

Basics of Statistical Learning

David Dalpiaz

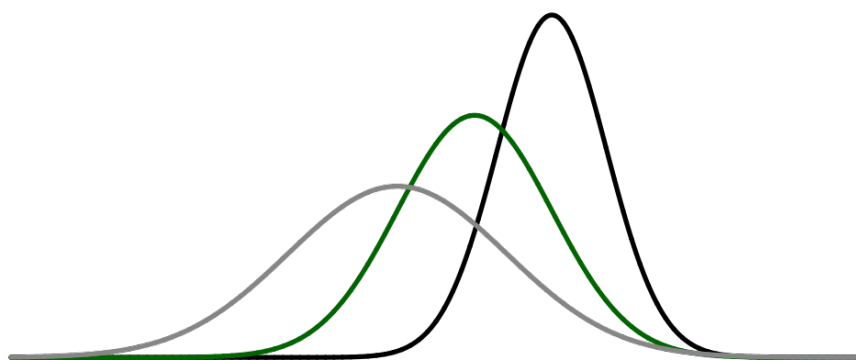
2019-08-30

Contents

1	Introduction	7
1.1	Regression: Powerlifting	8
1.1.1	Background	8
1.1.2	Data	8
1.1.3	EDA	8
1.1.4	Modeling	13
1.1.5	Model Evaluation	14
1.1.6	Discussion	16
1.2	Classification: Handwritten Digits	17
1.2.1	Background	17
1.2.2	Data	17
1.2.3	EDA	17
1.2.4	Modeling	18
1.2.5	Model Evaluation	18
1.2.6	Discussion	19
1.3	Clustering: NBA Players	20
1.3.1	Background	20
1.3.2	Data	20
1.3.3	EDA	22
1.3.4	Modeling	24
1.3.5	Model Evaluation	25
1.3.6	Discussion	28
2	Computing	29
2.1	Resources	29
2.1.1	R	30
2.1.2	RStudio	30
2.1.3	R Markdown	30
2.2	BSL Idioms	30
2.2.1	Reference Style	30
2.2.2	BSL Style Overrides	31
2.2.3	Objects and Functions	31
2.2.4	Print versus Return	32

2.2.5	Help	33
2.2.6	Keyboard Shortcuts	33
2.3	Common Issues	34

Preface



Welcome to Basics of Statistical Learning!

- TODO: Warning about development.
- TODO: Warning about PDF version.
- TODO: Transfer acknowledgements.
- TODO: discuss <https://davidalpiaz.github.io/r4sl/>
- TODO: course vs book
- TODO: stat432.org
- TODO: <https://yihui.name/en/2013/06/fix-typo-in-documentation/>
- TODO: <http://varianceexplained.org/r/ds-ml-ai/>

License



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/)

Chapter 1

Introduction

```
library(readr)
library(tibble)
library(dplyr)
library(purrr)
library(ggplot2)
library(ggthemes)
library(lubridate)
library(randomForest)
library(rpart)
library(rpart.plot)
library(cluster)
library(caret)
library(factoextra)
library(rsample)
library(janitor)
library(rvest)
library(dendextend)
library(knitr)
library(kableExtra)
library(ggthemes)
```

- TODO: Show package messaging? check conflicts!
- TODO: Should this be split into three analyses with different packages?

1.1 Regression: Powerlifting

1.1.1 Background

- TODO: <https://www.openpowerlifting.org/>
- TODO: <https://en.wikipedia.org/wiki/Powerlifting>

1.1.2 Data

- TODO: Why `readr::col_factor()` and not just `col_factor()`?
- TODO: Characters should be character and “categories” should be factors.
- TODO: Is `na.omit()` actually a good idea?

```
p1 = read_csv("data/pl.csv", col_types = cols(Sex = readr::col_factor()))
```

```
p1
```

```
## # A tibble: 3,604 x 8
##   Name          Sex  Bodyweight  Age Squat Bench Deadlift Total
##   <chr>         <fct>      <dbl> <dbl> <dbl> <dbl>   <dbl> <dbl>
## 1 Ariel Stier   F          60      32 128.   72.5    150   350
## 2 Nicole Bueno F          60      26 110    60     135   305
## 3 Lisa Peterson F         67.5     28 118.   67.5    138.   322.
## 4 Shelby Bandula F         67.5     26  92.5   67.5    140   300
## 5 Lisa Lindhorst F         67.5     28  92.5   62.5    132.   288.
## 6 Laura Burnett F         67.5     30  90     45     108.   242.
## 7 Suzette Bradley F          75     38 125     75     158.   358.
## 8 Norma Romero F          75     20  92.5   42.5    125   260
## 9 Georgia Andrews F         82.5     29 108.   52.5    120   280
## 10 Christal Bundang F          90     30 100     55     125   280
## # ... with 3,594 more rows
```

1.1.3 EDA

```
set.seed(1)

# test-train split
pl_tst_trn_split = initial_split(pl, prop = 0.80)
pl_trn = training(pl_tst_trn_split)
pl_tst = testing(pl_tst_trn_split)

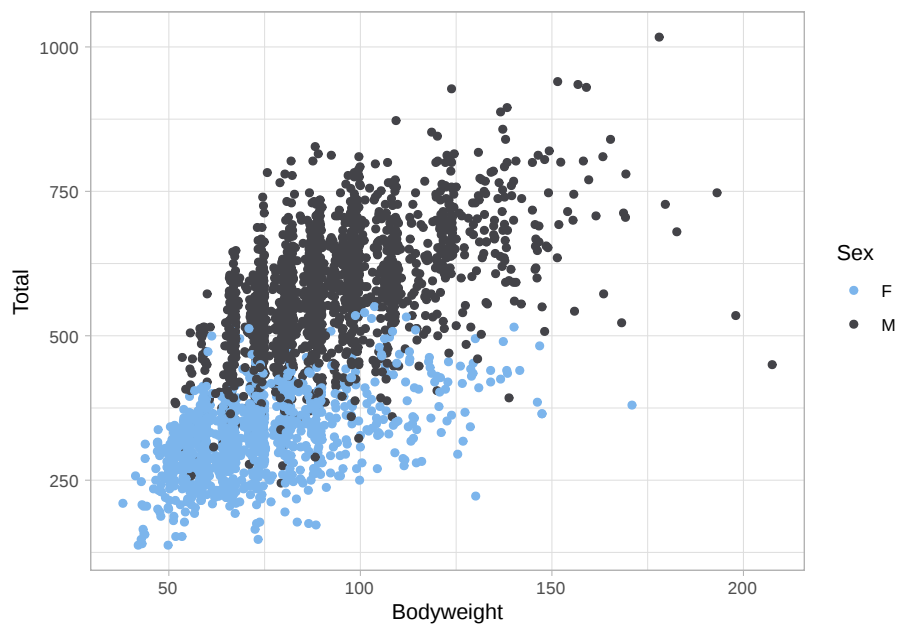
# estimation-validation split
pl_est_val_split = initial_split(pl_trn, prop = 0.80)
```

