthermore, Pitcher 3 has only 12 observations (pitches) in the *pitchclassificationtrain* set which is not an efficient sample size to train and test a model for future predictions. Therefore, I will take this in consideration when determining the model to be used for final predictions. I will test separate models for individual pitchers and the total model performance for addressing Pitcher 3 and Pitcher 6. As expected, the correlation matrix show's significant ($p < .05$) correlations amongst independent variables ruling out regression based models such as logistic regression.

Based on the data and research question, I will fit and evaluate the performance of three machine learning classification algorithms: decision tree (DT), k-nearest neighbor (K-NN), and support vector machine (SVM).

- Table: Basic Means of Variables
- Table: Total Observations(pitches) for each Pitcher in Training Set
- Table: Means of Variables for Individual Pitcher by Type
- Table: Correlation Matrix of Independent Variables

```
#
# Prepare Data
#

# Means of current data
tablemean <- train %>%
  group_by(type) %>%
  summarise(mph = mean(initspeed),
            spin = mean(spinrate),
            breakx = mean(breakx),
            breakz = mean(breakz),
            initx = mean(initposx),
            initz = mean(initposz),
            ext = mean(extension))
# Total observations
totalobs <- train %>%
  group_by(pitcherid) %>%
  summarise(N = n())
# Descriptives of Individual SP Type
SP_type <- train %>%
  group_by(pitcherid, type) %>%
  summarise(mph = mean(initspeed),
            spin = mean(spinrate),
            breakx = mean(breakx),
            breakz = mean(breakz),
            initx = mean(initposx),
            initz = mean(initposz),
            ext = mean(extension),
            pitches = n())
SP_type$Pitcher <- c("Pitcher1", "", "", "", "Pitcher2", "", "", "", "Pitcher3", "", "
SP_type <- SP_type %>%
  ungroup() %>%
  select(Pitcher,type:pitches)
# Correlations among variables
source("C:/Users/jadam/Desktop/jfa_book/_data/correlation_matrix.R")
# Run correlation matrix
cor_matrixFR <- as.data.frame(correlation_matrix(train[, -c(1:4,12)], digits = 2, use =
```

```r
# Table 2: Basic Means of Variables
knitr::kable(tablemean, align = "c", caption = 'Basic Means of Variables', digits = 3) %>%
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

Table 7.1: Basic Means of Variables

type	mph	spin	breakx	breakz	initx	initz	ext
2	76.985	2512.840	4.892	-6.841	-1.772	5.952	6.193
3	82.459	1867.164	1.059	-0.033	-1.383	5.754	6.207
4	83.821	1364.294	-5.607	3.453	-1.744	5.895	6.196
7	84.628	988.921	-2.907	2.479	-1.023	5.993	6.206
8	88.903	2346.131	1.233	4.711	-1.815	5.813	6.205
9	91.151	2065.167	-7.126	6.907	-1.828	5.856	6.204
10	92.141	2131.526	-3.185	9.447	-1.627	5.942	6.195

```r
# Table 3: Total Observations(pitches) for each Pitcher in Training Set
knitr::kable(totalobs, align = "c", caption = 'Total Observations(pitches) for each Pitcher in Tr
  kableExtra::kable_styling(latex_options = "HOLD_position") %>%
  footnote(general = 'Pitcher 3 has n=12 observations', general_title = "Note", footnote_as_chunk
```

Table 7.2: Total Observations(pitches) for each Pitcher in Training Set

pitcherid	N
1	1049
2	2137
3	12
4	1840
5	5609

Note Pitcher 3 has n=12 observations

```r
# Table 4: Means of Variables for Individual Pitcher by Type
knitr::kable(SP_type, align = "c", caption = 'Means of Variables for Individual Pitcher by Type',
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

