R Training Documentation

2023-10-17

Introduction

Source files, directories, important functions in R

The working directory is the folder your R session 'lives in.' In general, it is good practice to have your working directory to be set to the folder which houses the open file. getwd() displays the current working directory of the session. You can set the working directory to the path of the desired folder by using setwd(…).

If ever you are confused about a package or function in R studio, typing '?' in front of the argument will display the documentation for that function in your session. For information on Vectorsurv specific functions, please see the documentation below.

```
#To get current directory
getwd()
```

```
## [1] "C:/Users/Christina/Desktop/R-tutorial"
```

```
#To get information on function in R documentation
?getwd()


#'VS_functions.R' should be in the same directory as the .rmd
#'files which access its functions, you can set the working directory

#Path to folder on my computer: "C:\Users\Christina\Desktop\R-tutorial"
#this path will be slightly different for you depending on the file location.

#Set below to the location of the downloaded files on your machine,
#make sure to change the direction of the slash from "\" to "/" to indicate opening of the folder.

setwd("/Users/Christina/Desktop/R-tutorial")
```

There is one R files. The .R file is a source file that contains all of the functions you will need to run the .RMarkdown files. You do not need to touch the source file. The .RMarkdown is a pre-coded sample report that you can customize to suit your needs.

```
#loads relevant packages and functions
source("VS_functions.R")
```

Retriving Data

getArthroCollections(. . .)

Description

getArthroCollections(. . .) obtains collections data on a year range (start_year, end_year) and agency_id. It prompts the user for their Gateway username and password before retrieving the associated data. Agency id is the number associated with your agency through Vectorsurv. If you have access to multiple agencies, id can be used to specific what data you wish to retrieve. You can only retrieve data from agencies linked to your Gateway account.

Usage

getArthroCollections(start_year, end_year, agency_code)

Arguments

- start_year: Beginning of year range
- end_year: End of year range
- agency_id: Agency ID number, if left blank, the defualt will return data for all agencies linked to you account. The majority of users are linked to one agency

```
#Example
collections = getArthroCollections(2022,2023, 55)
```

getPools(. . .)

Description

getPools(. . .) similar to getArthroCollections() obtains pools on a year range (start_year, end_year) and agency_id. It prompts the user for their Gateway username and password before retrieving the associated data. getPools() retrieve data for both mosquito and tick pools.

Usage

getPools(start_year, end_year, agency_id)

Arguments

- start_year: Beginning of year range
- end_year: End of year range
- agency_id: Agency ID number

```
#Example
pools = getPools(2022,2023, 55)
```

Write Data to file

You can save retrieved data as a .csv file in your current directory using write.csv(). That same data can be retrieved using read.csv(). Writing data to a .csv can make the rendering process more efficient when generating reports in R. We recommend that you write the data pulled from our API into a csv and then load that data when generating reports.

read.csv(. . .)

```r
#creates a file named "collections_18_23.csv" in your current directory
 write.csv(x = collections, file = "collections_18_23.csv")

#loads collections data
collections = read.csv("collections_18_23.csv")
```

Basic subseting, filtering, grouping, pivoting

Data can be subset to contain columns of interest. Sub-setting can also be used to reorder the columns in a dataframe.Do not subset collections or pools data before inputting them into Vectorsurv calculator functions to avoid losing essential columns. It is recommended to subset after calculations are complete and before inputting into a table generator. Remember, subseting, filtering, grouping and summarizing will not change the value of the data unless it is reassigned to the same variable name. We recomment creating a new variable for processed data.

Subseting

```r
#Subset using column names or index number

colnames(collections) #displays column names and associated index
```

```
##  [1] "X"                         "collection_id"
##  [3] "collection_num"            "collection_date"
##  [5] "collection_date_date_only" "comments"
##  [7] "identified_by"             "num_trap"
##  [9] "site"                      "surv_year"
## [11] "trap_nights"               "trap_problem_bit"
## [13] "user"                      "add_date"
## [15] "deactive_date"             "updated"
## [17] "id"                        "num_count"
## [19] "sex_id"                    "sex_type"
## [21] "sex_name"                  "species_id"
## [23] "species_full_name"         "species_display_name"
## [25] "agency_id"                 "agency_code"
## [27] "agency_name"               "trap_id"
## [29] "trap_acronym"              "trap_name"
## [31] "trap_presence"
```

```r
#Subseting by name
head(collections[c("collection_date","species_display_name","num_count")])
```

```
##            collection_date species_display_name num_count
## 1 2023-10-20T07:00:00.000Z         Ae melanimon        52
## 2 2023-10-20T07:00:00.000Z           Cx pipiens        16
## 3 2023-10-20T07:00:00.000Z          Cx tarsalis        42
## 4 2023-10-20T07:00:00.000Z         Ae melanimon         4
## 5 2023-10-20T07:00:00.000Z          Cx tarsalis         1
## 6 2023-10-20T07:00:00.000Z         Ae melanimon        43
```

```
#by index
head(collections[c(3,23,17)])
```

```
##   collection_num species_full_name      id
## 1           8332   Aedes melanimon 6226776
## 2           8332     Culex pipiens 6226777
## 3           8332    Culex tarsalis 6226778
## 4           8333   Aedes melanimon 6226779
## 5           8333    Culex tarsalis 6226780
## 6           8334   Aedes melanimon 6226872
```

```
#to save a subset
collections_subset = collections[c(3,23,17)]
```

Filtering and subsetting in dplyr

Dplyr is a powerful package for filtering and sub-setting data. It follows logic similar to SQL queries.

