

St344 Individual Coursework U1619685

Stephen Brownsey

```
library(tidyverse)
library(lubridate)
library(broom)
library(car)
library(coin)
library(pgirmess)
library(Hmisc)
library(rio)
library(gridExtra)
```

Lab work

This section of the technical appendix will cover the code produced as part of lab 6, which was used to guide the starting point for the individual project work in general, especially in terms of datasets and provided a good base for exploratory data analysis in particular for pointing out the effect of different days.

```
#function to lower case column names as in general it is better
#and easier to refer to everything in the lower case + I am a Hadley fanboy
```

```
hadley_format <- function(data){
  for (i in 1:length(colnames(data))) {
    colnames(data)[i] = tolower(colnames(data)[i])
  }
  data
}
```

```
#Setting up the file path and loading in the .csv files
```

```
file_loc <- "C:/Users/Stephen/Desktop/University Work/Year 3 uni/St344/"
monthYears <- paste0(month.abb[c(3:12,1:8)], "_", c(rep(18,10),rep(19,8)))
d <- list()
for (i in 1:length(monthYears)) {
  filename <- paste0(file_loc, "Appointments_GP_Daily_Aug19/CCG_CSV_", monthYears[i], ".csv")
  d[[i]] <- import(filename, setclass = "tibble")
}
```

```
#Checking whether each file has the same number of columns
```

```
a <- TRUE
for(i in 1:length(d)){
  cols_1 <- length(d[[1]])
  if(cols_1 - length(d[[i]]) != 0){
    a <- FALSE
  }
  a
}
a
```

```
## [1] TRUE

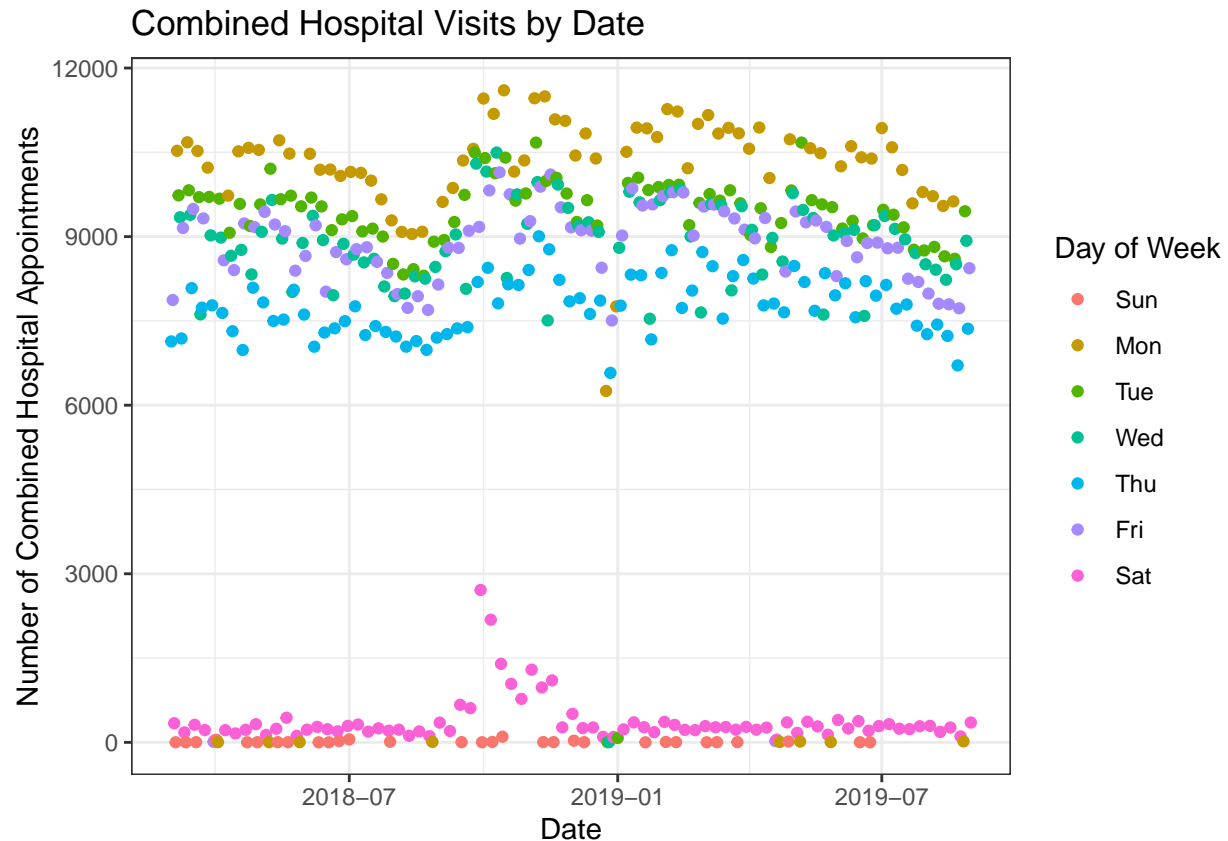
#Loading in the Coventry Data from the files
covData <- tibble()
for (i in 1:length(monthYears)) {
  covData <- rbind(covData, filter(d[[i]], CCG_NAME=="NHS Coventry and Rugby CCG"))
}

#This code removes the d variable from the environment and should speed things up
#As the d variable is over 500mb and can cause performance issues in rstudio
remove(d)

covData <- select(covData, Appointment_Date, APPT_STATUS, HCP_TYPE, APPT_MODE, TIME_BETWEEN_BOOK_AND_APPT,
                  COUNT_OF_APPOINTMENTS) %>%
  mutate(Appointment_Date = parse_date_time(Appointment_Date, "%d-%b-%Y")) %>%
  #Changing the letters to lower case to match Hadley Wickhams style guide
  hadley_format() %>%
  #updating the appointment_date variable
  mutate(appointment_data = mdy(appointment_date))

#Looking at number of appointments by each mode
appt_by_mode <- covData %>%
  group_by(appt_mode) %>%
  summarise(count = sum(count_of_appointments))

#Produce a plot line in the lab to ensure my code thus far is correct
covData %>%
  group_by(appointment_date) %>%
  summarise(count = sum(count_of_appointments)) %>%
  ggplot() +
  geom_point(aes(x = appointment_date, y = count,
                 colour = wday(appointment_date, label = TRUE, abbr = TRUE))) +
  labs(x = "Date", y = "Number of Combined Hospital Appointments",
       title = "Combined Hospital Visits by Date",
       colour = "Day of Week") +
  theme_bw() +
  #Can customise colours in due course if required inside next line
  scale_colour_discrete()
```



```
#importing next dataset which involves which patients are registered at each practice
app_gp_coverage <- import(paste0(file_loc,
                                "Appointments_GP_Daily_Aug19/APPOINTMENTS_GP_COVERAGE.csv"),
                          setclass = "tibble") %>%
mutate(Appointment_Month = parse_date_time(Appointment_Month, "%d-%b-%Y")) %>%
hadley_format()

##combining the two datasets for future use
#Filtered to only contain extra information where necessary
data <- left_join(covData, app_gp_coverage, by = c("ccg_code" = "commissioner_organisation_code"))
data <- inner_join(covData, app_gp_coverage, by = c("ccg_code" = "commissioner_organisation_code")) %>%
  filter(year(appointment_date) == year(appointment_month) &
         month(appointment_date) == month(appointment_month))
```

