Ridits `right`, `left`, `center`, native and foreign

Roger B. Newson
roger.newson@kcl.ac.uk
http://www.rogernewsonresources.org.uk

Cancer Prevention Group, King's College London

Presented at the 2021 UK Stata Conference, 9–10 September, 2021
Downloadable from the conference website at
http://ideas.repec.org/s/boc/usug21.html

What are right, left and center ridits?

- Given a random variable X, the **left ridit function** $L_X(\cdot)$ of X is defined by the formula

$$L_X(u) = \Pr(X < u).$$

- And the **right ridit function** $R_X(\cdot)$ of X (also known as a cumulative distribution function) is defined by the formula

$$R_X(u) = \Pr(X \leq u).$$

- And the **center ridit function** $C_X(\cdot)$ of X is defined by the formula

$$C_X(u) = \frac{1}{2}[L_X(u) + R_X(u)],$$

and is what most people usually mean by a "ridit function".

What are right, left and center ridits?

- Given a random variable X, the **left ridit function** $L_X(\cdot)$ of X is defined by the formula

$$L_X(u) = \Pr(X < u).$$

- And the **right ridit function** $R_X(\cdot)$ of X (also known as a cumulative distribution function) is defined by the formula

$$R_X(u) = \Pr(X \leq u).$$

- And the **center ridit function** $C_X(\cdot)$ of X is defined by the formula

$$C_X(u) = \frac{1}{2}[L_X(u) + R_X(u)],$$

and is what most people usually mean by a “ridit function”.

What are right, left and center ridits?

- Given a random variable X, the **left ridit function** $L_X(\cdot)$ of X is defined by the formula

$$L_X(u) = \Pr(X < u).$$

- And the **right ridit function** $R_X(\cdot)$ of X (also known as a cumulative distribution function) is defined by the formula

$$R_X(u) = \Pr(X \leq u).$$

- And the **center ridit function** $C_X(\cdot)$ of X is defined by the formula

$$C_X(u) = \frac{1}{2}\left[L_X(u) + R_X(u) \right],$$

and is what most people usually mean by a "ridit function".

What are right, left and center ridits?

- Given a random variable X, the **left ridit function** $L_X(\cdot)$ of X is defined by the formula

$$L_X(u) = \Pr(X < u).$$

- And the **right ridit function** $R_X(\cdot)$ of X (also known as a cumulative distribution function) is defined by the formula

$$R_X(u) = \Pr(X \leq u).$$

- And the **center ridit function** $C_X(\cdot)$ of X is defined by the formula

$$C_X(u) = \frac{1}{2}\left[L_X(u) + R_X(u) \right],$$

and is what most people usually mean by a "ridit function".

So what are native and foreign ridits?

- Given a random variable X, the left, right and center **native ridits** of X are the random variables $L_X(X)$, $R_X(X)$, and $C_X(X)$, respectively.
- And, given a second random variable W, possibly with a different distribution from X, the left, right or center **foreign ridits of X with respect to W** are the random variables $L_W(X)$, $R_W(X)$, and $C_W(X)$, respectively.
- In the inaugural article on ridits, Bross (1958)[2] stated that the word ridit was short for "with respect to an identified distribution", by analogy with logits and probits.
- *However*, Bross allegedly stated later that he *really* named them after his wife Rida. . .

So what are native and foreign ridits?

- Given a random variable X, the left, right and center **native ridits** of X are the random variables $L_X(X)$, $R_X(X)$, and $C_X(X)$, respectively.
- And, given a second random variable W, possibly with a different distribution from X, the left, right or center **foreign ridits of** X **with respect to** W are the random variables $L_W(X)$, $R_W(X)$, and $C_W(X)$, respectively.
- In the inaugural article on ridits, Bross (1958)[2] stated that the word ridit was short for "with respect to an identified distribution", by analogy with logits and probits.
- *However*, Bross allegedly stated later that he *really* named them after his wife Rida. . .

So what are native and foreign ridits?

- Given a random variable X, the left, right and center **native ridits** of X are the random variables $L_X(X)$, $R_X(X)$, and $\mathrm{C}_X(X)$, respectively.
- And, given a second random variable W, possibly with a different distribution from X, the left, right or center **foreign ridits of X with respect to W** are the random variables $L_W(X)$, $R_W(X)$, and $C_W(X)$, respectively.
- In the inaugural article on ridits, Bross (1958)[2] stated that the word ridit was short for "with respect to an identified distribution", by analogy with logits and probits.
- *However*, Bross allegedly stated later that he *really* named them after his wife Rida...

So what are native and foreign ridits?

- Given a random variable X, the left, right and center **native ridits** of X are the random variables $L_X(X)$, $R_X(X)$, and $C_X(X)$, respectively.
- And, given a second random variable W, possibly with a different distribution from X, the left, right or center **foreign ridits of** X **with respect to** W are the random variables $L_W(X)$, $R_W(X)$, and $C_W(X)$, respectively.
- In the inaugural article on ridits, Bross (1958)[2] stated that the word ridit was short for "with respect to an identified distribution", by analogy with logits and probits.
- *However*, Bross allegedly stated later that he *really* named them after his wife Rida...

So what are native and foreign ridits?

- Given a random variable X, the left, right and center **native ridits** of X are the random variables $L_X(X)$, $R_X(X)$, and $\mathrm{C}_X(X)$, respectively.
- And, given a second random variable W, possibly with a different distribution from X, the left, right or center **foreign ridits of** X **with respect to** W are the random variables $L_W(X)$, $R_W(X)$, and $C_W(X)$, respectively.
- In the inaugural article on ridits, Bross (1958)[2] stated that the word ridit was short for "with respect to an identified distribution", by analogy with logits and probits.
- *However*, Bross allegedly stated later that he *really* named them after his wife Rida. . .

