clustering

Stephen Brownsey

25/12/2019

2 step markov chain I guess

This section is going cover two step clustering based on various clustering methods for comparison. Visit data will be combined into one bee result for each orchard, note in principle this is fundementally flawed as the data does depend on previous years and the assumption that independent of the past is false as pesticide last year affects bee population this year as would be expected. In these scenarios each implementation will have a maximum of 8 clusters. Two after pre-bloom, 4 after and 8 after post bloom.

```r
load("data")
markov_data <- data_2012 %>%
  select(ends_with(".pre"), ends_with(".blm"), ends_with(".pos"), orchard) %>%
  unique()

#function to generate the agglomaerative clustering
aggl <- function(data){
  agnes(dist(data, method = "euclidian"),
                  diss=TRUE, method = "ward")
}

#function to reduce copy pasting for each node
aggl_c <- function(data, ends, c_num){
  #define temp dataset of this stage by cluster
    temp <- data %>%
    filter(cluster == c_num) %>%
    select(ends_with(ends))

#If statement as if 1 element in dataset it'll error
if(nrow(temp) > 1){
  temp$cluster <- cutree(aggl(temp) , k = 2)

   output <- data %>%
    filter(cluster == c_num) %>%
    select(-cluster) %>%
    mutate(cluster = temp$cluster)
   #Else statement for if 1 element in dataset
  }else{
    #If only 1 element in the dataset -> set the cluster number to 1
    output <- data %>%
      filter(cluster == c_num) %>%
      select(-cluster) %>%
      mutate(cluster = 1)
  }
  output
}


####Agglomerative heirachicale clustering
```

```
#pre-bloom step
temp <- markov_data %>%
  select(ends_with(".pre"))

temp$cluster <- cutree(aggl(temp) , k = 2)

aggl_node1 <- markov_data %>%
  mutate(cluster = temp$cluster)

#during bloom step 1
aggl_node2 <- aggl_c(aggl_node1, ".blm", 1)

#during bloom step 2
aggl_node3 <- aggl_c(aggl_node1, ".blm", 2)


#post bloom step 1
aggl_node4 <- aggl_c(aggl_node2, ".pos", 1)

#post bloom step 2
aggl_node5 <- aggl_c(aggl_node2, ".pos", 2)

#post bloom step 3
aggl_node6 <- aggl_c(aggl_node3, ".pos", 1)

#post bloom step 4
#Note this node only has 1 element in it so it is automatically set to 1
aggl_node7 <- aggl_c(aggl_node3, ".pos", 2)

orchard_node_agglom <- tibble(orchard = aggl_node4 %>% filter(cluster == 1) %>% select(orchard) %>% as_
  bind_rows(tibble(orchard = aggl_node4 %>% filter(cluster == 2) %>% select(orchard) %>% as_vector, nod
  bind_rows(tibble(orchard = aggl_node5 %>% filter(cluster == 1) %>% select(orchard) %>% as_vector, nod
  bind_rows(tibble(orchard = aggl_node5 %>% filter(cluster == 2) %>% select(orchard) %>% as_vector, nod
  bind_rows(tibble(orchard = aggl_node6 %>% filter(cluster == 1) %>% select(orchard) %>% as_vector, nod
  bind_rows(tibble(orchard = aggl_node6 %>% filter(cluster == 2) %>% select(orchard) %>% as_vector, nod
  bind_rows(tibble(orchard = aggl_node7 %>% filter(cluster == 1) %>% select(orchard) %>% as_vector, nod
```

kmeans approach: Even running the 1001 times it is different everytime: store output later on but run it like 1million times at each stage.

```
#function to generate the kmeans clustering
k_clustering <- function(data, n = 1001){
#creating the dataset to then calculate the optimal cluster from
  for(i in 1:n){
  if(i == 1){
     k_list <- tibble(kmeans(data, 2)$cluster)
  }else{
     k_list <- bind_cols(k_list, tibble(kmeans(data, 2)$cluster))
  }

  }
  apply(k_list[ ,1:length(k_list)], 1, mfv1) %>%
    enframe(value = "cluster") %>%
    select(cluster)
```

```r
}
#pre-bloom step
#Getting the most common clustering

kmeans_c <- function(data, ends, c_num){

  data <- data %>%
    filter(cluster == c_num) %>%
    select(-cluster)

  if(nrow(data) > 2){
    temp <- data  %>%
    select(ends_with(ends))

  cluster <- k_clustering(temp, 2)
 output <- bind_cols(data, cluster)
  }else{
    #Not more than 2 rows and get the error message as:
    #number of cluster centres must lie between 1 and nrow(x)
    output <- data %>%
      mutate(cluster = 1)
  }

 output
}


temp <- k_clustering(markov_data %>% select(-orchard))

kmeans_node1 <- markov_data %>%
  bind_cols(temp)


#during bloom step 1
kmeans_node2 <- kmeans_c(kmeans_node1, ".blm", 1)

#during bloom step 2
kmeans_node3 <- kmeans_c(kmeans_node1, ".blm", 2)


#post bloom step 1
kmeans_node4 <- kmeans_c(kmeans_node2, ".pos", 1)

#post bloom step 2
kmeans_node5 <- kmeans_c(kmeans_node2, ".pos", 2)

#post bloom step 3
kmeans_node6 <- kmeans_c(kmeans_node3, ".pos", 1)

#post bloom step 4
kmeans_node7 <- kmeans_c(kmeans_node3, ".pos", 2)

orchard_node_kmeans <- tibble(orchard = kmeans_node4 %>% filter(cluster == 1) %>% select(orchard) %>% a
  bind_rows(tibble(orchard = kmeans_node4 %>% filter(cluster == 2) %>% select(orchard) %>% as_vector, no
  bind_rows(tibble(orchard = kmeans_node5 %>% filter(cluster == 1) %>% select(orchard) %>% as_vector, no
  bind_rows(tibble(orchard = kmeans_node5 %>% filter(cluster == 2) %>% select(orchard) %>% as_vector, no
  bind_rows(tibble(orchard = kmeans_node6 %>% filter(cluster == 1) %>% select(orchard) %>% as_vector, no
  bind_rows(tibble(orchard = kmeans_node6 %>% filter(cluster == 2) %>% select(orchard) %>% as_vector, no
  bind_rows(tibble(orchard = kmeans_node7 %>% filter(cluster == 2) %>% select(orchard) %>% as_vector, no
```

Just a look at the difference in the final clusters between the two methods

```
clusterings <- orchard_node_agglom %>%
  arrange(orchard) %>%
  bind_cols(node_km = orchard_node_kmeans %>% arrange(orchard) %>%
              select(node) %>% as_vector)
clusterings
```

```
## # A tibble: 19 x 3
##    orchard  node node_km
##    <fct>   <dbl>   <dbl>
##  1 A           8      10
##  2 B           9      10
##  3 C           9      10
##  4 D           9      10
##  5 E          12      12
##  6 F          13      13
##  7 G          10       8
##  8 H.nooil    11       9
##  9 I           8      10
## 10 J           8       9
## 11 K           8      10
## 12 L           8      10
## 13 M           8      10
## 14 N          14      12
## 15 O.nooil    12      12
## 16 P           8      10
## 17 Q           8       9
## 18 R           8      10
## 19 S          11       9
```

Updating the bee values to match the environments

In this particular analysis, the interest is in the affects of pesticides on bee population. As such, it is necessary to correct, as best we can, for extra factors (confounding?) to make the bee values representative.

All these graphs were plotted as part of EDA by the use of lapply to the a basic plotting function. Here the relationships between some of the main extra factors (are they confounding?) will be examined.

