A1844790 - Dylan Rohan

Assessment 2 - Scaffolded Case Study and Data

Data Clean

Step 1:

```r
# Reading in data
affairs_df <- read.csv("affairs.csv")
affairs_df <- as_tibble(affairs_df)

# Child, Religious, Sex, and Rate are ordinal/nomial so it makes sense to sto
re them as factors (will address education and occupation later).
affairs_df$religious <- as_factor(affairs_df$religious)
affairs_df$rate <- as_factor(affairs_df$rate)
affairs_df$child <- as_factor(affairs_df$child)
affairs_df$sex <- as_factor(affairs_df$sex)


# Printing first 6 rows
knitr::kable(head(affairs_df, 6))
```

affair	sex	age	ym	child	religious	education	occupation	rate
0	male	37	10.00	no	3	18	7	4
0	female	27	4.00	no	4	14	6	4
0	female	32	15.00	yes	1	12	1	4
0	male	57	15.00	yes	5	18	6	5
0	male	22	0.75	no	2	17	6	3
0	female	32	1.50	no	2	17	5	5

Step 2:

The outcome variable is ‘*affair*’ – with ‘0’ indicating they’d never had an affair and a ‘1’ indicating they have had at least one affair.,

The predictor variables are the gender (*sex*), age (*age*), number of years of marriage (*ym*), whether they have children (*child*), how devout they consider themselves to be to their religion (*religious*), their level of education (*education*), their occupation according to the Hollinghead classification (*occupation*), and how happy they are in their marriage (*rate*).

Step 3:

```r
# Skim() is great for an overview but very cumbersome to view
skim(affairs_df)
```

Data summary

Name	affairs_df
Number of rows	601
Number of columns	9
_______________________	
Column type frequency:	
factor	4
numeric	5
________________________	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
sex	0	1	FALSE	2	fem: 315, mal: 286
child	0	1	FALSE	2	yes: 430, no: 171
religious	0	1	FALSE	5	4: 190, 2: 164, 3: 129, 5: 70
rate	0	1	FALSE	5	5: 232, 4: 194, 3: 93, 2: 66

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
affair	0	1	0.25	0.43	0.00	0	0	0	1	▇▁▁▁▂
age	0	1	32.49	9.29	17.50	27	32	37	57	▃▇▂▂▁
ym	0	1	8.18	5.57	0.12	4	7	15	15	▇▆▃▃▇
education	0	1	16.17	2.40	9.00	14	16	18	20	▁▂▇▇▇
occupation	0	1	4.19	1.82	1.00	3	5	6	7	▅▂▂▇▆

There are currently:

- No missing values
- 601 rows (but three will be removed below)
- The variables have been read in correctly, but data types will be altered in a few steps time to better reflect the information they provide. There is also some cleaning to be done.

Double check and data cleaning:

```r
# There are no NA values
sum(is.na(affairs_df))
```

```
# Clear

# Skim doesn't show you what values are present (spelling mistakes, invalid v
alues, etc.)
unique(affairs_df$affair)
unique(affairs_df$sex)
unique(affairs_df$age)
unique(affairs_df$ym)
unique(affairs_df$child)
unique(affairs_df$religious)
unique(affairs_df$education)
unique(affairs_df$occupation)
unique(affairs_df$rate)
# No errors identified
# Clear

# Skim doesn't show you conditional errors
# Does their age and years married make sense?
filter(affairs_df, age <= 17.5)

a <- filter(affairs_df, age <= 17.5 & ym==10)
knitr::kable(head(a))
```

affair	sex	age	ym	child	religious	education	occupation	rate
0	female	17.5	10	no	4	14	4	5

A 17.5-year-old woman had been married for 10 years? Married since the age of 7.5?

This American magazine printed from 1967. The subject was highly religious so perhaps not unheard of, but the American census suggests the law requires you be at least 14 to marry (Hetzel and Cappetta 1971). Which would make this an unlawful marriage; or an outlier at a minimum. If the American census did not recognize the union, I see no reason why I should.

```
# remove unlawful child marriages
filter(affairs_df, (age-ym) < 14 )
clean <- affairs_df %>% slice(-c(which(affairs_df$age <= 17.5 & affairs_df$ym
==10, arr.ind=TRUE)))
# Removed

# Is their level of education feasibly possible?
filter(clean, age < 23, education == 20)
b <- filter(clean, age < education)
knitr::kable(head(b))
```

affair	sex	age	ym	child	religious	education	occupation	rate
0	male	17.5	1.50	yes	3	18	6	5
0	female	17.5	0.75	no	2	18	5	4

A1844790 - Dylan Rohan

You couldn't attend school from the womb in the 1960s to my knowledge. In the American system, children begin formal education aged 3 or 4, meaning you can't be younger than 23 and have a PhD without being some kind of savant. A savant is, by definition, extremely different from the population anyway. These occurrences will be excluded.

```
clean <- subset(clean, age>education)
# Removed

# checking for oddities between participant's education and occupation
filter(clean, occupation == 7, education < 15 )

## # A tibble: 5 x 9
##   affair sex      age    ym child religious education occupation rate
##    <int> <fct>  <dbl> <dbl> <fct> <fct>         <int>      <int> <fct>
## 1      0 female    47    15 yes   5                14          7 2
## 2      0 male      57    15 yes   2                14          7 2
## 3      0 male      52    15 yes   2                14          7 4
## 4      1 female    27    10 yes   4                12          7 3
## 5      1 male      32     7 yes   3                14          7 4

# ^highly skilled jobs, but did not have any tertiary education. But unions a
nd regulations might have been less stringent in the 1960s?
# Questionable, but clear
```

Three rows removed due to impossible or unlawful outliers considering the 1960s context. This took the initial 601 rows down to **598 rows**, the **9 columns** containing the observations for each subject are still present.

Step 4:

```
# Setting the binary values to yes/no factor values
clean$affair <- ifelse(clean$affair == 1, "yes", "no")
clean$affair <- as.factor(clean$affair)

# Counting values
sum(clean$affair == "yes")
## [1] 150

sum(clean$affair == "no")
## [1] 448

# No longer care about numeric information for education and occupation, conv
erting to factors
clean$education <- as.factor(clean$education)
clean$occupation <- as.factor(clean$occupation)
knitr::kable(head(clean))

#  affair sex      age    ym child religious education occupation rate
# <fct>  <fct>  <dbl> <dbl> <fct> <fct>     <fct>     <fct>      <fct>
```

affair	sex	age	ym	child	religious	education	occupation	rate
no	male	37	10.00	no	3	18	7	4
no	female	27	4.00	no	4	14	6	4
no	female	32	15.00	yes	1	12	1	4
no	male	57	15.00	yes	5	18	6	5
no	male	22	0.75	no	2	17	6	3
no	female	32	1.50	no	2	17	5	5

The tibble now has 150 'yes' values and 448 'no' values in place of their binary counterparts. All Character values have been converted to factors. Occupation and education variables concerned me because the context between each value doesn't increase linearly, it's exponential. The same can be said for the lifestyle that comes with each level of the Hollinghead classification system. To reflect the differences better, I am treating them as factors with levels (ordinal).