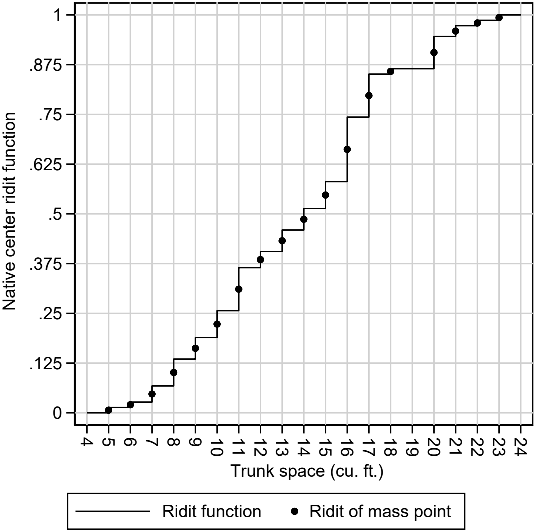
An unfamiliar version of a familiar dataset

The SSC package `xauto` extends the `auto` data:

```
. xauto, clear;

Contains data from C:\Program Files\Stata17\ado\base/a/auto.dta
 Observations:            74                  1978 Automobile Data extended by Roger Newson
    Variables:            17                  13 Apr 2020 17:45
                                              (_dta has notes)
--------------------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
--------------------------------------------------------------------------------------------
foreign         byte    %8.0g      origin     Car origin
make            str17   %-17s                 Make and model
price           int     %8.0gc                Price
mpg             byte    %8.0g                 Mileage (mpg)
rep78           byte    %8.0g                 Repair record 1978
headroom        float   %6.1f                 Headroom (in.)
trunk           byte    %8.0g                 Trunk space (cu. ft.)
weight          int     %8.0gc                Weight (lbs.)
length          int     %8.0g                 Length (in.)
turn            byte    %8.0g                 Turn circle (ft.)
displacement    int     %8.0g                 Displacement (cu. in.)
gear_ratio      float   %6.2f                 Gear ratio
firm            str7    %9s                   Firm
odd             byte    %8.0g      odd        Even or odd sequence number
us              byte    %8.0g      us         US or non-US model
tons            double  %10.0g                Weight (US tons)
npm             double  %10.0g                Fuel consumption (nipperkins/mile)
--------------------------------------------------------------------------------------------
Sorted by: foreign  make
     Note: Dataset has changed since last saved.
```

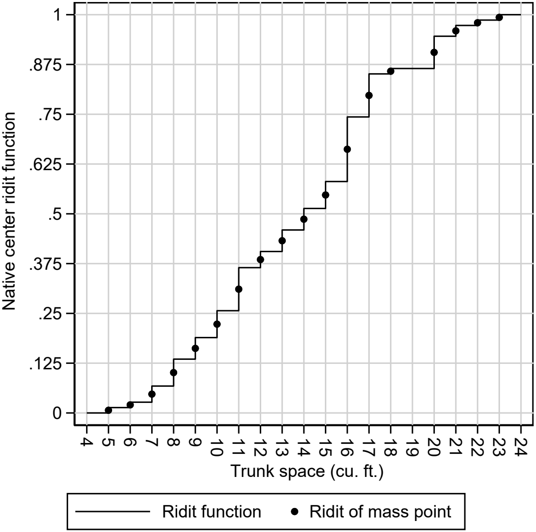
Native center ridit function of variable `trunk` in the `xauto` data

- The horizontal axis gives values of trunk space, from 4 to 24 cubic feet.
- The vertical axis gives the **native center ridit function**, discontinuous at the mass points, and flat elsewhere.
- The left or right ridit functions would be left–continuous or right–continuous, respectively, at the mass points.



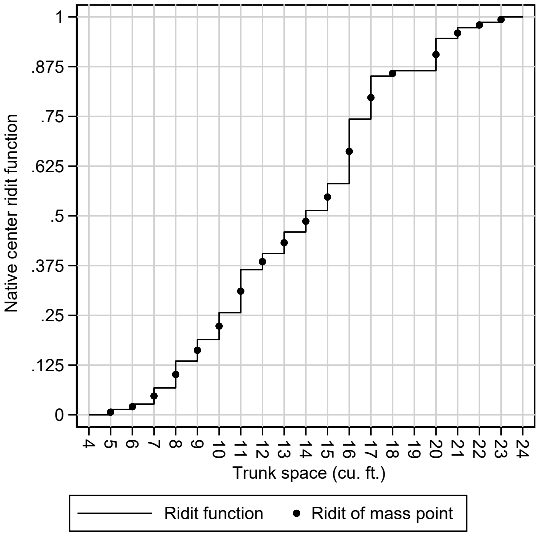
Native center ridit function of variable `trunk` in the `xauto` data

- The horizontal axis gives values of trunk space, from 4 to 24 cubic feet.
- The vertical axis gives the **native center ridit function**, discontinuous at the mass points, and flat elsewhere.
- The left or right ridit functions would be left–continuous or right–continuous, respectively, at the mass points.



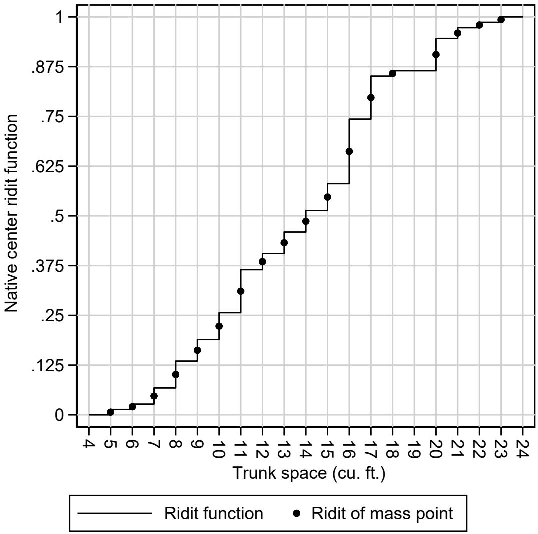
Native center ridit function of variable `trunk` in the `xauto` data

- The horizontal axis gives values of trunk space, from 4 to 24 cubic feet.
- The vertical axis gives the **native center ridit function**, discontinuous at the mass points, and flat elsewhere.
- The left or right ridit functions would be left–continuous or right–continuous, respectively, at the mass points.



Native center ridit function of variable `trunk` in the `xauto` data

- The horizontal axis gives values of trunk space, from 4 to 24 cubic feet.
- The vertical axis gives the **native center ridit function**, discontinuous at the mass points, and flat elsewhere.
- The left or right ridit functions would be left–continuous or right–continuous, respectively, at the mass points.



