# Managing large datasets in Stata

Erin Hengel

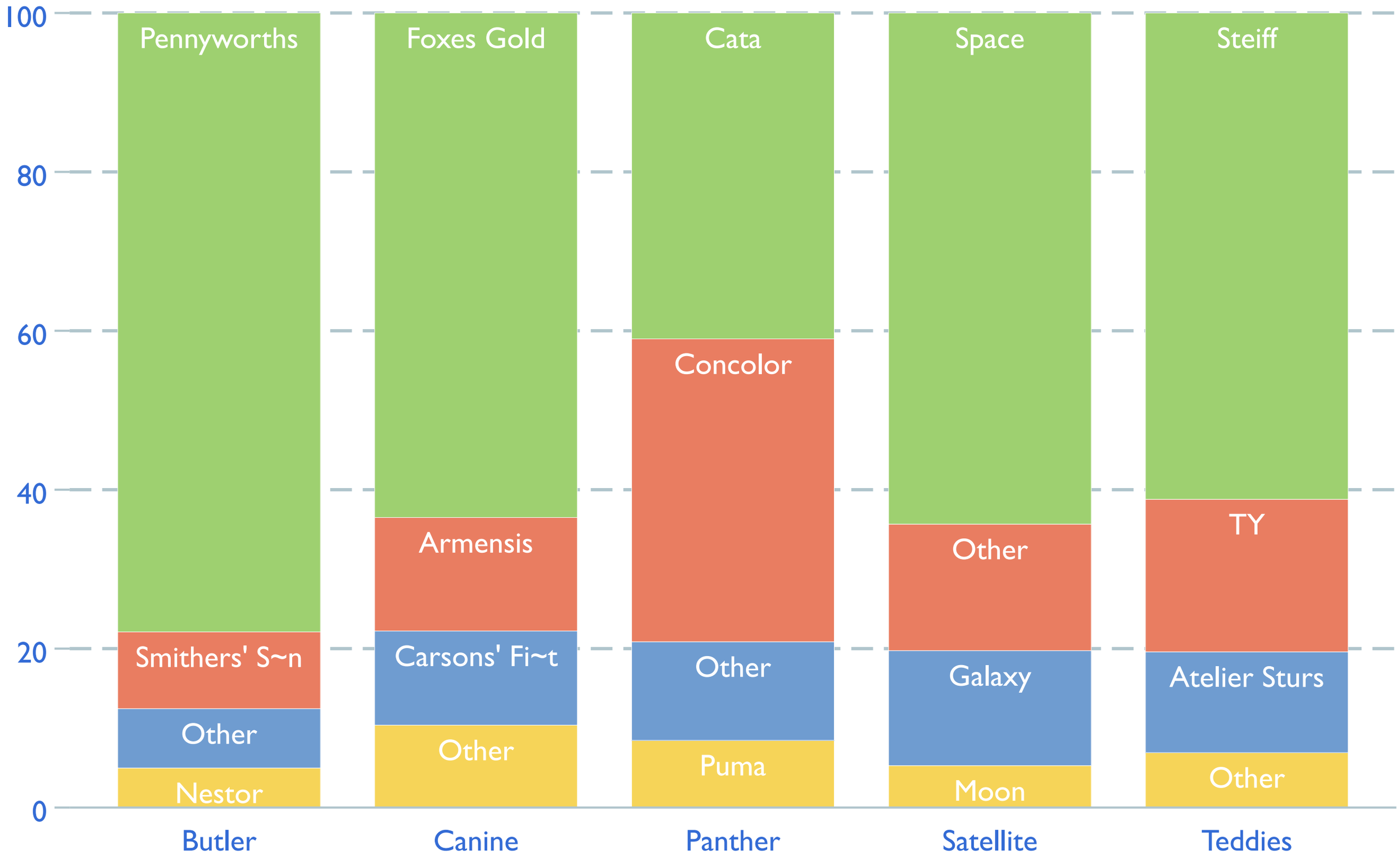
# Outline

1. Graphics in Stata **don't suck**.
2. Follow some **good practices** when managing data.
3. **Practice** with `import excel`, `reshape`, `merge`, `append` and `collapse`.
4. Use **schemes** to override Stata's ugly default graphs.
5. Create **publication quality tables** with `tabout`.