Step 5:

```
skim(clean)
```

Data summary

Name	clean
Number of rows	598
Number of columns	9
_______________________	
Column type frequency:	
factor	7
numeric	2
________________________	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
affair	0	1	FALSE	2	no: 448, yes: 150
sex	0	1	FALSE	2	fem: 313, mal: 285
child	0	1	FALSE	2	yes: 429, no: 169
religious	0	1	FALSE	5	4: 189, 2: 163, 3: 128, 5: 70
education	0	1	FALSE	7	14: 153, 16: 115, 18: 110, 17: 89
occupation	0	1	FALSE	7	5: 203, 6: 142, 1: 113, 4: 67
rate	0	1	FALSE	5	5: 230, 4: 193, 3: 93, 2: 66

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
age	0	1	32.56	9.25	17.50	27	32	37	57	▃▇▂▂▁
ym	0	1	8.20	5.57	0.12	4	7	15	15	▇▆▃▃▇

```
sum(clean$affair == "yes")
## [1] 150
sum(clean$child == "yes")
## [1] 429
mean(clean$age)
## [1] 32.56271
mean(as.integer(clean$religious))
## [1] 3.117057
```

1) How many people responded to having an affair? 150
2) How many responded to having children? 429
3) What is the mean age of respondents? 32.56
4) What is the mean response on the religious scale? 3.12 (Approximately 3)
5) Do you think you should normalize the numeric variables? Yes
6) Why/Why not? Because we have different ranges for each variable. Larger values will have more weighting but may not be better predictors. By normalizing we create uniform scales which will allow R to better interpret variable importance.

Exploratory Data Analysis

Step 1: female proportionality of response to affair

```
# For women responding No
count(filter(clean, affair == "no", sex == "female"))/count(filter(clean, aff
air == "no"))
##           n
## 1 0.5379464

# For women responding yes
count(filter(clean, affair == "yes", sex == "female"))/count(filter(clean, af
fair == "yes"))
##      n
## 1 0.48

# Added after tute:
CrossTable(clean$affair, clean$sex)
##              | clean$sex
## clean$affair |      male |    female | Row Total |
## -------------|-----------|-----------|-----------|
##           no |       207 |       241 |       448 |
##              |     0.199 |     0.181 |           |
##              |     0.462 |     0.538 |     0.749 |
##              |     0.726 |     0.770 |           |
##              |     0.346 |     0.403 |           |
## -------------|-----------|-----------|-----------|
##          yes |        78 |        72 |       150 |
##              |     0.593 |     0.540 |           |
##              |     0.520 |     0.480 |     0.251 |
##              |     0.274 |     0.230 |           |
##              |     0.130 |     0.120 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       285 |       313 |       598 |
##              |     0.477 |     0.523 |           |
## -------------|-----------|-----------|-----------|
```

- 74.9% of participants had **never** had an affair.
- 53.79% of participants that indicated they'd **never had** an affair were female
- 48% of participants that indicated they **have had** an affair were female.

Gender does not appear to be a great indicator of whether or not someone will have an affair, it's almost as likely as a coin flip.

Step 2: parenthood status proportionality of response to affair

```
# For parents responding yes
count(filter(clean, affair == "yes", child == "yes"))/count(filter(clean, aff
air == "yes"))
##      n
## 1 0.82
# For women responding no
count(filter(clean, affair == "no", child == "no"))/count(filter(clean, affai
r == "no"))
##           n
## 1 0.3169643

# Added after tute:
CrossTable(clean$affair, clean$child)
##              | clean$child
## clean$affair |        no |       yes | Row Total |
## -------------|-----------|-----------|-----------|
##           no |       142 |       306 |       448 |
##              |     1.871 |     0.737 |           |
##              |     0.317 |     0.683 |     0.749 |
##              |     0.840 |     0.713 |           |
##              |     0.237 |     0.512 |           |
## -------------|-----------|-----------|-----------|
##          yes |        27 |       123 |       150 |
##              |     5.588 |     2.201 |           |
##              |     0.180 |     0.820 |     0.251 |
##              |     0.160 |     0.287 |           |
##              |     0.045 |     0.206 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       169 |       429 |       598 |
##              |     0.283 |     0.717 |           |
## -------------|-----------|-----------|-----------|
```

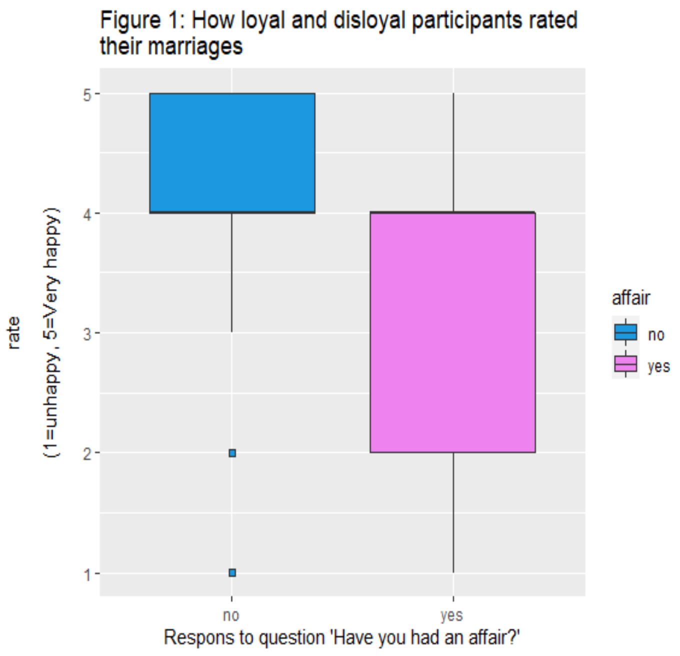
- 429 subjects had children, 169 subjects did not.
- 82% of people that **did have** an affair indicated they were parents.
- 31.70% of people that **did not have** an affair indicated they were childless.

In the 1960s, it appears having children increased the likely hood of having an affair. Perhaps due to the added economic, physical and mental stresses that family life would have entailed. Parental status appears to be a good predictor of whether or not someone will have an affair.

A1844790 - Dylan Rohan

Step 3:

```
ggplot(clean, aes(x = affair, y=as.integer(rate), fill=affair)) +
  geom_boxplot(outlier.shape = 22, outlier.fill="#1b98e0") +
  scale_fill_manual(values = c("#1b98e0", "violet"), breaks=waiver()) +
  ggtitle("Figure 1: How loyal and disloyal participants rated \ntheir marria
ges") +
  xlab("Respons to question 'Have you had an affair?'") +
  ylab("rate\n\n(1=unhappy, 5=Very happy)\n")
```