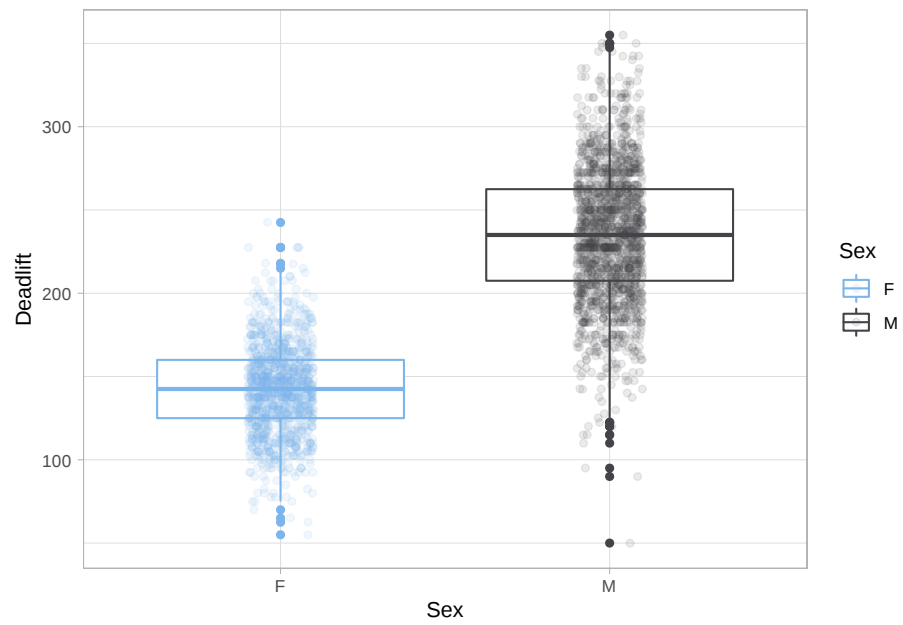
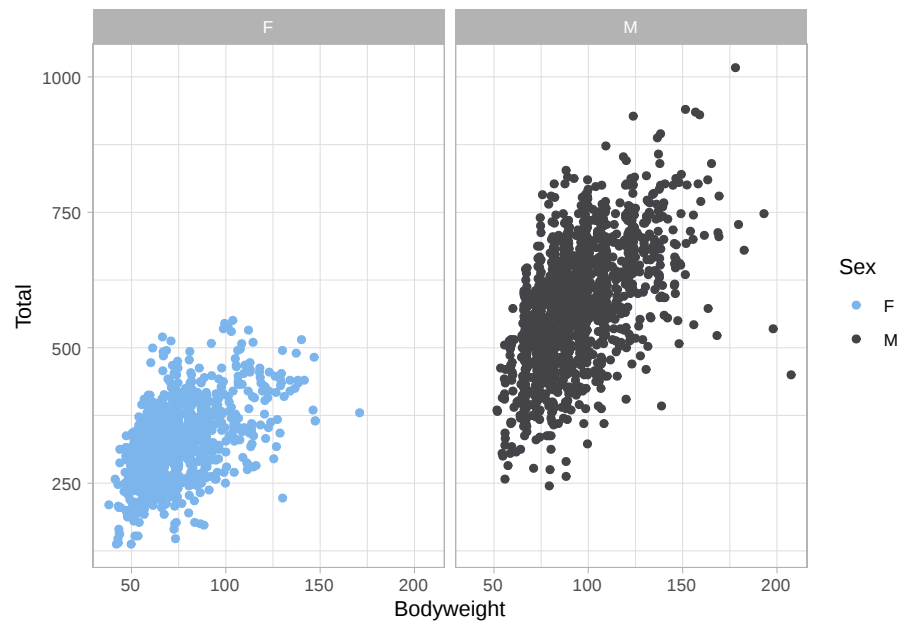


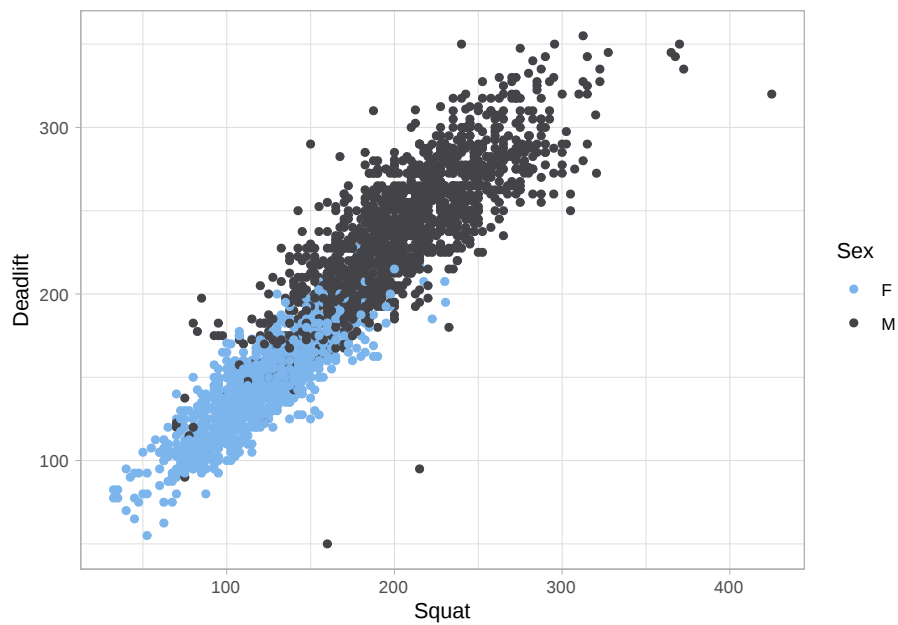
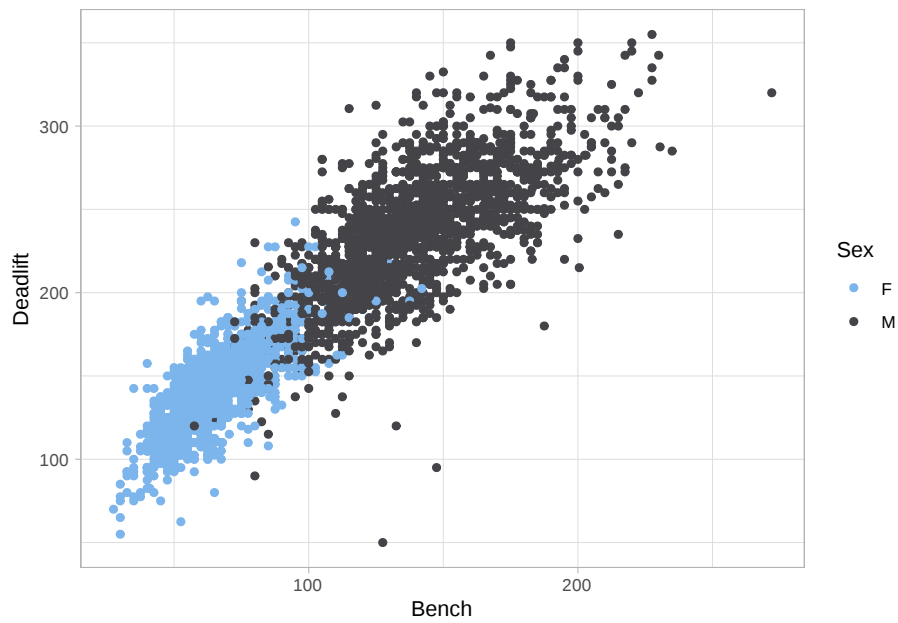
```
pl_est = training(pl_est_val_split)
pl_val = testing(pl_est_val_split)

rm(pl)
```

- TODO: Train can be used however you want. (Including EDA.)
- TODO: Test can only be used after all model decisions have been made!