Table 7.3: Means of Variables for Individual Pitcher by Type

Pitcher	type	mph	spin	breakx	breakz	initx	initz	ext	pitches
Pitcher1	2	77.185	2951.308	5.829	-6.466	-1.854	6.397	6.183	257
	4	81.256	1432.336	-6.688	0.901	-1.838	6.413	6.198	255
	9	88.111	2206.983	-8.489	3.433	-2.062	6.275	6.186	421
	10	89.380	2232.940	-5.992	7.134	-1.886	6.405	6.194	116
Pitcher2	2	79.726	2574.145	5.550	-6.765	-2.184	5.860	6.199	505
	4	87.640	1619.410	-5.492	3.156	-2.352	5.743	6.185	257
	9	93.987	2208.271	-6.268	7.541	-2.265	5.758	6.214	614
	10	93.990	2241.333	-1.666	8.986	-2.268	5.856	6.196	761
Pitcher3	3	84.724	2044.592	-0.095	4.388	3.885	6.662	6.212	2
	9	86.873	2041.951	9.506	4.935	4.145	6.437	6.218	4
	10	87.670	2098.654	4.792	8.584	4.069	6.510	6.136	6
Pitcher4	2	81.272	2624.056	4.391	-2.727	-1.920	5.849	6.196	231
	4	87.025	1430.257	-7.692	3.305	-2.151	5.694	6.183	192
	8	88.950	2490.009	1.946	4.423	-1.990	5.846	6.210	444
	9	93.422	2309.789	-7.346	7.695	-2.076	5.860	6.192	303
	10	93.364	2336.793	-4.694	9.769	-1.968	5.923	6.196	670
Pitcher5	2	72.034	2167.257	3.957	-9.054	-1.235	5.861	6.190	490
	3	82.437	1865.416	1.070	-0.077	-1.435	5.746	6.207	203
	4	82.316	1211.610	-4.574	4.660	-1.331	5.807	6.203	626
	7	84.628	988.921	-2.907	2.479	-1.023	5.993	6.206	901
	8	88.813	2068.383	-0.143	5.268	-1.476	5.751	6.195	230
	9	90.447	1927.528	-7.096	7.429	-1.568	5.782	6.208	1610
	10	90.928	1981.326	-3.100	9.710	-1.167	5.956	6.194	1549

```
# Table 5: Correlation Matrix of Independent Variables
knitr::kable(cor_matrixFR, align = "c", caption = 'Correlation Matrix of Independent Va
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

Table 7.4: Correlation Matrix of Independent Variables

	initspeed	breakx	breakz	initposx	initposz	extension	spinrate
initspeed							
breakx	-0.54***						
breakz	0.87***	-0.60***					
initposx	-0.21***	0.07***	0.02*				
initposz	-0.13***	0.08***	-0.07***	0.21***			
extension	0.01	-0.01	0.01	0.01	-0.01		
spinrate	0.11***	0.37***	-0.10***	-0.45***	0.05***	-0.01	

Step 2: Prepare the data to be fitted to each of the models.

The independent variables were first normalized to ensure the units were properly scaled. Prior to determining which algorithm to use for predicting the final pitch type, the *pitchclassificationtrain* dataset was split (75%/25%) into a training and testing set in order to evaluate model performance for the three different machine learning algorithms. The training set will be used to train each of the models which would then predict pitch type on the testing set. Model performance is evaluated based on the models ability to accurately predict the pitch type in the testing set. In addition, the training set was further separated for each of the five pitchers to run six separate models (five for each pitcher and one with data from all five pitchers) for the DT and K-NN. Models will be evaluated and compared based on their ability to accurately predict pitch type in the testing set. Because Pitcher 6 does not have any data to train on, total model performance will be used to predict pitch type for Pitcher 6 in the *pitchclassificationtest* set. Additionally, due to the limited amount of data available for Pitcher 3, I expect to use the total model performance to predict pitch type for Pitcher 3 in the *pitchclassificationtest* set as well. If accuracy for the total model is greater then accuracy for the separate models, the total model will be used to predict performance for all pitchers. Otherwise, the individual pitcher data will be used to predict that pitchers pitch type in the *pitchclassificationtest* set. For instance, if the K-NN model had a greater predicted pitch type accuracy for Pitcher 2 compared to the total K-NN model then the model for Pitcher 2 will be used to predict pitch type for Pitcher 2 in the *pitchclassificationtest* data set.

Step 3: Evaluate model performance by examining its accuracy in predicting pitch type in the testing set

For each of the three algorithms, separate models were trained on the training set and then predictions were made on the testing set (with the dependent variable, pitch type, removed). The results from the models predictions were compared to the actual results with performance being represented by an accuracy percentage.

7.1.1 Decision Tree

Six separate decision tree's were created, five for each pitcher and a total model using data from all five pitchers. After training the data for each model and making predictions on the testing set, the total model performance was 84% accurate in predicting pitch type. Greater performance was found for the separate models for Pitcher 1 (93%), Pitcher 2 (94%), Pitcher 4 (87%), and Pitcher 5 (88%) with an expected low accuracy of 66.67% for Pitcher 3 (Table 5).

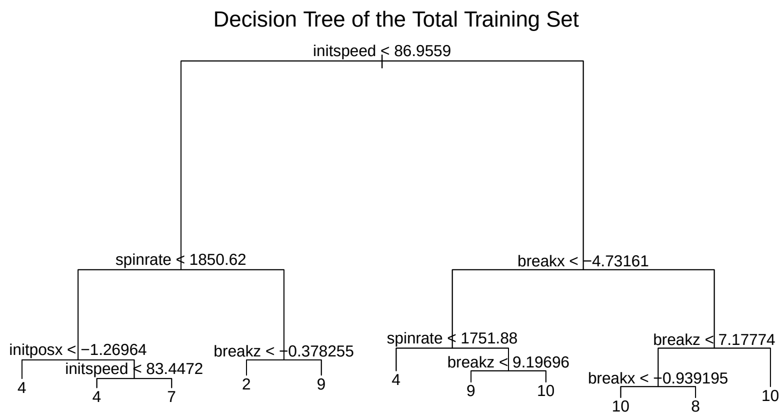
```
# Set up dataset for 6 different models
trainM <- train %>%
  select(initspeed:type)
```

