St344 Individual Coursework U1619685

Stephen Brownsey

```r
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------------

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```r
library(broom)
```

```
## Warning: package 'broom' was built under R version 3.6.2
```

```r
library(car)
```

```
## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some
```

```r
library(coin)
```

```
## Warning: package 'coin' was built under R version 3.6.2

## Loading required package: survival
```

```r
library(pgirmess)
```

```
## Warning: package 'pgirmess' was built under R version 3.6.2
```

```r
library(Hmisc)
```

```
## Warning: package 'Hmisc' was built under R version 3.6.2

## Loading required package: lattice

## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
library(rio)
```

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 in terms of datasets and provided a good base for exploratory data analysis in particuar for pointing out the effect of different days.

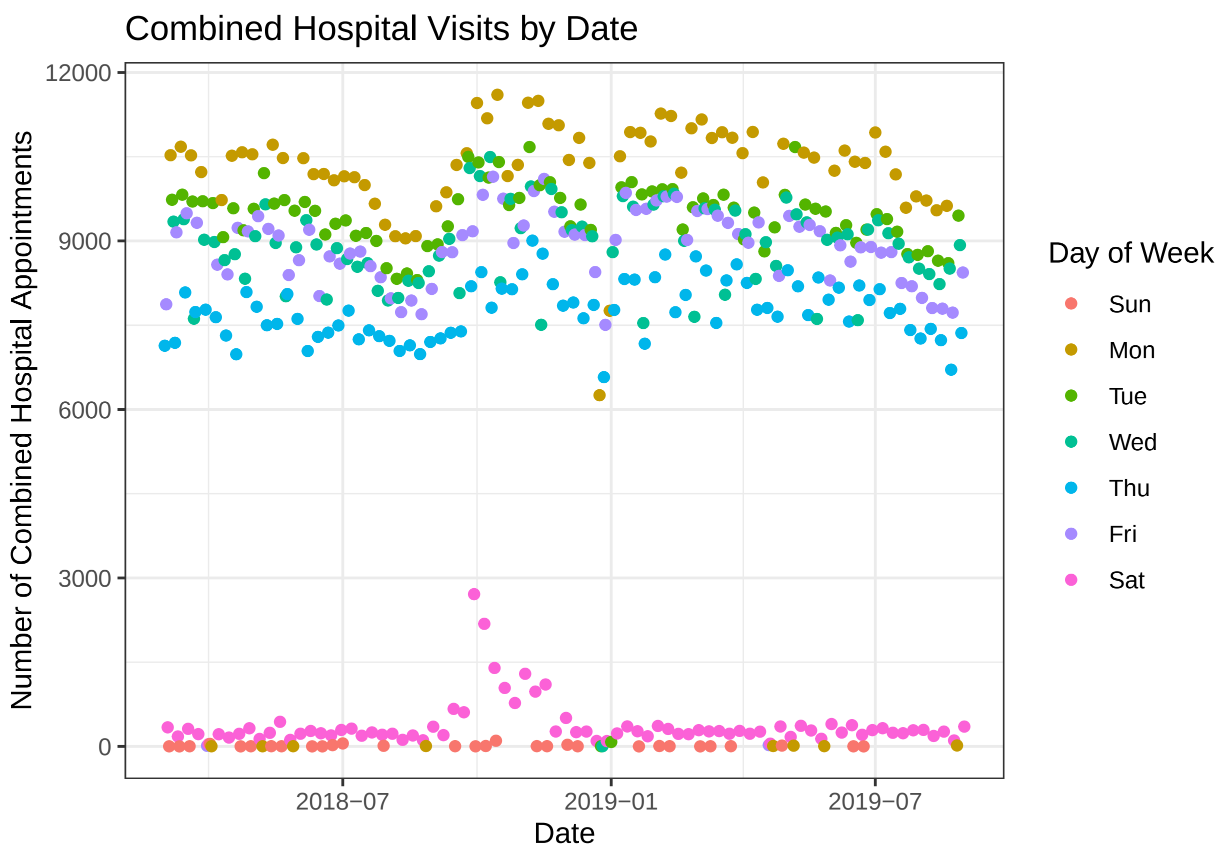
```
#function to lower case column names as in general it is better and easier to refer to
#everything in the lower case
hadley_format <- function(data){
  for (i in 1:length(colnames(data))) {
  colnames(data)[i] = tolower(colnames(data)[i])
  }
  data
}
```

```
#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"))
}
```

```r
#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_API
                  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()
```




```r
#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

INSERT R Markdown intro: 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. There has also been no subsetting based on hcp_type -> might need to add that in potentially.

Want really to compare results for which we have two measurements for in almost a paired t-test but only 18 months is slightly problematic. . .

