outbreakR : Insect outbreak reconstructions in R

T. Michael Ellis (toddellis.tx AT gmail.com)

Table of Contents

Background

During my master's work, I reconstructed western spruce budworm outbreaks using tree-ring records for north-central Washington State. The tools I used to reconstruct insect outbreaks – based on DOS software from the early-to-mid '90s, **OUTBREAK** (Holmes & Swetnam 1996) – were borrowed from my graduate advisor's doctoral research (Flower 2014).

OUTBREAK has long been the standard software used for outbreak reconstructions – specifically Douglas-fir tussock moth and western spruce budworm – but its methods are black-boxed within the software and based on datasets from the American Southwest. Studies in insect behaviors and genetics have shown that population behaviors can differ between groups, which suggests that OUTBREAK may prevent researchers from editing key inputs for the underlying equations to best reflect their study area.

See Jerry A. Powell's edited 1995 anthology, *Biosystematic Studies of Conifer-feeding Choristoneura in the Western United States*, for an overview of behavioral studies, and Brunet et al.'s (2016) genetic study of the 2-year and western spruce budworms.

To bypass this potential issue, Flower et al. (2014) effectively translated OUTBREAK's methods into a series of **Microsoft Excel** sheets, which allowed users to include custom parameters like minimum outbreak duration.

The Excel methods remain complex and convoluted – formatting a new dataset required an exorbitant amount of time, errors were common, and software updates would break the methods entirely, requiring rewriting portions of the code.

I hope to combine the best of these two methdos – Excel's customizability, OUTBREAK's ease-of-use – with the wealth of statistical packages and data visualization tools provided by the R programming language.

As of now, I've written two parts to these methods: The development of corrected indices – i.e., growth indices with the climate signal removed – and site-level outbreak records.

1. Corrected indices

The `corIndex()` function below uses input host site ring-width indices and a nonhost chronology to remove the climatic signal (shared between host and non-host indices) from the host sites. This outputs 'corrected' indices, or CIs, where positive values represent positive growth from non-climatic influences, and negative values represent limited growth from, we assume, biological means. In the sample cases, we assume negative growth impacts are caused by regionally-endemic insect outbreaks like the western spruce budworm.

To describe the inputs (`corIndex <- function( host, nonhost, scale = TRUE)`):

(1) `host` should be your host ring-width indices (i.e., multiple trees over multiple columns). See the `dplR` package for help producing these.

As of now, the input format for this remains one of my biggest concerns. Most data analysis makes use of narrow-format data, but `dplR`, *the* R package of choice for any dendrochronological study, defaults not only to wide-format data – which needlessly increases the size of the data, as well as any processing time – but forces important year data to be contained as row-name information, which can be difficult to access in any analyses. Right now I have the input set to match `dplR`'s wide-format data, and the output is transformed to narrow-format data.

Additionally, cores from individual trees should be averaged together ahead of time. `dplR`'s `treeMean()` function is a good option for this if you haven't done it externally. The way `treeMean()` handles tree IDs isn't ideal, so this is something to think about.

(2) `nonhost` should be a single non-host site / region chronology. See the `dplR` package for help producing these.

Again, this is currently set to take `dplR`'s input format for chronologies – with year data as the row names and a single column for the nonhost chronology. Definitely will require editing, e.g., options for standard or residual chronologies are completely ignored, and it currently assumes the input is *only* comprised of two columns: year and chronology. Given that one may want to try either a principal components output or basal area increments, this may be challenging to keep with `dplR` formats.

(3) `scale` is an optional feature which will normalize the resultant corrected indices if set to `TRUE`. This is on by default.

I'm unsure if this should even be optional, given the outbreak-reconstruction function that follows assumes normalized data. This needs to be adjusted either here or in the outbreak-reconstruction function.

```
corIndex <- function( host, nonhost, scale = TRUE) {

  require( tidyverse)
```

```
  # assumes dplR format with years as rowname data
  # for whatever reason, this doesn't work within the subsequent series of pipe-down
steps
  nonhost$year <- as.numeric( rownames( nonhost))

  host2 <- host %>%
    # again, assuming dplR format
    mutate( year = as.numeric( rownames( host))) %>%
    # changes dplR's wide-format data to narrow
    gather( "treeID", "host", -"year", na.rm = TRUE) %>%
    # merges with nonhost data
    merge( nonhost, by = "year") %>%
    # reorders data
    arrange( treeID, year) %>% ## unnecessary?
    # ensures next line's equation performed uniquely for each tree
    group_by( treeID) %>%
    # calculate corrected indices
    mutate( ci = host - ( sd( host) / sd( nonhost)) *
                        ( nonhost - mean( nonhost)))

  # option to normalize the corrected indices
  if( scale == TRUE) {
    host2 <- host2 %>%
      group_by( treeID) %>% ## unnecessary?
      mutate( ci = scale(ci))
  } else {
    host2
  }

  host <- host2
  rm(host2)
  host

}
```