```
SP1 <- train %>%
  filter(pitcherid == 1) %>%
  select(initspeed:type)
SP2 <- train %>%
  filter(pitcherid == 2) %>%
  select(initspeed:type)
SP3 <- train %>%
  filter(pitcherid == 3) %>%
  select(initspeed:type)
SP4 <- train %>%
  filter(pitcherid == 4) %>%
  select(initspeed:type)
SP5 <- train %>%
  filter(pitcherid == 5) %>%
  select(initspeed:type)
# Decision Tree
treeM <- tree(type ~ ., data = trainM)
sp1D <- tree(type ~ ., data = SP1)
sp2D <- tree(type ~ ., data = SP2)
sp3D <- tree(type ~ ., data = SP3)
sp4D <- tree(type ~ ., data = SP4)
sp5D <- tree(type ~ ., data = SP5)
# Misclassifications
missclassM <- summary(treeM)[[7]][1]/summary(treeM)[[7]][2]
missclass1 <- summary(sp1D)[[7]][1]/summary(sp1D)[[7]][2]
missclass2 <- summary(sp2D)[[7]][1]/summary(sp2D)[[7]][2]
missclass3 <- summary(sp3D)[[7]][1]/summary(sp3D)[[7]][2]
missclass4 <- summary(sp4D)[[7]][1]/summary(sp4D)[[7]][2]
missclass5 <- summary(sp5D)[[7]][1]/summary(sp5D)[[7]][2]

# Model Accuracy
##Split training data into training and testing set
set.seed(27)
splitM = sample.split(trainM$type, SplitRatio = 0.75)
split1 = sample.split(SP1$type, SplitRatio = 0.75)
split2 = sample.split(SP2$type, SplitRatio = 0.75)
split3 = sample.split(SP3$type, SplitRatio = 0.75)
split4 = sample.split(SP4$type, SplitRatio = 0.75)
split5 = sample.split(SP5$type, SplitRatio = 0.75)
##Training & Test set
training_set = subset(trainM, splitM == TRUE)
test_set = subset(trainM, splitM == FALSE)
training_set1 = subset(SP1, split1 == TRUE)
test_set1 = subset(SP1, split1 == FALSE)
training_set2 = subset(SP2, split2 == TRUE)
```