```
#Graphing function for visual aids on the appointment counts
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  + week_num + season_median, 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() +
    labs(title = paste0(filter, " Weekly Appointment Count with Poisson Regression Line"),
         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()  +
    labs(title = paste0("Weekly Appointment Count by Appointment Type"),
         x = "Week Number", y = "Weekly Appointment Count", colour = "Appointment Type")
  }

  output

}
```

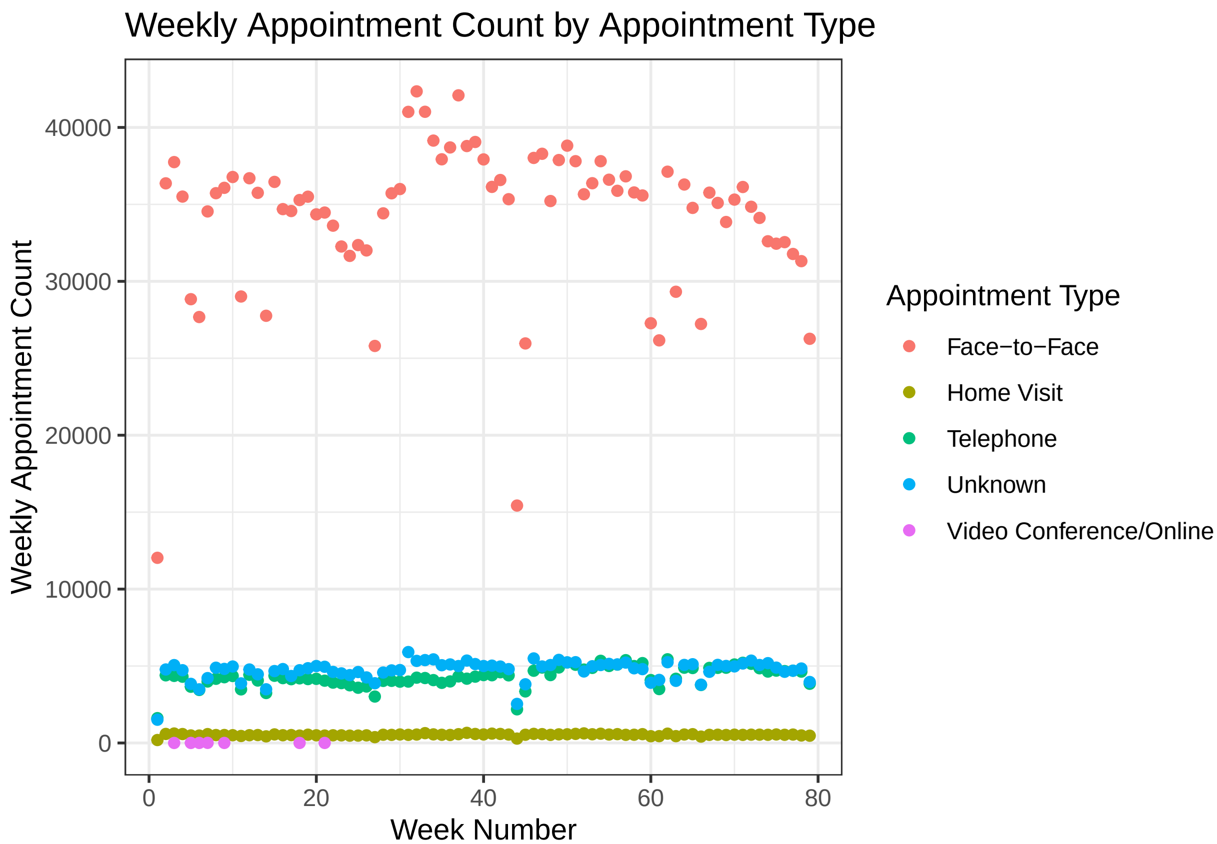
```
#Seasons data as per: https://www.timeanddate.com/calendar/seasons.html
seasons <- tibble(season = c("winter", "spring", "summer","autumn",
                             "winter", "spring", "summer"),
                  start_date = c(as.Date("2017-12-21"), as.Date("2018-03-20"),
                                 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)) %>%
```

```r
  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"
```

```r
#Saving the data before I remove the anonyamous 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()
```