2. Outbreak reconstructions

The `outbreak()` function takes corrected indices – the output of the above `corIndex()` function – and determines years of insect outbreaks based on user-defined critea.

To describe the function's options (`outbreak <- function( ci, min = 4, sd = -1.28, perc = TRUE)`:

(4) `ci` represents the corrected indices produced by `corIndex()`

At this stage, we're assuming narrow-format data.

(5) `min` denotes the minimum number of consecutive years that must be impacted by low growth.

It must be between 2 and 10 years, and is by default set to 4. This should be customized to most closely match available historical outbreak records.

The minimum outbreak duration used by the OUTBREAK software is always set to 8 years.

From briefly experimenting with this, a 2-year minimum does not produce usable data.

(6) `sd` denotes the standard deviation at least one year of growth falls under within each outbreak period.

This is set to -1.28 as an arbitrary standard developed for OUTBREAK, but may be customized to better match historical records.

Until I'm able to come up with a good explanation for *why* this should be changed, or what changing it *means* (e.g., what is -1.28 standard deviations to -1.31?), it's probably safe to go with the original -1.28 OUTBREAK setting. It's also likely that customizing the minimum outbreak duration will be enough to match your study area's historical records.

(7) `prop` is an optional setting to report the proportion of trees reporting outbreak conditions by year.