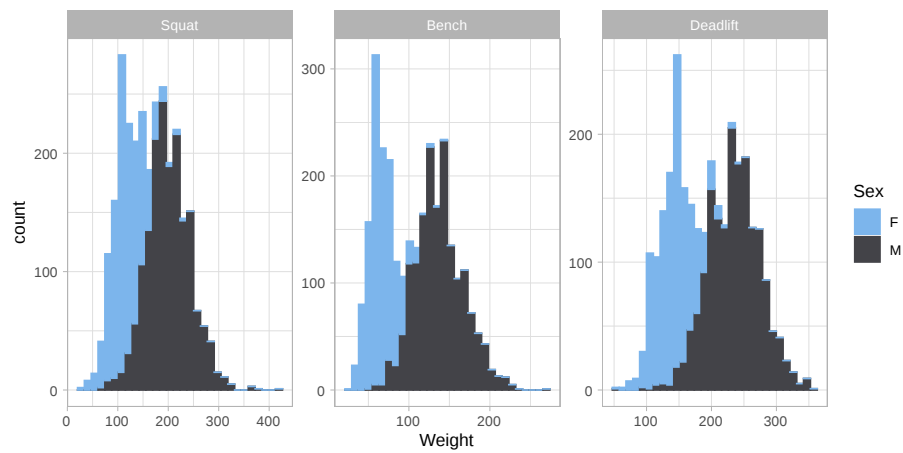
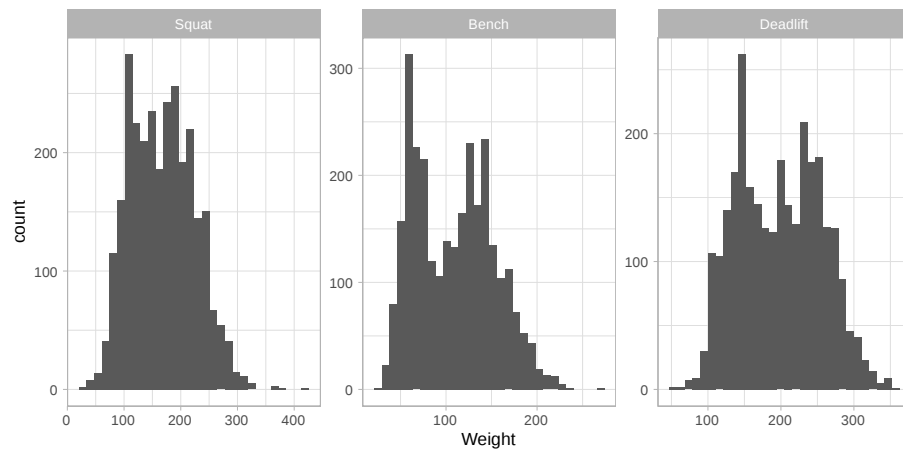


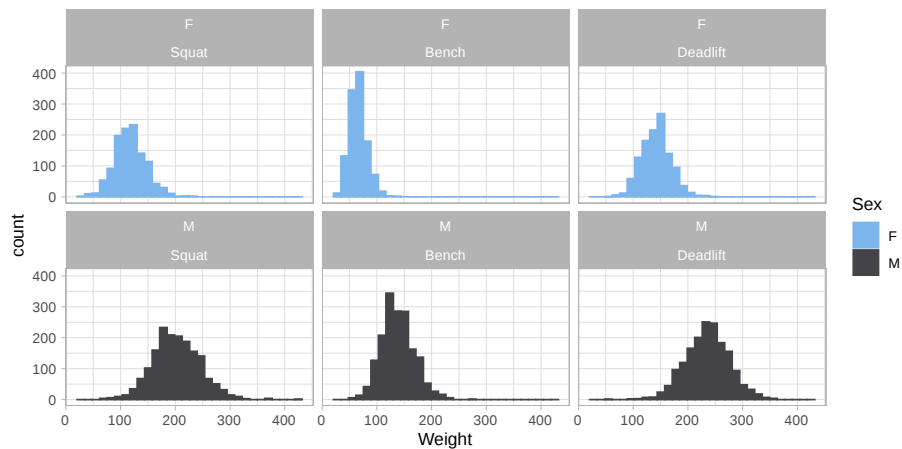


```
pl_trn_tidy = gather(pl_trn, key = "Lift", value = "Weight",  
                     Squat, Bench, Deadlift)
```

```
pl_trn_tidy$Lift = factor(pl_trn_tidy$Lift, levels = c("Squat", "Bench", "Deadlift"))
```

- TODO: <https://www.tidyverse.org/>
- TODO: https://en.wikipedia.org/wiki/Tidy_data
- TODO: <http://vita.had.co.nz/papers/tidy-data.pdf>





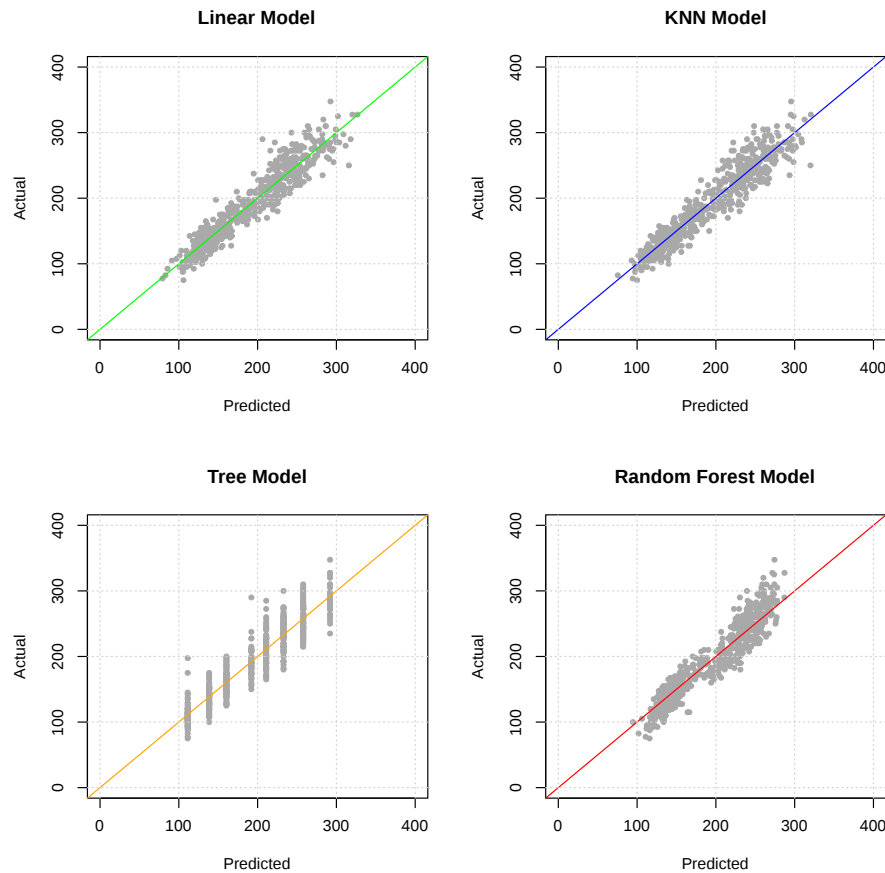
1.1.4 Modeling

```
dl_mod_form = formula(Deadlift ~ Sex + Bodyweight + Age + Squat + Bench)

set.seed(1)
lm_mod = lm(dl_mod_form, data = pl_est)
knn_mod = caret::knnreg(dl_mod_form, data = pl_est)
rf_mod = randomForest(dl_mod_form, data = pl_est)
rp_mod = rpart(dl_mod_form, data = pl_est)
```

- TODO: Note: we are not using `Name`. Why? We are not using `Total`. Why?
- TODO: look what happens with `Total`! You'll see it with `lm()`, you'll be optimistic with `randomForest()`.
- TODO: What variables are allowed? (With respect to real world problem.)
- TODO: What variables lead to the best predictions?