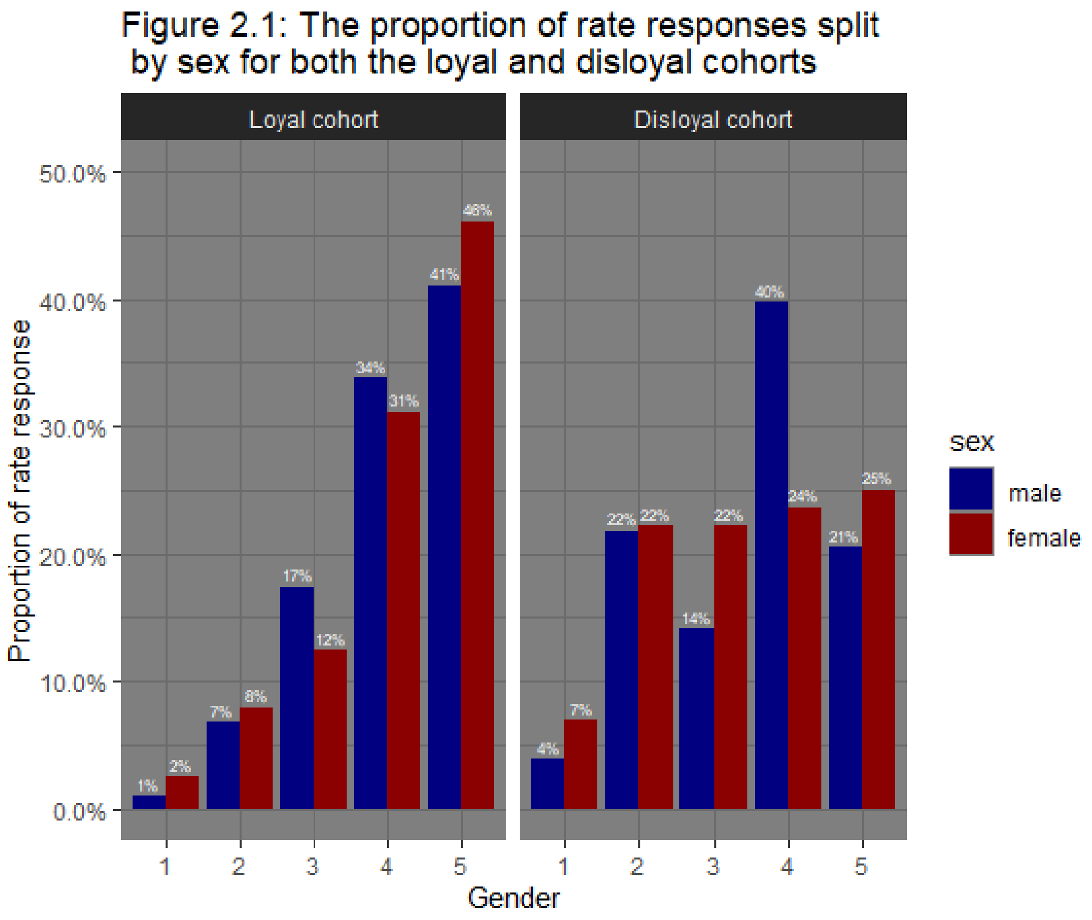


In both cases the median sits on a rate value of 4. However, the spread for the bulk of the data was much smaller for the loyal cohort (between 3 and 5 with a few outliers). Whereas the disloyal cohort had a spread from the minimum to the maximum rate values (1 to 5). It is difficult to comment on the reliability of 'rate' as a predictor because there is a good deal of whisker overlap between populations. However, the disloyal cohort skews to lower/unhappy values while the loyal cohort skews to higher/very happy values.

*consideration: We should also consider that there may be a sampling bias in the data. Magazine cost, accessibility, regularity of purchase, etc. could be used to scrutinize these effects but that is outside the scope of this assignment.

Step 4:

```
#..prop.. is also stat(prop) now
levels <- c('no'="Loyal cohort", 'yes'= "Disloyal cohort" )
ggplot(clean, aes(x = rate, group =  sex, fill = sex)) +
  geom_bar( aes(y = stat(prop)),
            stat = "count",
            position = position_dodge()) +
  scale_fill_manual(values = c("navy", "dark red"), breaks=waiver()) +
  geom_text( aes(label = scales::percent( accuracy = 1, (stat(prop))),
                 y = stat(prop)),
             stat = "count",
             vjust = -.5,
             position = position_dodge(0.9),
             size = 2,
             color = "white") +
  facet_grid(~affair, labeller = as_labeller(levels)) +
  ggtitle("Figure 2.1: The proportion of rate responses split\n by sex for bo
th the loyal and disloyal cohorts" ) +
  xlab("Gender") +
  ylab("Proportion of rate response") +
  scale_y_continuous(limits = c(0,0.5),
                     labels = scales::percent) +
  theme_dark()
```

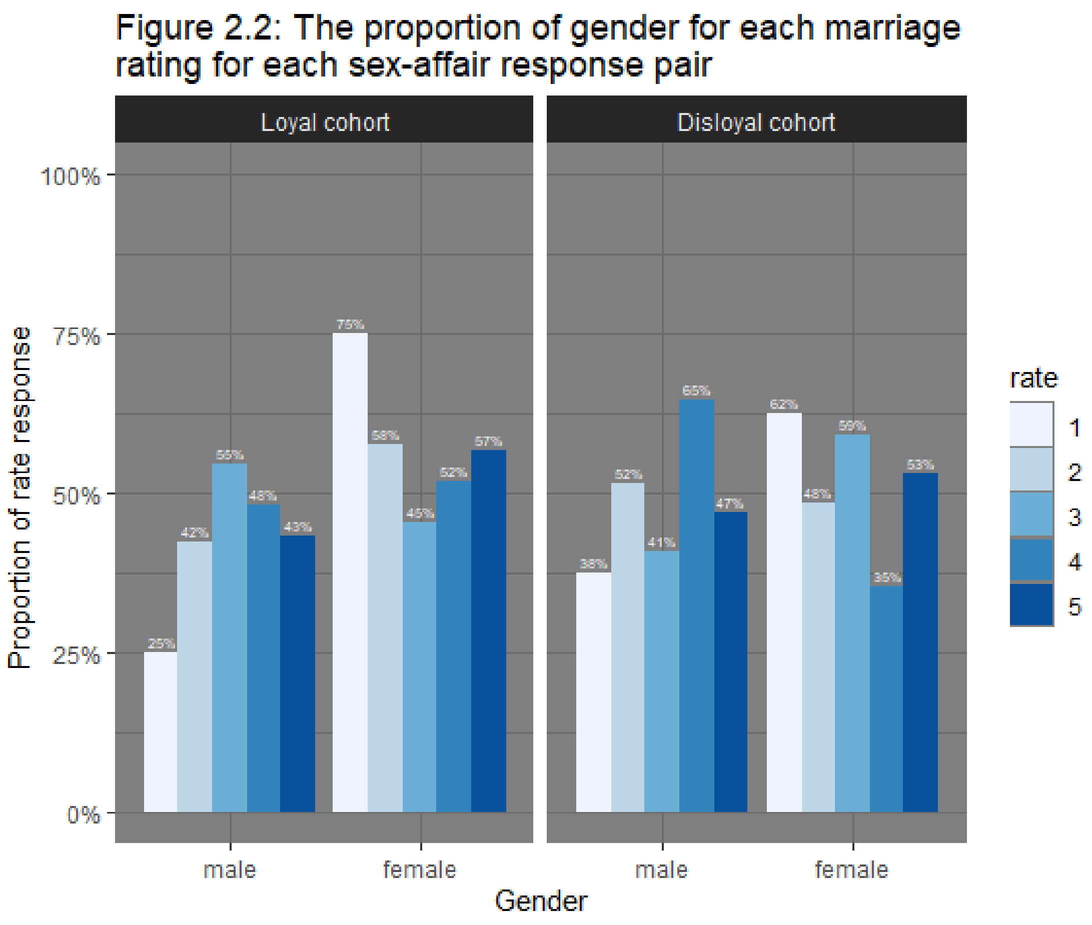


From Figure 2.1, we see that the disloyal males have a multi-modal distribution with most of the data present at level 4 - one less than the maximum happiness value. The disloyal women have a more uniform distribution, except for ratings of 1; indicating few disloyal women ranked themselves as the unhappiest they could be with their marriage. This is very different to what we see in the loyal cohort which is unimodal and left skewed.

