Productivity Index - Output - By Price

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 Price 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
- P are prices
- PC are price changes
- PI are price indicies, often defined by a price from a base year *baseyr*
- *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)
```

	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!

We can do the structuring work in a function

This funciton standardizes the length of the category or species numbers e.g.,(numbers of 33, 440, and 1 are converted to 033, 440, and 001)

```
print(numbers0)
```

```
## function(x) {
##   xx<-rep_len(x = NA, length.out = length(x))
##   for (i in 1:length(x)){
##     xx[i]<-paste0(paste(rep_len(x = 0,
##                                 length.out = nchar(max(x))-nchar(x[i])),
##                         collapse = ""),
##                   as.character(x[i]))
##   }
##   return(xx)
## }
## <bytecode: 0x000000001fe74628>
```

2.0.3 Lets get started

```
category<- c(1,2)
ii<-1 #The category iteration value
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:

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

```
  if (length(VColumns0) == 0) {

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

  } else {
  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

	QE1_0Finfish	VE1_0Finfish	Q1_0Finfish	V1_0Finfish
2013	3011	4080	3011	3080
2014	3211	4270	3211	3370
2015	3410	4700	3410	3700
2016	3315	4480	3315	3380

2.0.6 Price for each species ($P_{t,i,s}$; e.g., Salmon and Flounder)

We first measure output price for each species in each of the categories (e.g., Finfish & Others and Shellfish) using detailed landings time series data on value ($) and pounds (lbs).

Price for a species (s) of category (i) in year (t) =

$$P_{t,i,s} = V_{t,i,s}/Q_{t,i,s}$$

where:

- $P_{t,i,s}$ is the price per individual species (s), category (i), for each year (t)
- $Q_{t,i,s}$ is the quantity (lb) per individual species (s), category (i), for each year (t)
- $V_{t,i}$ is the value ($) per category (i), for each year (t)

Here we calculate the price for each species

```
# Find which columns in this table are price Columns - we will need this for later
PColumns<-paste0("P", substr(x = VColumns,#names(temp)[VColumns],
                             start = 2,
                             stop = nchar(VColumns)))#nchar(names(temp)[VColumns])))

#####Price for each species
tempP<-data.frame(data = rep_len(x = NA, length.out = nrow(temp)))
for (c0 in 1:length(VColumns)) {

  NameBase<-substr(start = 2,
                   stop = nchar(VColumns[c0]),
                   x = VColumns[c0])

  Q0<-temp[,names(temp) %in% paste0("Q", NameBase)]
  V0<-temp[,names(temp) %in% paste0("V", NameBase)] #to make sure its the same column
  tempP[,c0]<-V0/Q0
  names(tempP)[c0]<-paste0("P", NameBase ) #name the column
}

tempP<-as.matrix(tempP)
tempP[tempP %in% Inf]<-NA
tempP<-data.frame(tempP)
temp<-cbind.data.frame(temp, data.frame(tempP))
```

	P1_1Salmon	P1_2Cod	P1_4SeaBass
1	NA	1.400000	NA
2	NA	1.421053	0.1000000
3	NA	1.450000	0.1222222
4	5.00000	1.200000	NA

	P1_1Salmon	P1_2Cod	P1_4SeaBass
5	10.00000	1.291667	NA
6	12.50000	1.260870	NA
7	16.36364	1.400000	0.1000000
8	15.45455	1.391304	NA
9	20.00000	1.458333	NA
10	12.00000	1.454546	NA

There may be instances where price cannot (or should not) be calculated because there is no or too few Q or V data for that species in a year or ever. The next goal will be to calculate the price 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:

2.0.6.1 1. If there are instances for a species where there are too few pairs of V and/or Q are completely missing from the timeseries or where a percent of V 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 $V_{t,i,s}$, don't calculate that species $P_{t,i,s}$

```
#Find which columns in this table are price Columns
cc<-c() #Empty
for (c0 in 1:length(VColumns)) {

  #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 "PColumns
  if (sum(temp[,names(temp) %in% PColumns[c0]] %in% c(0, NA, NaN))/nrow(temp) > pctmiss |
      sum(temp[,names(temp) %in% VColumns[c0]] %in% c(0, NA, NaN))/nrow(temp) > pctmiss) {
    cc<-c(cc, c0)#Collect offending columns
  }
}

if (length(cc)>0){
  PColumns<-PColumns[-cc]
  # VColumns<-VColumns[-cc]
}
```

	P1_1Salmon	P1_2Cod
2007	NA	1.400000
2008	NA	1.421053
2009	NA	1.450000
2010	5.00000	1.200000
2011	10.00000	1.291667
2012	12.50000	1.260870
2013	16.36364	1.400000
2014	15.45455	1.391304
2015	20.00000	1.458333
2016	12.00000	1.454546

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

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

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 = PColumns, temp)
```

	P1_1Salmon	P1_2Cod
2007	5.00000	1.400000
2008	NA	1.421053
2009	NA	1.450000
2010	5.00000	1.200000
2011	10.00000	1.291667
2012	12.50000	1.260870
2013	16.36364	1.400000
2014	15.45455	1.391304
2015	20.00000	1.458333
2016	12.00000	1.454546

2.0.6.3 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:

```
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>
```

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

	P1_1Salmon	P1_2Cod
2007	5.00000	1.400000
2008	5.00000	1.421053
2009	5.00000	1.450000
2010	5.00000	1.200000
2011	10.00000	1.291667
2012	12.50000	1.260870
2013	16.36364	1.400000
2014	15.45455	1.391304
2015	20.00000	1.458333
2016	12.00000	1.454546

```
VVColumns<-paste0("V", substr(x = PColumns, start = 2, stop = nchar(PColumns)))
temp<-ReplaceFirst(colnames = VVColumns, temp)
```

2.0.7 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 $V_{t,i,s}$.

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

2.0.7.1 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.

```
VVColumns<-paste0("V", substr(x = PColumns, start = 2, stop = nchar(PColumns)))
temp<-ReplaceFirst(colnames = VVColumns, temp)
```

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

2.0.7.2 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
2007	100	2800
2008	100	2700
2009	100	2900
2010	100	3000
2011	100	3100
2012	150	2900
2013	180	2800
2014	170	3200
2015	200	3500
2016	180	3200

2.0.7.3 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 P, 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	P	NA
2	1_0Finfish	NA	NA	NA	NA	NA	NA	NA	P	NA
3	1_0Finfish	NA	NA	NA	NA	NA	NA	NA	V	NA
4	1_0Finfish	NA	NA	NA	NA	NA	NA	NA	V	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

2.0.8 Value of species $VV_{t,i}$ where P was able to be calculated

$R_{t,i}$, 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 $P_{t,i}$ (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 $P_{t,i}$
- $V_{t,i,s,available}$ are the $V_{t,i,s}$ where P 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	VV1_0Finfish
2007	100	2800	2900
2008	100	2700	2800
2009	100	2900	3000

	V1_1Salmon	V1_2Cod	VV1_0Finfish
2010	100	3000	3100
2011	100	3100	3200
2012	150	2900	3050
2013	180	2800	2980
2014	170	3200	3370
2015	200	3500	3700
2016	180	3200	3380

2.0.9 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 (c0 in 1:length(QColumns)) {

  #for renaming the columns
  NameBase<-substr(start = 2,
                   stop = nchar(QColumns[c0]),
                   x = QColumns[c0])

  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[,c0]<-V0/VV
  names(tempR)[c0]<-paste0("R", NameBase ) #name the column
}

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

	R1_1Salmon	R1_2Cod	R1_4SeaBass
1	0.0344828	0.9655172	NA
2	0.0357143	0.9642857	0.0428571
3	0.0333333	0.9666667	0.0366667
4	0.0322581	0.9677419	0.0290323
5	0.0312500	0.9687500	0.0250000
6	0.0491803	0.9508197	0.0327869
7	0.0604027	0.9395973	0.0335570
8	0.0504451	0.9495549	NA
9	0.0540541	0.9459459	NA
10	0.0532544	0.9467456	NA

2.0.9.1 Analysis Warnings Checks

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

	x
1	1.000000
2	1.042857
3	1.036667
4	1.029032
5	1.025000
6	1.032787
7	1.033557
8	1.000000
9	1.000000
10	1.000000

Is there a warning?

Rows of R_{t,i,s} for 1_0Finfish did not sum to 1

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

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

Where:

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

Such that:

- category's (i) Price 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) = $ln(\frac{P_{t,i,s}}{P_{s,t-1,i}} = [ln(P_{t,i,s}) - ln(P_{s,t-1,i})]$

We use this *PriceChange* function:

```
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>
```

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

```
  # Calculate
  P0<-temp[, names(temp) %in% paste0("P", NameBase)]
  R0<-temp[, names(temp) %in% paste0("R", NameBase)] #to make sure its the same column
  tempPCW[,c]<-PriceChange(R0, P0)
  names(tempPCW)[c]<-paste0("PCW", NameBase ) #name the column
}

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

2.0.11 Price Changes for the category ($PC_{t,i}$; e.g., Finfish)

$$PC_{t,i} = ln(\frac{P_{t,i}}{P_{t-1,i}}) = \sum_{s=1}^{n}(PCW_{t,i,s})$$

Where:

- $PC_{t,i}$ = Price change for a category (i)

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

	PCW1_1Salmon	PCW1_2Cod	PC1_0Finfish
1	0.0000000	0.0000000	0.0000000
2	0.0000000	0.0144018	0.0144018
3	0.0000000	0.0194695	0.0194695
4	0.0000000	-0.1830357	-0.1830357
5	0.0220102	0.0712743	0.0932846
6	0.0089738	-0.0231613	-0.0141875
7	0.0147572	0.0989356	0.1136927
8	-0.0031679	-0.0058852	-0.0090532
9	0.0134715	0.0445941	0.0580655
10	-0.0274080	-0.0024612	-0.0298692

2.0.12 Price Index for the each category (PI_t)

We calculate the price index first by comparing by multiplying the previous years PI_{t-1} by that year's price change PC_t, where the PI of the first year $PI_{t=firstyear} = 1$

$$PI_{t,i} = PI_{t-1,i} * \exp(ln(\frac{PC_{t,i}}{PC_{t-1,i}})) = PI_{t-1,i} * \exp(PC_{t,i})$$

Where

$$PI_{i,t=firstyear} = 1$$

```
#Note that the first row of this column is = 1
tempPI_yr1<-data.frame(c(1, rep_len(x = NA, length.out = nrow(temp)-1)))
rownames(tempPI_yr1)<-rownames(temp)

PC0<-temp[,names(temp) %in% paste0("PC", NameBasecategory)] #this is equal to ln(P_it/P_it-1)
```

```
  # Calculate
for (t in 2:nrow(tempPI_yr1)){  #Since the first row is defined, we need to start at the second row
    tempPI_yr1[t,1]<-tempPI_yr1[t-1,1]*exp(PC0[t])
}
```

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

$$PI_t = PI_t / PI_{t=baseyear}$$

In this example, our base year is 2010. Notice that the $PI_{t,i=baseyr} = 1$

```
tempPI_yrb<-tempPI_yr1[,1]/tempPI_yr1[rownames(tempPI_yr1) %in% baseyr,1]
```

In the future, we will use this *PriceIndex* function to do this step:

