Productivity Index - Output - By Quantity

Emily Markowitz (Emily.Markowitz@noaa.gov) and Sun-Ling Wang

Feb. 27, 2020

Contents

1 Math Theory: General Total Factor Productivity (TFP) Equation

The general form of the TFP can be measured as aggregate output (Y) divided by real total inputs (X). Rates of TFP growth are constructed using the Törnqvist index approach. The TFP growth over two time periods is defined as:

$$ln(TFP_t/TFP_{t-1}) = \sum_{i=1}^{n}((\frac{R_{t,i} + R_{t-1,i}}{2}) * ln(\frac{Y_{t,i}}{Y_{t-1,i}}))) - \sum_{j=1}^{m}((\frac{W_{j,t} + W_{j,t-1}}{2}) * ln(\frac{X_{j,t}}{X_{j,t-1}})))$$

Such that:

- Output $= \sum_{i=1}^{n}((\frac{R_{it}+R_{it-1}}{2}) * ln(\frac{Y_{it}}{Y_{it-1}}))$
- Input $= \sum_{j=1}^{n}((\frac{W_{jt}+W_{jt-1}}{2}) * ln(\frac{X_{jt}}{X_{jt-1}}))$

where:

- Y_i are individual outputs. This will later be refered to as Q_i in the following equations.
- X_j are individual inputs
- R_i are output revenue shares
- W_j are input cost shares

- t and $t-1$ are time subscripts, where 1 is the minimum year in the dataset
- i is category, e.g., Finfish (=1), Shellfish (=2)
- s is species, e.g., Salmon, Alewife, Surf Clams

---

2 Output Method: From Quantity to Quantity Measures

2.0.1 Variable Summary

Variables

- Q are individual quantity outputs in pounds (lbs).
- V are individual value outputs in dollars ($)
- R are output revenue shares
- $baseyr$ is the year to base all indicides from

Inidicies

- t and $t-1$ are time subscripts, where 1 is the minimum year in the dataset
- i is category, e.g., Finfish (=1), Shellfish (=2)
- s is species, e.g., Salmon, Alewife, Surf Clams

2.0.2 Data requirements

We need time series data for the value of all species (V_t/Total), value of all species in a category (i) ($V_{i=1}$; e.g., Finfish), value for an individual species in a category (i) ($V_{i=1,s=n}$; e.g., Salmon and Summer Flounder), the quantity of individual species in a category (i) ($Q_{i=1,s=n}$; e.g., Salmon and Flounder):

2.0.2.1 Edit Data

Here we summate the total value for each category and total V for all categories because there may be instances where these values may not be the sum of their parts (though they are here). The caluclation Price Index aims to deal with this potiental issue.

```
temp<-read.csv(file = paste0(dir.data, "Tornqvist Index-Calculations_OutputEx.csv"))
rownames(temp)<-temp$year
temp$year<-NULL

temp.q<-temp[,grepl(pattern = "Q", x = names(temp))]
temp.q$QE0_0Total<-rowSums(temp.q, na.rm = T)
temp.q$QE1_0Finfish<-rowSums(temp.q[,grepl(x = names(temp.q), pattern = "Q1") ], na.rm = T)
temp.q$QE2_0Shellfish<-rowSums(temp.q[,grepl(x = names(temp.q), pattern = "Q2") ], na.rm = T)

temp.v<-temp[,grepl(pattern = "V", x = names(temp))]
temp.v$VE0_0Total<-rowSums(temp.v, na.rm = T)
temp.v$VE1_0Finfish<-rowSums(temp.v[,grepl(x = names(temp.v), pattern = "V1") ], na.rm = T)
temp.v$VE2_0Shellfish<-rowSums(temp.v[,grepl(x = names(temp.v), pattern = "V2") ], na.rm = T)

temp<-orgional.data<-cbind.data.frame(temp.q, temp.v)
```

```
# temp<-orgional.data
# temp$Q1_6Flounder<-rnorm(n = nrow(temp), mean = 300, sd = 100)
# temp$Q0_0Total<-temp0$QE0_0Total
```

	Q1__1Salmon	Q1__2Cod	Q2__1Shrimp	Q2__2Clam	Q1__3Flounder	Q1__4SeaBass	QE0__0Total	QE1__0Fi
2007	NA	2000	100	150	NA	1000	3250	
2008	NA	1900	120	160	NA	1200	3380	
2009	NA	2000	110	140	NA	900	3150	
2010	20	2500	90	NA	NA	NA	2610	
2011	10	2400	80	NA	NA	NA	2490	
2012	12	2300	100	NA	NA	NA	2412	
2013	11	2000	100	140	NA	1000	3251	
2014	11	2300	110	110	NA	900	3431	
2015	10	2400	90	130	NA	1000	3630	
2016	15	2200	100	160	NA	1100	3575	

2.0.2.2 The nameing conventions of the column names.

For example, in "V1__0Finfish":

- "V"… refers to the variable represented in the column (here V = "Value")
- …"1"… refers to the category iteration (here, = Finfish)
- …"__"… is simply a seperator in the title
- …"0".. refers to the total of the specific category.
- …"Finfish" is purely descriptive (here the name of the category), so you can follow along with what is happening!

Similarly for "Q2__2Clam":

- "Q"… refers to the variable represented in the column (here Q = "Quantity")
- …"2"… refers to the category iteration (here, = Shellfish)
- …"__"… is simply a seperator in the title
- …"2".. refers to the iteration of the species, such that this organism happens to be the second species of this category.
- …"Clams" is purely descriptive (here the name of the species), so you can follow along with what is happening!

2.0.3 Lets get started

```
ii<-1 #The category iteration value
category<- c(1,2)
baseyr<-2010
pctmiss<-0.50 #If data are missing by the below percentage, remove data
```

In most of the following examples, we will just focus on the finfish (i=1) side of the equation. Here *baseyr* is set to 2010 and the *pctmiss* (The percent of data in a column that we will allow to be missing for analysis; more on that later) is set to 0.5%.

Here I am just going to do some housekeeping:

```r
place <- "Test"

  warnings.list<-list()
  figures.list<-list()

  ########Housekeeping
  # Here I am just going to collect some housekeeping items
  temp<-data.frame(temp)

  NumberOfSpecies<-numbers0(x = c(0, strsplit(x =
                                              strsplit(x = names(temp)[2],
                                                       split = "_")[[1]][2],
                                            split = "[a-zA-Z]")[[1]][1]))[1]




  category<-unique(as.character(lapply(X = strsplit(x = as.character(names(temp)),
                                                  split = paste0("_")),
                                       function(x) x[1])))
  category<-unique(substr(x = category, start = 2, stop = nchar(category)))
  category<-category[!grepl(pattern = "[a-zA-Z]", x = category)]
  category<-category[!(category %in% numbers0(c(0, (category)[1]))[1])]

  temp0<-data.frame(rep_len(x = NA, length.out = nrow(temp)))
  tempPC<-data.frame(rep_len(x = NA, length.out = nrow(temp0)))
  tempQC<-data.frame(rep_len(x = NA, length.out = nrow(temp0)))

  maxyr<-max(rownames(temp))
  minyr<-min(rownames(temp))

  NameBaseTotal<-paste0(paste(rep_len(x = 0, length.out = nchar(category[1])), collapse = ""),
                      "_", NumberOfSpecies, "Total")

########Category Specific############


    QColumns0<-QColumns<-grep(pattern = paste0("Q", category[ii],"_"),
                              x = substr(x = names(temp),
                                         start = 1,
                                         stop = (2+nchar(category[ii]))))

    VColumns0<-VColumns<-grep(pattern = paste0("V", category[ii],"_"),
                              x = substr(x = names(temp),
                                         start = 1,
                                         stop = (2+nchar(category[ii]))))

    NameBasecategory<-names(temp)[grepl(pattern = paste0("VE", category[ii],"_"),
                                        x = substr(x = names(temp),
                                                   start = 1,
                                                   stop = (3+nchar(category[ii]))))]

    NameBasecategory<-substr(x = NameBasecategory, start = 3, stop = nchar(NameBasecategory))
```

2.0.4 Remove any V and Q data where V column has less data than the specifed *pctmiss*

```
for (i in 1:length(VColumns)) {

  #if the percent missing is less in V or Q columns for a species than the percentmissingtrheshold, w
  if (sum(is.na(temp[VColumns[i]]))/nrow(temp) > pctmiss | #V
      sum(is.na(temp[QColumns[i]]))/nrow(temp) > pctmiss ) {#Q

    names(temp)[VColumns[i]]<-paste0("REMOVED_", names(temp)[VColumns[i]])
    VColumns0<-VColumns0[!(VColumns0 %in% VColumns[i])]
    names(temp)[QColumns[i]]<-paste0("REMOVED_", names(temp)[QColumns[i]])
    QColumns0<-QColumns0[!(QColumns0 %in% QColumns[i])]
  }
}

if (length(VColumns0) == 0 ) {

  warnings.list[length(warnings.list)+1]<-paste0("FYI: ", NameBasecategory, " is no longer being calcu

} else {

VColumns<-names(temp)[VColumns0]
QColumns<-names(temp)[QColumns0]

a<-"No warning."
}
```

No warning.

	Q1__1Salmon	Q1__2Cod	Q2__1Shrimp	Q2__2Clam	REMOVED_Q1__3Flounder	Q1__4SeaBass
2007	NA	2000	100	150	NA	1000
2008	NA	1900	120	160	NA	1200
2009	NA	2000	110	140	NA	900
2010	20	2500	90	NA	NA	NA
2011	10	2400	80	NA	NA	NA
2012	12	2300	100	NA	NA	NA
2013	11	2000	100	140	NA	1000
2014	11	2300	110	110	NA	900
2015	10	2400	90	130	NA	1000
2016	15	2200	100	160	NA	1100

2.0.5 Caluclate Category Sums of V and Q

Because we removed some columns for not meeting a perecent missing threshold of 0.5% and those columns will not be used at all in any part of the further analysis, we need to re-calculate the totals of V and Q for the catagories and the fishery as a whole.

```
  # Q
  temp.q<-data.frame(temp[,QColumns])
  if (ncol(temp.q)>1) {
    temp.q<-rowSums(temp.q, na.rm = T)
  }
```

```
    temp[ncol(temp)+1]<-temp.q
    names(temp)[ncol(temp)]<-paste0("Q",NameBasecategory)

    # V
    temp.v<-data.frame(temp[,VColumns])
    if (ncol(temp.v)>1) {
      temp.v<-rowSums(temp.v, na.rm = T)
    }
    temp[ncol(temp)+1]<-temp.v
    names(temp)[ncol(temp)]<-paste0("V",NameBasecategory)
```

	QE1_0Finfish	VE1_0Finfish	Q1_0Finfish	V1_0Finfish
2007	3000	3800	3000	2800
2008	3100	4020	3100	2820
2009	2900	3910	2900	3010
2010	2520	3190	2520	3190
2011	2410	3280	2410	3280
2012	2312	3150	2312	3150
2013	3011	4080	3011	3080
2014	3211	4270	3211	3370
2015	3410	4700	3410	3700
2016	3315	4480	3315	3380

There may be instances where there are no or too few Q data for that species in a year or ever. The next goal will be to calculate the quantity change, so we need to have a value in there that won't show change. If we left a 0 in the spot, then the price change from 0 to the next year would be huge and misrepresented on the index. To avoid this, we have to deal with four senarios:

2.0.5.1 1. If there are instances for a species where there are too few pairs of Q are completely missing from the timeseries or where a percent of Q is missing from the timeseries, we will remove the offending price columns entierly, so they don't influence the downstream price change or price index calculations.

Let's say here that if 50% of the data is missing in a given $Q_{t,i,s}$, don't use that to calculate that species $Q_{t,i}$

```
#Find which columns in this table are quantity Columns
    cc<-c() #Empty
    for (c in 1:length(QColumns)) {
      #If price could never be caluclated at any point in the timeseries (is 0/NaN/NA) for a column (c)
      #Remove the column from the analysis.
      #We will not be removing the column from the data, but simply remove it from the varaible "QColum
      if (sum(temp[,QColumns[c]] %in% c(0, NA, NaN))/nrow(temp) > pctmiss) {
        cc<-c(cc, c)#Collect offending columns
      }
    }
    if (length(cc)>0){
      QColumns<-QColumns[-cc]
    }
```

	Q1_1Salmon	Q1_2Cod	Q1_4SeaBass
2007	NA	2000	1000
2008	NA	1900	1200

	Q1_1Salmon	Q1_2Cod	Q1_4SeaBass
2009	NA	2000	900
2010	20	2500	NA
2011	10	2400	NA
2012	12	2300	NA
2013	11	2000	1000
2014	11	2300	900
2015	10	2400	1000
2016	15	2200	1100

```
    if (length(QColumns) < 2) {

    warnings.list[length(warnings.list)+1]<-paste0("FYI: ", NameBasecategory, " is no longer being calcu
  } else {
  a<-"No warning."
}
```

No warning.

2.0.5.2 2. If the first value of $Q_{t,i,s}$ is 0/NA in a timeseries, we (impute) let the next available non-zero/non-NA value of Q in the timeseries inform the past.

$$where \begin{cases} if : Q_{t,i=1} = 0, then : Q_{t,i=1} = Q_{t,i=1+1...} \\ if : Q_{t,i\neq 1} = 0, then : Q_{t,i} = Q_{t-1,i} \end{cases}$$

We use this *ReplaceFirst* function:

```
print(ReplaceFirst)
```

```
## function(colnames, temp) {
##   for (c0 in 1:length(colnames)) {
##
##     #If the first value of the timeseries of this column (c) is 0/NaN/NA
##     #Change the first value (and subsequent 0/NaN/NA values) to the first available non-0/NaN/NA valu
##     if (temp[1,colnames[c0]] %in% c(0, NA, NaN, NULL)) {
##       findfirstvalue<-temp[which(!(temp[,colnames[c0]]  %in% c(0, NA, NaN, NULL))),
##                            colnames[c0]][1]
##       temp[1,colnames[c0]]<-findfirstvalue
##     }
##   }
##   return(temp)
## }
## <bytecode: 0x000000001f297400>
```

```
temp<-ReplaceFirst(colnames = QColumns, temp)
```

	Q1_1Salmon	Q1_2Cod	Q1_4SeaBass
2007	20	2000	1000
2008	NA	1900	1200
2009	NA	2000	900
2010	20	2500	NA
2011	10	2400	NA

	Q1_1Salmon	Q1_2Cod	Q1_4SeaBass
2012	12	2300	NA
2013	11	2000	1000
2014	11	2300	900
2015	10	2400	1000
2016	15	2200	1100

2.0.5.3 3. If there is a value in the middle of $P_{t,i,s}$'s timeseries that is 0/NA, we (impute) let the most recent past available non-zero/non-NA of $P_{t,i,s}$ in the timeseries inform the future.