Demand on Coventry GPs

Since there is no data for number of GPs per practice, there is insufficient data to work out an appointment/GP number and hence demand will be modelled by overall appointments in the Cov Area. If this data had been available, then the average time of each visit type and number could have been averaged out over the GPs to find a demand average for GPs but sadly this wasn't available. Appointments which were not attended are still used for the count as the GP still had to be available to answer the person. The counts have been summed by week to take account of differences between days and too try and *absorb* some of that variability and allows for working days to not be a factor. There has also been no subsetting based on `hcp_type` -> as all appointments is theoretically workload (if admin went through the roof would still be an increased workload for example)

```

#Seasons data as per: https://www.timeanddate.com/calendar/seasons.html
seasons <- tibble(season = c("winter", "spring", "summer", "autumn",
                             "winter", "spring", "summer"),
                  start_date = c(as.Date("2017-12-21"), as.Date("2018-03-20"),
                                as.Date("2018-06-20"), as.Date("2018-09-23"),
                                as.Date("2018-12-21"), as.Date("2019-03-20"),
                                as.Date("2019-06-20")),
                  end_date = c(as.Date("2018-03-19"), as.Date("2018-06-19"),
                               as.Date("2018-09-22"), as.Date("2018-12-20"),
                               as.Date("2019-03-19"), as.Date("2019-06-19"),
                               as.Date("2019-09-22")))

#Some changes to the data, adding in season and week number to each observation
data <- data %>%
  #Detecting week number of appointment date
  mutate(week_num = isoweek(appointment_date)) %>%
  mutate(year = year(appointment_date)) %>%
  mutate(year = if_else(year == 2018, 0, 1)) %>%
  mutate(week_num = week_num + (52 * year)) %>%
  #Taking account of the fact that 1 actually refers to week 53
  mutate(week_num = if_else(week_num == 1, 53, week_num)) %>%
  mutate(week_num = week_num - 8) %>%
  #Changing class of appointment date variable for easier comparisons
  mutate(appointment_date = appointment_date %>% substr(1,10) %>% as.Date()) %>%
  #Adding
  mutate(season = if_else(between(appointment_date,
                                  seasons$start_date[1], seasons$end_date[1]), "winter",
                          if_else(between(appointment_date,
                                  seasons$start_date[2], seasons$end_date[2]), "spring",
                          if_else(between(appointment_date,
                                  seasons$start_date[3], seasons$end_date[3]), "summer",
                          if_else(between(appointment_date,
                                  seasons$start_date[4], seasons$end_date[4]), "autumn",
                          if_else(between(appointment_date,
                                  seasons$start_date[5], seasons$end_date[5]), "winter",
                          if_else(between(appointment_date,
                                  seasons$start_date[6], seasons$end_date[6]), "spring",
                          if_else(between(appointment_date,
                                  seasons$start_date[7], seasons$end_date[7]), "summer",
                          "Error"))))))))

# Making adjustment as some weeks may contain dates from different seasons
# Use median as 7 days in a week so only modal one will occur
temp <- data %>%
  mutate(season_num = if_else(season == "winter", 1,
                              if_else(season == "spring", 2,
                              if_else(season == "summer", 3, 4)))) %>%
  select(week_num, appointment_date, season, season_num) %>%
  unique() %>%
  select(week_num, season, season_num) %>%
  group_by(week_num) %>%
  summarise(count = median(season_num)) %>%
  mutate(season_median = if_else(count == 1, "winter",

```

```

        if_else(count == 2, "spring",
        if_else(count == 3, "summer", "autumn")))) %>%
select(week_num, season_median)

data <- data %>%
  inner_join(temp)

## Joining, by = "week_num"
#Saving the data before I remove the anonymous weeks
#For use in part two of the report
part_2_data <- data

#Count summaries of the data by appointment mode and week
appt_by_mode <- data %>%
  group_by(appt_mode, week_num, season_median) %>%
  summarise(count = sum(count_of_appointments)) %>%
  ungroup()

#Looking into the two low results for Face-to-Face
appt_by_mode %>%
  filter(count < 20000 & appt_mode == "Face-to-Face")

## # A tibble: 2 x 4
##   appt_mode    week_num season_median count
##   <chr>         <dbl> <chr>         <int>
## 1 Face-to-Face      1 winter         12032
## 2 Face-to-Face     44 winter         15428

#Returns week 9 (first week and not a full one, only 4 results) and 52 (over Christmas so less visits)
data %>%
  filter(week_num %in% c(9, 52)) %>%
  select(appointment_date) %>%
  unique()

## # A tibble: 13 x 1
##   appointment_date
##   <date>
## 1 2018-04-23
## 2 2018-04-24
## 3 2018-04-25
## 4 2018-04-26
## 5 2018-04-27
## 6 2018-04-28
## 7 2018-04-29
## 8 2019-02-18
## 9 2019-02-19
## 10 2019-02-20
## 11 2019-02-21
## 12 2019-02-22
## 13 2019-02-23

#No appointments on 30th December - CHECK
data %>%
  filter(str_detect(appointment_date, '12-30'))

```

```
## # A tibble: 0 x 20
## # ... with 20 variables: appointment_date <date>, appt_status <chr>,
## #   hcp_type <chr>, appt_mode <chr>, time_between_book_and_appt <chr>,
## #   ccg_code <chr>, count_of_appointments <int>, appointment_data <date>,
## #   stpcd <chr>, regional_local_office_code <chr>, region_code <chr>,
## #   appointment_month <dtm>, `included practices` <int>, `open
## #   practices` <int>, `patients registered at included practices` <int>,
## #   `patients registered at open practices` <int>, week_num <dbl>,
## #   year <dbl>, season <chr>, season_median <chr>

##Therefore we remove these two weeks (9 and 52) as they seem to be anomalous in comparison to others.
data <- data %>%
  filter(!week_num %in% c(9, 52)) %>%
  filter(appt_mode %in% c("Face-to-Face", "Telephone", "Home Visit"))

#Looking at median season result
median_season <- appt_by_mode %>%
  filter(appt_mode == "Face-to-Face") %>%
  group_by(season_median) %>%
  summarise(F2F = median(count)) %>%
  ungroup() %>%
  cbind(appt_by_mode %>%
    filter(appt_mode == "Telephone") %>%
    group_by(season_median) %>%
    summarise(telephone = median(count)) %>%
    ungroup() %>%
    select(telephone)) %>%
  cbind(appt_by_mode %>%
    filter(appt_mode == "Home Visit") %>%
    group_by(season_median) %>%
    summarise(home_visit = median(count)) %>%
    ungroup() %>%
    select(home_visit))
median_season

##   season_median   F2F telephone home_visit
## 1      autumn 38791     4244     558.0
## 2      spring 35542     4290     516.5
## 3      summer 33854     4153     519.0
## 4      winter 36603     4770     566.0

#Percentage of appointments which are telephone: Just approx for using value in report
(100*sum(median_season$telephone))/(sum(median_season$telephone
+ median_season$F2F + median_season$home_visit))

## [1] 10.61819
```

Model exploration

Plotting a couple of models to see whether

```
model1 <- glm(count ~ 0 + week_num, family = "poisson", data = appt_by_mode %>%
  filter(appt_mode == "Face-to-Face"))
```

```

modell1a <- glm(count ~ week_num, family = "poisson", data = appt_by_mode %>%
  filter(appt_mode == "Face-to-Face"))
#Face to Face Model
modell2 <- glm(count ~ 0 + season_median + week_num , family = "poisson", data = appt_by_mode %>%
  filter(appt_mode == "Face-to-Face"))
modell2a <- glm(count ~ season_median + week_num , family = "poisson", data = appt_by_mode %>%
  filter(appt_mode == "Face-to-Face"))

modell1 %>% summary()

```

```

##
## Call:
## glm(formula = count ~ 0 + week_num, family = "poisson", data = appt_by_mode %>%
##   filter(appt_mode == "Face-to-Face"))
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -513.9    176.8    476.6    632.7    827.6
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## week_num 0.1553239   0.0000112   13871  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##      Null deviance: 51118458  on 79  degrees of freedom
## Residual deviance: 20434126  on 78  degrees of freedom
## AIC: 20435097
##
## Number of Fisher Scoring iterations: 8

```

```

modell1a %>% summary()

##
## Call:
## glm(formula = count ~ week_num, family = "poisson", data = appt_by_mode %>%
##   filter(appt_mode == "Face-to-Face"))
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -137.936    -9.912     7.303    13.652    42.475
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 1.044e+01  1.229e-03 8488.950  < 2e-16 ***
## week_num    1.187e-04  2.667e-05   4.452 8.52e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##      Null deviance: 65217  on 78  degrees of freedom

```

```
## Residual deviance: 65198 on 77 degrees of freedom
## AIC: 66170
##
## Number of Fisher Scoring iterations: 4
```

```
model2 %>% summary()
```

```
##
## Call:
## glm(formula = count ~ 0 + season_median + week_num, family = "poisson",
##      data = appt_by_mode %>% filter(appt_mode == "Face-to-Face"))
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -132.527   -5.475    7.944   14.691   28.665
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## season_medianautumn 1.056e+01  1.735e-03 6084.55  <2e-16 ***
## season_medianspring 1.041e+01  1.464e-03 7108.70  <2e-16 ***
## season_mediansummer 1.039e+01  1.673e-03 6212.59  <2e-16 ***
## season_medianwinter 1.040e+01  1.793e-03 5800.15  <2e-16 ***
## week_num            3.146e-04  2.749e-05   11.45  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##      Null deviance: 51118458 on 79 degrees of freedom
## Residual deviance:   55361 on 74 degrees of freedom
## AIC: 56340
##
## Number of Fisher Scoring iterations: 4
```

```
model2a %>% summary()
```

```
##
## Call:
## glm(formula = count ~ season_median + week_num, family = "poisson",
##      data = appt_by_mode %>% filter(appt_mode == "Face-to-Face"))
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -132.527   -5.475    7.944   14.691   28.665
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.056e+01  1.735e-03 6084.55  <2e-16 ***
## season_medianspring -1.526e-01  1.769e-03  -86.30  <2e-16 ***
## season_mediansummer -1.629e-01  1.801e-03  -90.46  <2e-16 ***
## season_medianwinter -1.561e-01  1.996e-03  -78.21  <2e-16 ***
## week_num          3.146e-04  2.749e-05   11.45  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```