So what packages on SSC compute ridits?

- The `ridit()` function of Nicholas J. Cox's `egenmore` package computes unweighted native center ridits for a variable.
- The package `wridit` allows the ridits to be weighted and/or folded (transformed to a scale from -1 to 1 instead of from 0 to 1), as recommended by Brockett and Levene (1977)[1].
- It also has a `handedness()` option, with possible values `left`, `right`, and `center` (the default).
- In Stata Version 16, `wridit` has a second module `fridit`, specifying foreign ridits for a variable, with respect to the distribution of a variable of the same name in a second data frame, specified by the `fframe()` option.
- These foreign ridits may be weighted by a variable in the second data frame, using the `weight()` suboption of the `fframe()` option.
- And all these programs have `reverse` and `percent` options.

So what packages on SSC compute ridits?

- The `ridit()` function of Nicholas J. Cox's `egenmore` package computes unweighted native center ridits for a variable.
- The package `wridit` allows the ridits to be weighted and/or folded (transformed to a scale from -1 to 1 instead of from 0 to 1), as recommended by Brockett and Levene (1977)[1].
- It also has a `handedness()` option, with possible values `left`, `right`, and `center` (the default).
- In Stata Version 16, `wridit` has a second module `fridit`, specifying foreign ridits for a variable, with respect to the distribution of a variable of the same name in a second data frame, specified by the `fframe()` option.
- These foreign ridits may be weighted by a variable in the second data frame, using the `weight()` suboption of the `fframe()` option.
- And all these programs have `reverse` and `percent` options.

So what packages on SSC compute ridits?

- The `ridit()` function of Nicholas J. Cox's `egenmore` package computes unweighted native center ridits for a variable.
- The package `wridit` allows the ridits to be weighted and/or folded (transformed to a scale from -1 to 1 instead of from 0 to 1), as recommended by Brockett and Levene (1977)[1].
- It also has a `handedness()` option, with possible values `left`, `right`, and `center` (the default).
- In Stata Version 16, `wridit` has a second module `fridit`, specifying foreign ridits for a variable, with respect to the distribution of a variable of the same name in a second data frame, specified by the `fframe()` option.
- These foreign ridits may be weighted by a variable in the second data frame, using the `weight()` suboption of the `fframe()` option.
- And all these programs have `reverse` and `percent` options.

So what packages on SSC compute ridits?

- The `ridit()` function of Nicholas J. Cox's `egenmore` package computes unweighted native center ridits for a variable.
- The package `wridit` allows the ridits to be weighted and/or folded (transformed to a scale from -1 to 1 instead of from 0 to 1), as recommended by Brockett and Levene (1977)[1].
- It also has a `handedness()` option, with possible values `left`, `right`, and `center` (the default).
- In Stata Version 16, `wridit` has a second module `fridit`, specifying foreign ridits for a variable, with respect to the distribution of a variable of the same name in a second data frame, specified by the `fframe()` option.
- These foreign ridits may be weighted by a variable in the second data frame, using the `weight()` suboption of the `fframe()` option.
- And all these programs have `reverse` and `percent` options.

So what packages on SSC compute ridits?

- The `ridit()` function of Nicholas J. Cox's `egenmore` package computes unweighted native center ridits for a variable.
- The package `wridit` allows the ridits to be weighted and/or folded (transformed to a scale from -1 to 1 instead of from 0 to 1), as recommended by Brockett and Levene (1977)[1].
- It also has a `handedness()` option, with possible values `left`, `right`, and `center` (the default).
- In Stata Version 16, `wridit` has a second module `fridit`, specifying foreign ridits for a variable, with respect to the distribution of a variable of the same name in a second data frame, specified by the `fframe()` option.
- These foreign ridits may be weighted by a variable in the second data frame, using the `weight()` suboption of the `fframe()` option.
- And all these programs have `reverse` and `percent` options.

So what packages on SSC compute ridits?

- The `ridit()` function of Nicholas J. Cox's `egenmore` package computes unweighted native center ridits for a variable.
- The package `wridit` allows the ridits to be weighted and/or folded (transformed to a scale from -1 to 1 instead of from 0 to 1), as recommended by Brockett and Levene (1977)[1].
- It also has a `handedness()` option, with possible values `left`, `right`, and `center` (the default).
- In Stata Version 16, `wridit` has a second module `fridit`, specifying foreign ridits for a variable, with respect to the distribution of a variable of the same name in a second data frame, specified by the `fframe()` option.
- These foreign ridits may be weighted by a variable in the second data frame, using the `weight()` suboption of the `fframe()` option.
- And all these programs have `reverse` and `percent` options.

So what packages on SSC compute ridits?

- The `ridit()` function of Nicholas J. Cox's `egenmore` package computes unweighted native center ridits for a variable.
- The package `wridit` allows the ridits to be weighted and/or folded (transformed to a scale from -1 to 1 instead of from 0 to 1), as recommended by Brockett and Levene (1977)[1].
- It also has a `handedness()` option, with possible values `left`, `right`, and `center` (the default).
- In Stata Version 16, `wridit` has a second module `fridit`, specifying foreign ridits for a variable, with respect to the distribution of a variable of the same name in a second data frame, specified by the `fframe()` option.
- These foreign ridits may be weighted by a variable in the second data frame, using the `weight()` suboption of the `fframe()` option.
- And all these programs have `reverse` and `percent` options.

So what use are these packages?

- I have been developing **ridit splines**[3, 4] as non–mechanistic regression models (like those from `npregress series`).
- A ridit spline in a variable X is a spline in a ridit of X.
- We usually compute a ridit and use the SSC package `polyspline`[5] to compute an unrestricted spline basis in the ridit.
- The parameters of the ridit spline are values of the spline corresponding to **reference values** of the ridit, such as 0 to 1 by increments of 0.25, corresponding to the corresponding **percentiles** of the original X–variable.
- Note that a percentile function is a generalized inverse of a ridit function.
- A ridit spline does not often produce extreme predictions, because it is based on a ridit function with range bounded between 0 and 1, with regular spline knots. This feature is very useful when stabilizing (or Winsorising) inverse–probability weights.