1.1.5 Model Evaluation



```
calc_rmse = function(actual, predicted) {
  sqrt(mean((actual - predicted) ^ 2))
}

c(calc_rmse(actual = pl_val$Deadlift, predicted = predict(lm_mod, pl_val)),
  calc_rmse(actual = pl_val$Deadlift, predicted = predict(knn_mod, pl_val)),
  calc_rmse(actual = pl_val$Deadlift, predicted = predict(rp_mod, pl_val)),
  calc_rmse(actual = pl_val$Deadlift, predicted = predict(rf_mod, pl_val)))

## [1] 18.26654 19.19625 21.68142 19.23643

reg_preds = map(list(lm_mod, knn_mod, rp_mod, rf_mod), predict, pl_val)
map_dbl(reg_preds, calc_rmse, actual = pl_val$Deadlift)

## [1] 18.26654 19.19625 21.68142 19.23643
```

- TODO: Never supply `data = df` to `predict()`. You have been warned.

```
knitr::include_graphics("img/sim-city.jpg")
```



```
calc_mae = function(actual, predicted) {
  mean(abs(actual - predicted))
}
```

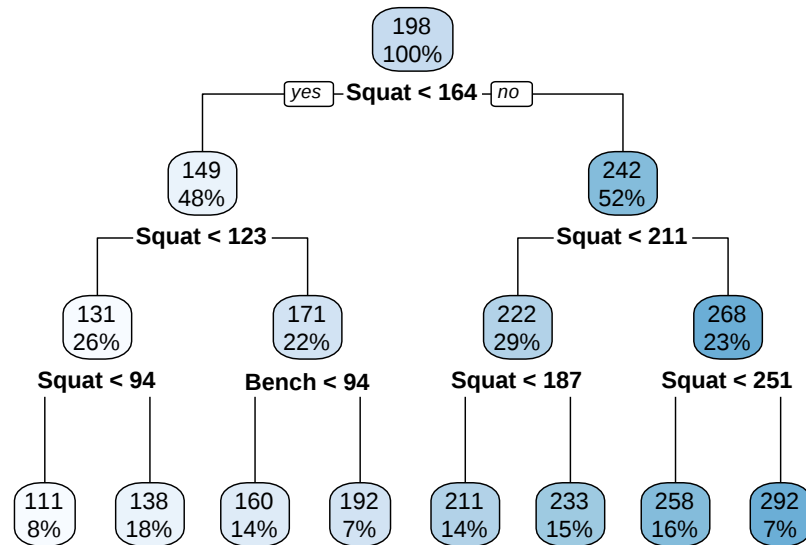
```
map_dbl(reg_preds, calc_mae, actual = pl_val$Deadlift)
```

```
## [1] 14.38953 14.99748 17.14823 15.28626
```

```
reg_results = tibble(
  Model = c("Linear", "KNN", "Tree", "Forest"),
  RMSE = map_dbl(reg_preds, calc_rmse, actual = pl_val$Deadlift),
  MAE = map_dbl(reg_preds, calc_mae, actual = pl_val$Deadlift))
```

Model	RMSE	MAE
Linear	18.26654	14.38953
KNN	19.19625	14.99748
Tree	21.68142	17.14823
Forest	19.23643	15.28626

1.1.6 Discussion



```
lm_mod_final = lm(dl_mod_form, data = pl_trn)
```

```
calc_rmse(actual = pl_tst$Deadlift,
           predicted = predict(lm_mod_final, pl_tst))
```

```
## [1] 22.29668
```

- TODO: Is this a good model?
- TODO: Is this model useful?

```
william_biscarri = tibble(
  Name = "William Biscarri",
  Age = 28,
  Sex = "M",
  Bodyweight = 83,
  Squat = 130,
  Bench = 90
)
```

```
predict(lm_mod_final, william_biscarri)
```

```
##          1
## 175.495
```


1.2 Classification: Handwritten Digits

1.2.1 Background

- TODO: https://en.wikipedia.org/wiki/MNIST_database
- TODO: <http://yann.lecun.com/exdb/mnist/>

1.2.2 Data

- TODO: How is this data pre-processed?
- TODO: <https://gist.github.com/daviddalpia/ae62ae5ccd0bada4b9acd6dbc9008706>
- TODO: <https://github.com/itsrainingdata/mnistR>
- TODO: <https://pjreddie.com/projects/mnist-in-csv/>
- TODO: <http://varianceexplained.org/r/digit-eda/>

```
mnist_trn = read_csv(file = "data/mnist_train_subest.csv")
mnist_tst = read_csv(file = "data/mnist_test.csv")
```

```
mnist_trn_y = as.factor(mnist_trn$X1)
mnist_tst_y = as.factor(mnist_tst$X1)
```

```
mnist_trn_x = mnist_trn[, -1]
mnist_tst_x = mnist_tst[, -1]
```

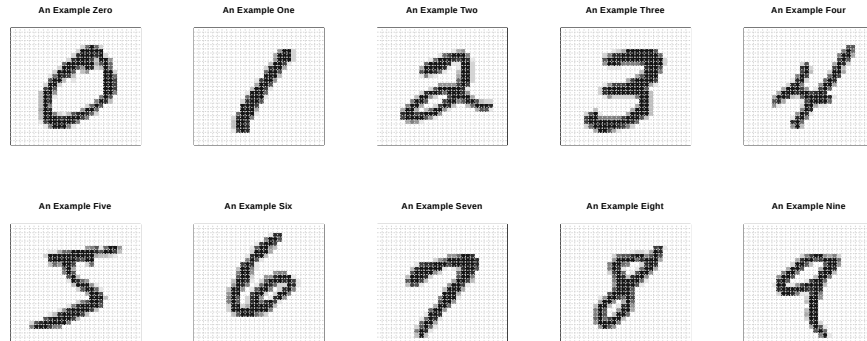
- TODO: If we were going to tune a model, we would need a validation split as well. We're going to be lazy and just fit a single random forest.
- TODO: This is an agreed upon split.

1.2.3 EDA

```
pixel_positions = expand_grid(j = sprintf("%02.0f", 1:28),
                              i = sprintf("%02.0f", 1:28))
pixel_names = paste("pixel", pixel_positions$i, pixel_positions$j, sep = "-")
```

```
colnames(mnist_trn_x) = pixel_names
colnames(mnist_tst_x) = pixel_names
```

```
show_digit = function(arr784, col = gray(12:1 / 12), ...) {
  image(matrix(as.matrix(arr784), nrow = 28)[, 28:1],
        col = col, xaxt = "n", yaxt = "n", ...)
  grid(nx = 28, ny = 28)
}
```