```r
#General plot for understanding
appt_by_mode %>%
  appt_count()
```




```r
#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
```

```r
#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
```

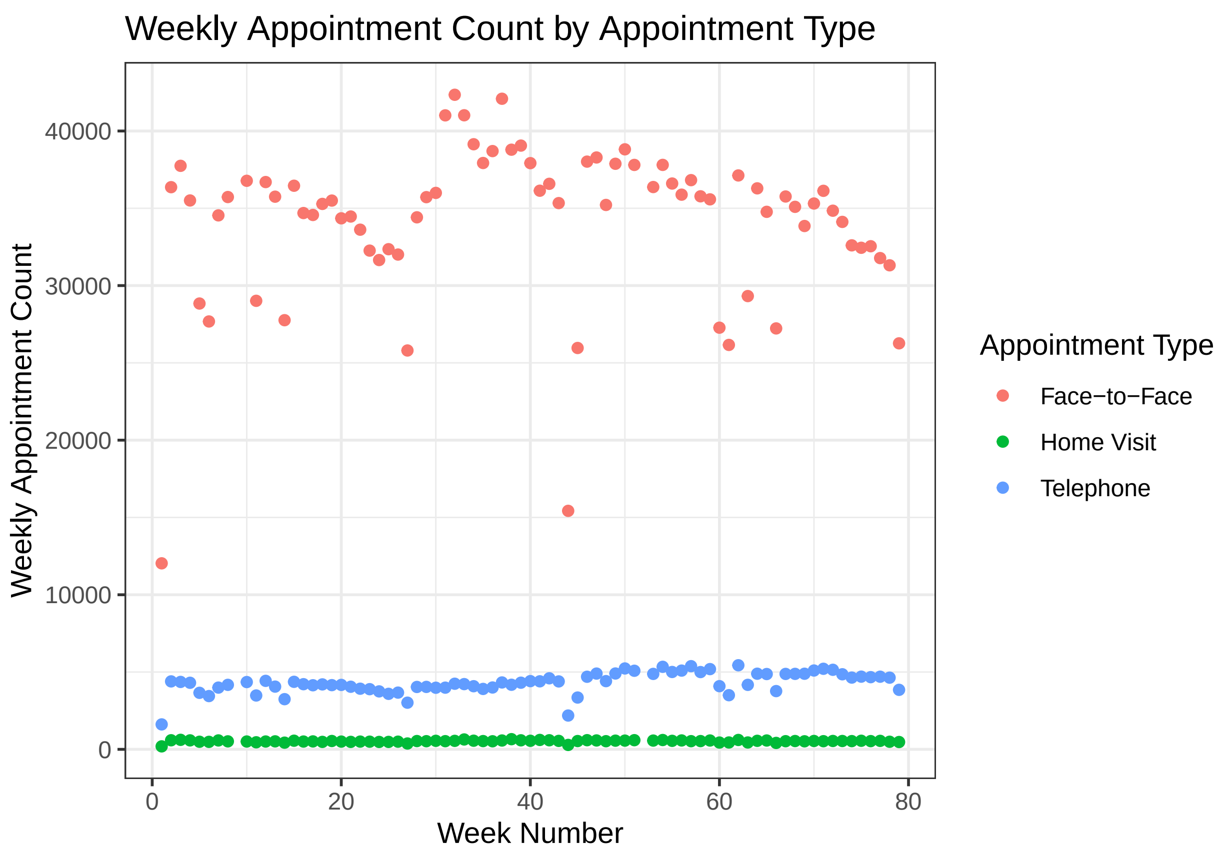
```
##   5 2018-04-27
##   6 2018-04-28
##   7 2018-04-29
##   8 2019-02-18
##   9 2019-02-19
##  10 2019-02-20
##  11 2019-02-21
##  12 2019-02-22
##  13 2019-02-23
```

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