So what use are these packages?

- I have been developing **ridit splines**[3, 4] as non–mechanistic regression models (like those from `npregress series`).
- A ridit spline in a variable *X* is a spline in a ridit of *X*.
- We usually compute a ridit and use the SSC package `polyspline`[5] to compute an unrestricted spline basis in the ridit.
- The parameters of the ridit spline are values of the spline corresponding to **reference values** of the ridit, such as 0 to 1 by increments of 0.25, corresponding to the corresponding **percentiles** of the original *X*–variable.
- Note that a percentile function is a generalized inverse of a ridit function.
- A ridit spline does not often produce extreme predictions, because it is based on a ridit function with range bounded between 0 and 1, with regular spline knots. This feature is very useful when stabilizing (or Winsorising) inverse–probability weights.

So what use are these packages?

- I have been developing **ridit splines**[3, 4] as non–mechanistic regression models (like those from `npregress series`).
- A ridit spline in a variable X is a spline in a ridit of X.
- We usually compute a ridit and use the SSC package `polyspline`[5] to compute an unrestricted spline basis in the ridit.
- The parameters of the ridit spline are values of the spline corresponding to **reference values** of the ridit, such as 0 to 1 by increments of 0.25, corresponding to the corresponding **percentiles** of the original X–variable.
- Note that a percentile function is a generalized inverse of a ridit function.
- A ridit spline does not often produce extreme predictions, because it is based on a ridit function with range bounded between 0 and 1, with regular spline knots. This feature is very useful when stabilizing (or Winsorising) inverse–probability weights.

So what use are these packages?

- I have been developing **ridit splines**[3, 4] as non–mechanistic regression models (like those from `npregress series`).
- A ridit spline in a variable X is a spline in a ridit of X.
- We usually compute a ridit and use the SSC package `polyspline`[5] to compute an unrestricted spline basis in the ridit.
- The parameters of the ridit spline are values of the spline corresponding to **reference values** of the ridit, such as 0 to 1 by increments of 0.25, corresponding to the corresponding **percentiles** of the original X–variable.
- Note that a percentile function is a generalized inverse of a ridit function.
- A ridit spline does not often produce extreme predictions, because it is based on a ridit function with range bounded between 0 and 1, with regular spline knots. This feature is very useful when stabilizing (or Winsorising) inverse–probability weights.

So what use are these packages?

- I have been developing **ridit splines**[3, 4] as non–mechanistic regression models (like those from `npregress series`).
- A ridit spline in a variable *X* is a spline in a ridit of *X*.
- We usually compute a ridit and use the SSC package `polyspline`[5] to compute an unrestricted spline basis in the ridit.
- The parameters of the ridit spline are values of the spline corresponding to **reference values** of the ridit, such as 0 to 1 by increments of 0.25, corresponding to the corresponding **percentiles** of the original *X*–variable.
- Note that a percentile function is a generalized inverse of a ridit function.
- A ridit spline does not often produce extreme predictions, because it is based on a ridit function with range bounded between 0 and 1, with regular spline knots. This feature is very useful when stabilizing (or Winsorising) inverse–probability weights.

So what use are these packages?

- I have been developing **ridit splines**[3, 4] as non–mechanistic regression models (like those from `npregress series`).
- A ridit spline in a variable X is a spline in a ridit of X.
- We usually compute a ridit and use the SSC package `polyspline`[5] to compute an unrestricted spline basis in the ridit.
- The parameters of the ridit spline are values of the spline corresponding to **reference values** of the ridit, such as 0 to 1 by increments of 0.25, corresponding to the corresponding **percentiles** of the original X–variable.
- Note that a percentile function is a generalized inverse of a ridit function.
- A ridit spline does not often produce extreme predictions, because it is based on a ridit function with range bounded between 0 and 1, with regular spline knots. This feature is very useful when stabilizing (or Winsorising) inverse–probability weights.

So what use are these packages?

- I have been developing **ridit splines**[3, 4] as non–mechanistic regression models (like those from `npregress series`).
- A ridit spline in a variable *X* is a spline in a ridit of *X*.
- We usually compute a ridit and use the SSC package `polyspline`[5] to compute an unrestricted spline basis in the ridit.
- The parameters of the ridit spline are values of the spline corresponding to **reference values** of the ridit, such as 0 to 1 by increments of 0.25, corresponding to the corresponding **percentiles** of the original *X*–variable.
- Note that a percentile function is a generalized inverse of a ridit function.
- A ridit spline does not often produce extreme predictions, because it is based on a ridit function with range bounded between 0 and 1, with regular spline knots. This feature is very useful when stabilizing (or Winsorising) inverse–probability weights.

Problem: Ridit splines fitted to a training set and tested in a test set

- Sometimes we may find a regression model in a training set, and test its predictions in a test set.
- *However*, if the regression model fitted to the training set is (or includes) a ridit spline, then the ridit spline model tested in the test set *must* be based on foreign ridits (as computed using `fridit`), or else the spline model will *not* be the one fitted to the training set.
- (This is because variables of the same name usually have different native ridit functions in the training set and in the test set, even if the spline functions used are the same.)
- We will demonstrate the correct procedure using an example from the `xauto` data, using US cars as the training set and non–US cars as the test set.

Problem: Ridit splines fitted to a training set and tested in a test set

- Sometimes we may find a regression model in a training set, and test its predictions in a test set.
- *However*, if the regression model fitted to the training set is (or includes) a ridit spline, then the ridit spline model tested in the test set *must* be based on foreign ridits (as computed using `fridit`), or else the spline model will *not* be the one fitted to the training set.
- (This is because variables of the same name usually have different native ridit functions in the training set and in the test set, even if the spline functions used are the same.)
- We will demonstrate the correct procedure using an example from the `xauto` data, using US cars as the training set and non–US cars as the test set.

Problem: Ridit splines fitted to a training set and tested in a test set

- Sometimes we may find a regression model in a training set, and test its predictions in a test set.
- *However*, if the regression model fitted to the training set is (or includes) a ridit spline, then the ridit spline model tested in the test set *must* be based on foreign ridits (as computed using `fridit`), or else the spline model will *not* be the one fitted to the training set.
- (This is because variables of the same name usually have different native ridit functions in the training set and in the test set, even if the spline functions used are the same.)
- We will demonstrate the correct procedure using an example from the `xauto` data, using US cars as the training set and non–US cars as the test set.