This returns a percentage of sampled, living trees for each year. This is set to `TRUE` as a default.

```
outbreak <- function( ci, min = 4, sd = -1.28, prop = TRUE) {

  require( tidyverse)
  require( data.table) # shift()

  # ensure minimum outbreak duration falls within correct range
  if( !is.na( min) && ( min < 2 || min > 10)) {

    stop( "minimum outbreak duration must be >= 2 years and <= 10 years")

  }

  # create running count w/ 2-year consecutive outbreak record
  ci2 <- ci %>%
    group_by( treeID) %>%
    # creates binary low-growth timeseries
    mutate( conYrs = ifelse( ( ci < 0 | shift( ci, 1, type = "lag") < 0) *
                             ( ci < 0 | shift( ci, 1, type = "lead") < 0),
                             1, 0)) %>%
    # creates running count from binary data
    mutate( conYrs = sequence( rle( conYrs)$lengths) * conYrs)

  ## n.b. the following corrects for a possible bug,
  ## where a handful of NAs are created at the earliest year
  ## of recorded growth for a small number of treeIDs
  ## these NAs *should not* be outbreak years on any of the datasets I've tried,
  ## so this fix may be enough to avoid issues
  ## I suspect it's caused by grouping by treeID and rolling between the most
  ## recent year (w/ an outbreak) to the earliest year of a new tree
  ci2[ is.na(ci2["conYrs"]), "conYrs"] <- 0

  if( is.na( min)) {

    # should probably combine earlier duration stop-point here
    stop( "minimum outbreak duration must be >= 2 years and <= 10 years")
```

```
} else if( min >= 2 && min <= 10) {

  # if minimum outbreak duration is 2 years
  # reminder: this 2-year setting may not work!
  if( min == 2) {
    ci2 <- ci2 %>%
      group_by( treeID) %>%
      mutate( outbreakBinary = ifelse( conYrs > 2 - 1 ||
                                 shift( conYrs, 1, type = "lead") >= 2,
                                 1, 0))
  }

  # if minimum outbreak duration is 3 years
  if( min == 3) {
    ci2 <- ci2 %>%
      group_by( treeID) %>%
      mutate( outbreakBinary = ifelse( conYrs > 3 - 1 |
                                         shift( conYrs, 1, type = "lead") >= 3 |
                                         shift( conYrs, 2, type = "lead") >= 3,
                                       1, 0))
  }

  # if minimum outbreak duration is 4 years -- DEFAULT
  if( min == 4) {
    ci2 <- ci2 %>%
      group_by( treeID) %>%
      mutate( outbreakBinary = ifelse( conYrs > 4 - 1 |
                                         shift( conYrs, 1, type = "lead") >= 4 |
                                         shift( conYrs, 2, type = "lead") >= 4 |
                                         shift( conYrs, 3, type = "lead") >= 4,
                                       1, 0))
  }

  # if minimum outbreak duration is 5 years
  if( min == 5) {
    ci2 <- ci2 %>%
      group_by( treeID) %>%
      mutate( outbreakBinary = ifelse( conYrs > 5 - 1 |
                                         shift( conYrs, 1, type = "lead") >= 5 |
                                         shift( conYrs, 2, type = "lead") >= 5 |
                                         shift( conYrs, 3, type = "lead") >= 5 |
                                         shift( conYrs, 4, type = "lead") >= 5,
                                       1, 0))
  }

  # if minimum outbreak duration is 6 years
  if( min == 6) {
    ci2 <- ci2 %>%
      group_by( treeID) %>%
      mutate( outbreakBinary = ifelse( conYrs > 6 - 1 |
                                         shift( conYrs, 1, type = "lead") >= 6 |
                                         shift( conYrs, 2, type = "lead") >= 6 |
                                         shift( conYrs, 3, type = "lead") >= 6 |
                                         shift( conYrs, 4, type = "lead") >= 6 |
```

```
                                        shift( conYrs, 5, type = "lead") >= 6,
                                      1, 0))
  }

  # if minimum outbreak duration is 7 years
  if( min == 7) {
    ci2 <- ci2 %>%
      group_by( treeID) %>%
      mutate( outbreakBinary = ifelse( conYrs > 7 - 1 |
                                        shift( conYrs, 1, type = "lead") >= 7 |
                                        shift( conYrs, 2, type = "lead") >= 7 |
                                        shift( conYrs, 3, type = "lead") >= 7 |