We use this *ReplaceMid* function:

```r
print(ReplaceMid)
```

```
## function(colnames, temp) {
##   for (c0 in 1:length(colnames)) {
##     #If a middle value of the timeseries of this column (c) is 0/NaN/NA
##     #Change the currently 0/NaN/NA value to the previous available non-0/NaN/NA value
##     if (sum(temp[,colnames[c0]] %in% c(0, NA, NaN, NULL))>0) {
##       troublenumber<-which(temp[,colnames[c0]] %in% c(0, NA, NaN, NULL))
##       for (r in 1:length(troublenumber)){
##         findlastvalue<-temp[troublenumber[r]-1, colnames[c0]][1]
##         temp[troublenumber[r],colnames[c0]]<-findlastvalue
##       }
##     }
##   }
##   return(temp)
## }
## <bytecode: 0x000000001f63fb58>
```

```r
temp<-ReplaceMid(colnames = QColumns, temp)
```

	Q1_1Salmon	Q1_2Cod	Q1_4SeaBass
2007	20	2000	1000
2008	20	1900	1200
2009	20	2000	900
2010	20	2500	900
2011	10	2400	900
2012	12	2300	900
2013	11	2000	1000
2014	11	2300	900
2015	10	2400	1000
2016	15	2200	1100

2.0.6 Impute values of $V_{t,i,s}$ where P was able to be calculated

To ensure that the price index does not rise or fall to quickly with changes (that are really because of NA values) we fill in the missing instances of $V_{t,i,s}$.

$$where \begin{cases} if : V_{t,i=1} = 0, then : V_{t,i=1} = V_{t,i=1+1...} \\ if : V_{t,i\neq 1} = 0, then : V_{t,i} = V_{t-1,i} \end{cases}$$

2.0.6.1 1. If the first value of $V_{t,i,s}$ is 0/NA in a timeseries, we let the next available non-zero value of $V_{t,i,s}$ in the timeseries inform the past.

```
QColumns<-QColumns[!(grepl(pattern = "REMOVED", x = QColumns))]
VVColumns<-paste0("V", substr(x = QColumns, start = 2, stop = nchar(QColumns)))
temp<-ReplaceFirst(colnames = VVColumns, temp)
```

	V1_1Salmon	V1_2Cod	V1_4SeaBass
2007	100	2800	120
2008	NA	2700	120
2009	NA	2900	110
2010	100	3000	90
2011	100	3100	80
2012	150	2900	100
2013	180	2800	100
2014	170	3200	NA
2015	200	3500	NA
2016	180	3200	NA

2.0.6.2 2. If there is a value in the middle of $V_{t,i,s}$'s timeseries that is 0/NA, we let the most recent past available non-zero of $V_{t,i,s}$ in the timeseries inform the future.

```
temp<-ReplaceMid(colnames = VVColumns, temp)
```

	V1_1Salmon	V1_2Cod	V1_4SeaBass
2007	100	2800	120
2008	100	2700	120
2009	100	2900	110
2010	100	3000	90
2011	100	3100	80
2012	150	2900	100
2013	180	2800	100
2014	170	3200	100
2015	200	3500	100
2016	180	3200	100

2.0.6.3 Analysis Warnings Checks

Just so we can get a sense of the data, we want to see how many species are significantly increasing or decreasing over time for V and Q.

We'll use the below function to collect our info:

```
## function(Columns, temp) {
##
##   lm_check<-data.frame(col = rep_len(x = NA, length.out = length(Columns)),
##                        slope = rep_len(x = NA, length.out = length(Columns)),
##                        intercept = rep_len(x = NA, length.out = length(Columns)),
##                        R2 = rep_len(x = NA, length.out = length(Columns)),
##                        R2adj = rep_len(x = NA, length.out = length(Columns)),
##                        Pr = rep_len(x = NA, length.out = length(Columns)),
##                        Fstat = rep_len(x = NA, length.out = length(Columns)))
##
```

```
##   for (c0 in 1:length(Columns)) {
##     if (length(is.na(temp[,Columns[c0]])) == length(temp[,Columns[c0]])) {
##
##       lm_check$col[c0]<-NA
##       lm_check$slope[c0]<-NA
##       lm_check$intercept[c0]<-NA
##       lm_check$R2[c0]<-NA
##       lm_check$R2adj[c0]<-NA
##       lm_check$Pr[c0]<-NA
##       lm_check$Fstat[c0]<-NA
##
##     } else {
##
##     temp0<-summary(lm(rownames(temp) ~ temp[,Columns[c0]]))
##     lm_check$col[c0]<-Columns[c0]
##     lm_check$slope[c0]<-temp0$coefficients[2]
##     lm_check$intercept[c0]<-temp0$coefficients[1]
##     lm_check$R2[c0]<-temp0$r.squared
##     lm_check$R2adj[c0]<-temp0$adj.r.squared
##     lm_check$Pr[c0]<-temp0$coefficients[8]
##     lm_check$Fstat[c0]<-temp0$fstatistic[1]
##     }
##   }
##
##   lm_check$var<-substr(x = Columns, 1,1)
##   lm_check$slopecheck<-"Insig"
##   lm_check$slopecheck<-ifelse(lm_check$slope >= 0 & lm_check$Pr<=0.05, "Sig Pos", "Insig")
##   lm_check$slopecheck<-ifelse(lm_check$slope < 0 & lm_check$Pr<=0.05, "Sig Neg", lm_check$slopecheck)
##
##   return(lm_check)
## }
## <bytecode: 0x0000000013b30ce8>
```

	NameBasecategory	col	slope	intercept	R2	R2adj	Pr	Fstat	var	slopecheck
1	1_0Finfish	NA	NA	NA	NA	NA	NA	NA	V	NA
2	1_0Finfish	NA	NA	NA	NA	NA	NA	NA	V	NA
3	1_0Finfish	NA	NA	NA	NA	NA	NA	NA	V	NA
4	1_0Finfish	NA	NA	NA	NA	NA	NA	NA	Q	NA
5	1_0Finfish	NA	NA	NA	NA	NA	NA	NA	Q	NA
6	1_0Finfish	NA	NA	NA	NA	NA	NA	NA	Q	NA

How many slopes are significantly increaseing or decreaseing

var	Freq

2.0.7 Value of species $VV_{t,i}$ where Q available

$R_{t,i}$, as defined and discussed in the subsequent step, will need to sum to 1 across all species in a category. Therefore, you will need to sum a new total of $V_{t,i}$ available (called $VV_{t,i}$) for the category using only values for species that were used to calculate $Q_{t,i}$ (called $V_{t,i,s,available}$).

$$VV_{t,i} = \sum_{s=1}^{n} (V_{t,i,s,available})$$

where:

- $VV_{t,i}$ is the new total of $V_{t,i}$ (called $VV_{t,i}$) for the category using only values for species that were used to calculate $Q_{t,i}$
- $V_{t,i,s,available}$ are the $V_{t,i,s}$ where $Q_{t,i,s}$ were able to be calculated

```
temp0<-data.frame(temp[,names(temp) %in% VVColumns],
                  rowSums(temp[,names(temp) %in% VVColumns], na.rm = T))#)
names(temp0)[ncol(temp0)]<-paste0("VV",NameBasecategory)
temp0<-data.frame(temp0)
temp[ncol(temp)+1]<-temp0[ncol(temp0)]
```

	V1_1Salmon	V1_2Cod	V1_4SeaBass	VV1_0Finfish
2007	100	2800	120	3020
2008	100	2700	120	2920
2009	100	2900	110	3110
2010	100	3000	90	3190
2011	100	3100	80	3280
2012	150	2900	100	3150
2013	180	2800	100	3080
2014	170	3200	100	3470
2015	200	3500	100	3800
2016	180	3200	100	3480

2.0.8 Revenue Share for each species ($R_{t,i,s}$; e.g., Salmon and Flounder)

$$R_{t,i,s} = V_{t,i,s}/VV_{t,i}$$

where:

- $R_{t,i,s}$ is the revenue share per individual species (s), category (i), for each year (t)
- $V_{t,i,s}$ is the value ($) per individual species (s), category (i), for each year (t)

Here we divide $V_{t,i,s}$ by $VV_{t,i}$ because $VV_{t,i}$ only includes species used to calculate $V_{t,i,s}$ as per the above price calculations.

```
tempR<-data.frame(data = rep_len(x = NA, length.out = nrow(temp)))
for (c in 1:length(QColumns)) {

        #for renaming the columns
NameBase<-substr(start = 2,
                         stop = nchar(QColumns[c]),
                         x = QColumns[c])

VV<-(temp[,names(temp) %in% paste0("VV", NameBasecategory)])  # sum of V where P was calculated
V0<-temp[,names(temp) %in% paste0("V", NameBase)] #V of species; to make sure its the same column
tempR[,c]<-V0/VV
names(tempR)[c]<-paste0("R", NameBase ) #name the column
}
```

```r
tempR<-data.frame(tempR)
temp<-cbind.data.frame(temp, tempR)
```

	R1_1Salmon	R1_2Cod	R1_4SeaBass
1	0.0331126	0.9271523	0.0397351
2	0.0342466	0.9246575	0.0410959
3	0.0321543	0.9324759	0.0353698
4	0.0313480	0.9404389	0.0282132
5	0.0304878	0.9451220	0.0243902
6	0.0476190	0.9206349	0.0317460
7	0.0584416	0.9090909	0.0324675
8	0.0489914	0.9221902	0.0288184
9	0.0526316	0.9210526	0.0263158
10	0.0517241	0.9195402	0.0287356

2.0.8.1 Analysis Warnings Checks

As an additional check, let's make sure that each row sums to 1.

	x
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1

Is there a warning?

No warning.

2.0.9 Revenue Share-Weighted Qunatity Changes for each species ($QCW_{t,i,s}$; e.g., Salmon and Flounder)

$$QCW_{t,i,s} = \frac{R_{t,i,s} + R_{s,t-1,i}}{2} * ln(\frac{Q_{t,i,s}}{Q_{s,t-1,i}}) = \frac{R_{t,i,s} + R_{s,t-1,i}}{2} * [ln(Q_{t,i,s}) - ln(Q_{s,t-1,i})]$$

Where:

- $QCW_{t,i,s}$ = Revenue share-weighted quantity change for a species (s)

Such that:

- category's (i) Quantity Change for each species (s) = $\frac{R_{t,i,s}+R_{s,t-1,i}}{2}$
- category's (i) Revenue Share for each species (s) = $ln(\frac{Q_{t,i,s}}{Q_{s,t-1,i}} = [ln(Q_{t,i,s}) - ln(Q_{s,t-1,i})]$

We use this *PriceChange* function. For all intensive purposes, replace P with Q:

```r
print(PriceChange)
```

```
## function(R0, P0) {
##   PCW<-rep_len(x = 0, length.out = length(P0))
##   for (t in 2:length(P0)) {
##     AverageR<-((R0[t]+R0[t-1])/2) #Average Revenue Share
##     PC<-ln(P0[t]/P0[t-1]) #Price Change
##     PCW[t]<-AverageR*PC #Revenue Share-Weighted Price Chage
##   }
##   return(PCW)
## }
## <bytecode: 0x000000001ed4e038>
```

```r
      #Find which columns in this table are price and revenue share columns
tempQCW<-data.frame(data = rep_len(x = NA, length.out = nrow(temp)))
for (c in 1:length(QColumns)){
#For nameing columns
NameBase<-substr(start = 2,
                         stop = nchar(QColumns[c]),
                         x = QColumns[c])

        # Calculate
Q0<-temp[, names(temp) %in% paste0("Q", NameBase)]
R0<-temp[, names(temp) %in% paste0("R", NameBase)] #to make sure its the same column
tempQCW[,c]<-PriceChange(R0, Q0)
names(tempQCW)[c]<-paste0("QCW", NameBase ) #name the column
}

temp<-cbind.data.frame(temp, tempQCW)
```

2.0.10 Quantity Changes for the category ($QC_{t,i}$; e.g., Finfish)

$$QC_{t,i} = ln(\frac{Q_{t,i}}{Q_{t-1,i}}) = \sum_{s=1}^{n}(QCW_{t,i,s})$$

Where:

- $QC_{t,i}$ = Quantity change for a category (i)

```r
temp[ncol(temp)+1]<-rowSums(tempQCW, na.rm = T)
names(temp)[ncol(temp)]<-paste0("QC", NameBasecategory)
```

	QCW1__1Salmon	QCW1__2Cod	QCW1__4SeaBass	QC1__0Finfish
1	0.0000000	0.0000000	0.0000000	0.0000000
2	0.0000000	-0.0474927	0.0073686	-0.0401241
3	0.0000000	0.0476292	-0.0109989	0.0366303
4	0.0000000	0.2089644	0.0000000	0.2089644
5	-0.0214306	-0.0384862	0.0000000	-0.0599168
6	0.0071203	-0.0397029	0.0000000	-0.0325827
7	-0.0046142	-0.1278630	0.0033828	-0.1290945
8	0.0000000	0.1279717	-0.0032286	0.1247431
9	-0.0048429	0.0392239	0.0029045	0.0372855
10	0.0211563	-0.0800763	0.0026235	-0.0562965

2.0.11 Quantity Index for the each category ($QI_{t,i}$)

We calculate the quantity index first by comparing by multiplying the previous years QI_{t-1} by that year's quantity change QC_t, where the QI of the first year $QI_{t=firstyear,i} = 1$

$$QI_{t,i} = QI_{t-1,i} * \exp(ln(\frac{Q_{t,i,s}}{Q_{t-1,i,s}})) = QI_{t-1,i} * \exp(QC_{t,i})$$

Where

$$QI_{i,t_{firstyear}} = 1$$

Note that the first row of this column is = 1

Then, to change the price index into base year dollars, we use the following equation:

$$QI_t = QI_t/QI_{t=baseyear}$$

```
tempQI<-PriceIndex(temp, BaseColName = NameBasecategory, baseyr, var = "QC")
temp[ncol(temp)+1]<-(tempQI)
names(temp)[ncol(temp)]<-paste0("QI", NameBasecategory)
```

	QI1_0Finfish
2007	0.8142640
2008	0.7822391
2009	0.8114241
2010	1.0000000
2011	0.9418429
2012	0.9116497
2013	0.8012406
2014	0.9076914
2015	0.9421740
2016	0.8905983

2.1 Redo Analysis for Shellfish

Now lets redo that whole analysis up to this point (via function) for the two species of the shellfish group, as we will need them for the next steps of this analysis.