```

## (Dispersion parameter for poisson family taken to be 1)
##
## Null deviance: 65217 on 78 degrees of freedom
## Residual deviance: 55361 on 74 degrees of freedom
## AIC: 56340
##
## Number of Fisher Scoring iterations: 4
BIC(model1)

## [1] 20435100
BIC(model1a)

## [1] 66175.13
BIC(model2)

## [1] 56351.36
BIC(model2a)

## [1] 56351.36
#Model 2 and 2a the same for F2F and significantly better than Model 1 so added seasons improves model
#Both model2 and 2a are equivalent as the intercept term is the same as the spring term in the other

#Let's see if it is the same for Telephone

model1 <- glm(count ~ 0 + week_num, family = "poisson", data = appt_by_mode %>%
  filter(appt_mode == "Telephone"))

model1a <- glm(count ~ week_num, family = "poisson", data = appt_by_mode %>%
  filter(appt_mode == "Telephone"))
#Telephone Model
model2 <- glm(count ~ 0 + season_median + week_num, family = "poisson", data = appt_by_mode %>%
  filter(appt_mode == "Telephone"))
#Home Visit Model
model2 <- glm(count ~ 0 + season_median + week_num, family = "poisson", data = appt_by_mode %>%
  filter(appt_mode == "Home Visit"))

model1 %>% summary()

##
## Call:
## glm(formula = count ~ 0 + week_num, family = "poisson", data = appt_by_mode %>%
## filter(appt_mode == "Telephone"))
##
## Deviance Residuals:
## Min 1Q Median 3Q Max
## -161.01 54.49 142.10 186.40 250.44
##
## Coefficients:
## Estimate Std. Error z value Pr(>|z|)
## week_num 1.276e-01 3.111e-05 4101 <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##

```

```
## (Dispersion parameter for poisson family taken to be 1)
##
## Null deviance: 5024387 on 79 degrees of freedom
## Residual deviance: 1785395 on 78 degrees of freedom
## AIC: 1786203
##
## Number of Fisher Scoring iterations: 7
model1a %>% summary()

##
## Call:
## glm(formula = count ~ week_num, family = "poisson", data = appt_by_mode %>%
## filter(appt_mode == "Telephone"))
##
## Deviance Residuals:
## Min 1Q Median 3Q Max
## -39.089 -3.062 1.396 5.548 11.893
##
## Coefficients:
## Estimate Std. Error z value Pr(>|z|)
## (Intercept) 8.220e+00 3.590e-03 2289.70 <2e-16 ***
## week_num 3.617e-03 7.532e-05 48.02 <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
## Null deviance: 8806.7 on 78 degrees of freedom
## Residual deviance: 6495.7 on 77 degrees of freedom
## AIC: 7304.8
##
## Number of Fisher Scoring iterations: 4
model2a %>% summary()

##
## Call:
## glm(formula = count ~ season_median + week_num, family = "poisson",
## data = appt_by_mode %>% filter(appt_mode == "Face-to-Face"))
##
## Deviance Residuals:
## Min 1Q Median 3Q Max
## -132.527 -5.475 7.944 14.691 28.665
##
## Coefficients:
## Estimate Std. Error z value Pr(>|z|)
## (Intercept) 1.056e+01 1.735e-03 6084.55 <2e-16 ***
## season_medianspring -1.526e-01 1.769e-03 -86.30 <2e-16 ***
## season_mediansummer -1.629e-01 1.801e-03 -90.46 <2e-16 ***
## season_medianwinter -1.561e-01 1.996e-03 -78.21 <2e-16 ***
## week_num 3.146e-04 2.749e-05 11.45 <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for poisson family taken to be 1)
##
## Null deviance: 65217 on 78 degrees of freedom
## Residual deviance: 55361 on 74 degrees of freedom
## AIC: 56340
##
## Number of Fisher Scoring iterations: 4
BIC(model1)

## [1] 1786205
BIC(model1a)

## [1] 7309.582
BIC(model2a)

## [1] 56351.36
##Again model2(a) is the best so that's what will be used in the plotting function
##Final Models

glm(count ~ 0 + season_median + week_num , family = "poisson", data = appt_by_mode %>%
  filter(appt_mode == "Face-to-Face")) %>%
  summary()

##
## Call:
## glm(formula = count ~ 0 + season_median + week_num, family = "poisson",
## data = appt_by_mode %>% filter(appt_mode == "Face-to-Face"))
##
## Deviance Residuals:
## Min 1Q Median 3Q Max
## -132.527 -5.475 7.944 14.691 28.665
##
## Coefficients:
## Estimate Std. Error z value Pr(>|z|)
## season_medianautumn 1.056e+01 1.735e-03 6084.55 <2e-16 ***
## season_medianspring 1.041e+01 1.464e-03 7108.70 <2e-16 ***
## season_mediansummer 1.039e+01 1.673e-03 6212.59 <2e-16 ***
## season_medianwinter 1.040e+01 1.793e-03 5800.15 <2e-16 ***
## week_num 3.146e-04 2.749e-05 11.45 <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
## Null deviance: 51118458 on 79 degrees of freedom
## Residual deviance: 55361 on 74 degrees of freedom
## AIC: 56340
##
## Number of Fisher Scoring iterations: 4
glm(count ~ 0 + season_median + week_num , family = "poisson", data = appt_by_mode %>%
  filter(appt_mode == "Telephone")) %>%
  summary()
```

```
##
## Call:
## glm(formula = count ~ 0 + season_median + week_num, family = "poisson",
##      data = appt_by_mode %>% filter(appt_mode == "Telephone"))
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -39.215  -1.978   1.361   5.722  11.142
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## season_medianautumn 8.211e+00  5.125e-03 1602.12  <2e-16 ***
## season_medianspring 8.237e+00  4.200e-03 1961.46  <2e-16 ***
## season_mediansummer 8.184e+00  4.800e-03 1704.92  <2e-16 ***
## season_medianwinter 8.222e+00  5.044e-03 1630.09  <2e-16 ***
## week_num            3.778e-03  7.685e-05   49.16  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##      Null deviance: 5024386.8  on 79  degrees of freedom
## Residual deviance:   6339.6  on 74  degrees of freedom
## AIC: 7154.8
##
## Number of Fisher Scoring iterations: 4
glm(count ~ 0 + season_median + week_num , family = "poisson", data = appt_by_mode %>%
      filter(appt_mode == "Home Visit")) %>%
summary()

##
## Call:
## glm(formula = count ~ 0 + season_median + week_num, family = "poisson",
##      data = appt_by_mode %>% filter(appt_mode == "Home Visit"))
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -16.0719  -0.6320   0.2222   1.2904   4.5027
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## season_medianautumn 6.3074293  0.0142006 444.166  < 2e-16 ***
## season_medianspring 6.2025845  0.0118739 522.372  < 2e-16 ***
## season_mediansummer 6.1825092  0.0135689 455.640  < 2e-16 ***
## season_medianwinter 6.2292596  0.0143393 434.419  < 2e-16 ***
## week_num            0.0011161  0.0002213   5.043 4.59e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##      Null deviance: 438047.05  on 79  degrees of freedom
## Residual deviance:    705.96  on 74  degrees of freedom
## AIC: 1355.2
```

