Fixing Incorrect UPC Version Changes

0. Note

All examples are based on modified datasets built for documentational purpose. There will be some discrepancies with the true datasets.

1. The Problem

In Nielsen data, a UPC can be reassigned to an entirely different product, or to a similar product with different size or packaging. UPC versions (`upc_ver_uc`) are generated based on four "core attributes": `product_module_code`, `brand_code` (equivalent to `brand_descr`), `multi`, and `size1_code_uc`. A new UPC version is created when any one of these attributes changes[1].

In a number of cases, a UPC version change is triggered when `brand_descr` is slightly modified, while all the other core attributes remain unchanged. This situation might occur when the brand spelling changes, or when details are added to or removed from brand names. For example:

1) Brand Spelling Changes:

```
##           upc upc_ver_uc            upc_descr product_module_code
## 1: 1111111111          1 409 G&S C CS L T NDS                7777
## 2: 1111111111          2   F409 G&S C LST NDS                7777
```

2) Details Added:

```
##           upc upc_ver_uc          upc_descr                    brand_descr
## 1: 2222222222          1 C-T-W TBWR CL NY D             CLOROX TOILET WAND
## 2: 2222222222          2 C-T-W TBWR CL NY D CLOROX TOLT WND CLN/PLSH CLOTH
```

We treat such cases as "false positive" version changes, since the different versions actually represent the **same product**. When a false positive version change occurs, the product's price time series is broken down to two or more pieces. Since our base price detection algorithm does not predict base prices for the first and the last 10 observations, it is better to feed it with a complete time series rather than several small chunks of prices. Therefore, it is necessary to correct these false positives in order to improve the performance of our price processing algorithm.

[1] Kilts Nielsen Retail Scanner Dataset Manual (2014), 11

2. The Fix

2.1. Various Small Problems

2.1.1. "Missing" Versions

Since we are only interested in RMS price movements, and the meta data(from previous RMS build steps) is built from the movement files, it is safe to consider only the `upc`-`upc_ver_uc` pairs that appear in the meta data. Note that the number of unique pairs in the meta data = 1767405.

However, remember that a new UPC version is created whenever a change occurs in RMS or HMS data. Hence, for some UPCs the version numbers may not be continuous. For instance:

1) Meta Data (Part) Example:

2) Product Information Table Example:

```
##           upc upc_ver_uc                  upc_descr product_module_code
## 1: 4444444444          1   SPIDERMAN M/MIN N CC GMI                8888
## 2: 4444444444          2 RSI-NBL M/MIN SPD N CC GMI                8888
## 3: 4444444444          3   MVL HRS M/MIN GOC CC GMI                8888
```

Given this situation, we should not simply change version `N` to `N-1` when a false positive occurs.

2.1.2. Inconsistent Version Numbering

Intuitively, the version number (`upc_ver_uc`) increases or stays the same throughout the years. However, for some UPCs the version number actually decreases (or even worse, changes irregularly) as new versions are created. For example:

1) Decreasing Version Numbers:

```
##           upc upc_ver_uc year
## 1: 5555555555          2 2013
## 2: 5555555555          1 2014
```

2) Irregular Changes:

```
##           upc upc_ver_uc year
## 1: 7777777777          3 2006
## 2: 7777777777          3 2007
## 3: 7777777777          3 2008
## 4: 7777777777          3 2009
## 5: 7777777777          3 2010
## 6: 7777777777          3 2011
## 7: 7777777777          4 2012
## 8: 7777777777          1 2013
## 9: 7777777777          1 2014
```

Fortunately, such inconsistencies are not hard to deal with[2]. We just need to keep in mind that an "old" UPC version does not necessarily have a small version number.