For more information on data manipulation using dplyr Click Here

Dplyr utilizes the pipe operator '%>%' to send data into functions.

```
#Subsetting columns with dplyr 'select'
collections %>%
  select(collection_date,species_display_name,num_count)%>%
  head()
```

```
##            collection_date species_display_name num_count
## 1 2023-10-20T07:00:00.000Z         Ae melanimon        52
## 2 2023-10-20T07:00:00.000Z           Cx pipiens        16
## 3 2023-10-20T07:00:00.000Z          Cx tarsalis        42
## 4 2023-10-20T07:00:00.000Z         Ae melanimon         4
## 5 2023-10-20T07:00:00.000Z          Cx tarsalis         1
## 6 2023-10-20T07:00:00.000Z         Ae melanimon        43
```

```
#filtering with dplyr 'filter'
collections %>%
  filter(species_display_name=="Cx pipiens") %>%
  head()
```

```
##    X collection_id collection_num          collection_date
## 1  2       3033442           8332 2023-10-20T07:00:00.000Z
## 2  9       3033489           8334 2023-10-20T07:00:00.000Z
## 3 17       3033497           8337 2023-10-20T07:00:00.000Z
## 4 19       3033498           8338 2023-10-20T07:00:00.000Z
## 5 23       3033525           8347 2023-10-20T07:00:00.000Z
## 6 24       3033610           8348 2023-10-20T07:00:00.000Z
##   collection_date_date_only
## 1                      TRUE
## 2                      TRUE
## 3                      TRUE
## 4                      TRUE
## 5                      TRUE
## 6                      TRUE
##                                                                   comments
## 1 Submitted using API - MVTrapID# 82801\n\t\t\t\tRandomNumber - 788422245
## 2 Submitted using API - MVTrapID# 82796\n\t\t\t\tRandomNumber - 811478541
## 3 Submitted using API - MVTrapID# 82799\n\t\t\t\tRandomNumber - 643517194
## 4 Submitted using API - MVTrapID# 82804\n\t\t\t\tRandomNumber - 582396875
## 5 Submitted using API - MVTrapID# 82811\n\t\t\t\tRandomNumber - 181335976
## 6 Submitted using API - MVTrapID# 82813\n\t\t\t\tRandomNumber - 685144888
##    identified_by num_trap   site surv_year trap_nights trap_problem_bit user
## 1  Anna;Cutshall        1  13000      2023           1            FALSE  331
## 2  Anna;Cutshall        1  13007      2023           1            FALSE  331
## 3  Anna;Cutshall        1  13010      2023           1            FALSE  331
## 4  Anna;Cutshall        1  17554      2023           1            FALSE  331
## 5 James;Brodigan        1 125844      2023           1            FALSE  331
## 6 James;Brodigan        1 125846      2023           1            FALSE  331
##                   add_date deactive_date                  updated      id
## 1 2023-10-20T18:47:24.194Z            NA 2023-10-20T18:47:25.753Z 6226777
## 2 2023-10-20T19:09:08.137Z            NA 2023-10-20T19:09:09.287Z 6226874
## 3 2023-10-20T19:15:44.615Z            NA 2023-10-20T19:15:45.726Z 6226895
## 4 2023-10-20T19:15:47.298Z            NA 2023-10-20T19:15:48.502Z 6226899
## 5 2023-10-20T19:33:20.393Z            NA 2023-10-20T19:33:21.363Z 6226953
## 6 2023-10-20T19:35:26.652Z            NA 2023-10-20T19:35:27.644Z 6227157
##   num_count sex_id sex_type        sex_name species_id species_full_name
## 1        16      4   female Females - Mixed         65     Culex pipiens
## 2        25      4   female Females - Mixed         65     Culex pipiens
## 3        21      4   female Females - Mixed         65     Culex pipiens
## 4         2      4   female Females - Mixed         65     Culex pipiens
## 5         1      4   female Females - Mixed         65     Culex pipiens
## 6         1      4   female Females - Mixed         65     Culex pipiens
##   species_display_name agency_id agency_code          agency_name trap_id
## 1           Cx pipiens        55        SAYO Sacramento-Yolo MVCD       2
## 2           Cx pipiens        55        SAYO Sacramento-Yolo MVCD       2
## 3           Cx pipiens        55        SAYO Sacramento-Yolo MVCD       2
## 4           Cx pipiens        55        SAYO Sacramento-Yolo MVCD       3
## 5           Cx pipiens        55        SAYO Sacramento-Yolo MVCD      13
## 6           Cx pipiens        55        SAYO Sacramento-Yolo MVCD      13
##   trap_acronym                  trap_name trap_presence
## 1          CO2 Carbon dioxide baited trap         FALSE
## 2          CO2 Carbon dioxide baited trap         FALSE
## 3          CO2 Carbon dioxide baited trap         FALSE
## 4         GRVD                Gravid trap         FALSE
## 5       BGSENT                BG Sentinel         FALSE
## 6       BGSENT                BG Sentinel         FALSE
```

```
#filtering multiple arguments using '%in%'
collections %>%
  filter(species_display_name %in% c("Cx pipiens", "Cx tarsalis"),
         num_trap %in% c(1:5))%>%
  head()
```

```
##    X collection_id collection_num          collection_date
## 1  2       3033442           8332 2023-10-20T07:00:00.000Z
## 2  3       3033442           8332 2023-10-20T07:00:00.000Z
## 3  5       3033443           8333 2023-10-20T07:00:00.000Z
## 4  9       3033489           8334 2023-10-20T07:00:00.000Z
## 5 10       3033489           8334 2023-10-20T07:00:00.000Z
## 6 17       3033497           8337 2023-10-20T07:00:00.000Z
##   collection_date_date_only
## 1                      TRUE
## 2                      TRUE
## 3                      TRUE
## 4                      TRUE
## 5                      TRUE
## 6                      TRUE
##                                                                   comments
## 1 Submitted using API - MVTrapID# 82801\n\t\t\t\tRandomNumber - 788422245
## 2 Submitted using API - MVTrapID# 82801\n\t\t\t\tRandomNumber - 788422245
## 3 Submitted using API - MVTrapID# 82802\n\t\t\t\tRandomNumber - 456498256
## 4 Submitted using API - MVTrapID# 82796\n\t\t\t\tRandomNumber - 811478541
## 5 Submitted using API - MVTrapID# 82796\n\t\t\t\tRandomNumber - 811478541
## 6 Submitted using API - MVTrapID# 82799\n\t\t\t\tRandomNumber - 643517194
##   identified_by num_trap  site surv_year trap_nights trap_problem_bit user
## 1 Anna;Cutshall        1 13000      2023           1            FALSE  331
## 2 Anna;Cutshall        1 13000      2023           1            FALSE  331
## 3 Anna;Cutshall        1 13000      2023           1            FALSE  331
## 4 Anna;Cutshall        1 13007      2023           1            FALSE  331
## 5 Anna;Cutshall        1 13007      2023           1            FALSE  331
## 6 Anna;Cutshall        1 13010      2023           1            FALSE  331
##                   add_date deactive_date                  updated      id
## 1 2023-10-20T18:47:24.194Z            NA 2023-10-20T18:47:25.753Z 6226777
## 2 2023-10-20T18:47:24.194Z            NA 2023-10-20T18:47:25.753Z 6226778
## 3 2023-10-20T18:47:26.712Z            NA 2023-10-20T18:47:27.486Z 6226780
## 4 2023-10-20T19:09:08.137Z            NA 2023-10-20T19:09:09.287Z 6226874
## 5 2023-10-20T19:09:08.137Z            NA 2023-10-20T19:09:09.287Z 6226875
## 6 2023-10-20T19:15:44.615Z            NA 2023-10-20T19:15:45.726Z 6226895
##   num_count sex_id sex_type        sex_name species_id species_full_name
## 1        16      4   female Females - Mixed         65     Culex pipiens
## 2        42      4   female Females - Mixed         70    Culex tarsalis
## 3         1      3     male           Males         70    Culex tarsalis
## 4        25      4   female Females - Mixed         65     Culex pipiens
## 5        35      4   female Females - Mixed         70    Culex tarsalis
## 6        21      4   female Females - Mixed         65     Culex pipiens
##   species_display_name agency_id agency_code          agency_name trap_id
## 1           Cx pipiens        55        SAYO Sacramento-Yolo MVCD       2
## 2          Cx tarsalis        55        SAYO Sacramento-Yolo MVCD       2
## 3          Cx tarsalis        55        SAYO Sacramento-Yolo MVCD       3
## 4           Cx pipiens        55        SAYO Sacramento-Yolo MVCD       2
## 5          Cx tarsalis        55        SAYO Sacramento-Yolo MVCD       2
## 6           Cx pipiens        55        SAYO Sacramento-Yolo MVCD       2
##   trap_acronym                  trap_name trap_presence
## 1          CO2 Carbon dioxide baited trap         FALSE
## 2          CO2 Carbon dioxide baited trap         FALSE
## 3         GRVD                Gravid trap         FALSE
## 4          CO2 Carbon dioxide baited trap         FALSE
## 5          CO2 Carbon dioxide baited trap         FALSE
## 6          CO2 Carbon dioxide baited trap         FALSE
```

