This page documents the proposal that we will implement in the future.  Please see [**(Current) Store Price Domain**](about:///display/EDDF/(Current)+Store+Price+Domain)for the current implementation.

**Background**

reg\_retl\_a: Regular Price, also known as Regular Retail, is a Target pricing program. Price that reflects base price or shelf price. It is the base price defined in the item management system for the given store and day, prior to any promotional markdowns that may occur

sls\_retl\_a: A point-in-time snapshot of the merchant-established \*todays\* retail price, charged by Target Corporation to its guests, to purchase a single unit of an item, prior to any store discretionary markdowns or transaction level discounts.

ext\_sls\_prc\_a: Total adjusted amount (after item level promotional pricing and item level discounts) of an item. For item level tables there will be one row per item. For transaction level tables item level detail is summarized up to the transaction level.  Note that to get to the per-unit price the customer paid, we will need to divide by sales (i.e. sls\_unit\_q from prd\_ssl\_fnd.mdse\_slstr\_item\_line)

We currently keep records with nonsensical values of sls\_retl\_a or reg\_retl\_a.  However, for models with price, using poor values of sls\_retl\_a can affect the model.  We attempt to correct the sls\_retl\_a and reg\_retl\_a by using the following algorithm.  For now, we will not adjust ext\_sls\_prc\_a, even though many nonsensical values exist.  Because this represents the amount the customer actually paid, we will preserve the values as-is for now.

**History**

**Section 1: Imputation**

1)    **Reg\_retl\_a imputation:** For each store-item, find a 7-day window prior to the date of interest.  We require that the 7-day window contains at least 5 data points.  Find the median of reg\_retl\_a.  If the reg\_retl\_a differs from the median value by greater than 15%, change the reg\_retl\_a value to the median value.  In the event that the store-item does not contain sufficient data in the past 7 days, use the Ad Patch-item 7-day window median value for comparison purposes.  Ad Patches may change over time, but we will use the most current mapping, with the assumption that the Ad Patch definition does not change much over time, and if they do, the changes are not substantial.  If the 7-day Ad Patch data are missing, we will use chain level data.  In the event that we don’t have sufficient data at the 7-day level, use 30-day instead.  In the event that we don’t have data in the past 30 days at either the store-item level, Ad Patch-item level, or chain-item level, leave the record as is.  In the event that reg\_retl\_a is negative or less than a dime after this step, throw an error and provide the specific use case to Luyen.  She will further refine the algorithm to handle the problematic use case.

Note: Ad Patch can be found in prd\_loc\_fnd.co\_loc\_dim.  There may not be a one-to-one mapping between co\_loc\_i to Ad Patch, so we need to make sure to assign co\_loc\_i to a unique Ad Patch.

2)    **Sls\_retl\_a imputation:** Using the refined reg\_retl\_a values from step 1, compute abs(reg\_retl\_a – sls\_retl\_a)/reg\_retl\_a.  If this value is greater than 75%, use the same steps described in reg\_retl\_a imputation, but do this using the sls\_retl\_a variable instead.  After this step, we will have a sls\_retl\_a final prices.  In the event that sls\_retl\_a is negative or less than a dime after this step, throw an error and provide the specific use case to Luyen.

Here is an example of what we would like to do.

Item A, store 1, processing date 11/8/2000 (only reg\_retl\_a values shown)

7-days prior transaction data

11/1/2000: 5.99, 5.55, 5.99, 5.99, 6.05

11/2/2000: 5.99, 5.99, 0

11/3/2000: 5.99, 5.99

11/4/2000: 6.05, 5.99, 5.99, 5.99

11/5/2000: 5.99, 5.99, 3.99, 5.99

11/6/2000: 3.99, 6.05

11/7/2000: 5.99

-----------------------------------------------

11/8/2000: 5.99, 0, 6.05, 5.55, 3.99

The median for the past 7 days is 5.99.  We will replace the values of 0 and 3.99 to 5.99.  We would not change 6.05 and 5.55 because they are within the acceptable window.

For the first day, we won't have prior transactions.  To handle this particular corner case, use 7 days after the first transaction.

For the second day, we only have 1 day of prior transactions. Use that day and 6 days after the second day for a total of 7 days.

Use similar logic until we reach day 8, when we have prior transactions.