Temperature

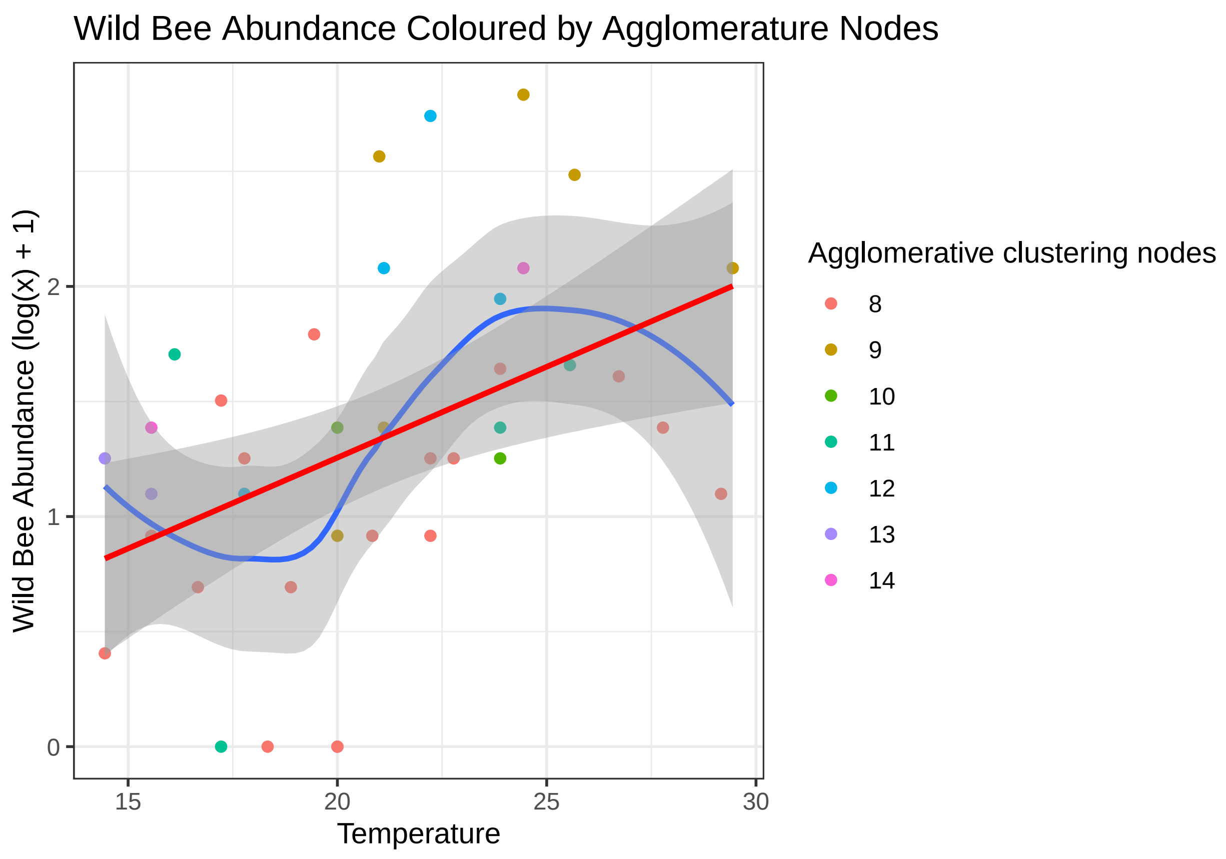
It is known that temperature affects the speed at which bees fly, as such this will have an effect on bee abundance and possibly richness although more bees might not necessarily mean more species are observed. From the previous analysis carried out it is known that a log(x) + 1 transposition of bee count allows for a "better" model fit and as such in general this will the case for our observations.

```
##agglom
wild_bee_abundance_agglom <- data_2012 %>%
  inner_join(orchard_node_agglom) %>%
  ggplot(aes(x = temp, y = log(wildAbF + 1))) +
  geom_point(aes(colour = as_factor(node))) +
  geom_smooth() +
  geom_smooth(method = "lm", colour = "red") +
  labs(x = "Temperature", y = "Wild Bee Abundance (log(x) + 1)",
       title = "Wild Bee Abundance Coloured by Agglomerature Nodes",
       colour = "Agglomerative clustering nodes") +
  theme_bw()
```

```
## Joining, by = "orchard"
```

```
wild_bee_abundance_agglom
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

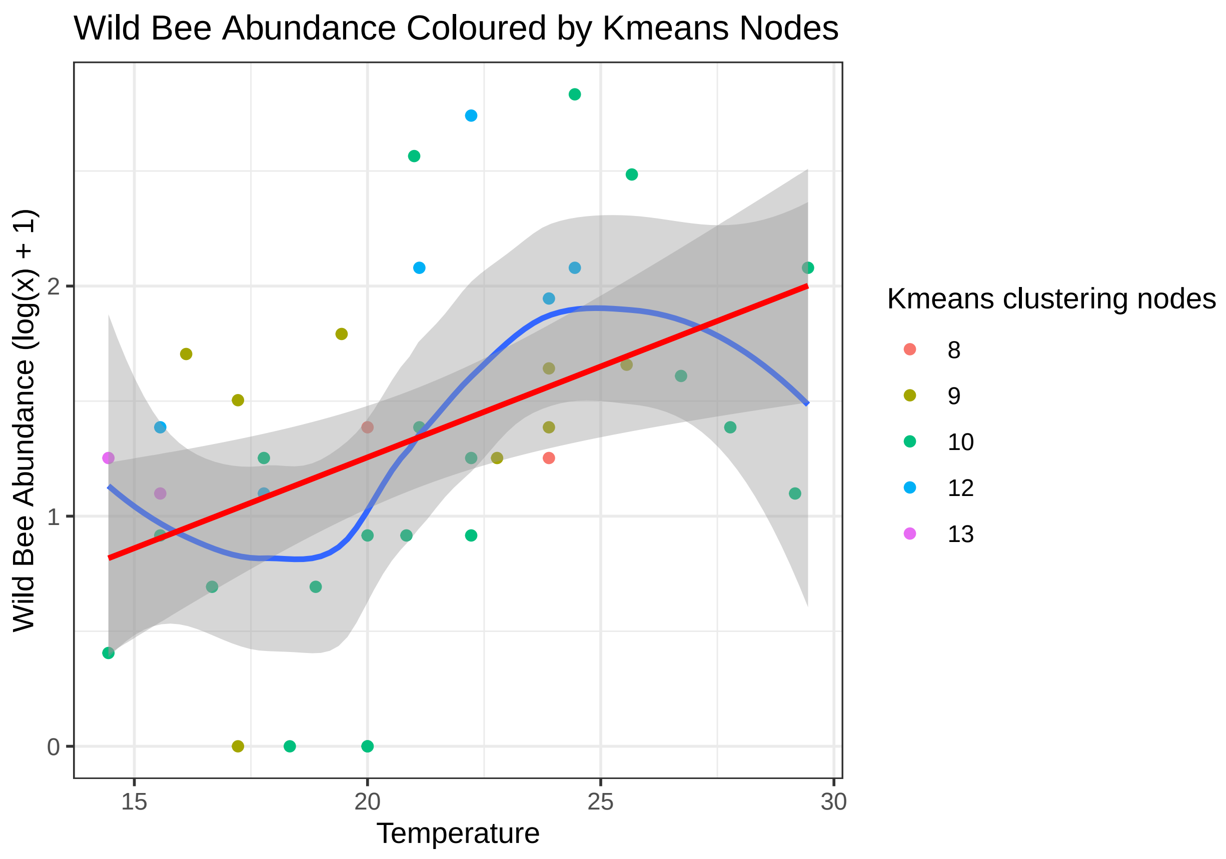


```
##kmeans
wild_bee_abundance_kmeans <- data_2012 %>%
  inner_join(orchard_node_kmeans) %>%
  ggplot(aes(x = temp, y = log(wildAbF + 1))) +
  geom_point(aes(colour = as_factor(node))) +
  geom_smooth() +
  geom_smooth(method = "lm", colour = "red") +
  labs(x = "Temperature", y = "Wild Bee Abundance (log(x) + 1)",
       title = "Wild Bee Abundance Coloured by Kmeans Nodes",
      colour = "Kmeans clustering nodes") +
  theme_bw()
```

```
## Joining, by = "orchard"
```

```
wild_bee_abundance_kmeans
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