We use this *ImplicitQuantityOutput* function to calculate the Implicit Quanity Output at Species and category Level:

```
print(ImplicitQuantityOutput.speciescat.q)
```

```
## function(temp, ii, baseyr, maxyr, minyr, pctmiss, warnings.list) {
##
##   ########Housekeeping
##   # Here I am just going to collect some housekeeping items
##   temp<-data.frame(temp)
##
##   NumberOfSpecies<-numbers0(x = c(0, strsplit(x =
##                                                   strsplit(x = names(temp)[1],
##                                                             split = "_")[[1]][2],
```

```
##                                                 split = "[a-zA-Z]")[[1]][1]))[1]
##
##
##     NameBaseTotal<-substr(x = sort(names(temp)[grep(x = names(temp),
##                                                      pattern = paste0(NumberOfSpecies, "Total"))], decre
##                           start = 3, stop = nchar(sort(names(temp)[grep(x = names(temp),
##                                                                          pattern = paste0(NumberOfSpecie
##                                                        decreasing = T)[1]))
##
##     QColumns0<-QColumns<-grep(pattern = paste0("Q", ii,"_"),
##                               x = substr(x = names(temp),
##                                          start = 1,
##                                          stop = (2+nchar(ii))))
##
##     VColumns0<-VColumns<-grep(pattern = paste0("V", ii,"_"),
##                               x = substr(x = names(temp),
##                                          start = 1,
##                                          stop = (2+nchar(ii))))
##
##     NameBasecategory<-names(temp)[grepl(pattern = paste0("VE", ii,"_"),
##                                         x = substr(x = names(temp),
##                                                    start = 1,
##                                                    stop = (3+nchar(ii))))]
##
##     NameBasecategory<-substr(x = NameBasecategory, start = 3, stop = nchar(NameBasecategory))
##
##
##
##
##
##     ###Remove any V and Q data where V column has less data than the specifed $pctmiss$
##
##     # intersectcol<-base::intersect(x = substr(x = names(temp[,VColumns]), start = 2, stop = nchar(name
##     #                               y = substr(x = names(temp[,QColumns]), start = 2, stop = nchar(name
##     # unioncol<-base::union(x = substr(x = names(temp[,VColumns]), start = 2, stop = nchar(names(temp[
##     #                       y = substr(x = names(temp[,QColumns]), start = 2, stop = nchar(names(temp[
##     #
##     # differentcol<-base::setdiff(unioncol, intersectcol)
##     #
##     # if (length(differentcol) != 0) {
##     #   for (i in 1:length(differentcol)) {
##     #     names(temp)[grep(pattern = differentcol[i], x = names(temp))]<-
##     #       paste0("REMOVED_", names(temp)[grep(pattern = differentcol[i], x = names(temp))])
##     #   }
##     # }
##
##
##     for (i in 1:length(VColumns)) {
##
##       #if the percent missing is less in V or Q columns for a species than the percentmissingtrheshold
##       if (sum(is.na(temp[VColumns[i]]))/nrow(temp) > pctmiss | #V
##           sum(is.na(temp[QColumns[i]]))/nrow(temp) > pctmiss ) {#Q
##
##         names(temp)[VColumns[i]]<-paste0("REMOVED_", names(temp)[VColumns[i]])
```

```
##       VColumns0<-VColumns0[!(VColumns0 %in% VColumns[i])]
##       names(temp)[QColumns[i]]<-paste0("REMOVED_", names(temp)[QColumns[i]])
##       QColumns0<-QColumns0[!(QColumns0 %in% QColumns[i])]
##     }
##   }
##
##   if (length(VColumns0) == 0 #& length(QColumns0) == 0
##       ) {
##
##     warnings.list[length(warnings.list)+1]<-paste0("FYI: ", NameBasecategory, " is no longer being ca
##
##   } else {
##
##     VColumns<-names(temp)[VColumns0]
##     QColumns<-names(temp)[QColumns0]
##
##   ###Caluclate Category Sums of $V$ and $Q$
##
##   # Because we removed some columns for not meeting a perecent missing threshold of `r pctmiss`% and
##
##   # Q
##   temp.q<-data.frame(temp[,QColumns])
##   if (ncol(temp.q)>1) {
##     temp.q<-rowSums(temp.q, na.rm = T)
##   }
##   temp[ncol(temp)+1]<-temp.q
##   names(temp)[ncol(temp)]<-paste0("Q",NameBasecategory)
##
##   # V
##   temp.v<-data.frame(temp[,VColumns])
##   if (ncol(temp.v)>1) {
##     temp.v<-rowSums(temp.v, na.rm = T)
##   }
##   temp[ncol(temp)+1]<-temp.v
##   names(temp)[ncol(temp)]<-paste0("V",NameBasecategory)
##
##
##   # There may be instances where there are no or too few Q data for that species in a year or ever. 
## #### 1. If there are instances for a species where there are too few pairs of $Q$ are completely miss
##   # Let's say here that if `r pctmiss*100 `% of the data is missing in a given $Q_{t,i,s}$, don't use
##   #Find which columns in this table are price Columns
##
##   cc<-c() #Empty
##   for (c in 1:length(QColumns)) {
##     #If price could never be caluclated at any point in the timeseries (is 0/NaN/NA) for a column (c)
##     #Remove the column from the analysis.
##     #We will not be removing the column from the data, but simply remove it from the varaible "QColum
##     if (sum(temp[,QColumns[c]] %in% c(0, NA, NaN))/nrow(temp) > pctmiss) {
##       cc<-c(cc, c)#Collect offending columns
##     }
##   }
##   if (length(cc)>0){
##     QColumns<-QColumns[-cc]
##   }
```

```
##
##
##   #CHECK IF ANALYSIS FAILED:
##
##   if (length(QColumns) < 2) {
##
##     warnings.list[length(warnings.list)+1]<-paste0("FYI: ", NameBasecategory, " is no longer being ca
##
##   } else {
##
##
##     #### 2. If the first value of $Q_{t,i,s}$ is 0/NA in a timeseries, we (impute) let the next avail
##     # $$where \begin{cases} if: Q_{t,i=1} = 0, then: Q_{t,i=1} = Q_{t,i=1+1...} \\ if: Q_{t,i\neq1} =
##     temp<-ReplaceFirst(colnames = QColumns, temp)
##
##     #### 3. If there is a value in the middle of $P_{t,i,s}$'s timeseries that is 0/NA, we (impute) l
##     temp<-ReplaceMid(colnames = QColumns, temp)
##
##
##     ###Impute values of $V_{t,i,s}$ where P was able to be calculated
##
##     # To ensure that the price index does not rise or fall to quickly with changes (that are really l
##     # $$where \begin{cases} if: V_{t,i=1} = 0, then: V_{t,i=1} = V_{t,i=1+1...} \\ if: V_{t,i\neq1} =
##
##       #### 1. If the first value of $V_{t,i,s}$ is 0/NA in a timeseries, we let the next available no
##
##     QColumns<-QColumns[!(grepl(pattern = "REMOVED", x = QColumns))]
##     VVColumns<-paste0("V", substr(x = QColumns, start = 2, stop = nchar(QColumns)))
##     temp<-ReplaceFirst(colnames = VVColumns, temp)
##
##     #### 2. If there is a value in the middle of $V_{t,i,s}$'s timeseries that is 0/NA, we let the mo
##     temp<-ReplaceMid(colnames = VVColumns, temp)
##
##     ###Value of species $VV_{t,i}$ where Q available
##     # $R_{t,i}$, as defined and discussed in the subsequent step, will need to sum to 1 across all sp
##     # $$VV_{t,i} = \sum_{s=1}^{n}(V_{t,i,s, available})$$
##     #   where:
##     #   - $VV_{t,i}$ is the new total of $V_{t,i}$ (called $VV_{t,i}$) for the category using only va
##     #   - $V_{t,i,s, available}$ are the $V_{t,i,s}$ where $Q_{t,i,s}$ were able to be calculated
##
##     temp0<-data.frame(temp[,names(temp) %in% VVColumns],
##                       # ifelse(length(VVColumns) == 1,
##                               # temp[,names(temp) %in% VVColumns],
##                               rowSums(temp[,names(temp) %in% VVColumns], na.rm = T))#)
##     names(temp0)[ncol(temp0)]<-paste0("VV",NameBasecategory)
##     temp0<-data.frame(temp0)
##     temp[ncol(temp)+1]<-temp0[ncol(temp0)]
##
##
##     ####Analysis Warnings Checks
##
##     # Just so we can get a sense of the data, we want to see how many species are significantly incre
##     # We'll use the below function to collect our info:
##
```

```
##     Columns<-c(paste0("V", substr(x = QColumns, start = 2, stop = nchar(QColumns))),
##                QColumns)
##
##       lm_check<-data.frame(NameBasecategory, lmCheck(Columns, temp))
##
##     warnings.list[length(warnings.list)+1]<-list(lm_check)
##     names(warnings.list)[[length(warnings.list)]]<-paste0("FYI ", NameBasecategory, " species lm_che
##
##     # How many slopes are significantly increaseing or decreaseing
##
##     lm_sig_slope <- data.frame(table(lm_check[, c("var", "slopecheck")]))
##     lm_sig_slope <- lm_sig_slope[order(lm_sig_slope$var),]
##
##     warnings.list[length(warnings.list)+1]<-list(lm_sig_slope)
##     names(warnings.list)[[length(warnings.list)]]<-paste0("FYI ", NameBasecategory, " species lm_sig
##
##
##
##     ###Revenue Share for each species ($R_{t,i,s}$; e.g., Salmon and Flounder)
##
##     # $$R_{t,i,s} = V_{t,i,s}/VV_{t,i}$$
##     #   where:
##     #   - $R_{t,i,s}$ is the revenue share per individual species (s), category (i), for each year (
##     #   - $V_{t,i,s}$ is the value ($) per individual species (s), category (i), for each year (t)
##     # Here we divide $V_{t,i,s}$ by $VV_{t,i}$ because $VV_{t,i}$ only includes species used to calc
##
##     tempR<-data.frame(data = rep_len(x = NA, length.out = nrow(temp)))
##     for (c in 1:length(QColumns)) {
##
##       #for renaming the columns
##       NameBase<-substr(start = 2,
##                        stop = nchar(QColumns[c]),
##                        x = QColumns[c])
##
##       VV<-(temp[,names(temp) %in% paste0("VV", NameBasecategory)])  # sum of V where P was calculate
##       V0<-temp[,names(temp) %in% paste0("V", NameBase)] #V of species; to make sure its the same col
##       tempR[,c]<-V0/VV
##       names(tempR)[c]<-paste0("R", NameBase ) #name the column
##     }
##
##     tempR<-data.frame(tempR)
##     temp<-cbind.data.frame(temp, tempR)
##
##     #Note if there is an error
##     if (sum(rowSums(tempR, na.rm = T)) != nrow(temp)) {
##       warnings.list[length(warnings.list)+1]<-paste0("FYI: Rows of R_{s,i,t} for ",NameBasecategory,
##     }
##
##     #remove duplicates
##     temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##     temp <- temp[, !duplicated(colnames(temp))]
##     temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##     temp <- temp[, !duplicated(colnames(temp))]
##
```

```
##
##
##
##
##
##     ###Revenue Share-Weighted Qunatity Changes for each species ($QCW_{t,i,s}$; e.g., Salmon and Flou
##
##  #    $$QCW_{t,i,s} = \frac{R_{t,i,s} + R_{s,t-1,i}}{2} * ln(\frac{Q_{t,i,s}}{Q_{s,t-1,i}}) = \frac{R
##  #      Where:
##  #      - $QCW_{t,i,s}$ = Revenue share-weighted quantity change for a species (s)
##  #    Such that:
##  #      - category's (i) Quantity Change for each species (s) = $\frac{R_{t,i,s} + R_{s,t-1,i}}{2}$
##  #      - category's (i) Revenue Share for each species (s) = $ln(\frac{Q_{t,i,s}}{Q_{s,t-1,i}} = [ln
##  #    We use this *PriceChange* function. For all intensive purposes, replace $P$ with $Q$:
##
##     #Find which columns in this table are price and revenue share columns
##     tempQCW<-data.frame(data = rep_len(x = NA, length.out = nrow(temp)))
##     for (c in 1:length(QColumns)){
##       #For nameing columns
##       NameBase<-substr(start = 2,
##                        stop = nchar(QColumns[c]),
##                        x = QColumns[c])
##
##       # Calculate
##       Q0<-temp[, names(temp) %in% paste0("Q", NameBase)]
##       R0<-temp[, names(temp) %in% paste0("R", NameBase)] #to make sure its the same column
##       tempQCW[,c]<-PriceChange(R0, Q0)
##       names(tempQCW)[c]<-paste0("QCW", NameBase ) #name the column
##     }
##
##     temp<-cbind.data.frame(temp, tempQCW)
##
##
##     ###Quantity Changes for the category ($QC_{t,i}$; e.g., Finfish)
##
##     # $$QC_{t,i} = ln(\frac{Q_{t,i}}{Q_{t-1,i}}) = \sum_{s=1}^n(QCW_{t,i,s}) $$
##     # Where:
##     # - $QC_{t,i}$ = Quantity change for a category (i)
##
##     temp[ncol(temp)+1]<-rowSums(tempQCW, na.rm = T)
##     names(temp)[ncol(temp)]<-paste0("QC", NameBasecategory)
##
##
##     #remove duplicates
##     temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##     temp <- temp[, !duplicated(colnames(temp))]
##     temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##     temp <- temp[, !duplicated(colnames(temp))]
##
##
##     ###Quantity Index for the each category ($QI_{t,i}$)
##
##     # We calculate the quantity index first by comparing by multiplying the previous years $QI_{t-1}$
##     # $$QI_{t,i} = QI_{t-1,i}*\exp(ln(\frac{Q_{t,i,s}}{Q_{t-1,i,s}})) = QI_{t-1,i}*\exp(QC_{t,i})$$
```

```
##     # Where
##     # $$QI_{i, t_{first year}} = 1$$
##
##     #Note that the first row of this column is = 1
##
##     # Then, to change the price index into base year dollars, we use the following equation:
##     # $$QI_{t} = QI_{t}/QI_{t = baseyear}$$
##
##     tempQI<-PriceIndex(temp, BaseColName = NameBasecategory, baseyr, var = "QC")
##     temp[ncol(temp)+1]<-(tempQI)
##     names(temp)[ncol(temp)]<-paste0("QI", NameBasecategory)
##
##
##
##
##
##     #remove duplicates
##     temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##     temp <- temp[, !duplicated(colnames(temp))]
##     temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##     temp <- temp[, !duplicated(colnames(temp))]
##
##
##
##     #############WARNINGS
##
##
##   }
##   }
##
##   return(list(temp, warnings.list))
## }
## <bytecode: 0x0000000016d2bd40>
```

```
ii<-2 #The category index value

tempS<-ImplicitQuantityOutput.speciescat.q(temp, ii, baseyr, maxyr, minyr,
                         pctmiss = pctmiss,
                         warnings.list = warnings.list)
temp<-cbind.data.frame(temp, tempS[[1]])
warnings.list<-tempS[[2]]
###Remove duplicate columns
temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
  temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
  temp <- temp[, !duplicated(colnames(temp))]
```

What does the Shellfish data look like?

	R2_1Shrimp	R2_2Clam	QCW2_1Shrimp	QCW2_2Clam	QC2_0Shellfish	QI2_0Shellfish
2007	0.4444444	0.5555556	0.0000000	0.0000000	0.0000000	1.0918916
2008	0.4545455	0.5454545	0.0819526	0.0355288	0.1174814	1.2280076
2009	0.5000000	0.5000000	-0.0415282	-0.0698005	-0.1113287	1.0986305
2010	0.4375000	0.5625000	-0.0940644	0.0000000	-0.0940644	1.0000000
2011	0.5000000	0.5000000	-0.0552108	0.0000000	-0.0552108	0.9462857

	R2__1Shrimp	R2__2Clam	QCW2__1Shrimp	QCW2__2Clam	QC2__0Shellfish	QI2__0Shellfish
2012	0.5263158	0.4736842	0.1145079	0.0000000	0.1145079	1.0610904
2013	0.5454545	0.4545455	0.0000000	0.0000000	0.0000000	1.0610904
2014	0.5500000	0.4500000	0.0522040	-0.1090710	-0.0568670	1.0024330
2015	0.5000000	0.5000000	-0.1053521	0.0793507	-0.0260014	0.9767043
2016	0.5217391	0.4782609	0.0538255	0.1015627	0.1553882	1.1408993

2.1.1 Value for all fisheries for species where Q was able to be calculated

$R_{t,i}$, defined and discussed in the subsequent step, will need to sum to 1 across all species in a category. Therefore, you will need to sum a new total of $V_{t,i}$ (called VV_t) for the category using only values for species that were used to calculate $QI_{t,i}$.

$$VV_t = \sum_{s=1}^{n} (VV_{t,i})$$

where:

- VV_t is the new total of $V_{t,i}$ for the entire fishery using only values for species that were used to calculate $P_{t,i}$

VV1	__0Finfish VV2	__0Shellfish VV0	__0Total
2007	3020	1800	4820
2008	2920	2200	5120
2009	3110	1800	4910
2010	3190	1600	4790
2011	3280	1800	5080
2012	3150	1900	5050
2013	3080	2200	5280
2014	3470	2000	5470
2015	3800	2000	5800
2016	3480	2300	5780

V1__	0Finfish V2__	0Shellfish
2007	2800	1800
2008	2820	2200
2009	3010	1800
2010	3190	700
2011	3280	900
2012	3150	1000
2013	3080	2200
2014	3370	2000
2015	3700	2000
2016	3380	2300

2.1.2 Revenue Share for the each category ($R_{t,i}$)

$$R_{t,i} = V_{t,i}/V_t$$

where:

- $R_{t,i}$ is the revenue share per individual species (s), category (i), for each year (t)
- $V_{t,i}$ is the value ($) per individual species (s), category (i), for each year (t)

Here, we don't use VV_t beacause we want to expand the proportion to include all of the species caught, regardless if they were used in the quantity calculations.

```
tempR<-data.frame(data = rep_len(x = NA, length.out = nrow(temp)))

for (i in 1:length(category)) {

    CatCol<-names(temp)[grep(pattern = paste0("V", category[i],"_", NumberOfSpecies),
                             x = substr(x = names(temp),
                                        start = 1,
                                        stop = nchar(paste0("V", category[i],"_", NumberOfSpecies))))]
    NameBasecategory<-substr(x = CatCol, start = 2, stop = nchar(CatCol))

    tempR[,i]<-temp[,paste0("V", NameBasecategory)]/temp[,paste0("V", NameBaseTotal)]
    names(tempR)[i]<-paste0("R", NameBasecategory)
}
temp<-cbind.data.frame(temp, tempR)
```

	R1_0Finfish	R2_0Shellfish	V1_0Finfish	V2_0Shellfish	V0_0Total
2007	0.6086957	0.3913043	2800	1800	4600
2008	0.5617530	0.4382470	2820	2200	5020
2009	0.6257796	0.3742204	3010	1800	4810
2010	0.8200514	0.1799486	3190	700	3890
2011	0.7846890	0.2153110	3280	900	4180
2012	0.7590361	0.2409639	3150	1000	4150
2013	0.5833333	0.4166667	3080	2200	5280
2014	0.6275605	0.3724395	3370	2000	5370
2015	0.6491228	0.3508772	3700	2000	5700
2016	0.5950704	0.4049296	3380	2300	5680

2.1.2.1 Analysis Warnings Checks

As an additional check, let's make sure that each row sums to 1.

	x
2007	1
2008	1
2009	1
2010	1
2011	1
2012	1
2013	1
2014	1
2015	1
2016	1

Is there a warning?