1.2.4 Modeling

```
set.seed(42)
mnist_rf = randomForest(x = mnist_trn_x, y = mnist_trn_y, ntree = 100)
```

1.2.5 Model Evaluation

```
mnist_tst_pred = predict(mnist_rf, mnist_tst_x)
mean(mnist_tst_pred == mnist_tst_y)
```

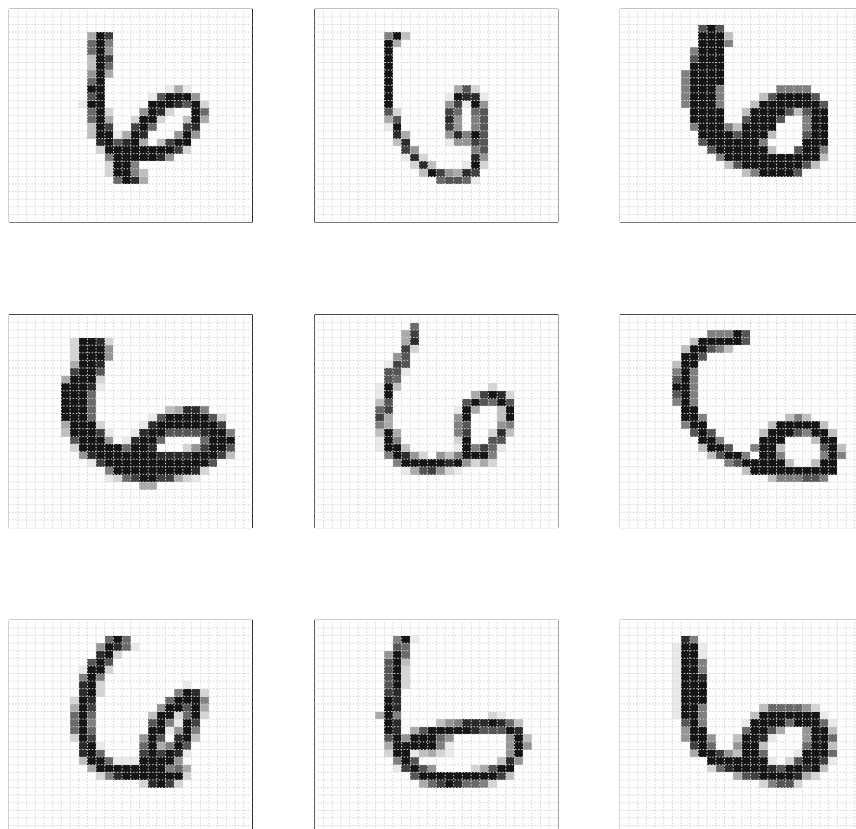
```
## [1] 0.8839
```

```
table(predicted = mnist_tst_pred, actual = mnist_tst_y)
```

```
##          actual
## predicted    0    1    2    3    4    5    6    7    8    9
##      0 959    0   14    6    1   15   22    1   10   10
##      1  0 1112    5    5    1   16    5    9    5    6
##      2  1    2  928   31    3    5   19   24   17    8
##      3  0    2   11  820    1   24    0    1   13   13
##      4  4    0   13    1  839   21   39   11   18   40
##      5  3    1    1   88    3  720   18    1   25    9
##      6  7    2   15    3   25   15  848    0   18    2
##      7  2    1   29   24    1   14    2  928   15   30
##      8  4   14   13   22    5   19    5    4  797    3
##      9  0    1    3   10  103   43    0   49   56  888
```

1.2.6 Discussion

```
par(mfrow = c(3, 3))
plot_mistake(actual = 6, predicted = 4)
```

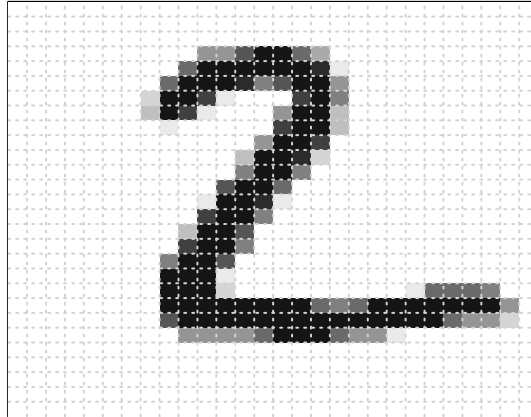


```
mnist_obs_to_check = 2
predict(mnist_rf, mnist_tst_x[mnist_obs_to_check, ], type = "prob")[1, ]

##      0      1      2      3      4      5      6      7      8      9
## 0.09 0.03 0.25 0.14 0.02 0.14 0.25 0.01 0.05 0.02
mnist_tst_y[mnist_obs_to_check]

## [1] 2
## Levels: 0 1 2 3 4 5 6 7 8 9
```

```
show_digit(mnist_tst_x[mnist_obs_to_check, ])
```



1.3 Clustering: NBA Players

1.3.1 Background

- https://www.youtube.com/watch?v=cuLprHh_BRg
- https://www.youtube.com/watch?v=1FBwSO_1Mb8
- https://www.basketball-reference.com/leagues/NBA_2019.html

1.3.2 Data

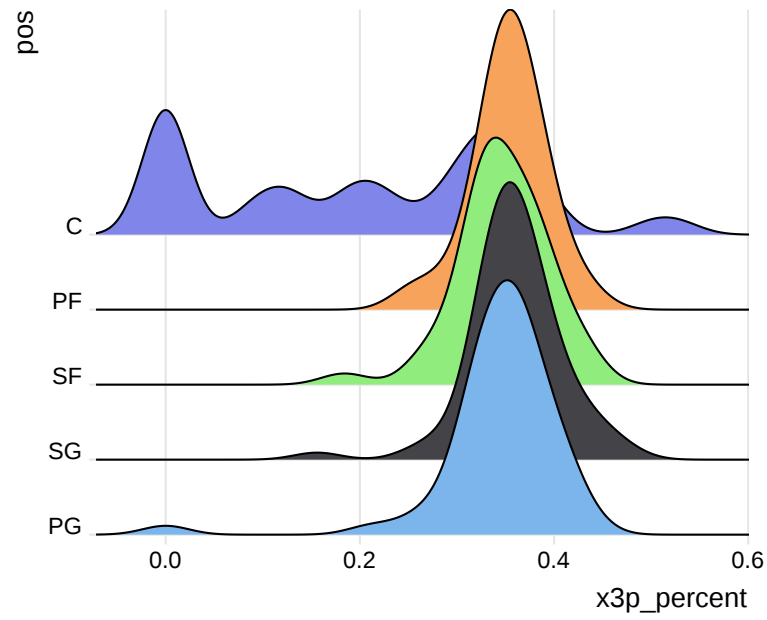
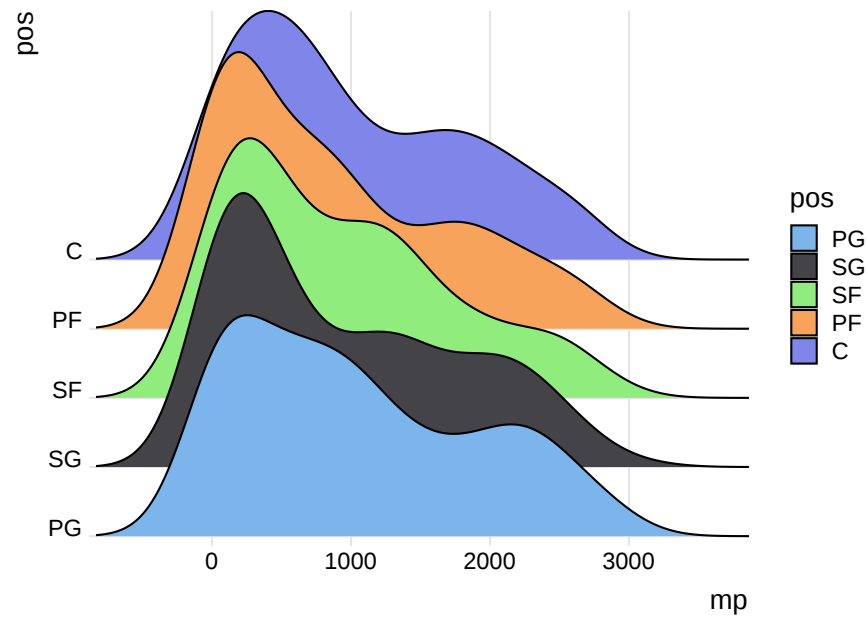
- https://www.basketball-reference.com/leagues/NBA_2019_totals.html
- https://www.basketball-reference.com/leagues/NBA_2019_per_minute.html