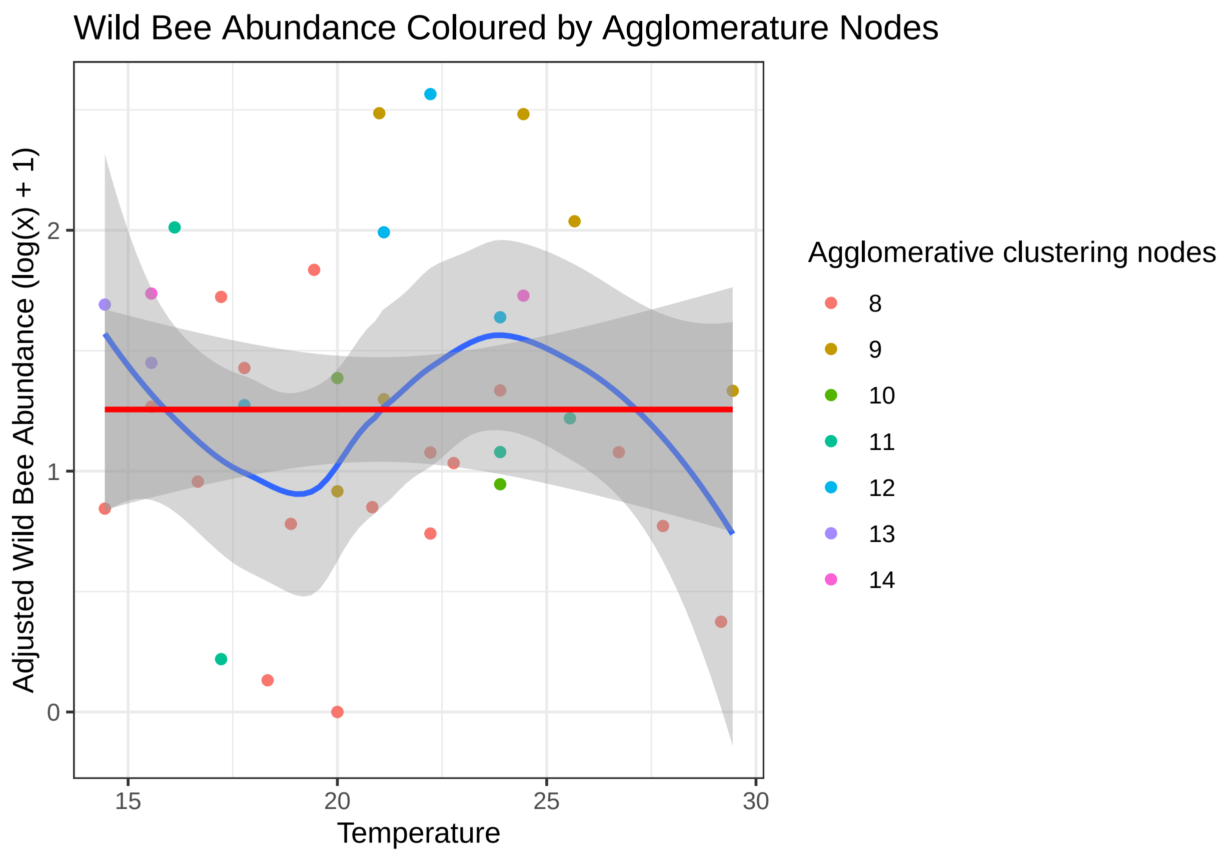


```
##Linear model
lm_temp <- lm(log(data_2012$wildAbF + 1) ~  data_2012$temp)
temp_factor <- tidy(lm_temp) %>%
  slice(2) %>%
  select(estimate) %>%
  as_vector()

#Adjusting bee value as though temp is always 20
adjusted_data <- data_2012 %>%
  inner_join(clusterings) %>%
  mutate(adjusted_bees = (((20 - temp) * temp_factor) + log(wildAbF + 1)))
```

```
## Joining, by = "orchard"
```

```
wild_bee_abundance_agglom_adjusted <- adjusted_data %>%
  ggplot(aes(x = temp, y = adjusted_bees)) +
  geom_point(aes(colour = as_factor(node))) +
  geom_smooth() +
  geom_smooth(method = "lm", colour = "red") +
  labs(x = "Temperature", y = "Adjusted Wild Bee Abundance (log(x) + 1)",
       title = "Wild Bee Abundance Coloured by Agglomerature Nodes",
       colour = "Agglomerative clustering nodes") +
  theme_bw()
wild_bee_abundance_agglom_adjusted
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

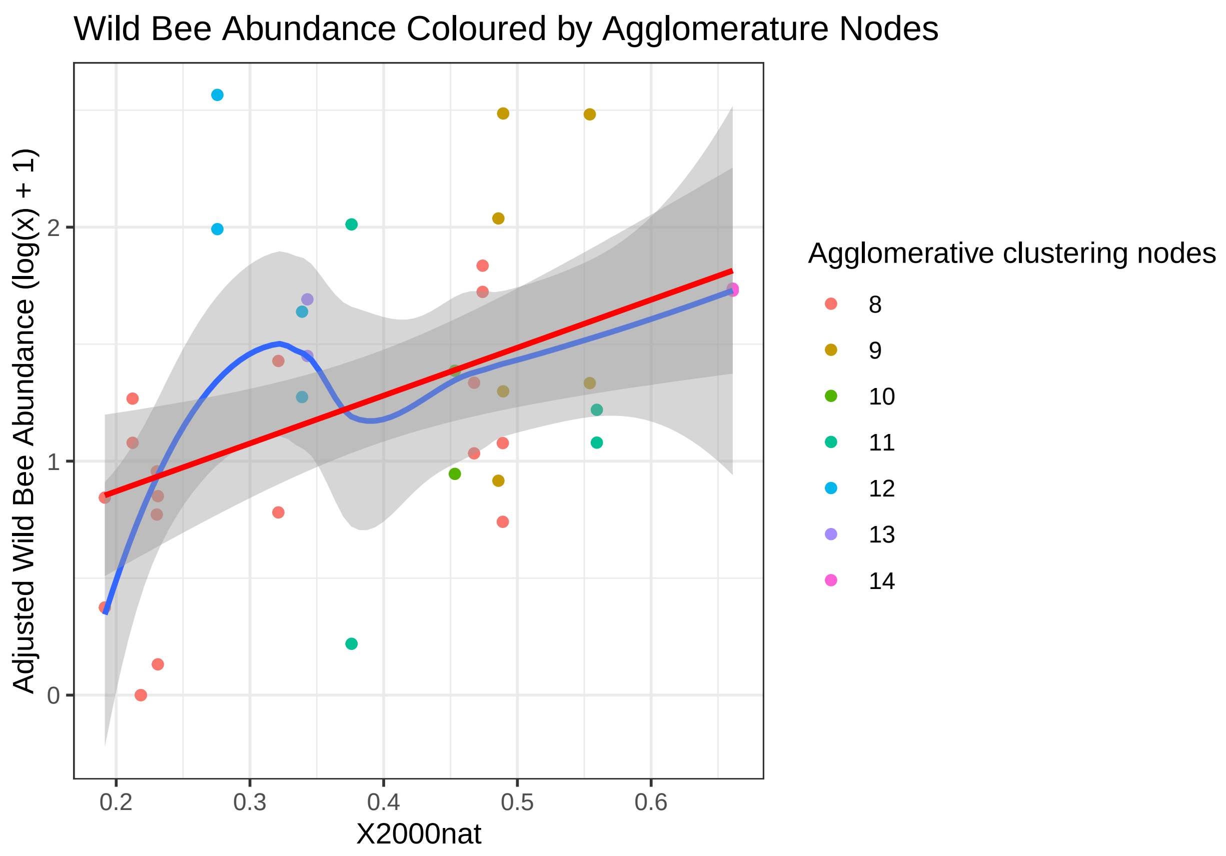


X2000nat

Based on previous research undertaken in the white paper it is known that the X2000nat variable has an effect on the the bee count so it also necessary to adjust for this. Wher Again the value will be adjust as though the X2000nat variable is constant. In this case the mean will be chosen as the constant value and the bee count will be adjusted for this in the same way as above.