No warning.

2.1.3 Revenue Share-Weighted Qunatity Changes for each category ($QCW_{t,i}$; e.g., Finfish and Shellfish)

$$QCW_{t,i} = \frac{R_{t,i,s} + R_{s,t-1,i}}{2} * ln(\frac{QI_{t,i,s}}{QI_{s,t-1,i}}) = \frac{R_{t,i,s} + R_{s,t-1,i}}{2} * [ln(QI_{t,i,s}) - ln(QI_{s,t-1,i})]$$

Where:

- $QCW_{t,i}$ = Revenue share-weighted quantity change for each category (i)

Such that:

- category's (i) Quantity Change for each category (i) = $\frac{R_{t,i}+R_{t-1,i}}{2}$
- category's (i) Revenue Share for each category (i) = $ln(\frac{QI_{t,i}}{QI_{t-1,i}} = [ln(QI_{t,i}) - ln(QI_{t-1,i})]$

```
#Find which columns in this table are price and revenue share columns
tempQCW<-data.frame(data = rep_len(x = NA, length.out = nrow(temp)))
for (i in 1:length(category)) {

    CatCol<-names(temp)[grep(pattern = paste0("VE", category[i], "_", NumberOfSpecies),
                             x = substr(x = names(temp),
                                        start = 1,
                                        stop = nchar(paste0("VE", category[i], "_", NumberOfSpecies))))]
    NameBasecategory<-substr(x = CatCol, start = 3, stop = nchar(CatCol))

    # Calculate
    Q0<-temp[, names(temp) %in% paste0("QI", NameBasecategory)]
    R0<-temp[, names(temp) %in% paste0("R", NameBasecategory)] #to make sure its the same column
    tempQCW[,i]<-PriceChange(R0, Q0)
    names(tempQCW)[i]<-paste0("QCW", NameBasecategory ) #name the column
}

temp<-cbind.data.frame(temp, tempQCW)
```

2.1.4 Quantity Changes for the entire fishery (QC_t)

$$QC_t = ln(\frac{QI_{t,i}}{QI_{t-1,i}}) = \sum_{s=1}^{n}(QCW_{t,i})$$

Where:

- QC_t = Quantity change for the entire fishery

```
temp[ncol(temp)+1]<-rowSums(tempQCW, na.rm = T)
names(temp)[ncol(temp)]<-paste0("QC", NameBaseTotal)
```

	QCW1_0Finfish	QCW2_0Shellfish	QC0_0Total
1	0.0000000	0.0000000	0.0000000
2	-0.0234816	0.0487284	0.0252468
3	0.0217499	-0.0452255	-0.0234756
4	0.1510636	-0.0260638	0.1249998
5	-0.0480755	-0.0109113	-0.0589868

	QCW1_0Finfish	QCW2_0Shellfish	QC0_0Total
6	-0.0251493	0.0261235	0.0009742
7	-0.0866462	0.0000000	-0.0866462
8	0.0755254	-0.0224371	0.0530883
9	0.0238009	-0.0094036	0.0143972
10	-0.0350218	0.0587217	0.0236999

2.1.5 Quantity Index for the entier fishery (QI_t)

We calculate the quantity index first by comparing by multiplying the previous years QI_{t-1} by that year's quantity change QC_t, where the QI of the first year $QI_{t=firstyear,i} = 1$

$$QI_t = QI_{t-1} * \exp(ln(\frac{Q_{t,i}}{Q_{t-1,i}})) = QI_{t-1} * \exp(QC_t)$$

Where

$$QI_{t_{firstyear}} = 1$$

Note that the first row of this column is = 1

Then, to change the price index into base year dollars, we use the following equation:

$$QI_t = QI_t / QI_{t=baseyear}$$

```
tempQI<-PriceIndex(temp, BaseColName = NameBaseTotal, baseyr, var = "QC")
temp[ncol(temp)+1]<-(tempQI)
names(temp)[ncol(temp)]<-paste0("QI", NameBaseTotal)
```

	QI0_0Total
2007	0.8809353
2008	0.9034593
2009	0.8824970
2010	1.0000000
2011	0.9427192
2012	0.9436381
2013	0.8653175
2014	0.9124970
2015	0.9257294
2016	0.9479312

2.1.6 Sum Total Simple Sum Quantity Output Index

$$QEI_t = QE_t / QE_{t=baseyr}$$

Where:

- QE_t is the sum of Q before these calculations; the simple sum
- QEI_t is the index of the sum of Q before these equations

```r
  #remove duplicates
  temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
  temp <- temp[, !duplicated(colnames(temp))]
  temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
  temp <- temp[, !duplicated(colnames(temp))]


  QE<-names(temp)[names(temp) %in% paste0("QE", NameBaseTotal)]
  temp[,ncol(temp)+1]<-temp[,QE]/temp[,QE][rownames(temp) %in% baseyr]
  names(temp)[ncol(temp)]<-paste0("QEI", NameBaseTotal)
```

	QE0_0Total	QEI0_0Total
1	3250	1.2452107
2	3380	1.2950192
3	3150	1.2068966
4	2610	1.0000000
5	2490	0.9540230
6	2412	0.9241379
7	3251	1.2455939
8	3431	1.3145594
9	3630	1.3908046
10	3575	1.3697318

2.2 Other Analysis Warnings Checks

To make sure our analyses worked as inteded, let's see if we can back calculate our numbers.

We want the calcuated V to equal this check:

2.2.0.1 When back calculated, growth rate?

$$ln(Q_t/Q_{t-1}) = \sum((\frac{R_{i,t} + R_{i,t-1}}{2}) * ln(\frac{Q_{t,i}}{Q_{t-1,i}}))$$

```r
#Part 1
  names0<-c(paste0("QI",NameBaseTotal))
  for (i in 1:length(category)) {
    names0<-c(names0,
              names(temp)[grep(pattern = paste0("QI", category[i], "_", NumberOfSpecies), names(temp))]
              names(temp)[grep(pattern = paste0("R", category[i], "_", NumberOfSpecies), names(temp))])
  }

  temp0<-temp[,names0]

  temp0[,(ncol(temp0)+1)]<-c(NA, ln(temp0[-nrow(temp0),paste0("QI",NameBaseTotal)]/
                                        temp0[-1,paste0("QI",NameBaseTotal)]))
  names(temp0)[ncol(temp0)]<-"part1"

  #Part 2
  temp00<-data.frame()
  for (i in 1:length(category)) {
```

```
    R0<-temp0[,grepl(pattern = paste0("R", category[i]), x = names(temp0))]
    Q0<-temp0[,grepl(pattern = paste0("QI", category[i]), x = names(temp0))]

    for (r in 2:(nrow(temp))){
      temp00[r,i]<-(((R0[r] + R0[r-1])/2) * ln(Q0[r] / Q0[r-1]) )
    }
    names(temp00)[i]<-paste0("ln", category[i])
  }

  temp0[,(ncol(temp0)+1)]<-rowSums(temp00)
  names(temp0)[ncol(temp0)]<-"part2"
```

	part1	part2
2007	NA	NA
2008	-0.0252468	0.0252468
2009	0.0234756	-0.0234756
2010	-0.1249998	0.1249998
2011	0.0589868	-0.0589868
2012	-0.0009742	0.0009742
2013	0.0866462	-0.0866462
2014	-0.0530883	0.0530883
2015	-0.0143972	0.0143972
2016	-0.0236999	0.0236999

Is there a warning?

```
  if (length(setdiff(as.character(temp0$part1),
                     as.character(temp0$part2))) != 0) {
  warnings.list[length(warnings.list)+1]<-"Warning: When back calculated, ln(Q_t/Q_{t-1}) = did not equa
    a<-warnings.list[length(warnings.list)][[1]]
} else {
    a<-"No warning."
}
```

Warning: When back calculated, ln(Q_t/Q_{t-1}) = did not equal sum(((R_{i, t} - R_{i, t-1})(2)) ln((Q_{t,i})(Q_{t-1,i}))*

2.2.1 View Total Outputs

	QE0_0Total	VE0_0Total	VV0_0Total	V0_0Total	QC0_0Total	QI0_0Total	QEI0_0Total
2007	3250	5600	4820	4600	0.0000000	0.8809353	1.2452107
2008	3380	6220	5120	5020	0.0252468	0.9034593	1.2950192
2009	3150	5710	4910	4810	-0.0234756	0.8824970	1.2068966
2010	2610	3890	4790	3890	0.1249998	1.0000000	1.0000000
2011	2490	4180	5080	4180	-0.0589868	0.9427192	0.9540230
2012	2412	4150	5050	4150	0.0009742	0.9436381	0.9241379
2013	3251	6280	5280	5280	-0.0866462	0.8653175	1.2455939
2014	3431	6270	5470	5370	0.0530883	0.9124970	1.3145594
2015	3630	6700	5800	5700	0.0143972	0.9257294	1.3908046
2016	3575	6780	5780	5680	0.0236999	0.9479312	1.3697318

2.2.2 Missing Data

```
#value
a<-temp
a<-a[,grep(pattern = "V[1-9]+_[1-9]+", x = names(a))]
if(length(grep(pattern = "REMOVED_", x = names(a)) &
          grep(pattern = "Total", x = names(a), ignore.case = T)) != 0 ){
  a<-a[,-c(grep(pattern = "REMOVED_", x = names(a)), grep(pattern = "Total", x = names(a), ignore.case
}
ncol0<-ncol(a)
aa<-0
a<-data.frame(a)
if(ncol(a) != 0){
  for (iii in 1:ncol(a)) {
    aa<-c(aa, ifelse(sum(a[,iii] %in% c(NA, NaN, 0)) == nrow(a), iii, NA))
  }
  vv<-(aa[!(is.na(aa))])
} else {
  pp<-0
}
#quantity
a<-temp
a<-a[,grep(pattern = "Q[1-9]+_[1-9]+", x = names(a))]
if(length(grep(pattern = "REMOVED_", x = names(a)) &
          grep(pattern = "Total", x = names(a), ignore.case = T)) != 0 ){
  a<-a[,-c(grep(pattern = "REMOVED_", x = names(a)), grep(pattern = "Total", x = names(a), ignore.case
}
ncol0<-ncol(a)
aa<-0
a<-data.frame(a)
if(ncol(a) != 0){
  for (iii in 1:ncol(a)) {
    aa<-c(aa, ifelse(sum(a[,iii] %in% c(NA, NaN, 0)) == nrow(a), iii, NA))
  }
  qq<-(aa[!(is.na(aa))])
} else {
  pp<-0
}
```

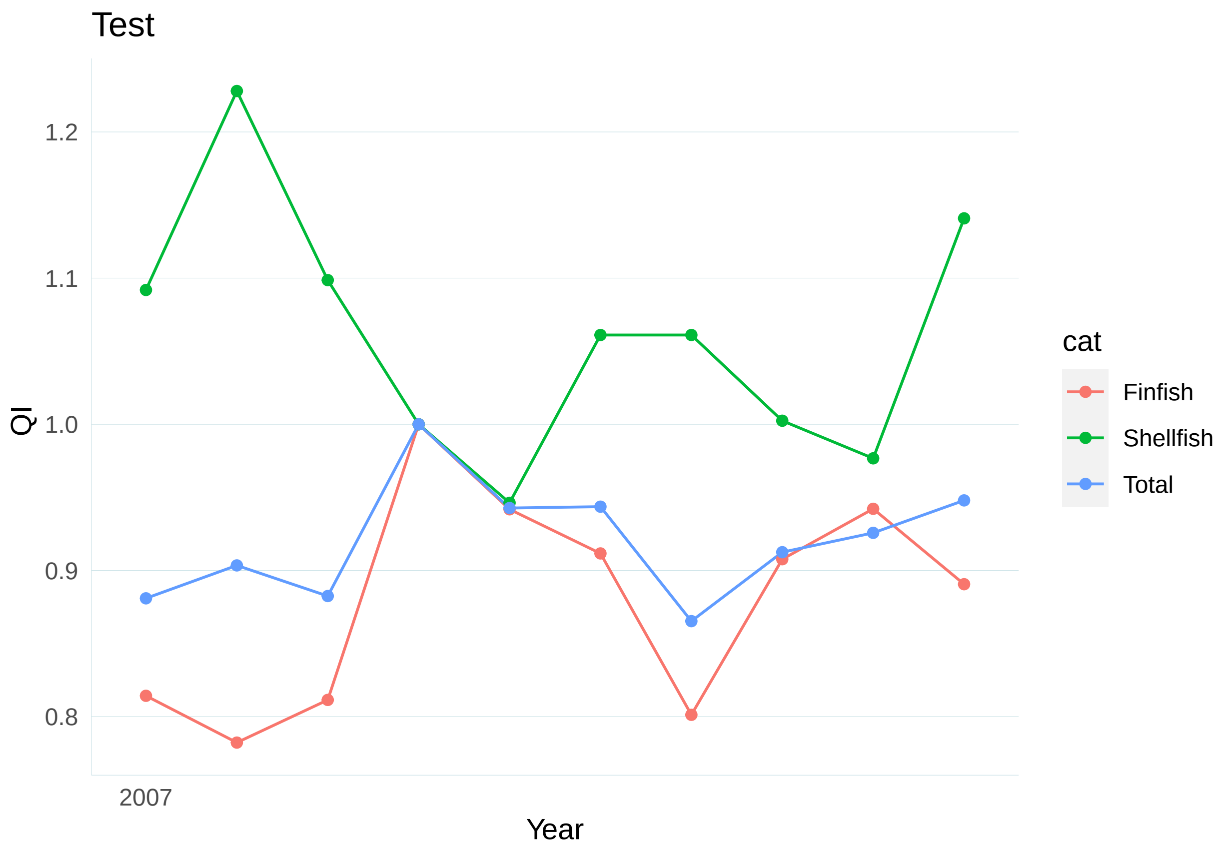
FYI: 0 of species V columns are completely empty, 1 of species Q columns are completely empty.