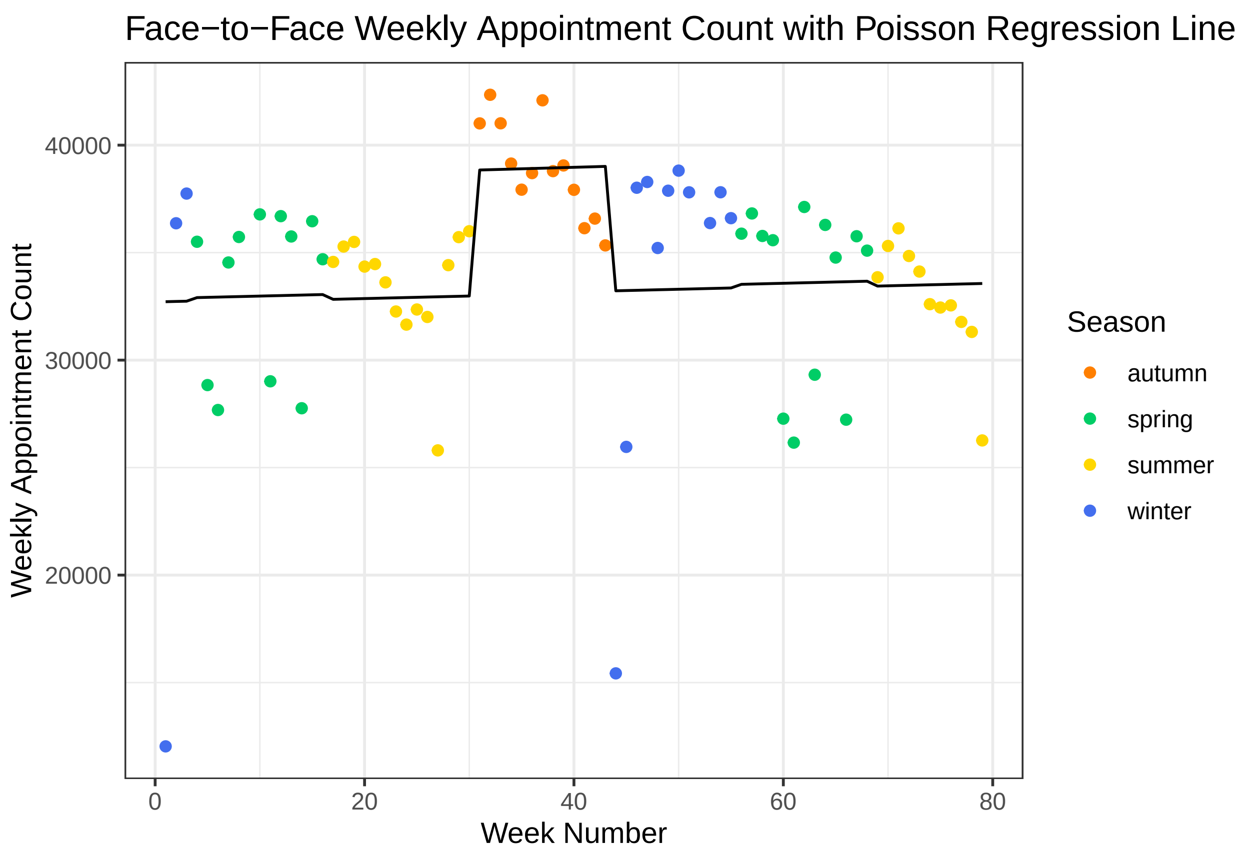
```
## # A tibble: 0 x 20
## # ... with 20 variables: appointment_date <date>, appt_status <chr>,
## #   hcp_type <chr>, appt_mode <chr>, time_between_book_and_appt <chr>,
## #   ccg_code <chr>, count_of_appointments <int>, appointment_data <date>,
## #   stpcd <chr>, regional_local_office_code <chr>, region_code <chr>,
## #   appointment_month <dttm>, `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>
```