A1844790 - Dylan Rohan

There is another way to interpret the question, so I've provided that below:

```
ggplot(clean,
       aes(x = sex, group =  rate, fill = rate)) +
  geom_bar( aes(y = stat(prop)),
           stat = "count",
           position = position_dodge()) +
  scale_fill_brewer()+
  geom_text( aes(label = scales::percent( accuracy = 1, (stat(prop))),
                 y = stat(prop)),
            stat = "count",
            vjust = -.5,
            position = position_dodge(.9),
            size = 1.8,
            color = "white") +
  scale_y_continuous(limits = c(0,1),
                     labels = scales::percent) +
  facet_grid(~affair,
             labeller = as_labeller(levels)) +
  ggtitle("Figure 2.2: The proportion of gender for each marriage \nrating fo
r each sex-affair response pair" ) +
  xlab("Gender") +
  ylab("Proportion of rate response ") +
  theme_dark()
```

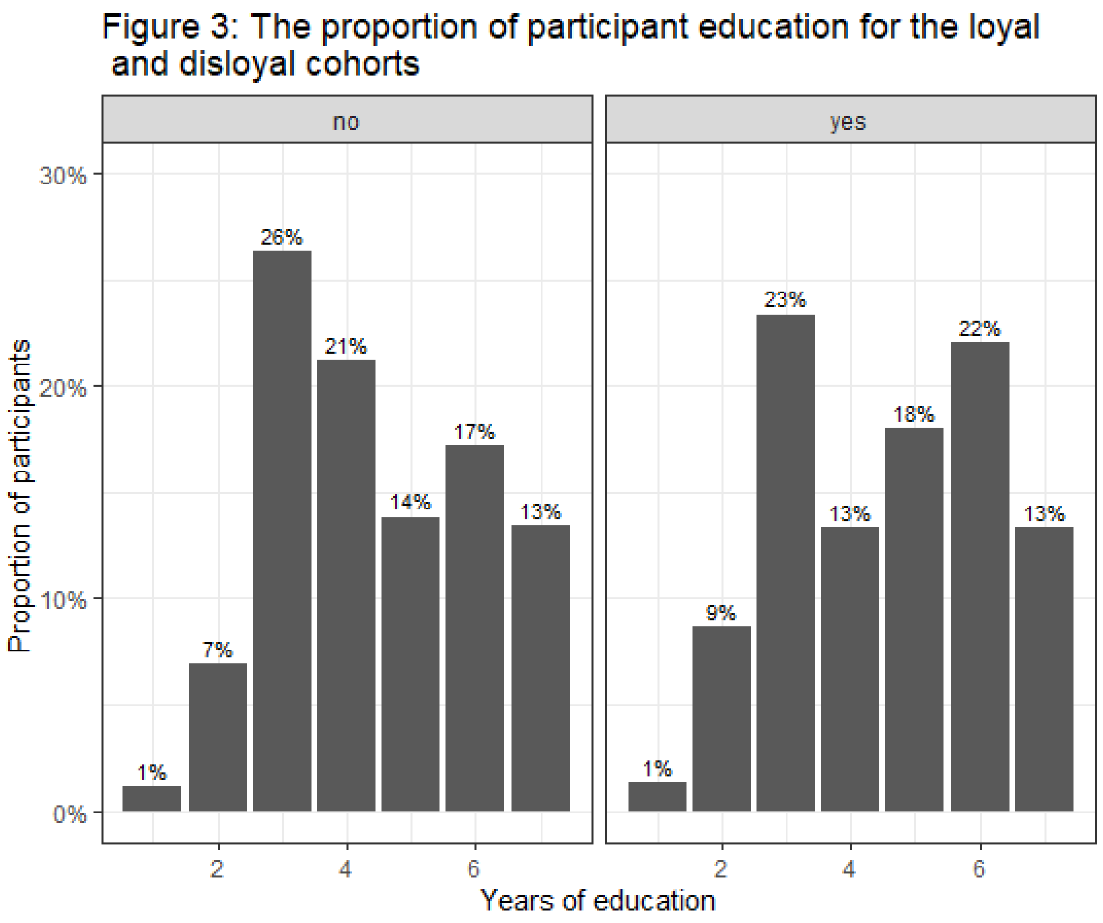


The population of men and population of women that indicated they'd been disloyal both have a multi-modal distribution. The males appear to have more of a left skew and the women more of a right skew; indicating that disloyal men were happier in their marriages than the disloyal women. Sadly, there were also larger proportions of dissatisfaction in loyal women as well.

*Consideration: The rate variable has a lot of unknown variation as participants have different tolerance and threshold levels. It's also worth mentioning that there is no record of why they had an affair or with what gender their affair was with. Perhaps a better variable would have been the cause, then we could have established a ranking of cause severity instead of a list of arbitrary tolerance/thresholds for each individual. It would still be arbitrary, but the ranking would be consistent thus minimizing variation between values.

Step 5:

```
ggplot(clean, aes(x=as.numeric(education), y=..prop..)) +
  geom_bar() +
  facet_wrap(~affair) +
  scale_y_continuous(limits = c(0,0.3),
                     labels = scales::percent) +
  geom_text( aes(label = scales::percent( accuracy = 1, (stat(prop))),
                 y = stat(prop)),
             stat = "count",
             vjust = -.5,
             position = position_dodge(width =0.9),
             size = 3)+
  ggtitle("Figure 3: The proportion of participant education for the loyal\n
and disloyal cohorts" ) +
  xlab("Years of education") +
  ylab("Proportion of participants ") +
  theme_bw()
```



There was a great deal of similarity between the cohorts, both were multimodal and many of the categories have identical proportions between cohorts. The loyal cohort appears to have a right skew when it comes to education, but the disloyal cohort appears to be spread more randomly. Years of education does not appear to hold much information regarding whether or not someone will have an affair.

*Consideration: Perhaps it would have been better to also indicate the discrepancy of education levels between the married couple and also investigate the education level of the person they had the affair with.

Something else to consider is that we cannot assume that someone is more intelligent because they've studied longer. The quality and type of studies, their genetic predispositions, etc. will determine intelligence. That being the case, we don't really have a true measure of education or intelligence, but a loose commentary on their socioeconomic status.

Split and Preprocess

step 1 and 2:

```r
# Setting seed for reproducibility
set.seed(1234)
# Indexing each row
index <- sample(1:nrow(clean))
repro_clean <-clean[index, ]

set.seed(1234)
# Creating training and test sets
clean_split <- initial_split(repro_clean)

clean_training <- training(clean_split)
knitr::kable(head(clean_training, 6))
```

affair	sex	age	ym	child	religious	education	occupation	rate
no	female	37	7.0	no	4	18	5	5
yes	male	22	1.5	no	2	12	3	3
yes	male	27	1.5	yes	3	17	5	4
no	female	52	15.0	yes	5	12	1	3
no	female	27	7.0	yes	2	12	1	2
no	female	22	1.5	no	2	14	1	5

```r
# Setting seed for reproducibility
set.seed(1234)
clean_testing <- testing(clean_split)
knitr::kable(head(clean_testing, 6))
```

affair	sex	age	ym	child	religious	education	occupation	rate
no	female	27	4.0	yes	4	12	1	5
no	male	57	15.0	yes	5	20	6	5
no	female	22	1.5	no	3	16	5	5

affair	sex	age	ym	child	religious	education	occupation	rate
no	male	37	15.0	yes	4	20	6	5
no	male	47	15.0	yes	4	16	6	4
no	male	22	4.0	yes	4	14	2	4

```
nrow(clean_training)

## [1] 448

nrow(clean_testing)

## [1] 150
```

There are 448 rows in the training set and 150 in the testing set.