2.2.3 Graph 1: Price Index

In theory, PI should be negative slope after the baseyear and positive after the base year, but because this data was fabricated without thinking of this, we don't see that here. The index value for the base year is = 1, however.

```
title00<- "_QI-Line"

a0<-data.frame(temp[,grepl(
  pattern = paste0("QI[0-9]+_", NumberOfSpecies),
  x = names(temp))])

a0$Year<-rownames(a0)
```

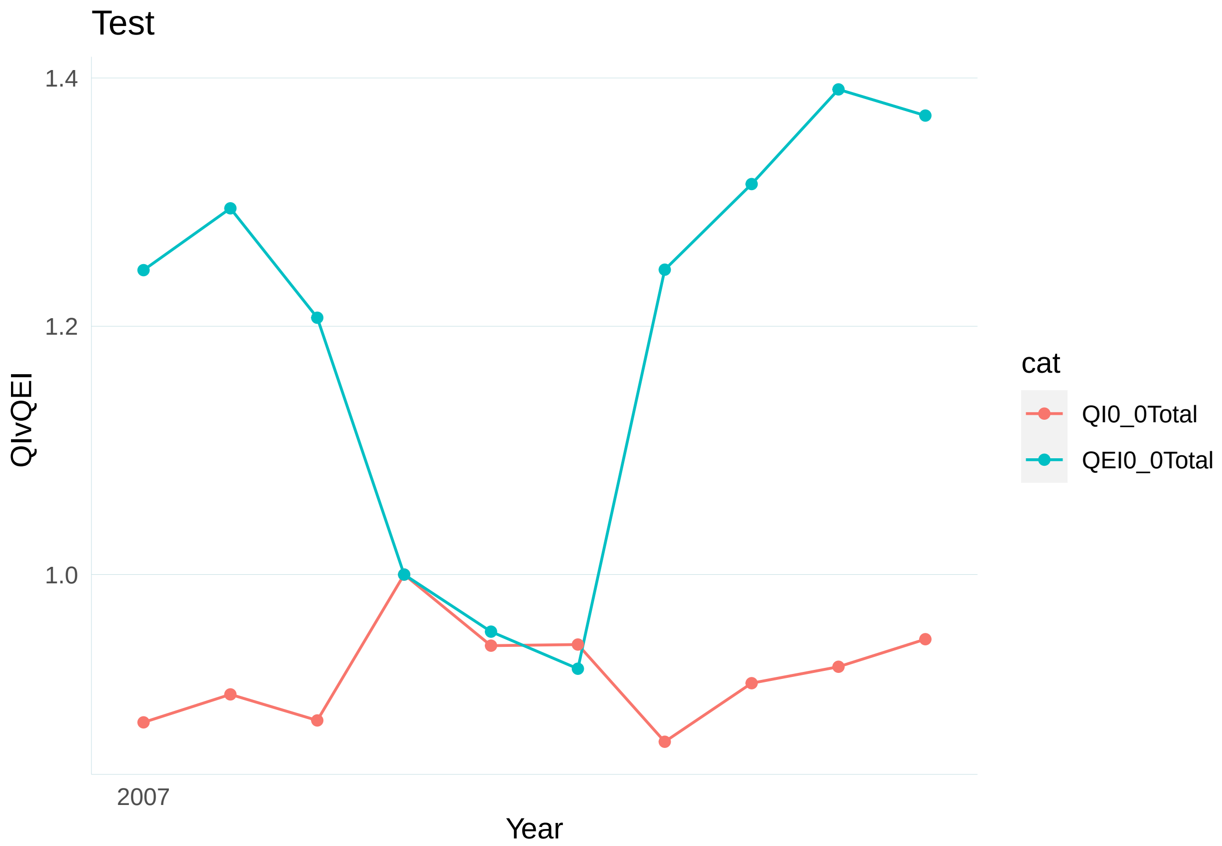
```r
a <- gather(a0, Category, val, names(a0)[grepl(pattern = NumberOfSpecies, x = names(a0))], factor_key

a$cat<-as.character(lapply(X = strsplit(x = as.character(a$Category), split = paste0("_", NumberOfSpe

temp0<-a

plotnlines(dat = temp0, title00, place)
```



2.2.4 Graph 2: Quantity Index Compare

```r
title00<- "_QIvQEI-Line"

temp0<-temp
temp0$Year<-rownames(temp0)

temp0<-data.frame(temp0[,names(temp0) %in% c("Year",
                                              paste0("QI", NameBaseTotal),
                                              paste0("QEI", NameBaseTotal))])
temp0$Year<-rownames(temp)

temp0<-gather(temp0, cat, val,
              names(temp0)[1]:names(temp0)[length(names(temp0))-1],
              factor_key = T)
```

```
plotnlines(dat = temp0,  title00, place)
```



2.2.5 Graph 3: Quantity Compare

```
title00<- "_QvQE-Line"

temp0<-temp
temp0$Year<-rownames(temp0)

temp0<-data.frame(temp0[,names(temp0) %in% c("Year",
                                             paste0("Q", NameBaseTotal),
                                             paste0("QE", NameBaseTotal))])
temp0$Year<-rownames(temp)

temp0<-gather(temp0, cat, val,
              names(temp0)[1]:names(temp0)[length(names(temp0))-1],
              factor_key = T)

plotnlines(dat = temp0,  title00, place)
```

2.3 Do same analysis via a function!

Now that we know the method, we can simplify most of it into a function and do this whole analysis in 4 easy steps:

- A. Import and Edit data
- B. Enter base year
- C. Run the function
- D. Obtain the implicit quantity estimates