```
##
```

```
## Number of Fisher Scoring iterations: 4
```

Writing a plotting function to allow for more efficient code for generating plots by appointment type, default option is no filtering (all appointment types combined).

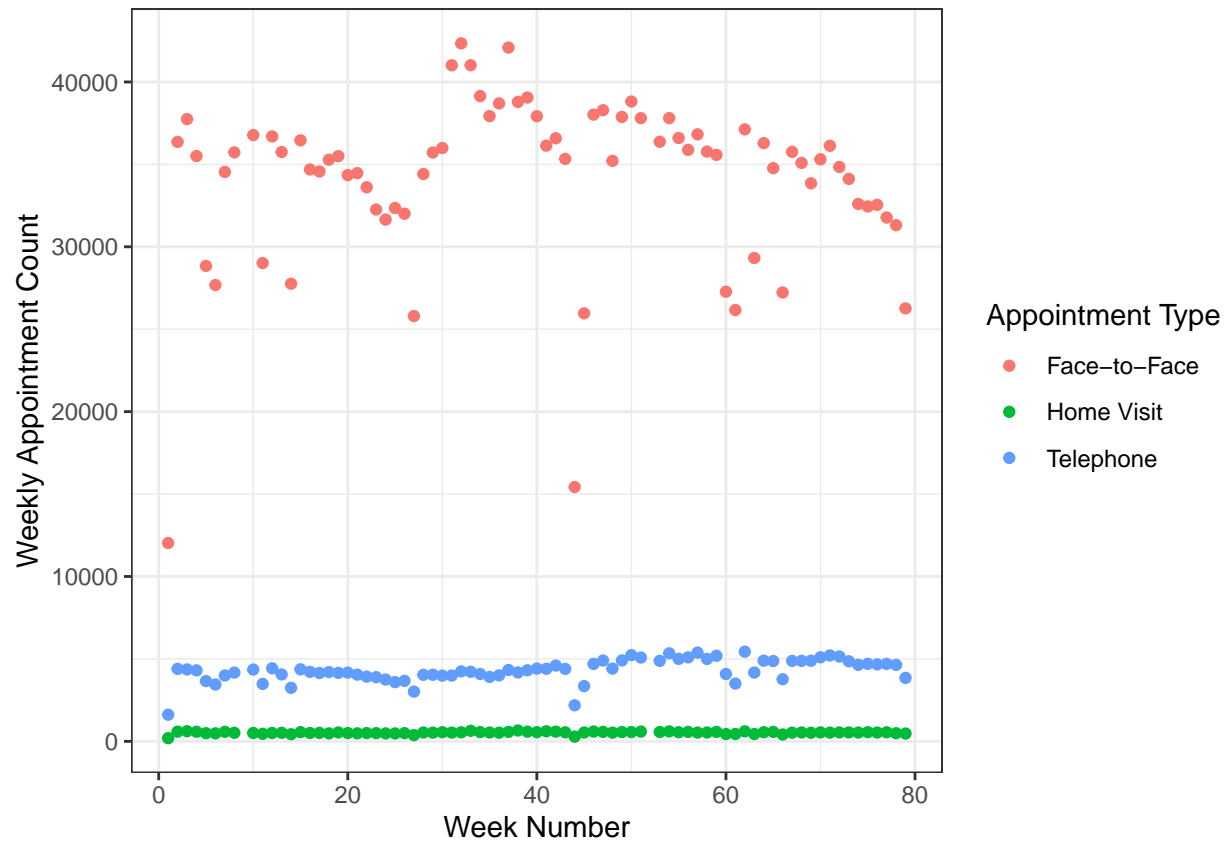
```
#Graphing function for visual aids on the appointment counts
```

```
#The filter relates to how the data is subsetting by appointment type.
```

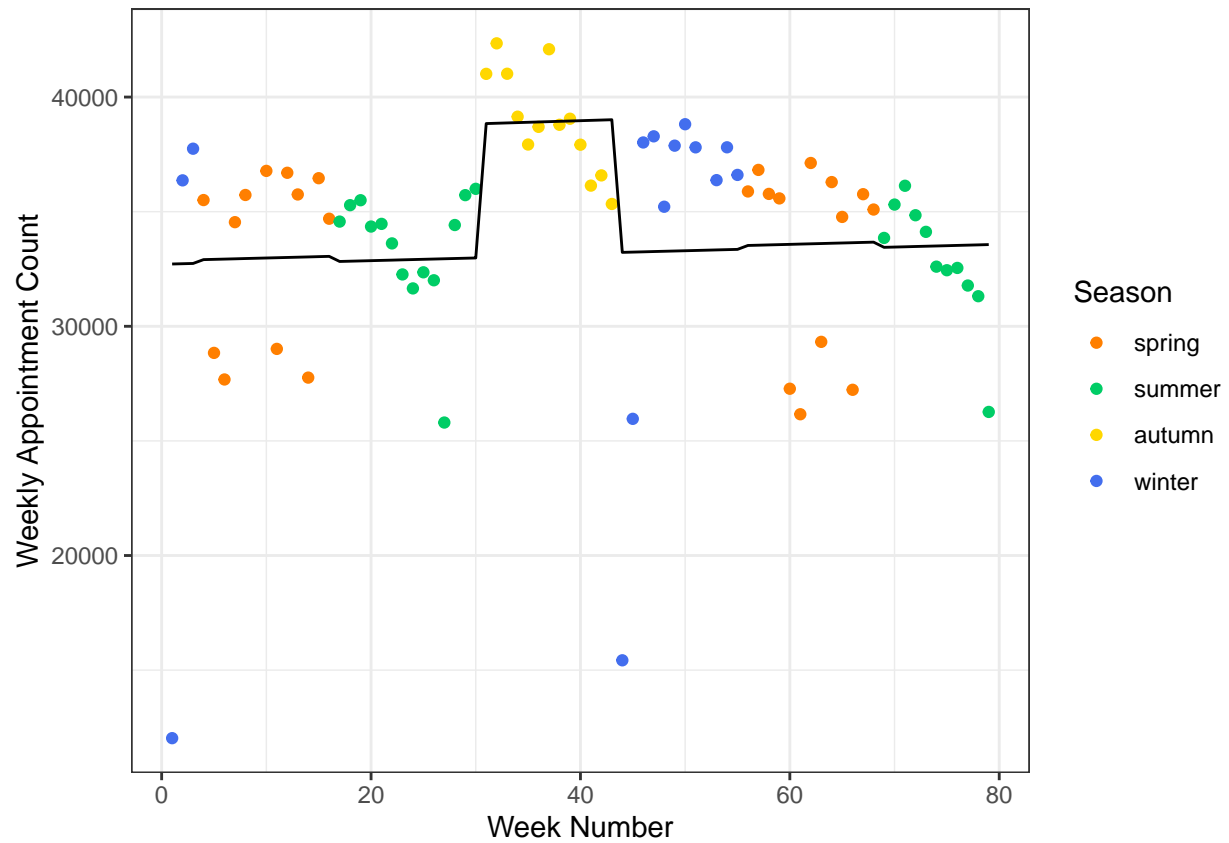
```
appt_count <- function(data, filter = "No Filter"){  
  if(!filter == "No Filter"){  
    #Updating data with filter  
    data <- data %>%  
      filter(appt_mode == filter)  
  
    model <- glm(count ~ 0 + season_median + week_num , family = "poisson", data = data)  
    #Plot with no filter (include legend)  
    output <- data %>%  
      mutate(poisson_estimate = predict(model, type = "response")) %>%  
      ggplot(aes(x = week_num)) +  
      geom_point(aes(y = count, colour = season_median)) +  
      geom_line(aes(y = poisson_estimate)) +  
      theme_bw() +  
      #title = paste0(filter, " Weekly Appointment Count with Poisson Regression Line"),  
      labs(x = "Week Number", y = "Weekly Appointment Count", colour = "Season") +  
      scale_colour_manual("Season", values = c("darkorange1", "springgreen3", "gold", "royalblue2"))  
  
  }else{  
    output <- data %>%  
      ggplot(aes(x = week_num, y = count, colour = appt_mode)) +  
      geom_point() +  
      theme_bw() +  
      #title = paste0("Weekly Appointment Count by Appointment Type"),  
      labs(x = "Week Number", y = "Weekly Appointment Count", colour = "Appointment Type")  
  }  
  