# Example graphs

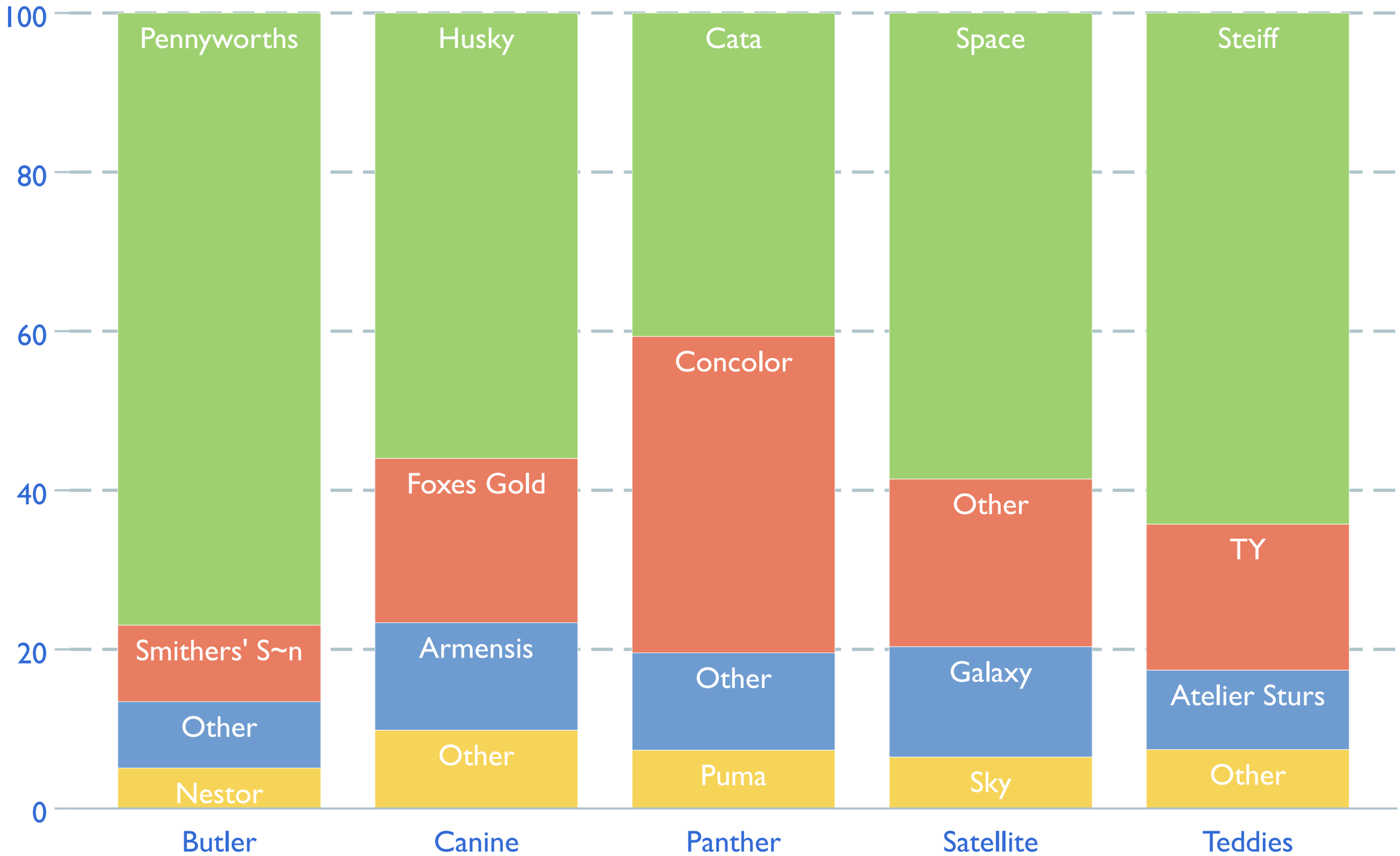
# Sales by brand name

Percent of total sales (EUR), North West European Market



# Sales by brand name

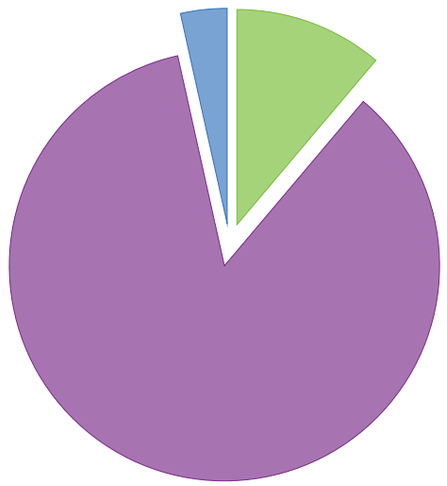
Percent of total sales (KG), North West European Market