2.3.1 Function

We use this *ImplicitQuantityOutput* function to calculate the Implicit Quanity Output at Fishery Level:

```
print(ImplicitQuantityOutput.q)
```

```
## function(temp, baseyr, pctmiss = 1.00,
##                                   title0 = "", place = ""){
##
##   temp.orig<-temp
##
##   warnings.list<-list()
##   figures.list<-list()
##
```

```
##    ########Housekeeping
##    # Here I am just going to collect some housekeeping items
##    temp<-data.frame(temp)
##
##    NumberOfSpecies<-numbers0(x = c(0, strsplit(x =
##                                                 strsplit(x = names(temp)[2],
##                                                          split = "_")[[1]][2],
##                                               split = "[a-zA-Z]")[[1]][1]))[1]
##
##
##
##
##    category<-unique(as.character(lapply(X = strsplit(x = as.character(names(temp)),
##                                                     split = paste0("_")),
##                                        function(x) x[1])))
##    category<-unique(substr(x = category, start = 2, stop = nchar(category)))
##    category<-category[!grepl(pattern = "[a-zA-Z]", x = category)]
##    category<-category[!(category %in% numbers0(c(0, (category)[1]))[1])]
##
##    temp0<-data.frame(rep_len(x = NA, length.out = nrow(temp)))
##    tempQCW<-data.frame(rep_len(x = NA, length.out = nrow(temp0)))
##    tempQC<-data.frame(rep_len(x = NA, length.out = nrow(temp0)))
##
##    maxyr<-max(rownames(temp))
##    minyr<-min(rownames(temp))
##
##    category<-category00<-sort((category))
##    category.rm<-c()
##
##
##    NameBaseTotal<-paste0(paste(rep_len(x = 0, length.out = nchar(category[1])), collapse = ""),
##                          "_", NumberOfSpecies, "Total")
##
##    for (ii in 1:length(category)) {
##
##      QColumns0<-QColumns<-grep(pattern = paste0("Q", category[ii],"_"),
##                                x = substr(x = names(temp),
##                                           start = 1,
##                                           stop = (2+nchar(category[ii]))))
##
##      VColumns0<-VColumns<-grep(pattern = paste0("V", category[ii],"_"),
##                                x = substr(x = names(temp),
##                                           start = 1,
##                                           stop = (2+nchar(category[ii]))))
##
##      NameBasecategory<-names(temp)[grepl(pattern = paste0("VE", category[ii],"_"),
##                                          x = substr(x = names(temp),
##                                                     start = 1,
##                                                     stop = (3+nchar(category[ii]))))]
##
##      NameBasecategory<-substr(x = NameBasecategory, start = 3, stop = nchar(NameBasecategory))
##
##
##      if (length(VColumns0) < 2) {
```

```
##
##       warnings.list[length(warnings.list)+1]<-paste0("FYI: ", NameBasecategory, " is no longer being
##       category.rm<-c(category.rm, ii)
##
##     }  else  {
##
##     #if there are still columns to assess that haven't been "removed"
##     # if (length(VColumns) != 0) {
##     ###Append species and category level calculations
##     temp00<-ImplicitQuantityOutput.speciescat.q(temp, ii=category[ii],
##                                                             baseyr, maxyr, minyr,
##                                                             pctmiss, warnings.list)
##     warnings.list1<-temp00[[2]]
##     warnings.list1<-unique(warnings.list1)
##
##
##     #If data for a catagory is no longer available after precentmissingthreshold etc, remove it from
##     if (sum(names(temp00[1][[1]]) %in% paste0("QI", NameBasecategory)) == 0) {
##       category.rm<-c(category.rm, ii)
##     } else {
##       temp1<-temp00[[1]]
##       #remove duplicates
##       temp1<-temp1[, !(grepl(pattern = "\\.[0-9]+", x = names(temp1)))]
##       temp1 <- temp1[, !duplicated(colnames(temp1))]
##       temp0<-cbind.data.frame(temp0, temp1)
##     }
##
##     ###Remove duplicate columns
##     temp0<-temp0[, !(grepl(pattern = "\\.[0-9]+", x = names(temp0)))]
##
##     # warnings.list1<-ImplicitQuantityOutput.speciescat.q(temp, ii=category[ii],
##     #                                                                 baseyr, maxyr, minyr,
##     #                                                                 pctmiss, warnings.list)[[2]]
##
##     warnings.list<-c(warnings.list, warnings.list1)
##
##     }
##
##
##   }
##
##   if(!(is.null(category.rm))) {
##     category<-category[-category.rm]
##   }
##
##   temp<-temp0#[,2:ncol(temp0)]
##
##
##
##   #remove duplicates
##   temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##   temp <- temp[, !duplicated(colnames(temp))]
##   temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##   temp <- temp[, !duplicated(colnames(temp))]
```

```
##
##
##   ###Value for all fisheries for species where Q was able to be calculated
##   # $R_{t,i}$, defined and discussed in the subsequent step, will need to sum to 1 across all species
##   # $$VV_{t} = \sum_{s=1}^{n}(VV_{t,i})$$
##   #   where:
##   #   - $VV_{t}$ is the new total of $V_{t,i}$ for the entire fishery using only values for species t
##
##   #Total VV
##   temp0<-data.frame(temp[,grep(pattern = paste0("VV", "[0-9]+_", NumberOfSpecies),
##                               x = names(temp))],
##                     rowSums(temp[,grep(pattern = "VV", x = names(temp))], na.rm = T))
##   names(temp0)[ncol(temp0)]<-paste0("VV",NameBaseTotal)
##   temp0<-data.frame(temp0)
##   temp[ncol(temp)+1]<-temp0[ncol(temp0)]
##
##
##   #Total V
##   # names(temp)[names(temp) %in% paste0("V", NameBaseTotal)]<-paste0("REMOVED_V", NameBaseTotal)
##
##   temp0<-temp[grep(x = names(temp),
##                    pattern = paste0("V[0-9]+_", NumberOfSpecies))]
##   temp0<-temp0[,!(grepl(x = names(temp0), pattern = c("VV")))]
##   temp0<-temp0[,!(grepl(x = names(temp0), pattern = c("REMOVED_")))]
##
##   temp[ncol(temp)+1]<-rowSums(temp0, na.rm = T)
##   names(temp)[ncol(temp)]<-paste0("V", NameBaseTotal)
##
##
##
##
##   ###Revenue Share for the each category ($R_{t,i}$)
##   # $$R_{t,i} = V_{t,i}/V_{t}$$
##   #   where:
##   #   - $R_{t,i}$ is the revenue share per individual species (s), category (i), for each year (t)
##   #   - $V_{t,i}$ is the value ($) per individual species (s), category (i), for each year (t)
##   # Here, we don't use $VV_{t}$ beacause we want to expand the proportion to include all of the speci
##   tempR<-data.frame(data = rep_len(x = NA, length.out = nrow(temp)))
##
##   for (i in 1:length(category)) {
##
##     CatCol<-names(temp)[grep(pattern = paste0("V", category[i],"_", NumberOfSpecies),
##                             x = substr(x = names(temp),
##                                        start = 1,
##                                        stop = nchar(paste0("V", category[i],"_", NumberOfSpecies)))
##     NameBasecategory<-substr(x = CatCol, start = 2, stop = nchar(CatCol))
##
##     tempR[,i]<-temp[,paste0("V", NameBasecategory)]/temp[,paste0("V", NameBaseTotal)]
##     names(tempR)[i]<-paste0("R", NameBasecategory)
##   }
##   temp<-cbind.data.frame(temp, tempR)
##
##   # Is there a warning?
##   if (sum(rowSums(tempR, na.rm = T)) != nrow(temp)) {
```

```
##     warnings.list[length(warnings.list)+1]<-paste0("Warning: Rows of R_{t,i} for ",NameBaseTotal," di
##     a<-warnings.list[length(warnings.list)][[1]]
##   } else {
##     a<-"No warning."
##   }
##
##
##
##
##
##   #remove duplicates
##   temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##   temp <- temp[, !duplicated(colnames(temp))]
##   temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##   temp <- temp[, !duplicated(colnames(temp))]
##
##
##
##
##
##
##   ###Revenue Share-Weighted Qunatity Changes for each category ($QCW_{t,i}$; e.g., Finfish and Shelli
##  #  $$QCW_{t,i} = \frac{R_{t,i,s} + R_{s,t-1,i}}{2} * ln(\frac{QI_{t,i,s}}{QI_{s,t-1,i}}) = \frac{R_
##  #    Where:
##  #    - $QCW_{t,i}$ = Revenue share-weighted quantity change for each category (i)
##  #  Such that:
##  #    - category's (i) Quantity Change for each category (i) = $\frac{R_{t,i} + R_{t-1,i}}{2}$
##  #    - category's (i) Revenue Share for each category (i) = $ln(\frac{QI_{t,i}}{QI_{t-1,i}} = [ln(QI
##
##
##   #Find which columns in this table are price and revenue share columns
##   tempQCW<-data.frame(data = rep_len(x = NA, length.out = nrow(temp)))
##   for (i in 1:length(category)) {
##
##     CatCol<-names(temp)[grep(pattern = paste0("VE", category[i], "_", NumberOfSpecies),
##                              x = substr(x = names(temp),
##                                         start = 1,
##                                         stop = nchar(paste0("VE", category[i], "_", NumberOfSpecies)
##     NameBasecategory<-substr(x = CatCol, start = 3, stop = nchar(CatCol))
##
##     # Calculate
##     Q0<-temp[, names(temp) %in% paste0("QI", NameBasecategory)]
##     R0<-temp[, names(temp) %in% paste0("R", NameBasecategory)] #to make sure its the same column
##     tempQCW[,i]<-PriceChange(R0, Q0)
##     names(tempQCW)[i]<-paste0("QCW", NameBasecategory ) #name the column
##   }
##
##   temp<-cbind.data.frame(temp, tempQCW)
##
##
##
##
##   ###Quantity Changes for the entire fishery ($QC_{t}$)
##   # $$QC_{t} = ln(\frac{QI_{t,i}}{QI_{t-1,i}}) = \sum_{s=1}^n(QCW_{t,i}) $$
```

```
##   #   Where:
##   #   - $QC_{t}$ = Quantity change for the entire fishery
##
##   temp[ncol(temp)+1]<-rowSums(tempQCW, na.rm = T)
##   names(temp)[ncol(temp)]<-paste0("QC", NameBaseTotal)
##
##
##   #remove duplicates
##   temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##   temp <- temp[, !duplicated(colnames(temp))]
##   temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##   temp <- temp[, !duplicated(colnames(temp))]
##
##
##   ###Quantity Index for the entier fishery ($QI_{t}$)
##   # We calculate the quantity index first by comparing by multiplying the previous years $QI_{t-1}$ b
##   # $$QI_{t} = QI_{t-1}*\exp(ln(\frac{Q_{t,i}}{Q_{t-1,i}})) = QI_{t-1}*\exp(QC_{t})$$
##   # Where
##   # $$QI_{t_{first year}} = 1$$
##   # Note that the first row of this column is = 1
##   # Then, to change the price index into base year dollars, we use the following equation:
##   # $$QI_{t} = QI_{t}/QI_{t = baseyear}$$
##
##   tempQI<-PriceIndex(temp, BaseColName = NameBaseTotal, baseyr, var = "QC")
##   temp[ncol(temp)+1]<-(tempQI)
##   names(temp)[ncol(temp)]<-paste0("QI", NameBaseTotal)
##
##
##
##
##
##     #remove duplicates
##   temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##   temp <- temp[, !duplicated(colnames(temp))]
##   temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##   temp <- temp[, !duplicated(colnames(temp))]
##
##
##
##   ### Sum Total Simple Sum Quantity Output Index
##   # $$QEI_t = QE_t/QE_{t=baseyr}$$
##   #   Where:
##   #   - $QE_t$ is the sum of Q before these calculations; the simple sum
##   #   - $QEI_t$ is the index of the sum of Q before these equations
##
##   QE<-names(temp)[names(temp) %in% paste0("QE", NameBaseTotal)]
##   # QE<-QE[substr(x = QE, start = 1, stop = 2) %in% "QE"]
##   temp[,ncol(temp)+1]<-temp[,QE]/temp[,QE][rownames(temp) %in% baseyr]
##   names(temp)[ncol(temp)]<-paste0("QEI", NameBaseTotal)
##
##
##
##
##   ## Other Analysis Warnings Checks
```

```
##   # To make sure our analyses worked as inteded, let's see if we can back calculate our numbers.
##   # We want the calcuated V to equal this check:
##
##   ####When back calculated, growth rate?
##   # $$ln(Q_t/Q_{t-1}) = \sum( ( \frac{R_{i, t} + R_{i, t-1}}{2})  * ln(\frac{Q_{t,i}}{Q_{t-1,i}}))$$
##
##   #Part 1
##   names0<-c(paste0("QI",NameBaseTotal))
##   for (i in 1:length(category)) {
##   names0<-c(names0,
##   names(temp)[grep(pattern = paste0("QI", category[i], "_", NumberOfSpecies), names(temp))],
##   names(temp)[grep(pattern = paste0("R", category[i], "_", NumberOfSpecies), names(temp))])
##   }
##
##   temp0<-temp[,names0]
##
##   temp0[,(ncol(temp0)+1)]<-c(NA, ln(temp0[-nrow(temp0),paste0("QI",NameBaseTotal)]/
##                                     temp0[-1,paste0("QI",NameBaseTotal)]))
##   names(temp0)[ncol(temp0)]<-"part1"
##
##   #Part 2
##   temp00<-data.frame()
##   for (i in 1:length(category)) {
##     R0<-temp0[,grepl(pattern = paste0("R", category[i]), x = names(temp0))]
##     Q0<-temp0[,grepl(pattern = paste0("QI", category[i]), x = names(temp0))]
##
##   for (r in 2:(nrow(temp))){
##   temp00[r,i]<-(((R0[r] + R0[r-1])/2) * ln(Q0[r] / Q0[r-1]) )
##   }
##     names(temp00)[i]<-paste0("ln", category[i])
##   }
##
##   temp0[,(ncol(temp0)+1)]<-rowSums(temp00)
##   names(temp0)[ncol(temp0)]<-"part2"
##
##
##   if (length(setdiff(as.character(temp0$part1),
##                      as.character(temp0$part2))) != 0) {
##     warnings.list[length(warnings.list)+1]<-"Warning: When back calculated, ln(Q_t/Q_{t-1}) = did no
##   }
##
##
##
##   ####4. Missing Data
##
##   #value
##   a<-temp
##   a<-a[,grep(pattern = "V[1-9]+_[1-9]+", x = names(a))]
##   if(length(grep(pattern = "REMOVED_", x = names(a)) &
##             grep(pattern = "Total", x = names(a), ignore.case = T)) != 0 ){
##     a<-a[,-c(grep(pattern = "REMOVED_", x = names(a)), grep(pattern = "Total", x = names(a), ignore.
##   }
##   ncol0<-ncol(a)
##   aa<-0
```

```
##    a<-data.frame(a)
##    if(ncol(a) != 0){
##      for (iii in 1:ncol(a)) {
##        aa<-c(aa, ifelse(sum(a[,iii] %in% c(NA, NaN, 0)) == nrow(a), iii, NA))
##      }
##      vv<-(aa[!(is.na(aa))])
##    } else {
##      pp<-0
##    }
##
##    #quantity
##    a<-temp
##    a<-a[,grep(pattern = "Q[1-9]+_[1-9]+", x = names(a))]
##    if(length(grep(pattern = "REMOVED_", x = names(a)) &
##              grep(pattern = "Total", x = names(a), ignore.case = T)) != 0 ){
##      a<-a[,-c(grep(pattern = "REMOVED_", x = names(a)), grep(pattern = "Total", x = names(a), ignore.c
##    }
##    ncol0<-ncol(a)
##    aa<-0
##    a<-data.frame(a)
##    if(ncol(a) != 0){
##      for (iii in 1:ncol(a)) {
##        aa<-c(aa, ifelse(sum(a[,iii] %in% c(NA, NaN, 0)) == nrow(a), iii, NA))
##      }
##      qq<-(aa[!(is.na(aa))])
##    } else {
##      pp<-0
##    }
##
##
##
##    warnings.list[length(warnings.list)+1]<-paste0("FYI: ", ifelse(length(vv)==1, 0, length(vv)-1) ,
##                                                   " of species V columns are completely empty, ",
##                                                   ifelse(length(qq)==1, 0, length(qq)-1) ,
##                                                   " of species Q columns are completely empty.")
##
##
##
##    ###########################################GRAPHS
##    #remove duplicates
##    temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##    temp <- temp[, !duplicated(colnames(temp))]
##    temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##    temp <- temp[, !duplicated(colnames(temp))]
##
##
##    #####Calculated Q by Species
##    title00<- "_QI-Category"
##
##    a0<-data.frame(temp[,grepl(
##      pattern = paste0("QI[0-9]+_", NumberOfSpecies),
##      x = names(temp))])
##
##    a0$Year<-rownames(temp)
```

```
##
##    a <- gather(a0, Category, val, names(a0)[grepl(pattern = NumberOfSpecies, x = names(a0))], factor_
##
##    a$cat<-as.character(lapply(X = strsplit(x = as.character(a$Category), split = paste0("_", NumberOf
##
##    temp0<-a
##
##    g<-plotnlines(dat = temp0, title00, place)
##
##    figures.list[[length(figures.list)+1]]<-g
##    names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##
##    #####Summed Q By Species
##    title00<- "_QE-Category"
##
##    a0<-data.frame(temp[,grepl(
##      pattern = paste0("QE[0-9]+_", NumberOfSpecies),
##      x = names(temp))])
##
##    a0$Year <- rownames(temp)
##
##    a <- gather(a0, Category, val, names(a0)[grepl(pattern = NumberOfSpecies, x = names(a0))], factor_
##
##    a$cat<-as.character(lapply(X = strsplit(x = as.character(a$Category), split = paste0("_", NumberOf
##
##    temp0<-a
##
##    g<-plotnlines(dat = temp0, title00, place)
##
##    figures.list[[length(figures.list)+1]]<-g
##    names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##    #####Qunantity Index
##    title00<- "_QI-Line"
##
##    a0<-data.frame(temp[,grepl(
##      pattern = paste0("QI[0-9]+_", NumberOfSpecies),
##      x = names(temp))])
##
##    a0$Year <- rownames(temp)
##
##    a <- gather(a0, Category, val, names(a0)[grepl(pattern = NumberOfSpecies, x = names(a0))], factor_
##
##    a$cat<-as.character(lapply(X = strsplit(x = as.character(a$Category), split = paste0("_", NumberOf
##
##    temp0<-a
##
##    g<-plotnlines(dat = temp0, title00, place)
##
##    figures.list[[length(figures.list)+1]]<-g
##    names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##
```

```
##   ########VV
##   title00<- "_VV-Line"
##
##   a0<-data.frame(temp[,grepl(
##     pattern = paste0("VV[0-9]+_", NumberOfSpecies),
##     x = names(temp))])
##
##   a0$Year <- rownames(temp)
##
##   a <- gather(a0, Category, val, names(a0)[grepl(pattern = NumberOfSpecies, x = names(a0))], factor_
##
##   a$cat<-as.character(lapply(X = strsplit(x = as.character(a$Category),
##                                           split = paste0("_", NumberOfSpecies)), function(x) x[2]))
##
##   temp0<-a
##
##   g<-plotnlines(dat = temp0, title00, place)
##
##   figures.list[[length(figures.list)+1]]<-g
##   names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##
##
##
##   ########VE
##   title00<- "_VE-Line"
##
##   a0<-data.frame(temp[,grepl(
##     pattern = paste0("VE[0-9]+_", NumberOfSpecies),
##     x = names(temp))])
##   a0<-a0[,!grepl(pattern = "REMOVED_", x = names(a0))]
##   # a0<-a0[,!grepl(pattern = "VV[0-9]+_", x = names(a0))]
##
##   a0$Year <- rownames(temp)
##
##   a <- gather(a0, Category, val, names(a0)[grepl(pattern = NumberOfSpecies, x = names(a0))], factor_
##
##   a$cat<-as.character(lapply(X = strsplit(x = as.character(a$Category), split = paste0("_", NumberOf
##
##   temp0<-a
##
##   g<-plotnlines(dat = temp0, title00, place)
##
##   figures.list[[length(figures.list)+1]]<-g
##   names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##
##   ########V
##   title00<- "_V-Line"
##
##   a0<-data.frame(temp[,grepl(
##     pattern = paste0("V[0-9]+_", NumberOfSpecies),
##     x = names(temp))])
##   a0<-a0[,!grepl(pattern = "REMOVED_", x = names(a0))]
```

```
##    a0<-a0[,!grepl(pattern = "VV[0-9]+_", x = names(a0))]
##
##    a0$Year <- rownames(temp)
##
##    a <- gather(a0, Category, val, names(a0)[grepl(pattern = NumberOfSpecies, x = names(a0))], factor_
##
##    a$cat<-as.character(lapply(X = strsplit(x = as.character(a$Category),
##                                            split = paste0("_", NumberOfSpecies)),
##                              function(x) x[2]))
##
##    temp0<-a
##
##    g<-plotnlines(dat = temp0, title00, place)
##
##    figures.list[[length(figures.list)+1]]<-g
##    names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##
##
##    ########V and VV
##    title00<- "_VvVV-Line"
##
##    a0<-data.frame(temp[,grepl(
##      pattern = paste0("V[0-9]+_", NumberOfSpecies),
##      x = names(temp))])
##    a0<-a0[,!grepl(pattern = "REMOVED_", x = names(a0))]
##
##    a0$Year <- rownames(temp)
##
##    a <- gather(a0, Category, val, names(a0)[grepl(pattern = NumberOfSpecies, x = names(a0))], factor_
##
##    a$cat<-paste0(as.character(lapply(X = strsplit(x = as.character(a$Category), split = paste0("_", Nu
##                  "_",
##                  as.character(lapply(X = strsplit(x = as.character(a$Category), split = paste0("_", Nu
##
##    temp0<-a
##
##    g<-plotnlines(dat = temp0, title00, place)
##
##    # figures.list[[length(figures.list)+1]]<-g
##    # names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##
##    ########VE
##    title00<- "_VE-Line"
##
##    a0<-data.frame(temp[,grepl(
##      pattern = paste0("VE[0-9]+_", NumberOfSpecies),
##      x = names(temp))])
##
##    a0$Year <- rownames(temp)
##
##    temp0<-a0
##
```

```
##   temp0<-gather(temp0, cat, val,
##                 names(temp0)[1]:names(temp0)[length(names(temp0))-1],
##                 factor_key = T)
##
##   temp0$cat<-paste0(as.character(lapply(X = strsplit(x = as.character(temp0$cat),
##                                                     split = paste0("_", NumberOfSpecies)), function
##                      "_",
##                      as.character(lapply(X = strsplit(x = as.character(temp0$cat),
##                                                     split = paste0("_", NumberOfSpecies)), function
##
##
##   g<-plotnlines(dat = temp0, title00, place)
##
##   figures.list[[length(figures.list)+1]]<-g
##   names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##   #### Quantity Index Compare
##   # For comparison, let's recreate those graphs to make sure we are getting the same output:
##   title00<-"_QIvQEI-Line"
##
##   temp0<-temp
##   temp0$Year<-rownames(temp0)
##
##   temp0<-data.frame(temp0[,names(temp0) %in% c("Year",
##                                                paste0("QI", NameBaseTotal),
##                                                paste0("QEI", NameBaseTotal))])
##   temp0$Year<-rownames(temp)
##
##   temp0<-gather(temp0, cat, val,
##                 names(temp0)[1]:names(temp0)[length(names(temp0))-1],
##                 factor_key = T)
##
##   temp0$cat<-paste0(as.character(lapply(X = strsplit(x = as.character(temp0$cat),
##                                                     split = paste0("_", NumberOfSpecies)), function
##                      "_",
##                      as.character(lapply(X = strsplit(x = as.character(temp0$cat),
##                                                     split = paste0("_", NumberOfSpecies)), function
##
##   g<-plotnlines(dat = temp0, title00, place)
##
##   figures.list[[length(figures.list)+1]]<-g
##   names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##
##
##   #### Revenue Share
##   title00<-"_R-Line"
##   temp0<-temp
##
##   temp0<-temp0[grep(pattern = paste0("R[0-9]+_", NumberOfSpecies), x = names(temp0))]
##   temp0$Year<-rownames(temp)
##
##   temp0<-gather(temp0, cat, val,
##                 names(temp0)[1]:names(temp0)[length(names(temp0))-1],
```

```
##                factor_key = T)
##
##   plotnlines(dat = temp0, title00, place)
##
##   figures.list[[length(figures.list)+1]]<-g
##   names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##   #### Quantity Change
##   title00<-"_QC-Line"
##   temp0<-temp
##
##   temp0<-temp0[grep(pattern = paste0("QC[0-9]+_", NumberOfSpecies), x = names(temp0))]
##   temp0$Year<-rownames(temp)
##
##   temp0<-gather(temp0, cat, val,
##                 names(temp0)[1]:names(temp0)[length(names(temp0))-1],
##                 factor_key = T)
##
##   temp0$cat<-paste0(as.character(lapply(X = strsplit(x = as.character(temp0$cat),
##                                                     split = paste0("_", NumberOfSpecies)), function
##                     "_",
##                     as.character(lapply(X = strsplit(x = as.character(temp0$cat),
##                                                     split = paste0("_", NumberOfSpecies)), function
##
##   g<-plotnlines(dat = temp0, title00, place)
##
##   figures.list[[length(figures.list)+1]]<-g
##   names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##
##
##   #################Number Missing V Per Year
##   title00<- "_VNumberMissing-Line"
##
##   a0<-data.frame(temp.orig[,grepl(
##     pattern = paste0("V[0-9]+_"),
##     x = names(temp.orig)) &
##       !(grepl(
##         pattern = paste0("V[0-9]+_", NumberOfSpecies),
##         x = names(temp.orig)))])
##
##   total.no.v<-ncol(a0)*nrow(a0)
##
##   cat0<-data.frame(names0 = (names(temp.orig)[grepl(pattern = paste0("VE[0-9]+_", NumberOfSpecies),
##                                                    x = names(temp.orig))]))
##
##   cat0$no<-as.character(lapply(X = strsplit(x = as.character(cat0$names0), split = "_"),
##                               function(x) x[1]))
##   cat0$numberofspp<-NA
##   for (i in 1:nrow(cat0)) {
##     cat0$numberofspp[i]<-length(grep(pattern = cat0$no[i],
##                                     x = substr(x = names(a0),
##                                                start = 1,
##                                                stop = max(nchar(cat0$no))) ))
```

```
##   }
##
##
##   cat0$no<-as.character(gsub(pattern = "[a-zA-Z]", replacement = "", x = cat0$no))
##
##   cat0$catname<-as.character(lapply(X = strsplit(x = as.character(cat0$names0), split = NumberOfSpec
##                                     function(x) x[2]))
##   cat0$label<-paste0(cat0$catname, " (n=",  cat0$numberofspp,")")
##
##   a<-temp.orig
##   a$Year<-rownames(a)
##   a<-a[,!grepl(pattern = NumberOfSpecies, x = names(a))]
##   a00<-gather(data = a, spp, val, names(a)[1]:names(a)[length(names(a))-1], factor_key = T)
##   a00<-a00[grep(pattern = "V", x = substr(x = a00$spp, start = 1, stop = 1)),]
##   a00$na<-0
##   a00$na[(is.na(a00$val)) | a00$val %in% 0]<-1
##   a00$no<-gsub(pattern = "[a-zA-Z]", replacement = "", x = as.character(lapply(X = strsplit(x = as.c
##                                                                               function(x) x[1])) ) 
##   a00$x<-1
##
##   aa<-a00
##
##   a00<-merge(x = aa, y = cat0, by = "no")
##
##   #SUM
##   a<-aggregate(x = a00[,c("na", "x")],
##                by = list("Year" = a00$Year, "Category" = a00$catname),
##                FUN = sum, na.rm = T)
##
##   a<-rbind.data.frame(a,
##                       data.frame(Year = aggregate(x = a00$na, by = list("Year" = a00$Year), FUN = su
##                                  Category = "Total",
##                                  na = aggregate(x = a00$na, by = list("Year" = a00$Year), FUN = sum,
##                                  x = nrow(temp.orig)*ncol(temp.orig)))
##
##   a$x.perc<-a$na/a$x*100
##   a$bins<-round_any(a$x.perc, 10)
##
##   xnames<-as.numeric(paste0(a$Year))
##   xnames[!(xnames %in% seq(from = min((as.numeric(xnames))),
##                            to = max(as.numeric(xnames)),
##                            by = 10))]<-""
##
##   g<-ggplot(data = a, aes(x = factor(Year), y = na, color = Category)) +
##     geom_line(aes(group = Category)) +
##     geom_point() +
##     theme(
##       panel.grid.major.y = element_line( color=NOAALightBlue, size = .1 ),
##       panel.grid.minor.y = element_blank(),
##       panel.grid.major.x = element_blank(),
##       panel.grid.minor.x = element_blank(),
##       axis.line = element_line( color=NOAALightBlue, size = .1 ),
##       axis.ticks = element_blank(), # remove ticks
##       panel.background = element_blank()
```

```
##     ) +
##     scale_x_discrete(labels= xnames) +
##     guides(fill=FALSE) +
##     ggtitle(paste0(place, " ", title00))
##
##   figures.list[[length(figures.list)+1]]<-g
##   names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##   ################Percent Missing V
##   # title00<- "_PctMissingV_Line"
##   #
##   # g<-ggplot(data = a, aes(x = factor(Year), y = x.perc, color = Category)) +
##   #   geom_line(aes(group = Category)) +
##   #   geom_point() +
##   #   theme(
##   #     panel.grid.major.y = element_line( color=NOAALightBlue, size = .1 ),
##   #     panel.grid.minor.y = element_blank(),
##   #     panel.grid.major.x = element_blank(),
##   #     panel.grid.minor.x = element_blank(),
##   #     axis.line = element_line( color=NOAALightBlue, size = .1 ),
##   #     axis.ticks = element_blank(), # remove ticks
##   #     panel.background = element_blank()
##   #   ) +
##   #   scale_x_discrete(labels= xnames) +
##   #   guides(fill=FALSE) +
##   #   ggtitle(paste0(place, " ", title00))
##   #
##   # figures.list[[length(figures.list)+1]]<-g
##   # names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##
##   # How many V columns have X percentage data missing
##   title00<- "_VPctMissing-Bar"
##
##   a00<-data.frame(table(a[,names(a) %in% c("bins", "Category")]))
##
##   cat00<-merge(y = cat0[,c("catname", "numberofspp")],
##                x = a,
##                by.y = "catname", by.x = "Category")
##
##   cat00$label<-paste0(cat00$Category, " (n=",  cat00$numberofspp,")")
##
##   cat000<-data.frame(table(cat00[,names(cat00) %in% c("label", "bins")]))
##
##   xnames<-paste0(sort(as.numeric(paste(unique(a$bins)))), "%")
##
##   g<-ggplot(data = cat000, aes(x = factor(bins), y = Freq, fill = label)) +
##     geom_bar(stat="identity", position=position_dodge()) +
##     theme(
##       panel.grid.major.y = element_line( color=NOAALightBlue, size = .1 ),
##       panel.grid.minor.y = element_blank(),
##       panel.grid.major.x = element_blank(),
##       panel.grid.minor.x = element_blank(),
##       axis.line = element_line( color=NOAALightBlue, size = .1 ),
```

```
##       axis.ticks = element_blank(), # remove ticks
##       panel.background = element_blank()
##     )  +
##     scale_x_discrete(labels = xnames) +
##     ggtitle(paste0(place, " ", title00))
##
##   figures.list[[length(figures.list)+1]]<-g
##   names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##
##   ###OVERVIEW
##
##
##
##
##   #Species
##   spp.output<-list()
##   spptable0<-data.frame(Analysis  = title0,
##                         Place = place,
##                         Catagory = rep_len(x = NA, length.out = length(category)),
##                         TotCount = rep_len(x = NA, length.out = length(category)),
##                         RmCount = rep_len(x = NA, length.out = length(category)),
##                         UsedCount = rep_len(x = NA, length.out = length(category)))
##   cat1<-(as.character(lapply(X = strsplit(x = as.character(names(temp)),
##                                           split = paste0("_")),
##                              function(x) x[1])))
##   cat2<-(as.character(lapply(X = strsplit(x = as.character(names(temp)),
##                                           split = paste0("_")),
##                              function(x) x[2])))
##
##   for (i in 1:length(category)) {
##     #Orgionally
##     spp.pre<-unique(cat2[grep(pattern = paste0("V", category[i]), x = cat1)], cat2[grep(pattern = pa
##     cat.pre<-spp.pre[grep(pattern = NumberOfSpecies, x = spp.pre)]
##     cat.pre<-unique(gsub(pattern = "[0-9]", replacement = "", x = cat.pre))
##     spp.pre<-spp.pre[!grepl(pattern = NumberOfSpecies, x = spp.pre)]
##     spp.pre<-gsub(pattern = "[0-9]", replacement = "", x = spp.pre)
##     spp.pre<-sort(x = as.character(spp.pre), decreasing = F)
##
##     #In Analysis
##     spp.pst<-cat2[grep(pattern = paste0("R", category[i]), x = cat1)]
##     cat.pst<-spp.pst[grep(pattern = NumberOfSpecies, x = spp.pst)]
##     cat.pst<-gsub(pattern = "[0-9]", replacement = "", x = cat.pst)
##     spp.pst<-spp.pst[!grepl(pattern = NumberOfSpecies, x = spp.pst)]
##     spp.pst<-gsub(pattern = "[0-9]", replacement = "", x = spp.pst)
##     spp.pst<-sort(x = as.character(spp.pst), decreasing = F)
##
##     spp.output[[i]]<-list("pre" = spp.pre,
##                           "pst" = spp.pst)
##     names(spp.output)[[i]]<-cat.pst
##     spptable0$Catagory[i]<- cat.pst
##     spptable0$TotCount[i]<-length(spp.pre)
##     spptable0$UsedCount[i]<-ifelse(is.na(length(spp.pst)), 0, length(spp.pst))
##     spptable0$RmCount[i]<-spptable0$TotCount[i] - spptable0$UsedCount[i]
```

```
##   }
##
##   # #########Number Species Inc and Dec
##   #
##   # temp0<-temp
##   # temp0$Year<-rownames(temp0)
##   #
##   # temp0<-data.frame(temp0[,names(temp0) %in% c("Year",
##   #                                                names(temp)[substr(names(temp), start = 1, stop = 1
##   #                                                substr(names(temp)[grepl(pattern = "P", x = names(te
##   #                                                paste0("QEI", NameBaseTotal))])
##   # temp0$Year<-rownames(temp)
##   #
##   # temp0<-gather(temp0, cat, val,
##   #               names(temp0)[1]:names(temp0)[length(names(temp0))-1],
##   #               factor_key = T)
##   #
##   # temp0$cat<-paste0(as.character(lapply(X = strsplit(x = as.character(temp0$cat),
##   #                                                      split = paste0("_", NumberOfSpecies)), functi
##   #                   "_",
##   #                   as.character(lapply(X = strsplit(x = as.character(temp0$cat),
##   #                                                      split = paste0("_", NumberOfSpecies)), functi
##   #
##   # g<-plotnlines(dat = temp0, title00, place)
##   #
##   # figures.list[[length(figures.list)+1]]<-g
##   # names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##
##
##
##   # tables0<-list(spptable0, spp.output)
##
##   # return(list(temp, warnings.list, figures.list, tables0))
##   return(list(temp, warnings.list, figures.list, spptable0, spp.output))
##
## }
## <bytecode: 0x000000001e97f280>
```