Group by

In addition to filtering and sub-setting, data can be group by variables and summarized.

```
#groups by species and collection date and sums the number counted

collections %>%
  group_by(collection_date, species_display_name) %>%
  summarise(sum_count = sum(num_count, na.rm=T))%>%
  head()
```

```
## # A tibble: 6 x 3
## # Groups:   collection_date [1]
##   collection_date          species_display_name sum_count
##   <chr>                    <chr>                    <int>
## 1 2022-01-04T08:00:00.000Z An freeborni                 1
## 2 2022-01-04T08:00:00.000Z Cs incidens                  2
## 3 2022-01-04T08:00:00.000Z Cs inornata                  2
## 4 2022-01-04T08:00:00.000Z Cx pipiens                  22
## 5 2022-01-04T08:00:00.000Z Cx stigmatosoma              2
## 6 2022-01-04T08:00:00.000Z Cx tarsalis                  1
```

```
#groups by species and collection date and takes the average the number counted

collections %>%
  group_by(collection_date, species_display_name) %>%
  summarise(sum_count = mean(num_count, na.rm=T))%>%
  head()
```

```
## # A tibble: 6 x 3
## # Groups:   collection_date [1]
##   collection_date          species_display_name sum_count
##   <chr>                    <chr>                    <dbl>
## 1 2022-01-04T08:00:00.000Z An freeborni              1
## 2 2022-01-04T08:00:00.000Z Cs incidens               1
## 3 2022-01-04T08:00:00.000Z Cs inornata               1
## 4 2022-01-04T08:00:00.000Z Cx pipiens                2.44
## 5 2022-01-04T08:00:00.000Z Cx stigmatosoma           1
## 6 2022-01-04T08:00:00.000Z Cx tarsalis               1
```

Pivoting

Data can be manipulated into long and wide (spread sheet) forms using pivot_wider and pivot_longer. By default data from the API is in long form. Here we pivot on species and sex condition names using num_count as values. The end result is data with num_count values in the columns named species_sex. For more on pivoting see ??pivot_longer and ??pivot_wider

```
#l
collections_wide = pivot_wider(collections,
                               names_from = c("species_display_name","sex_name"),
                               values_from = "num_count")
```

Calculations

Abundance

getAbundance(. . .)

Description

getAbundance(. . .) uses any amount of arthro collections data to calculate the abundance for the specified parameters. The function calculates using the methods of the Gateway Abundance calculator.

Usage

getAbundance(collections,interval, species_list = NULL, trap_list = NULL, species_seperate = FALSE)

Arguments

- collections: Collections data retrieved from getArthroCollections(. . .)
- interval: Calculation interval for abundance, accepts "collection_date","Biweek","Week", and "Month.
- species_list: Species filter for calculating abundance. Species_display_name is the accepted notation. To see a list of species present in your data run unique(collections$species_display_name). If species is unspecified, the default NULL will return data for all species in data.
- trap_list: Trap filter for calculating abundance. Trap_acronym is the is the accepted notation. Run unique(collections$trap_acronym) to see trap types present in your data. If trap_list is unspecified, the default NULL will return data for all trap types.
- species_seperate: Should the species in species_list have abundance calculated separately? Setting to FALSE calculates the combined abundance. The same result can be performed by calculating on one species at the time.

```r
getAbundance(collections,
             interval = "Biweek",
             species_list = c("Cx tarsalis", "Cx pipiens"),
             trap_list = "CO2",
             species_seperate = FALSE)
```

```
##    EPIYEAR Biweek Count Trap_Events Abundance
## 1     2023     10   882          65     13.57
## 2     2023     11  3254         142     22.92
## 3     2023     12  4395         153     28.73
## 4     2023     13 15803         182     86.83
## 5     2023     14 24939         226    110.35
## 6     2023     15 24113         217    111.12
## 7     2023     16 19062         255     74.75
## 8     2023     17 12865         226     56.92
## 9     2023     18 10088         213     47.36
## 10    2023     19  7161         211     33.94
## 11    2023     20  5934         211     28.12
## 12    2023     21  2780         140     19.86
## 13    2022      4     9           3      3.00
## 14    2022      9  1358         126     10.78
## 15    2022     10  1202         133      9.04
## 16    2022     11  1969         145     13.58
## 17    2022     12  3503         159     22.03
## 18    2022     13  5630         159     35.41
## 19    2022     14 10444         154     67.82
## 20    2022      15  9722          178     54.62
## 21    2022      16  7949          186     42.74
## 22    2022      17  6501          180     36.12
## 23    2022      18  6038          166     36.37
## 24    2022      19  3798          163     23.30
## 25    2022      20  1869          120     15.57
## 26    2022      21  1189           84     14.15
```

Abundance Anomaly (comparison to 5 year average)

getAbundanceAnomaly()

Description

getAbundanceAnomaly(..) requires at least five years prior to the target_year of arthro collections data to calculate for the specified parameters. The function uses the methods of the Gateway Abundance Anomaly calculator, and will not work if there is fewer than five years of data present.

