# An introduction to the agouti package

## Introduction

This vignette describes the use of the **agouti** R package, a generic R package for exploratory analyses of multi-resolution data.

Install **agouti** using:

```
#install.packages('agouti') # update to Tim's github
```

### What is multi-resolution data and why are standard exploratory analyses difficult

Multi-resolution data is also referred to as mixed frequency, mixed data and mis-aligned data. Examples of multi-resolution data can be found in various fields including epidemiology and econometrics. The key characteristic of multi-resolution data is that the response variable/s are at a lower resolution to the covariate/s.

Examples of multi-resoltion data include an outcome of daily deaths of respiratory disease with covariates recorded hourly or an outcome may be the number of deaths in a country with covariates at 5kmx5km resolution. Another, less intuitive example of multi-resolution data is that of a series of lagged variables. For example, daily growth in stock prices, the main covariate is the lagged growth rate. Instead of treating each lag as separate variables they can be treated as a single variable where one response, stock growth on day X, has Y lags associated (i.e. more than one covariate value for the same response).

Currently, users of multi-resolution data have to seek out separate R packages depending on each use case. For example, with spatial data you could use the `disaggregation` package, for econometric data there is `MIDAS` and for time-series data with lags there is `dlnm`. These packages provide complex statistical models and **agouti** does not replace these packages but rather is a more generalised package that can help users with more varied types of multi-resolution data in the preliminary stages of analyses. **agouti** provides generic functions for multi-resolution data that help format, summarise, visualise and carry out simple exploratory analyses.

Data visualisation is a difficult problem with aggregated outputs. Even standard plots like scatter plots do not work because there are many covariate values for each response value. Simple plots, like using the aggregate output value for all covariate values don't make much sense. There are two particular problems that the **agouti** package visualizations address. Firstly, the methods are scalable to hundreds or thousands of aggregate output data points. Secondly, the methods can help to detect the importance of non-linearity.

### Key functions and how to use this vignette

The key functions of **agouti** are `as_disag`, `agouti_summary`, plotting and `agoutiGLM`. The function `as_disag()` takes multi-resolution data with various structures formats it and makes it of class `disag`. The second feature is to summarise, `agouti_summary()` takes an object of class `disag` and provides a summary of all high-resolution variables. Visualisation is the third feature. **agouti** provides four plotting functions to investigate high-resolution covariates against a low-resolution response. The final feature is to run a simple analysis with `agoutiGLM`.

This vignette will demonstrate examples of using the **agouti** package with multi-resolution data of different formats. We advise readers work through the first example irrespective of the data structure in their own dataset before proceeding to the example most similar to their data.

NOTE: This package introduces some novel visualization tools for multi-resolution data. We believe this is an important avenue of research but admit we have not fully got to grips with their interpretations. Please feel free to message us with any thoughts.

## Examples:

1. You think you have a single tidy dataframe
2. Timeseries data with a lagged covariate
3. Spatial data including shapefiles and rasters

### Example 1: You think you have a single tidy dataframe

In this example, you think your data is already tidy. You have your high resolution covariate/s, low resolution response and an ID variable linking high and low resolution data in a single dataframe.

An example, data set in this format is provided with **agouti**. The `madagascar_malaria` data has low resolution response variables of cases and case rate of malaria in different districts of Madagascar identified by the ID variable:

NOTE: the low-resolution response variable/s is/are repeated for all rows with the same ID. This is purely for practical reasons and the response data within an ID should NOT be summed.

```
data(madagascar_malaria)
knitr::kable(head(madagascar_malaria), digits=2)
```

| ID | cellid | cases | case_rate | pop | agg_pop | Elevation | EVI | LSTmean | longitude | latitude |
|---|---|---|---|---|---|---|---|---|---|---|
| 10101918 | 42639 | 1827.16 | 0.01 | 340.92 | 345186.3 | 0.50 | -1.34 | -1.86 | 48.60 | -17.40 |
| 10101918 | 42641 | 1827.16 | 0.01 | 911.81 | 345186.3 | 0.58 | 0.26 | -0.89 | 48.69 | -17.40 |
| 10101918 | 42643 | 1827.16 | 0.01 | 2200.74 | 345186.3 | 0.52 | 1.67 | -1.59 | 48.77 | -17.40 |
| 10101918 | 42644 | 1827.16 | 0.01 | 2044.40 | 345186.3 | 0.45 | 1.64 | -1.59 | 48.81 | -17.40 |
| 10101918 | 42878 | 1827.16 | 0.01 | 74.33 | 345186.3 | 0.51 | -2.13 | -1.95 | 48.56 | -17.44 |
| 10101918 | 42879 | 1827.16 | 0.01 | 1056.74 | 345186.3 | 0.56 | -0.36 | -0.85 | 48.60 | -17.44 |

This dataset contains the following variables:

- `ID` an identifying value that is at the level of the response (i.e. district)
- `cellid` an identifying value at the resolution of the covariates
- `cases` and `case_rate` these are our response variables and within the same ID all response values are the same
- `pop` population size at the resolution of cellid
- `agg_pop` population aggregated to the resolution of ID
- `Elevation`:`LSTmean` potential covariates (Elevation, Enhanced vegetation index and Land surface temperature mean)
- `longitude`:`latitude` other spatial identifying information

**Format** To check if this is in the correct format and convert it to the correct format if not, use the function `as_disag()`.

In this example, the data was already in the correct format so the returned object `disag_data` is in the same format as `madagascar_malaria` but the class of the resulting dataframe has changed.

Now we have an object of class `disag` we can summarise, plot and run a simple analysis.

```
disag_data <- agouti::as_disag(madagascar_malaria, response_var="case_rate")
class(madagascar_malaria)
#> [1] "tbl_df"     "tbl"        "data.frame"
class(disag_data)
#> [1] "disag"      "tbl_df"     "tbl"        "data.frame"
knitr::kable(head(disag_data), digits=2)
```

| ID | cellid | cases | case_rate | pop | agg_pop | Elevation | EVI | LSTmean | longitude | latitude |
|---|---|---|---|---|---|---|---|---|---|---|
| 10101918 | 42639 | 1827.16 | 0.01 | 340.92 | 345186.3 | 0.50 | -1.34 | -1.86 | 48.60 | -17.40 |
| 10101918 | 42641 | 1827.16 | 0.01 | 911.81 | 345186.3 | 0.58 | 0.26 | -0.89 | 48.69 | -17.40 |
| 10101918 | 42643 | 1827.16 | 0.01 | 2200.74 | 345186.3 | 0.52 | 1.67 | -1.59 | 48.77 | -17.40 |
| 10101918 | 42644 | 1827.16 | 0.01 | 2044.40 | 345186.3 | 0.45 | 1.64 | -1.59 | 48.81 | -17.40 |
| 10101918 | 42878 | 1827.16 | 0.01 | 74.33 | 345186.3 | 0.51 | -2.13 | -1.95 | 48.56 | -17.44 |
| 10101918 | 42879 | 1827.16 | 0.01 | 1056.74 | 345186.3 | 0.56 | -0.36 | -0.85 | 48.60 | -17.44 |

**Summarise** The function `agouti_summary()` provides a summary of the variables that are at a higher resolution than the response. Because the raw data was already in the correct format you can use this raw data in `agouti_summary` however, a warning will be printed to encourage you to first use `as_disag()`.

```
agouti::agouti_summary(disag_data)
#> Adding missing grouping variables: `ID`
#> Adding missing grouping variables: `ID`
#> [1] "Aggregate outputs table with ID column: ID"
#> [1] "28892 total rows in 109 groups."
#> [1] "Min group size: 1. Median group size: 230. Max group size: 867."
#> # A tibble: 3 x 8
#>   type              cellid   pop Elevation    EVI LSTmean longitude latitude
#>   <chr>              <dbl> <dbl>     <dbl>  <dbl>   <dbl>     <dbl>    <dbl>
#> 1 Median of medians  53150  657.    -0.197 -0.269  -0.453      47.2    -19.2
#> 2 Median of mins     51951  124.    -0.984 -1.07   -1.38       47.1    -19.5
#> 3 Median of maxs     54862 6655.     1.58   1.46    0.510      47.5    -19.0
```

This summary shows the number of rows in our dataset. This represents the resolution of the covariates. These covariates fall within 109 *groups* delineated the ID variable. There are a range of group sizes from a group with only a single row of covariates to a group with 867 rows of covariates

The default in this function summarises all variables that are not aggregated, calculated the median within each group and then presents the median of these medians. Alternatively you can specify which variables to summarise with the argument `high_res`. For example, to summarise just the population and EVI variables:

```
agouti::agouti_summary(disag_data, high_res=c("pop","EVI"))
#> Adding missing grouping variables: `ID`
#> Adding missing grouping variables: `ID`
#> [1] "Aggregate outputs table with ID column: ID"
```

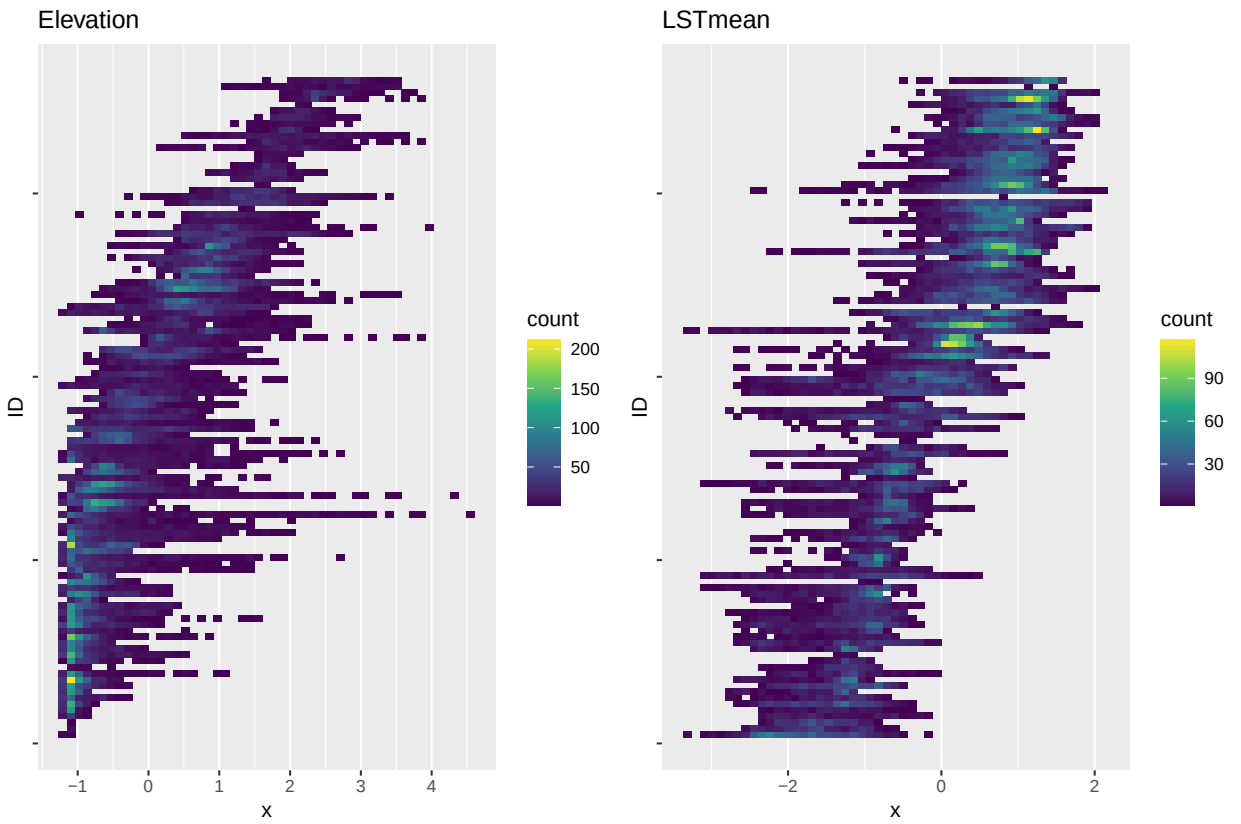
```
#> [1] "28892 total rows in 109 groups."
#> [1] "Min group size: 1. Median group size: 230. Max group size: 867."
#> # A tibble: 3 x 3
#>   type                pop    EVI
#>   <chr>             <dbl>  <dbl>
#> 1 Median of medians  657. -0.269
#> 2 Median of mins     124. -1.07
#> 3 Median of maxs    6655.  1.46
```

**Visualise** The **agouti** package provides four plotting functions.

**1) `heat_hist()`** First of all we might be interested to look at the coviarates in isolation. The function `heat_hist()` plots a histogram grouped by ID on the y axis ordered by the median covariate value within each group.