Step 3:

The step_downsample() function is used to remove rows in a data set to ensure levels are interpreted equally. If you have strings as nominal values in your data set, this function will prevent them from being interpreted as ordinal. R interprets levels based on the ASCII, so there are inadvertent levels present in all our qualitative predictors as a result of setting them as factors:

```
levels(clean$religious) #"1" "2" "3" "4" "5"

levels(clean$rate)      #"1" "2" "3" "4" "5"

levels(clean$child)     #"no"  "yes"

levels(clean$sex)       #"male"   "female"

levels(clean$education) # "9"  "12" "14" "16" "17" "18" "20"

levels(clean$occupation)# "1" "2" "3" "4" "5" "6" "7"
```

Step 4:

```
# Preparing recipe
affair_recipe <- recipe(affair ~ ., data = clean_training) %>%
  themis::step_downsample(affair) %>%
  step_dummy(sex, child, religious, rate, education, occupation) %>% #for nom
inal data, I nhindsight should have used all_nominal()
  step_normalize(all_predictors())%>%
  prep()

affair_recipe
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          8
##
## Training data contained 448 data points and no missing data.
##
## Operations:
##
## $terms
## <list_of<quosure>>
##
## [[1]]
## <quosure>
## expr: ^affair
## env:  0x0000000020735c10
##
##
## $under_ratio
## [1] 1
##
## $ratio
## [1] NA
##
## $role
## [1] NA
##
## $trained
## [1] TRUE
##
## $column
## [1] "affair"
##
## $target
## [1] 117
##
## $skip
## [1] TRUE
##
## $id
## [1] "downsample_pdHMv"
##
## $seed
## [1] 84294
##
## $id
## [1] "downsample_pdHMv"
```

```
##
## attr(,"class")
## [1] "step_downsample" "step"
## Dummy variables from sex, child, religious, rate, education, occupation [t
rained]
## Centering and scaling for age, ym, sex_female, child_yes, religious_X2, r.
.. [trained]
```

Step 5:

```
# Preprocessing the training set
affair_training_prepro <- affair_recipe %>%
  juice()           # Using clean_training data from recipe
affair_training_prepro

## # A tibble: 234 x 25
##        age     ym affair sex_female child_yes religious_X2 religious_X3
##      <dbl>  <dbl> <fct>       <dbl>     <dbl>        <dbl>        <dbl>
##  1 -0.0703 -0.811 no         -1.03     -1.74        -0.619       -0.586
##  2  0.491   0.270 no          0.964     0.573       -0.619       -0.586
##  3 -0.632  -1.51  no         -1.03     -1.74        -0.619        1.70
##  4  2.74    1.17  no         -1.03      0.573       -0.619       -0.586
##  5 -1.19   -1.26  no          0.964    -1.74        -0.619        1.70
##  6 -0.0703  0.270 no          0.964     0.573       -0.619       -0.586
##  7 -0.632  -1.40  no          0.964    -1.74        -0.619       -0.586
##  8  1.05    1.17  no         -1.03      0.573        1.61        -0.586
##  9 -0.632  -0.271 no          0.964     0.573       -0.619       -0.586
## 10 -1.19   -1.46  no          0.964    -1.74        -0.619       -0.586
## # ... with 224 more rows, and 18 more variables

#Preprocessing the testing set
affair_test_prepro <- affair_recipe %>%
  bake(clean_testing)
affair_test_prepro

## # A tibble: 150 x 25
##       age     ym affair sex_female child_yes religious_X2 religious_X3
##     <dbl>  <dbl> <fct>       <dbl>     <dbl>        <dbl>        <dbl>
##  1 -0.632 -0.811 no          0.964     0.573       -0.619       -0.586
##  2  2.74   1.17  no         -1.03      0.573       -0.619       -0.586
##  3 -1.19  -1.26  no          0.964    -1.74        -0.619        1.70
##  4  0.491  1.17  no         -1.03      0.573       -0.619       -0.586
##  5  1.61   1.17  no         -1.03      0.573       -0.619       -0.586
##  6 -1.19  -0.811 no         -1.03      0.573       -0.619       -0.586
##  7  2.74   1.17  no         -1.03      0.573       -0.619       -0.586
##  8  0.491  1.17  no         -1.03      0.573       -0.619       -0.586
##  9 -1.19  -1.40  no          0.964    -1.74         1.61        -0.586
## 10  0.491 -0.811 no         -1.03      0.573       -0.619       -0.586
## # ... with 140 more rows, and 18 more variables
```

A1844790 - Dylan Rohan

Step 6:

```
skim(affair_training_prepro)
```

Data summary

Name	affair_training_prepro
Number of rows	234
Number of columns	25
_______________________	
Column type frequency:	
factor	1
numeric	24
________________________	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
affair	0	1	FALSE	2	no: 117, yes: 117

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
age	0	1	0	1	-1.70	-0.63	-0.07	0.49	2.74	▃▇▃▂▁
ym	0	1	0	1	-1.51	-0.81	-0.27	1.17	1.17	▃▃▃▃▇
sex_female	0	1	0	1	-1.03	-1.03	0.96	0.96	0.96	▇▁▁▁▇
child_yes	0	1	0	1	-1.74	0.57	0.57	0.57	0.57	▂▁▁▁▇
religious_X2	0	1	0	1	-0.62	-0.62	-0.62	1.61	1.61	▇▁▁▁▂
religious_X3	0	1	0	1	-0.59	-0.59	-0.59	1.70	1.70	▇▁▁▁▂
religious_X4	0	1	0	1	-0.59	-0.59	-0.59	1.68	1.68	▇▁▁▁▂
religious_X5	0	1	0	1	-0.33	-0.33	-0.33	-0.33	3.02	▇▁▁▁▁
rate_X2	0	1	0	1	-0.45	-0.45	-0.45	-0.45	2.20	▇▁▁▁▂
rate_X3	0	1	0	1	-0.43	-0.43	-0.43	-0.43	2.34	▇▁▁▁▂
rate_X4	0	1	0	1	-0.69	-0.69	-0.69	1.44	1.44	▇▁▁▁▃
rate_X5	0	1	0	1	-0.69	-0.69	-0.69	1.44	1.44	▇▁▁▁▃
education_X12	0	1	0	1	-0.24	-0.24	-0.24	-0.24	4.11	▇▁▁▁▁
education_X14	0	1	0	1	-0.59	-0.59	-0.59	1.70	1.70	▇▁▁▁▂
education_X16	0	1	0	1	-0.45	-0.45	-0.45	-0.45	2.20	▇▁▁▁▂
education_X17	0	1	0	1	-0.45	-0.45	-0.45	-0.45	2.20	▇▁▁▁▂
education_X18	0	1	0	1	-0.49	-0.49	-0.49	-0.49	2.02	▇▁▁▁▂
education_X20	0	1	0	1	-0.40	-0.40	-0.40	-0.40	2.46	▇▁▁▁▁
occupation_X2	0	1	0	1	-0.15	-0.15	-0.15	-0.15	6.75	▇▁▁▁▁
occupation_X3	0	1	0	1	-0.34	-0.34	-0.34	-0.34	2.95	▇▁▁▁▁
occupation_X4	0	1	0	1	-0.40	-0.40	-0.40	-0.40	2.51	▇▁▁▁▁
occupation_X5	0	1	0	1	-0.72	-0.72	-0.72	1.38	1.38	▇▁▁▁▃
occupation_X6	0	1	0	1	-0.51	-0.51	-0.51	-0.51	1.94	▇▁▁▁▂
occupation_X7	0	1	0	1	-0.18	-0.18	-0.18	-0.18	5.68	▇▁▁▁▁

There are 117 ‘yes’ and 117 ‘no’ values from the down-sampling, thus removing any unwanted weighting. The factor variables have been converted to numerical dummy variables (categorical variables now indicate the absence (0) or presence (1) of an influence). The step_normalize() function uses the training data to make an estimate of the mean and standard deviation. It subtracts the mean from the data to center it, and then divides the values by the standard deviation in order to scale it; effectively an estimated z-score. The mean is effectively on 0 and the standard deviation is now exactly 1 from normalizing the data.