```
print(PriceIndex)
```

```
## function(temp, BaseColName, baseyr, var = "PC") {
##   ###Price Index for the entire commercial fishery ($PI_t$)
##
##   # We calculate the price index first by comparing by multiplying the previous years $PI_{t-1}$ by
##   # $$PI_t = PI_{t-1}*exp(ln(\frac{P_{i,t}}{P_{i,t-1}})) = PI_{t-1}*exp(PC_{t})$$
##   # Where
##   # $$PI_{i, t_{first year}} = 1$$
##
##   #Note that the first row of this column is = 1
##   tempPI_yr1<-c(1, rep_len(x = NA, length.out = nrow(temp)-1))
##
##   PC0<-temp[,names(temp) %in% paste0(var, BaseColName)] #this is equal to ln(P_it/P_it-1)
##
##   # Calculate
##   for (t in 2:length(tempPI_yr1)){  #Since the first row is defined, we need to start at the second
##     tempPI_yr1[t]<-tempPI_yr1[t-1]*exp(PC0[t])
##   }
##
##   tempPI_yr1<-data.frame(tempPI_yr1)
##   rownames(tempPI_yr1)<-rownames(temp)
##
##   # Then, to change the price (calulated later) into base year dollars, we use the following equatio
##   # $$PI_{t} = PI_{t}/PI_{t = baseyear}$$
##   # In this example, we'll decide that the base year is `r baseyr`, for whatever reason. Notice that
##
##   tempPI_yrb<-tempPI_yr1/tempPI_yr1[rownames(tempPI_yr1) %in% baseyr,]
##
##   tempPI<-data.frame(tempPI_yrb)
##   names(tempPI)<-paste0(substr(x = var, start = 1, stop = 1), "I", BaseColName)
##
##   return(tempPI)
## }
## <bytecode: 0x000000001ef7d750>
```

And we add the PI to the data

```
  tempPI<-PriceIndex(temp, BaseColName = NameBasecategory, baseyr, var = "PC")
  temp[ncol(temp)+1]<-(tempPI)
```

```
  names(temp)[ncol(temp)]<-paste0("PI", NameBasecategory)
```

	tempPI_yr1	tempPI_yrb	PI1_0Finfish
2007	1.0000000	1.160864	1.160864
2008	1.0145060	1.177703	1.177703
2009	1.0344514	1.200857	1.200857
2010	0.8614275	1.000000	1.000000
2011	0.9456527	1.097774	1.097774
2012	0.9323310	1.082309	1.082309
2013	1.0445909	1.212628	1.212628
2014	1.0351767	1.201699	1.201699
2015	1.0970642	1.273542	1.273542
2016	1.0647803	1.236065	1.236065

2.0.13 Implicit Quantity/Output for each category ($Q_{t,i}$; Finfish & others and Shellfish)

Note here that all columns of V are being used, despite having been removed earlier in the analysis when PI could not be calculated and PI columns have functionally been removed from the analysis.

$$Q_{t,i} = V_{t,i}/PI_{t,i}$$

```
temp[,paste0("Q", NameBasecategory)]<-NULL

  temp[,ncol(temp)+1]<-temp[,names(temp) %in% paste0("V", NameBasecategory)]/
    temp[,names(temp) %in% paste0("PI", NameBasecategory)]

  names(temp)[ncol(temp)]<-paste0("Q", NameBasecategory)
```

	x
1	2411.997
2	2394.491
3	2506.543
4	3190.000
5	2987.864
6	2910.443
7	2539.938
8	2804.362
9	2905.283
10	2734.484

2.0.14 Implicit Quantity Index for each category ($QI_{t,i}$; Finfish, Others, and Shellfish)

$$QI_{t,i} = Q_{t,i}/Q_{t=baseyr,i}$$

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

  names(temp)[ncol(temp)]<-paste0("QI", NameBasecategory)
```

	x
1	0.7561119
2	0.7506241
3	0.7857501
4	1.0000000
5	0.9366346
6	0.9123647
7	0.7962190
8	0.8791104
9	0.9107469
10	0.8572051

2.0.15 Analysis Warnings Checks

2.0.15.1 1. When back calculated, V_t should equal $PI_t * Q_t$

$$V_{t,i} = PI_{t,i} * Q_{t,i}$$

```
temp0<-temp[names(temp) %in% c(paste0("Q",NameBasecategory),
                               paste0("PI",NameBasecategory),
                               paste0("V",NameBasecategory))]

temp0[,(ncol(temp0)+1)]<-temp0[,paste0("Q",NameBasecategory)]*
  temp0[,paste0("PI",NameBasecategory)]
names(temp0)[ncol(temp0)]<-paste0("V", NameBasecategory, "_Check")
```

	V1_0Finfish	PI1_0Finfish	Q1_0Finfish	V1_0Finfish_Check
2007	2800	1.160864	2411.997	2800
2008	2820	1.177703	2394.491	2820
2009	3010	1.200857	2506.543	3010
2010	3190	1.000000	3190.000	3190
2011	3280	1.097774	2987.864	3280
2012	3150	1.082309	2910.443	3150
2013	3080	1.212628	2539.938	3080
2014	3370	1.201699	2804.362	3370
2015	3700	1.273542	2905.283	3700
2016	3380	1.236065	2734.484	3380

Is there a warning?

No warning.

2.0.15.2 2. When back calculated, $Q_{t,i}$ should equal $V_{t,i}/PI_{t,i}$

$$Q_{t,i} = V_{t,i}/PI_{t,i}$$

```
temp0[,(ncol(temp0)+1)]<-temp0[,paste0("V",NameBasecategory)]/temp0[,paste0("PI",NameBasecategory)]
names(temp0)[ncol(temp0)]<-paste0("Q", NameBasecategory, "_Check")
```

	V1_0Finfish	PI1_0Finfish	Q1_0Finfish	V1_0Finfish_Check	Q1_0Finfish_Check
2007	2800	1.160864	2411.997	2800	2411.997
2008	2820	1.177703	2394.491	2820	2394.491
2009	3010	1.200857	2506.543	3010	2506.543
2010	3190	1.000000	3190.000	3190	3190.000
2011	3280	1.097774	2987.864	3280	2987.864
2012	3150	1.082309	2910.443	3150	2910.443
2013	3080	1.212628	2539.938	3080	2539.938
2014	3370	1.201699	2804.362	3370	2804.362
2015	3700	1.273542	2905.283	3700	2905.283
2016	3380	1.236065	2734.484	3380	2734.484

Is there a warning?

No warning.