```r
##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"))
```

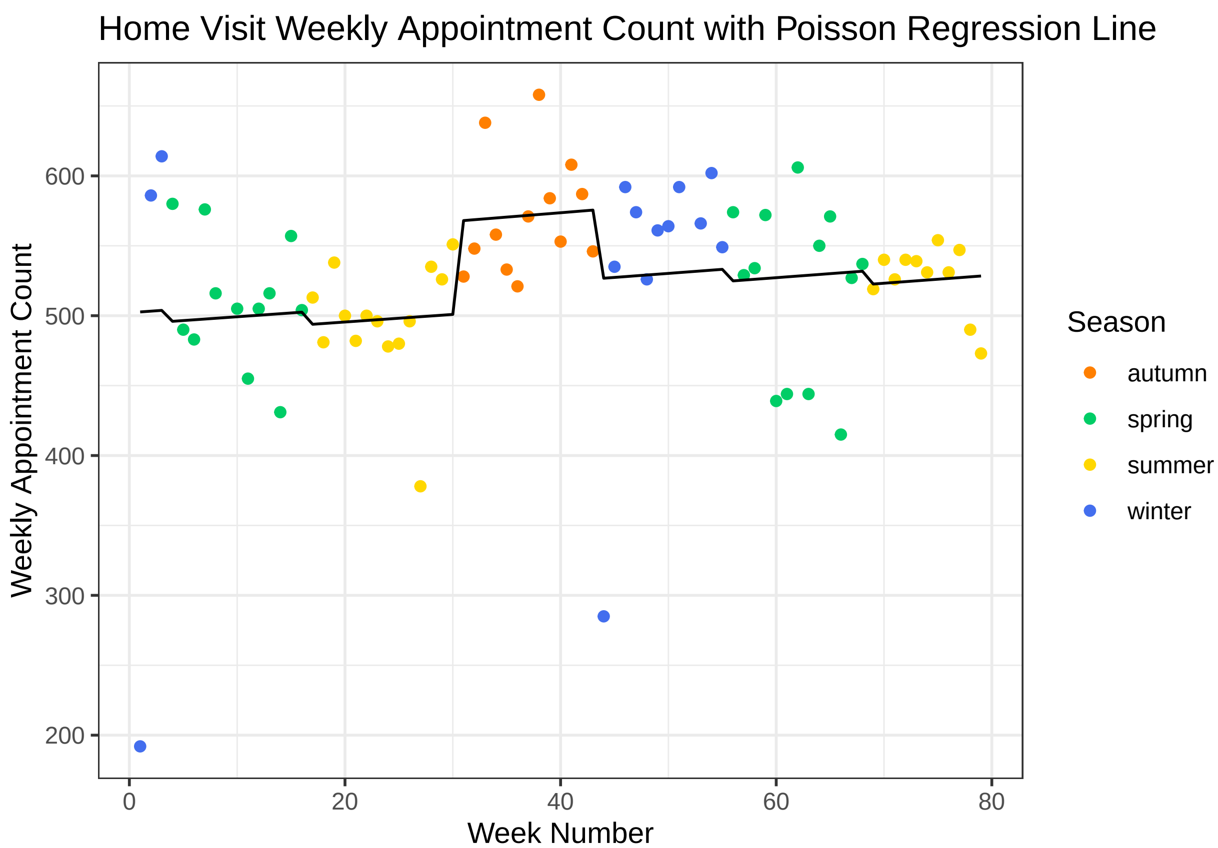
```r
#Graphically exploring the different visit types and
appt_by_mode <- data %>%
  group_by(appt_mode, week_num, season_median) %>%
  summarise(count = sum(count_of_appointments)) %>%
  ungroup()