Usage

getAbundanceAnomaly(collections,interval,target_year, species_list = NULL, trap_list = NULL, species_seperate = FALSE)

Arguments

- collections: Collections data retrieved from getArthroCollections(…)
- interval: Calculation interval for abundance, accepts "collection_date","Biweek","Week", and "Month.
- target_year: Year to calculate analysis on. Collections data must have a year range of at least (target_year - 5, target_year).
- species_list: Species filter for calculating abundance. Species_display_name is the accepted notation. To see a list of species present in your data run unique(collections$species_display_name). If species is unspecified, the default NULL will return data for all species in data.
- trap_list: Trap filter for calculating abundance. Trap_acronym is the is the accepted notation. Run unique(collections$trap_acronym) to see trap types present in your data. If trap_list is unspecified, the default NULL will return data for all trap types.
- species_seperate: Should the species in species_list have abundance calculated separately? Setting to FALSE calculates the combined abundance. The same result can be performed by calculating on one species at the time.

```
collections_18_23 = getArthroCollections(2018,2023, 55)

getAbundanceAnomaly(collections_18_23,
                    interval = "Biweek",
                    target_year = 2023,
                    species_list = c("Cx tarsalis", "Cx pipiens"),
                    trap_list = "CO2",
                    species_seperate = FALSE)
```

```
##    Biweek EPIYEAR Count Trap_Events Abundance Five_Year_Avg  Delta
## 1      10    2023   882          65     13.57        12.926   4.98
## 2      11    2023  3254         142     22.92        19.666  16.55
## 3      12    2023  4395         153     28.73        37.988 -24.37
## 4      13    2023 15803         182     86.83        54.496  59.33
## 5      14    2023 24939         226    110.35        81.972  34.62
## 6      15    2023 24113         217    111.12        75.588  47.01
## 7      16    2023 19062          255     74.75        78.528  -4.81
## 8      17    2023 12865          226     56.92        66.406 -14.28
## 9      18    2023 10088          213     47.36        61.704 -23.25
## 10     19    2023  7161          211     33.94        51.736 -34.40
## 11     20    2023  5934          211     28.12        32.970 -14.71
## 12     21    2023  2780          140     19.86        20.082  -1.11
```

Infection Rate

getInfectionRate()

Description

getInfectionRate(..) requires at least five years prior to the target_year of arthro collections data to calculate for the specified parameters. The function uses the methods of the Gateway Abundance Anomaly calculator, and will not work if there is fewer than five years of data present.

Usage

getInfectionRate(pools,interval, target_year, target_disease,pt_estimate, species_list = c(NULL), trap_list = c(NULL))

Arguments

- pools: Pools data retrieved from getPools(…)
- interval: Calculation interval for abundance, accepts "collection_date","Biweek","Week", and "Month.
- target_year: Year to calculate infection rate for. This year must be present in the data.
- target_disease: The disease to calculate infection rate for–i.e. "WNV". Disease acronyms are the accepted input. To see a list of disease acronyms, run unique(pools$target_acronym).
- pt_estimate: The estimation type for infection rate. Options include: "mle","bc-"mle", "mir."
- species_list: Species filter for calculating abundance. Species_display_name is the accepted notation. To see a list of species present in your data run unique(pools$species_display_name). If species is unspecified, the default NULL will return data for all species in data.
- trap_list: Trap filter for calculating abundance. Trap_acronym is the is the accepted notation. Run unique(pools$trap_acronym) to see trap types present in your data. If trap_list is unspecified, the default NULL will return data for all trap types.

```r
IR = getInfectionRate(pools,
                      interval = "Week",
                      target_year = 2023,
                      target_disease = "WNV",
                      pt_estimate = "mle",
                      species_list = c("Cx pipiens"),
                      trap_list = c("CO2","GRVD") )
IR
```

```
##    Year Week Disease Point_Estimate    Lower_CI  Upper_CI
## 1  2023   20     WNV      0.0000000  0.00000000  4.617179
## 2  2023   21     WNV      0.0000000  0.00000000  4.119261
## 3  2023   22     WNV      0.0000000  0.00000000  3.156551
## 4  2023   23     WNV      0.5727378  0.03289081  2.738134
## 5  2023   24     WNV      0.0000000  0.00000000  1.747176
## 6  2023   25     WNV      0.5095548  0.02921417  2.445178
## 7  2023   26     WNV      2.8603431  1.45376937  5.081510
## 8  2023   27     WNV      5.8779801  3.18495627  9.940152
## 9  2023   28     WNV      7.3108593  4.66480012 10.887089
## 10 2023   29     WNV     11.5973679  7.96664790 16.265986
## 11 2023   30     WNV     11.9602412  7.42715801 18.168158
## 12 2023   31     WNV      9.9743699  6.40363989 14.753549
## 13 2023   32     WNV     12.6118506  7.38382600 20.046595
## 14 2023   33     WNV      9.9985982  5.74908449 16.099310
## 15 2023   34     WNV     16.5956182 10.93686159 24.017062
## 16 2023   35     WNV      8.4449005  3.45657161 17.297528
## 17 2023   36     WNV      8.7729202  4.48933276 15.456824
## 18 2023   37     WNV      3.5282350  1.31243740  7.706321
## 19 2023   38     WNV      2.7297894  0.72200878  7.245097
## 20 2023   39     WNV      0.0000000  0.00000000  2.223982
## 21 2023   40     WNV      1.8502043  0.33078143  5.950046
## 22 2023   41     WNV      1.0256410  0.05906777  4.851477
## 23 2023   42     WNV      0.0000000  0.00000000 20.603165
```

Vector Index

getVectorIndex()

Description

getVectorIndex()(..) requires at least five years prior to the target_year of arthro collections data to calculate for the specified parameters. The function uses the methods of the Gateway Abundance Anomaly calculator, and will not work if there is fewer than five years of data present.