  output  
}
```

```
#Generating the required dataset to run the appt_count graphing function on:
```

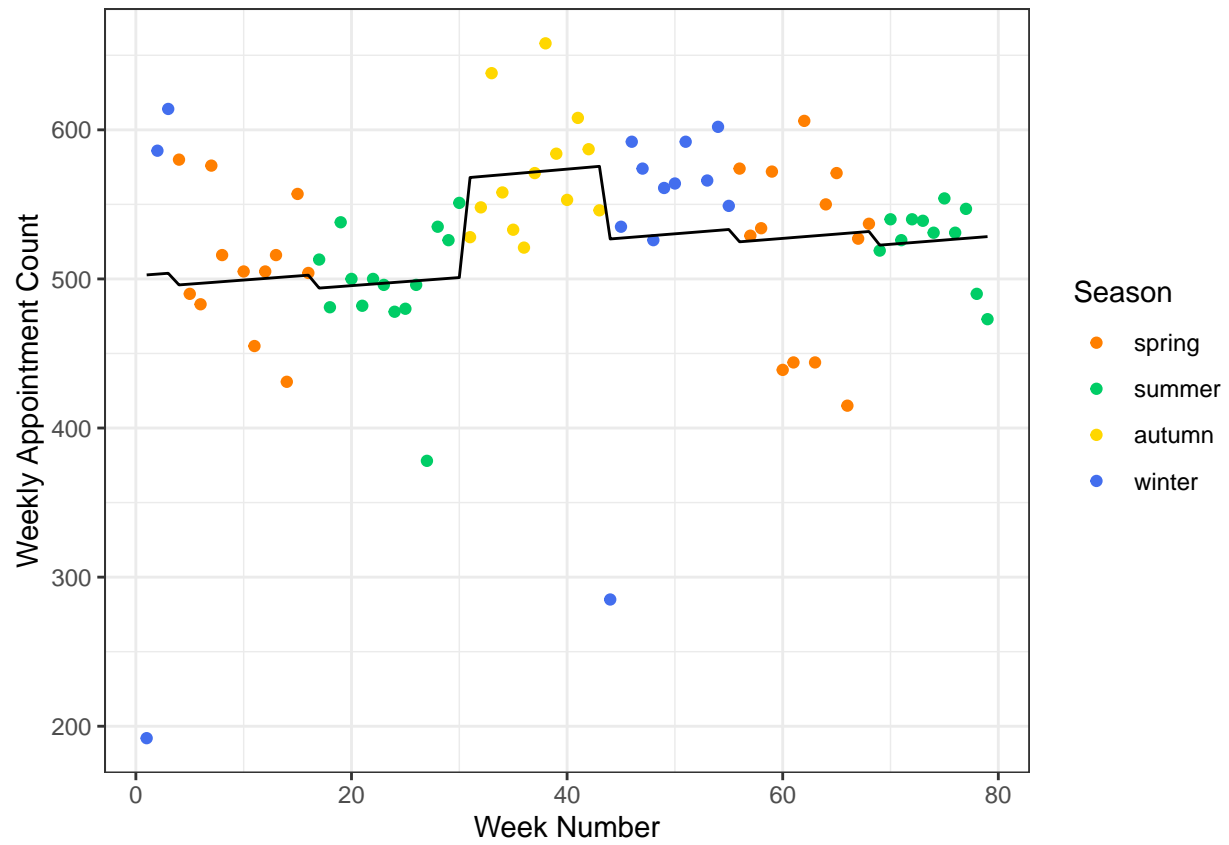
```
appt_by_mode <- data %>%  
  group_by(appt_mode, week_num, season_median) %>%  
  summarise(count = sum(count_of_appointments)) %>%  
  ungroup() %>%  
  mutate(season_median = factor(season_median,  
                                levels = c("spring", "summer", "autumn", "winter")))  
  
p1 <- appt_by_mode %>%  
  appt_count()  
p1
```



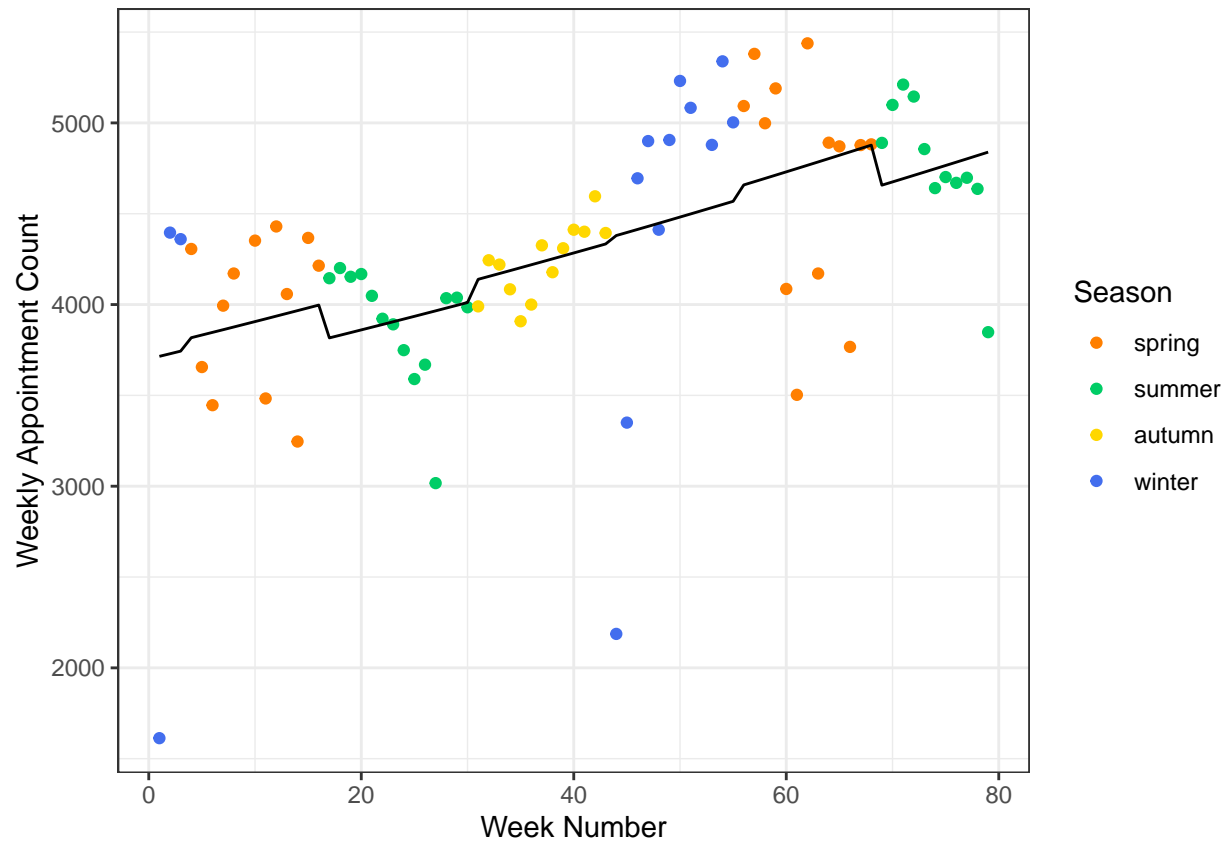
```
p2 <- appt_by_mode %>%
  appt_count("Face-to-Face")
p2
```



```
#Autumn shift not taken account of in the year 1 vs year 2 comparison and could yeild a no change when
p3 <- appt_by_mode %>%
  appt_count("Home Visit")
p3
```



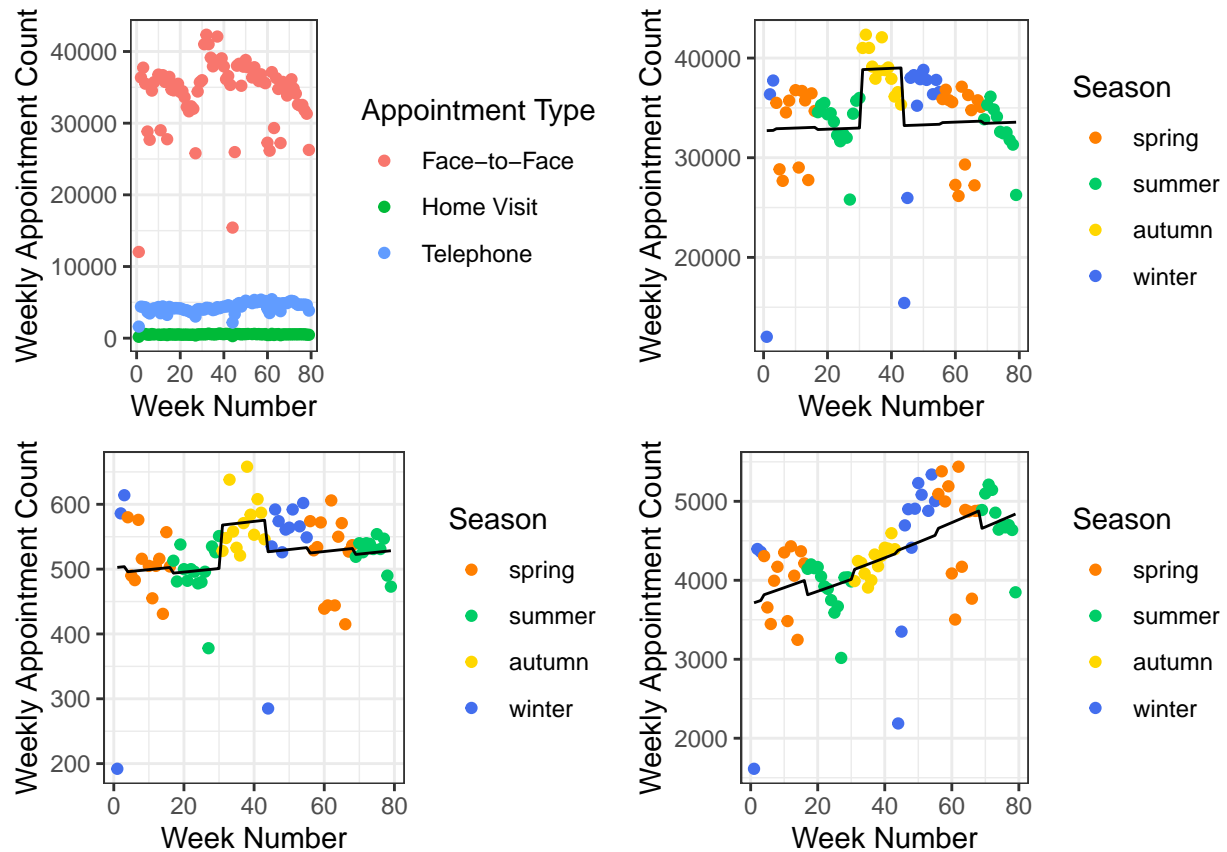
```
p4 <- appt_by_mode %>%
  appt_count("Telephone")
p4
```