```
wild_bee_abundance_agglom_x2000 <- adjusted_data %>%
  ggplot(aes(x = X2000nat, y = adjusted_bees)) +
  geom_point(aes(colour = as_factor(node))) +
  geom_smooth() +
  geom_smooth(method = "lm", colour = "red") +
  labs(x = "X2000nat", y = "Adjusted Wild Bee Abundance (log(x) + 1)",
       title = "Wild Bee Abundance Coloured by Agglomerature Nodes",
       colour = "Agglomerative clustering nodes") +
  theme_bw()
wild_bee_abundance_agglom_x2000
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



```
#Outcome is 0.388
mean_x2000 <- adjusted_data %>%
  summarise(mean_x2000 = mean(X2000nat)) %>%
  as_vector()


lm_X2000nat <- lm(adjusted_data$adjusted_bees ~  adjusted_data$X2000nat)
X2000_factor <- tidy(lm_X2000nat) %>%
  slice(2) %>%
  select(estimate) %>%
  as_vector()

#Adjusting the data for this variable

adjusted_data <- adjusted_data %>%
  mutate(adjusted_bees = ((mean_x2000 - X2000nat) * X2000_factor) + adjusted_bees)
```

So if you look at them individually, it can only be adjusted for one. So need to combine a linear model of all *affecting* factors to best adjust at the end.

Analysising other potential confounders

This section will look at finding other parameters to include in the bee variable, dataset used at this point is the original just with the bee variable considered going through a log(x + 1) transformation. The only bee variable to be considered at this point is the wild bee abundance, in future more could be considered in this way or it would be expected to be similar for all bee variables and I could just use the variables decided upon in this way for all the bee variables considered.

Local Diversity

The aim here is to whether adding local diversity as a adjustment factor is going to be beneficial or not. This will be achieved by running a F test to see whether the variances are equal, a shapiro test to confirm the bee data can be assumed to be normal. Note, independence assumption is not validated, this is due more than 1 result being used from each orchard, I decided to include these as I thought the extra power gained from twice the observations was worth the penalty of this assumption.

```
#When adding temperature in other variables have an effect
logged_data <- data_2012 %>%
  inner_join(clusterings) %>%
  mutate(adjusted_bees = (((20 - temp) * temp_factor) + log(wildAbF + 1)))
```

```
## Joining, by = "orchard"
```

```
#Just logged
logged_data <- data_2012 %>%
  inner_join(clusterings) %>%
  mutate(adjusted_bees = log(wildAbF + 1))
```

```
## Joining, by = "orchard"
```

```
#X2000nat adjustment
logged_data <- data_2012 %>%
  inner_join(clusterings) %>%
  mutate(adjusted_bees = log(wildAbF + 1))%>%
  mutate(adjusted_bees = ((mean_x2000 - X2000nat) * X2000_factor) + adjusted_bees)
```

```
## Joining, by = "orchard"
```

```
#Testing underlying normal data assumption, again demonstrating why the transformation is neccessary:
#Not normal
shapiro.test(logged_data$wildAbF)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  logged_data$wildAbF
## W = 0.79764, p-value = 9.361e-06
```

```
#After transformation -> normal
shapiro.test(logged_data$adjusted_bees)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  logged_data$adjusted_bees
## W = 0.97681, p-value = 0.6046
```

```
#Checking if the two subsets have the same variance have the same variance
#Outcome shows that they can be assumed to have the same variance as p.value > 0,05
simple <- logged_data %>% filter(local.diversity == 0) %>% select(adjusted_bees) %>% as_vector()
diverse <- logged_data %>% filter(local.diversity == 1) %>% select(adjusted_bees) %>% as_vector()
```

```r
tidy(var.test(simple, diverse)) %>%
  select(statistic, p.value)
```

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

## # A tibble: 1 x 2
##   statistic p.value
##       <dbl>   <dbl>
## 1      1.17   0.725
```

```r
#Performing a two sample t-test on the data to see whether they are the same.
#Since H0: Means the same, H1: means are different ~ p.value not significant

#This implies local.diversity does NOT have an effect on the original bee counts.
#I thought this was an interesting point and worth noting, same happens with Region.
tidy(t.test(wildAbF ~ local.diversity, logged_data, var.equal = TRUE))
```

```
## # A tibble: 1 x 9
##   estimate1 estimate2 statistic p.value parameter conf.low conf.high method
##       <dbl>     <dbl>     <dbl>   <dbl>     <dbl>    <dbl>     <dbl> <chr>
## 1       3.5      4.18    -0.532   0.598        36    -3.27      1.91 " Two~
## # ... with 1 more variable: alternative <chr>
```

```r
#However the the adjusted version of bee count does list local diversity
#as a factor which has an effect on the log transformed bee counts
tidy(t.test(adjusted_bees ~ local.diversity, logged_data, var.equal = TRUE))
```

```
## # A tibble: 1 x 9
##   estimate1 estimate2 statistic p.value parameter conf.low conf.high method
##       <dbl>     <dbl>     <dbl>   <dbl>     <dbl>    <dbl>     <dbl> <chr>
## 1      1.22      1.41    -0.896   0.376        36   -0.617     0.239 " Two~
## # ... with 1 more variable: alternative <chr>
```

From This analysis, it can be concluded that local.diversity does NOT play a factor in bee count ### Region Region has 3 possible options - as such an ANOVA test will be used to see whether there is a difference between the means in wild abundance based on region.

```r
#Checking Anova assumption that the variances are the same
# H0: Variances the same: H1: Atleast one variance is different
leveneTest(adjusted_bees ~ region, data = logged_data)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  2  0.1954 0.8234
##       35
```

```r
#Since pr > 0.05 we can assume variances are the same

#Running the anova  test
anova1 <- aov(adjusted_bees ~ region, data = logged_data)

summary(anova1)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## region       2  0.522  0.2611   0.624  0.542
## Residuals   35 14.652  0.4186
```

```
#probablity shows that region is not an affecting factor to  wildAbF

#Checking that the following anova assumption is TRUE:
#Residuals of the response variable are normally distributed is NOT true
shapiro.test(residuals(anova1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(anova1)
## W = 0.96856, p-value = 0.3546
```

Day

Looking at the day to see whether that also has an effect, since the data can be paired by day 1 and day 2 a paired t-test will be carried out on this to test whether there is a difference in bee count. When using just the X2000nat variable, then it becomes very close to significant.

```
day_1 <- logged_data %>% filter(day == 1) %>% select(adjusted_bees) %>% as_vector()
day_2 <- logged_data %>% filter(day == 2) %>% select(adjusted_bees) %>% as_vector()
#This demonstrates that the variances come from the same distribution:
tidy(var.test(day_1, day_2)) %>%
  select(statistic, p.value)
```

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

```
## # A tibble: 1 x 2
##   statistic p.value
##       <dbl>   <dbl>
## 1     0.936   0.891
```

```
#Hence we can run a paired t-test with equal variances
tidy(t.test(adjusted_bees ~ day, logged_data, var.equal = TRUE, paired = TRUE))
```