Tune and Fit a Model

Step 1:

```
# Model specification
model_spec <- nearest_neighbor(mode = "classification",
                               engine = "kknn",
                               neighbors = tune())

# fitting model
affair_knn <- model_spec %>%
  fit(affair~., data = affair_training_prepro)
```

Step 2:

```
# Setting seed for reproducibility
set.seed(1234)
index <- sample(1:nrow(affair_training_prepro))
prepro_training <- affair_training_prepro[index, ]
affair_cv <- vfold_cv( data = prepro_training, v = 5, starta=affair)
affair_cv

## #  5-fold cross-validation
## # A tibble: 5 x 2
##   splits           id
##   <list>           <chr>
## 1 <split [187/47]> Fold1
## 2 <split [187/47]> Fold2
## 3 <split [187/47]> Fold3
## 4 <split [187/47]> Fold4
## 5 <split [188/46]> Fold5
```

Step 3:

```
k_grid <- grid_regular(neighbors(c(5,75)),levels = 25)
k_grid

## # A tibble: 25 x 1
##    neighbors
##        <int>
##  1         5
##  2         7
##  3        10
##  4        13
##  5        16
##  6        19
##  7        22
##  8        25
##  9        28
## 10        31
## # ... with 15 more rows
```

Step 4:

```
tune_k <- tune::tune_grid(model_spec,
                          preprocess = affair~.,
                          resamples = affair_cv,
                          grid = k_grid)
```

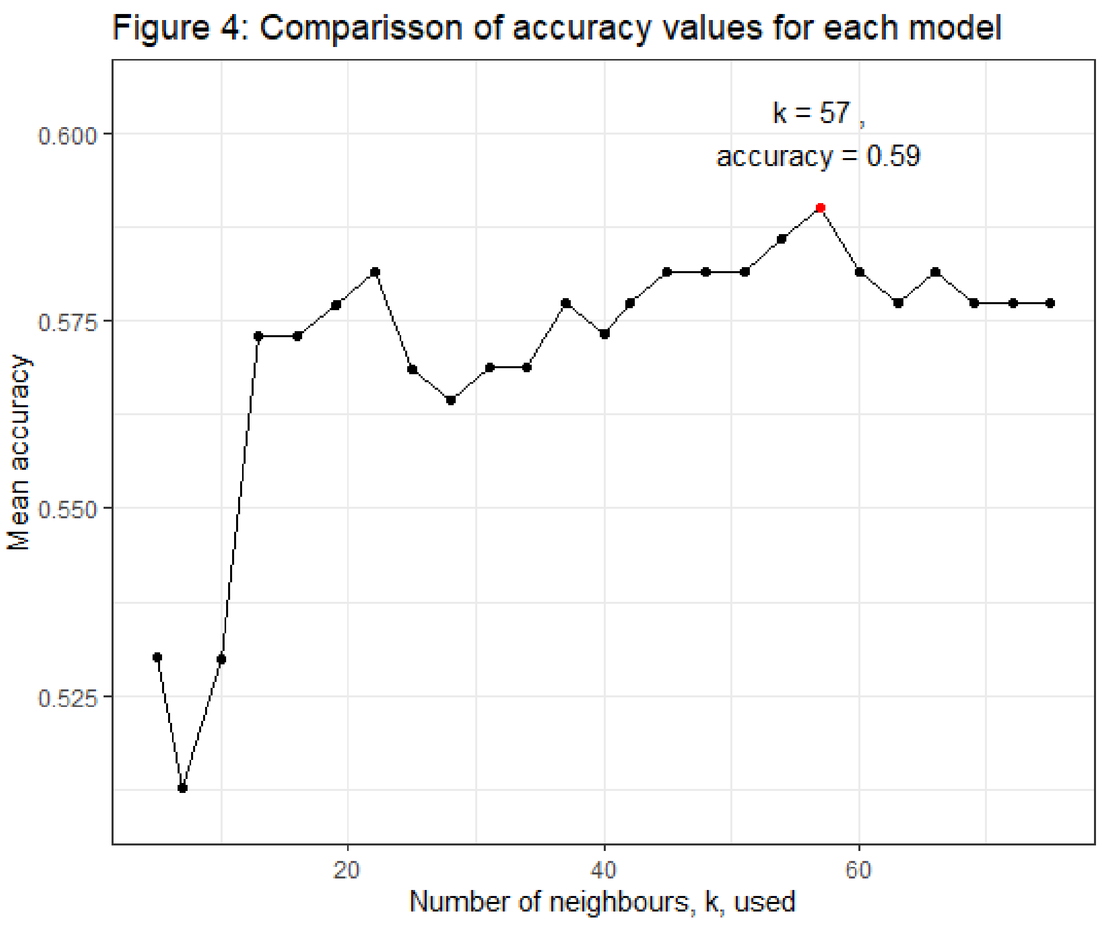
Step 5:

```
tune_df <- collect_metrics(tune_k)
tune_df

## # A tibble: 50 x 7
##    neighbors .metric  .estimator  mean     n std_err .config
##        <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1         5 accuracy binary     0.530     5  0.0328 Preprocessor1_Model01
##  2         5 roc_auc  binary     0.562     5  0.0384 Preprocessor1_Model01
##  3         7 accuracy binary     0.513     5  0.0224 Preprocessor1_Model02
##  4         7 roc_auc  binary     0.571     5  0.0369 Preprocessor1_Model02
##  5        10 accuracy binary     0.530     5  0.0229 Preprocessor1_Model03
##  6        10 roc_auc  binary     0.572     5  0.0380 Preprocessor1_Model03
##  7        13 accuracy binary     0.573     5  0.0354 Preprocessor1_Model04
##  8        13 roc_auc  binary     0.591     5  0.0416 Preprocessor1_Model04
##  9        16 accuracy binary     0.573     5  0.0300 Preprocessor1_Model05
## 10        16 roc_auc   binary     0.597     5  0.0394 Preprocessor1_Model05
## # ... with 40 more rows
```