```
test_set2 = subset(SP2, split2 == FALSE)
training_set3 = subset(SP3, split3 == TRUE)
test_set3 = subset(SP3, split3 == FALSE)
training_set4 = subset(SP4, split4 == TRUE)
test_set4 = subset(SP4, split4 == FALSE)
training_set5 = subset(SP5, split5 == TRUE)
test_set5 = subset(SP5, split5== FALSE)
## Training Tree
treeD_training <- tree(type ~ ., training_set)
sp1D_training <- tree(type ~ ., training_set1)
sp2D_training <- tree(type ~ ., training_set2)
sp3D_training <- tree(type ~ ., training_set3)
sp4D_training <- tree(type ~ ., training_set4)
sp5D_training <- tree(type ~ ., training_set5)
## Make predictions on the test set
tree.predM = predict(treeD_training, test_set[,-8], type="class")
tree.pred1 = predict(sp1D_training, test_set1[,-8], type="class")
tree.pred2 = predict(sp2D_training, test_set2[,-8], type="class")
tree.pred3 = predict(sp3D_training, test_set3[,-8], type="class")
tree.pred4 = predict(sp4D_training, test_set4[,-8], type="class")
tree.pred5 = predict(sp5D_training, test_set5[,-8], type="class")
##Accuracy
m <- confusionMatrix(table(tree.predM, test_set$type))$overall[1]
m1 <- confusionMatrix(table(tree.pred1, test_set1$type))$overall[1]
m2 <- confusionMatrix(table(tree.pred2, test_set2$type))$overall[1]
m3 <- confusionMatrix(table(tree.pred3, test_set3$type))$overall[1]
m4 <- confusionMatrix(table(tree.pred4, test_set4$type))$overall[1]
m5 <- confusionMatrix(table(tree.pred5, test_set5$type))$overall[1]

# Table: Decision Tree Model
dtmodel <- data.frame(Model = c('Total Model', 'Pitcher1', 'Pitcher2', 'Pitcher3', 'Pitcher4', 'P
                 Accuracy = c(m,m1,m2,m3,m4,m5))
```

Decision Tree Plot

```
# Plot the decison Tree of the total Model
plot(treeD_training)
text(treeD_training, cex= 1.1)
mtext("Decision Tree of the Total Training Set", line = 1, cex = 1.5)
```



Decision Tree Table:

```r
# Kable of DT
knitr::kable(dtmodel, align = "c", caption = 'Decision Tree Model Performance', digits
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

Table 7.5: Decision Tree Model Performance

Model	Accuracy
Total Model	0.84
Pitcher1	0.93
Pitcher2	0.94
Pitcher3	0.67
Pitcher4	0.87
Pitcher5	0.88

7.1.2 K-Nearest Neighbor

The same six separate model approach was used to train and test the data using K-NN. The K-NN algorithm greatly improved the predictive performance for the total model and each of the separate pitcher models (other than Pitcher 3). Total model accurately predicted 91% of the pitch type in the testing set with Pitcher 1 (96%), Pitcher 2 (96%), Pitcher 4 (93%), and Pitcher 5 (90%) all having greater accuracy then the decision tree model performance.

Functions:

- Normalize = Normalizes the data
- PreProcess = used to normalized the data then split the data into testing and training sets

```
# Normalize function
normfun <- function(x){
  return((x - min(x)) / (max(x) - min(x)))
}

# PreProcess function
preprocess <- function(x){

  train.n <- as.data.frame(lapply(x[, -c(1,9)], normfun))
  train.n$type <- x$type
  # Split Train Data to test model
  set.seed(27)
  split = sample.split(train.n$type, SplitRatio = 0.75)
  training_set = subset(train.n, split == TRUE)
  test_set = subset(train.n, split == FALSE)
  return(list(training_set, test_set))
}
```

Preprocess

```
# Subset data for the separate models
trainM_knn <- train %>%
  select(pitcherid,initspeed:type)
SP1_knn <- train %>%
  filter(pitcherid == 1) %>%
  select(pitcherid,initspeed:type)
SP2_knn <- train %>%
  filter(pitcherid == 2) %>%
  select(pitcherid,initspeed:type)
SP3_knn <- train %>%
  filter(pitcherid == 3) %>%
  select(pitcherid,initspeed:type)
SP4_knn <- train %>%
  filter(pitcherid == 4) %>%
  select(pitcherid,initspeed:type)
SP5_knn <- train %>%
  filter(pitcherid == 5) %>%
  select(pitcherid,initspeed:type)

##Split training data into training and testing set
# Total model
```

```
dfL <- preprocess(trainM_knn)
training_setk <-dfL[[1]]
test_setk <- dfL[[2]]
# SP1
dfL <- preprocess(SP1_knn)
training_setk1 <-dfL[[1]]
test_setk1 <- dfL[[2]]
# SP 2
dfL <- preprocess(SP2_knn)
training_setk2 <-dfL[[1]]
test_setk2 <- dfL[[2]]
# SP 3
dfL <- preprocess(SP3_knn)
training_setk3 <-dfL[[1]]
test_setk3 <- dfL[[2]]
# SP 4
dfL <- preprocess(SP4_knn)
training_setk4 <-dfL[[1]]
test_setk4 <- dfL[[2]]
# SP 5
dfL <- preprocess(SP5_knn)
training_setk5 <-dfL[[1]]
test_setk5 <- dfL[[2]]
```