```
## # A tibble: 1 x 8
##   estimate statistic p.value parameter conf.low conf.high method
##      <dbl>     <dbl>   <dbl>     <dbl>    <dbl>     <dbl> <chr>
## 1   -0.342     -2.02  0.0588        18   -0.697    0.0142 Paire~
## # ... with 1 more variable: alternative <chr>
```

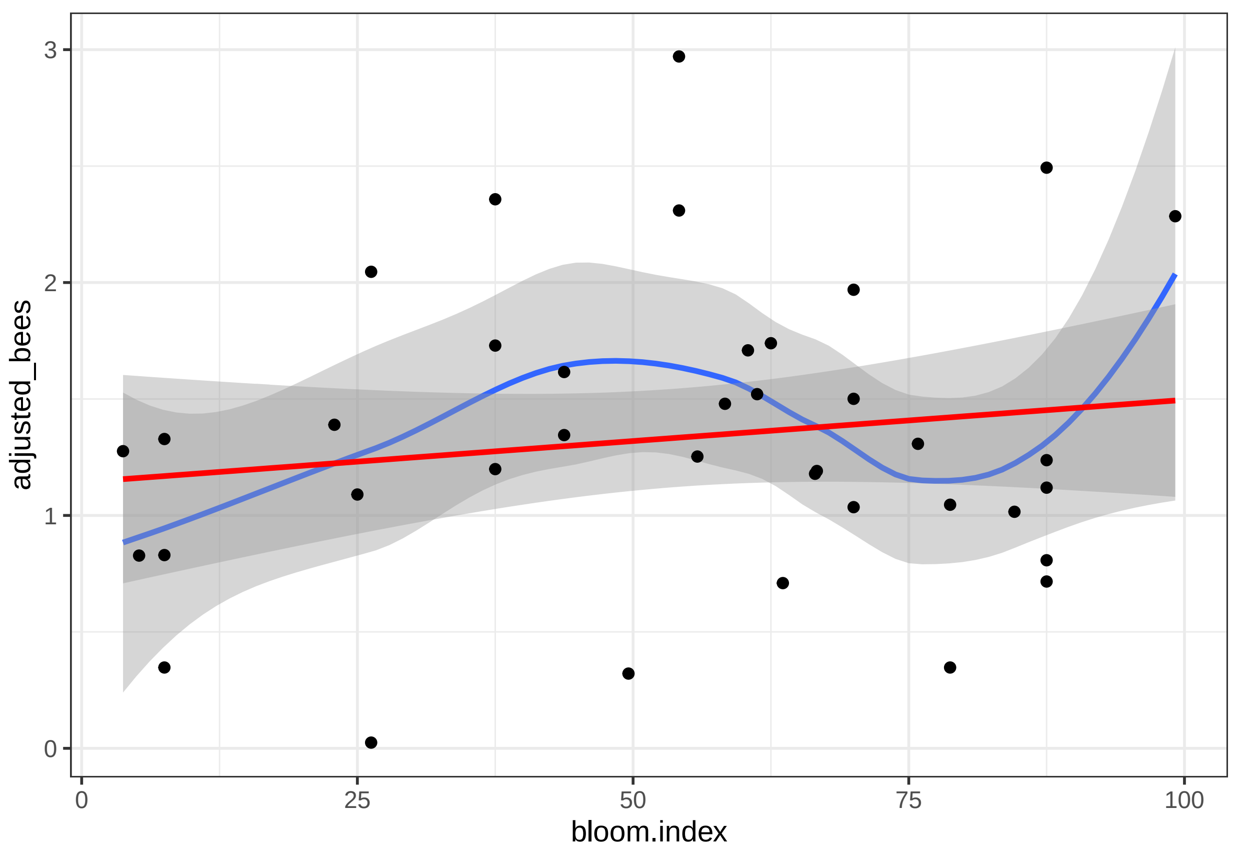
```
#Bit surprising but it demonstrates that there is no statistical significance between days
#So combining values to start with is probably appropriate
```

Bloom

First it shall be graphed to see if there is any obvious correlation.

```
logged_data %>%
  ggplot(aes(x = bloom.index, y = adjusted_bees)) +
  geom_smooth() +
  geom_smooth(method = "lm", colour = "red") +
  theme_bw() +
  geom_point()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



There doesn't seem to be any obvious correlation, but just to be sure I'll split the bloom into thee categories and test these with an Anova test.

```
logged_data <- logged_data %>%
          mutate(bloom_category = if_else(bloom.index <= 33, 1,
                                  if_else(bloom.index <= 66, 2, 3)))

#Running the anova  test
anova1 <- aov(adjusted_bees ~ bloom_category, data = logged_data)
summary(anova1)
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)
## bloom_category  1   0.19  0.1897   0.456  0.504
## Residuals      36  14.98  0.4162
```

```
#As expected there is no evidence to suggest that bloom index (category)
#Has an effect on the bee counts (although only 1 DF in bloom_category thought it should be 2??)
```

Bloom is a continuous variable but for the sakes of deciding whether to include it or not

Final Confounders

Based on the analysis above, the confounders that could be taken into account are: Region and local diversity based on the statistical tests, temperature and X2000nat based on the literature.

```
#Checking the variables to see if any are heavily correlated (binary/categorical ones won't be)
#So checking other two  and we get a low correlation so acceptable to use them together
cor(logged_data$temp, logged_data$X2000nat)
```

```
## [1] 0.2065843
```

```
f_lm <- lm(adjusted_bees ~ temp + X2000nat + local.diversity + region, data = logged_data)
#Interesting values here seem to suggest that only significant one is X2000nat
summary(f_lm)
```

```
##
## Call:
## lm(formula = adjusted_bees ~ temp + X2000nat + local.diversity +
##     region, data = logged_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.23257 -0.35949 -0.06686  0.23707  1.30560
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -0.33682    0.61856  -0.545  0.58985
## temp             0.08343    0.02620   3.184  0.00323 **
## X2000nat         0.09182    0.79392   0.116  0.90865
## local.diversity  0.20663    0.28923   0.714  0.48014
## regionLO        -0.42271    0.36305  -1.164  0.25290
## regionS         -0.08406    0.46728  -0.180  0.85837
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5782 on 32 degrees of freedom
## Multiple R-squared:  0.2949, Adjusted R-squared:  0.1848
## F-statistic: 2.677 on 5 and 32 DF,  p-value: 0.03941
```

Notes, from running the adjustments: When you use temperature as an original adjustment, only significant one in the model is X2000nat and when you use X2000nat in the original data only significant one is temp. When you use neither and just apply the linear model on the logged data then both temp and nat are significant -> suggesting that just these two variables would be best to adjust by. This suggest to me that fitting a joint lm using both temp and X2000nat to adjust the data is probably best to avoid overfitting and keep the model as parsimonious as possible. Despite region and local.diversity being significant for temp adjustment originally, unlikely to be worth implementing but could be considered to cover model changes section. To take this further, could apply lasso regression on the full dataset to see what that comes up with as tends to have a lower MSE than a standard LM.

```
#Using the logged_data with no adjustments
logged_data <- data_2012 %>%
  inner_join(clusterings) %>%
  mutate(adjusted_bees = log(wildAbF + 1))
```

```
## Joining, by = "orchard"
```

```
#Defining linear model with temp and X2000nat as factors
temp_nat_lm <- tidy(lm(adjusted_bees ~ temp + X2000nat, data = logged_data))
#Extracting temp and nat estimate values
temp_factor <- temp_nat_lm %>%
  select(estimate) %>%
  slice(2) %>%
  as_vector()
nat_factor <- temp_nat_lm %>%
  select(estimate) %>%
  slice(3) %>%
  as_vector()

#Now adjustments shall be made for these two variables:
#Temperature will be set to 20 degrees
#X2000 nat will be set to the meean x2000 nat value
temp_nat_adjusted <- logged_data %>%
  mutate(adjusted_bees = (adjusted_bees + ((20 - temp) * temp_factor) + ((mean_x2000 - X2000nat) * X200

#Checking that after the adjustment they are not significant at all in the model
tidy(lm(adjusted_bees ~ temp + X2000nat, data = temp_nat_adjusted))
```

```
## # A tibble: 3 x 5
##   term         estimate std.error statistic p.value
##   <chr>           <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)  1.24e+ 0    0.549   2.25e+ 0  0.0309
## 2 temp        -9.42e-18    0.0249 -3.79e-16  1.000
## 3 X2000nat     9.12e- 2    0.735   1.24e- 1  0.902
```

```
#As expected now the bee abundance has been adjusted for these two parameters
```