Use `heat_hist` to look in turn at the climatic covariates in the madagascar_malaria data set:

```
p1<- agouti::heat_hist(disag_data$Elevation, disag_data$ID)+ ggplot2::ggtitle("Elevation")
p2<- agouti::heat_hist(disag_data$LSTmean, disag_data$ID)+ ggplot2::ggtitle("LSTmean")
gridExtra::grid.arrange(p1,p2,ncol=2)
```

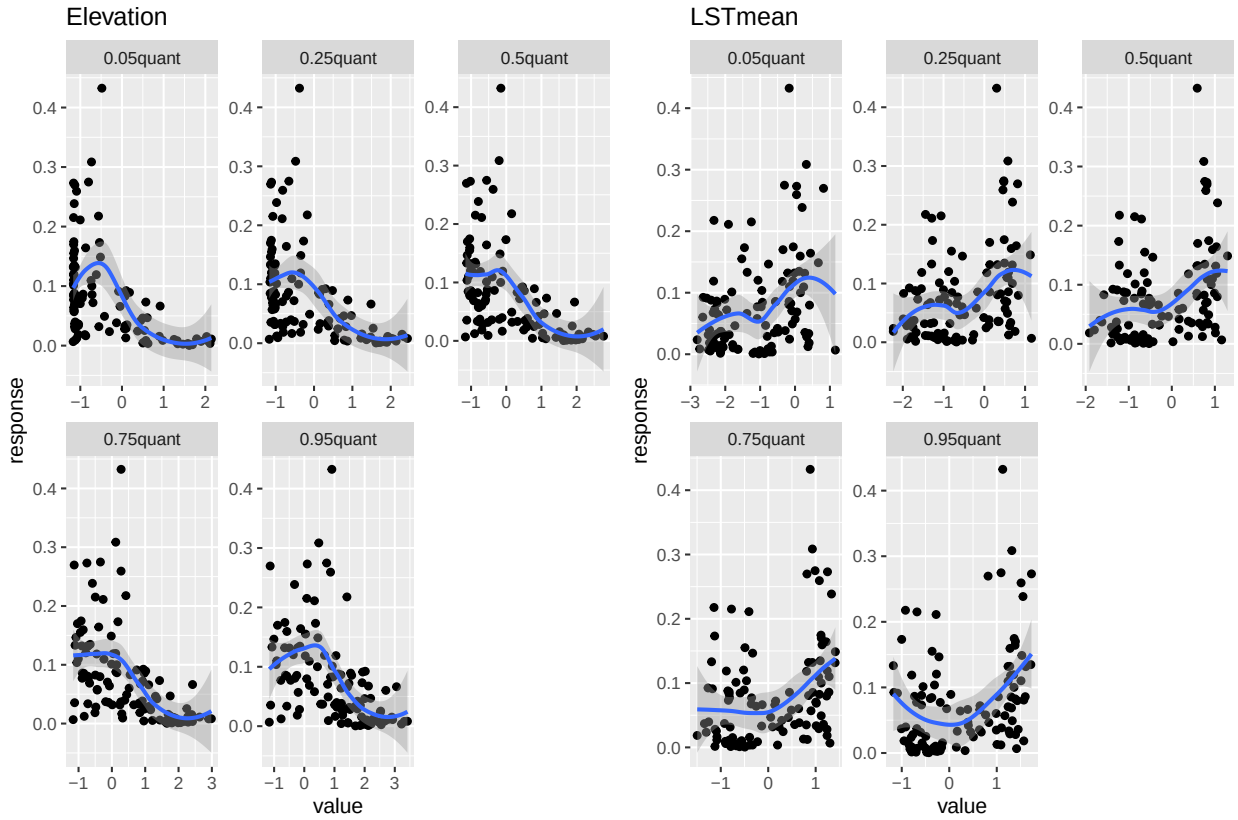



**2) `group_summary_plot()`** `heat_hist` looked at the covariates in isolation but we might want to visualize the covariates against the response. This is not straightforward for multi-resolution data.

The next three types of plot offer different but complimentary approaches to exploratory analyses of multi-resolution data. Not all plots may be appropriate for a given data set but we hope these novel plots aid users of multi-resolution data in exploratory tasks such as looking at the distribution of data and exploring central tendency as well as outliers.

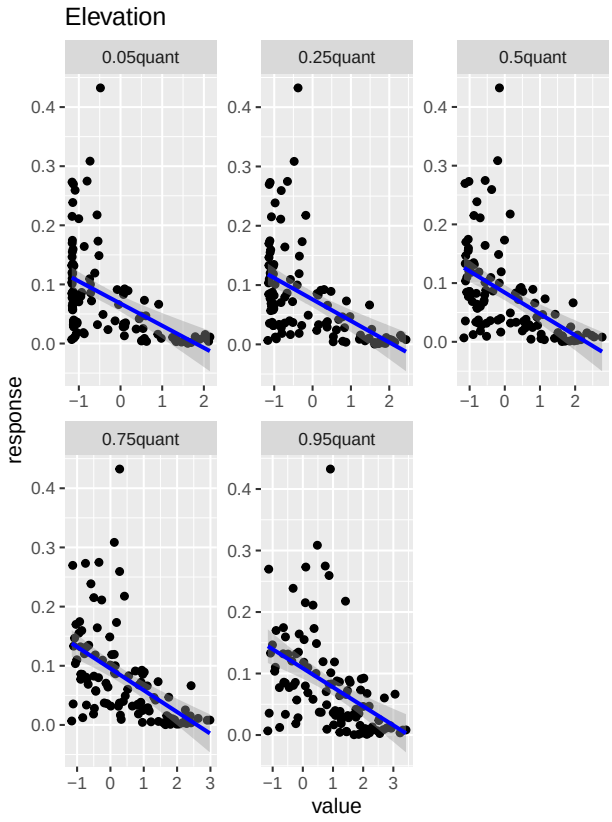
`group_summary_plot()` summarises the covariates within a group (ID) by quantile and then plots these quantiles against the response data with a trend line. Another way of explaining this, is that within each facet each group (ID) is displayed by a single point. In the 0.05quant facet, the points represent the 5th quantile of the covariate values within each group (ID)

```
p1 <- agouti::group_summary_plot(case_rate ~ Elevation, data = disag_data, ID = ID, weights = pop)+
  ggplot2::ggtitle("Elevation")
p2 <- agouti::group_summary_plot(case_rate ~ LSTmean, data = disag_data, ID = ID, weights = pop)+
  ggplot2::ggtitle("LSTmean")
gridExtra::grid.arrange(p1,p2,ncol=2)
#> `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
#> `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




These are novel plots and so we are still working on how to interpret these. Below is our current interpretation of the presented example.

In this data set, these plots show an association between each covariate and the response albeit in different directions. Similar trends in the data across all quantiles are seen for Elevation and for LST mean similar trends in the data are seen across the 5th, 25th and 50th quantiles with a slight change in the trend after that possibly suggesting that at the upper extremes of temperature there is a different association between mean land surface temperature and malaria case rate.



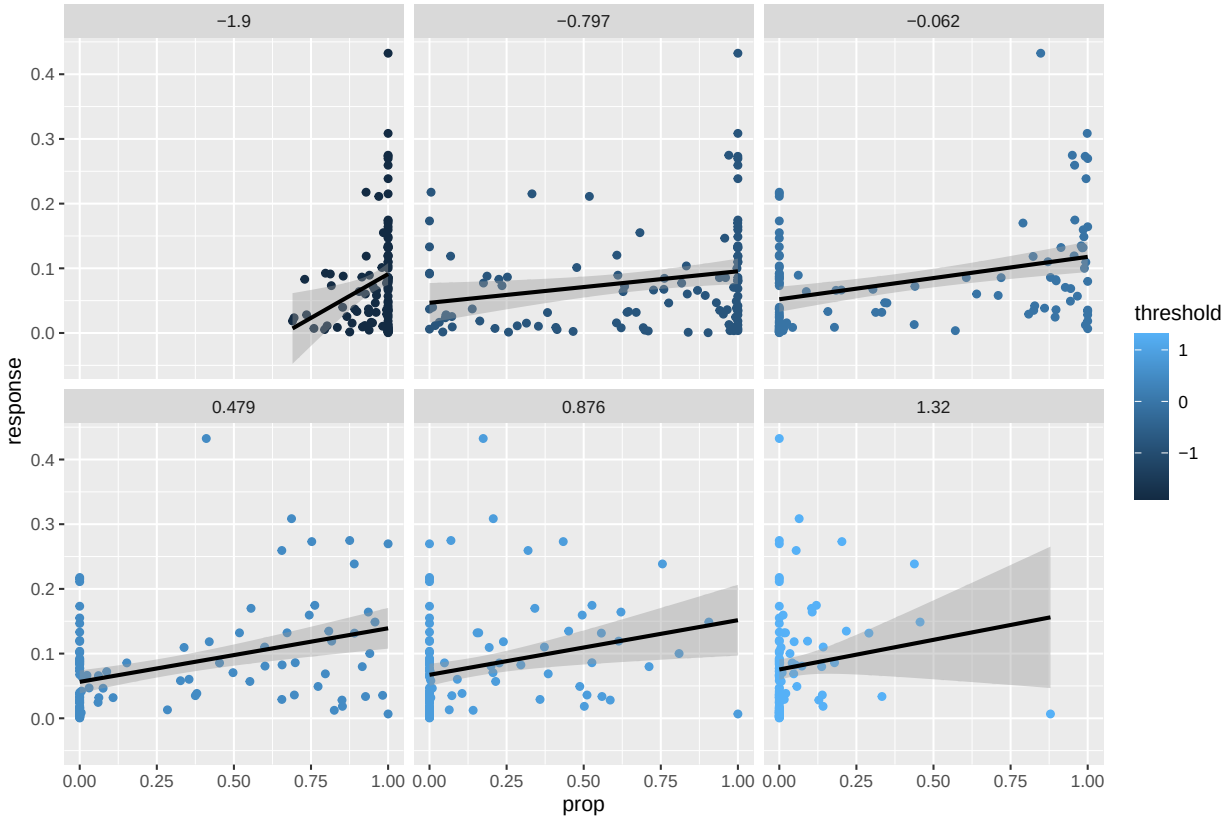

Now look at these plotted with a linear model instead of just using loess.

**3) `thresh_sm()`** An alternative way of visualizing the covariates and response is via small multiples with the function `thresh_sm()`. Small multiples refers to a series of graphs that "use the same basic graphic to display different slices of a data set" (ref).

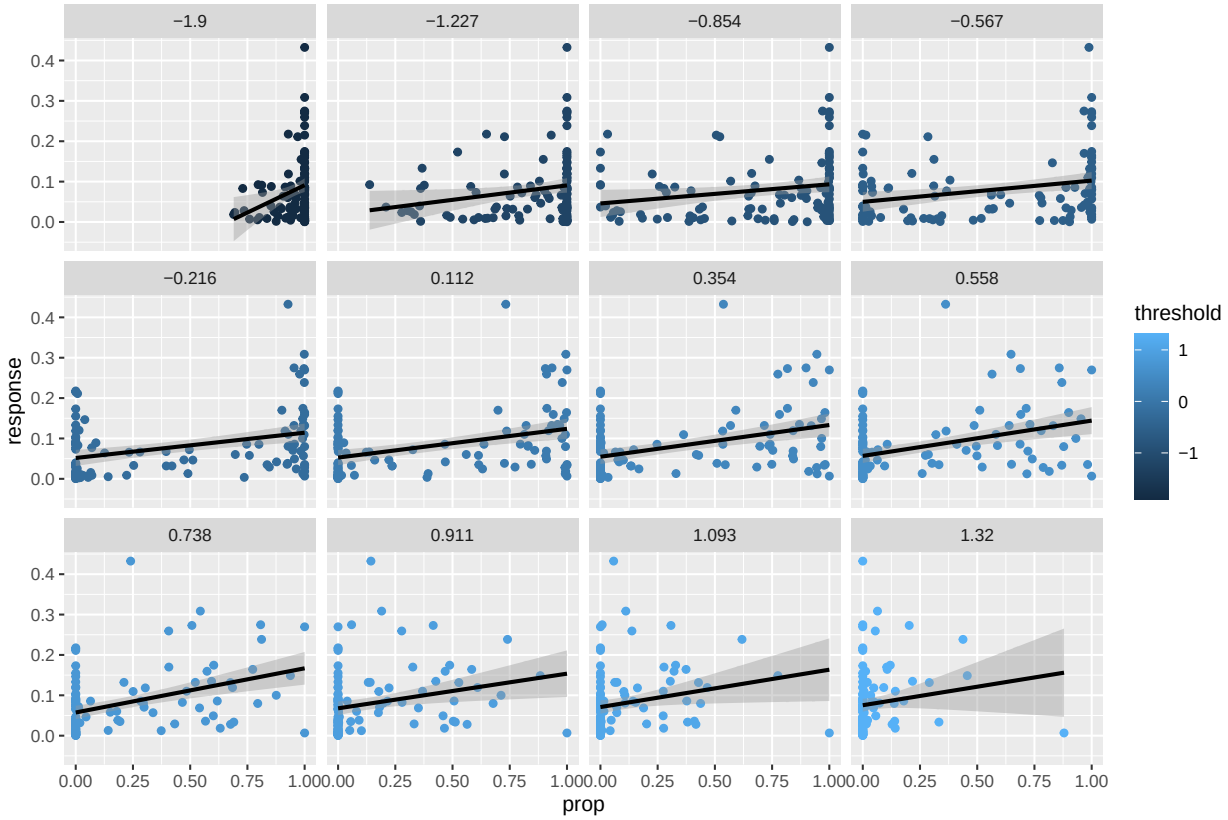
The plot is created by calculating, within each group (ID), the proportion of observations that are above a threshold value. A scatter plot with a linear model is then plotted of the relationship between this proportion and the aggregated output. The threshold is varied and small multiples of the different values are plotted.

The number of multiples can be selected by the argument *small_mult*, the default is six but we advise trying multiple values for this argument as the most appropriate value will vary with different data. The lowest and uppermost threshold are set using the lower_percentile and upper_percentile arguments and default to 5th and 9th percent quantiles. The number of thresholds considered between these two values is determined by the small_mult argument.