appt_by_mode %>%
  appt_count()
```



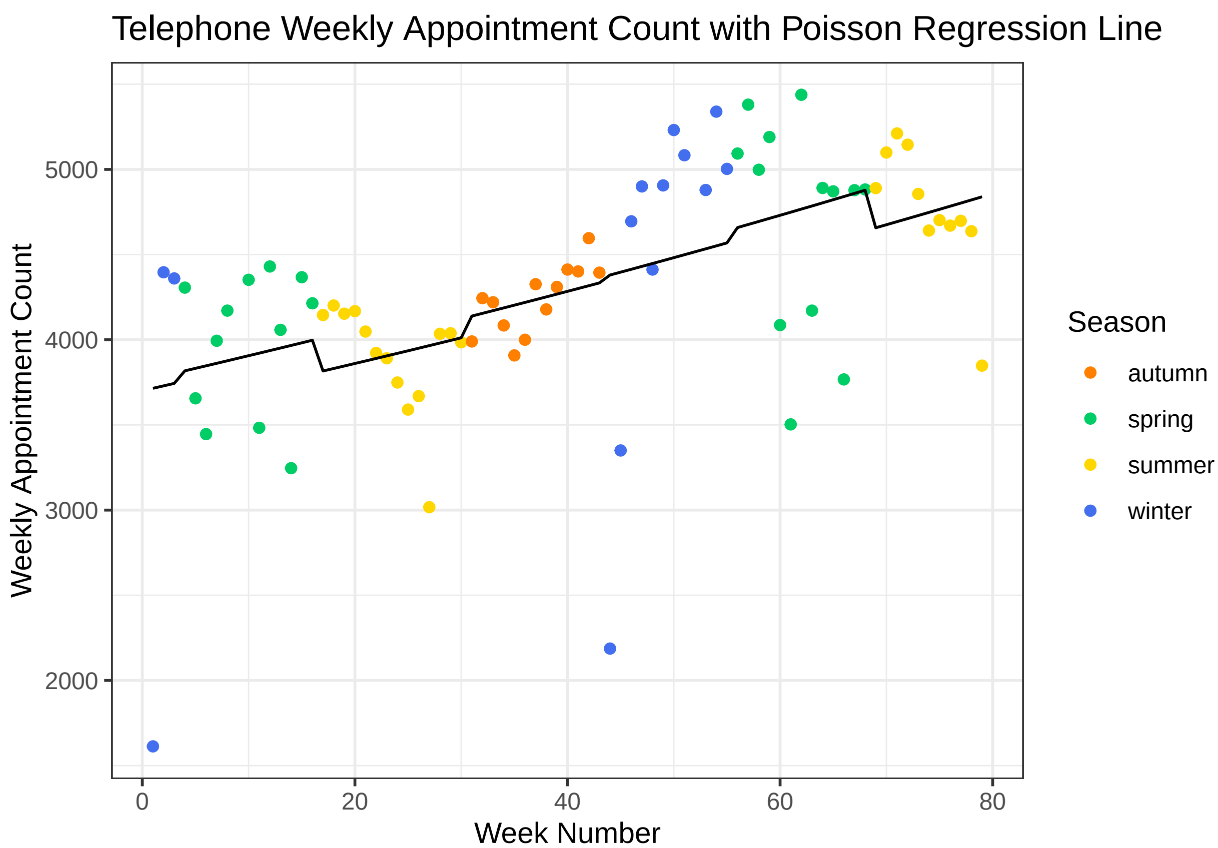

```
appt_by_mode %>%
  appt_count("Face-to-Face")
```




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




```
appt_by_mode %>%
  appt_count("Telephone")
```




```r
#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")
```

```r
model1 <- glm(count ~ 0  + week_num, family = "poisson", data = appt_by_mode %>%
              filter(appt_mode == "Face-to-Face"))
model2 <- glm(count ~ 0  + week_num + season_median, family = "poisson", data = appt_by_mode %>%
              filter(appt_mode == "Face-to-Face"))
# summary(glm(count ~  week_num , family = "poisson", data = appt_by_mode %>%
#                filter(appt_mode == "Face-to-Face")))
# summary(glm(count ~  week_num + season_median, family = "poisson", data = appt_by_mode %>%
#                filter(appt_mode == "Face-to-Face")))
# summary(glm(count ~ 0  + week_num + season_median, family = "poisson", data = appt_by_mode %>%
#                filter(appt_mode == "Face-to-Face")))


model3 <-  glm(count ~ 0  + week_num, family = "poisson", data = appt_by_mode %>%
              filter(appt_mode == "Telephone"))
model4 <- glm(count ~ 0  + week_num + season_median, family = "poisson", data = appt_by_mode %>%
              filter(appt_mode == "Telephone"))
model3 %>% 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
## -158.65    46.62   142.10   185.57   250.44
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## week_num 1.273e-01  3.141e-05    4053   <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: 4890202  on 77  degrees of freedom
## Residual deviance: 1722731  on 76  degrees of freedom
## AIC: 1723517
##
## Number of Fisher Scoring iterations: 7
```

```
model4 %>% summary()
```

```
##
## Call:
## glm(formula = count ~ 0 + week_num + season_median, family = "poisson",
##     data = appt_by_mode %>% filter(appt_mode == "Telephone"))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -38.892   -2.330    1.258    5.884   11.368
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## week_num            3.829e-03  7.789e-05   49.16   <2e-16 ***
## season_medianautumn 8.210e+00  5.147e-03 1595.06   <2e-16 ***
## season_medianspring 8.232e+00  4.329e-03 1901.47   <2e-16 ***
## season_mediansummer 8.182e+00  4.841e-03 1690.09   <2e-16 ***
## season_medianwinter 8.216e+00  5.152e-03 1594.77   <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: 4890202.1  on 77  degrees of freedom
## Residual deviance:    6291.9  on 72  degrees of freedom
## AIC: 7086.5
##
## Number of Fisher Scoring iterations: 4
```

```
# summary(glm(count ~ 0  + week_num + season_median, family = "poisson", data = appt_by_mode %>%
#                filter(appt_mode == "Home Visit")))
```

```r
model1 %>% 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
## -506.6   152.0   477.4   630.5   827.6
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## week_num 1.551e-01  1.131e-05   13710   <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: 49757289  on 77  degrees of freedom
## Residual deviance: 19742887  on 76  degrees of freedom
## AIC: 19743833
##
## Number of Fisher Scoring iterations: 8
```

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

```
##
## Call:
## glm(formula = count ~ 0 + week_num + season_median, family = "poisson",
##     data = appt_by_mode %>% filter(appt_mode == "Face-to-Face"))
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -131.524    -5.875     8.147    14.953    29.465
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## week_num            3.577e-04  2.785e-05   12.84   <2e-16 ***
## season_medianautumn 1.056e+01  1.743e-03 6056.56   <2e-16 ***
## season_medianspring 1.040e+01  1.510e-03 6885.48   <2e-16 ***
## season_mediansummer 1.039e+01  1.686e-03 6163.74   <2e-16 ***
## season_medianwinter 1.040e+01  1.830e-03 5679.63   <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: 49757289  on 77  degrees of freedom
## Residual deviance:    54940  on 72  degrees of freedom
## AIC: 55895
##
## Number of Fisher Scoring iterations: 4
```