                                        shift( conYrs, 4, type = "lead") >= 7 |
                                        shift( conYrs, 5, type = "lead") >= 7 |
                                        shift( conYrs, 6, type = "lead") >= 7,
                                      1, 0))
  }

  # if minimum outbreak duration is 8 years
  if( min == 8) {
    ci2 <- ci2 %>%
      group_by( treeID) %>%
      mutate( outbreakBinary = ifelse( conYrs > 8 - 1 |
                                        shift( conYrs, 1, type = "lead") >= 8 |
                                        shift( conYrs, 2, type = "lead") >= 8 |
                                        shift( conYrs, 3, type = "lead") >= 8 |
                                        shift( conYrs, 4, type = "lead") >= 8 |
                                        shift( conYrs, 5, type = "lead") >= 8 |
                                        shift( conYrs, 6, type = "lead") >= 8 |
                                        shift( conYrs, 7, type = "lead") >= 8,
                                      1, 0))
  }

  # if minimum outbreak duration is 9 years
  if( min == 9) {
    ci2 <- ci2 %>%
      group_by( treeID) %>%
      mutate( outbreakBinary = ifelse( conYrs > 9 - 1 |
                                        shift( conYrs, 1, type = "lead") >= 9 |
                                        shift( conYrs, 2, type = "lead") >= 9 |
                                        shift( conYrs, 3, type = "lead") >= 9 |
                                        shift( conYrs, 4, type = "lead") >= 9 |
                                        shift( conYrs, 5, type = "lead") >= 9 |
                                        shift( conYrs, 6, type = "lead") >= 9 |
                                        shift( conYrs, 7, type = "lead") >= 9 |
                                        shift( conYrs, 8, type = "lead") >= 9,
                                      1, 0))
  }

  # if minimum outbreak duration is 10 years
  if( min == 10) {
    ci2 <- ci2 %>%
      group_by( treeID) %>%
      mutate( outbreakBinary = ifelse( conYrs > 10 - 1 |
                                        shift( conYrs, 1, type = "lead") >= 10 |
```

```
                                              shift( conYrs, 2, type = "lead") >= 10 |
                                              shift( conYrs, 3, type = "lead") >= 10 |
                                              shift( conYrs, 4, type = "lead") >= 10 |
                                              shift( conYrs, 5, type = "lead") >= 10 |
                                              shift( conYrs, 6, type = "lead") >= 10 |
                                              shift( conYrs, 7, type = "lead") >= 10 |
                                              shift( conYrs, 8, type = "lead") >= 10 |
                                              shift( conYrs, 9, type = "lead") >= 10,
                                            1, 0))
    }

  }

  ## n.b. in this case, the outbreaks during the most recent years
  ## may have been called NA due to the inability to look beyond
  ## the coring year
  ## e.g., min == 4 with a tree cored in 2014 that started recording an outbreak in 2
012
  ci2[ is.na( ci2[ "outbreakBinary"]), "outbreakBinary"] <- 0

  # determine outbreaks using set standard deviation
  if( !is.na( sd)) {

    ci2 <- ci2 %>%
      # personal learning experience: prior treeID grouping carried over!
      ungroup() %>%
      # uses binary outbreak data to group periods of outbreak and non-outbreak
      mutate( obGroups = cumsum( c( 0, abs( diff( outbreakBinary))))) %>%
      group_by( obGroups) %>%
      # at least one year of each outbreak period must fall below the set standard de
viation
      mutate( outbreak = ( as.numeric( any( ci < sd)) * outbreakBinary)) %>%
      ungroup()

  } else {

    stop( "set a minimum standard deviation threshold one outbreak year must fall bel
ow")

  }

  # determine proportion of site trees with outbreak conditions by year
  # may need tinkering depending on needs: use of summarize removes all other outbrea
k data
  if( prop == TRUE) {

    ci2 <- ci2 %>%
      group_by( year) %>%
      summarize( outbreakProp = mean( outbreak) * 100)

  } else {

    ci2

  }
```

```
  ci <- ci2
  rm( ci2)
  ci

}
```