Problem: Ridit splines fitted to a training set and tested in a test set

- Sometimes we may find a regression model in a training set, and test its predictions in a test set.
- *However*, if the regression model fitted to the training set is (or includes) a ridit spline, then the ridit spline model tested in the test set *must* be based on foreign ridits (as computed using `fridit`), or else the spline model will *not* be the one fitted to the training set.
- (This is because variables of the same name usually have different native ridit functions in the training set and in the test set, even if the spline functions used are the same.)
- We will demonstrate the correct procedure using an example from the `xauto` data, using US cars as the training set and non–US cars as the test set.

Problem: Ridit splines fitted to a training set and tested in a test set

- Sometimes we may find a regression model in a training set, and test its predictions in a test set.
- *However*, if the regression model fitted to the training set is (or includes) a ridit spline, then the ridit spline model tested in the test set *must* be based on foreign ridits (as computed using `fridit`), or else the spline model will *not* be the one fitted to the training set.
- (This is because variables of the same name usually have different native ridit functions in the training set and in the test set, even if the spline functions used are the same.)
- We will demonstrate the correct procedure using an example from the `xauto` data, using US cars as the training set and non–US cars as the test set.

Fuel consumption and weight in tons in the `xauto` data

- In the training set of US cars, we fit a ridit spline in `tons` (weight of a car in US tons) to predict `npm` (fuel consumption in nipperkins per mile), and save the estimation results for use in the test set.
- We then use the SSC package `xcontract` to create a **frequency data frame**, with 1 observation per value of `tons` in the training set, and data on frequencies and percents of each value in the training set.
- Inputting the test set of non–US cars, we compute the foreign ridits of `tons` in the test set, with respect to the distribution of `tons` in the frequency data frame (and in the training set).
- We can then use `polyspline` to compute a **foreign ridit spline basis** in the test set, defined using the same spline formula as in the training set.
- We can then use the estimation results from the training set to compute out–of–sample predicted values for `npm` in the test set.

Fuel consumption and weight in tons in the `xauto` data

- In the training set of US cars, we fit a ridit spline in `tons` (weight of a car in US tons) to predict `npm` (fuel consumption in nipperkins per mile), and save the estimation results for use in the test set.
- We then use the SSC package `xcontract` to create a **frequency data frame**, with 1 observation per value of `tons` in the training set, and data on frequencies and percents of each value in the training set.
- Inputting the test set of non–US cars, we compute the foreign ridits of `tons` in the test set, with respect to the distribution of `tons` in the frequency data frame (and in the training set).
- We can then use `polyspline` to compute a **foreign ridit spline basis** in the test set, defined using the same spline formula as in the training set.
- We can then use the estimation results from the training set to compute out–of–sample predicted values for `npm` in the test set.

Fuel consumption and weight in tons in the `xauto` data

- In the training set of US cars, we fit a ridit spline in `tons` (weight of a car in US tons) to predict `npm` (fuel consumption in nipperkins per mile), and save the estimation results for use in the test set.
- We then use the SSC package `xcontract` to create a **frequency data frame**, with 1 observation per value of `tons` in the training set, and data on frequencies and percents of each value in the training set.
- Inputting the test set of non–US cars, we compute the foreign ridits of `tons` in the test set, with respect to the distribution of `tons` in the frequency data frame (and in the training set).
- We can then use `polyspline` to compute a **foreign ridit spline basis** in the test set, defined using the same spline formula as in the training set.
- We can then use the estimation results from the training set to compute out–of–sample predicted values for `npm` in the test set.

Fuel consumption and weight in tons in the `xauto` data

- In the training set of US cars, we fit a ridit spline in `tons` (weight of a car in US tons) to predict `npm` (fuel consumption in nipperkins per mile), and save the estimation results for use in the test set.
- We then use the SSC package `xcontract` to create a **frequency data frame**, with 1 observation per value of `tons` in the training set, and data on frequencies and percents of each value in the training set.
- Inputting the test set of non–US cars, we compute the foreign ridits of `tons` in the test set, with respect to the distribution of `tons` in the frequency data frame (and in the training set).
- We can then use `polyspline` to compute a **foreign ridit spline basis** in the test set, defined using the same spline formula as in the training set.
- We can then use the estimation results from the training set to compute out–of–sample predicted values for `npm` in the test set.

Fuel consumption and weight in tons in the `xauto` data

- In the training set of US cars, we fit a ridit spline in `tons` (weight of a car in US tons) to predict `npm` (fuel consumption in nipperkins per mile), and save the estimation results for use in the test set.
- We then use the SSC package `xcontract` to create a **frequency data frame**, with 1 observation per value of `tons` in the training set, and data on frequencies and percents of each value in the training set.
- Inputting the test set of non–US cars, we compute the foreign ridits of `tons` in the test set, with respect to the distribution of `tons` in the frequency data frame (and in the training set).
- We can then use `polyspline` to compute a **foreign ridit spline basis** in the test set, defined using the same spline formula as in the training set.
- We can then use the estimation results from the training set to compute out–of–sample predicted values for `npm` in the test set.

Fuel consumption and weight in tons in the `xauto` data

- In the training set of US cars, we fit a ridit spline in `tons` (weight of a car in US tons) to predict `npm` (fuel consumption in nipperkins per mile), and save the estimation results for use in the test set.
- We then use the SSC package `xcontract` to create a **frequency data frame**, with 1 observation per value of `tons` in the training set, and data on frequencies and percents of each value in the training set.
- Inputting the test set of non–US cars, we compute the foreign ridits of `tons` in the test set, with respect to the distribution of `tons` in the frequency data frame (and in the training set).
- We can then use `polyspline` to compute a **foreign ridit spline basis** in the test set, defined using the same spline formula as in the training set.
- We can then use the estimation results from the training set to compute out–of–sample predicted values for `npm` in the test set.