KNN:

- Building the KNN model for 5 SP
  - train = training data
  - test = testing data
  - cl = classifier/outcome/dependent variable
  - k = think 5 is standard

```
# Build KNN Model
knn.M = knn(train = training_setk[, -8],
              test = test_setk[, -8],
              cl = training_setk[, 8],
              k = 3,
              prob = TRUE)
# SP1
knn.1 = knn(train = training_setk1[, -8],
             test = test_setk1[, -8],
             cl = training_setk1[, 8],
             k = 9,
             prob = TRUE)
```

```
# SP2
knn.2 = knn(train = training_setk2[, -8],
             test = test_setk2[, -8],
             cl = training_setk2[, 8],
             k = 5,
             prob = TRUE)
# SP3
knn.3 = knn(train = training_setk3[, -8],
             test = test_setk3[, -8],
             cl = training_setk3[, 8],
             k = 3,
             prob = TRUE)
# SP4
knn.4 = knn(train = training_setk4[, -8],
             test = test_setk4[, -8],
             cl = training_setk4[, 8],
             k = 5,
             prob = TRUE)
# SP5
knn.5 = knn(train = training_setk5[, -8],
             test = test_setk5[, -8],
             cl = training_setk5[, 8],
             k = 5,
             prob = TRUE)
```

- Model Evaluation

```
# Model Evaluation
am <- confusionMatrix(table(knn.M,test_setk[, 8]))$overall[1]
m1 <- confusionMatrix(table(knn.1,test_setk1[, 8]))$overall[1]
m2 <- confusionMatrix(table(knn.2,test_setk2[, 8]))$overall[1]
m3 <- confusionMatrix(table(knn.3,test_setk3[, 8]))$overall[1]
m4 <- confusionMatrix(table(knn.4,test_setk4[, 8]))$overall[1]
m5 <- confusionMatrix(table(knn.5,test_setk5[, 8]))$overall[1]
```

Table: K-NN Model Performance

```
# Table of KNN
knnmodel <- data.frame(Model = c('Total Model', 'Pitcher1', 'Pitcher2', 'Pitcher3', 'Pitcher4', '
                 Accuracy = c(am,m1,m2,m3,m4,m5))

# Kable of KNN
knitr::kable(knnmodel, align = "c", caption = 'K-NN Model Performance', digits = 2) %>%
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

Table 7.6: K-NN Model Performance

Model	Accuracy
Total Model	0.91
Pitcher1	0.96
Pitcher2	0.95
Pitcher3	0.67
Pitcher4	0.93
Pitcher5	0.90

Then I made KNN predictions and merged together (see .Rmd file, code was set echo=FALSE)

7.1.3 Support Vector Machine (SVM)

As a result of K-NN resulting in an accuracy score above 90% for each separate pitcher model, a multiclass support vector algorithm was ran on the total model to improve the models performance for predicting Pitcher 3 and Pitcher 6 in the *pitchclassificationtest* set. The SVM resulted in a slight improvement in overall model performance (92%) compared to the K-NN total model.

Preprocess

```
# Test and Train Model
# Split training data into training and testing set
splitsvm = sample.split(train$type, SplitRatio = 0.75)
training_setsvm = subset(train, splitsvm == TRUE)
test_setsvm = subset(train, splitsvm == FALSE)
```

SVM

- Build the SVM Model

```
# Fit the model
svm1 = svm(formula = type ~ .,
         data = training_setsvm[, -c(1:4)],
         type = 'C-classification',
         kernel = 'radial')
```

- Model Evaluation

```
# Model Evaluation: Predict on test set
y_predsvm <- predict(svm1, test_setsvm[, -c(1:4, 12)])
# Accuracy: Confusion Matrix
svm_acc <- confusionMatrix(table(y_predsvm, test_setsvm$type))$overall[1]