Example data

I've included four host study site ring-width indices (as produced by `dplR`: `rwiMPD.csv`, `rwiSMD.csv`, `rwiTMD.csv`, and `rwiVLD.csv`), one non-host chronology developed using `dplR` and principal components analysis (`OHP_PC1.csv`), and master ring-width indices that includes all four of the study sites (`rwiMaster.csv`). For this example, I'll use the master host data.

N.B. The current requirement for `dplR`-friendly formats requires extra lines like `row.names = 1`. One reason I want to think about how input formats are handled.

```
host <- read.csv( "ex_input/host/rwiMaster.csv", header = T, row.names = 1)
nonhost <- read.csv( "ex_input/nonhost/OHP_PC1.csv", header = T, row.names = 1)

foo <- corIndex( host = host, nonhost = nonhost, scale = TRUE)
bar <- outbreak( foo, min = 4, sd = -1.28, prop = TRUE)
```

Which we can then quickly visualize:

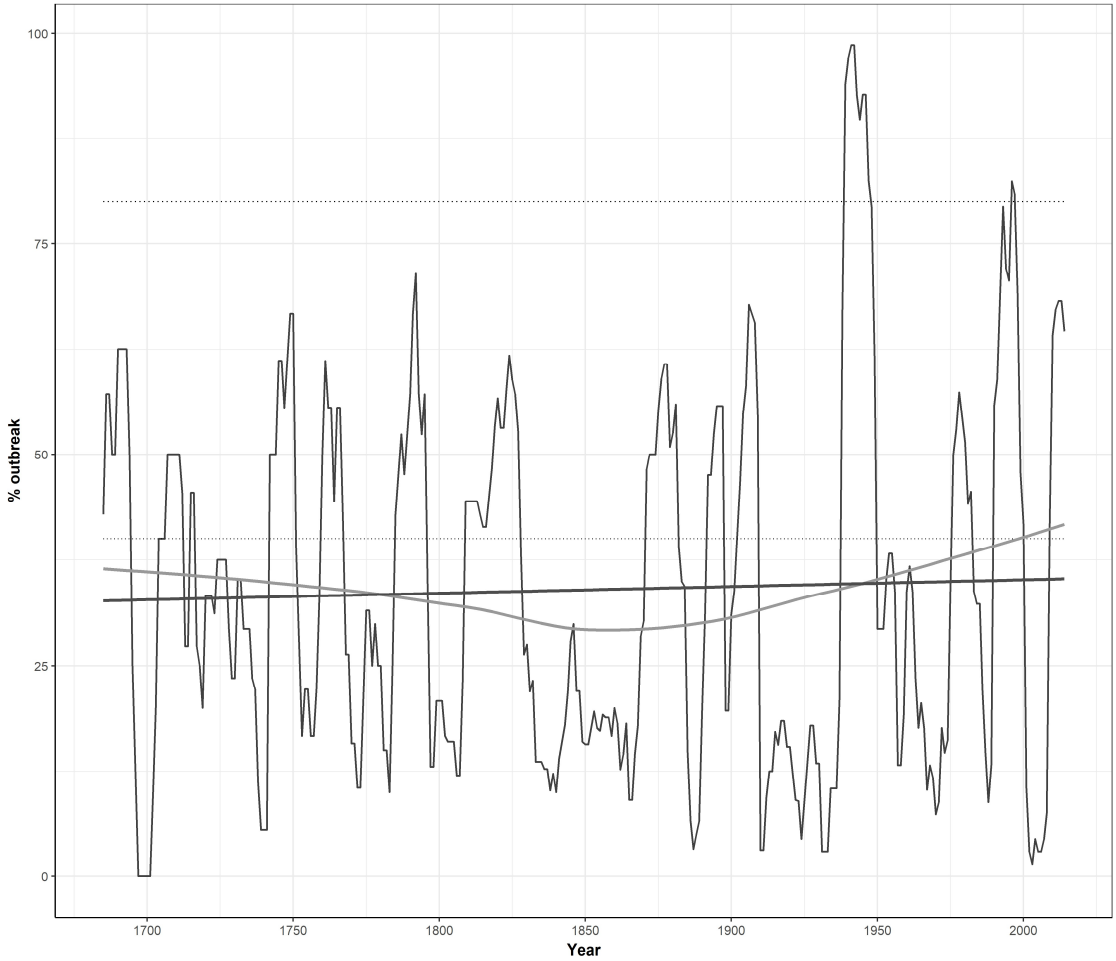

Fig. 1 *Proportion of outbreaks across all four sample study sites at once*

Next steps

What we produce from the `outbreak()` function is another question mark for these tools' goals. What all do we want to return back to the researcher with this function? A proportion itself would be most valuable, I think, but logistic regression on binary tree-level outbreak data may also be another option. Or perhaps outbreak durations or the corrected indices themselves. One vital figure I want to include here is tree population for each year, which is often visualized alongside proportional outbreak timeseries data.

Another option I'd like to think about would be how to keep site-level information, which would enable you to use the `group_by()` function provided by `tidyverse` to get outbreak statistics for multiple sites at once. This would provide easy tools that'd work well with R's many data visualization options (like the ability to facet by a factor group). I would hope with these tools, the ability to develop and model outbreak data would be easier than ever, and I'd love to further develop the customizable options to allow for more insect species and forest types, or even non-insect data that may also reside in the corrected indices.

See below for a quick example of multiple sites being explored at once!

```
# read a single study site's RWI file
foo <- read.csv( "ex_input/host/rwiMPD.csv", header = T, row.names = 1) %>%
  # create corrected indices using our nonhost data
  corIndex( nonhost) %>%
  # run the outbreak function
  outbreak() %>%
  # rename the `outbreakProp` column to reflect the site ID
  rename( MPD = outbreakProp)
# rinse, repeat
bar <- read.csv( "ex_input/host/rwiSMD.csv", header = T, row.names = 1) %>%
  corIndex( nonhost) %>%
  outbreak() %>%
  rename( SMD = outbreakProp)
baz <- read.csv( "ex_input/host/rwiTMD.csv", header = T, row.names = 1) %>%
  corIndex( nonhost) %>%
  outbreak() %>%
  rename( TMD = outbreakProp)
ob <- read.csv( "ex_input/host/rwiVLD.csv", header = T, row.names = 1) %>%
  corIndex( nonhost) %>%
  outbreak() %>%
  rename( VLD = outbreakProp) %>%
  # merge all four study sites together
  merge( foo, by = "year", all = T) %>%
  merge( bar, by = "year", all = T) %>%
  merge( baz, by = "year", all = T) %>%
  # translate from wide format to narrow
  gather( "siteID", "outbreakProp", -year, na.rm = T) %>%
  # arrange data by alphabet and year to benefit visualization
  arrange( siteID, year)
```