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.p* function to calculate the Implicit Quanity Output at Species and category Level:

```
print(ImplicitQuantityOutput.speciescat.p)
```

```
## 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"))], decr
##                         start = 3, stop = nchar(sort(names(temp)[grep(x = names(temp),
##                                                                       pattern = paste0(NumberOfSpeci
##                                                    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 related V and Q data where V column has less data than the $pctmiss$
##   # VColumns0<-VColumns
##   # QColumns0<-QColumns<-which(names(temp) %in%
##   #                              paste0("Q", substr(x = names(temp)[VColumns],
##   #                                                 start = 2,
##   #                                                 stop = nchar(names(temp)[VColumns]))))
##
##   ###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
##     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) {
##     VColumns<-names(temp)[VColumns0]
##     QColumns<-names(temp)[QColumns0]
##   }
##
##   # if (length(VColumns) == 0) {
##   #
##   #   warnings.list[length(warnings.list)+1]<-paste0("FYI: ", NameBasecategory, " is no longer being
##   #
##   # } else {
##
##
##
##
##   ###Caluclate Catagory Sums of $V$ and $Q$
##
##   # Because we removed some columns for not meeting a perecent missing threshold and those columns wi
##
##   ####
##   #if there are still columns to assess that haven't been "removed"
##   PColumns<-c()
##   if (length(VColumns0) == 0) {
##     warnings.list[length(warnings.list)+1]<-paste0("FYI: ", NameBasecategory, " is no longer being ca
```

```
##
##   } else {
##
##     # Because we removed some columns for not meeting a perecent missing threshold of `r pctmiss`% ar
##
##     # 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)
##
##
##
##   # #Caulculate the summed quantity
##   # temp[,(ncol(temp)+1)]<-rowSums(temp[, grep(pattern = paste0("Q", strsplit(x = NameBasecategory, s
##   #                                    x = names(temp))], na.rm = T)
##   # names(temp)[ncol(temp)]<-paste0("QE", NameBasecategory)
##
##
##   # Find which columns in this table are price Columns - we will need this for later
##   PColumns<-paste0("P", substr(x = VColumns,#names(temp)[VColumns],
##                                start = 2,
##                                stop = nchar(VColumns)))#nchar(names(temp)[VColumns])))
##
##   #####Price for each species
##   tempP<-data.frame(data = rep_len(x = NA, length.out = nrow(temp)))
##   for (c0 in 1:length(VColumns)) {
##
##     NameBase<-substr(start = 2,
##                      stop = nchar(VColumns[c0]),
##                      x = VColumns[c0])
##
##     Q0<-temp[,names(temp) %in% paste0("Q", NameBase)]
##     V0<-temp[,names(temp) %in% paste0("V", NameBase)] #to make sure its the same column
##     tempP[,c0]<-V0/Q0
##     names(tempP)[c0]<-paste0("P", NameBase ) #name the column
##   }
##
##   tempP<-as.matrix(tempP)
##   tempP[tempP %in% Inf]<-NA
##   tempP<-data.frame(tempP)
##   temp<-cbind.data.frame(temp, data.frame(tempP))
##
##   # There may be instances where price cannot be calculated because there is no Q or V data for that
```

```
##
##   #### 2.3. V and/or Q are completely missing from the timeseries. In this case, we will remove the c
##
##   #Find which columns in this table are price Columns
##   cc<-c() #Empty
##   for (c0 in 1:length(VColumns)) {
##
##     #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 "PColum
##     if (sum(temp[,names(temp) %in% PColumns[c0]] %in% c(0, NA, NaN))/nrow(temp) > pctmiss |
##         sum(temp[,names(temp) %in% VColumns[c0]] %in% c(0, NA, NaN))/nrow(temp) > pctmiss) {
##       cc<-c(cc, c0)#Collect offending columns
##     }
##   }
##
##   if (length(cc)>0){
##     PColumns<-PColumns[-cc]
##     # VColumns<-VColumns[-cc]
##   }
##
##   # }
##
##   if (length(PColumns) < 2) {
##
##     warnings.list[length(warnings.list)+1]<-paste0("FYI: ", NameBasecategory, " is no longer being ca
##
##   } else {
##
##
##   # 2.1. If the first value of P is 0 in a timeseries, we let the next available non-zero value of P
##     temp<-ReplaceFirst(colnames = PColumns, temp)
##
##   # 2.2. If there is a value in the middle of P's timeseries that is 0, we let the most recent past a
##
##     #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
##     temp<-ReplaceMid(colnames = PColumns, temp)
##
##   ###Fill in values of $V_{i,t,s}$ 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 be
##   # $$where \begin{cases} if: V_{i,t=1} = 0, then: V_{i,t=1} = V_{i,t=1+1...} \\ if: V_{i,t\neq1} = 0
##
##   #### 1. If the first value of $V_{i,t,s}$ is 0/NA in a timeseries, we let the next available non-ze
##   VVColumns<-paste0("V", substr(x = PColumns, start = 2, stop = nchar(PColumns)))
##   temp<-ReplaceFirst(colnames = VVColumns, temp)
##
##   #### 2. If there is a value in the middle of $V_{i,t,s}$'s timeseries that is 0/NA, we let the most
##   temp<-ReplaceMid(colnames = VVColumns, temp)
##
##
##
##
##   ###Value of species where P was able to be calculated
```

```
##   # $R_{i,t}$, defined and discussed in the subsequent step, will need to sum to 1 across all species
##   # $$VV_{i,t} = \sum_{s=1}^{n}(V_{s,i,t, available})$$
##   #   where:
##   #   - $VV_{i,t}$ is the new total of $V_{i,t}$ (called $VV_{i,t}$) for the category using only valu
##   #   - $V_{s,i,t, available}$ are the $V_{s,i,t}$ where P were able to be calculated
##
##   # VVColumns<-paste0("V", substr(x = PColumns, start = 2, stop = nchar(PColumns)))
##   #
##   # temp[ncol(temp)+1]<-rowSums(data.frame(temp[,names(temp) %in% VVColumns]), na.rm = T)
##   # names(temp)[ncol(temp)]<-paste0("VV",NameBasecategory)
##
##   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)]
##
##
##   ####Analysis Warnings Checks
##
##   # Just so we can get a sense of the data, we want to see how many species are significantly increas
##   # We'll use the below function to collect our info:
##
##   Columns<-c(PColumns,
##              paste0("V", substr(x = PColumns, start = 2, stop = nchar(PColumns))),
##              paste0("Q", substr(x = PColumns, start = 2, stop = nchar(PColumns))))
##
##   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_check
##
##   # 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$var),]
##
##   warnings.list[length(warnings.list)+1]<-list(lm_sig_slope)
##   names(warnings.list)[[length(warnings.list)]]<-paste0("FYI ", NameBasecategory, " species lm_sig_sl
##
##
##
##
##   ###Revenue Share for each species ($R_{s,i,t}$; e.g., Salmon and Flounder)
##
##   # $$R_{s,i,t} = V_{s,i,t}/VV_{i,t}$$
##   #   where:
##   #   - $R_{s,i,t}$ is the revenue share per individual species (s), category (i), for each year (t)
##   #   - $V_{s,i,t}$ is the value ($) per individual species (s), category (i), for each year (t)
##
##   tempR<-data.frame(data = rep_len(x = NA, length.out = nrow(temp)))
##   for (c0 in 1:length(QColumns)) {
##
##     #for renaming the columns
```

```
##     NameBase<-substr(start = 2,
##                      stop = nchar(QColumns[c0]),
##                      x = QColumns[c0])
##
##     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 colum
##     tempR[,c0]<-V0/VV
##     names(tempR)[c0]<-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," c
##   }
##
##   #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 Price Changes for each species ($PCW_{t,i,s}$; e.g., Salmon and Flounder
##
##   #    $$PCW_{t,i,s} = \frac{R_{t,i,s} + R_{s,t-1,i}}{2} * ln(\frac{P_{t,i,s}}{P_{s,t-1,i}}) = \frac
##   #      Where:
##   #      - $PCW_{t,i,s}$ = Revenue share-weighted quantity change for a species (s)
##   #    Such that:
##   #      - category's (i) Price 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{P_{t,i,s}}{Q_{s,t-1,i}} = [
##   #    We use this *PriceChange* function.
##
##   #Find which columns in this table are price and revenue share columns
##   tempPCW<-data.frame(data = rep_len(x = NA, length.out = nrow(temp)))
##   for (c in 1:length(PColumns)){
##     #For nameing columns
##     NameBase<-substr(start = 2,
##                      stop = nchar(PColumns[c]),
##                      x = PColumns[c])
##
##     # Calculate
##     P0<-temp[, names(temp) %in% paste0("P", NameBase)]
##     R0<-temp[, names(temp) %in% paste0("R", NameBase)] #to make sure its the same column
##     tempPCW[,c]<-PriceChange(R0, P0)
##     names(tempPCW)[c]<-paste0("PCW", NameBase ) #name the column
##   }
##
##   temp<-cbind.data.frame(temp, tempPCW)
##
##
```

```
##
##
##
##
##   ###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(tempPCW, na.rm = T)
##   names(temp)[ncol(temp)]<-paste0("PC", 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))]
##
##
##   #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))]
##
## ###6. Price Indexes for each category ($PI_{i,t}$; Finfish & others and Shellfish)
## # We calculate the price index first by comparing by multiplying the previous years $PI_{t-1}$ by tha
##   ###Price Index for the each category ($PI_t$)
##   #
##   # We calculate the price index first by comparing by multiplying the previous years $PI_{t-1}$ by t
##   # $$PI_t = PI_{t-1}*exp(ln(\frac{P_{i,t}}{P_{i,t-1}})) = PI_{t-1}*exp(PC_{t})$$
##   # Where
##   # $$PI_{i, t_{first year}} = 1$$
##   #Note that the first row of this column is = 1
##   #
##   # Then, to change the price (calulated later) into base year dollars, we use the following equatio
##     # $$PI_{t} = PI_{t}/PI_{t = baseyear}$$
##     # In this example, we'll decide that the base year is `r baseyr`, for whatever reason. Notice tha
##
##   tempPI<-PriceIndex(temp, BaseColName = NameBasecategory, baseyr, var = "PC")
##   temp[ncol(temp)+1]<-(tempPI)
##   names(temp)[ncol(temp)]<-paste0("PI", 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))]
##
##   ###7. Implicit Quantity/Output for each category ($Q_{i,t}$; Finfish & others and Shellfish)
##   temp[,paste0("Q", NameBasecategory)]<-NULL
##
##   temp[,ncol(temp)+1]<-temp[,names(temp) %in% paste0("V", NameBasecategory)]/
##     temp[,names(temp) %in% paste0("PI", NameBasecategory)]
##
##   names(temp)[ncol(temp)]<-paste0("Q", NameBasecategory)
##
##
##   ###Implicit Quantity Index for each category ($QI_{t,i}$; Finfish, Others, and Shellfish)
##   # $$QI_{t,i}=Q_{t,i}/Q_{t=baseyr,i}$$
##
##   temp[,ncol(temp)+1]<-temp[,names(temp) %in% paste0("Q", NameBasecategory)]/
##     temp[rownames(temp) %in% baseyr, names(temp) %in% paste0("Q", NameBasecategory)]
##
##   names(temp)[ncol(temp)]<-paste0("QI", NameBasecategory)
##
##   ##############WARNINGS
##   ####1. When back calculated, $V_t$ did not equal $P_t * Q_{t}$
##   # $$V_i = P_t * Q_i$$
##
##   temp0<-temp[names(temp) %in% c(paste0("Q",NameBasecategory),
##                                  paste0("PI",NameBasecategory),
##                                  paste0("V",NameBasecategory))]
##
##   temp0[,(ncol(temp0)+1)]<-temp0[,paste0("Q",NameBasecategory)]*temp0[,paste0("PI",NameBasecategory)]
##   names(temp0)[ncol(temp0)]<-paste0("V", NameBasecategory, "_Check")
##
##   if ((length(setdiff(as.character(temp0[,paste0("V", NameBasecategory, "_Check")]),
##                       as.character(temp0[,paste0("V", NameBasecategory)])))) != 0)) {
##     warnings.list[length(warnings.list)+1]<-"Warning: When back calculated, V_{i,t} did not equal PI
##   }
##
##
##   ####2. When back calculated, $Q_{t}$ did not equal $V_t / P_{t}$
##   # $$Q_{i,t} = V_t / P_{i,t}$$
##
##   temp0[,(ncol(temp0)+1)]<-temp0[, paste0("V", NameBasecategory)]/temp0[, paste0("PI", NameBasecateg
##   names(temp0)[ncol(temp0)]<-paste0("Q", NameBasecategory, "_Check")
##
##  if (length(setdiff(as.character(temp0[,paste0("Q", NameBasecategory, "_Check")]),
##                                  as.character(temp0[,paste0("Q", NameBasecategory)])))) != 0) {
##     warnings.list[length(warnings.list)+1]<-"Warning: When back calculated, Q_{i,t} did not equal V_
##   }
##   }
## }
##   return(list(temp, warnings.list))
## }
```

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

tempS<-ImplicitQuantityOutput.speciescat.p(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)))]
```

What does the Shellfish data look like?

	PCW2_1Shrimp	PCW2_2Clam	PC2_0Shellfish	PI2_0Shellfish	Q2_0Shellfish	QI2_0Shellfish
2007	0.0000000	0.0000000	0.0000000	1.030337	1747.0009	2.495716
2008	0.0183493	0.0648402	0.0831894	1.119717	1964.7830	2.806833
2009	-0.0087575	-0.0805788	-0.0893363	1.024023	1757.7726	2.511104
2010	-0.0237392	0.0000000	-0.0237392	1.000000	700.0000	1.000000
2011	0.1730144	0.0000000	0.1730144	1.188883	757.0129	1.081447
2012	-0.0604413	0.0000000	-0.0604413	1.119154	893.5320	1.276474
2013	0.0977034	0.0488994	0.1466028	1.295862	1697.7119	2.425303
2014	-0.0998625	0.0614193	-0.0384432	1.246990	1603.8619	2.291231
2015	0.0553143	-0.0293044	0.0260098	1.279850	1562.6836	2.232405
2016	0.0393171	-0.0549436	-0.0156266	1.260005	1825.3889	2.607699

2.1.1 Value for all fisheries for species where P 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 $PI_{t,i}$.

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

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 $P_{t,i}$

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 price 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
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.

2.1.3 Revenue Share-Weighted Price Changes for each category ($PCW_{t,i}$; e.g., Salmon and Flounder)

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

Where:

- $PCW_{t,i}$ = Revenue share-weighted price change for a category (i)

Such that:

- Price Change for each category (i) = $\frac{R_{t,i}+R_{t-1,i}}{2}$
- Revenue Share for each category (i) = $ln(\frac{PI_{t,i}}{PI_{t-1,i}} = [ln(PI_{t,i}) - ln(PI_{t-1,i})]$

```
#Find which columns in this table are price and revenue share columns
 tempPCW<-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))


   R0 = temp[, names(temp) %in% paste0("R", NameBasecategory)]

   P0 = temp[, names(temp) %in% paste0("PI", NameBasecategory)]

   tempPCW[,i]<-PriceChange(R0, P0)

   names(tempPCW)[i]<-paste0("PCW", NameBasecategory)

 }

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

2.1.4 Price Changes for the entire fishery ($PC_{t,i}$; e.g., Finfish)

$$PC_t = ln(\frac{P_t}{P_{t-1}}) = \sum_{s=1}^{n}(PCW_{t,i})$$

Where:

- PC_t = Price change for the entire fishery

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

	PCW1_0Finfish	PCW2_0Shellfish	PC0_0Total
1	0.0000000	0.0000000	0.0000000
2	0.0084283	0.0345050	0.0429332
3	0.0115603	-0.0362914	-0.0247311
4	-0.1323193	-0.0065778	-0.1388971
5	0.0748488	0.0341928	0.1090416
6	-0.0109508	-0.0137889	-0.0247398
7	0.0763088	0.0482052	0.1245141
8	-0.0054812	-0.0151679	-0.0206491
9	0.0370656	0.0094067	0.0464723
10	-0.0185815	-0.0059053	-0.0244869

2.1.5 Price Index for the entire commercial fishery (PI_t)

We calculate the price index first by comparing by multiplying the previous years PI_{t-1} by that year's price change PC_t, where the PI of the first year $PI_{t=firstyear} = 1$

$$PI_t = PI_{t-1} * \exp(ln(\frac{P_{t,i}}{P_{t-1,i}})) = PI_{t-1} * \exp(PC_t)$$

Where

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

```
tempPI<-PriceIndex(temp, BaseColName = NameBaseTotal, baseyr, var = "PC")
temp[ncol(temp)+1]<-(tempPI)
names(temp)[ncol(temp)]<-paste0("PI", NameBaseTotal)
```

	PI0_0Total
2007	1.128281
2008	1.177776
2009	1.149006
2010	1.000000
2011	1.115209
2012	1.087957
2013	1.232218
2014	1.207035
2015	1.264452
2016	1.233866

2.1.6 Total Implicit Quantity/Output for the entire commercial fishery ($Q_t = Y_t$)

To get quantity estimates for total output using total value of landings divided by price index as follow: $Y = Q = V/I$

$$Q_t = V_t/PI_t$$

```
temp[,ncol(temp)+1]<-temp[,names(temp) %in% paste0("V", NameBaseTotal)]/
  temp[, names(temp) %in% paste0("PI", NameBaseTotal)]
names(temp)[ncol(temp)]<-paste0("Q", NameBaseTotal)
```

	x
1	4077.000
2	4262.269
3	4186.228
4	3890.000
5	3748.177
6	3814.488
7	4284.956
8	4448.919
9	4507.880
10	4603.418

2.1.7 Total Implicit Quantity/Output Index

$$QI_t = Q_t / Q_{t=baseyr}$$

Where:

- QI is the sum of Q after these equations

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

	x
1	1.0480719
2	1.0956991
3	1.0761510
4	1.0000000
5	0.9635417
6	0.9805883
7	1.1015311
8	1.1436810
9	1.1588382
10	1.1833979

2.1.8 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)]
temp[,ncol(temp)+1]<-temp[,QE]/temp[,QE][rownames(temp) %in% baseyr]
names(temp)[ncol(temp)]<-paste0("QEI", NameBaseTotal)
```

	x
1	1.2452107
2	1.2950192
3	1.2068966
4	1.0000000
5	0.9540230
6	0.9241379
7	1.2455939
8	1.3145594
9	1.3908046
10	1.3697318

2.1.9 Solve Output portion of the equation for the Output Changes:

$$QC_t = \sum_{i=1}^{n}((\frac{R_{it} + R_{it-1}}{2}) * ln(\frac{Q_{it}}{Q_{it-1}}))$$

```
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))
  R0 = temp[, names(temp) %in% paste0("R", NameBasecategory)]
  Q0 = temp[, names(temp) %in% paste0("Q", NameBasecategory)]
  tempQCW[,i]<-PriceChange(R0, Q0)
  names(tempQCW)[i]<-paste0("QCW", NameBasecategory)
}
temp<-cbind.data.frame(temp, tempQCW)

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

	Q0_0Total	QI0_0Total	QC0_0Total
2007	4077.000	1.0480719	0.0000000
2008	4262.269	1.0956991	0.0444654
2009	4186.228	1.0761510	-0.0180726
2010	3890.000	1.0000000	-0.0808110
2011	3748.177	0.9635417	-0.0370504
2012	3814.488	0.9805883	0.0175616
2013	4284.956	1.1015311	0.1196593
2014	4448.919	1.1436810	0.0375242
2015	4507.880	1.1588382	0.0131616
2016	4603.418	1.1833979	0.0210303

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 1. When back calculated, V_t should equal $PI_t * Q_t$?

$$V_t = P_t * Q_t$$

```
temp0<-temp[names(temp) %in% c(paste0("Q",NameBaseTotal),
                               paste0("PI",NameBaseTotal),
                                      paste0("V",NameBaseTotal))]

temp0[,(ncol(temp0)+1)]<-temp0[,paste0("Q",NameBaseTotal)]*temp0[,paste0("PI",NameBaseTotal)]
names(temp0)[ncol(temp0)]<-paste0("V", NameBaseTotal, "_Check")
```

	V0_0Total	PI0_0Total	Q0_0Total	V0_0Total_Check
2007	4600	1.128281	4077.000	4600
2008	5020	1.177776	4262.269	5020
2009	4810	1.149006	4186.228	4810
2010	3890	1.000000	3890.000	3890
2011	4180	1.115209	3748.177	4180
2012	4150	1.087957	3814.488	4150
2013	5280	1.232218	4284.956	5280
2014	5370	1.207035	4448.919	5370
2015	5700	1.264452	4507.880	5700
2016	5680	1.233866	4603.418	5680

Is there a warning?

```
  if (length(setdiff(as.character(temp0[,paste0("V", NameBaseTotal, "_Check")]),
                     as.character(temp0[,paste0("V", NameBaseTotal)])))) != 0) {
  warnings.list[length(warnings.list)+1] <- "Warning: When back calculated, V_t did not equal PI_t x Q_

    a<-warnings.list[length(warnings.list)][[1]]
} else {
    a<-"No warning."
}
```

No warning.

2.2.0.2 2. When back calculated, Q_t should V_t/PI_t?

$$Q_{t,i} = V_t/PI_{t,i}$$

```
  temp0<-temp[names(temp) %in% c(paste0("Q",NameBaseTotal),
                                 paste0("PI",NameBaseTotal),
                                        paste0("V",NameBaseTotal))]
  temp0[,(ncol(temp0)+1)]<-temp0[,paste0("V",NameBaseTotal)]/temp0[,paste0("PI",NameBaseTotal)]
  names(temp0)[ncol(temp0)]<-paste0("Q", NameBaseTotal, "_Check")
```

	V0_0Total	PI0_0Total	Q0_0Total	Q0_0Total_Check
2007	4600	1.128281	4077.000	4077.000
2008	5020	1.177776	4262.269	4262.269
2009	4810	1.149006	4186.228	4186.228
2010	3890	1.000000	3890.000	3890.000
2011	4180	1.115209	3748.177	3748.177
2012	4150	1.087957	3814.488	3814.488
2013	5280	1.232218	4284.956	4284.956
2014	5370	1.207035	4448.919	4448.919
2015	5700	1.264452	4507.880	4507.880
2016	5680	1.233866	4603.418	4603.418

Is there a warning?

```
  if (length(setdiff(as.character(temp0[,paste0("Q", NameBaseTotal, "_Check")]),
                     as.character(temp0[,paste0("Q", NameBaseTotal)])))) != 0) {
```

```
  warnings.list[length(warnings.list)+1]<-"Warning: When back calculated, Q_t did not equal V_t/PI_t"
    a<-warnings.list[length(warnings.list)][[1]]
} else {
    a<-"No warning."
}
```

No warning.

2.2.0.3 3. 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("Q",NameBaseTotal))
  for (i in 1:length(category)) {
    names0<-c(names0,
              names(temp)[grep(pattern = paste0("Q", 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("Q",NameBaseTotal)]/
                                      temp0[-1,paste0("Q",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("Q", 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"
```

	Q0_0Total	Q1_0Finfish	R1_0Finfish	Q2_0Shellfish	R2_0Shellfish	part1	part2
2007	4077.000	2411.997	0.6086957	1747.0009	0.3913043	NA	NA
2008	4262.269	2394.491	0.5617530	1964.7830	0.4382470	-0.0444404	0.0444654
2009	4186.228	2506.543	0.6257796	1757.7726	0.3742204	0.0180017	-0.0180726
2010	3890.000	3190.000	0.8200514	700.0000	0.1799486	0.0733908	-0.0808110
2011	3748.177	2987.864	0.7846890	757.0129	0.2153110	0.0371395	-0.0370504
2012	3814.488	2910.443	0.7590361	893.5320	0.2409639	-0.0175368	0.0175616
2013	4284.956	2539.938	0.5833333	1697.7119	0.4166667	-0.1163037	0.1196593
2014	4448.919	2804.362	0.6275605	1603.8619	0.3724395	-0.0375509	0.0375242
2015	4507.880	2905.283	0.6491228	1562.6836	0.3508772	-0.0131659	0.0131616
2016	4603.418	2734.484	0.5950704	1825.3889	0.4049296	-0.0209719	0.0210303

	Q0_0Total	Q1_0Finfish	R1_0Finfish	Q2_0Shellfish	R2_0Shellfish	part1	part2

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	PC0_0Total	PI0_0Total	Q0_0Total	QI0_0Total
2007	3250	5600	4700	4600	0.0000000	1.128281	4077.000	1.0480719
2008	3380	6220	5000	5020	0.0429332	1.177776	4262.269	1.0956991
2009	3150	5710	4800	4810	-0.0247311	1.149006	4186.228	1.0761510
2010	2610	3890	4700	3890	-0.1388971	1.000000	3890.000	1.0000000
2011	2490	4180	5000	4180	0.1090416	1.115209	3748.177	0.9635417
2012	2412	4150	4950	4150	-0.0247398	1.087957	3814.488	0.9805883
2013	3251	6280	5180	5280	0.1245141	1.232218	4284.956	1.1015311
2014	3431	6270	5370	5370	-0.0206491	1.207035	4448.919	1.1436810
2015	3630	6700	5700	5700	0.0464723	1.264452	4507.880	1.1588382
2016	3575	6780	5680	5680	-0.0244869	1.233866	4603.418	1.1833979

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.cas
  }
  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.cas
}
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
}
#Price
a<-temp
a<-a[,grep(pattern = "P[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.cas
}
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))
  }
  pp<-(aa[!(is.na(aa))])
} else {
  pp<-0
}
```

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

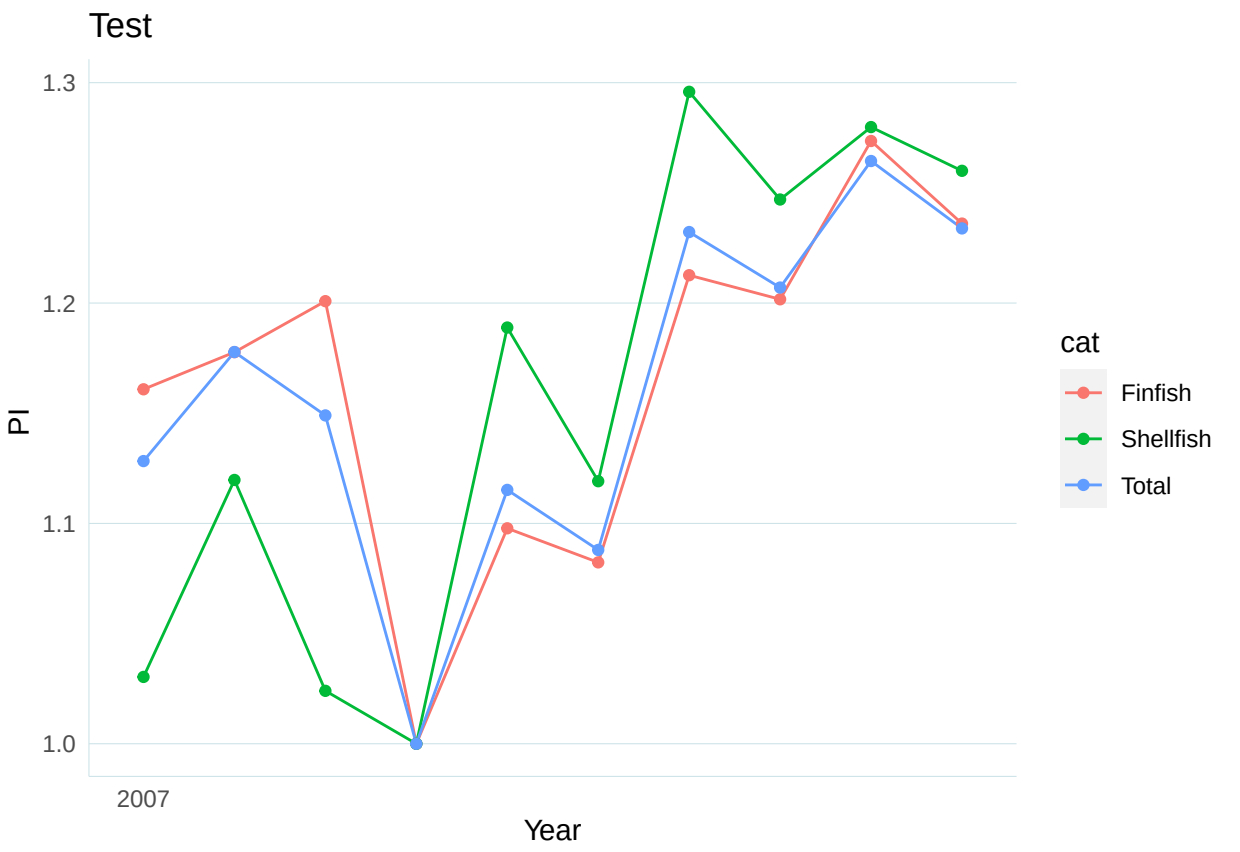
2.2.3 Graph 1: Price Index

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<- "_PI-Line"

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

a0$Year<-rownames(a0)