```r
#No idea why no pr > Chisq??? No idea what this is doing :/
anova(glm(count ~ 0 + week_num , family = "poisson", data = appt_by_mode %>%
              filter(appt_mode == "Face-to-Face")),
      glm(count ~0 +  week_num +season_median, family = "poisson", data = appt_by_mode %>%
        filter(appt_mode == "Face-to-Face")))
```

```
## Analysis of Deviance Table
##
## Model 1: count ~ 0 + week_num
## Model 2: count ~ 0 + week_num + season_median
##   Resid. Df Resid. Dev Df Deviance
## 1        76   19742887
## 2        72      54940  4 19687947
```

```r
#Basically shows model 2 with season so much better
BIC(model1)
```

```
## [1] 19743836
```

```r
BIC(model2)
```

```
## [1] 55906.42
```

Fitting a poison model to the data as in the lab

Doesn't work at the moment can delete this section later as implemented into function.

```r
# po_data <- appt_by_mode
#
# my_model <- glm(count ~ 0  + week_num + season_median + appt_mode, family = "poisson", data = po_data
# summary(my_model)
#
# poisson_data <- po_data %>%
#   mutate(poisson_estimate = predict(my_model, type = "response")) %>%
#   ggplot() +
#   geom_point(aes(x = week_num, y = count), colour = "red") +
#   geom_line(aes(x = week_num, y = poisson_estimate))
#   +
#   geom_jitter(aes(x = week_num, y = poisson_estimate), colour = "blue")
#
#   +
#   smooth.spline(poisson_estimate)
# poisson_data
#
#
#
# #Massively overfitted to be honest
# po_data <- appt_by_mode %>%
#    mutate(year_week = as_factor(if_else(week_num > 52, week_num - 52, week_num))) %>%
#   filter(appt_mode == "Telephone")
# my_model_1 <- glm(count ~ 0  + week_num + season_median, family = "poisson", data = po_data)
# my_model_2 <- glm(count ~ 0  + week_num + season_median  + year_week, family = "poisson", data = po_d
#
# poisson_data <- po_data %>%
#   mutate(poisson_estimate = predict(my_model_1, type = "response")) %>%
```

```r
#   mutate(overfitted_estimate = predict(my_model_2, type = "response")) %>%
#   ggplot() +
#   geom_point(aes(x = week_num, y = count), colour = "red") +
#   geom_line(aes(x = week_num, y = poisson_estimate))# +
#   #geom_line(aes(x = week_num, y = overfitted_estimate))
# +
#   geom_jitter(aes(x = week_num, y = poisson_estimate), colour = "blue")
# poisson_data
#
#
# summary(my_model)
```

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.

```r
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)
}

checking_seasons(data, "Face-to-Face") %>%
  bind_rows(checking_seasons(data, "Telephone")) %>%
  bind_rows(checking_seasons(data, "Home Visit"))
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.

## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.

## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.

## # 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. Obviously the cold weather will 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.

```r
#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),
```

```r
                    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
```

```r
}

#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"))
```

```
## Warning in wilcox.test.default(test_data$year_2, test_data$year_1, paired =
## TRUE, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(test_data$year_2, test_data$year_1, paired =
## TRUE, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(test_data$year_2, test_data$year_1, paired =
## TRUE, : cannot compute exact p-value with ties

## 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
```

```r
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
```

```r
#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

```
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 Uknown 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")

data_2 %>%
  group_by(appt_status, time_between_book_and_appt) %>%
  summarise(c = mean(count))
```

```
## Warning in mean.default(count): argument is not numeric or logical:
## returning NA

## Warning in mean.default(count): argument is not numeric or logical:
## returning NA

## Warning in mean.default(count): argument is not numeric or logical:
```

```
## returning NA

## Warning in mean.default(count): argument is not numeric or logical:
## returning NA

## Warning in mean.default(count): argument is not numeric or logical:
## returning NA

## Warning in mean.default(count): argument is not numeric or logical:
## returning NA

## Warning in mean.default(count): argument is not numeric or logical:
## returning NA

## Warning in mean.default(count): argument is not numeric or logical:
## returning NA

## Warning in mean.default(count): argument is not numeric or logical:
## returning NA

## Warning in mean.default(count): argument is not numeric or logical:
## returning NA

## Warning in mean.default(count): argument is not numeric or logical:
## returning NA

## Warning in mean.default(count): argument is not numeric or logical:
## returning NA

## Warning in mean.default(count): argument is not numeric or logical:
## returning NA

## Warning in mean.default(count): argument is not numeric or logical:
## returning NA

## # 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
```

```r
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()