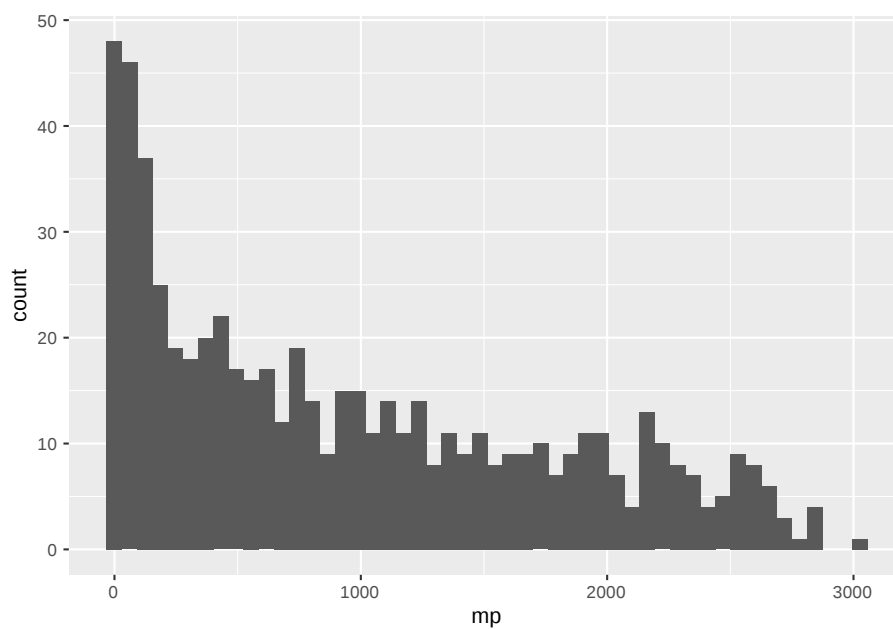
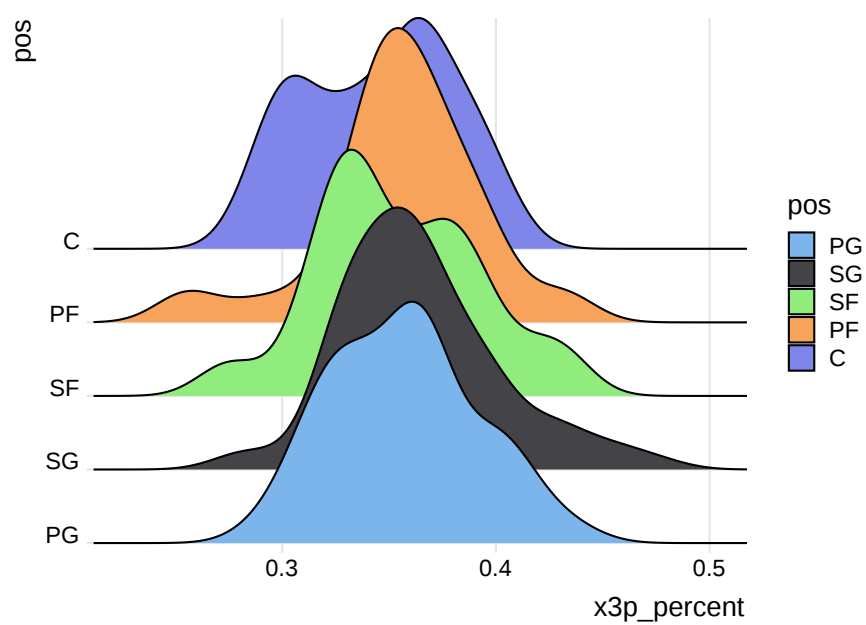
- https://www.basketball-reference.com/leagues/NBA_2019_per_poss.html
- https://www.basketball-reference.com/leagues/NBA_2019_advanced.html

```
nba = scrape_nba_season_player_stats()
nba$pos = factor(nba$pos, levels = c("PG", "SG", "SF", "PF", "C"))
```

```
## # A tibble: 100 x 93
##   player_team pos      age tm      g      gs      mp      fg      fga fg_percent
##   <chr>         <fct> <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Álex Abrin~ SG      25 OKC      31      2    588     56    157     0.357
## 2 Quincy Acy~ PF      28 PHO      10      0    123      4     18     0.222
## 3 Jaylen Ada~ PG      22 ATL      34      1    428     38    110     0.345
## 4 Steven Ada~ C       25 OKC      80     80   2669    481    809     0.595
## 5 Bam Adebay~ C       21 MIA      82     28   1913    280    486     0.576
## 6 Deng Adel ~ SF      21 CLE      19      3    194     11     36     0.306
## 7 DeVaughn A~ SG      25 DEN       7      0     22      3     10      0.3
## 8 LaMarcus A~ C       33 SAS      81     81   2687    684   1319     0.519
## 9 Rawle Alki~ SG      21 CHI      10      1    120     13     39     0.333
## 10 Grayson Al~ SG      23 UTA      38      2    416     67    178     0.376
## # ... with 90 more rows, and 83 more variables: x3p <dbl>, x3pa <dbl>,
## #   x3p_percent <dbl>, x2p <dbl>, x2pa <dbl>, x2p_percent <dbl>,
## #   e_fg_percent <dbl>, ft <dbl>, fta <dbl>, ft_percent <dbl>, orb <dbl>,
## #   drb <dbl>, trb <dbl>, ast <dbl>, stl <dbl>, blk <dbl>, tov <dbl>,
## #   pf <dbl>, pts <dbl>, fg_pm <dbl>, fga_pm <dbl>, fg_percent_pm <dbl>,
## #   x3p_pm <dbl>, x3pa_pm <dbl>, x3p_percent_pm <dbl>, x2p_pm <dbl>,
## #   x2pa_pm <dbl>, x2p_percent_pm <dbl>, ft_pm <dbl>, fta_pm <dbl>,
## #   ft_percent_pm <dbl>, orb_pm <dbl>, drb_pm <dbl>, trb_pm <dbl>,
## #   ast_pm <dbl>, stl_pm <dbl>, blk_pm <dbl>, tov_pm <dbl>, pf_pm <dbl>,
## #   pts_pm <dbl>, fg_pp <dbl>, fga_pp <dbl>, fg_percent_pp <dbl>,
## #   x3p_pp <dbl>, x3pa_pp <dbl>, x3p_percent_pp <dbl>, x2p_pp <dbl>,
## #   x2pa_pp <dbl>, x2p_percent_pp <dbl>, ft_pp <dbl>, fta_pp <dbl>,
## #   ft_percent_pp <dbl>, orb_pp <dbl>, drb_pp <dbl>, trb_pp <dbl>,
## #   ast_pp <dbl>, stl_pp <dbl>, blk_pp <dbl>, tov_pp <dbl>, pf_pp <dbl>,
## #   pts_pp <dbl>, o_rtg_pp <dbl>, d_rtg_pp <dbl>, per <dbl>,
## #   ts_percent <dbl>, x3p_ar <dbl>, f_tr <dbl>, orb_percent <dbl>,
## #   drb_percent <dbl>, trb_percent <dbl>, ast_percent <dbl>,
## #   stl_percent <dbl>, blk_percent <dbl>, tov_percent <dbl>,
## #   usg_percent <dbl>, ows <dbl>, dws <dbl>, ws <dbl>, ws_48 <dbl>,
## #   obpm <dbl>, dbpm <dbl>, bpm <dbl>, vorp <dbl>
```