```
agouti::thresh_sm(case_rate ~ LSTmean, data = disag_data, ID = ID, weights = pop, small_mult = 6)
#> `geom_smooth()` using formula = 'y ~ x'
```

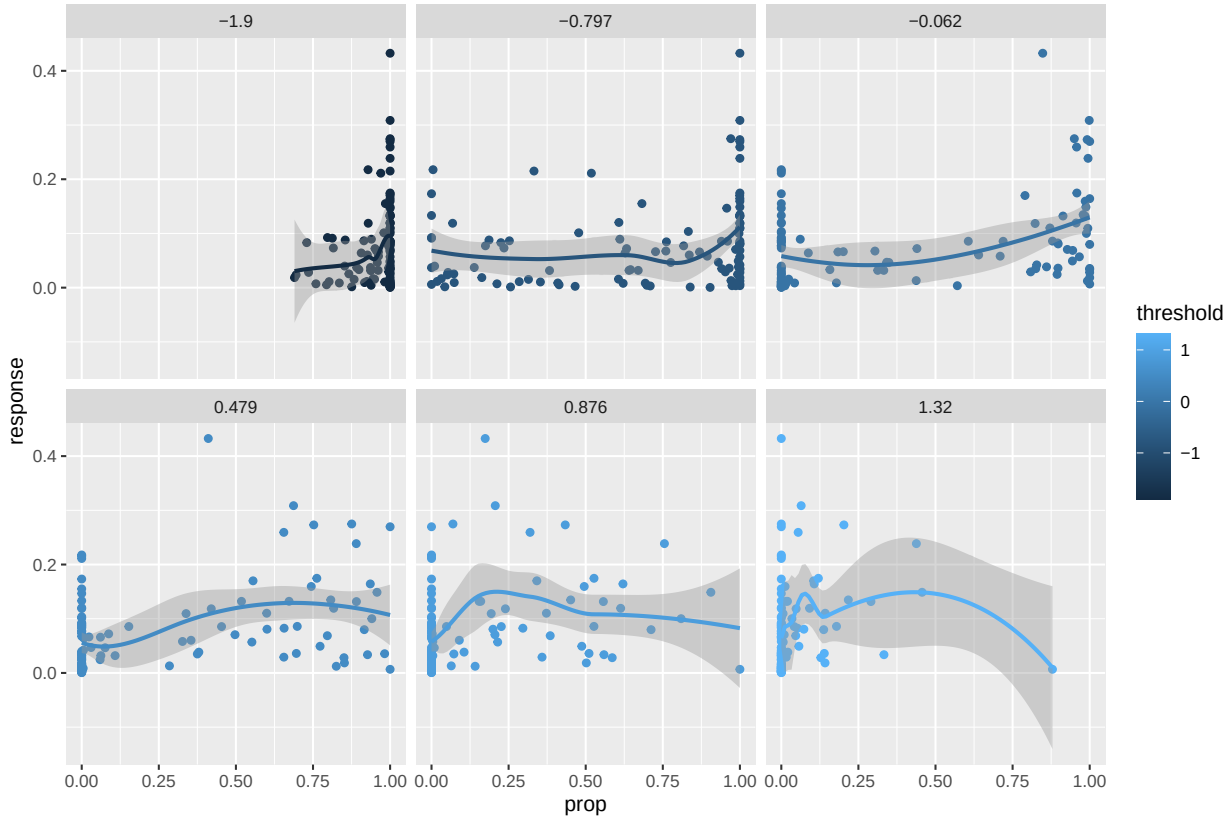



```
agouti::thresh_sm(case_rate ~ LSTmean, data = disag_data, ID = ID, weights = pop, small_mult = 12)
#> `geom_smooth()` using formula = 'y ~ x'
```




Irrespective of the number of small multiples used, these figures show that overall there appears to be a positive relationship between malaria case rate and LSTmean.

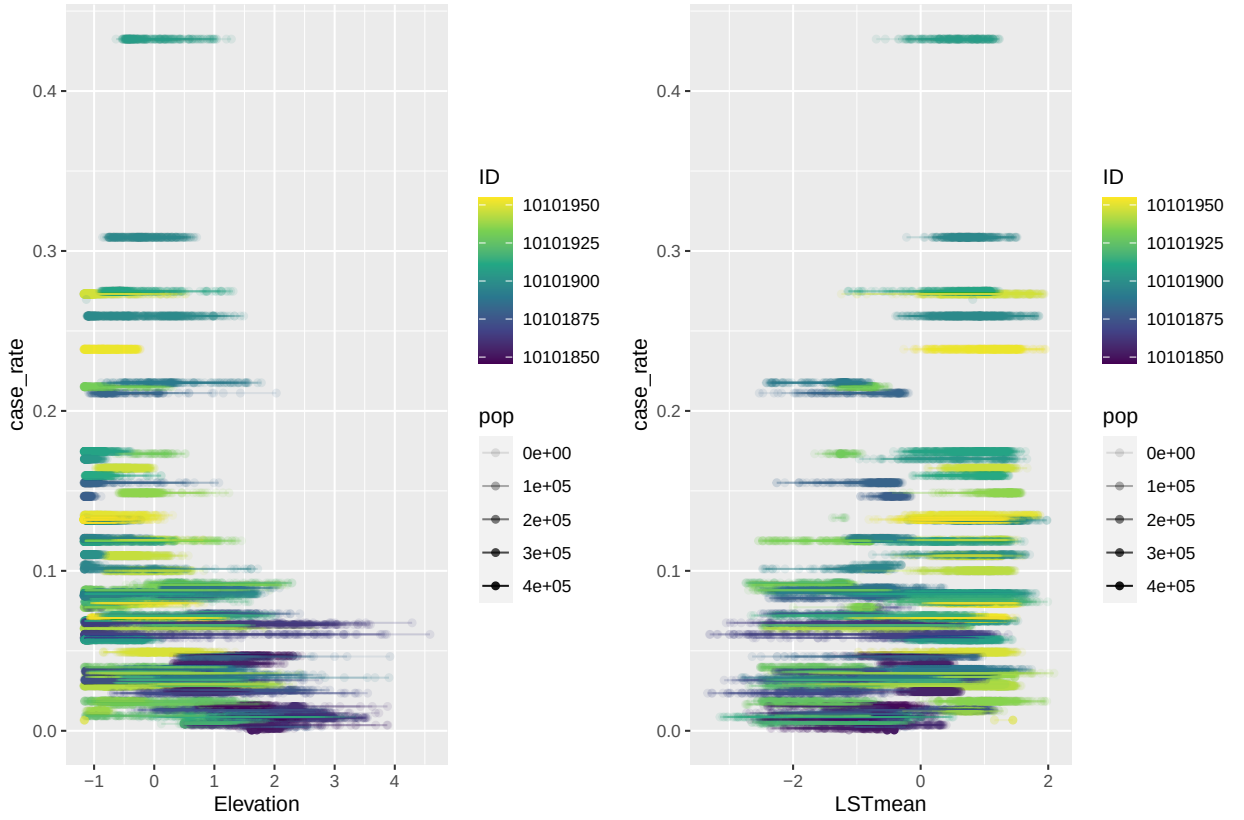
Now have a look at this plot if we use loess instead of a linear model (The default is a linear model).




**4) `link_plot()`** The final plotting function available is `link_plot()`. This is a scatter plot with data from the aggregate unit "linked" by a line and colour coded to match. As discussed in the introduction scatter plots are not necessarily the best way to plot multi-resolution data and this plot will not be appropriate for all data sets.

Link plot could help to elucidate whether there is a larger range of covariate values for those with a higher or lower response value but with lots of data and lots of aggregate units this plot could also very quickly become too cluttered to interpret.