2.3.2 A. Import and Edit data

```r
temp<-read.csv(file = paste0(dir.data, "Tornqvist Index-Calculations_OutputEx.csv"))
rownames(temp)<-temp$year
temp$year<-NULL

temp.q<-temp[,grepl(pattern = "Q", x = names(temp))]
temp.q$QE0_0Total<-rowSums(temp.q, na.rm = T)
temp.q$QE1_0Finfish<-rowSums(temp.q[,grepl(x = names(temp.q), pattern = "Q1") ], na.rm = T)
temp.q$QE2_0Shellfish<-rowSums(temp.q[,grepl(x = names(temp.q), pattern = "Q2") ], na.rm = T)

temp.v<-temp[,grepl(pattern = "V", x = names(temp))]
temp.v$VE0_0Total<-rowSums(temp.v, na.rm = T)
temp.v$VE1_0Finfish<-rowSums(temp.v[,grepl(x = names(temp.v), pattern = "V1") ], na.rm = T)
```

```
temp.v$VE2_0Shellfish<-rowSums(temp.v[,grepl(x = names(temp.v), pattern = "V2") ], na.rm = T)

temp<-orgional.data<-cbind.data.frame(temp.q, temp.v)
```

2.3.3 B. Enter base year

```
baseyr<-baseyr
```

2.3.4 C. Run the function

```
temp00<-ImplicitQuantityOutput.q(temp = orgional.data, baseyr, pctmiss)
temp<-temp00[[1]]
warnings.list0<-temp00[[2]]
figures.list0<-temp00[[3]]
```

2.3.5 D. Obtain the implicit quantity estimates

	QE0_0Total	VE0_0Total	VV0_0Total	V0_0Total	QC0_0Total	QI0_0Total	QEI0_0Total
2007	3250	5600	4820	4600	0.0000000	0.8809353	1.2452107
2008	3380	6220	5120	5020	0.0252468	0.9034593	1.2950192
2009	3150	5710	4910	4810	-0.0234756	0.8824970	1.2068966
2010	2610	3890	4790	3890	0.1249998	1.0000000	1.0000000
2011	2490	4180	5080	4180	-0.0589868	0.9427192	0.9540230
2012	2412	4150	5050	4150	0.0009742	0.9436381	0.9241379
2013	3251	6280	5280	5280	-0.0866462	0.8653175	1.2455939
2014	3431	6270	5470	5370	0.0530883	0.9124970	1.3145594
2015	3630	6700	5800	5700	0.0143972	0.9257294	1.3908046
2016	3575	6780	5780	5680	0.0236999	0.9479312	1.3697318

Did all of the analyses work as intended?

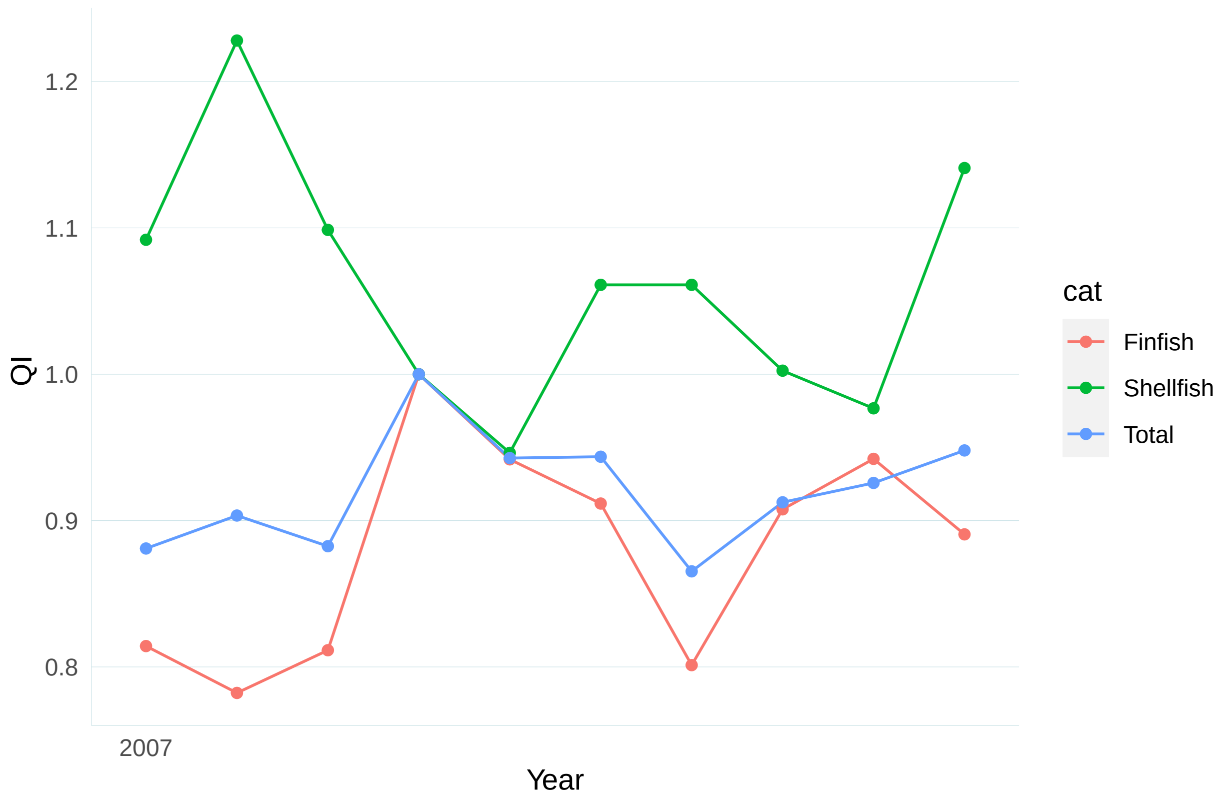
list(NameBasecategory = c(1, 1, 1, 1, 1, 1), col = c(NA, NA, NA, NA, NA, NA), slope = c(NA, NA, NA, NA, NA, NA), intercept = c(NA, NA, NA, NA, NA, NA), R2 = c(NA, NA, NA, NA, NA, NA), R2adj = c(NA, NA, NA, NA, NA, NA), Pr = c(NA, NA, NA, NA, NA, NA), Fstat = c(NA, NA, NA, NA, NA, NA), var = c("V", "V", "V", "Q", "Q", "Q"), slopecheck = c(NA, NA, NA, NA, NA, NA)), list(var = character(0), Freq = integer(0)), list(NameBasecategory = c(1, 1, 1, 1, 1, 1), col = c(NA, NA, NA, NA, NA, NA), slope = c(NA, NA, NA, NA, NA, NA), intercept = c(NA, NA, NA, NA, NA, NA), R2 = c(NA, NA, NA, NA, NA, NA), R2adj = c(NA, NA, NA, NA, NA, NA), Pr = c(NA, NA, NA, NA, NA, NA), Fstat = c(NA, NA, NA, NA, NA, NA), var = c("V", "V", "V", "Q", "Q", "Q"), slopecheck = c(NA, NA, NA, NA, NA, NA)), list(var = character(0), Freq = integer(0)), list(NameBasecategory = c(1, 1, 1, 1), col = c(NA, NA, NA, NA), slope = c(NA, NA, NA, NA), intercept = c(NA, NA, NA, NA), R2 = c(NA, NA, NA, NA), R2adj = c(NA, NA, NA, NA), Pr = c(NA, NA, NA, NA), Fstat = c(NA, NA, NA, NA), var = c("V", "V", "Q", "Q"), slopecheck = c(NA, NA, NA, NA)), Warning: When back calculated, ln(Q_t/Q_{t-1}) = did not equal sum((R_{i, t} - R_{i, t-1}) / 2) x ln((Q_{i,t}) / (Q_{i,t-1})), FYI: 0 of species V columns are completely empty, 2 of species Q columns are completely empty.