```
#For the purposes of the question "Unknown" will not be considered
# appt_by_mode %>%
#   appt_count("Unknown")

#Joining the data appt_by_mode data to original data
data <- data %>%
  inner_join(appt_by_mode)

## Joining, by = c("appt_mode", "week_num", "season_median")
#Nice to visualise side by side in Rstudio but not effective in a pdf report.
pic <- grid.arrange(p1,p2,p3,p4,nrow = 2)
```



```
# Telephone Pic for report:
telephone_plot <- p4 + theme(axis.text=element_text(size=12),
  axis.title=element_text(size=14),
  plot.title = element_text(size = 22),
  legend.text = element_text(size = 12),
  legend.title = element_text(size = 14))

#For Html
telephone_plot <- p4
```

Statistical significance tests

Looking at the graph generated above, makes sense that season would be a factor to the count. So will be looking into that in the following code. Use Anova test if variance assumption holds - otherwise use the kruskal wallace test. Where the assumption is that all the groups have the same variance.

```
#Function to run the test depending on assumptions
checking_seasons <- function(data, appt_type){
  #Variance assumption (checking they are the same)
  season_check <- data %>% filter(appt_mode == appt_type) %>%
    select(appt_mode, season_median, count) %>%
    unique()
  #if < 0.05 use anova otherwise use kruskal wallace
  var_value <- leveneTest(count ~ season_median, data = season_check)[1,3]
  if(var_value < 0.05){
    test <- "Kruskal"
```

```

#Kruskal Wallace test
p_value <- kruskal.test(count ~ season_median, data = season_check)$p.value

}else{
  test <- "Anova"
  #Running the anova test
  anova1 <- aov(count ~ season_median, data = season_check)
  p_value <- summary(anova1)[[1]][1,5]
}
tibble(appt_type = appt_type, test = test, p_value = p_value)
}

#Outputting result of seasons tests by appointment type:
checking_seasons(data, "Face-to-Face") %>%
  bind_rows(checking_seasons(data, "Telephone")) %>%
  bind_rows(checking_seasons(data, "Home Visit"))

```

```

## # A tibble: 3 x 3
##   appt_type    test    p_value
##   <chr>        <chr>    <dbl>
## 1 Face-to-Face Anova    0.00216
## 2 Telephone    Kruskal 0.363
## 3 Home Visit   Anova    0.0715

```

Splitting the data by season for compared t-test would give an option to test if the seasons themselves have increased between the two years. Cold weather could lead to more illnesses. The following function uses a paired t-test to calculate whether the differences are statistically significant or not. It will either use a Welch's test if the variances are not equal, a students t-test if they are or if the data is not normal then it will perform a Wilcoxon Sign Ranked Test on the data. Autumn data cannot be considered as only 1 year of autumn data, which is a shortcoming in the data analysis.

```

#Extracting the data to be used in these tests
stat_data <- data %>%
  #Give a 0 or 1 as a year column for paired t-testing.
  mutate(year == year(appointment_date)) %>%
  select(appt_mode, week_num, season_median, year, count) %>%
  unique()

stat_test <- function(data = stat_data, type = "NA", season = "NA"){
  #Updating data

  temp_data <- data
  if(type != "NA"){
    temp_data <- temp_data %>%
      filter(appt_mode == type)
  }

  if(season != "NA"){
    temp_data <- temp_data %>%
      filter(season_median == season)
  }

  #Test the normality assumption of the data
  normal <- TRUE
  if(shapiro.test(temp_data$count)$p.value < 0.05){

```

```

  normal <- FALSE
}

#Checking validity of equal variance
year_1 <- temp_data %>%
  filter(year == 0) %>%
  select(count) %>%
  as_vector()
year_2 <- temp_data %>%
  filter(year == 1) %>%
  select(count) %>%
  as_vector()

#Setting seed to ensure reproducibility
set.seed(666)
min_length <- min(length(year_1), length(year_2))

#Defining data in another format for latter parts
test_data <- tibble(year_1 = year_1 %>% sample(min_length),
                    year_2 = year_2 %>% sample(min_length))

#t-test section
if(normal){

#Sampling for F-test if numbers are different
#Replace = FALSE by default.

variance <- TRUE
#Compares the p.value from F-test and changes the variance check accordingly
if(tidy(var.test(test_data$year_1,
                 test_data$year_2)) %>%
  select(p.value) %>% as_vector() < 0.05){
  #Varaince test significant -> variances NOT Equal
  variance <- FALSE
}
if(variance == FALSE){
  test <- "Welch"
  #Run Welch test where variance assumption NOT assumed
  p_value <- tidy(t.test((test_data$year_2 - test_data$year_1), mu = 0,
                        alternative = "greater")) %>%
  select(p.value) %>% as_vector()
  if(t_test < 0.05){
    return("H1 True")
  }else{
    return("H0 True")
  }
}else{
  test <- "Students"
  #Run students were the variance is assumed
  p_value <- tidy(t.test((test_data$year_2 - test_data$year_1), mu = 0,
                        var.equal=TRUE ,alternative = "greater")) %>%
  select(p.value) %>% as_vector()
}
}

```

```

}

}else{
  test <- "Wilcoxon"
  #Run the wilcoxon sign ranked test if normality assumption failed
  p_value <- tidy(wilcox.test(test_data$year_2 , test_data$year_1,
                             paired = TRUE, alternative = "greater")) %>%
  select(p.value) %>% as_vector()
}

#Conclusion
if(p_value < 0.05){
  outcome <- "H1 True"
}else{
  outcome <- "H0 True"
}

output <- tibble(appt_mode = type, season = season, test = test,
                 p_value = p_value, conclusion = outcome)
output

}

```

```

#Now run the paired T-Test on the data
stat_table <- stat_test(type = "Telephone") %>%
  rbind(stat_test(type = "Face-to-Face")) %>%
  rbind(stat_test(type = "Home Visit")) %>%
  rbind(stat_test(stat_data, "Telephone", "summer")) %>%
  rbind(stat_test(stat_data, "Telephone", "winter")) %>%
  rbind(stat_test(stat_data, "Telephone", "spring")) %>%
  rbind(stat_test(stat_data, "Face-to-Face", "summer")) %>%
  rbind(stat_test(stat_data, "Face-to-Face", "winter")) %>%
  rbind(stat_test(stat_data, "Face-to-Face", "spring")) %>%
  rbind(stat_test(stat_data, "Home Visit", "spring")) %>%
  rbind(stat_test(stat_data, "Home Visit", "winter")) %>%
  rbind(stat_test(stat_data, "Home Visit", "spring"))

```

```

## Multiple parameters; naming those columns num.df, denom.df
## Multiple parameters; naming those columns num.df, denom.df
## Multiple parameters; naming those columns num.df, denom.df
## Multiple parameters; naming those columns num.df, denom.df

```

```
stat_table
```

```

## # A tibble: 12 x 5
##   appt_mode season test      p_value conclusion
##   <chr>      <chr> <chr>      <dbl> <chr>
## 1 Telephone NA      Wilcoxon 0.0000130 H1 True
## 2 Face-to-Face NA      Wilcoxon 0.815      H0 True
## 3 Home Visit NA      Wilcoxon 0.0984      H0 True
## 4 Telephone summer Students 0.0000678 H1 True
## 5 Telephone winter Wilcoxon 0.0938      H0 True
## 6 Telephone spring Students 0.00366     H1 True

```

```
## 7 Face-to-Face summer Wilcoxon 0.897      H0 True
## 8 Face-to-Face winter Wilcoxon 0.0938     H0 True
## 9 Face-to-Face spring Wilcoxon 0.545      H0 True
## 10 Home Visit  spring Students 0.409      H0 True
## 11 Home Visit  winter Wilcoxon 0.219      H0 True
## 12 Home Visit  spring Students 0.409      H0 True
```

*#Only included by season for face to face as none of the others were
#statistically significant but included the analysis here for interest*

#Only non significant data point in Telephone is winter but not really enough variables to accurately t

The outcome from this table demonstrates that there is a only a statistically significant increase in the number of appointments between the 2018 and 2019 dates for telephone appointment types.

Second assignment Questions

Second question will look at whether booking time affects the percentage of the appointments not attended. Hypothesis is that as time increases so does the number of missed appoints.