Computing native ridits in the training set of US cars

In the training set, we use `wridit` to generate a variable `r_tons`, containing the native ridits of the variable `tons` (car weight in tons):

```
. wridit tons, gene(r_tons);

. lab var r_tons "Ridit of tons in US cars";

. summ r_tons, de format;

                  Ridit of tons in US cars
-------------------------------------------------------------
      Percentiles      Smallest
 1%     .0192308       .0192308
 5%     .0480769       .0192308
10%     .1057692       .0480769       Obs                  52
25%          .25       .0673077       Sum of wgt.          52

50%     .5048077                      Mean                 .5
                        Largest       Std. dev.      .2914065
75%          .75       .9326923
90%     .8942308       .9519231       Variance       .0849178
95%     .9519231       .9711538       Skewness      -.0001024
99%     .9903846       .9903846       Kurtosis       1.798588
```

We see that the native ridit is bounded in the open interval from 0 to 1, and has an approximately uniform distribution in that interval.

Computing the ridit spline basis in the training set

We then use `polyspline` to generate a cubic reference spline basis in `r_tons`, with reference points at ridits 0 to 1 by 0.25, corresponding to percentiles 0 to 100 by 25 of the original *X*–variable `tons`:

```
. polyspline r_tons, power(3) refpts(0(0.25)1)
>   gene(sp_) labprefix("Spline at ");
5 reference splines generated of degree: 3

. describe sp_*, fu;

Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------------------------
sp_1            float   %8.4f                 Spline at 0
sp_2            float   %8.4f                 Spline at .25
sp_3            float   %8.4f                 Spline at .5
sp_4            float   %8.4f                 Spline at .75
sp_5            float   %8.4f                 Spline at 1
```

We see that the spline basis is informatively labelled, and ready for fitting a regression model.

Fitting the ridit spline model in the training set

We then use `regress` to fit a model for `npm` (fuel consumption in nipperkins per mile) in the cubic reference spline basis:

```
. regress npm sp_*, vce(robust) noconst;

Linear regression                               Number of obs     =         52
                                                F(5, 47)          =    1228.82
                                                Prob > F          =     0.0000
                                                R-squared         =     0.9882
                                                Root MSE          =     1.5957

------------------------------------------------------------------------------
             |               Robust
         npm | Coefficient  std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
        sp_1 |   8.157939   .5708161    14.29   0.000     7.009605    9.306273
        sp_2 |   11.68852   .2920866    40.02   0.000     11.10092    12.27612
        sp_3 |   13.65935   .3334914    40.96   0.000     12.98845    14.33025
        sp_4 |    15.3916   .5129863    30.00   0.000      14.3596    16.42359
        sp_5 |   19.57925    1.67407    11.70   0.000     16.21146    22.94704
------------------------------------------------------------------------------
```

The parameters are mean fuel consumption rates (in nipperkins per mile) for the reference points of the ridit spline.

Listing the ridit spline model parameters using **parmest**

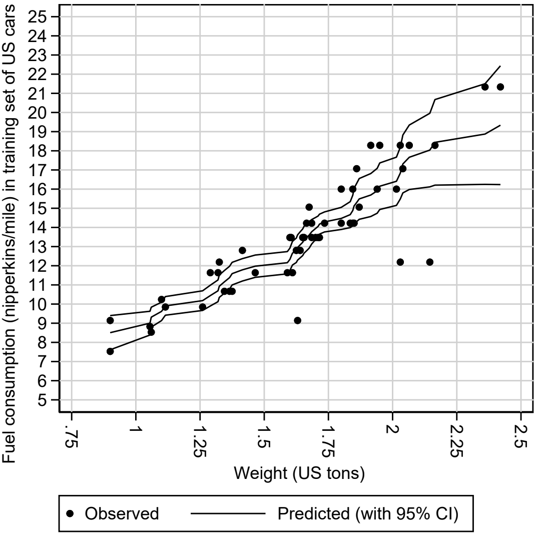
We then use the SSC package `parmest`, with the `label` option, to list the parameters of the ridit spline model for the 52 US cars:

```
. parmest, label escal(N) rename(es_1 N) format(estimate min* max* %8.3f)
>   list(N parm label estimate min* max*, abbr(32));

     +---------------------------------------------------------+
     |  N   parm            label   estimate    min95    max95 |
     |---------------------------------------------------------|
  1. | 52   sp_1      Spline at 0      8.158    7.010    9.306 |
  2. | 52   sp_2    Spline at .25     11.689   11.101   12.276 |
  3. | 52   sp_3     Spline at .5     13.659   12.988   14.330 |
  4. | 52   sp_4    Spline at .75     15.392   14.360   16.424 |
  5. | 52   sp_5      Spline at 1     19.579   16.211   22.947 |
     +---------------------------------------------------------+
```

The parameters are mean fuel consumption rates (in nipperkins per mile) for the reference ridits 0 to 1 by 0.25, corresponding to percentiles 0 to 100 by 25 of car weight in tons.

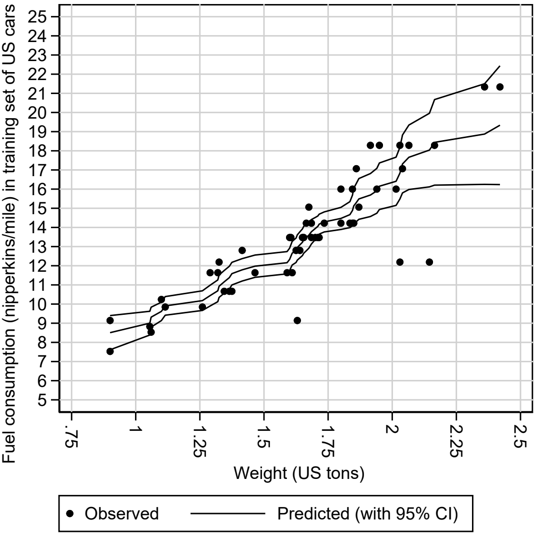
Observed and predicted values of fuel consumption in the training set

- The horizontal axis gives car weight in US tons.
- The vertical axis gives the observed values of fuel consumption, and the predicted values from the ridit spline (given as lines with confidence limits).
- The ridit spline predicts well (for most cars), at least in the training set of US cars.