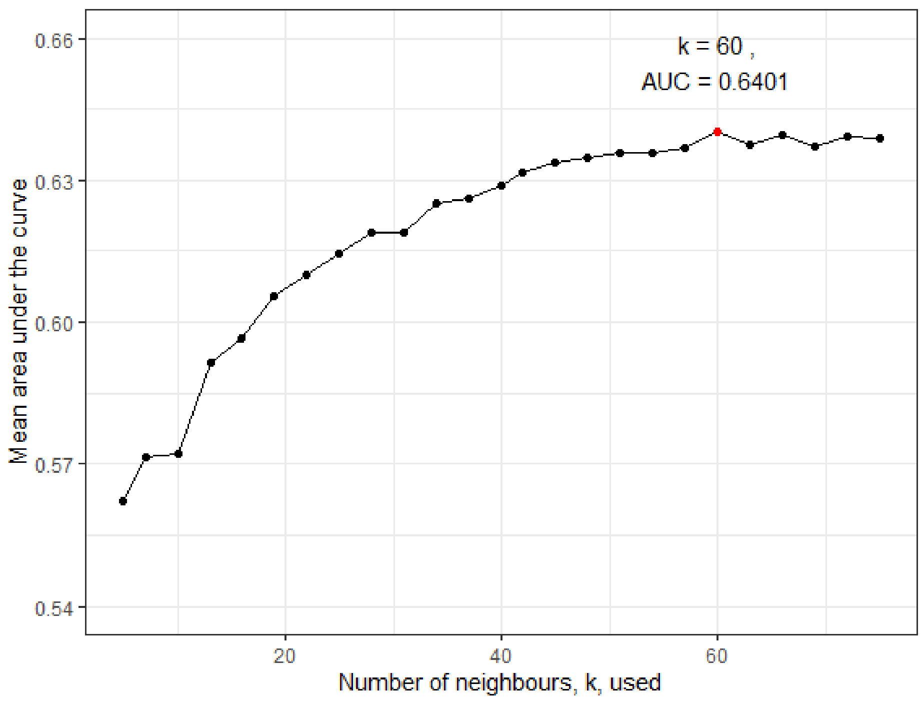
```
# Separating accuracy and roc_aus metrics
accuracy_df <- tune_df[tune_df$.metric=="accuracy",]
roc_auc_df <- tune_df[tune_df$.metric=="roc_auc",]

# Plotting the mean accuracy for each value of k
ggplot(accuracy_df, aes(x = neighbors, y = mean)) +
  geom_point() +
  geom_line() +
  ggtitle("Figure 4: Comparisson of accuracy values for each model" ) +
  xlab("Number of neighbours, k, used") +
  ylab("Mean accuracy") +
  geom_text(data = filter(accuracy_df, neighbors == 57),
            aes(x = neighbors,
                label =  paste("k =",neighbors,",\naccuracy =", round(mean, 2
))),
            nudge_y = 0.01,
            size = 4) +
  geom_point(data = filter(accuracy_df, neighbors == 57),
             aes(x = neighbors, y = mean), colour= "red") +
  scale_y_continuous(limits = c(0.51, 0.605)) +
  theme_bw() +
  theme(legend.position = "none")
```



```r
# Plotting the mean roc_auc for each value of k
ggplot(roc_auc_df, aes(x = neighbors, y = mean)) +
  geom_point() +
  geom_line() +
  ggtitle("Figure 5: Comparisson of the areas under the ROC curves for\n each
model" ) +
  xlab("Number of neighbours, k, used") +
  ylab("Mean area under the curve") +
  geom_text(data = filter(roc_auc_df, neighbors == 60),
            aes(x = neighbors,
                label =  paste("k =",neighbors,",\nAUC =", round(mean, 4))),
            nudge_y = 0.015,
            size = 4) +
  geom_point(data = filter(roc_auc_df, neighbors == 60),
             aes(x = neighbors,y = mean), color = "red") +
  scale_y_continuous(limits = c(0.54, 0.66)) +
  theme_bw()+
  theme(legend.position = "none")
```

Figure 5: Comparisson of the areas under the ROC curves for each model



The best mean accuracy was when k = 57, the best mean area under the ROC curve was found at k = 60.

Step 6:

```
# These functions were also used to create the annotation above:
highest_accuracy <- tune_k %>%
  select_best(metric = "accuracy")
highest_accuracy # best value was 57

highest_AUC <- tune_k %>%
  select_best(metric = "roc_auc")
highest_AUC # best value was 60
```

Step 7:

```
knn_final <-finalize_model(model_spec, highest_accuracy)
knn_final

## K-Nearest Neighbor Model Specification (classification)
##
## Main Arguments:
##   neighbors = 57
##
## Computational engine: kknn
```

Step 8:

```
knn_final <- finalize_model(model_spec, highest_accuracy) %>%
  fit(affair~., data = affair_training_prepro)
```

Evaluation

Step 1:

```
class_prediction <- knn_final %>%
  predict(new_data = affair_test_prepro )
knitr::kable(head(class_prediction, 6))
```

.pred_class
no
yes
no
no
no
no

Step 2:

```
class_prediction <- class_prediction %>%
  bind_cols(affair_test_prepro %>%
               select( affair))
knitr::kable(head(class_prediction, 6))
```

.pred_class	affair
no	no
yes	no
no	no
no	no
no	no
no	no

Step 3:

```
conf_matrix <- class_prediction %>%
  conf_mat(truth = affair, estimate = .pred_class )
conf_matrix
```

```
##           Truth
## Prediction no yes
##        no   74  18
##        yes 43  15
```

Step 4:

```
sens(class_prediction, affair, .pred_class)
## 1 sens    binary         0.632
```

The sensitivity of the model is 0.6325. Sensitivity is a measure of how accurately the model predicts a positive result. In this case, the model was able to predict a positive case (someone being disloyal) a little bit better than if you flipped a coin to decide. This will be detrimental as many of the people marked as disloyal by the model indicated that they had been loyal. Of course, disloyal people might be expected to lie on such a survey but we cannot assume the worst of people. That being the case, the model does not have the predictive power to fairly and compassionately determine the loyalty of a person.

```
spec(class_prediction, affair, .pred_class)
## 1 spec    binary         0.455
```

The sensitivity of the model was 0.4545. Specificity is a measure of how reliably the model can predict a negative case. Using this metric, the model appears to be about as good a classifier as flipping a coin to decide. It has less harmful impacts in reality as it doesn't accuse anyone of being disloyal, but by the same token it also allows disloyal partners to potentially continue an affair.

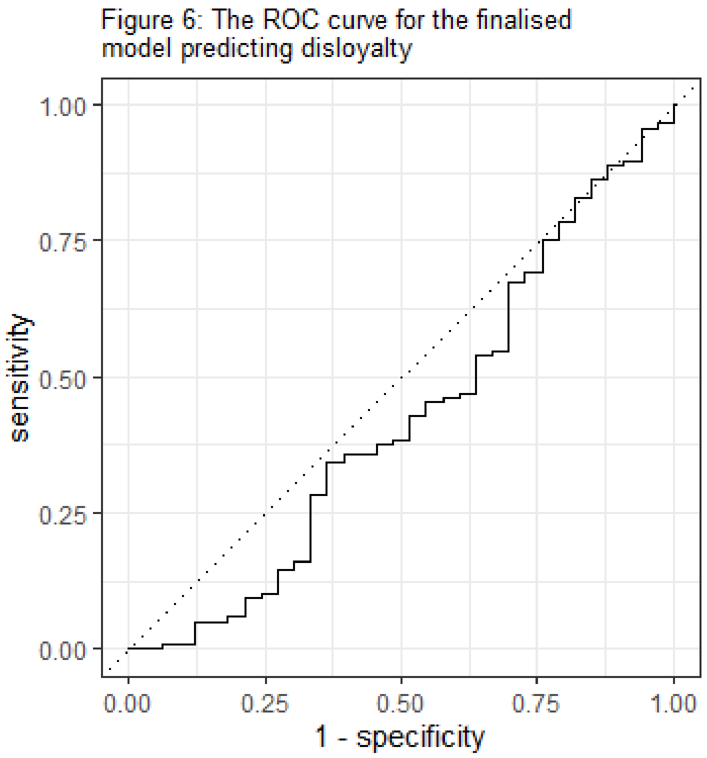
Step 5:

```
class_prediction <- class_prediction %>%
  bind_cols(predict(object = knn_final,
                    new_data = affair_test_prepro,
                    type = "prob"))
knitr::kable(head(class_prediction, 6))
```