```
data_2 <- part_2_data
```

#Checking unique types of booking times

```
data_2 %>%
  select(time_between_book_and_appt) %>%
  unique()
```

```
## # A tibble: 8 x 1
##   time_between_book_and_appt
##   <chr>
## 1 1 Day
## 2 Unknown / Data Issue
## 3 Same Day
## 4 8 to 14 Days
## 5 22 to 28 Days
## 6 2 to 7 Days
## 7 More than 28 Days
## 8 15 to 21 Days
```

#Checking unique types of appt_status

```
data_2 %>%
  select(appt_status) %>%
  unique()
```

```
## # A tibble: 4 x 1
##   appt_status
##   <chr>
## 1 Attended
## 2 Unknown
## 3 DNA
## 4 Appt Status Not Provided
```

*#Discard all data with Unknown / Data Issue and
#Appointment status as Unknown or Appt Status Not Provided
#As only interested in attended/Not attended appointments*

```

data_2 <- data_2 %>%
  filter(!time_between_book_and_appt == "Unknown / Data Issue") %>%
  filter(appt_status %in% c("Attended", "DNA")) %>%
  #Only looking at the GP appointments not attended
  filter(hcp_type == "GP")

#Having a look at mean count
data_2 %>%
  group_by(appt_status, time_between_book_and_appt) %>%
  summarise(c = mean(count))

## # A tibble: 14 x 3
## # Groups:   appt_status [2]
##   appt_status time_between_book_and_appt      c
##   <chr>        <chr>                    <dbl>
## 1 Attended    1 Day                               NA
## 2 Attended    15 to 21 Days                      NA
## 3 Attended    2 to 7 Days                        NA
## 4 Attended    22 to 28 Days                      NA
## 5 Attended    8 to 14 Days                       NA
## 6 Attended    More than 28 Days                  NA
## 7 Attended    Same Day                          NA
## 8 DNA         1 Day                               NA
## 9 DNA         15 to 21 Days                      NA
## 10 DNA        2 to 7 Days                        NA
## 11 DNA        22 to 28 Days                      NA
## 12 DNA        8 to 14 Days                       NA
## 13 DNA        More than 28 Days                  NA
## 14 DNA        Same Day                          NA

#temp variance to join by to get the total number of appointments for each day
total_count <- data_2 %>%
  select(appointment_date, time_between_book_and_appt, count_of_appointments) %>%
  group_by(appointment_date, time_between_book_and_appt) %>%
  summarise(total_count = sum(count_of_appointments)) %>%
  ungroup()

#Getting the data in the correct format for using
appt_count <- data_2 %>%
  select(appointment_date, time_between_book_and_appt, appt_status, count_of_appointments) %>%
  group_by(appointment_date, time_between_book_and_appt, appt_status) %>%
  #Has the counts for each day based on both attended and did not attend
  summarise(count = sum(count_of_appointments)) %>%
  ungroup() %>%
  inner_join(total_count) %>%
  mutate(percentage_attendance = 100 *(count/total_count)) %>%
  #Add factoring in for time between appt for future plotting and test outputs
  #Note couple of them had double white spaces
  mutate(time_between_book_and_appt = fct_relevel(time_between_book_and_appt, c("Same Day", "1 Day", "2
    "8 to 14 Days", "15 to 21 Days", "22 to 28 Days", "More than 28 Days")))

## Joining, by = c("appointment_date", "time_between_book_and_appt")

```

```
#Look at minimum and maximum counts to see difference between
appt_count %>%
```

```
  filter(appt_status == "DNA") %>%
  group_by(time_between_book_and_appt) %>%
  summarise(min_visits = min(total_count),
            max_visits = max(total_count),
            #median_visits = median(total_count),
            min_dna = min(percentage_attendance),
            max_dna = max(percentage_attendance),
            #median_dna = median(percentage_attendance)
  )
```

```
## # A tibble: 7 x 5
##   time_between_book_and_appt min_visits max_visits min_dna max_dna
##   <fct>                     <int>      <int>    <dbl>   <dbl>
## 1 Same Day                   3        4325     1.54    33.3
## 2 1 Day                       8         705     3.17     25
## 3 2 to 7 Days                12       1555     3.33    32.1
## 4 8 to 14 Days                1         714     4.53    100
## 5 15 to 21 Days               1         443     3.40    100
## 6 22 to 28 Days              1         307     2.44    100
## 7 More than 28 Days          1         209     2.17    100
```

```
#DNA = 100 - Attended so no need to store both
```

```
#Variables with < 20 in the number of appointments can lead to a very highly skewed output based on just
#Did not attend
```

```
anova_data <- appt_count %>%
  #Filtering to get did not attend status(reverse for otherone)
  filter(appt_status == "DNA") %>%
  filter(count > 20)
```

```
#Variation assumption
```

```
leveneTest(percentage_attendance ~ time_between_book_and_appt, data = anova_data)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
```

```
##           Df F value    Pr(>F)
## group      6  28.926 < 2.2e-16 ***
##           1090
```

```
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#Since this assumption is NOT valid, it means the anova is not suitable for this data
```

```
#Instead the kruskal-wallis test will be used
```

```
kruskal_time <- kruskal.test(percentage_attendance ~ time_between_book_and_appt, data = anova_data)
```

```
#https://stackoverflow.com/questions/2478272/kruskal-wallis-test-with-details-on-pairwise-comparisons
```

```
#Normally for an anova test a TukeyHSD test would be used to check which ones are not the same
```

```
#E.G have different means
```

```
#For a kruskal wallis test however, Nemenyi-Damico-Wolfe-Dunn test in the package
```

```
library(coin)
```

```
oneway_test(percentage_attendance ~ as.factor(time_between_book_and_appt), data = anova_data)
```

```
##
```

```
## Asymptotic K-Sample Fisher-Pitman Permutation Test
```

```
##
```

```
## data:  percentage_attendance by
```



```
## as.factor(time_between_book_and_appt) (Same Day, 1 Day, 2 to 7 Days, 8 to 14 Days, 15 to 21 Days)
## chi-squared = 842.37, df = 6, p-value < 2.2e-16
```

```
#Seems to be all we need
```

```
library(pgirmess)
```

```
#Looks at which comparisons do NOT have the same mean, individual comparisons
```

```
tukey_time <- kruskalmc(anova_data$percentage_attendence, anova_data$time_between_book_and_appt)
tukey_time
```

```
## Multiple comparison test after Kruskal-Wallis
```

```
## p.value: 0.05
```

```
## Comparisons
```

```
##               obs.dif critical.dif difference
## Same Day-1 Day      239.50283    107.88478      TRUE
## Same Day-2 to 7 Days 419.01606     86.26360      TRUE
## Same Day-8 to 14 Days 651.36842     86.52620      TRUE
## Same Day-15 to 21 Days 707.88069     92.30922      TRUE
## Same Day-22 to 28 Days 822.66095    165.75877      TRUE
## Same Day-More than 28 Days 945.33082   485.11292      TRUE
## 1 Day-2 to 7 Days    179.51323    107.88478      TRUE
## 1 Day-8 to 14 Days   411.86559    108.09487      TRUE
## 1 Day-15 to 21 Days  468.37786    112.77725      TRUE
## 1 Day-22 to 28 Days  583.15812    177.97103      TRUE
## 1 Day-More than 28 Days 705.82799   489.42034      TRUE
## 2 to 7 Days-8 to 14 Days 232.35236     86.52620      TRUE
## 2 to 7 Days-15 to 21 Days 288.86463     92.30922      TRUE
## 2 to 7 Days-22 to 28 Days 403.64489    165.75877      TRUE
## 2 to 7 Days-More than 28 Days 526.31476   485.11292      TRUE
## 8 to 14 Days-15 to 21 Days  56.51227     92.55466     FALSE
## 8 to 14 Days-22 to 28 Days 171.29253    165.89558      TRUE
## 8 to 14 Days-More than 28 Days 293.96240   485.15969     FALSE
## 15 to 21 Days-22 to 28 Days 114.78026    168.98388     FALSE
## 15 to 21 Days-More than 28 Days 237.45013   486.22436     FALSE
## 22 to 28 Days-More than 28 Days 122.66987   505.34059     FALSE
```