# Table of SVM
svmmodel <- data.frame(Model = c('Total Model'),
                  Accuracy = c(svm_acc))
```

Then I made final SVM predictions (see .Rmd file, code was set echo=FALSE)

Step 4: Determine the model with the highest accuracy scores to predict pitch type in the *pitchclassificationtest* data

The separate K-NN models for Pitcher 1, Pitcher 2, Pitcher 4, and Pitcher 5 reported accuracy scores above 90% (Table 7). Therefore, it was decided to use the total *pitchclassificationtrain* data for each of the four pitchers to train K-NN models and make final pitch type predictions for these four pitchers in the *pitchclassificationtest* data set.

SVM reported the highest predictive accuracy for the total model (92%). It was therefore decided to train SVM on the total *pitchclassificationtrain* data to make final pitch type prediction for Pitcher 6 as well as Pitcher 3 (due to low observation of training data) in the *pitchclassificationtest* data set.

Final Comparisons Among Models

- Table: Comparing Model Performance

```
# Table Comparing Models
comp <- data.frame("Total Model" = as.numeric(c(t(dtmodel)[2], t(knnmodel)[2], svmmodel[1,2])),
                  "Pitcher 1" = as.numeric(c(t(dtmodel)[4], t(knnmodel)[4], '')),
                  "Pitcher 2" = as.numeric(c(t(dtmodel)[6], t(knnmodel)[6], '')),
                  "Pitcher 3" = as.numeric(c(t(dtmodel)[8], t(knnmodel)[8], '')),
                  "Pitcher 4" = as.numeric(c(t(dtmodel)[10], t(knnmodel)[10], '')),
                  "Pitcher 5" = as.numeric(c(t(dtmodel)[12], t(knnmodel)[12], '')))

rownames(comp) <- c("Decision Tree Model", "K-NN Model", 'SVM Model')

# Kable Comparing Models
knitr::kable(comp, align = "c", caption = 'Comparing Model Performance', digits = 2) %>%
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

Table 7.7: Comparing Model Performance

	Total.Model	Pitcher.1	Pitcher.2	Pitcher.3	Pitcher.4	Pitcher.5
Decision Tree Model	0.84	0.93	0.94	0.67	0.87	0.88
K-NN Model	0.91	0.96	0.95	0.67	0.93	0.90
SVM Model	0.92	NA	NA	NA	NA	NA

```
knitr::include_graphics("C:/Users/jadam/Desktop/jfa_book/_images/nyytable.jpg")
```

Table 9: Predicted Model Decision

Variables	Description
Pitcher 1	K-NN: Pitcher specific model
Pitcher 2	K-NN: Pitcher specific model
Pitcher 3	SVM: Total model
Pitcher 4	K-NN: Pitcher specific model
Pitcher 5	K-NN: Pitcher specific model
Pitcher 6	SVM: Total model

The Table above is created with the LaTex code below

```
\begin{table}[h]
    \centering
    \caption{Predicted Model Decision}
    \begin{tabular}{l c}
    \hline
    Variables & Description \\
    \hline
    Pitcher 1 & K-NN: Pitcher specific model \\
    Pitcher 2 & K-NN: Pitcher specific model \\
    Pitcher 3 & SVM: Total model \\
    Pitcher 4 & K-NN: Pitcher specific model \\
    Pitcher 5 & K-NN: Pitcher specific model \\
    Pitcher 6 & SVM: Total model \\
    \hline
    \end{tabular}
    \label{tab:my_label}
\end{table}
```

Step 5: Make final predictions

After training K-NN on the total *pitchclassificationtrain* data for each pitcher. Final predictions were made using each of the four pitchers separate K-NN models. SVM was trained on the total *pitchclassificationtrain* data and final predictions were made for Pitcher 3 and Pitcher 6. When final predictions were made for each pitcher, the data was merged together to produce a final data set of all pitcher's with their predicted pitcher type.

Step 6: Check and visualize the predicted results to the original data. To see if patterns match.

The predicted results were displayed along with the actual (i.e., *pitchclassificationtrain*) data by pitch type and pitcher to visualize if patterns match. Although it appears that velocity had decreased from years 1-2 to year 3 (91, 92 mph vs 87, 89mpg) overall patterns appears similar (e.g., pitch 7 had the overall lowest spin rate, pitch 2 the largest vertical break). Interestingly, it appears that Pitcher 3 and Pitcher 6 are both left-handed pitchers due to both having an initial release point on the first base side of the rubber. This may reduce accuracy rating due to the fact that the data was essentially training on right-handed pitchers to predict pitch type for a left-handed pitcher.