```
p1 <- agouti::link_plot(case_rate ~ Elevation, data = disag_data, ID = ID, weights = pop)
p2 <- agouti::link_plot(case_rate ~ LSTmean, data = disag_data, ID = ID, weights = pop)
gridExtra::grid.arrange(p1,p2,ncol=2)
```




**Model** As preliminary analyses, a simple glm can be fitted using `agoutiGLM`. Below is the function to fit a glm once this function is ready to be used.

```
# agouti::agoutiGLM(case_rate ~ LSTmean, data = disag_data, ID = disag_data$ID,weights = disag_data$pop
```

### Example 2: Timeseries data with a lagged covariate

In this example, your data is a timeseries of class *ts*. An example included with the agouti package, is of daily stock growth

```
data("stock_vector")
knitr::kable(head(stock_vector), digits=6)
```

| x |
|---|
| -0.014641 |
| -0.011214 |
| 0.010774 |
| 0.003647 |
| 0.007826 |
| -0.001941 |

```
class(stock_vector)
#> [1] "ts"
```

This data set contains a single variable of class *ts*, representing daily stock growth. Daily stock growth is a function of the previous days stock growth as well as the day before that. So we can create a lagged growth variable that represents our predictor. Where, each lag considered represents a different value for the predictor all corresponding to the same outcome value.

**Format** This dataset can be transformed to the required format using the `as_disag()` function and specifying the number of lags to be considered as follows:

```
disag_ts <- agouti::as_disag(stock_vector, lags=5)
#> [1] "adding ID col"
class(disag_ts)
#> [1] "disag"      "data.frame"
knitr::kable(head(disag_ts[order(disag_ts$ID),],10), digits=6, row.names=FALSE)
```

| response | ID | lag | covariate |
|---|---|---|---|
| -0.014641 | 1 | lag1 | NA |
| -0.014641 | 1 | lag2 | NA |
| -0.014641 | 1 | lag3 | NA |
| -0.014641 | 1 | lag4 | NA |
| -0.014641 | 1 | lag5 | NA |
| -0.011214 | 2 | lag1 | -0.014641 |
| -0.011214 | 2 | lag2 | NA |
| -0.011214 | 2 | lag3 | NA |
| -0.011214 | 2 | lag4 | NA |
| -0.011214 | 2 | lag5 | NA |

```
knitr::kable(tail(disag_ts[order(disag_ts$ID),],10), digits=6, row.names=FALSE)
```

| response | ID | lag | covariate |
|---|---|---|---|
| -0.044012 | 4569 | lag1 | -0.018090 |
| -0.044012 | 4569 | lag2 | -0.020187 |
| -0.044012 | 4569 | lag3 | -0.039716 |
| -0.044012 | 4569 | lag4 | 0.010511 |
| -0.044012 | 4569 | lag5 | 0.001435 |
| -0.079764 | 4570 | lag1 | -0.044012 |
| -0.079764 | 4570 | lag2 | -0.018090 |
| -0.079764 | 4570 | lag3 | -0.020187 |
| -0.079764 | 4570 | lag4 | -0.039716 |
| -0.079764 | 4570 | lag5 | 0.010511 |

**Summarise** Using the function `agouti_summary()` we can summarise the variables that are at a higher resolution than the outcome.

```
agouti::agouti_summary(disag_ts,high_res="covariate")
#> Adding missing grouping variables: `ID`
#> Adding missing grouping variables: `ID`
#> [1] "Aggregate outputs table with ID column: ID"
#> [1] "22850 total rows in 4570 groups."
#> [1] "Min group size: 5. Median group size: 5. Max group size: 5."
#> # A tibble: 3 x 2
#>   type              covariate
#>   <chr>                 <dbl>
#> 1 Median of medians        NA
#> 2 Median of mins           NA
#> 3 Median of maxs           NA
```

This summary shows the number of rows in our dataset which is at the high resolution of the covariate. The covaraite falls within 4570 groups delineated by the ID variable and which represents the number of rows in the original dataset i.e. the low resolution outcome variable.

The table summarises all variables that are not aggregated, calculated the median within each group and then presents the median of these medians.

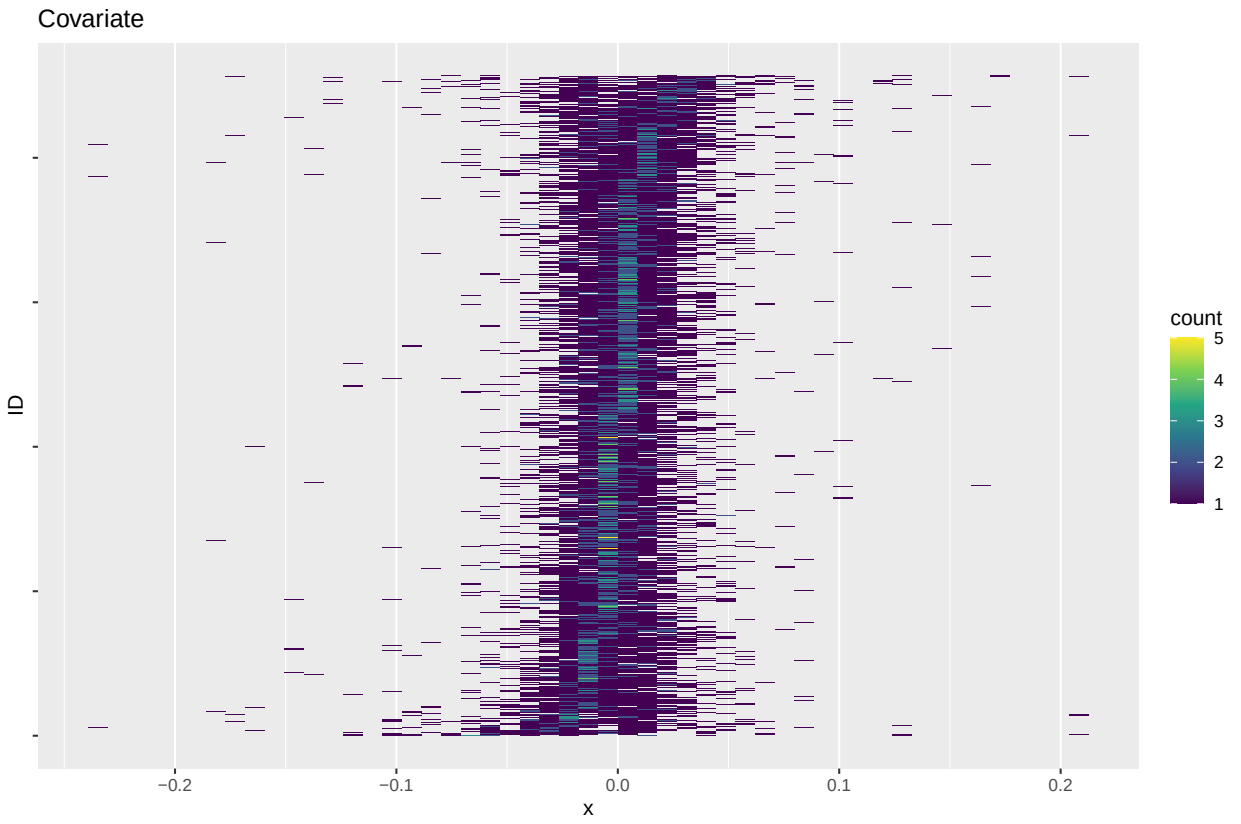
In this dataset, because we created 10 lags for every outcome value the first 10 outcomes contain NAs for some lags. As a result, the lagged growth variable summary has NAs. To consider the summary without the rows containing NA, add the argument $na.rm = TRUE$.

```
agouti::agouti_summary(disag_ts,high_res="covariate",na.rm=TRUE)
#> Adding missing grouping variables: `ID`
#> Adding missing grouping variables: `ID`
#> [1] "Aggregate outputs table with ID column: ID"
#> [1] "22835 total rows in 4569 groups."
#> [1] "Min group size: 1. Median group size: 5. Max group size: 5."
#> # A tibble: 3 x 2
#>   type              covariate
#>   <chr>                 <dbl>
#> 1 Median of medians  0.000532
#> 2 Median of mins    -0.0177
#> 3 Median of maxs     0.0186
```

### Visualise

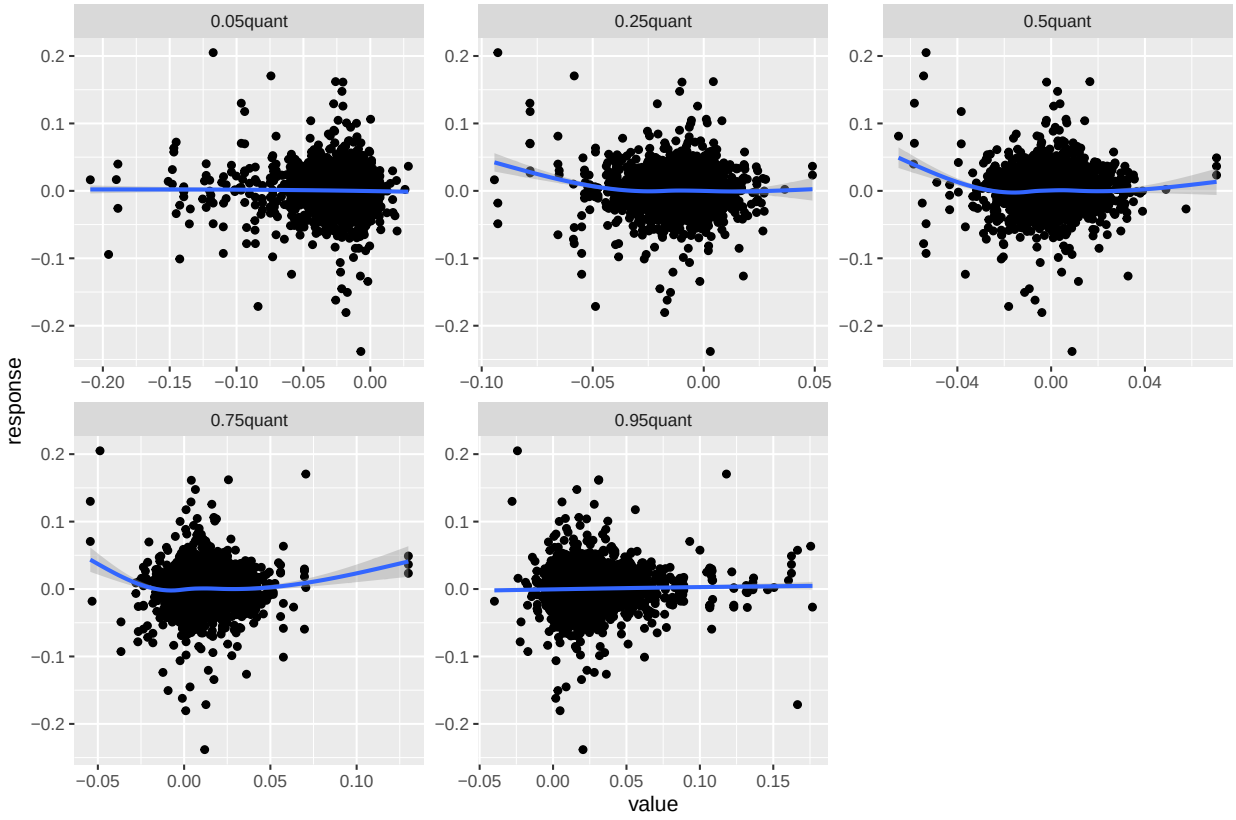
**1) `heat_hist()`** This data set contains a single covariate. The function `heat_hist()` plots a histogram grouped by ID on the y axis ordered by the median predictor (x) within each group.