Clustering Summarising

Functions involved with summarising the clusters:

```
#Gets the name of a variable as a string ~ now implemented in clusterise but keeping
get_name <- function(x) {
  deparse(substitute(x))
}

#Gets the bee variables that I am summarising for each cluster - makes it easy to change in future
clusterise <- function(bee_data, cluster_data, v_name = deparse(substitute(cluster_data))){
  inner_join(bee_data, cluster_data) %>%
    summarise(mean_honey_ab = mean(mean_honey_ab),
              mean_wild_ab = mean(mean_wild_ab),
              mean_wild_rich = mean(mean_wild_rich)) %>%
    mutate(cluster = v_name)
}

#The function to output the summaries of both agglom and kmeans clustering
cluster_summarise <- function(bee_data){
output <- list()

output$agglom_bees <- clusterise(bee_data, aggl_node1) %>%
  bind_rows(clusterise(bee_data, aggl_node2)) %>%
  bind_rows(clusterise(bee_data, aggl_node3)) %>%
  bind_rows(clusterise(bee_data, aggl_node4)) %>%
  bind_rows(clusterise(bee_data, aggl_node5)) %>%
  bind_rows(clusterise(bee_data, aggl_node6)) %>%
  bind_rows(clusterise(bee_data, aggl_node7)) %>%
  bind_rows(clusterise(bee_data, aggl_node4 %>% filter(cluster == 1), "agglom_node8")) %>%
  bind_rows(clusterise(bee_data, aggl_node4 %>% filter(cluster == 2), "agglom_node9")) %>%
  bind_rows(clusterise(bee_data, aggl_node5 %>% filter(cluster == 1), "agglom_node10")) %>%
  bind_rows(clusterise(bee_data, aggl_node5 %>% filter(cluster == 2), "agglom_node11")) %>%
```

```r
  bind_rows(clusterise(bee_data, aggl_node6 %>% filter(cluster == 1), "agglom_node12")) %>%
  bind_rows(clusterise(bee_data, aggl_node6 %>% filter(cluster == 2), "agglom_node13")) %>%
  #Only 1 result for cluster as none in the latter cluster
  #Unable to to bind as no common variables for an empty node15
  bind_rows(clusterise(bee_data, aggl_node7 %>% filter(cluster == 1), "agglom_node14"))


output$kmeans_bees <- clusterise(bee_data, kmeans_node1) %>%
  bind_rows(clusterise(bee_data, kmeans_node2)) %>%
  bind_rows(clusterise(bee_data, kmeans_node3)) %>%
  bind_rows(clusterise(bee_data, kmeans_node4)) %>%
  bind_rows(clusterise(bee_data, kmeans_node5)) %>%
  bind_rows(clusterise(bee_data, kmeans_node6)) %>%
  bind_rows(clusterise(bee_data, kmeans_node7)) %>%
  bind_rows(clusterise(bee_data, kmeans_node4 %>% filter(cluster == 1), "kmeans_node8")) %>%
  bind_rows(clusterise(bee_data, kmeans_node4 %>% filter(cluster == 2), "kmeans_node9")) %>%
  bind_rows(clusterise(bee_data, kmeans_node5 %>% filter(cluster == 1), "kmeans_node10")) %>%
  bind_rows(clusterise(bee_data, kmeans_node5 %>% filter(cluster == 2), "kmeans_node11")) %>%
  bind_rows(clusterise(bee_data, kmeans_node6 %>% filter(cluster == 1), "kmeans_node12")) %>%
  bind_rows(clusterise(bee_data, kmeans_node6 %>% filter(cluster == 2), "kmeans_node13")) %>%
  bind_rows(clusterise(bee_data, kmeans_node7 %>% filter(cluster == 1), "kmeans_node14")) %>%
  bind_rows(clusterise(bee_data, kmeans_node7 %>% filter(cluster == 2), "kmeans_node15"))

#Return a list containing two tibbles one for each type of clustering algorithm
output
}
```

Original summarising without any variable correction

```r
#original Bee data
bee_values <- logged_data %>%
  group_by(orchard) %>%
  summarise(mean_honey_ab = mean(apisAb),
            mean_wild_ab = mean(adjusted_bees),
            mean_wild_rich = mean(wildRichF)
            )

original_bee_summary <- cluster_summarise(bee_data = bee_values)

#Agglom
original_bee_summary$agglom_bees
```

```
## # A tibble: 14 x 4
##    mean_honey_ab mean_wild_ab mean_wild_rich cluster
##            <dbl>        <dbl>          <dbl> <chr>
##  1          6.56        1.33            2.59 aggl_node1
##  2          6.19        1.23            2.50 aggl_node2
##  3          7.94        1.71            2.94 aggl_node3
##  4          6.80        1.23            2.54 aggl_node4
##  5          3.75        1.23            2.33 aggl_node5
##  6          8.75        1.70            2.67 aggl_node6
##  7          5.5         1.73            3.75 aggl_node7
##  8          5.46        0.963           1.77 agglom_node8
##  9         10.8         2.04            4.83 agglom_node9
## 10          3.25        1.32            2.25 agglom_node10
## 11          4           1.19            2.38 agglom_node11
## 12          8           1.97            3.25 agglom_node12
## 13         10.2         1.18            1.5  agglom_node13
## 14          5.5         1.73            3.75 agglom_node14
```

```
#Kmeans
original_bee_summary$kmeans_bees
```

```
## # A tibble: 15 x 4
##    mean_honey_ab mean_wild_ab mean_wild_rich cluster
##            <dbl>        <dbl>          <dbl> <chr>
##  1          6.56         1.33           2.59 kmeans_node1
##  2          6.19         1.23           2.50 kmeans_node2
##  3          7.94         1.71           2.94 kmeans_node3
##  4          6.5          1.36           2.72 kmeans_node4
##  5          6.04         1.17           2.39 kmeans_node5
##  6          7.94         1.71           2.94 kmeans_node6
##  7        NaN          NaN            NaN    kmeans_node7
##  8          3.25         1.32           2.25 kmeans_node8
##  9          7.31         1.37           2.83 kmeans_node9
## 10          6.04         1.17           2.39 kmeans_node10
## 11        NaN          NaN            NaN    kmeans_node11
## 12          7.17         1.89           3.42 kmeans_node12
## 13         10.2          1.18           1.5  kmeans_node13
## 14        NaN          NaN            NaN    kmeans_node14
## 15        NaN          NaN            NaN    kmeans_node15
```

After Adjustment:

```
adjusted_bee_values <- temp_nat_adjusted %>%
  group_by(orchard) %>%
  summarise(mean_honey_ab = mean(apisAb),
            mean_wild_ab = mean(adjusted_bees),
            mean_wild_rich = mean(wildRichF)
            )

adjusted_bee_summary <- cluster_summarise(bee_data = adjusted_bee_values)

#Agglom
adjusted_bee_summary$agglom_bees
```

```
## # A tibble: 14 x 4
##    mean_honey_ab mean_wild_ab mean_wild_rich cluster
##            <dbl>        <dbl>          <dbl> <chr>
##  1          6.56        1.27            2.59 aggl_node1
##  2          6.19        1.15            2.50 aggl_node2
##  3          7.94        1.72            2.94 aggl_node3
##  4          6.80        1.19            2.54 aggl_node4
##  5          3.75        1.01            2.33 aggl_node5
##  6          8.75        1.90            2.67 aggl_node6
##  7          5.5         1.17            3.75 aggl_node7
##  8          5.46        1.06            1.77 agglom_node8
##  9         10.8         1.56            4.83 agglom_node9
## 10          3.25        1.06            2.25 agglom_node10
## 11          4           0.980           2.38 agglom_node11
## 12          8           2.05            3.25 agglom_node12
## 13         10.2         1.59            1.5  agglom_node13
## 14          5.5         1.17            3.75 agglom_node14
```

```
#Kmeans
adjusted_bee_summary$kmeans_bees
```

```
## # A tibble: 15 x 4
##    mean_honey_ab mean_wild_ab mean_wild_rich cluster
##            <dbl>        <dbl>          <dbl> <chr>
##  1          6.56         1.27           2.59 kmeans_node1
##  2          6.19         1.15           2.50 kmeans_node2
##  3          7.94         1.72           2.94 kmeans_node3
##  4          6.5          1.13           2.72 kmeans_node4
##  5          6.04         1.16           2.39 kmeans_node5
##  6          7.94         1.72           2.94 kmeans_node6
##  7        NaN          NaN            NaN    kmeans_node7
##  8          3.25         1.06           2.25 kmeans_node8
##  9          7.31         1.15           2.83 kmeans_node9
## 10          6.04         1.16           2.39 kmeans_node10
## 11        NaN          NaN            NaN    kmeans_node11
## 12          7.17         1.76           3.42 kmeans_node12
## 13         10.2          1.59           1.5  kmeans_node13
## 14        NaN          NaN            NaN    kmeans_node14
## 15        NaN          NaN            NaN    kmeans_node15
```

Although honey bee abundance is one of the summarised variables, this is likely to be related directly to the hive.acr variable.