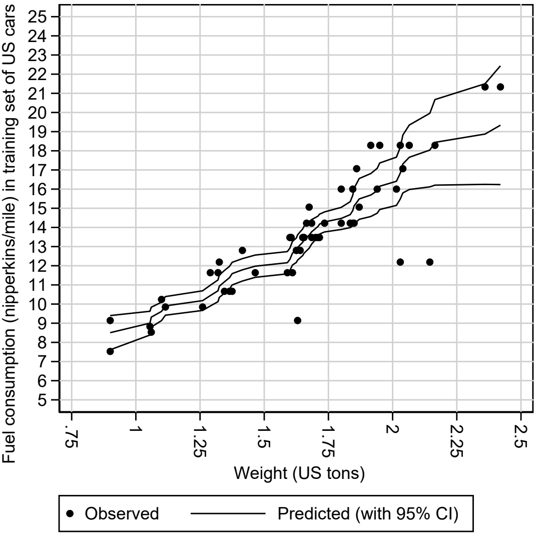
Observed and predicted values of fuel consumption in the training set

- The horizontal axis gives car weight in US tons.
- The vertical axis gives the observed values of fuel consumption, and the predicted values from the ridit spline (given as lines with confidence limits).
- The ridit spline predicts well (for most cars), at least in the training set of US cars.



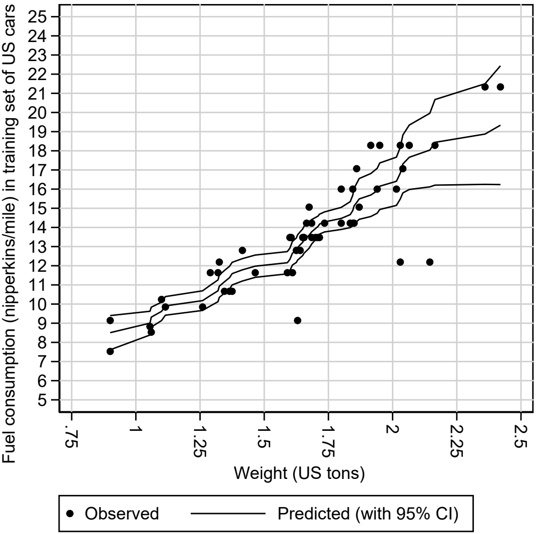
Observed and predicted values of fuel consumption in the training set

- The horizontal axis gives car weight in US tons.
- The vertical axis gives the observed values of fuel consumption, and the predicted values from the ridit spline (given as lines with confidence limits).
- The ridit spline predicts well (for most cars), at least in the training set of US cars.



Observed and predicted values of fuel consumption in the training set

- The horizontal axis gives car weight in US tons.
- The vertical axis gives the observed values of fuel consumption, and the predicted values from the ridit spline (given as lines with confidence limits).
- The ridit spline predicts well (for most cars), at least in the training set of US cars.



Saving the fitted model in files for use in the test set

To test the model in the test set, we must first save 2 files. These are a file `rstons.ster`, containing the estimation results for the ridit spline in the training set, and a file `tonsfreq.dta`, produced by the SSC package `xcontract`, with 1 observation per value of `tons`, and data on their frequencies and percents in the training set.

```
. estimates save rstons.ster, replace;
file rstons.ster saved

. xcontract tons, saving(tonsfreq.dta, replace)
>   list(, abbr(32));

     +-------------------------+
     |  tons   _freq  _percent |
     |-------------------------|
  1. |    .9       2      3.85 |
  2. | 1.055       1      1.92 |
  3. |  1.06       1      1.92 |
  4. |   1.1       1      1.92 |
  5. | 1.115       1      1.92 |
     |-------------------------|
  6. |  1.26       1      1.92 |
  7. |  1.29       1      1.92 |
```

With these 2 files, we can now test the ridit spline model in the test set of non–US cars.

Generating foreign ridits in the test set of non–US cars

In the test set, we input the training set frequencies in `tonsfreq.dta` into a data frame also called `tonsfreq`, and use `fridit` to create a new variable `r_tons`, containing ridits for non–US cars with respect to the distribution of weights in US cars:

```
cap frame drop tonsfreq;
frame create tonsfreq;
frame tonsfreq: use tonsfreq.dta, clear;
fridit tons, fframe(tonsfreq, weight(_freq))
  gene(r_tons);
lab var r_tons "Ridit of tons in US cars";
```

Distribution of foreign ridits in the test set of non–US cars

We use `summarize` to view the distribution of these foreign ridits:

```
. summ r_tons, de format;

                  Ridit of tons in US cars
-------------------------------------------------------------
      Percentiles      Smallest
 1%            0              0
 5%     .0384615       .0384615
10%     .0384615       .0384615       Obs                  22
25%     .0384615       .0384615       Sum of wgt.          22

50%     .0817308                      Mean           .1245629
                        Largest       Std. dev.      .1279236
75%     .1826923       .2403846
90%     .2596154       .2596154       Variance       .0163644
95%     .2884615       .2884615       Skewness       2.058216
99%     .5673077       .5673077       Kurtosis       7.523096
```

We see that one non–US car has a zero foreign ridit (being lighter than any US car), and that only one non–US car has a foreign ridit greater than 0.5 (being heavier than most US cars).

Generating a spline basis in the foreign ridits

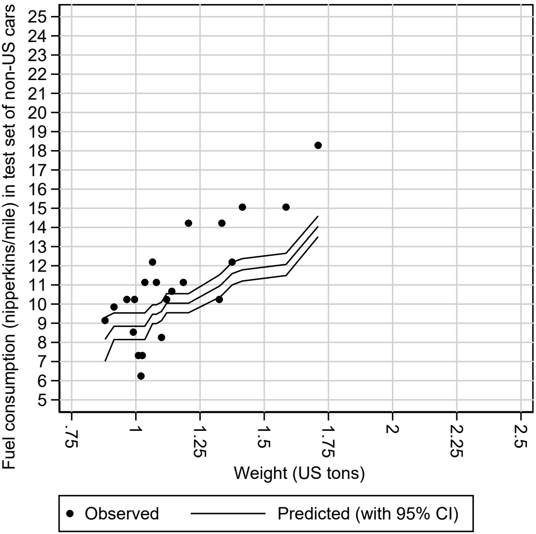
We now use `polyspline` to generate the ridit–spline basis in the test set as we did in the training set, this time using foreign ridits with respect to the training set:

```
. polyspline r_tons, power(3) refpts(0(0.25)1)
>   gene(sp_) labprefix("Spline at ");
5 reference splines generated of degree: 3

. describe sp_*, fu;

Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------------
sp_1            float   %8.4f                 Spline at 0
sp_2            float   %8.4f                 Spline at .25
sp_3            float   %8.4f                 Spline at .5
sp_4            float   %8.4f                 Spline at .75
sp_5            float   %8.4f                 Spline at 1
```

We can now use the estimation results from the training set to do out–of–sample prediction in the test set, using the ridit–spline model fitted to the training set.