```
agouti::heat_hist(disag_ts$covariate, disag_ts$ID)+ ggplot2::ggtitle("Covariate")
#> Warning: `fct_reorder()` removing 15 missing values.
#> i Use `.na_rm = TRUE` to silence this message.
#> i Use `.na_rm = FALSE` to preserve NAs.
#> Warning: Removed 15 rows containing non-finite values (`stat_bin2d()`).
```

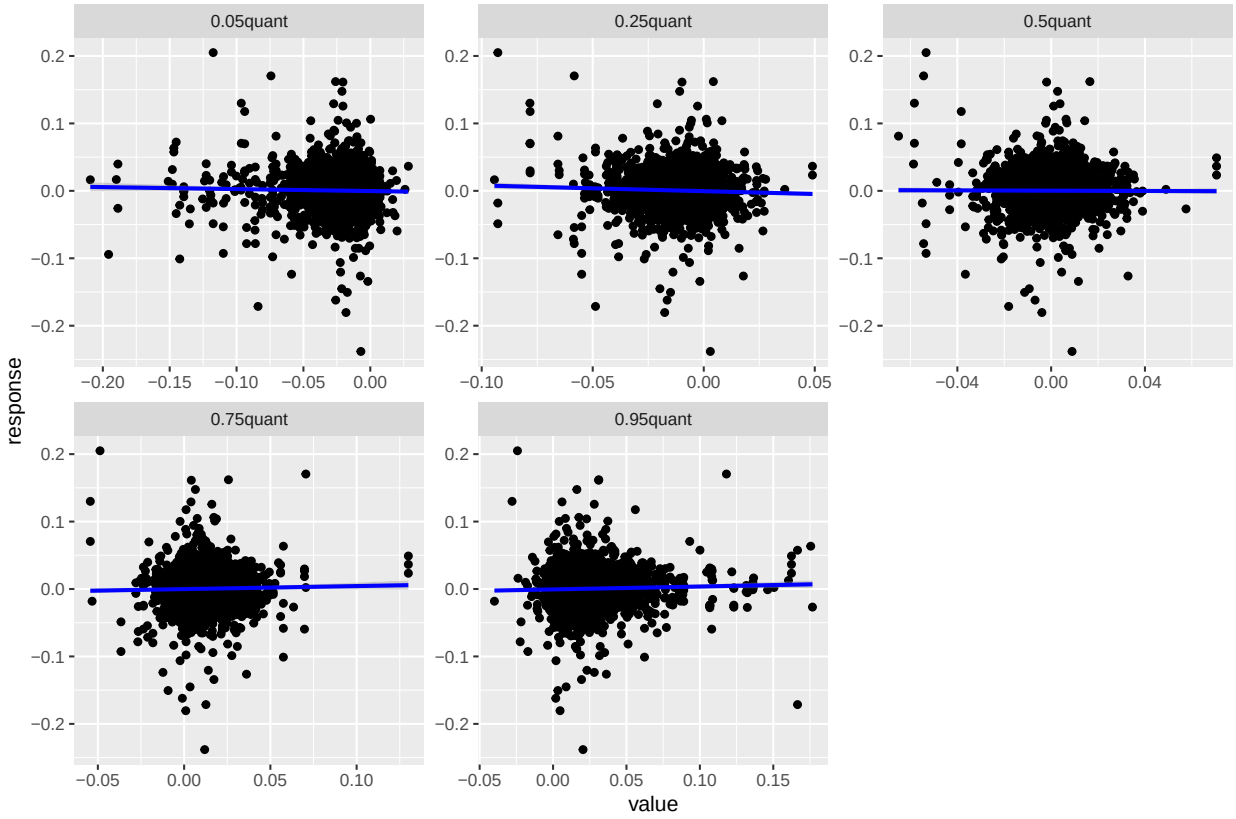



**2) `group_summary_plot()`** To plot the predictor against the response we can use `group_summary_plot()`. This summarises the covariate/s by quantile and then plots these quantiles against the response data with a trend line.

```
agouti::group_summary_plot(response ~ covariate, data = disag_ts, ID = ID)
#> `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



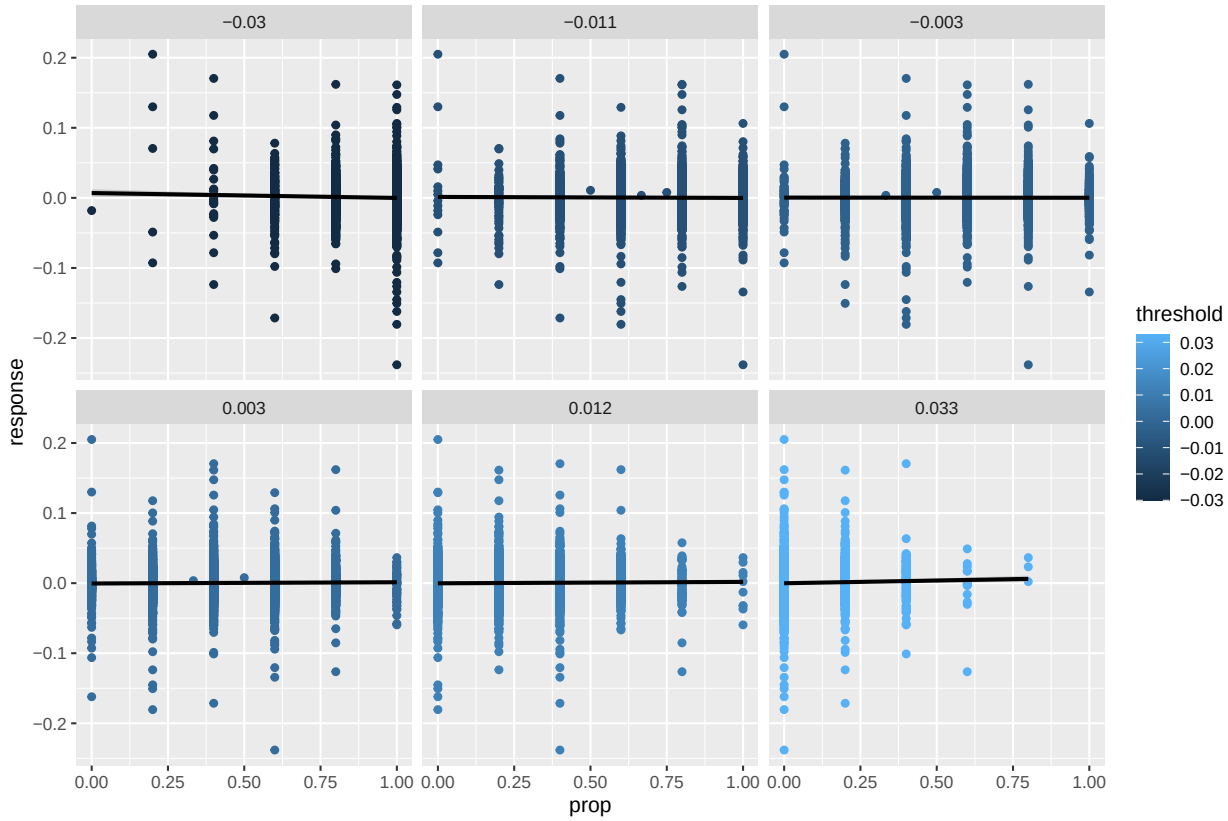

And now with a linear model instead of trend line.



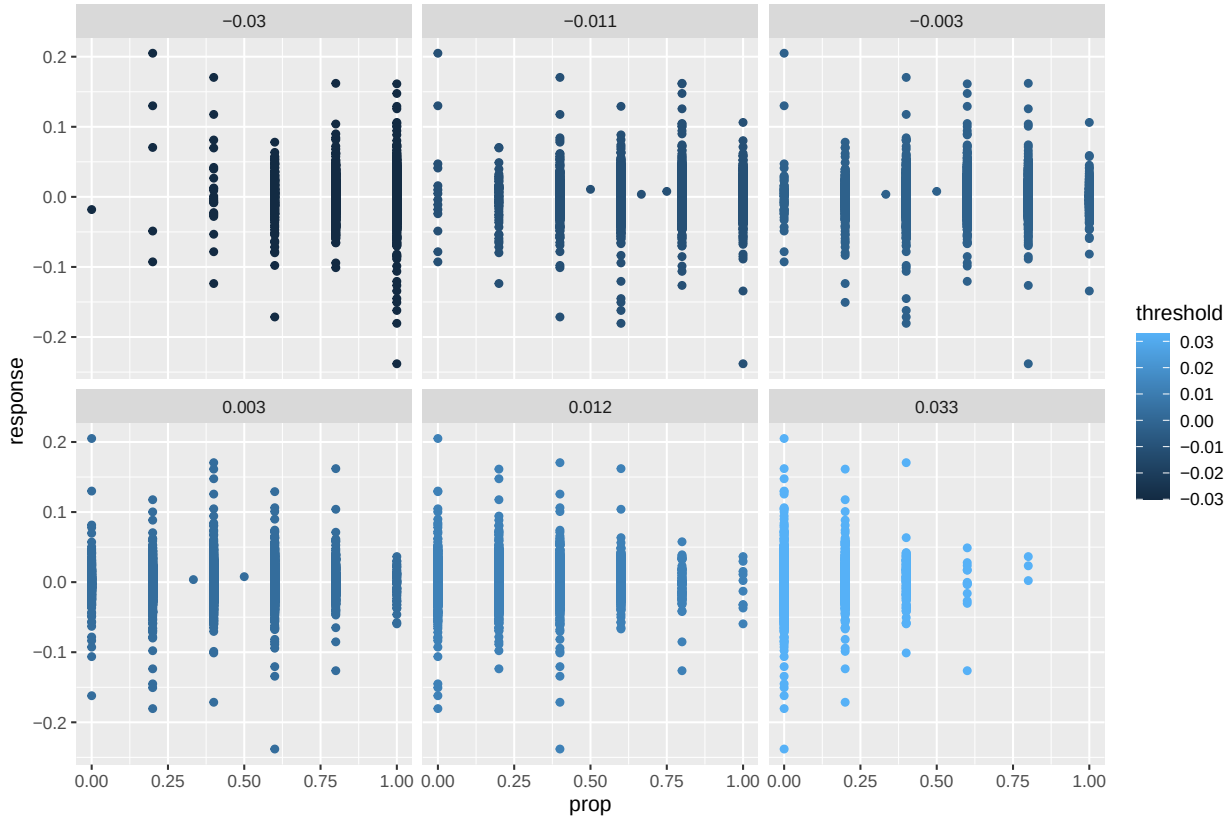
**3) `thresh_sm()`** An alternative way of visualising the covariates and response is via the function `thresh_sm()`. The plot is created by calculating, within each group, the proportion of observations that are above a threshold. A scatter plot with a linear model is plotted of the relationship between this proportion and the aggregated output. The threshold is varied and small multiples of the different values are plotted. The number of multiples can be selected by the argument *small_mult*, the default is six.