Comparing cluster values before and after adjustments, for now just looking at the agglomerative node clustering selection as it is, in my opinion, better than Kmeans.

```
##agglom
adjusted_bee_summary$agglom_bees %>%
  select(mean_wild_ab, cluster) %>%
  rename(adjusted = mean_wild_ab) %>%
  inner_join(original_bee_summary$agglom_bees %>%
             select(mean_wild_ab, cluster)) %>%
  select(cluster,mean_wild_ab, adjusted)
```

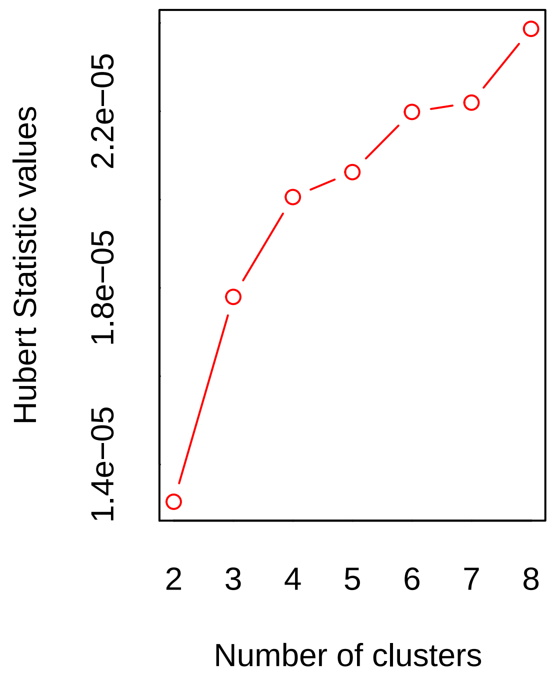
```
## Joining, by = "cluster"

## # A tibble: 14 x 3
##    cluster       mean_wild_ab adjusted
##    <chr>                <dbl>    <dbl>
##  1 aggl_node1           1.33     1.27
##  2 aggl_node2           1.23     1.15
##  3 aggl_node3           1.71     1.72
##  4 aggl_node4           1.23     1.19
##  5 aggl_node5           1.23     1.01
##  6 aggl_node6           1.70     1.90
##  7 aggl_node7           1.73     1.17
##  8 agglom_node8         0.963    1.06
##  9 agglom_node9         2.04     1.56
## 10 agglom_node10        1.32     1.06
## 11 agglom_node11        1.19     0.980
## 12 agglom_node12        1.97     2.05
## 13 agglom_node13        1.18     1.59
## 14 agglom_node14        1.73     1.17
```

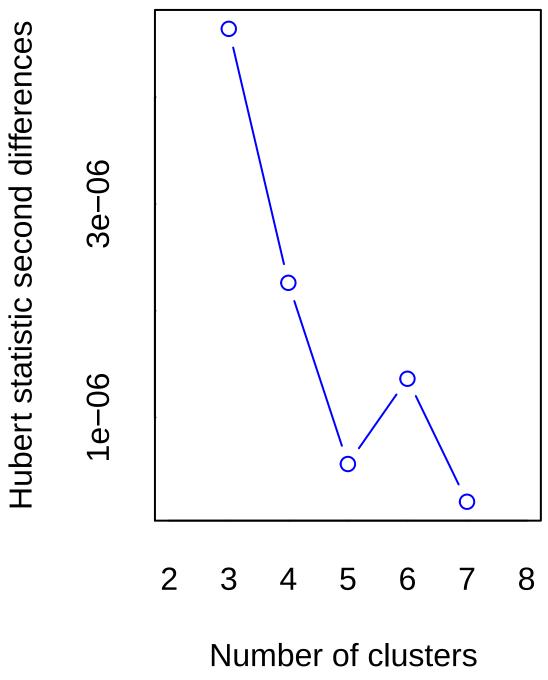
I guess potentially can see two *paths* one with a mean slightly lower than the other if you think of the structure.

Non-markov structuring

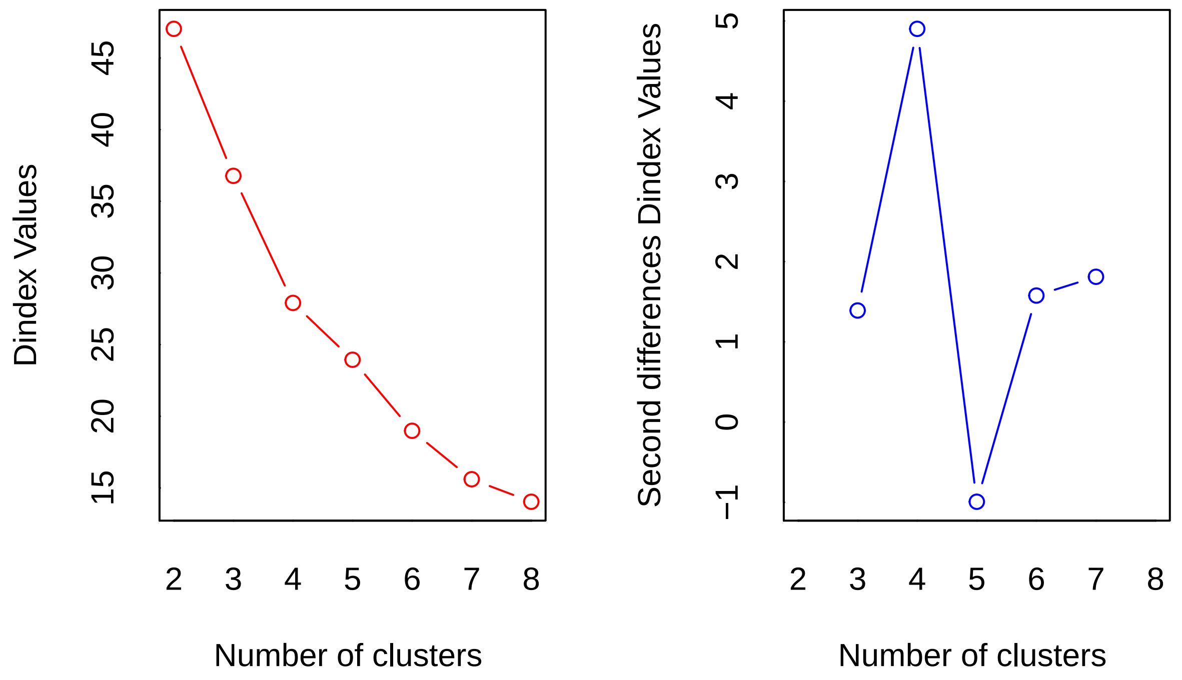
This section will approach looking at the data as one time point with 1 clustering point rather than 3 as above.

```
whole_cluster <- adjusted_data %>%
  arrange(orchard) %>%
  select(ends_with(".pre"),ends_with(".blm"), ends_with(".pos")) %>%
  filter(row_number() %% 2 == 1)

#Shows no difference here although from previous work ward method was best
# matrix of methods to compare
m <- c( "average", "single", "complete", "ward")
names(m) <- c( "average", "single", "complete", "ward")
distances <- c("euclidean", "maximum", "manhattan", "canberra",
               "binary", "minkowski")
names(distances) <- c("euclidean", "maximum", "manhattan",
                      "canberra", "binary", "minkowski")
clust_comps <- matrix(nrow = length(distances), ncol = length(m),
                      dimnames = list(distances,m))
# function to compute coefficient to see which is the best method
ac <- function(distance, linkage) {
  dista <- dist(whole_cluster , method = distance)
  #Agglomerative Nesting form of Hierarchical Clustering
  agnes(dista, method = linkage)$ac
}
for(i in 1:length(distances)) {
  for(j in 1:length(m)) {
    clust_comps[i,j] <- ac(distances[i], m[j])
  }
}

#In future or in write up, perhaps change, needs more clustering analysis
#Max of this suggests ward - binary as the best method
#Look at literature in more detail to see other benefits/drawbacks and assumptions
#Then decide on actual best method, for now using euclidian as used that before
clust_comps %>%
  View()

#Clusters of whole data combined - 7
fviz_nbclust(NbClust(whole_cluster, distance = "euclidean",
                                    min.nc = 2, max.nc = 8, method = "ward.D2", index = "all"))
```