```
##Looking at number of suitable results per category to analyse the wider variance of values as x axis
```

```
anova_data %>%
```

```
  group_by(time_between_book_and_appt) %>%
```

```
  summarise(count =n())
```

```
## # A tibble: 7 x 2
```

```
##   time_between_book_and_appt count
```

```
##   <fct>                <int>
```

```
## 1 Same Day                249
```

```
## 2 1 Day                   117
```

```
## 3 2 to 7 Days             249
```

```
## 4 8 to 14 Days            246
```

```
## 5 15 to 21 Days           193
```

```
## 6 22 to 28 Days           39
```

```
## 7 More than 28 Days         4
```

```
plot <- anova_data %>%
```

```
  ggplot(aes(x = time_between_book_and_appt, y = percentage_attendence)) +
```

```
  geom_violin(aes(fill = time_between_book_and_appt)) +
```

```
  geom_boxplot(aes(x = time_between_book_and_appt, y = percentage_attendence), width = 0.10) +
```

```

geom_violin(fill = NA) +
theme_bw() +
labs(y = "Percentage of Appointments Not Attended", x = "Appointment Waiting Time") +
#ggtitle("Percentage of Appointments Not Attended vs Appointment Waiting Time") +
theme(legend.position = "none") +
#Allowing labels to go onto 2 lines if required
scale_x_discrete(labels = function(x) str_wrap(x, width = 10))

#Changing axis ready for report
violin_plot <- plot + theme(axis.text=element_text(size=12),
  axis.title=element_text(size=14),
  plot.title = element_text(size = 22))

#Violin plot for html
violin_plot <- plot

##Number of DNAs by average number of days + 1 (so as to not divide by 0)
anova_data %>%
  group_by(time_between_book_and_appt) %>%
  summarise(median = median(percentage_attendance)) %>%
  ungroup() %>%
  #Adding +1 to account for same day and giving more than 28 a value of 33
  #to match increase from of 7 between previous
  cbind(ave_day = c(1 , 2, 5.5, 12, 19, 26, 33)) %>%
  mutate(per_day = median/ave_day) %>%
  mutate(per_thousand = median * 10)

```

##	time_between_book_and_appt	median	ave_day	per_day	per_thousand
## 1	Same Day	2.751376	1.0	2.7513757	27.51376
## 2	1 Day	6.366048	2.0	3.1830239	63.66048
## 3	2 to 7 Days	7.908992	5.5	1.4379986	79.08992
## 4	8 to 14 Days	10.075971	12.0	0.8396643	100.75971
## 5	15 to 21 Days	10.460251	19.0	0.5505395	104.60251
## 6	22 to 28 Days	12.138728	26.0	0.4668742	121.38728
## 7	More than 28 Days	15.332945	33.0	0.4646347	153.32945

Extra plots

Implemented a bar chart to showcase difference between seasonal values - using face to face appointments as that had the most obvious difference which is worth highlighting.

```

#Bar chart for season values ~ look at median - to account for bit drop offs or increases.
bar_df <- stat_data %>%
  filter(appt_mode == "Face-to-Face") %>%
  group_by(season_median) %>%
  summarise(count = median(count),
    n = n()) %>%
  arrange(desc(count)) %>%
  #Refactoring to get high to low on the graph
  mutate(season_median = factor(season_median,
    levels = c("autumn", "winter", "spring", "summer")))

#Base R implementation - not as good just interesting to see how it works in comparison

```

```

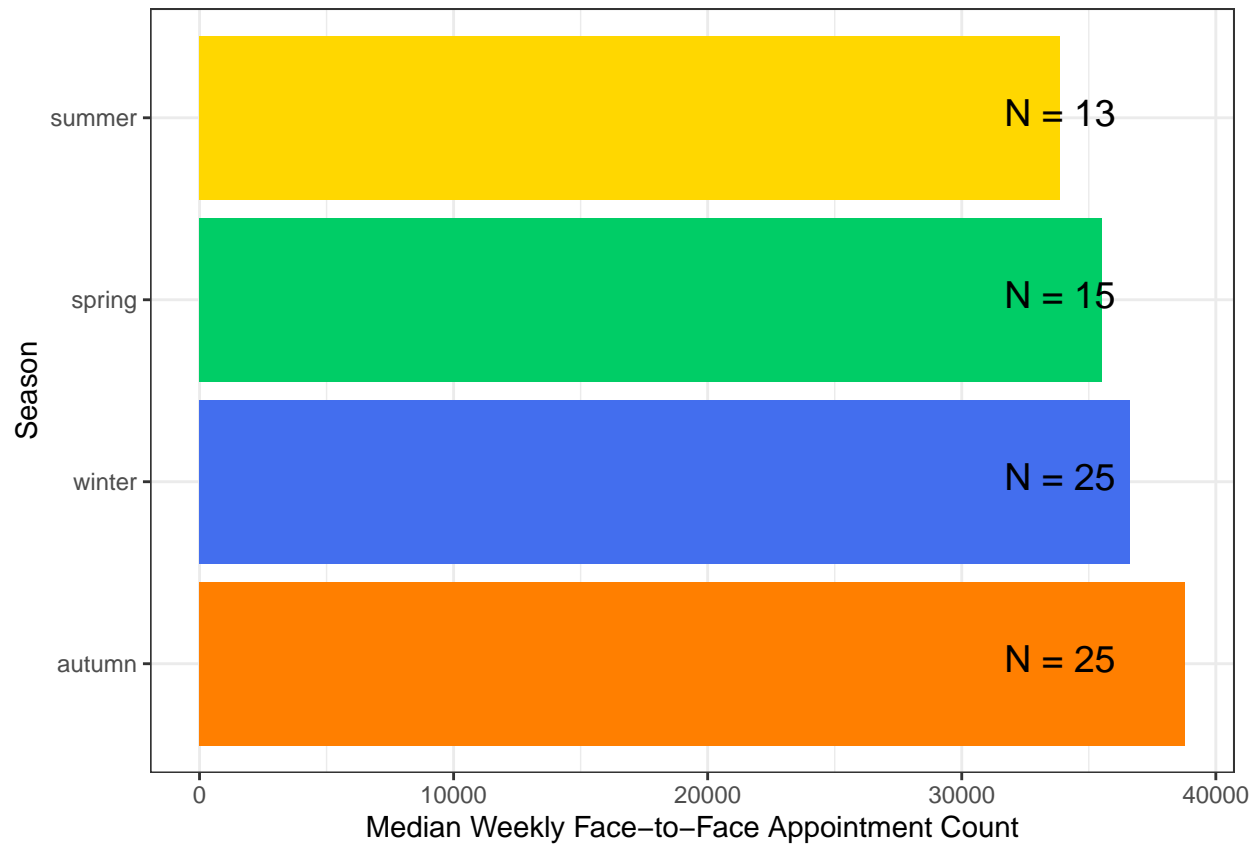
# barchart <- barplot(height = bar_df$count, names = bar_df$season_median,
#                     horiz = T,
#                     xlim = c(0, 40000),
#                     col = c("darkorange1", "royalblue2", "springgreen3", "gold"),
#                     las = 1,
#                     main = "Median Weekly Appointment Count by Season",
#                     xlab = "Median Weekly Count", ylab = "Season")

#GGplot implementation is much nicer
labels <- data_frame(
  x = c(1.1, 2.1, 3.1, 4.1),
  y = rep(1, 4),
  label = c("N = 25", "N = 25", "N = 15", "N = 13"))

bar_plot <- bar_df %>%
  ggplot( aes(count, x = season_median)) +
  geom_bar(stat="identity", position = "dodge", aes(fill = season_median)) +
  coord_flip() +
  #Colours to match seasons
  scale_fill_manual("Season", values = c("darkorange1", "royalblue2",
                                          "springgreen3", "gold" )) +

  theme_bw() +
  theme(legend.position = "none") +
  #title = "Median Weekly Face-to-Face Appointment Count by Season",
  labs(y = "Median Weekly Face-to-Face Appointment Count",
       x = "Season") +
  geom_text(data = labels, aes(x = x, y = y, label = label), vjust = 1, hjust = -7.25, size = 5)
bar_plot

```



```
#Saving the plots so it can be easily loaded and plotted for the .html file. (commented out as no point  
#save(telephone_plot, violin_plot, bar_plot, file = "html.RData")
```