Usage

getVectorIndex(collections, pools, interval, target_year, target_disease,pt_estimate, species_list=NULL,trap_list = NULL)

Arguments - collections: collections data retrieved from getCollections(…) - pools: Pools data retrieved from getPools(…)

Note: Years from pools and collections data must match

- interval: Calculation interval for abundance, accepts “collection_date”,“Biweek”,“Week”, and “Month.
- target_year: Year to calculate infection rate for. This year must be present in the data.
- target_disease: The disease to calculate infection rate. Disease acronyms are the accepted input. To see a list of disease acronyms, run unique(pools$target_acronym).
- pt_estimate: The estimation type for infection rate. Options include: “mle”,“bc-”mle”, “mir.”
- species_list: Species filter for calculating abundance. Species_display_name is the accepted notation. To see a list of species present in your data run unique(pools$species_display_name). If species is unspecified, the default NULL will return data for all species in data.
- trap_list: Trap filter for calculating abundance. Trap_acronym is the is the accepted notation. Run unique(pools$trap_acronym) to see trap types present in your data. If trap_list is unspecified, the default NULL will return data for all trap types.

```r
source("VS_functions.R")
pools=getPools(2023,2023,55)
collections= getArthroCollections(2023,2023,55)
```

```
getVectorIndex(collections, pools, interval = "Biweek",
                           2023,
                           target_disease = "WNV", pt_estimate = "bc-mle",
                           species_list=c("Cx tarsalis"),

                           trap_list =  c("CO2"))
```

```
##    Biweek EPIYEAR Count Trap_Events Abundance Year Disease Point_Estimate
## 1      10    2023   509          65      7.83 2023     WNV      0.0000000
## 2      11    2023  1910         142     13.45 2023     WNV      0.5205304
## 3      12    2023  2343         153     15.31 2023     WNV      0.4359300
## 4      13    2023 12226         182     67.18 2023     WNV      0.9791292
## 5      14    2023 21573         226     95.46 2023     WNV      3.2518724
## 6      15    2023 20979         217     96.68 2023     WNV      7.2460033
## 7      16    2023 17215         255     67.51 2023     WNV     10.4738708
## 8      17    2023 11019         226     48.76 2023     WNV      9.4266562
## 9      18    2023  8184         213     38.42 2023     WNV      5.9380352
## 10     19    2023  4625         211     21.92 2023     WNV      4.3159178
## 11     20    2023  3213         211     15.23 2023     WNV      0.7772615
## 12     21    2023  1479         140     10.56 2023     WNV      0.0000000
##      Lower_CI  Upper_CI VectorIndex
## 1  0.00000000  6.735594    0.000000
## 2  0.02996660  2.523691    7.001133
## 3  0.02501275  2.117993    6.674089
## 4  0.43252140  1.931931   65.777899
## 5  2.37760030  4.355111  310.423742
## 6  5.77210610  9.004829  700.543600
## 7  8.34799912 13.020608  707.091020
## 8  7.47048950 11.776108  459.643757
## 9  4.32273418  7.990012  228.139314
## 10 2.61019696  6.773983   94.604917
## 11 0.13915932  2.544228   11.837692
## 12 0.00000000  3.338641    0.000000
```

Tables

getPoolsComparisionTable()

Description

getPoolsComparisionTable() produces a frequency table for positive and negative pools counts by year and species. The more years present in the data, the larger the table.

Usage

getPoolsComparisionTable(pools,target_disease, species_seperate=F)

Arguments

- pools: Pools data retrieved from getPools(. . .)
- target_disease: The disease to calculate infection rate for–i.e. "WNV". Disease acronyms are the accepted input. To see a list of disease acronyms, run unique(pools$target_acronym).
- species_seperate: Should the pools comparison be split by species of each pool. Default is FALSE.

```
getPoolsComparisionTable(pools, "WNV", species_seperate = T)
```

```
## # A tibble: 3 x 6
## # Groups:   surv_year, species_display_name [3]
##   surv_year species_display_name Negative Confirmed Total PercentPositive
##       <int> <chr>                   <int>     <int> <int>           <dbl>
## 1      2023 An freeborni                1         0     1            0
## 2      2023 Cx pipiens               3820       247  4067            6.07
## 3      2023 Cx tarsalis              3392       393  3785           10.4
```

Styling Dataframes with kable

Professional looking tables can be produced using the kable and kableExtra packages.

```
AbAnOutput = getAbundance(collections,
                    interval = "Biweek",

                    species_list = c("Cx tarsalis", "Cx pipiens"),
                    trap_list = "CO2",
                    species_seperate = FALSE)
head(AbAnOutput)
```

```
##   EPIYEAR Biweek Count Trap_Events Abundance
## 1    2023     10   882          65     13.57
## 2    2023     11  3254         142     22.92
## 3    2023     12  4395         153     28.73
## 4    2023     13 15803         182     86.83
## 5    2023     14 24939         226    110.35
## 6    2023     15 24113         217    111.12
```

```
#As a kable table where column names, font_size, type and much more can be customized

AbAnOutput %>%
  kbl(col.names = c("Disease Year", "Biweek", "Count","Trap Events", "Abundance")) %>%
  kable_styling(bootstrap_options = "striped",
                font_size = 14,
                latex_options="scale_down")%>%
  footnote(general = "Table X: Combined biweekly Abundance Calculation for Cx. tarsalis, pipiens in CO2
```

Data using datatables

Interactive html only tables can be produced using the DT package. DT tables allow for sorting and filtering with in a webpage. These are ideal for viewing data but are not compatable with pdf or word formats.

```
#AbAnOutput %>%
  #datatable(colnames =  c("Disease Year", "Biweek", "Count","Trap Events", "Abundance"))
```