Notes: 75% was chosen as the cut-off because clearance prices can go as high as 70%.  The parameters of 7 day window, 30 day window,  and 15% acceptance window were somewhat arbitrarily.   We chose 7 day because we wanted the time frame to be long enough to yield sufficient transactions but not too long that the prices may not be relevant.  We also considered using the weighted median for prices, using sales units as the weights, versus the straight median for the imputations.  We decided on the straight median for ease of computation.

**Section 2: Carry Forward**

We currently use the most recent value for a particular store-item and carry that value forward until we reach the next non-missing date.  However, this value may be a promoted value and it is unlikely that the promoted value would be the correct value for an imputed period that lasts longer than a week.  Instead of using the most recent value, should use the weighted median using sales as the weights from the other stores in the same Ad Patch for the particular date.  In the event that Ad Patch data is missing, we will use chain level data.  In the event that chain level data is missing, we will revert to the current method of using the most recent value.  We should only use the carry forward algorithm for reg\_retl\_a.  Sls\_retl\_a should not be carried forward.  It should only be populated when a non-zero sale occurred.

**Schema**

**Section 3: Aggregate to chain level**

To aggregate from store level to chain level, we would like to add the following new variables at the item-chain-week level

1)    **wt\_reg\_retl\_a:** Volume-weighted reg\_retl\_a across all stores for a particular item-week

2)    **wt\_sls\_retl\_a:**Volume-weighted sls\_retl\_a across all stores for a particular item-week

3)    **wt\_ext\_sls\_prc\_a:** Volume-weighted ext\_sls\_prc\_a across all stores for a particular item-week.  Note that we would like this to represent the per-unit price the customer paid.

4)    **med\_reg\_retl\_a:** Median of reg\_retl\_a across all stores for a particular item-week

5)    **med\_sls\_retl\_a:** Median of sls\_retl\_a across all stores for a particular item-week

Note that the final datasets should not have missing, negative, or zero prices.

**Future**

**Section 4: Future price values**

For future price values, we have two types of prices: non-promoted and promoted

1)    Non-promoted prices (ie regular price): Using store-item level data, we will use the median values of reg\_retl\_a (i.e. med\_reg\_retl\_a) using the last 5 weeks.  If data do not exist at the store-item level, use the last 5 weeks of Ad-Patch-item level and use the median as the regular price in the future.  If data do not exist at that level, use the chain-item data.  In the event that no data exist for the last 5 weeks of data, use the most recent reg\_retl\_a value and carry forward that value into the future.

I have recently learned of prd\_price\_fnd.price\_publish\_store and prd\_sho\_fnd.mdse\_item\_loc for base prices.  I need to conduct research on this table for its usability.  It may subsume the logic for non-promoted prices in the future.

2)    Promoted price: For promoted weeks, we should calculate the promoted price based on the details of the offer type.  For example, if the offer is 2 for $1, then the future price should be $0.50 for the promoted week.  For promotions that are a percentage off the regular price, we need to predict the regular price first and then calculate the promoted price based on the regular price.  Because promotions are not necessarily chain-wide, we will need to calculate the promoted price at the store-item-week level using future promotion details. To aggregate to the chain level, we will take a straight average.  Using a straight average is most likely to be wrong because non-promoted stores will likely have lower sales than promoted stores.  However, we do not know what the future sales will be at the store level in order to create a weighted price at the chain level.  One future improvement is use the promotion lift variable to predict future sales assuming that every store is promoted, scale the prediction down to account for the fact that not every store will be promoted and use the scaled predictions and full promoted predictions to calculate a future promoted price.

Note that the examples we have discussed thus far are price-based offers and not basket-based offers.  For offers involving baskets, we do not know the future redemption rate to estimate a future price.  One method to approximate the redemption rate is to use past redemption rates for similar basket-based offers.  More research is required, and we will not handle basket-based offers for now. To estimate the effect of basket-based offers, we will rely on the basket promotion variable.

**Tracking**

TBD: Come up with a methodology to track poor prices.  Only item-chain level prices will be tracked because it may be too difficult to track at a item-store level and we don't have anyone using store level prices at this time.  On the other hand, if we only track at the chain level, we may lose on on small differences because chain level tends to be very robust.  I'll need to think about this.