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

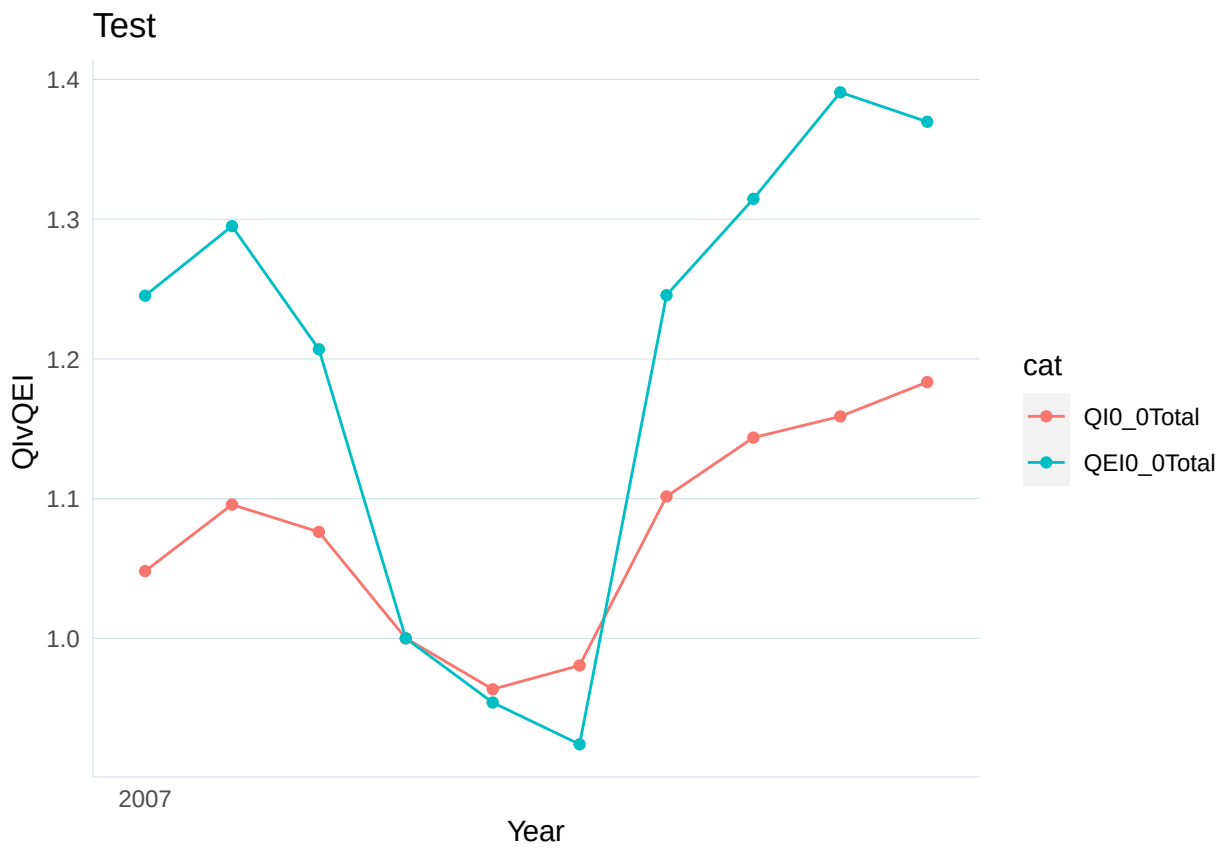
```
a$cat<-as.character(lapply(X = strsplit(x = as.character(a$Category), split = paste0("_", NumberOfSpe
```

```
temp0<-a

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



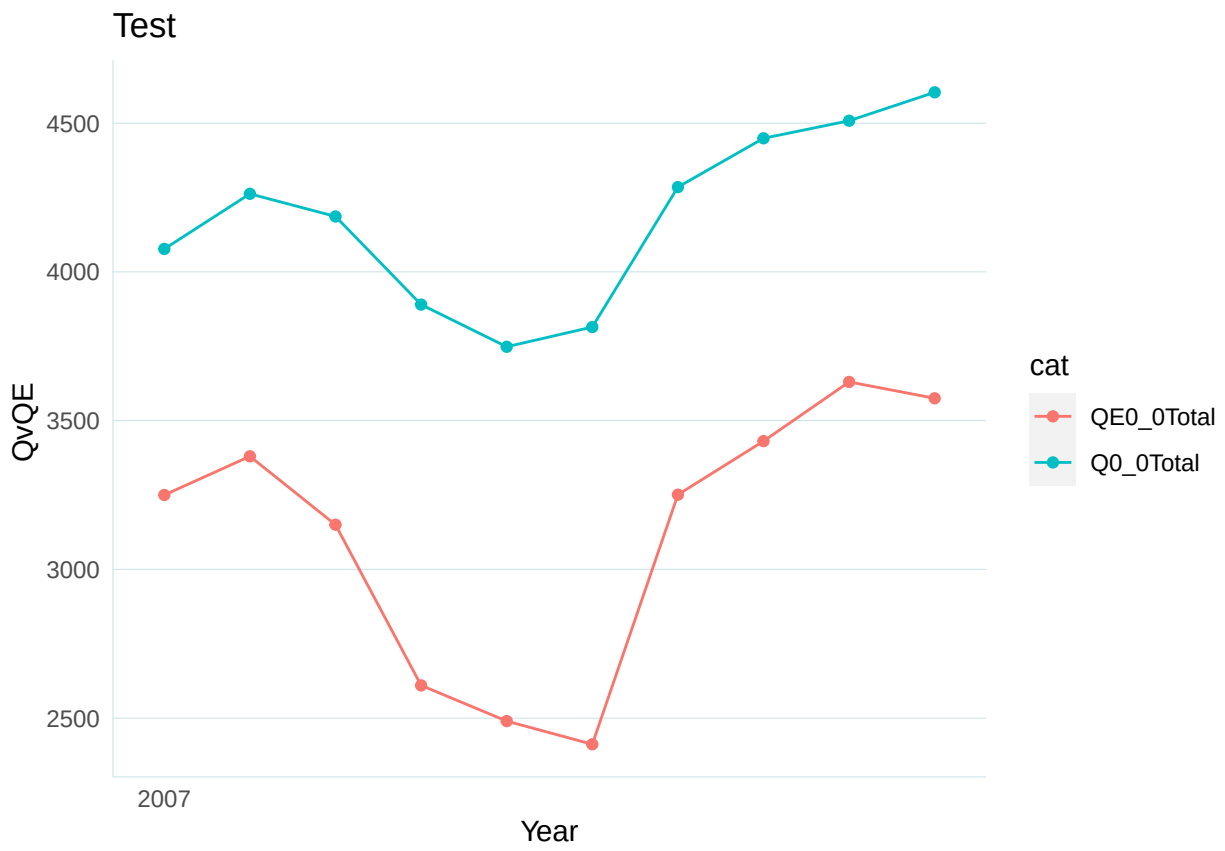
2.2.4 Graph 2: Quantity Index Compare