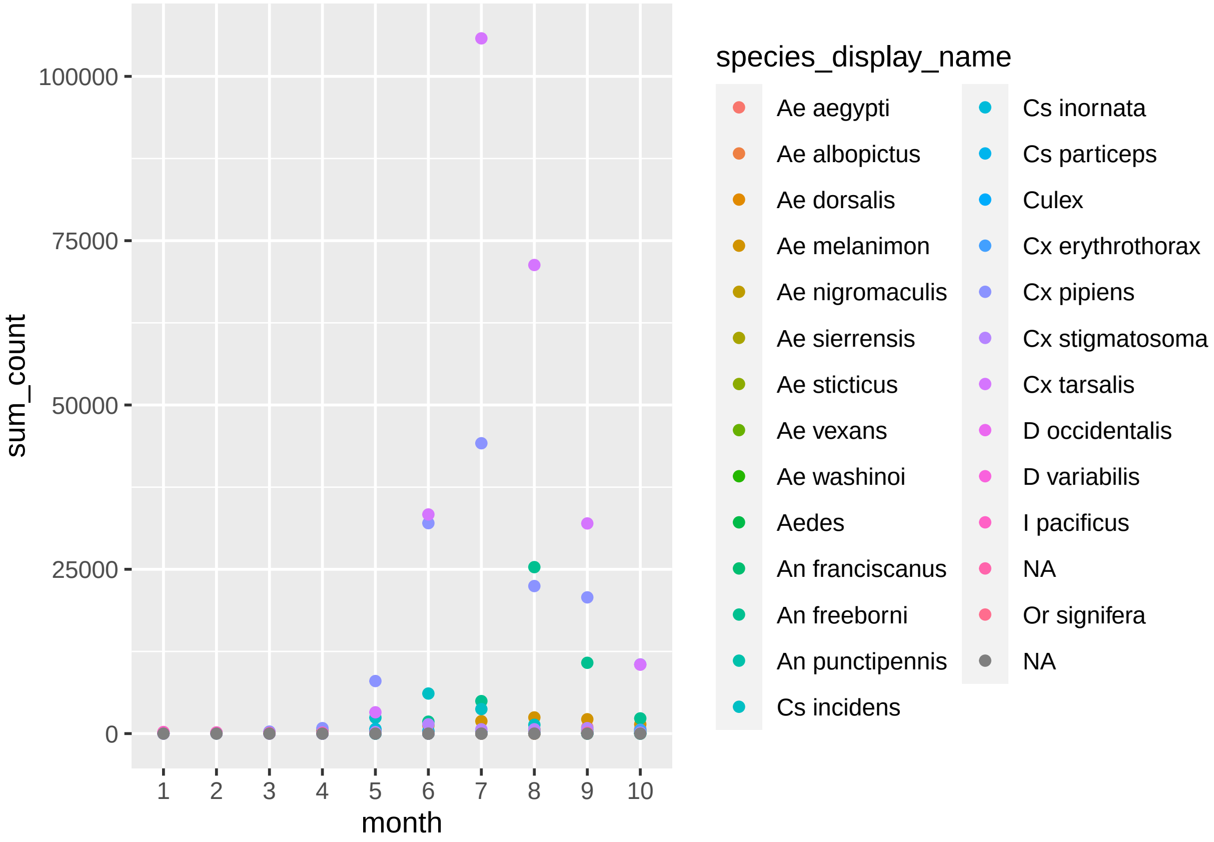
Disease Year	Biweek	Count	Trap Events	Abundance
2023	10	882	65	13.57
2023	11	3254	142	22.92
2023	12	4395	153	28.73
2023	13	15803	182	86.83
2023	14	24939	226	110.35
2023	15	24113	217	111.12
2023	16	19062	255	74.75
2023	17	12865	226	56.92
2023	18	10088	213	47.36
2023	19	7161	211	33.94
2023	20	5934	211	28.12
2023	21	2780	140	19.86

Table X: Combined biweekly Abundance Calculation for Cx. tarsalis, pipiens in CO2 traps

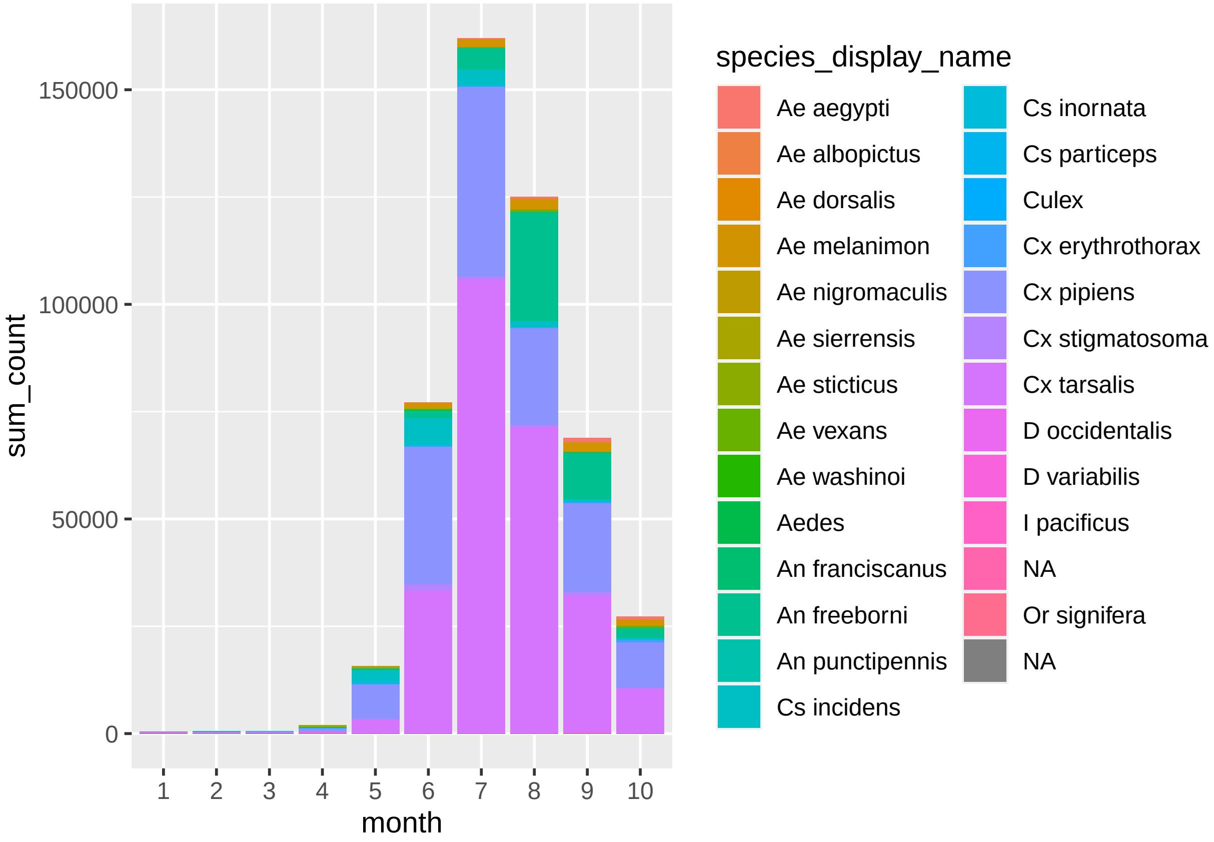
Charts and Graphs

Ggplot is a easy to use plotting library in R. Gplot syntax consists of creating a ggplot object with a dataframe and adding subsequent arguments to that object. Aesthetics (aes) in ggplot represents the data mapping aspect of the plot. A simple example using collections is shown below.