Table: Years 1-2 Table: Final Predictions Table: Actual Individual Pitcher by Pitch Type: Years 1-2 Table: Predicted Individual Pitcher by Pitch Type: Year 3

```
# Extract Model specific predictions
Finala <- final_KN %>%
  filter(pitcherid %in% c(1,2,3,4)) %>%
  rename('PredictedPitchType' = 'PitchPredKNN')
Finalb <- testsvm %>%
  filter(!pitcherid %in% c(1,2,3,4)) %>%
  rename('PredictedPitchType' = 'PitchPred_svm')
# Merge together
FinalNYY <- rbind(Finala, Finalb)

# Predicted Data: Final
NYYPred_by_pitch <- FinalNYY %>%
  group_by(PredictedPitchType) %>%
  summarise(mph = mean(initspeed),
            spin = mean(spinrate),
            breakx = mean(breakx),
            breakz = mean(breakz),
            initx = mean(initposx),
            initz = mean(initposz),
            ext = mean(extension))
NYYPred_by_pitcher <- FinalNYY %>%
  group_by(pitcherid, PredictedPitchType) %>%
```

```
  summarise(mph = mean(initspeed),
            spin = mean(spinrate),
            breakx = mean(breakx),
            breakz = mean(breakz),
            initx = mean(initposx),
            initz = mean(initposz),
            ext = mean(extension))

# Format table
NYYPred_by_pitcher$Pitcher <- c("Pitcher1", "", "", "", "Pitcher2", "", "", "", "Pitche
                                "Pitcher4", "", "", "", "", 'Pitcher5', "", "", "", ""
NYYPred_by_pitcher <- NYYPred_by_pitcher %>%
  ungroup() %>%
  select(Pitcher,PredictedPitchType:ext)

# Kable Model Comparisons - by pitch
knitr::kable(tablemean, align = "c", caption = 'Years 1-2', digits = 2) %>%
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

Table 7.8: Years 1-2

type	mph	spin	breakx	breakz	initx	initz	ext
2	76.99	2512.84	4.89	-6.84	-1.77	5.95	6.19
3	82.46	1867.16	1.06	-0.03	-1.38	5.75	6.21
4	83.82	1364.29	-5.61	3.45	-1.74	5.89	6.20
7	84.63	988.92	-2.91	2.48	-1.02	5.99	6.21
8	88.90	2346.13	1.23	4.71	-1.81	5.81	6.21
9	91.15	2065.17	-7.13	6.91	-1.83	5.86	6.20
10	92.14	2131.53	-3.19	9.45	-1.63	5.94	6.19

```
knitr::kable(NYYPred_by_pitch, align = "c", caption = 'Final Predictions', digits = 2)
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

Table 7.9: Final Predictions

PredictedPitchType	mph	spin	breakx	breakz	initx	initz	ext
2	75.67	2686.30	5.32	-5.93	-1.94	6.04	6.20
3	83.63	2008.46	4.93	4.61	3.68	6.41	6.22
4	82.13	1476.81	-6.11	2.43	-1.93	6.03	6.21
7	84.31	1050.90	-1.44	2.47	-0.60	6.03	6.17
8	86.48	2274.63	3.97	6.01	1.63	6.22	6.21
9	87.52	2125.88	-3.19	5.37	-0.43	6.03	6.20
10	89.14	2223.90	-2.05	8.91	-1.06	6.01	6.20

```
# Kable Model Comparisons - by pitcher
knitr::kable(SP_type[, -10], align = "c", caption = 'Actual Individual Pitcher by Pitch Type: Yea
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

Table 7.10: Actual Individual Pitcher by Pitch Type: Years 1-2

Pitcher	type	mph	spin	breakx	breakz	initx	initz	ext
Pitcher1	2	77.18	2951.31	5.83	-6.47	-1.85	6.40	6.18
	4	81.26	1432.34	-6.69	0.90	-1.84	6.41	6.20
	9	88.11	2206.98	-8.49	3.43	-2.06	6.28	6.19
	10	89.38	2232.94	-5.99	7.13	-1.89	6.41	6.19
Pitcher2	2	79.73	2574.14	5.55	-6.77	-2.18	5.86	6.20
	4	87.64	1619.41	-5.49	3.16	-2.35	5.74	6.19
	9	93.99	2208.27	-6.27	7.54	-2.26	5.76	6.21
	10	93.99	2241.33	-1.67	8.99	-2.27	5.86	6.20
Pitcher3	3	84.72	2044.59	-0.10	4.39	3.89	6.66	6.21
	9	86.87	2041.95	9.51	4.94	4.14	6.44	6.22
	10	87.67	2098.65	4.79	8.58	4.07	6.51	6.14
Pitcher4	2	81.27	2624.06	4.39	-2.73	-1.92	5.85	6.20
	4	87.02	1430.26	-7.69	3.31	-2.15	5.69	6.18
	8	88.95	2490.01	1.95	4.42	-1.99	5.85	6.21
	9	93.42	2309.79	-7.35	7.70	-2.08	5.86	6.19
	10	93.36	2336.79	-4.69	9.77	-1.97	5.92	6.20
Pitcher5	2	72.03	2167.26	3.96	-9.05	-1.24	5.86	6.19
	3	82.44	1865.42	1.07	-0.08	-1.43	5.75	6.21
	4	82.32	1211.61	-4.57	4.66	-1.33	5.81	6.20
	7	84.63	988.92	-2.91	2.48	-1.02	5.99	6.21
	8	88.81	2068.38	-0.14	5.27	-1.48	5.75	6.20
	9	90.45	1927.53	-7.10	7.43	-1.57	5.78	6.21
	10	90.93	1981.33	-3.10	9.71	-1.17	5.96	6.19