The stock data has some NA values, any rows with NA will be removed for plotting

```
agouti::thresh_sm(response ~ covariate, data = disag_ts, ID = ID, small_mult=6)
#> Warning in agouti::thresh_sm(response ~ covariate, data = disag_ts, ID = ID, : Rows with NA have bee
#> `geom_smooth()` using formula = 'y ~ x'
```



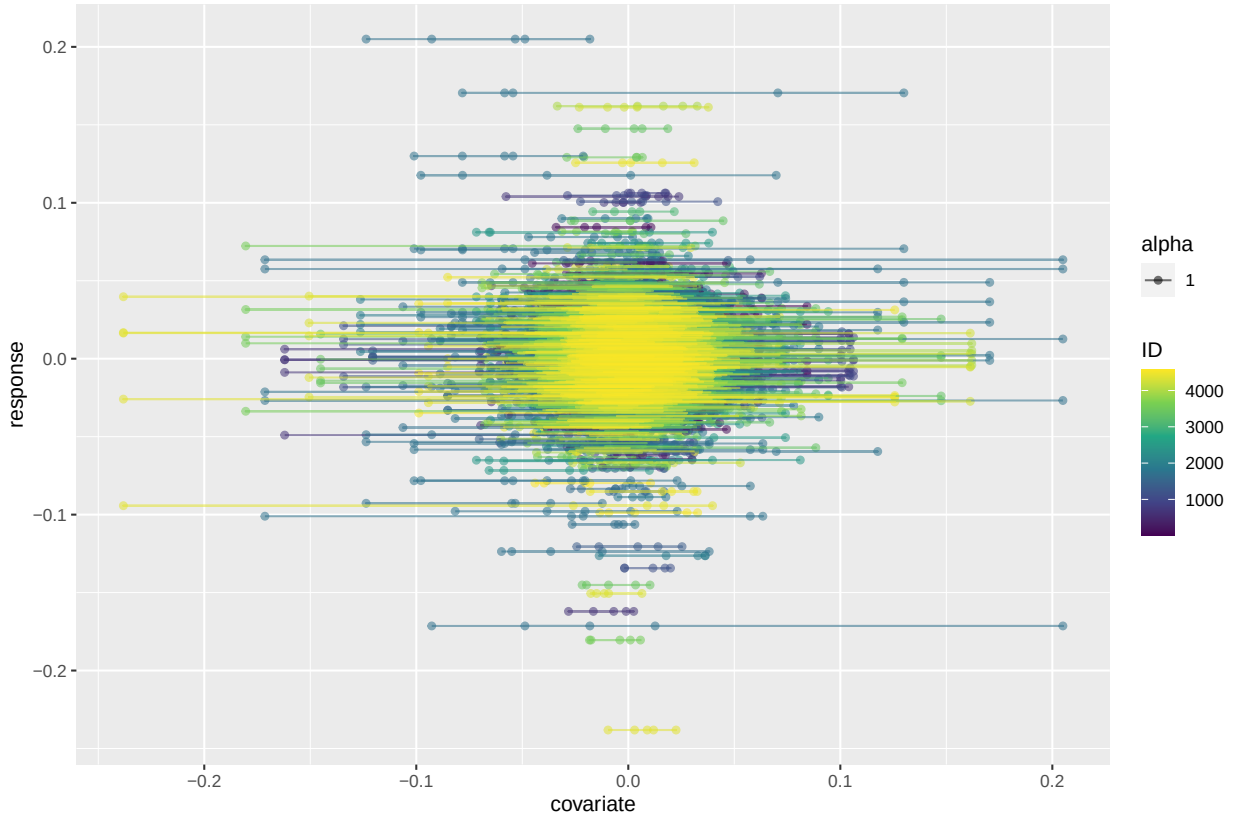

And now with a trend line instead of linear model. This fails to run geom_smooth for some facets. "Error: Compuation failed in stat_smooth(). Caused by error in 'smooth.construct.cr.smooth.spec()' x has insufficent unique values to support 10 knots, reduce k."




**4) `link_plot()`** The final plotting function available is `link_plot()`. This is a scatter plot with data from the aggregate unit "linked" by a line and colour coded to match.

In this example, link plot does not really add anything to our exploratory analyses of this data.

```
agouti::link_plot(response ~ covariate, data = disag_ts, ID = ID, weights = 1)
#> Warning: Removed 15 rows containing missing values (`geom_point()`).
#> Warning: Removed 15 rows containing missing values (`geom_path()`).
```



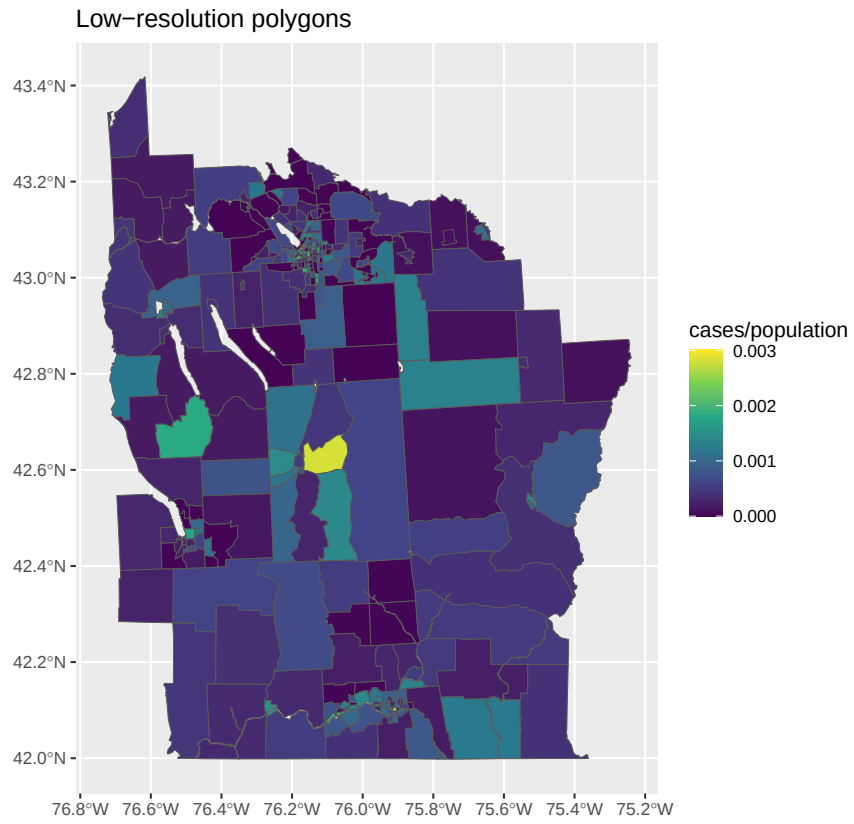

**Model**

**Example 3: Spatial data including shapefiles and rasters**

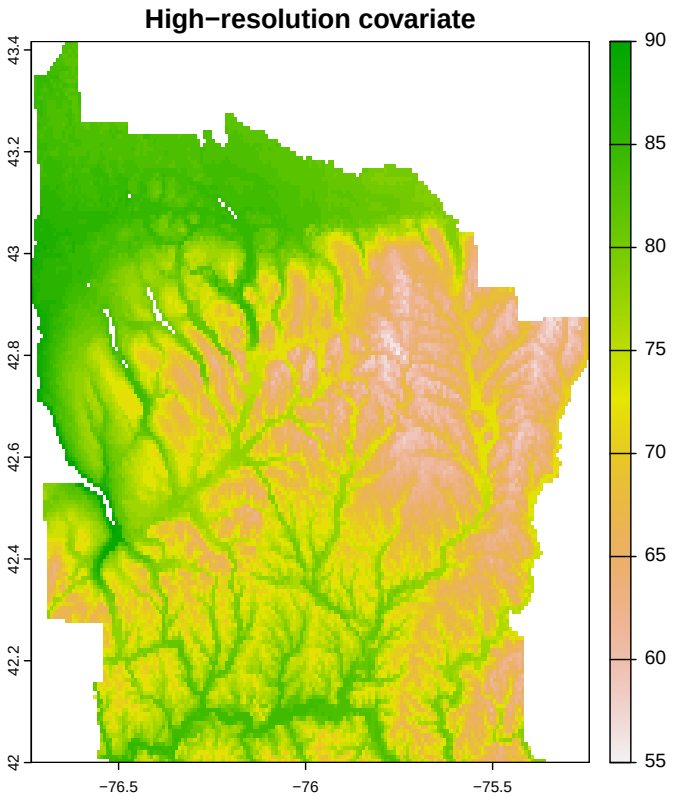
In this example you have a response variable that relates to polygons (often geographical administrative units such as counties in the UK). Your covariates are high-resolution rasters and you first need to extract the relevant information from your rasters along with the identifying polygon information.

Here we have two datasets 1) A shapefile (using data from the package `SpatialEpi` containing the polygons that are at the resolution of the response variable. 2) A raster or raster stack of high-resolution covariates. These raster covariates contain many pixels within a single polygon and each pixel has a value associated

```
# get polygon data from SpatialEpi pacakge
polygons <- sf::st_as_sf(NYleukemia$spatial.polygon)
df <- cbind(polygons, NYleukemia$data)
ggplot2::ggplot() + ggplot2::geom_sf(data = df, aes(fill = cases / population)) +
  ggplot2::scale_fill_viridis_c(lim = c(0, 0.003))+ggtitle("Low-resolution polygons")
```




```
# get raster data (using annual mean temp from bioclim cropped to the extent of the polygons)
covariate <- terra::rast("annual_mean_temp_newyork.tif")
covariate_clip <- raster::crop(covariate, raster::extent(polygons))
covariate_clip <- raster::mask(covariate_clip, polygons)
plot(covariate_clip, main="High-resolution covariate")
```




**Format** This dataset can be transformed to the required format using the `as_disag()` function:

```
names(df)[1] <-"ID"
disag_sf <- agouti::as_disag(data=df, rstack=covariate, response_var="cases")
class(disag_sf)
#> [1] "disag"      "data.frame"
knitr::kable(tail(disag_sf), digits=2, row.names = FALSE)
```

| annual_mean_temp_newyork | cell | ID | cases | population | geometry |
|---|---|---|---|---|---|
| 78 | 7565225 | 36109992300 | 2.02 | 2029 | MULTIPOLYGON (((-76.4903 42... |
| 78 | 7565226 | 36109992300 | 2.02 | 2029 | MULTIPOLYGON (((-76.4903 42... |
| 81 | 7568823 | 36109992300 | 2.02 | 2029 | MULTIPOLYGON (((-76.4903 42... |
| 79 | 7568824 | 36109992300 | 2.02 | 2029 | MULTIPOLYGON (((-76.4903 42... |
| 78 | 7568825 | 36109992300 | 2.02 | 2029 | MULTIPOLYGON (((-76.4903 42... |
| 78 | 7568826 | 36109992300 | 2.02 | 2029 | MULTIPOLYGON (((-76.4903 42... |

**Summarise** Using the function `agouti_summary()` we can summarise the variables that are at a higher resolution than the response. In this example, some groups are of size 1 so we need to specify the names of the covariates.