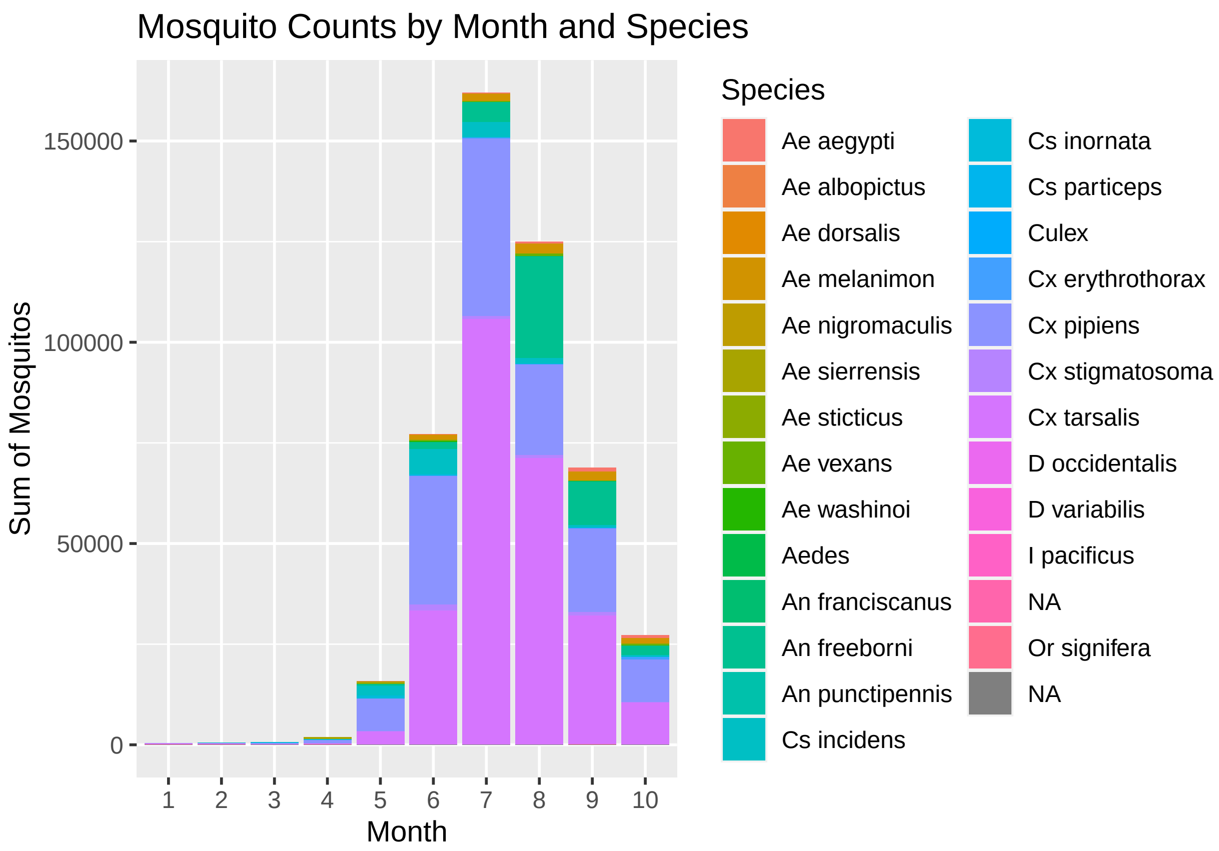
```r
#creates a month column and translates numerics
collections$month = as.factor(month(collections$collection_date))
collections_sums = collections %>%
  group_by(month, species_display_name) %>%
  summarise(sum_count = sum(num_count, na.rm=T))

#ggplot with dots a values for each species
ggplot(data=collections_sums,
       aes(x = month, y = sum_count, color = species_display_name))+
  geom_point()
```



```
#bar chart
ggplot(data=collections_sums,
       aes(x = month, y = sum_count, fill = species_display_name))+
  geom_bar(stat="identity")
```



```
#adding labels
ggplot(data=collections_sums,
       aes(x = month, y = sum_count, fill = species_display_name))+
  geom_bar(stat="identity") +
  labs(title = "Mosquito Counts by Month and Species",x = "Month",
       y = "Sum of Mosquitos",
       fill = "Species")
```



When plotting with libraries in R, it is easiest when the data is prepared in long form. Most calculator outputs from our functions are in wide form. The following wrapper functions help process and plot this data.

ProcessAbunAnom()

Description

ProcessAbunAnom() processes the output returned from getAbundanceAnomaly() into a long form suitable for plotting in ggplot.

Usage

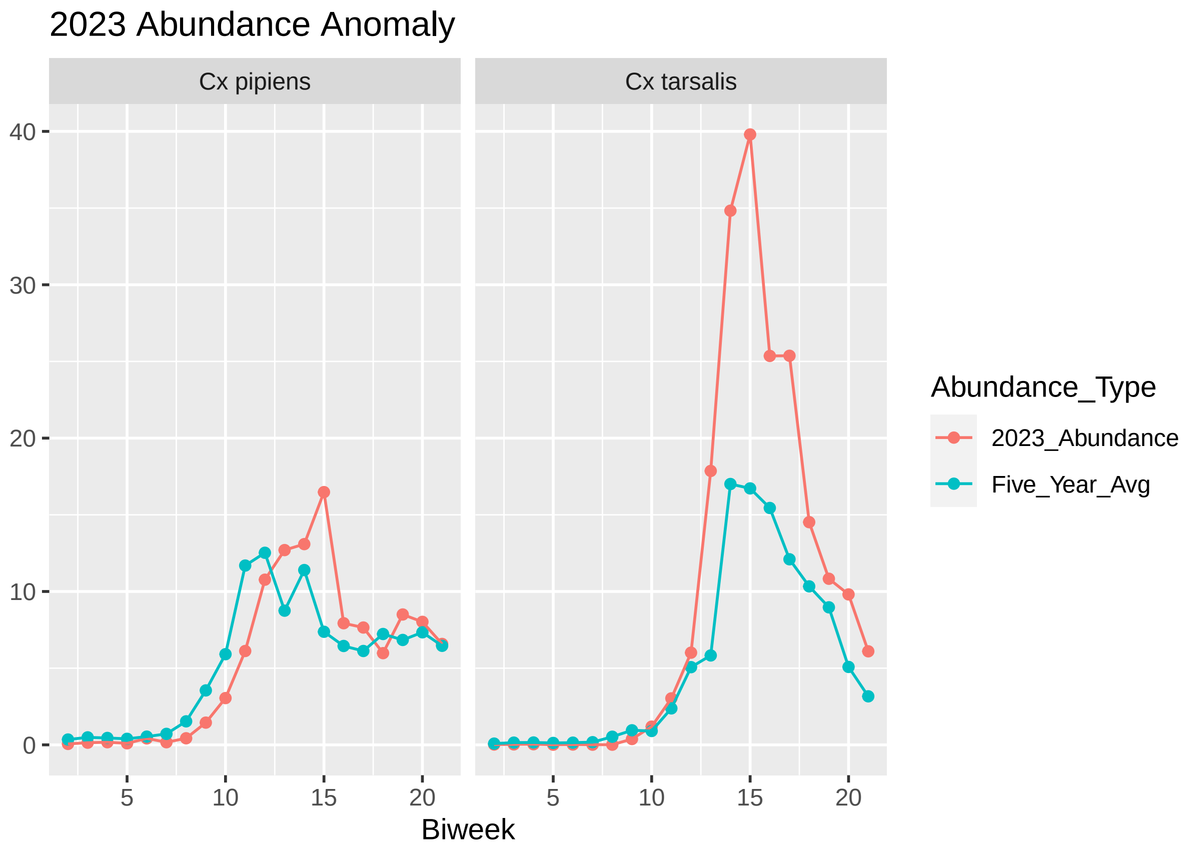
ProcessAbunAnom(AbAnomOutput)