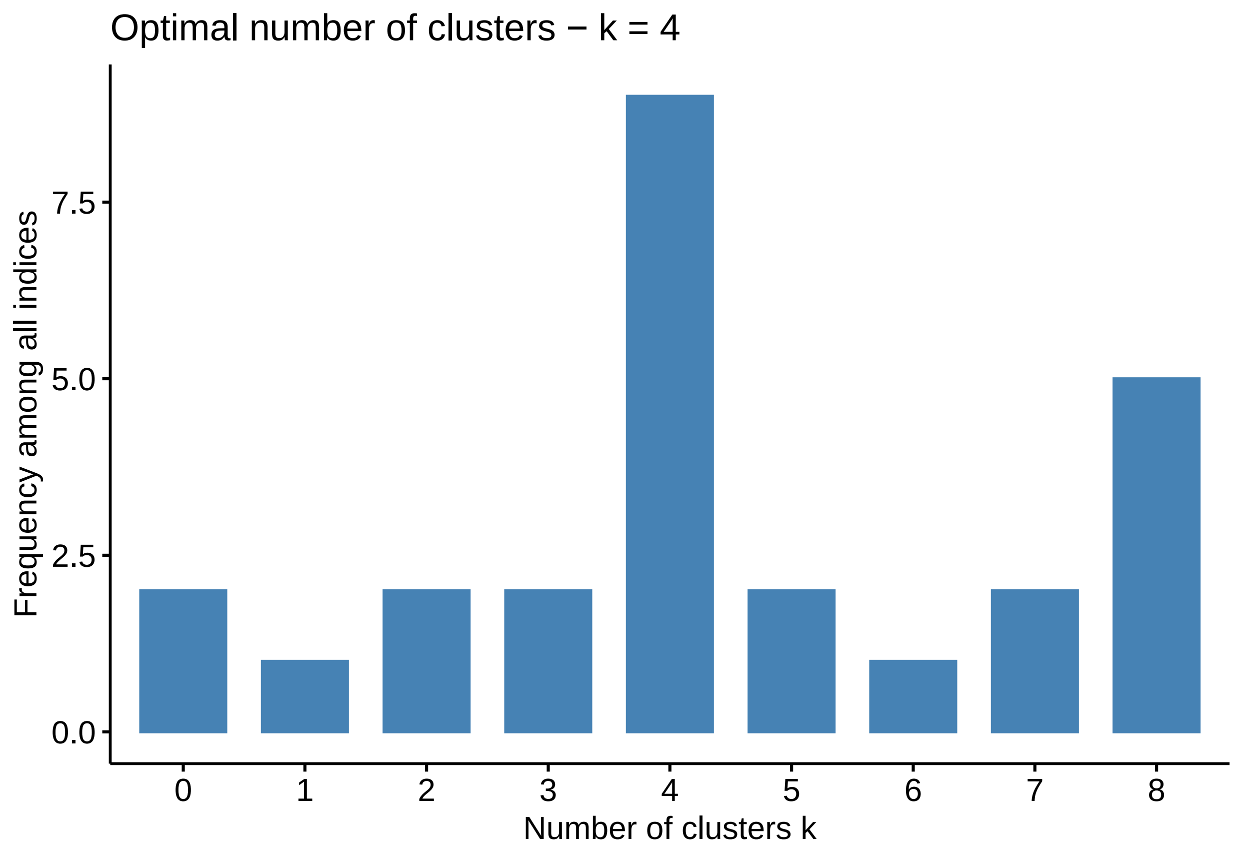




```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 2 proposed 2 as the best number of clusters
## * 2 proposed 3 as the best number of clusters
## * 9 proposed 4 as the best number of clusters
## * 2 proposed 5 as the best number of clusters
## * 1 proposed 6 as the best number of clusters
## * 2 proposed 7 as the best number of clusters
## * 5 proposed 8 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  4
##
##
## *******************************************************************
## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 1 proposed  1 as the best number of clusters
## * 2 proposed  2 as the best number of clusters
## * 2 proposed  3 as the best number of clusters
## * 9 proposed  4 as the best number of clusters
## * 2 proposed  5 as the best number of clusters
## * 1 proposed  6 as the best number of clusters
## * 2 proposed  7 as the best number of clusters
## * 5 proposed  8 as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  4 .
```



```
#Agglomerative clustering
clustered <- agnes(dist(whole_cluster, method = "euclidian"),
                   diss=TRUE, method = "ward")
# add cluster labels to the training data
whole_cluster$cluster <- cutree(clustered, k = 4)

whole_cluster
```

```
##    eiqB11F.pre eiqB11I.pre eiqB11F.blm eiqB11I.blm eiqB11T.blm eiqB11F.pos
## 1       143.09       0.000       58.10       19.04      0.0440       14.48
## 2       161.57       0.000       53.16        9.36      0.0000       31.37
## 3       161.57       0.000       53.16        9.36      0.0000       31.37
## 4       161.57       0.000       53.16        9.36      0.0000       31.37
## 5        41.85       0.000       44.67       14.25      0.0260       32.29
## 6         9.38       0.000       37.44       36.36      0.4600       24.39
## 7       163.95       2.310      112.90       12.62      0.0250       88.69
## 8      111.67      0.000     127.30       8.87      0.2200       53.11
## 9       94.76      0.960      52.27      19.69      0.0140       26.59
## 10     107.72      0.000       0.56       1.80      0.0000       41.85
## 11     196.87      0.330      63.57       4.94      0.0060       20.25
## 12     193.35      0.056      63.48       4.94      0.0031       19.87
## 13      91.93      0.000      53.30       8.13      0.0034       18.75
## 14      18.03      0.000      70.09      13.10      0.0000       20.53
## 15      21.18      6.740      17.79       5.49      0.0000        7.57
## 16     102.61      0.000      68.15       7.54      3.0000       29.44
## 17     106.47      0.000      33.00      28.70      0.0160       15.17
## 18     109.58      0.000      54.16      11.38      0.1700       16.79
## 19     113.28      0.000     102.91       3.61      0.0000       68.14
##    eiqB11I.pos eiqB11T.pos cluster
## 1        24.31       0.000       1
## 2        57.66       0.000       1
## 3        57.66       0.000       1
## 4        57.66       0.000       1
## 5         9.63       0.000       2
## 6        34.75       0.000       2
## 7        51.87       0.041       3
## 8         5.25       0.000       3
## 9        20.81       0.082       4
## 10        1.80       0.000       4
## 11       21.36       0.084       1
## 12       21.36       0.230       1
## 13        7.58       0.061       4
## 14        2.28       0.000       2
## 15        9.48       0.000       2
## 16       10.91       3.000       4
## 17       10.08       0.000       4
## 18       17.33       0.220       4
## 19       11.92       0.070       3
```

```
#Clustering summaries based on non-markov approach
adjusted_data %>%
  group_by(orchard) %>%
  summarise(mean_honey_ab = mean(apisAb),
            mean_wild_ab = mean(adjusted_bees),
            mean_wild_rich = mean(wildRichF)) %>%
  mutate(cluster = whole_cluster$cluster)  %>%
  group_by(cluster) %>%
  summarise(mean_honey_ab = mean(mean_honey_ab),
            mean_wild_ab = mean(mean_wild_ab),
            mean_wild_rich = mean(mean_wild_rich))
```

```
## # A tibble: 4 x 4
##   cluster mean_honey_ab mean_wild_ab mean_wild_rich
##     <int>         <dbl>        <dbl>          <dbl>
## 1       1          8.31        1.15            2.77
## 2       2          7.94        1.73            2.94
## 3       3          3.75        0.990           2.33
## 4       4          5.29        1.18            2.31
```