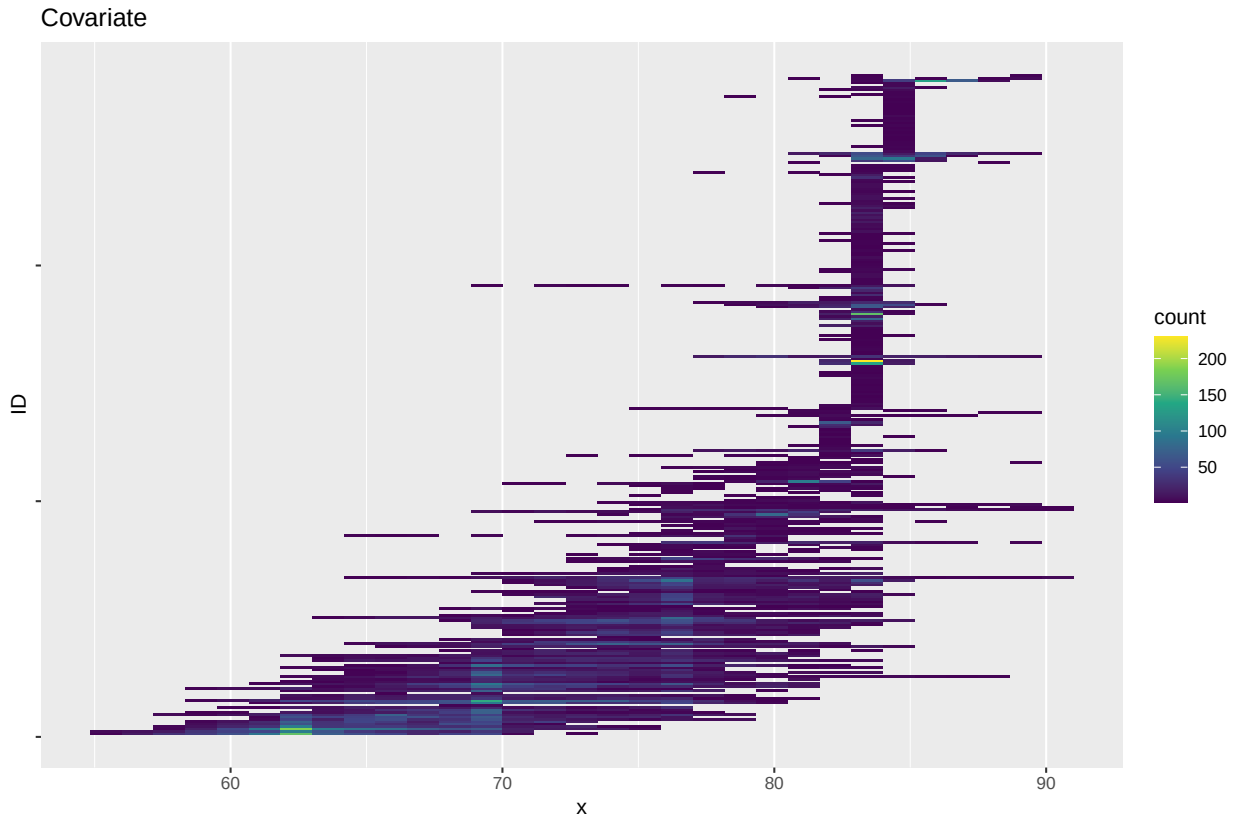
```
agouti::agouti_summary(disag_sf,high_res="annual_mean_temp_newyork")
#> Adding missing grouping variables: `ID`
#> Adding missing grouping variables: `ID`
#> [1] "Aggregate outputs table with ID column: ID"
#> [1] "21460 total rows in 281 groups."
#> [1] "Min group size: 1. Median group size: 9. Max group size: 882."
#> # A tibble: 3 x 2
#>   type              annual_mean_temp_newyork
#>   <chr>                                <dbl>
#> 1 Median of medians                     82.5
#> 2 Median of mins                        82
#> 3 Median of maxs                        84
```

This summary shows the number of rows in our dataset. The covaraite falls within 281 groups delineated by the ID variable and which represents the number of rows in the original dataset i.e. the low resolution response variable. The group sizes are varied in this dataset but the annual mean temperature is not so varied.

### Visualise

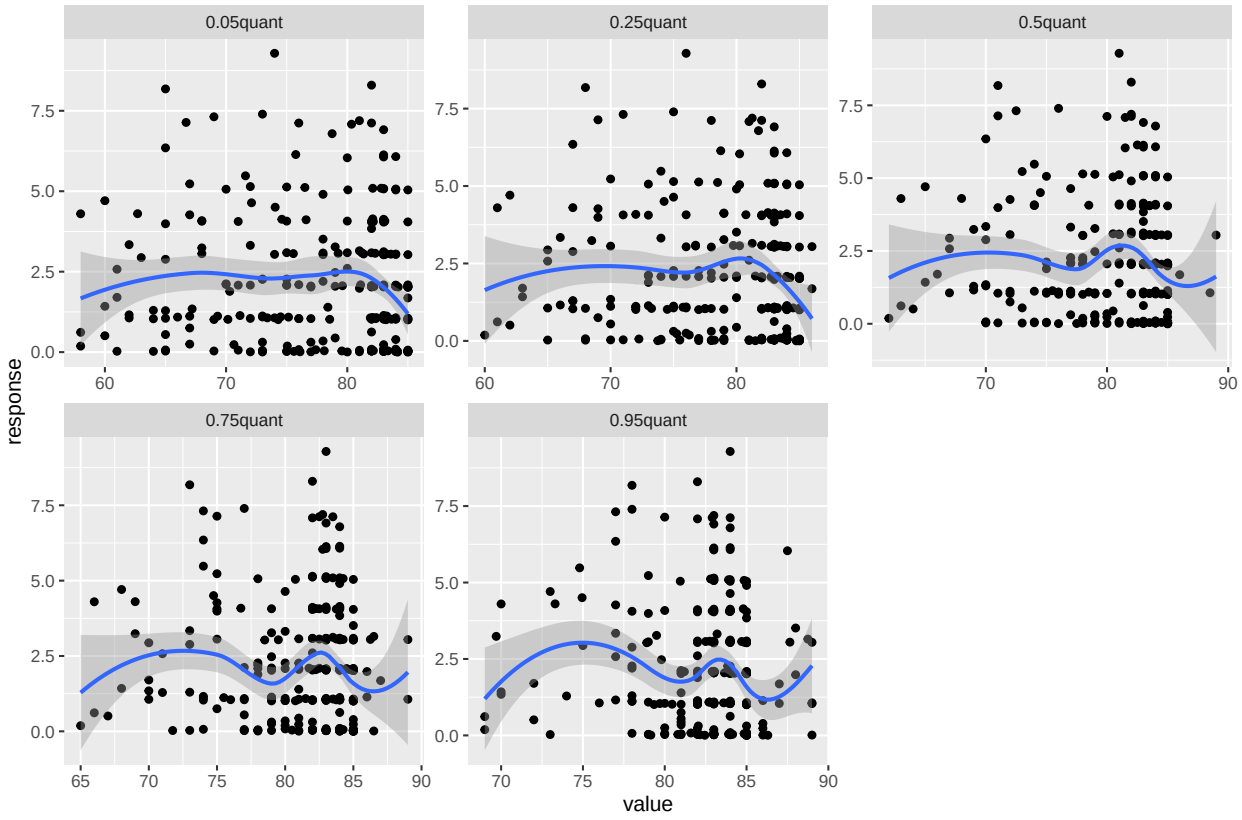
**1) `heat_hist()`** This data set contains a single covariate. The function `heat_hist()` plots a histogram grouped by ID on the y axis ordered by the median predictor (x) within each group.

```
agouti::heat_hist(disag_sf$annual_mean_temp_newyork, disag_sf$ID, breaks=30)+
  ggplot2::ggtitle("Covariate")
```