.pred_class	affair	.pred_no	.pred_yes
no	no	0.5847266	0.4152734
yes	no	0.4447313	0.5552687
no	no	0.8084423	0.1915577
no	no	0.5404249	0.4595751
no	no	0.5311186	0.4688814
no	no	0.6508665	0.3491335

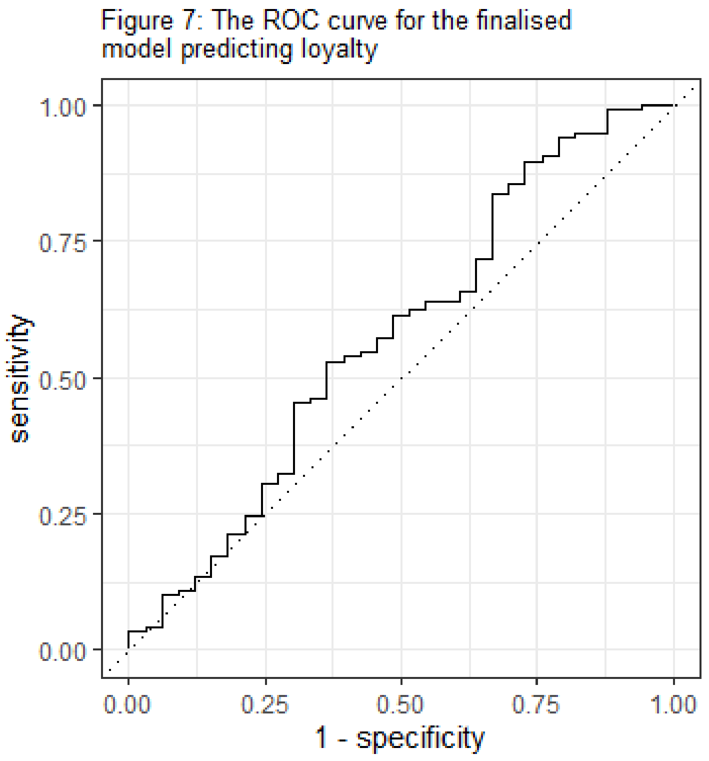
Step 6:

```
class_prediction %>%
  roc_curve(truth = affair, .pred_yes) %>%
  autoplot() +
  ggtitle("Figure 6: The ROC curve for the finalised \nmodel predicting dislo
yalty") +
  theme(plot.title = element_text(size = 10))
```



This model is not predicting disloyalty well. The line of y = x is the equivalent of a completely random classification. This figure indicates that the predictors do not provide enough information nor strong enough associations with the outcome variable to make meaningful predictions. Ideally, we would like to see the chart reach up to the top left where the true positive rate is high and the false positive rate is low. Even altering the chart to represent a model that predicts loyalty does not reach that area (figure 7).

```
class_prediction %>%
  roc_curve(truth = affair, .pred_no) %>%
  autoplot() +
  ggtitle("Figure 7: The ROC curve for the finalised \nmodel predicting loyal
ty") +
  theme(plot.title = element_text(size = 10))
```



Step 7:

```
class_prediction %>%
  roc_auc(truth = affair, .pred_yes)


##   .metric .estimator .estimate
## 1 roc_auc binary         0.424

class_prediction %>%
  roc_auc(truth = affair, .pred_no)


##   .metric .estimator .estimate
## 1 roc_auc binary         0.576
```

A completely random classification model would produce an area under the curve of 0.5. This further demonstrates that this model (with an area under the ROC curve of 0.42) is lacking predictive power. The same model predicting instead for loyalty is better, at 0.58, but still lacking in predictive power.

Step 8:

```r
# creating a tibble of the form seen in affairs_df
#a)
bono_prediction <- tibble(
  sex = factor("male", levels=c("female", "male")),
  age = 47,
  ym = 15,
  child = factor("no", levels = c("no", "yes")),
  religious = factor("2", levels = c("1","2", "3", "4", "5", "6", "7")),
  education = factor("20", levels = c("9","12","14","16","17","18","20")),
  occupation = factor("6", levels = c("1","2","3","4","5","6","7")),
  rate = factor("5", levels=c("1","2", "3", "4", "5")))

knitr::kable(head(bono_prediction))
```

sex	age	ym	child	religious	education	occupation	rate
male	47	15	no	2	20	6	5

```r
# b)
bono_prepro <- affair_recipe %>%
  bake(bono_prediction)
knitr::kable(head(bono_prepro))
```

age	ym	sex_female	child_yes	religious_X2	religious_X3	religious_X4	religious_X5	rate_X2	rate_X3	rate_X4	rate_X5
1.613602	1.170197	-1.032579	-1.73825	1.609002	-0.5859641	-0.5925321	-0.3294524	-0.4531053	-0.4254893	-0.6920672	1.438771

education_X12	education_X14	education_X16	education_X17	education_X18	education_X20	occupation_X2	occupation_X3	occupation_X4	occupation_X5	occupation_X6	occupation_X7
-0.2420168	-0.5859641	-0.4531053	-0.4531053	-0.4935942	2.462698	-0.1474475	-0.3373386	-0.3971635	-0.7192083	1.938911	-0.1752291

```r
# c)
bono_pred <- knn_final %>%
  predict(new_data = bono_prepro, type = "prob")
knitr::kable(head(bono_pred))
```

.pred_no	.pred_yes
0.451819	0.548181

Step 8.d)

We have predicted that Bono is more likely to be disloyal. While we could ruin his partners day, we will not be telling them. The predictive power of this model was found to be insufficient given the accuracy and area under the curves. The model is quite competitive with a random classifier, certainly not something a data scientist wants to see nor a model they should boast about producing.

Furthermore, having the ability to predict something of this nature does not give us the right to create conflict unnecessarily. Our actions could very well make the situation worse, or even dangerous for that individual. To do so would be to leap into George Orwell's 1984, where an organization has complete control of every outcome and there is no tolerance of emotion or privacy.

A1844790 - Dylan Rohan

References:

Hetzel, A. M. and M. Cappetta (1971). "Marriages; trends and characteristics, United States."

Kuhn, M. (2015). "Caret: classification and regression training." Astrophysics Source Code Library: ascl: 1505.1003.

Kuhn, M., et al. (2019). "Rsample: General resampling infrastructure." R package version 0.0 5.

Kuhn, M. and H. Wickham (2020). "Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles." Boston, MA, USA.[(accessed on 10 December 2020)].

Kuhn, M. and H. Wickham (2022). recipes: Preprocessing and Feature Engineering Steps for Modeling.

McNamara, A., et al. (2018). "skimr: Compact and flexible summaries of data." R package version 1(1).

Warnes, G. R., et al. (2018). "Package 'gmodels'." Vienna: R Foundation for Statistical Computing.

Wickham, H., et al. (2019). "Welcome to the {tidyverse}." Journal of Open Source Software 4(43): 1686.

Wickham, H., et al. (2016). "ggplot2: Create elegant data visualisations using the grammar of graphics." R package version 2(1).

Xie, Y. (2013). "knitr: A general-purpose Tool for dynamic report generation in R." R package version 1(1).