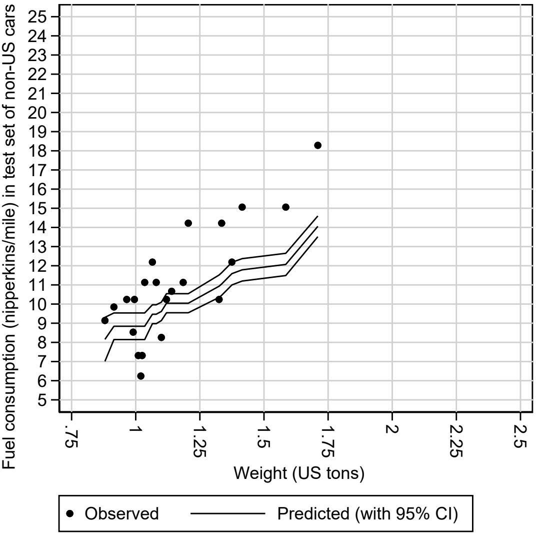
Observed and predicted values of fuel consumption in the test set

- The horizontal axis gives car weight in US tons.
- The vertical axis gives the observed values of fuel consumption, and the predicted values from the ridit spline (given as lines with confidence limits).
- We see that the heavier non–US cars consume more fuel per mile than we would expect if they were US cars of the same weight.



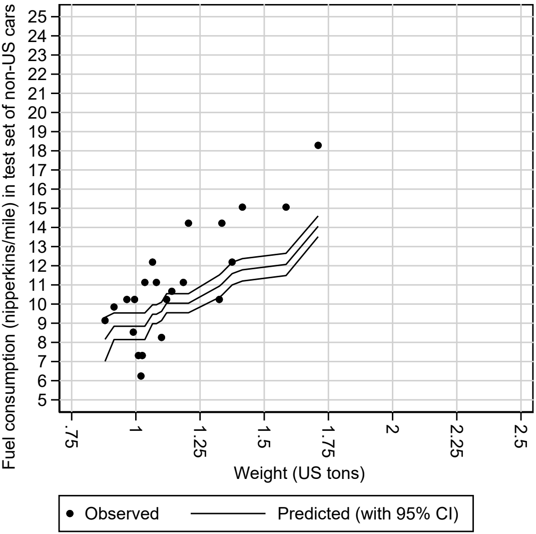
Observed and predicted values of fuel consumption in the test set

- The horizontal axis gives car weight in US tons.
- The vertical axis gives the observed values of fuel consumption, and the predicted values from the ridit spline (given as lines with confidence limits).
- We see that the heavier non–US cars consume more fuel per mile than we would expect if they were US cars of the same weight.



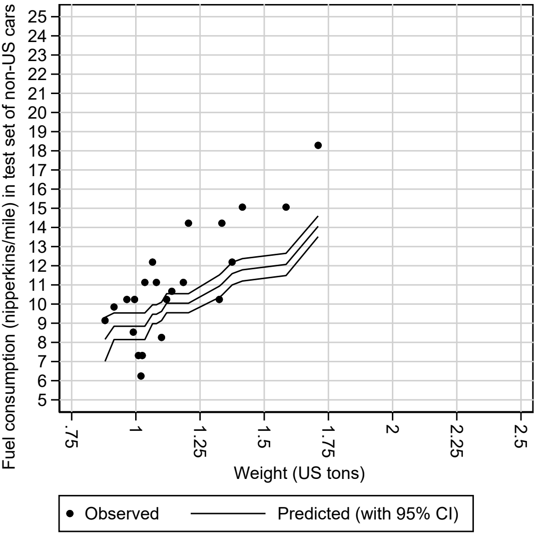
Observed and predicted values of fuel consumption in the test set

- The horizontal axis gives car weight in US tons.
- The vertical axis gives the observed values of fuel consumption, and the predicted values from the ridit spline (given as lines with confidence limits).
- We see that the heavier non–US cars consume more fuel per mile than we would expect if they were US cars of the same weight.



Observed and predicted values of fuel consumption in the test set

- The horizontal axis gives car weight in US tons.
- The vertical axis gives the observed values of fuel consumption, and the predicted values from the ridit spline (given as lines with confidence limits).
- We see that the heavier non–US cars consume more fuel per mile than we would expect if they were US cars of the same weight.



References

[1] Brockett, P. L., and Levene, A. 1977. On a characterization of ridits. *The Annals of Statistics* **5(6)**: 1245–1248.

[2] Bross, I. D. J. 1958. How to use ridit analysis. *Biometrics* **14(1)**: 18–38.

[3] Newson, R. B. Bland–Altman plots, rank parameters, and calibration ridit splines. Presented at the 2019 London Stata Conference, 5–6 September, 2019. Downloadable from the conference website at *http://ideas.repec.org/p/boc/usug19/01.html*

[4] Newson, R. B. Ridit splines with applications to propensity weighting. Presented at the 23rd UK User Meeting, 7–8 September, 2017. Downloadable from the conference website at *http://ideas.repec.org/p/boc/usug17/01.html*

[5] Newson, R. B. Easy–to–use packages for estimating rank and spline parameters. Presented at the Presented at the 20th UK Stata User Meeting, 1112 September, 2014. Downloadable from the conference website at *http://ideas.repec.org/p/boc/usug14/01.html*

The presentation, and the example do–files, can be downloaded from the conference website, and the packages used can be downloaded from SSC.