2.3.6 E. Graph

2.3.6.1 Graph 1: Price Index

For comparison, let's recreate those graphs to make sure we are getting the same output:

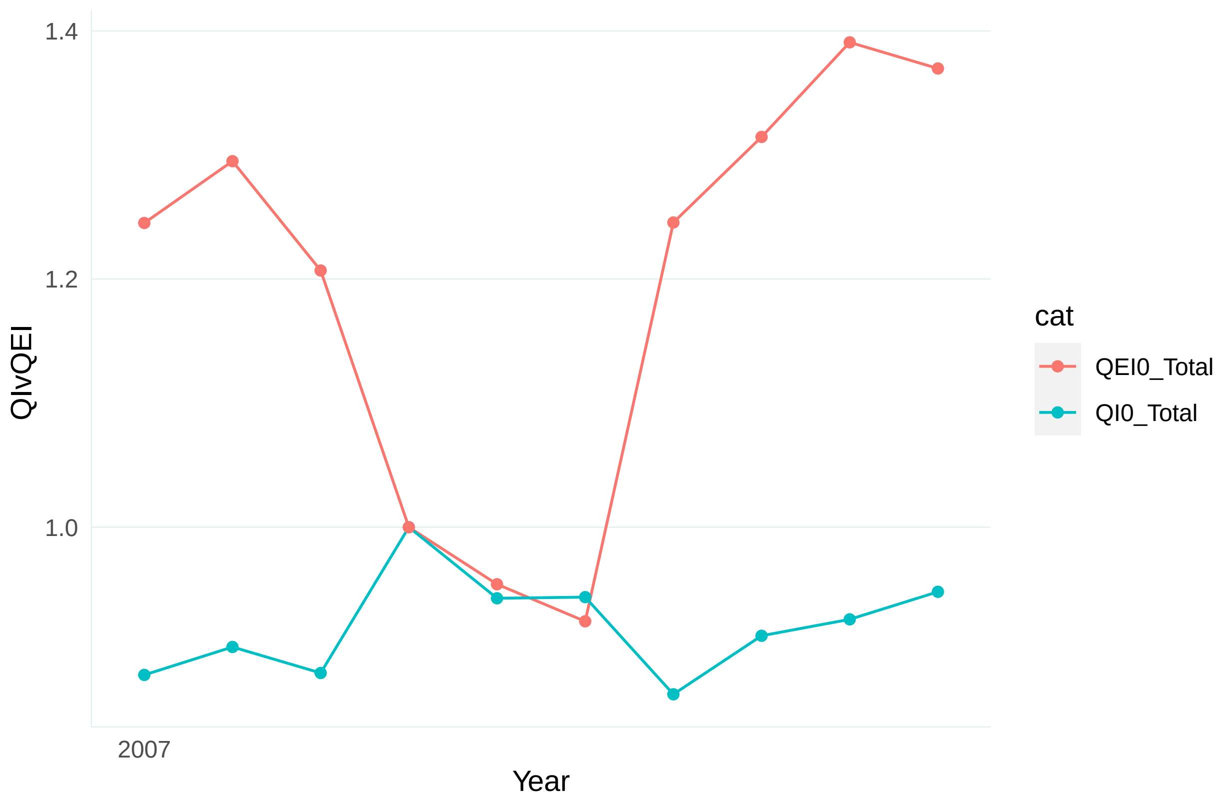
```
figures.list0$`_QI-Line`
```



2.3.6.2 Graph 2: Quantity Index Compare

For comparison, let's recreate those graphs to make sure we are getting the same output:

```
figures.list0$`_QIvQEI-Line`
```



2.3.6.3 Graph 3: Quantity Compare

```
figures.list0$`_VvVV-Line`
```

```
## NULL
```

2.4 Practice with real data (For National Data)

2.4.1 A. Import and Edit data

Load and subset Data

```
#Load Data (This data has been edited to include category columns)
landings.data<-read.csv(file = paste0(dir.data, "landings_edited.csv"))
landings.data<-landings.data[landings.data$Year < 2018,] #FUS 2018 hasn't been published yet
landings.data<-landings.data[landings.data$State %in% unique(state.codes$NAME),]
region<-"National"
#We'll categorize by this column I already added to the data
category0 = "category.orig"
```

Summary information about the commercial dataset:

Var1	Var2	Freq
	Tsn	Min. : 0

Var1	Var2	Freq
	Tsn	1st Qu.:160845
	Tsn	Median :167674
	Tsn	Mean :164501
	Tsn	3rd Qu.:169611
	Tsn	Max. :775091
	Tsn	NA's :98
	Year	Min. :1950
	Year	1st Qu.:1977
	Year	Median :1995
	Year	Mean :1991
	Year	3rd Qu.:2008
	Year	Max. :2017
	Year	NA
	State	West Florida :10405
	State	East Florida : 8973
	State	New York : 7106
	State	California : 6899
	State	North Carolina: 6436
	State	New Jersey : 5642
	State	(Other) :63642
	AFS.Name	FINFISH ** : 1467
	AFS.Name	OYSTER, EASTERN : 1187
	AFS.Name	SHARKS, UNCLASSIFIED **: 1169
	AFS.Name	BLUEFISH : 1103
	AFS.Name	SHAD, AMERICAN : 1083
	AFS.Name	SQUIDS ** : 1027
	AFS.Name	(Other) :102067
	Pounds	Min. : -321
	Pounds	1st Qu.: 3882
	Pounds	Median : 44779
	Pounds	Mean : 4871705
	Pounds	3rd Qu.: 483428
	Pounds	Max. :3410064761
	Pounds	NA's :11870
	Dollars	Min. : -4494
	Dollars	1st Qu.: 1739
	Dollars	Median : 21213
	Dollars	Mean : 1659774
	Dollars	3rd Qu.: 237607
	Dollars	Max. :540962350
	Dollars	NA's :12125
	category.orig	Finfish :82734
	category.orig	Other : 5683
	category.orig	Shellfish:20686
	category.orig	NA
	category.orig	NA
	category.orig	NA
	category.orig	NA

Edit/Restructure Data

```
temp00<-EditCommData(dat = landings.data, category0)
temp<-temp00[[1]]
```

	Q1_0010ALEWIFE.	Q1_0011ALFONSIN.	Q1_0014AMBERJACK...	Q1_0015AMBERJACK.GREATER.
1950	757043	NA	1955	NA
1951	765521	NA	2322	NA
1952	743937	NA	5299	NA
1953	757242	NA	3954	NA
1954	664708	NA	6601	NA

2.4.2 B. Enter base year

```
baseyr<-2010
pctmiss = 0.60
```

2.4.3 C. Run the function

```
temp00<-ImplicitQuantityOutput.q(temp, baseyr, pctmiss)
temp<-temp00[[1]]
warnings.list0<-temp00[[2]]
figures.list0<-temp00[[3]]
```

2.4.4 D. Obtain the implicit quantity estimates

	VE_Total	VV_Total	V_Total	QC_Total	QI_Total
1950	4910008722	5245971202	4908258998	0.0000000	0.0968994
1951	4468280396	4794434368	4465457584	0.0347238	0.1003232
1952	4454820075	4793838442	4452309258	-0.0843748	0.0922057
1953	4541451462	4888399889	4539800605	0.0532265	0.0972465
1954	4783727020	5140460448	4782314508	0.1183959	0.1094693
1955	4864502898	5204512660	4861925298	-0.0288226	0.1063592
1956	5315091181	5666629442	5310428386	0.1191829	0.1198217
1957	4812006928	5163232132	4806379376	-0.0871197	0.1098247
1958	4814125255	5142758622	4806065706	0.0335282	0.1135693
1959	5136276317	5465589023	5128247707	-0.0217100	0.1111303
1960	5014090964	5344020918	5006676562	-0.0955174	0.1010066
1961	5281445600	5609396384	5271996100	0.0969485	0.1112894
1962	5483719015	5828343984	5475910315	0.0764851	0.1201354
1963	4940953280	5287037049	4934601380	-0.0898713	0.1098096
1964	4658198362	5002256131	4649835362	0.0100066	0.1109140
1965	4883101538	5224600607	4876201638	0.1062257	0.1233444
1966	4406698497	4749537066	4401534097	-0.0039603	0.1228569
1967	4134834274	4477033943	4129052474	-0.1557475	0.1051379
1968	4374304496	4711291965	4367242096	0.0891058	0.1149363
1969	4451508990	4796818959	4445921890	0.1611264	0.1350311
1970	4937890925	5284503849	4934741680	0.1842227	0.1623456
1971	5174335253	5522323232	5172837053	0.0403260	0.1690262
1972	4986532651	5560262496	4985938142	0.0943772	0.1857554

	VE_Total	VV_Total	V_Total	QC_Total	QI_Total
1973	4999752596	5571956596	4998950386	0.4548172	0.2927292
1974	5142256144	5709021802	5139191827	-0.0003016	0.2926409
1975	5077858104	5647276505	5077572751	-0.0970783	0.2655672
1976	5585649671	6155200361	5585524133	0.2050485	0.3260062
1977	5371655443	5939447536	5371567038	0.0784171	0.3525997
1978	6158425762	6496185092	6157091268	0.1488658	0.4091980
1979	6468717520	6799570776	6468428160	0.1614218	0.4808816
1980	6558742947	6880048528	6558531303	0.0131135	0.4872292
1981	6021841354	6403961926	6020613522	-0.0266352	0.4744230
1982	6438791754	6821045734	6437693970	0.0672981	0.5074497
1983	6394874948	6653663183	6393134325	-0.0788122	0.4689918
1984	6400733217	6652340419	6399754108	0.0244638	0.4806066
1985	6327817404	6579565459	6327072795	-0.0373264	0.4629980
1986	6087762826	6247840847	6086540382	-0.0389759	0.4452994
1987	6970940970	7128195514	6969539624	0.2167491	0.5530763
1988	7345259874	7347764361	7344007016	0.1461676	0.6401250
1989	8703012869	8703931781	8699561960	0.0785886	0.6924611
1990	9763270615	9757431015	9758613232	0.1557746	0.8091843
1991	9591954950	9595085597	9589081821	-0.0045887	0.8054798
1992	9910321012	9906177543	9904573944	0.1764552	0.9609220
1993	9931205231	9926946663	9927354742	-0.1918693	0.7931592
1994	10054778439	10054278166	10050150258	0.0836528	0.8623634
1995	9630008417	9627817295	9624203652	0.0765093	0.9309319
1996	9340724555	9339575245	9335865765	-0.1173585	0.8278464
1997	9583115360	9580085917	9578044682	0.0645235	0.8830229
1998	8960217365	8957045167	8955423113	-0.2545416	0.6845828
1999	9062894698	9065736484	9058673959	0.0121246	0.6929336
2000	8833974470	8835607515	8828248850	0.0179899	0.7055121
2001	9253069890	9247643978	9241066331	0.0191377	0.7191441
2002	9208948179	9198735375	9192749037	-0.1498875	0.6190427
2003	9291777375	9282063460	9277597879	0.0152135	0.6285325
2004	9182300008	9232276392	9105839980	0.1991412	0.7670323
2005	9109639858	9129251203	9002386803	0.1002990	0.8479553
2006	9138912762	9009782297	8895011089	0.0760479	0.9149558
2007	8885395234	8790596336	8658691318	-0.0027861	0.9124101
2008	7882138532	7653096133	7525704203	0.0992319	1.0075949
2009	7788358546	7563486402	7438878112	-0.1664499	0.8530955
2010	8046768286	7763910479	7637061300	0.1588837	1.0000000
2011	9426480100	9190353064	9063446042	0.2447498	1.2773018
2012	9316801850	9089997956	8965918468	0.0145955	1.2960813
2013	9387237694	9211378665	9087013015	0.0442077	1.3546635
2014	9101596982	8907330761	8783287468	-0.1037322	1.2211840
2015	9284445654	9169488957	9048004259	0.0056710	1.2281289
2016	9311345804	9156939966	9032212602	0.0365184	1.2738072
2017	9565233796	9407628665	9286207791	-0.0895266	1.1647233

Did all of the analyses work as intended?

list(NameBasecategory = c(1, 1, *1, 1), col = c(NA, NA), slope = c(NA, NA,*

NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA), intercept = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA), R2 = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), R2adj = c(NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,

NA, NA), Pr = c(NA, NA), Fstat = c(NA, NA,

NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA), var = c(“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”), slopecheck = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,

NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA)), list(var =
character(0), Freq = integer(0)), list(NameBasecategory = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1), col = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA),
slope = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,

*NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), intercept = c(NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), R2 = c(NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,*

NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA), R2adj = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), Pr =
c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,

NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), Fstat = c(NA, NA), var = c(“V”, “Q”, “Q”), slopecheck = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,

NA, NA)), list(var = character(0), Freq = integer(0)), list(NameBasecategory = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1), col = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), slope = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), intercept = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), R2 = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), R2adj = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), Pr = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), Fstat = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), var = c("V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q", "Q"), slopecheck = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA)), list(NameBasecategory = c(1, 1, 1, 1), col = c(NA, NA), slope = c(NA, NA), intercept = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,

NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA), R2 = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), R2adj = c(NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,

NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), Pr = c(NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA), Fstat = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,

NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA), var = c(“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”), slopecheck = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,

NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA)), list(var =
character(0), Freq = integer(0)), list(NameBasecategory = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1), col = c(NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), slope = c(NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA), intercept = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA), R2 = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), R2adj = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), Pr = c(NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), Fstat = c(NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA),
var = c(“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”,
“Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”), slopecheck = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA)), list(NameBasecategory = c(1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1), col = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), slope = c(NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA), intercept = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA), R2 = c(NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,

NA, NA, NA, NA), R2adj = c(NA, NA), Pr = c(NA, NA), Fstat = c(NA, NA), var = c(“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”, “Q”), slopecheck = c(NA, NA)), Warning: When back calculated, ln(Q_t/Q_{t-1}) = did not equal sum((R_{i, t} - R_{i, t-1}) / 2) x ln((Q_{i,t}) / (Q_{i,t-1})), FYI: 32 of species V columns are completely empty, 34 of species Q columns are completely empty.

2.4.5 E. Graph

2.4.5.1 Graph 1: Price Index

For comparison, let’s recreate those graphs to make sure we are getting the same output:

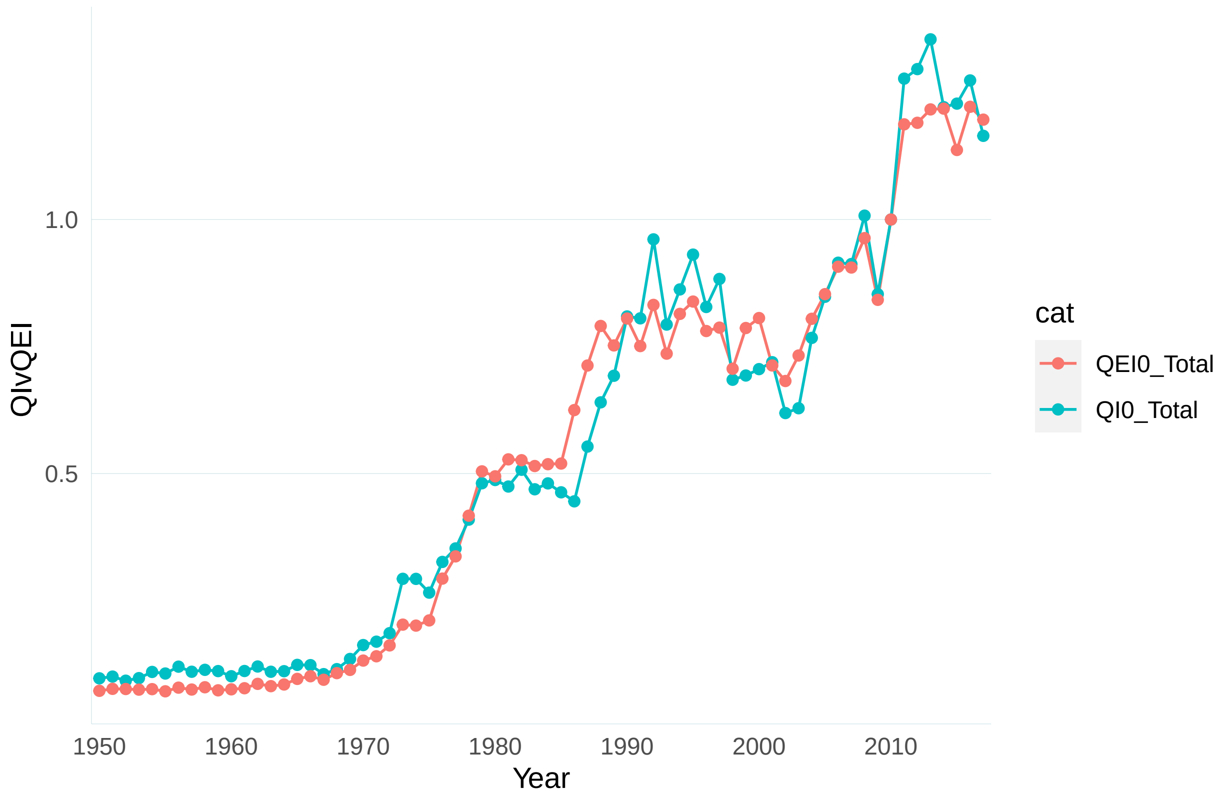
```
figures.list0$`_PI-Line`
```

```
## NULL
```

2.4.5.2 Graph 2: Quantity Index Compare

For comparison, let’s recreate those graphs to make sure we are getting the same output:

```
figures.list0$`_QIvQEI-Line`
```



2.4.5.3 Graph 3: Quantity Compare

```
figures.list0$`_QvQE-Line`
```

```
## NULL
```