```
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.p* function to calculate the Implicit Quanity Output at Fishery Level:

```
print(ImplicitQuantityOutput.p)
```

```
## function(temp, baseyr, pctmiss = 1.00,
##                                 title0 = "", place = ""){
##
##   temp.orig<-temp
##
##   warnings.list<-list(title0)
##   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)))
##   tempPCW<-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.orig),
##     #                                      start = 1,
##     #                                      stop = (2+nchar(category[ii]))))
##
##     VColumns0<-VColumns<-grep(pattern = paste0("V", category[ii],"_"),
##                               x = substr(x = names(temp.orig),
##                                          start = 1,
##                                          stop = (2+nchar(category[ii]))))
##
##     NameBasecategory<-names(temp)[grepl(pattern = paste0("VE", category[ii],"_"),
##                                         x = substr(x = names(temp.orig),
##                                                    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"
##       ###Append species and category level calculations
##       temp00<-ImplicitQuantityOutput.speciescat.p(temp = temp.orig, ii=category[ii],
##                                                   baseyr, maxyr, minyr,
##                                                   pctmiss, warnings.list)
##
##         temp1<-temp00[[1]]
##       #If data for a catagory is no longer available after precentmissingthreshold etc, remove it fr
##       if (sum(names(temp1) %in% paste0("PI", NameBasecategory)) == 0) {
##         category.rm<-c(category.rm, ii)
##       } else {
##         #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<-temp00[[2]]
##       warnings.list1<-unique(warnings.list1)
##
##       warnings.list<-c(warnings.list, warnings.list1)
##
##     }
##   }
##
##   warnings.list<-unique(warnings.list)
##
##   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))]
##
##
##     ###8. Value for all fisheries for species where P was able to be calculated
##     # $R_{i,t}$, defined and discussed in the subsequent step, will need to sum to 1 across all spec
```

```
##     # $$VV_{t} = \sum_{s=1}^{n}(VV_{i,t})$$
##     # where:
##     # - $VV_{t}$ is the new total of $V_{i,t}$ for the entire fishery using only values for species
##
##   #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
##   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 entire commercial fishery ($R_t$)
##     # $$R_{i,t} = V_{i,t}/V_{t}$$
##     # where:
##     # - $R_{i,t}$ is the revenue share per individual species (s), category (i), for each year (t)
##     # - $V_{i,t}$ 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 spe
##
##   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," d
##   }
##
##
##
##
```

```
##
##   #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))]
##
##
##     ###8. Price Changes for the entire commercial fishery ($PC_t$)
##     # Measure output price changes ($PC_t$) for total output ($Q_t$) using $R_{i,t}$ and $P_{i,t}$ es
##
##     # $$PC_{t} = ln(\frac{P_{t}}{P_{t-1}}) =  \sum_{i=1}^n([\frac{R_{i,t} + R_{i,t-1}}{2}] * [ln(P_{
##
##   #Find which columns in this table are price and revenue share columns
##   tempPCW<-data.frame(data = rep_len(x = NA, length.out = nrow(temp)))
##   for (i in 1:length(category)) {
##     CatCol<-names(temp)[grep(pattern = paste0("VE", category[i], "_", NumberOfSpecies),
##                              x = substr(x = names(temp),
##                                         start = 1,
##                                         stop = nchar(paste0("VE", category[i], "_", NumberOfSpecies)
##
##     NameBasecategory<-substr(x = CatCol, start = 3, stop = nchar(CatCol))
##
##
##     R0 = temp[, names(temp) %in% paste0("R", NameBasecategory)]
##
##     P0 = temp[, names(temp) %in% paste0("PI", NameBasecategory)]
##
##     tempPCW[,i]<-PriceChange(R0, P0)
##
##     names(tempPCW)[i]<-paste0("PCW", NameBasecategory)
##
##   }
##
##   temp<-cbind.data.frame(temp, tempPCW)
##
##
##
##   ###Quantity Changes for the entire fishery ($QC_{t}$)
##   # $$PC_{t} = ln(\frac{PI_{t,i}}{PI_{t-1,i}}) = \sum_{s=1}^n(PCW_{t,i}) $$
##   #   Where:
##   #   - $PC_{t}$ = Quantity change for the entire fishery
##
##   temp[,ncol(temp)+1]<-rowSums(tempPCW, na.rm = T)
##   names(temp)[ncol(temp)]<-paste0("PC", 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))]
##
```

```
##
##
##   # ###14.  Solve Output portion of the equation for the Output Changes:
##   # # $$QC = \sum_{i=1}^n((\frac{R_{it} + R_{it-1}}{2}) * ln(\frac{Q_{it}}{Q_{it-1}}))$$
##   # temp<-cbind.data.frame(temp, tempQC)
##   # temp[,ncol(temp)+1]<-rowSums(tempQC, na.rm = T)
##   # names(temp)[ncol(temp)]<-paste0("QC", NameBaseTotal)
##
##
##
##
##
##   ###Price Index for the entire commercial fishery ($PI_t$)
##   # We calculate the price index first by comparing by multiplying the previous years $PI_{t-1}$ by
##   # $$PI_t = PI_{t-1}*exp(ln(\frac{P_{i,t}}{P_{i,t-1}})) = PI_{t-1}*exp(PC_{t})$$
##   # Where
##   # $$PI_{i, t_{first year}} = 1$$
##   #Note that the first row of this column is = 1
##   #
##   # Then, to change the price (calulated later) into base year dollars, we use the following equatio
##   # $$PI_{t} = PI_{t}/PI_{t = baseyear}$$
##   # In this example, we'll decide that the base year is `r baseyr`, for whatever reason. Notice that
##   tempPI<-PriceIndex(temp, BaseColName = NameBaseTotal, baseyr, var = "PC")
##   temp[ncol(temp)+1]<-(tempPI)
##   names(temp)[ncol(temp)]<-paste0("PI", 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))]
##
##
##
##   ### 11. Total Implicit Quantity/Output for the entire commercial fishery ($Q_t = Y_t$)
##   # To get quantity estimates for total output using total value of landings divided by price index
##   temp[,ncol(temp)+1]<-temp[,names(temp) %in% paste0("V", NameBaseTotal)]/
##     temp[, names(temp) %in% paste0("PI", NameBaseTotal)]
##   names(temp)[ncol(temp)]<-paste0("Q", NameBaseTotal)
##
##
##
##
##
##   ### 12. Total Implicit Quantity/Output Index
##     Q<-names(temp)[names(temp) %in% paste0("Q", NameBaseTotal)]
##     temp[,ncol(temp)+1]<-temp[,Q]/temp[,Q][rownames(temp) %in% baseyr]
##     names(temp)[ncol(temp)]<-paste0("QI", NameBaseTotal)
##
##
##
```

```
##     ### 13. Sum Total Implicit Quantity/Output Index
##     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)
##
##
##
##     # ###14.  Solve Output portion of the equation for the Output Changes:
##     # $$QC = \sum_{i=1}^n((\frac{R_{it} + R_{it-1}}{2}) * ln(\frac{Q_{it}}{Q_{it-1}}))$$
##     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))
##       R0 = temp[, names(temp) %in% paste0("R", NameBasecategory)]
##       Q0 = temp[, names(temp) %in% paste0("Q", NameBasecategory)]
##       tempQCW[,i]<-PriceChange(R0, Q0)
##       names(tempQCW)[i]<-paste0("QCW", NameBasecategory)
##     }
##     temp<-cbind.data.frame(temp, tempQCW)
##
##     temp[,ncol(temp)+1]<-rowSums(tempQCW, na.rm = T)
##     names(temp)[ncol(temp)]<-paste0("QC", NameBaseTotal)
##
##
##   #Remove Duplicate Columns
##   temp<-temp[, !(grepl(pattern = "\\.[0-9]+", x = names(temp)))]
##   temp <- temp[, !duplicated(colnames(temp))]
##
##
##   ## 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:
##
##   ####1. When back calculated, $V_t$ did not equal $PI_t * Q_{t}$
##   # $$V_i = P_t * Q_i$$
##
##   temp0<-temp[names(temp) %in% c(paste0("Q",NameBaseTotal),
##                                  paste0("PI",NameBaseTotal),
##                                         paste0("V",NameBaseTotal))]
##
##   temp0[,(ncol(temp0)+1)]<-temp0[,paste0("Q",NameBaseTotal)]*temp0[,paste0("PI",NameBaseTotal)]
##   names(temp0)[ncol(temp0)]<-paste0("V", NameBaseTotal, "_Check")
##
##   if (length(setdiff(as.character(temp0[,paste0("V", NameBaseTotal, "_Check")]),
##                      as.character(temp0[,paste0("V", NameBaseTotal)]))) != 0) {
##   warnings.list[length(warnings.list)+1]<-"Warning: When back calculated, V_t did not equal PI_t * Q
##   }
##
##
##   ####2. When back calculated, $Q_{t}$ did not equal $V_t / PI_{t}$
```

```
##   # $$Q_{i,t} = V_t / P_{i,t}$$
##
##   temp0[,(ncol(temp0)+1)]<-temp0[,paste0("V",NameBaseTotal)]/temp0[,paste0("PI",NameBaseTotal)]
##   names(temp0)[ncol(temp0)]<-paste0("Q", NameBaseTotal, "_Check")
##
##   if (length(setdiff(as.character(temp0[,paste0("Q", NameBaseTotal, "_Check")]),
##                      as.character(temp0[,paste0("Q", NameBaseTotal)]))) != 0) {
##   warnings.list[length(warnings.list)+1]<-"Warning: When back calculated, Q_t did not equal V_t/PI_t
##   }
##
##
##   ####3. When back calculated, growth rate ?
##
##   # $$ln(Q_t/Q_{t-1}) = \sum( ( \frac{R_{i, t} + R_{i, t-1}}{2})  * ln(\frac{Q_{i,t}}{Q_{i,t-1}}))$$
##   #Part 1
##   #Part 1
##   names0<-c(paste0("Q",NameBaseTotal))
##   for (i in 1:length(category)) {
##     names0<-c(names0,
##               names(temp)[grep(pattern = paste0("Q", 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("Q",NameBaseTotal)]/
##                                       temp0[-1,paste0("Q",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("Q", 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.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))
##     }
##     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
##   }
##   #Price
##   a<-temp
##   a<-a[,grep(pattern = "P[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))
##    }
##     pp<-(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, and ",
##                                                  ifelse(length(pp)==1, 0, length(pp)-1) ," of ", nco
##                                                  " species P 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<- "_Q-Category"
##
##   a0<-data.frame(temp[,grepl(
##     pattern = paste0("Q[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)
##
##   #####Price Index
##   title00<- "_PI-Line"
##
##   a0<-data.frame(temp[,grepl(
##     pattern = paste0("PI[0-9]+_", NumberOfSpecies),
##     x = names(temp))])
##
##   a0$Year<-rownames(temp)
##
##   a <- gather(a0, Category, val, names(a0)[grepl(pattern = NumberOfSpecies, x = names(a0))], factor_l
##
##   a$cat<-as.character(lapply(X = strsplit(x = as.character(a$Category), split = paste0("_", NumberOfS
##
##   temp0<-a
##
##   g<-plotnlines(dat = temp0, title00, place)
##
##   figures.list[[length(figures.list)+1]]<-g
##   names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##
##   ########VV
##   title00<- "_VV-Line"
##
##   a0<-data.frame(temp[,grepl(
##     pattern = paste0("VV[0-9]+_", NumberOfSpecies),
##     x = names(temp))])
##
##   a0$Year<-rownames(temp)
##
##   a <- gather(a0, Category, val, names(a0)[grepl(pattern = NumberOfSpecies, x = names(a0))], factor_l
##
##   a$cat<-as.character(lapply(X = strsplit(x = as.character(a$Category), split = paste0("_", NumberOfS
##
##   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))])
##
##   names(a0)<-names(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$Category), split = paste0("_", NumberOf
##
##   temp0<-a
##
##   g<-plotnlines(dat = temp0, title00, place)
##
##   figures.list[[length(figures.list)+1]]<-g
##   names(figures.list)[length(figures.list)]<-paste0(title0, title00)
##
##
##   ########V
##   title00<- "_V-Line"
##
##   a0<-data.frame(temp[,grepl(
##     pattern = paste0("V[0-9]+_", NumberOfSpecies),
##     x = names(temp))])
##   a0<-a0[,!grepl(pattern = "REMOVED_", x = names(a0))]
##   a0<-a0[,!grepl(pattern = "VV[0-9]+_", x = names(a0))]
##
##   a0$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("_", 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)
##
##
##   ########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 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<-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<-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("
##
##   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)
##
##   #### 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)
##
##   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)
##
##   #### Price Change
##   title00<-"_PC-Line"
##   temp0<-temp
##
##   temp0<-temp0[grep(pattern = paste0("PC[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 = NumberOfSpeci
##                                     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.cl
##                                                                                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$Year), FUN = sum
##                                  Category = "Total",
##                                  na = aggregate(x = a00$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 = p
##       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]
##   }
##
##
##
##
##   return(list(temp, warnings.list, figures.list, spptable0, spp.output))
## }
```