```
knitr::kable(NYYPred_by_pitcher, align = "c", caption = 'Predicted Individual Pitcher b
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

Table 7.11: Predicted Individual Pitcher by Pitch Type: Year 3

Pitcher	PredictedPitchType	mph	spin	breakx	breakz	initx	initz	ext
Pitcher1	2	76.29	2940.96	5.74	-6.35	-1.85	6.40	6.20
	4	80.56	1432.89	-6.44	0.71	-1.83	6.40	6.23
	9	86.66	2209.15	-7.17	3.24	-2.05	6.29	6.20
	10	87.47	2271.93	-4.56	6.26	-1.89	6.39	6.20
Pitcher2	2	73.48	2573.31	5.55	-6.74	-2.18	5.85	6.19
	4	81.68	1635.80	-5.51	3.40	-2.35	5.74	6.20
	9	87.44	2186.73	-6.68	7.35	-2.25	5.76	6.20
	10	87.67	2237.46	-2.35	8.76	-2.26	5.84	6.21
Pitcher3	2	84.13	2269.58	5.47	4.57	4.21	6.41	6.12
	3	83.79	2024.08	5.33	5.03	4.10	6.46	6.22
	4	85.06	1694.66	10.66	5.92	4.00	6.53	6.21
	7	84.75	1623.93	10.24	5.84	4.01	6.45	6.24
	8	85.39	2135.16	5.38	7.17	4.11	6.49	6.20
	9	83.87	2156.94	2.97	5.60	4.14	6.46	6.17
	10	85.94	2083.01	5.63	8.41	4.06	6.51	6.20
Pitcher4	2	80.47	2620.86	4.38	-2.71	-1.91	5.85	6.21
	4	86.10	1442.49	-7.76	3.63	-2.15	5.70	6.21
	8	88.03	2499.23	2.07	4.32	-1.99	5.85	6.21
	9	92.55	2298.29	-7.29	7.27	-2.04	5.86	6.19
	10	92.57	2340.87	-4.82	9.68	-1.99	5.91	6.19
Pitcher5	2	71.85	2195.99	3.96	-8.94	-1.24	5.87	6.20
	3	81.76	1825.41	0.33	-0.33	-1.31	5.78	6.20
	4	82.04	1225.32	-4.70	4.70	-1.33	5.79	6.17
	7	84.27	1002.53	-2.42	2.19	-0.98	6.00	6.17
	8	88.30	2055.71	-0.50	4.75	-1.50	5.70	6.22
	9	90.07	1943.43	-7.33	7.49	-1.60	5.76	6.22
	10	90.53	1967.73	-3.23	9.61	-1.16	5.96	6.20
Pitcher6	9	87.03	2021.60	3.21	5.51	2.09	5.97	6.20
	10	91.60	2048.80	3.13	10.87	2.17	6.10	6.10

Chapter 8

Statistics

This chapter will consist of statistics relating to hypothesis testing

- ANOVA
- T-test
- Regression
- General linear model
- Bayesian Statistics

Chapter 9

Machine Learning

This chapter will consist of machine learning and predictive modeling. Mainly from the Udemy and Coursera courses I have taken

- Regressions
  - Linear
  - Logistic
- Decision Trees and Random Forest
- K-Nearest Neighbor
- K-means clustering
- Support Vector Machine
- Neural Nets