Arguments

- AbAnomOutput: Output from returned getAbundanceAnomaly()

```
AbAnOut = getAbundanceAnomaly(collections_18_23,
                              interval = "Biweek",
                              target_year = 2023,
                              species_list = c("Cx tarsalis", "Cx pipiens"),
                              species_seperate = TRUE) #species_seperate set to true will allow for us


AbAnOut_L = ProcessAbunAnom(AbAnOut)

#Example using ggplot
```

```
AbAnOut_L %>%  filter(Abundance_Type %in% c("2023_Abundance",
                                              "Five_Year_Avg"))%>%
  ggplot(aes(x=Biweek,
             y= Abundance_Calculation,
             color = Abundance_Type)) +
  geom_point()+
  geom_line() +
  facet_wrap(~species_display_name)+
  labs(title="2023 Abundance Anomaly",y = "")
```



plotInfectionRate()

Description

plotInfectionRate() plots the output returned from getInfectionRate() with confidence intervals using ggplot

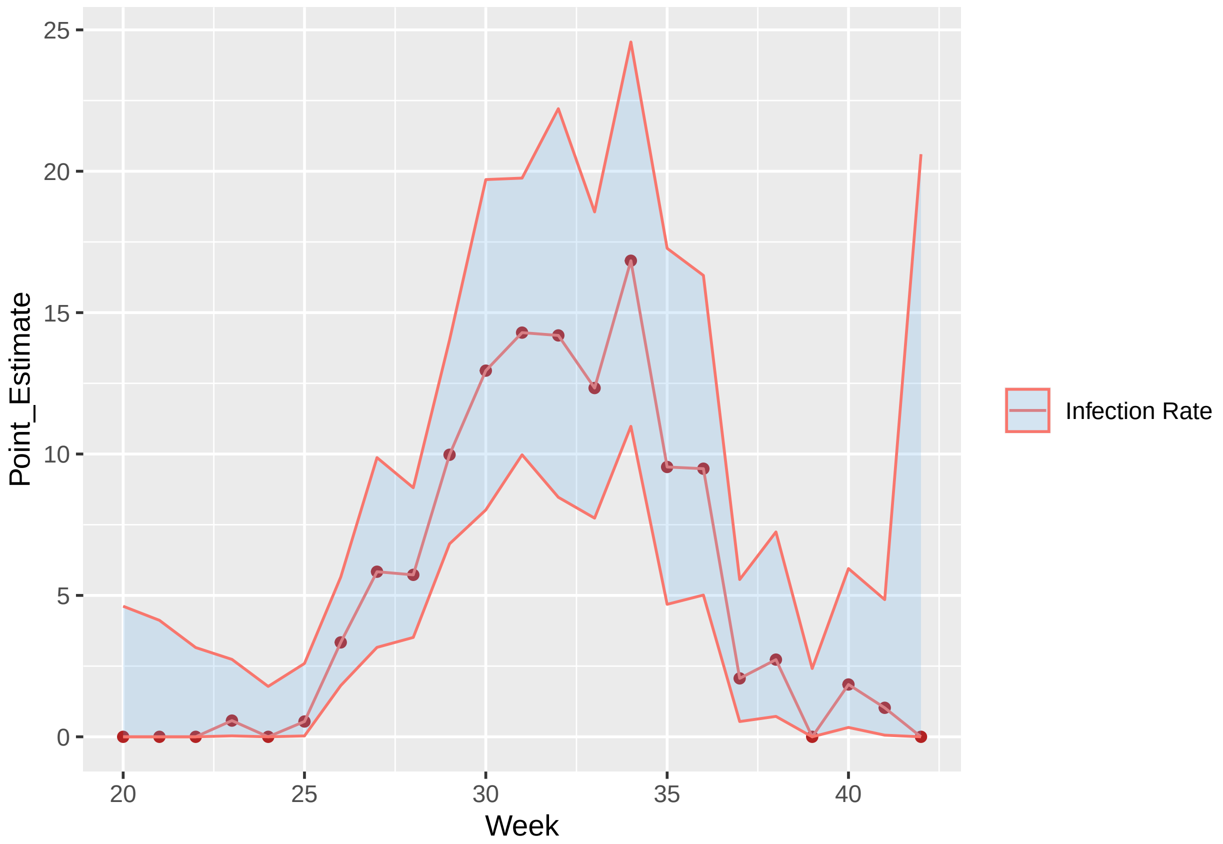
Usage

plotInfectionRate(InfRtOutput)

Arguments

- InfRtOutput: Output from returned getInfectionRate()

```
IR = getInfectionRate(pools,
                      interval = "Week",
                      target_year = 2023,
                      target_disease = "WNV",
                      pt_estimate = "mle",
                      species_list = c("Cx pipiens"),
                      trap_list = c("CO2","GRVD") )

plotInfectionRate(InfRtOutput = IR)
```


Point_Estimate
Week
Infection Rate