1.3.3 EDA





```
nba_for_clustering = nba %>%
  filter(mp > 2000) %>%
  column_to_rownames("player_team") %>%
  select(-pos, -tm)
```

1.3.4 Modeling

```

set.seed(42)

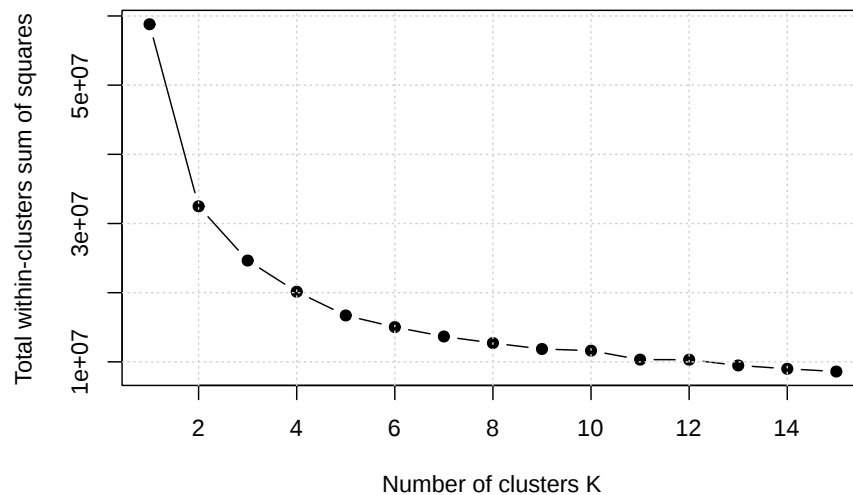
# function to compute total within-cluster sum of square
wss = function(k, data) {
  kmeans(x = data, centers = k, nstart = 10)$tot.withinss
}

# Compute and plot wss for k = 1 to k = 15
k_values = 1:15

# extract wss for 2-15 clusters
wss_values = map_dbl(k_values, wss, data = nba_for_clustering)

plot(k_values, wss_values,
     type="b", pch = 19, frame = TRUE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
grid()

```

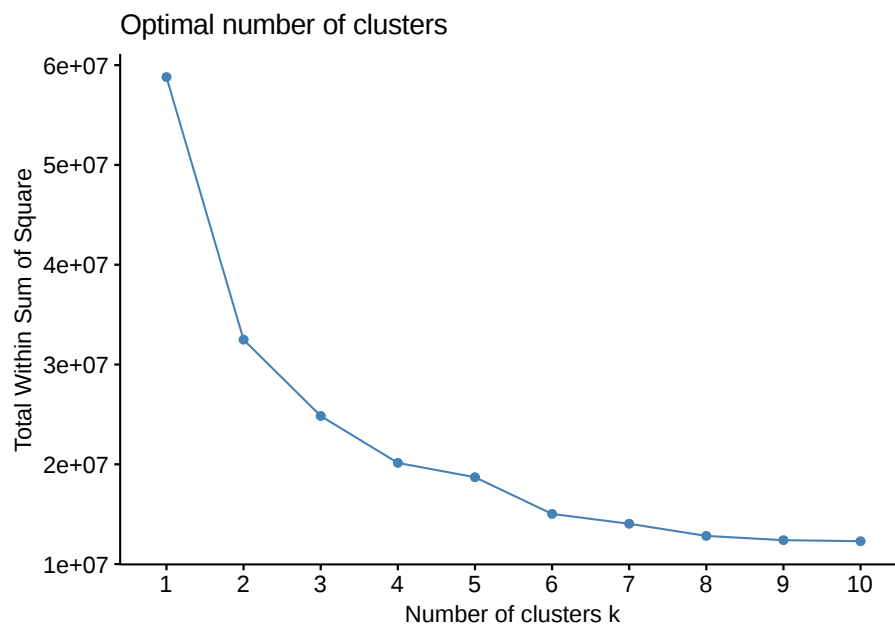


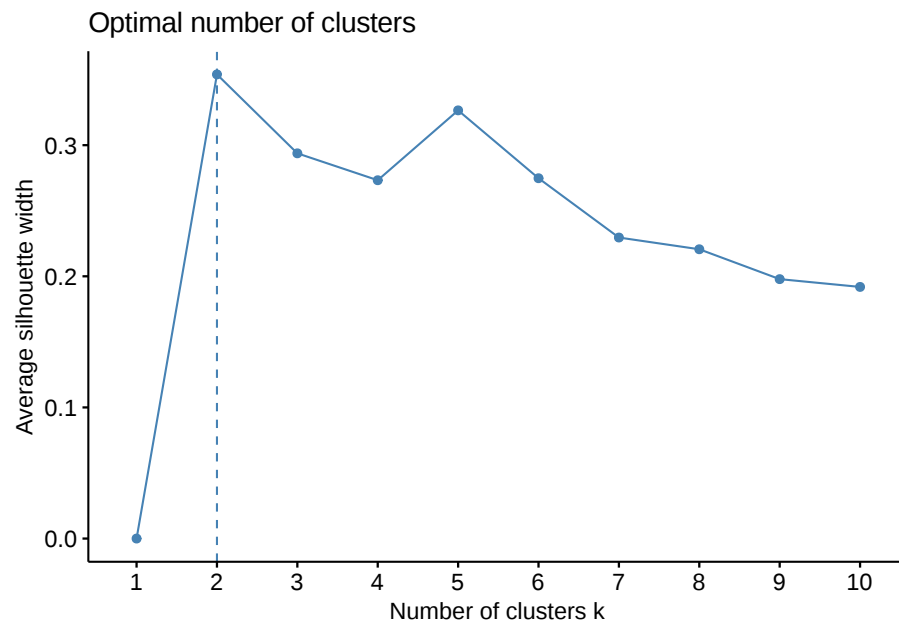
- TODO: K-Means likes clusters of roughly equal size.
- TODO: <http://varianceexplained.org/r/kmeans-free-lunch/>


```
nba_hc = hclust(dist(nba_for_clustering))  
nba_hc_clust = cutree(nba_hc, k = 5)  
table(nba_hc_clust)
```

```
## nba_hc_clust  
## 1 2 3 4 5  
## 38 13 28 11 1
```

1.3.5 Model Evaluation

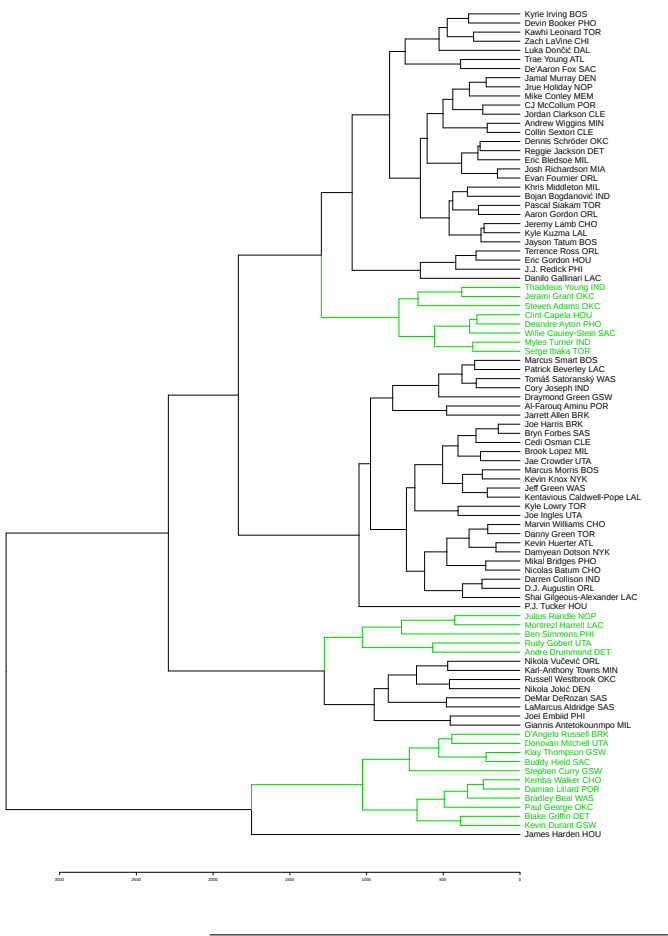




27



1.3.6 Discussion



Chapter 2

Computing

This is not a book about R. It is however, a book that uses R. Because of this, you will need to be familiar with R. The text will point out some thing about R along the way, but some previous study of R is necessary.