2.2. Detecting Potential False Positives

We define a "potential false positive" as a UPC version change triggered only by a brand name change. In other words, the fields `multi`, `size1_code_uc`, and `product_module_code` must not change between versions. After excluding general merchandises, 12733 potential false positives are found, as shown in the table `only_brand_change_cases` in `./Processed-Data/Version-Change-Tables.RData`:

```
##           upc max_rank old_version new_version        old_brand_descr
## 1: 8888888888   111111           1           2 HEALTH PRO FRESH BITES
##    new_brand_descr false_positive revenue_change_pct old_revenue_store
## 1:     FRESH BITES              0           -44.4444              0.88
##    new_revenue_store              old_upc_descr             new_upc_descr
## 1:              0.44 HLPFB THCL WHITENER BSC DG FRBT THCL WHITENER BSC DG
```

To explain some of the variables in this table: `max_rank` is the highest annual rank of the UPC by RMS revenue; `old_revenue_store` is the average per-store revenue of the UPC in the last year of the old version, while `new_revenue_store` is the average per-store revenue in the first year of the new version. The percentage change (normalized to 100) of per-store revenue, `revenue_change_pct`, provides a very rough estimate of the "stability" of a UPC through time. We may infer that a version change is incorrect if the revenue change is only 0.01%, or that a version change is correct if the average revenue increases by 1000%. However, such inference should only be used as a back-up tool. The difference between old and new brand names is the most important criterion for detecting false positives.

Since most of our analyses only focuses on top 50000 RMS products, and it is difficult to manually scan through 12733 potential false positive cases, we limit our attention to top 50000 UPCs each year. As a result, only 3342 cases need to be manually checked.

```
potential_in_top_50000 = only_brand_change_cases[max_rank <= 50000]
nrow(potential_in_top_50000)
```

```
## [1] 3342
```

2.3. Manual Check

Having manually checked all 3342 cases, We found 3235 of them to be false positives (96.8%). The detailed procedure of the manual fix is as follows:

1. Find all cases of clear brand *spelling* changes. Label them as `false_positive=1`.
2. Find all cases where the old brand name and the new brand name shares a common key word.
   (a) If the common word is clearly a brand name (e.g. "DOLE FRESH MAKES" $\Rightarrow$ "DOLE"), then mark the case as `false_positive=1`.

[2] They are also rare in the data. Among 23309 UPCs that have multiple versions, only 184 of them have this problem.

(b) If the common word is not a brand name (e.g. "CROISSANT POCKETS" $\Rightarrow$ "HOT POCKETS"), then mark the case as `false_positive=0`.

3. Among cases that are not labelled false positive, find pairs of old and new brand names that appear multiple times (see Example 1 below). Use Google to determine whether the two names represent the same brand; if so, change `false_positive` to 1.

4. Find all cases where the old and new UPC names are the same; label all of them as false positives. When applicable, this step overwrites existing `false_positive` values. The reason is that a same UPC name cannot correspond to different brands.

Example 1: Brand Name Changes with Multiple Occurences

As introduced in step (3) above, we search for pairs of old and new brand names that do not share a common word but appear multiple times in the data. The rationale is that a brand name may have two or more parts, and Nielsen may only recorded the first part in the old version, and the second part in the new version. For instance:

```
##           upc old_version new_version     old_brand_descr new_brand_descr
## 1: 9999999999           1           2 DIGESTIVE ADVANTAGE          SCHIFF
## 2: 9999999999           1           2 DIGESTIVE ADVANTAGE          SCHIFF
## 3: 9999999999           1           2 DIGESTIVE ADVANTAGE          SCHIFF
## 4: 9999999999           1           2 DIGESTIVE ADVANTAGE          SCHIFF
```

"Digestive Advantage" and "Schiff" look like drastically different brand names; but according to this link, the brand is actually called "Schiff® Digestive Advantage®".

If we want to be extremely careful, we should search through *all* cases with `false_positive=0`. However, if we want to save time, we only need to search the cases where a same pair of old and new brand names appears multiple times–if a number of UPCs see a *same* brand name change, then it is more likely that the name change is false positive.

2.4. Correcting UPC Versions

Having identified false positives, the only task left is to generate a new field `upc_ver_uc_corrected` in both the meta-data and the movement files. The procedure is as follows:

1. Take all false positives, find the list of relevant UPCs and modules.

2. Load movements, generate `upc_ver_uc_corrected=upc_ver_uc`

3. Update `upc_ver_uc_corrected` as needed. Note in this step that UPC version numbers may not be continuous; so we need to be more careful on that.

For detailed procedure, check:

- `2-Correct-Versions-And-Process-Top-75000.R`
- `2-Correct-Versions-And-Process-All-Other.R`

Note that the base price imputation algorithm is implemented in both scripts.