2.3.2 A. Import and Edit data

```
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.p(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	PC0_0Total	PI0_0Total	Q0_0Total	QI0_0Total
2007	3250	5600	4700	4600	0.0000000	1.128281	4077.000	1.0480719
2008	3380	6220	5000	5020	0.0429332	1.177776	4262.269	1.0956991
2009	3150	5710	4800	4810	-0.0247311	1.149006	4186.228	1.0761510
2010	2610	3890	4700	3890	-0.1388971	1.000000	3890.000	1.0000000
2011	2490	4180	5000	4180	0.1090416	1.115209	3748.177	0.9635417
2012	2412	4150	4950	4150	-0.0247398	1.087957	3814.488	0.9805883
2013	3251	6280	5180	5280	0.1245141	1.232218	4284.956	1.1015311
2014	3431	6270	5370	5370	-0.0206491	1.207035	4448.919	1.1436810
2015	3630	6700	5700	5700	0.0464723	1.264452	4507.880	1.1588382
2016	3575	6780	5680	5680	-0.0244869	1.233866	4603.418	1.1833979

Did all of the analyses work as intended?

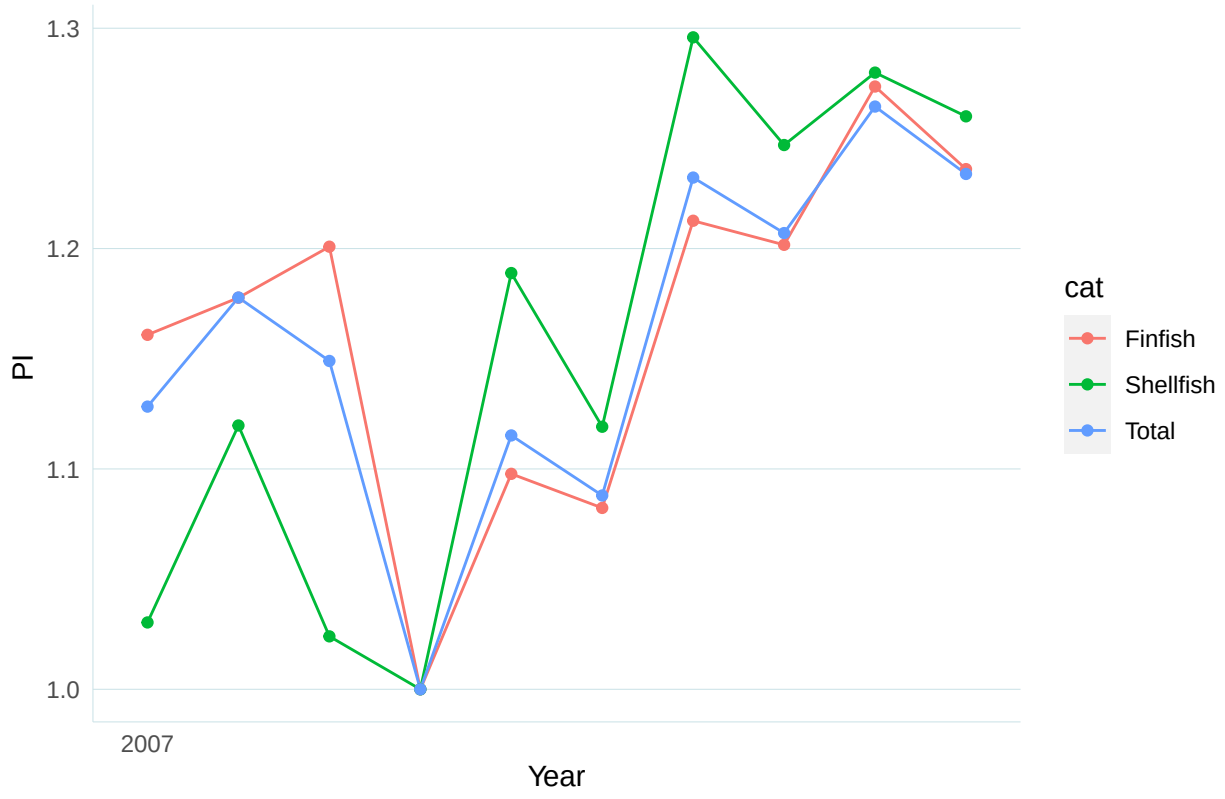
, list(NameBasecategory = c(1, 1, 1, 1, 1, 1), col = c(NA, NA, NA, NA, NA, NA), slope = c(NA, NA, NA, NA, NA, NA), intercept = c(NA, NA, NA, NA, NA, NA), R2 = c(NA, NA, NA, NA, NA, NA), R2adj = c(NA, NA, NA, NA, NA, NA), Pr = c(NA, NA, NA, NA, NA, NA), Fstat = c(NA, NA, NA, NA, NA, NA), var = c("P", "P", "V", "V", "Q", "Q"), slopecheck = c(NA, NA, NA, NA, NA, NA)), list(var = character(0), Freq = integer(0)), FYI: Rows of R_{s,i,t} for 1_0Finfish did not sum to 1, 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("P", "P", "V", "V", "Q", "Q"), slopecheck = c(NA, NA, 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, and 0 of 5 species P 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:

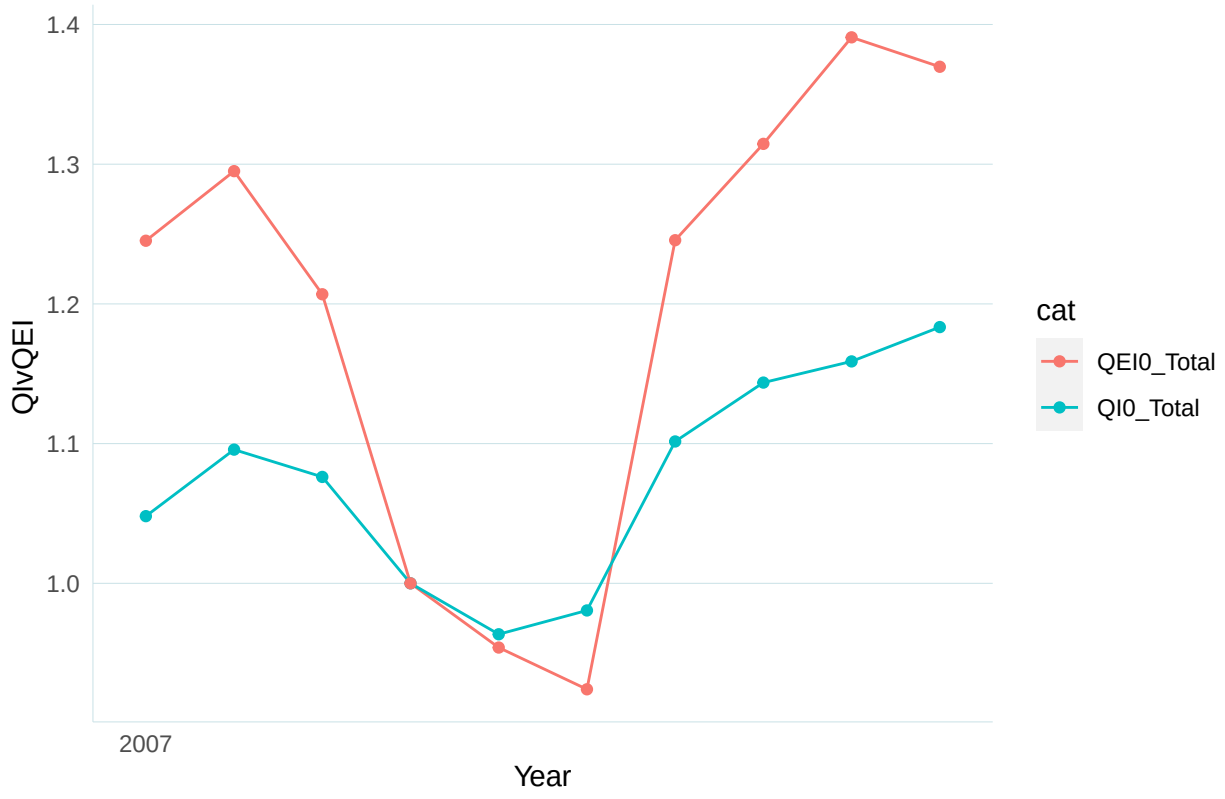
```
figures.list0$`_PI-Line`
```



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$`_QvQE-Line`
```

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
	Tsn	1st Qu.:160845
	Tsn	Median :167674
	Tsn	Mean :164501
	Tsn	3rd Qu.:169611
	Tsn	Max. :775091
	Tsn	NA's :98

Var1	Var2	Freq
	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

	Q1_0010ALEWIFE.	Q1_0011ALFONSIN.	Q1_0014AMBERJACK...	Q1_0015AMBERJACK.GREATER.
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.p(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	PC_Total	PI_Total	Q_Total	QI_Total	QC_Total
1950	4910008722	5245879816	4908258998	0.0000000	8.6266381	568965448	0.0745006	0.0000000
1951	4468280396	4794342982	4465457584	-0.1326278	7.5551320	591049580	0.0773923	0.0381150
1952	4454820075	4793747056	4452309258	0.0421719	7.8805597	564973736	0.0739779	-0.0451210
1953	4541451462	4888308503	4539800605	-0.0372843	7.5921488	597959905	0.0782971	0.0568086
1954	4783727020	5140369062	4782314508	-0.0472885	7.2414843	660405283	0.0864737	0.0993247
1955	4864502898	5204421274	4861925298	0.0329481	7.4840518	649638116	0.0850639	-0.0164319
1956	5315091181	5666538056	5310428386	-0.0166337	7.3605940	721467369	0.0944692	0.1048599
1957	4812006928	5163140746	4806379376	-0.0090782	7.2940753	658942933	0.0862823	-0.0906149
1958	4814125255	5142667236	4806065706	-0.0423587	6.9915601	687409621	0.0900097	0.0422949
1959	5136276317	5465497637	5128247707	0.0781218	7.5596545	678370645	0.0888261	-0.0132559
1960	5014090964	5343929532	5006676562	0.0699080	8.1070458	617571021	0.0808650	-0.0938965
1961	5281445600	5609304998	5271996100	-0.0461899	7.7410985	681039787	0.0891756	0.0977984
1962	5483719015	5828252598	5475910315	-0.0236598	7.5600957	724317588	0.0948424	0.0616349
1963	4940953280	5286945663	4934601380	-0.0131769	7.4611303	661374504	0.0866007	-0.0907915
1964	4658198362	5002164745	4649835362	-0.0649295	6.9920754	665015046	0.0870773	0.0055303
1965	4883101538	5224509221	4876201638	-0.0655344	6.5485457	744623593	0.0975013	0.1130747
1966	4406698497	4749445680	4401534097	-0.0876498	5.9990021	733711039	0.0960724	-0.0147363
1967	4134834274	4476942557	4129052474	0.0951561	6.5978857	625814489	0.0819444	-0.1590274
1968	4374304496	4711200579	4367242096	-0.0397328	6.3408728	688744630	0.0901845	0.0957860
1969	4451508990	4796727573	4445921890	-0.1335536	5.5481407	801335455	0.1049272	0.1514043
1970	4937890925	5284412463	4934741680	-0.0895397	5.0729532	972755207	0.1273730	0.1935499
1971	5174335253	5522231846	5172837053	0.0031173	5.0887917	1016515786	0.1331030	0.0440703
1972	4986532651	5560171110	4985938142	-0.0942769	4.6309569	1076653961	0.1409775	0.0574834
1973	4999752596	5571865210	4998950386	-0.4523659	2.9458506	1696946324	0.2221989	0.4549729
1974	5142256144	5708930416	5139191827	0.0231998	3.0149928	1704545280	0.2231939	0.0044691
1975	5077858104	5647185119	5077572751	0.0914573	3.3037386	1536917233	0.2012446	-0.1035216
1976	5585649671	6155108975	5585524133	-0.1184166	2.9347967	1903206503	0.2492067	0.2137455
1977	5371655443	5939356150	5371567038	-0.1207570	2.6009616	2065223540	0.2704212	0.0817414
1978	6158425762	6496093706	6157091268	-0.0943302	2.3668287	2601409703	0.3406297	0.2307737

	VE_Total	VV_Total	V_Total	PC_Total	PI_Total	Q_Total	QI_Total	QC_Total
1979	6468717520	6799479390	6468428160	-0.1125161	2.1149579	3058419361	0.4004707	0.1618372
1980	6558742947	6879957142	6558531303	-0.0026218	2.1094202	3109163033	0.4071151	0.0164200
1981	6021841354	6403870540	6020613522	-0.0005122	2.1083401	2855617772	0.3739158	-0.0855074
1982	6438791754	6820926341	6437693970	0.0020356	2.1126363	3047232540	0.3990059	0.0648060
1983	6394874948	6653589709	6393134325	-0.0207047	2.0693445	3089449037	0.4045337	0.0134783
1984	6400733217	6652182493	6399754108	-0.0222756	2.0237582	3162311651	0.4140744	0.0237116
1985	6327817404	6579440056	6327072795	0.0243680	2.0736790	3051134159	0.3995168	-0.0356234
1986	6087762826	6247599778	6086540382	-0.0929918	1.8895383	3221178579	0.4217825	0.0542829
1987	6970940970	7127985691	6969539624	-0.0793262	1.7454394	3993000044	0.5228451	0.2166038
1988	7345259874	7347571420	7344007016	-0.1617031	1.4848344	4946010728	0.6476327	0.2163968
1989	8703012869	8703831791	8699561960	0.0941885	1.6314869	5332290434	0.6982123	0.0762518
1990	9763270615	9757305102	9758613232	-0.0337325	1.5773706	6186633019	0.8100803	0.1486621
1991	9591954950	9594907037	9589081821	-0.0070790	1.5662439	6122342562	0.8016621	-0.0104752
1992	9910321012	9906052649	9904573944	-0.1425618	1.3581433	7292730944	0.9549132	0.1749307
1993	9931205231	9926845179	9927354742	0.1921233	1.6458251	6031840553	0.7898117	-0.1898258
1994	10054778439	10054255709	10050150258	-0.0727468	1.5303479	6567232322	0.8599161	0.0850252
1995	9630008417	9627786875	9624203652	-0.1179635	1.3600638	7076288433	0.9265722	0.0746565
1996	9340724555	9339533995	9335865765	0.0847791	1.4803976	6306323176	0.8257526	-0.1152012
1997	9583115360	9579995371	9578044682	-0.0399434	1.4224309	6733574597	0.8816971	0.0655529
1998	8960217365	8957006563	8955423113	0.1751904	1.6947881	5284096121	0.6919018	-0.2423940
1999	9062894698	9065699732	9058673959	0.0082069	1.7087543	5301332059	0.6941586	0.0032194
2000	8833974470	8835528763	8828248850	-0.0451387	1.6333383	5405033778	0.7077374	0.0193995
2001	9253069890	9247621688	9241066331	0.0241798	1.6733134	5522615375	0.7231336	0.0215234
2002	9208948179	9198719970	9192749037	0.1448039	1.9340373	4753139560	0.6223781	-0.1500453
2003	9291777375	9282057328	9277597879	-0.0073896	1.9197982	4832590103	0.6327814	0.0165753
2004	9182300008	9232270660	9105839980	-0.2017226	1.5690925	5803252408	0.7598803	0.1830358
2005	9109639858	9129244809	9002386803	-0.1121021	1.4026949	6417922382	0.8403654	0.1006744
2006	9138912762	9009775463	8895011089	-0.0917283	1.2797529	6950569249	0.9101104	0.0797414
2007	8885395234	8790589502	8658691318	-0.0313536	1.2402506	6981404770	0.9141481	0.0044151
2008	7882138532	7653089299	7525704203	-0.2368914	0.9786532	7689858000	1.0069132	0.0966097
2009	7788358546	7563486182	7438878112	0.1565040	1.1444521	6499946821	0.8511058	-0.1681399
2010	8046768286	7763910259	7637061300	-0.1349261	1.0000000	7637061300	1.0000000	0.1612190
2011	9426480100	9190352844	9063446042	-0.0785507	0.9244552	9804094647	1.2837523	0.2498879
2012	9316801850	9089997736	8965918468	-0.0245697	0.9020183	9939840567	1.3015269	0.0137513
2013	9387237694	9211378445	9087013015	-0.0273985	0.8776399	10353919943	1.3557466	0.0408118
2014	9101596982	8907330541	8783287468	0.0678094	0.9392162	9351720100	1.2245181	-0.1017983
2015	9284445654	9169488737	9048004259	0.0210287	0.9591759	9433102182	1.2351743	0.0086160
2016	9311345804	9156939746	9032212602	-0.0484661	0.9137970	9884266206	1.2942500	0.0467196
2017	9565233796	9407628445	9286207791	0.1014615	1.0113789	9181730027	1.2022596	-0.0737286

Did all of the analyses work as intended?