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) %>%
  summarise(count = sum(count_of_appointments)) %>%
  ungroup() %>%
  inner_join(total_count) %>%
  mutate(percentage_attendence = 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")
```

```r
#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 jus
#Did not attends
anova_data <- appt_count %>%
  filter(appt_status == "DNA") %>%
  filter(count > 20)

#Variation assumption
leveneTest(percentage_attendence ~ 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
```

```r
#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_attendence ~ 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_attendence ~ as.factor(time_between_book_and_appt), data = anova_data)
```

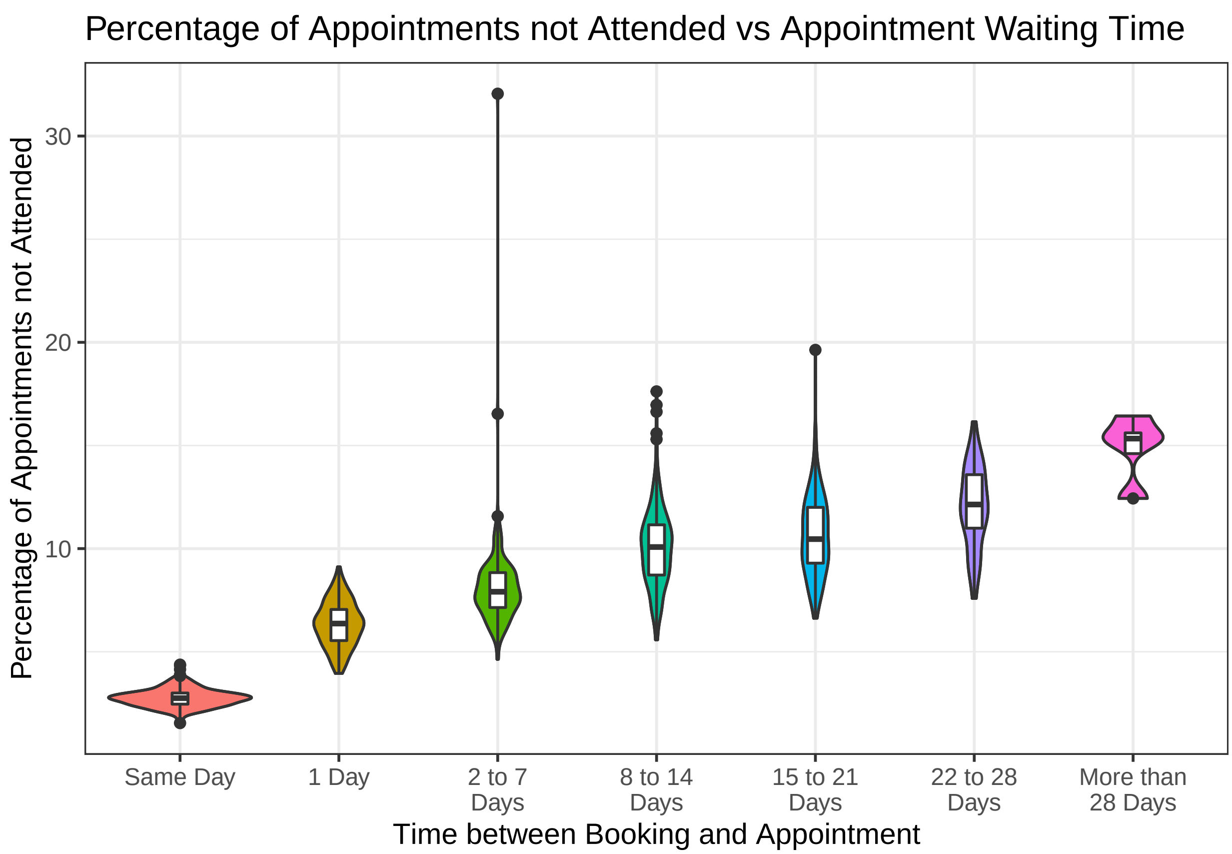
```
##
##  Asymptotic K-Sample Fisher-Pitman Permutation Test
##
## data:  percentage_attendence by
##   as.factor(time_between_book_and_appt) (Same Day, 1 Day, 2 to 7 Days, 8  to 14 Days, 15  to 21 Days
## chi-squared = 842.37, df = 6, p-value < 2.2e-16
```

```r
#Seems to be all we need
library(pgirmess)
```

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

```r
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 = "Time between Booking and Appointment",
       title = "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))
plot
```




```
#http://www.sthda.com/english/wiki/ggplot2-violin-plot-quick-start-guide-r-software-and-data-visualizat
#https://katherinemwood.github.io/post/violins/
```