The following (freely available) readings are highly recommended:

- [Hands-On Programming with R](#) - *Garrett Grolemund*
 - If you have never used R or RStudio before, Part 1, Chapters 1 - 3, will be useful.
- [R for Data Science](#) - *Garrett Grolemund, Hadley Wickham*
 - This book helps getting you up to speed working with data in R. While it is a lot of reading, Chapters 1 - 21 are highly recommended.
- [Advanced R](#) - *Hadley Wickham*
 - Part I, Chapters 1 - 8, of this book will help create a mental model for working with R. These chapters are not an easy read, so they should be returned to often. (Chapter 2 could be safely skipped for our purposes, but is important if you will use R in the long term.)

If you are a UIUC student who took the course STAT 420, the first six chapters of that book could serve as a nice refresher.

- [Applied Statistics with R](#) - *David Dalpiaz*
-

2.1 Resources

The following resources are more specific or more advanced, but could still prove to be useful.

2.1.1 R

- [Efficient R programming](#)
- [R Programming for Data Science](#)
- [R Graphics Cookbook](#)
- [Modern Dive](#)
- [The tidyverse Website](#)
 - [dplyr Website](#)
 - [readr Website](#)
 - [tibble Website](#)
 - [forcats Website](#)

2.1.2 RStudio

- [RStudio IDE Cheatsheet](#)
- [RStudio Resources](#)

2.1.3 R Markdown

- [R Markdown Cheatsheet](#)
- [R Markdown: The Definitive Guide](#) - *Yihui Xie, J. J. Allaire, Garrett Grolemund*
- [R4DS R Markdown Chapter](#)

2.1.3.1 Markdown

- [Daring Fireball - Markdown: Basics](#)
 - [GitHub - Mastering Markdown](#)
 - [CommonMark](#)
-

2.2 BSL Idioms

Things here supercede everythign above.

2.2.1 Reference Style

- [tidyverse Style Guide](#)

2.2.2 BSL Style Overrides

- TODO: = instead of <-
 - <http://thecoatlessprofessor.com/programming/an-opinionated-tale-of-why-you-should-replace---with-/>
- TODO: never use T or F, only TRUE or FALSE

```
FALSE == TRUE
```

```
## [1] FALSE
```

```
F == TRUE
```

```
## [1] FALSE
```

```
F = TRUE
```

```
F == TRUE
```

```
## [1] TRUE
```

- TODO: never ever ever use `attach()`
- TODO: never ever ever use `<<-`
- TODO: never ever ever use `setwd()` or set a working directory some other way
- TODO: a newline before and after any chunk
- TODO: use headers appropriately! (short names, good structure)
- TODO: never ever ever put spaces in filenames. use `-`. (others will use `_`)
- TODO: load all needed packages at the beginning of an analysis in a single chunk (TODO: pros and cons of this approach)
- TODO: one plot per chunk! no other printed output

Be consistent...

- with yourself!
- with your group!
- with your organization!

```
set.seed(1337);mu=10;sample_size=50;samples=100000;
xBars=rep(0, samples)
for(i in 1:samples)
{
  xBars[i]=mean(rpois(sample_size,lambda = mu))}
xBar_hist=hist(xBars,breaks=50,main="Histogram of Sample Means",xlab="Sample Means",col="darkorange")
mean(xBars>mu-2*sqrt(mu)/sqrt(sample_size)&xBars<mu+2*sqrt(mu)/sqrt(sample_size))
```

2.2.3 Objects and Functions

To understand computations in R, two slogans are helpful:

- Everything that exists is an object.
- Everything that happens is a function call.

— John Chambers

2.2.4 Print versus Return

```
cars_mod = lm(dist ~ speed, data = cars)

summary(cars_mod)

##
## Call:
## lm(formula = dist ~ speed, data = cars)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -29.069  -9.525  -2.272   9.215  43.201
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.5791     6.7584  -2.601  0.0123 *
## speed        3.9324     0.4155   9.464 1.49e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.38 on 48 degrees of freedom
## Multiple R-squared:  0.6511, Adjusted R-squared:  0.6438
## F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12

is.list(summary(cars_mod))

## [1] TRUE

names(summary(cars_mod))

## [1] "call"          "terms"          "residuals"      "coefficients"
## [5] "aliased"        "sigma"          "df"             "r.squared"
## [9] "adj.r.squared" "fstatistic"     "cov.unscaled"

str(summary(cars_mod))

## List of 11
## $ call      : language lm(formula = dist ~ speed, data = cars)
## $ terms     :Classes 'terms', 'formula' language dist ~ speed
## ..- attr(*, "variables")= language list(dist, speed)
## ..- attr(*, "factors")= int [1:2, 1] 0 1
## ..- attr(*, "dimnames")=List of 2
## .. $ : chr [1:2] "dist" "speed"
## .. $ : chr "speed"
```



```
## .. ..- attr(*, "term.labels")= chr "speed"
## .. ..- attr(*, "order")= int 1
## .. ..- attr(*, "intercept")= int 1
## .. ..- attr(*, "response")= int 1
## .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
## .. ..- attr(*, "predvars")= language list(dist, speed)
## .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
## .. ..- attr(*, "names")= chr [1:2] "dist" "speed"
## $ residuals      : Named num [1:50] 3.85 11.85 -5.95 12.05 2.12 ...
## ..- attr(*, "names")= chr [1:50] "1" "2" "3" "4" ...
## $ coefficients   : num [1:2, 1:4] -17.579 3.932 6.758 0.416 -2.601 ...
## ..- attr(*, "dimnames")=List of 2
## .. ..$ : chr [1:2] "(Intercept)" "speed"
## .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
## $ aliased        : Named logi [1:2] FALSE FALSE
## ..- attr(*, "names")= chr [1:2] "(Intercept)" "speed"
## $ sigma          : num 15.4
## $ df             : int [1:3] 2 48 2
## $ r.squared       : num 0.651
## $ adj.r.squared   : num 0.644
## $ fstatistic      : Named num [1:3] 89.6 1 48
## ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
## $ cov.unscaled    : num [1:2, 1:2] 0.19311 -0.01124 -0.01124 0.00073
## ..- attr(*, "dimnames")=List of 2
## .. ..$ : chr [1:2] "(Intercept)" "speed"
## .. ..$ : chr [1:2] "(Intercept)" "speed"
## - attr(*, "class")= chr "summary.lm"

# RStudio only
View(summary(cars_mod))
```

2.2.5 Help

- TODO: ?, google, stack overflow, (office hours, course forums)

2.2.6 Keyboard Shortcuts

- TODO: copy-paste, switch program, switch tab, etc...
- TODO: TAB!!!
- TODO: new chunk!
- TODO: style!
- TODO: keyboard shortcut for keyboard shortcut

2.3 Common Issues

- TODO: cannot find function called ""
-