, 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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 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, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, 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, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,

NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 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, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 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, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, 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, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 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, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, 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(“P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”,
“P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”,
“P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”,
“P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”,
“P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”,
“P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”,
“P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”,
“P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”,
“P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”,
“P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”,
“P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”,
“P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”,
“P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “P”, “V”, “V”, “V”, “V”, “V”, “V”,

“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”,
“V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “V”, “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”), 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, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
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)), FYI: Rows of R_{s,i,t} for 1_0000Finfish did not sum to 1, 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), 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), 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), 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), 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), 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), 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), 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), var = c("P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "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"), 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)), FYI: Rows of R_{s,i,t} for 2_0000Other did not sum to 1, list(NameBasecategory = c(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), 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), 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), 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),
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), 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), 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,

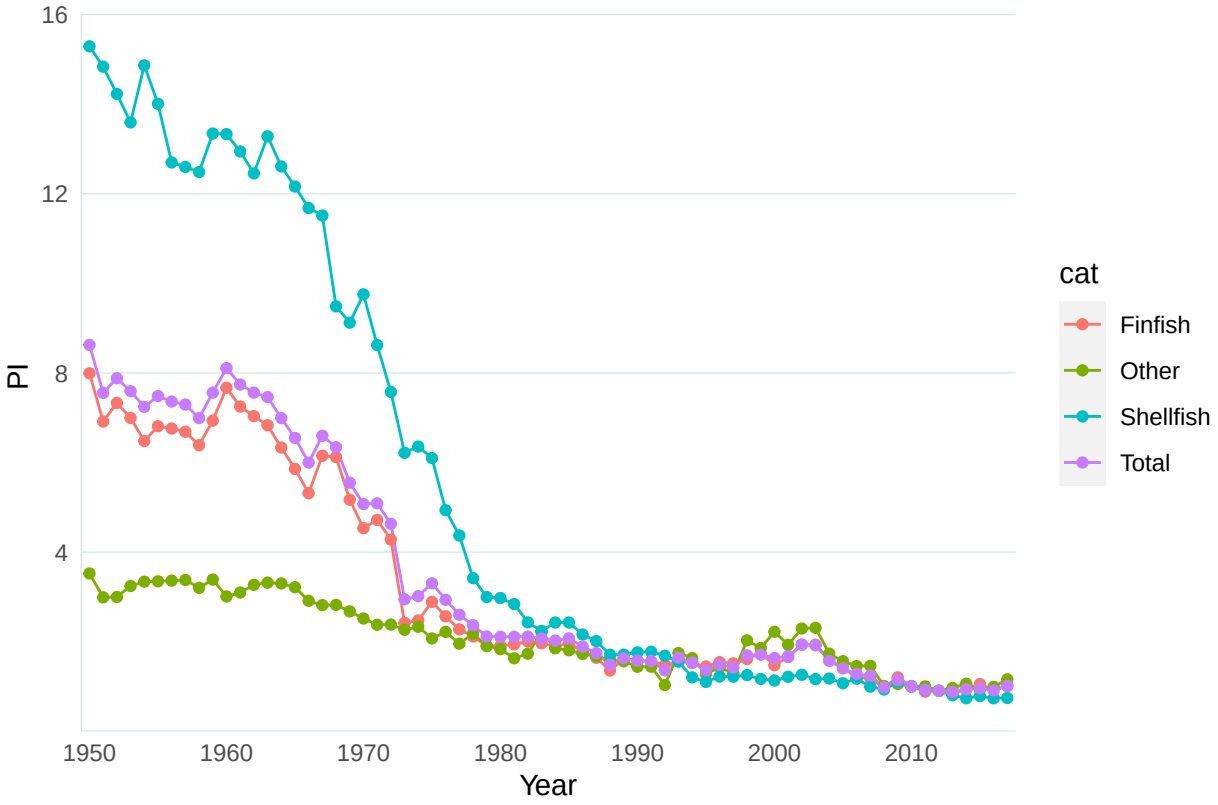
NA, NA), var = c("P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "P", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "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"), slopecheck = c(NA, NA)), FYI: Rows of R_{s,i,t} for 3_0000Shellfish did not sum to 1, 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, and 0 of 30 species P 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:

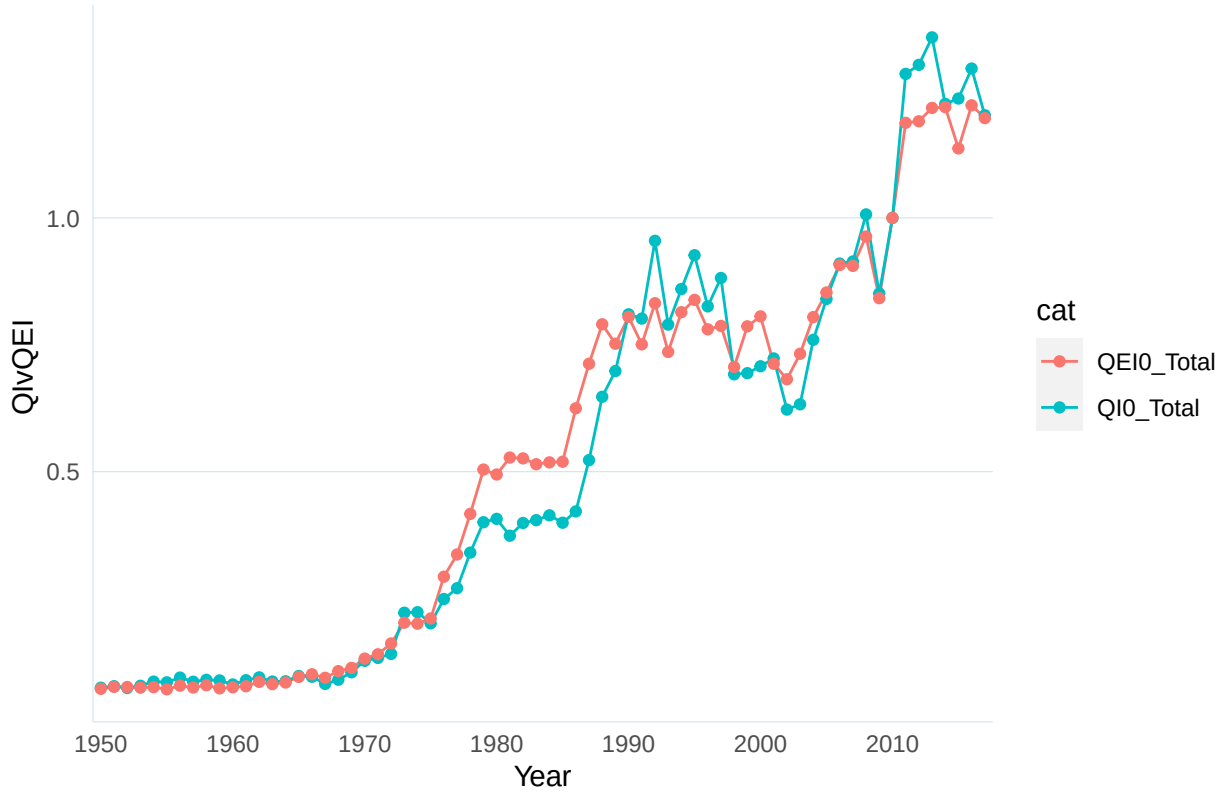
```
figures.list0$`_PI-Line`
```



2.4.5.2 Graph 2: Quantity Index Compare

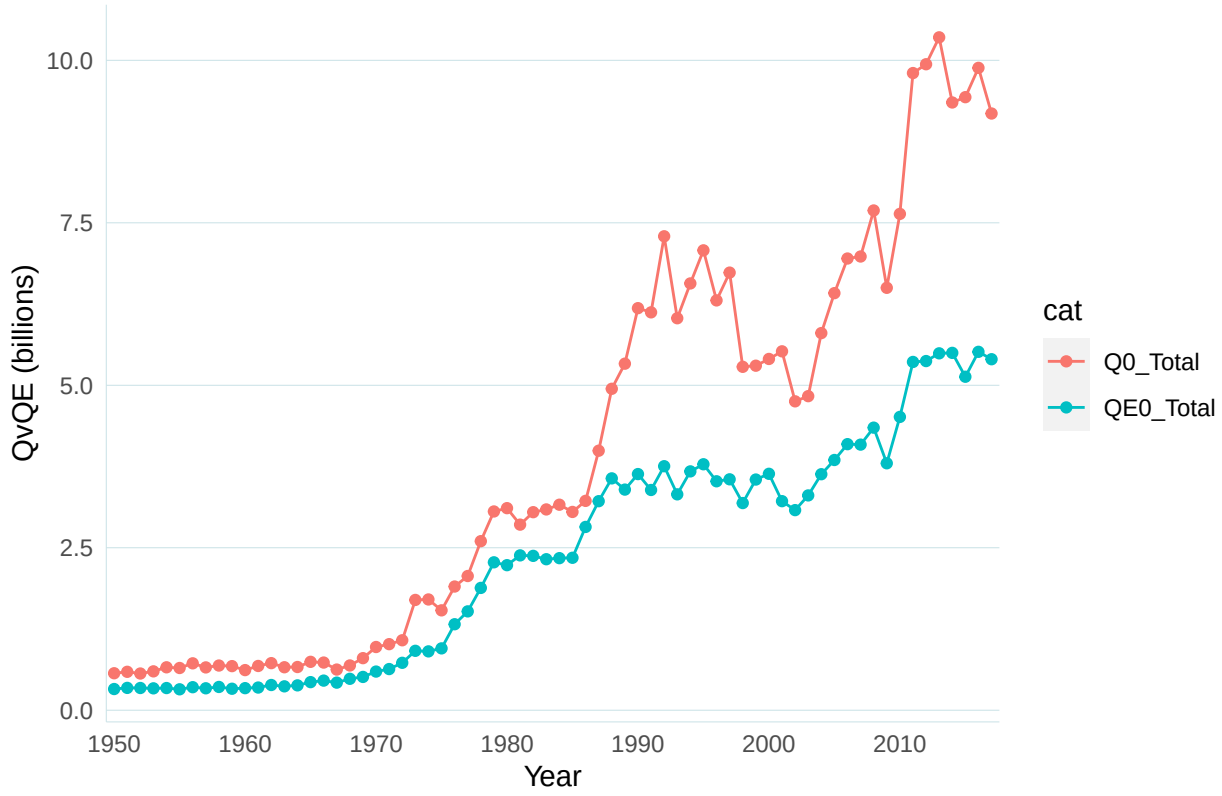
For comparison, let's recreate those graphs to make sure we are getting the same output:

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



2.4.5.3 Graph 3: Quantity Compare

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


10.0
7.5
5.0
2.5
0.0
QvQE (billions)
1950
1960
1970
1980
1990
2000
2010
Year
cat
Q0_Total
QE0_Total