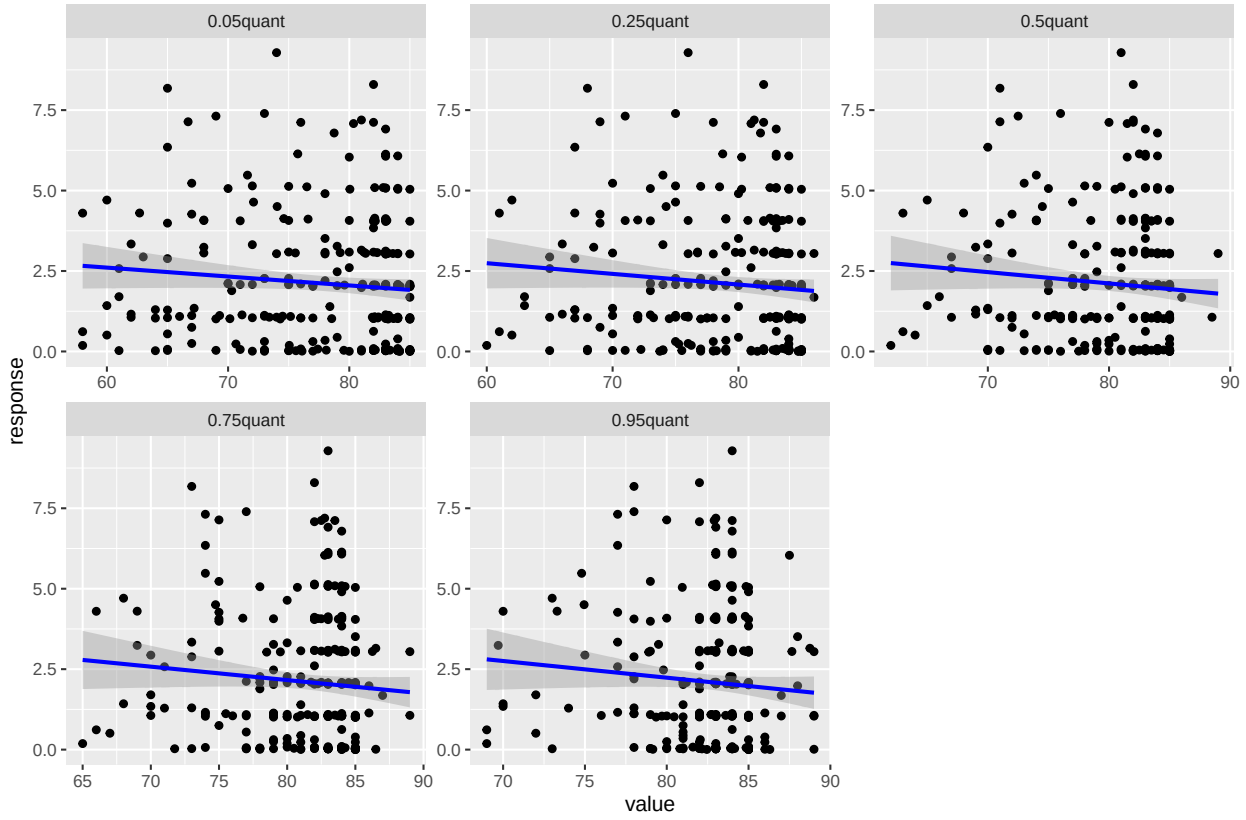


**2) `group_summary_plot()`** To plot the covariate against the response we can use `group_summary_plot()`. This summarises the covariate/s by quantile and then plots these quantiles against the response data.

```
agouti::group_summary_plot(cases ~ annual_mean_temp_newyork, data = disag_sf, ID = ID)
#> `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

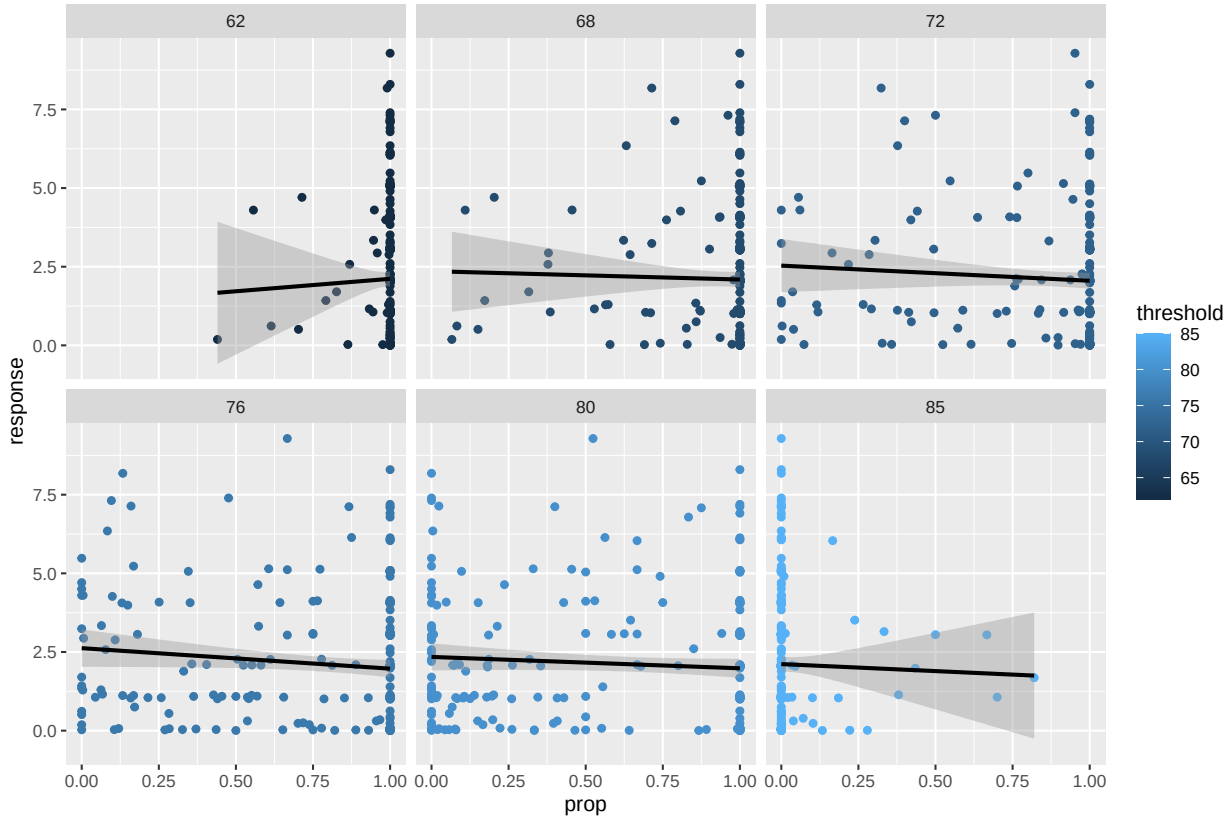



And now with a linear model instead of trend line.

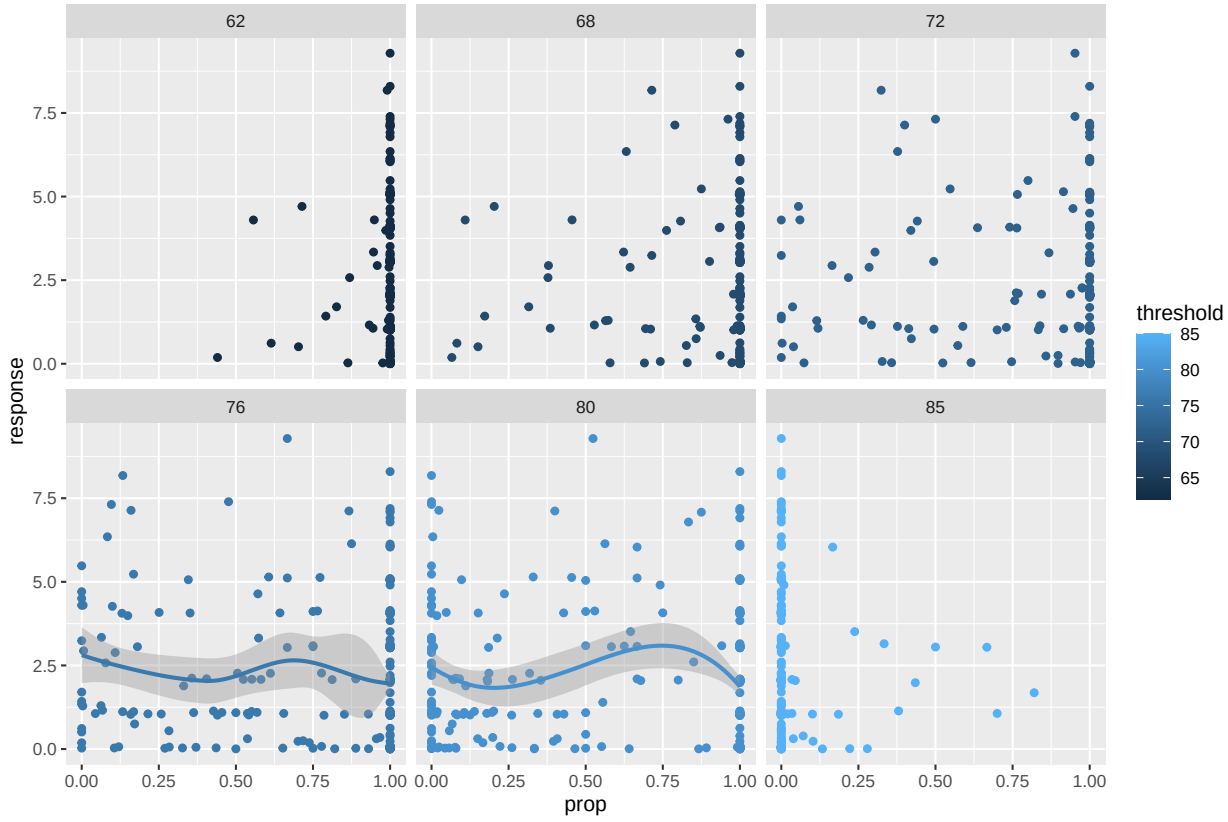



**3) `thresh_sm()`** An alternative way of visualising the covariates and response is via the function `thresh_sm()`. The plot is created by calculating, within each group, the proportion of observations that are above a threshold. A scatter plot with a linear model is plotted of the relationship between this proportion and the aggregated output. The threshold is varied and small multiples of the different values are plotted. The number of multiples can be selected by the argument *small_mult*, the default is six.

```
agouti::thresh_sm(cases ~ annual_mean_temp_newyork, data = disag_sf, ID = ID)
#> `geom_smooth()` using formula = 'y ~ x'
```

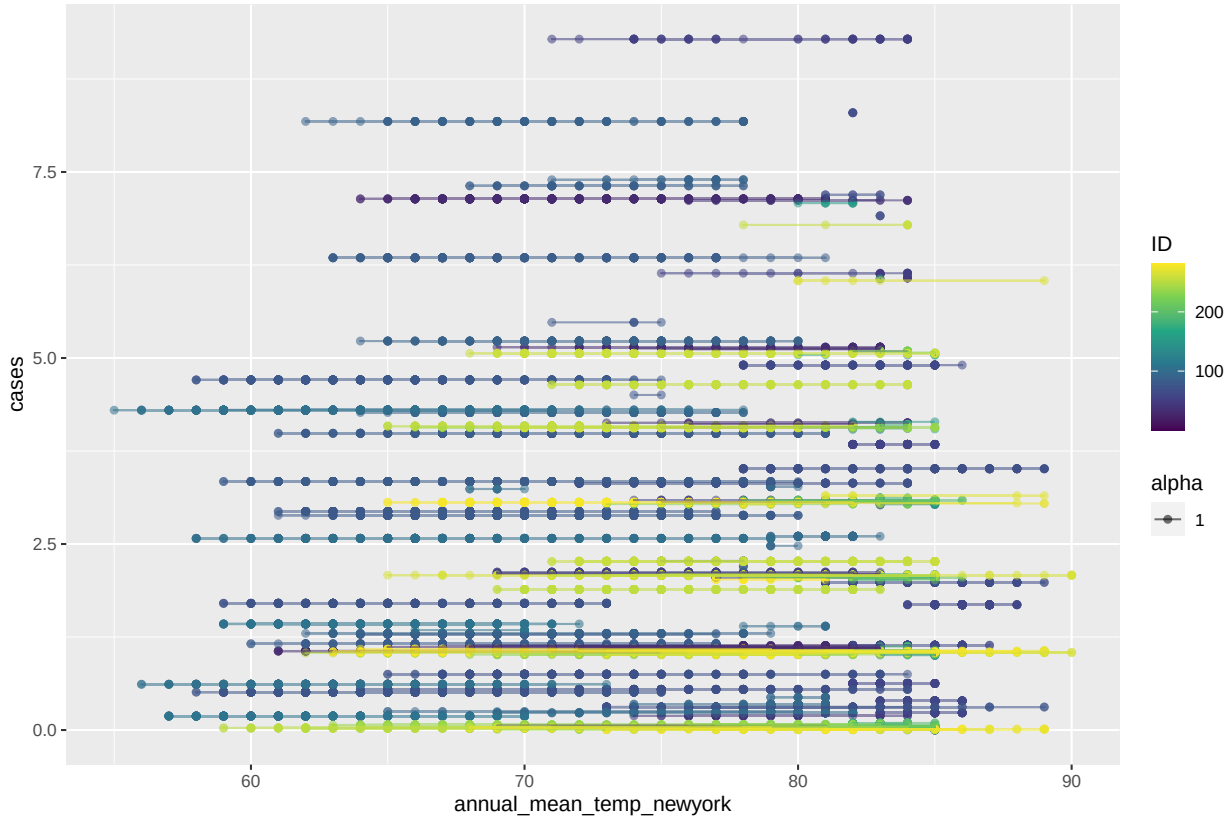



And now with a trend line instead of linear model. And now with a trend line instead of linear model. This fails to run geom_smooth for some facets. "Error: Compuation failed in stat_smooth(). Caused by error in 'smooth.construct.cr.smooth.spec()' x has insufficent unique values to support 10 knots, reduce k."




**4) `link_plot()`** The final plotting function available is `link_plot()`. This is a scatter plot with data from the aggregate unit "linked" by a line and colour coded to match.

```
agouti::link_plot(cases ~ annual_mean_temp_newyork, data = disag_sf, ID = ID, weights = 1)
```


cases
7.5
5.0
2.5
0.0
60
70
80
90
annual_mean_temp_newyork
ID
200
100
alpha
1

**Model**