We can now use R's family of `tidyverse` packages to quickly and easily create visualize our data in a variety of ways.

```
ob %>%
  ggplot( aes( x = year, y = outbreakProp)) +
  theme_classic() +
  geom_line() +
  geom_smooth( method = "loess", se = F, color = "grey60") +
  geom_smooth( method = "lm", se = F, color = "grey30") +
  labs( x = "Year", y = "% outbreak") +
  facet_wrap( ~ siteID)
```

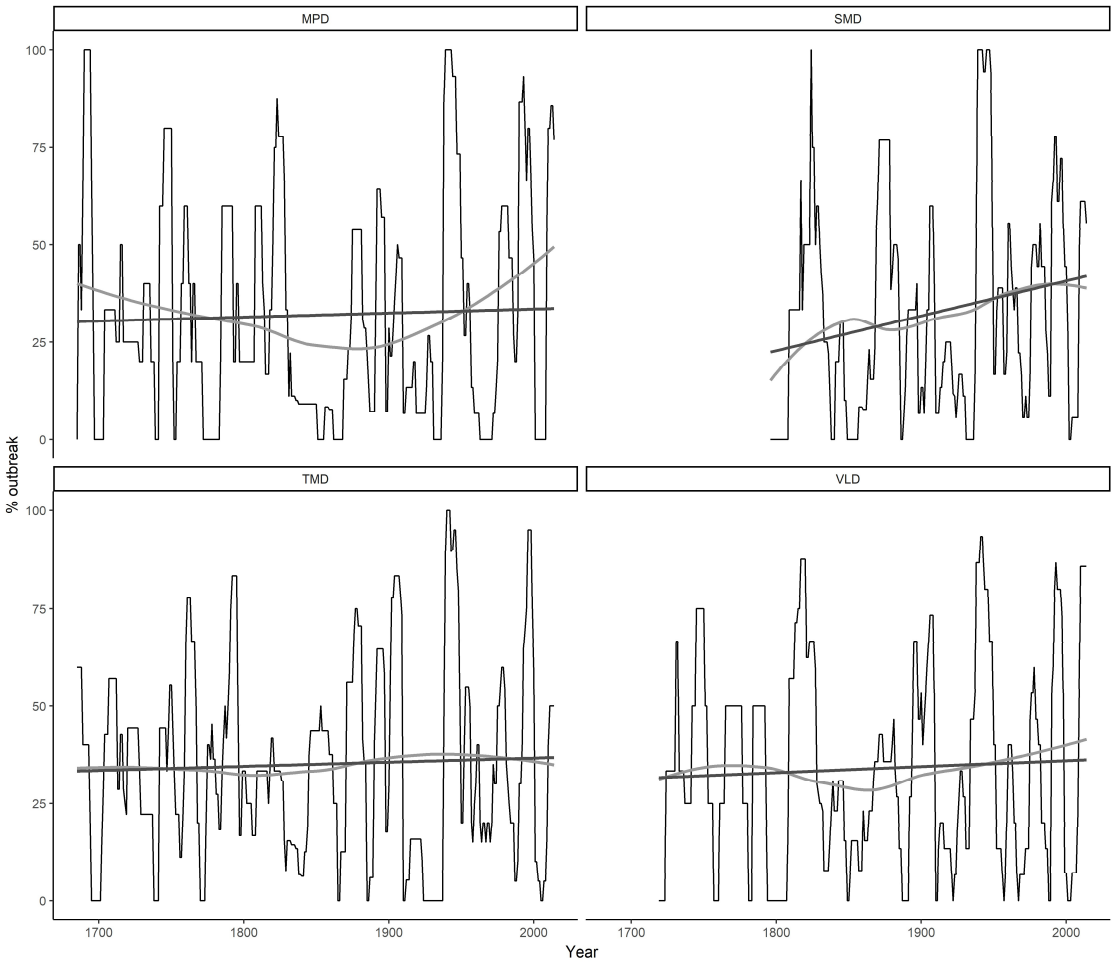

Fig. 2 *Our study sites' outbreak histories faceted by* `siteID`

```
ob %>%
  ggplot( aes( x = year, y = outbreakProp, color = siteID)) +
  theme_bw() +
  geom_line() +
  geom_smooth( method = "loess", se = F) +
  labs( x = "Year", y = "% outbreak") +
  xlim( 1900, 2014)
```

Fig. 3 *Isolating shared outbreak patterns across the 20th century*