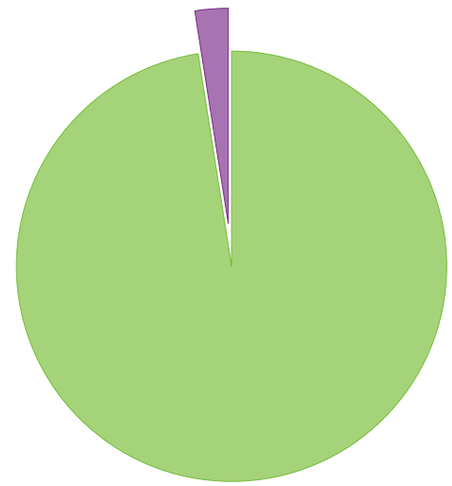
# Sales by size

Percent of total sales (EUR), North West European Market

Butler

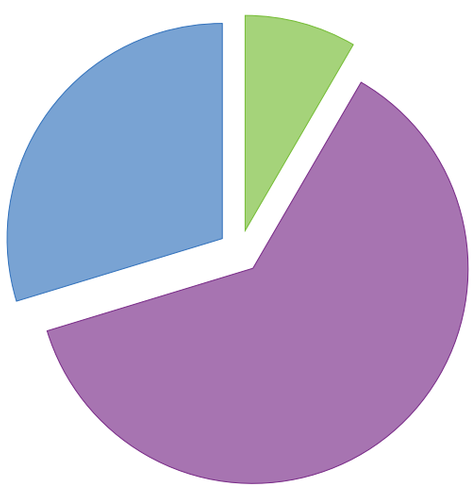


Canine

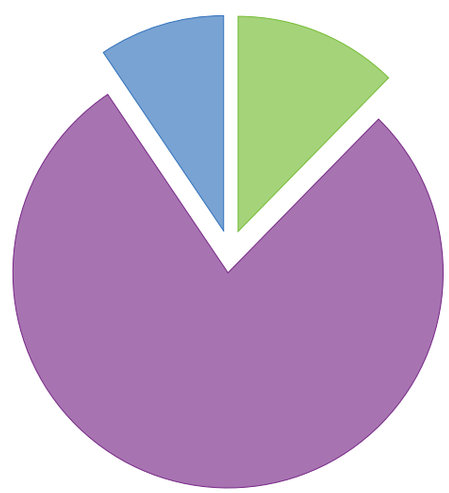


Karl Fazer

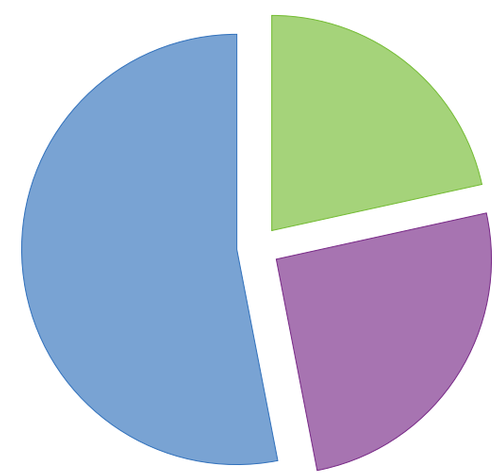
Panther



Satellite



Teddies



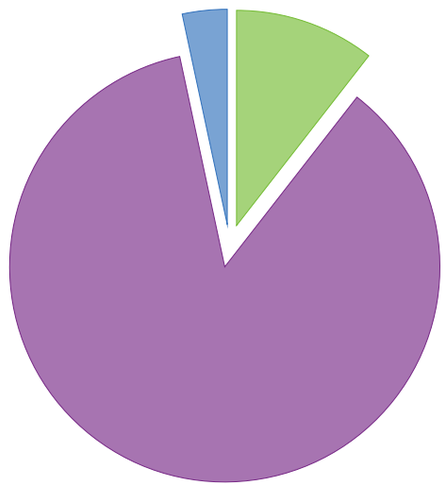
< 75 g  75-100 g  > 100 g

Graphs by Manufacturer

# Sales by size

Percent of total sales (KG), North West European Market

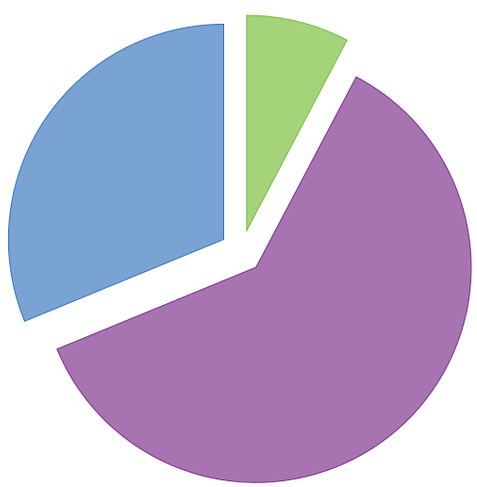
Butler



Canine

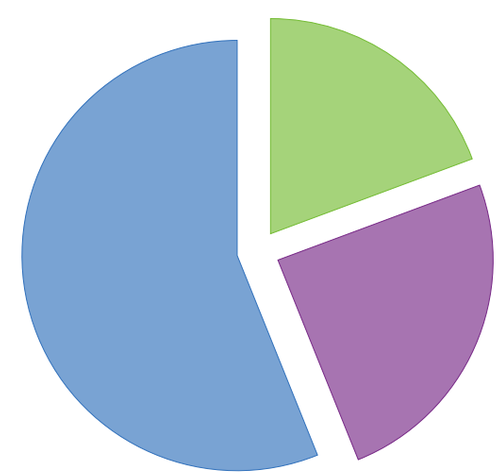
Karl Fazer

Panther



Satellite

Teddies

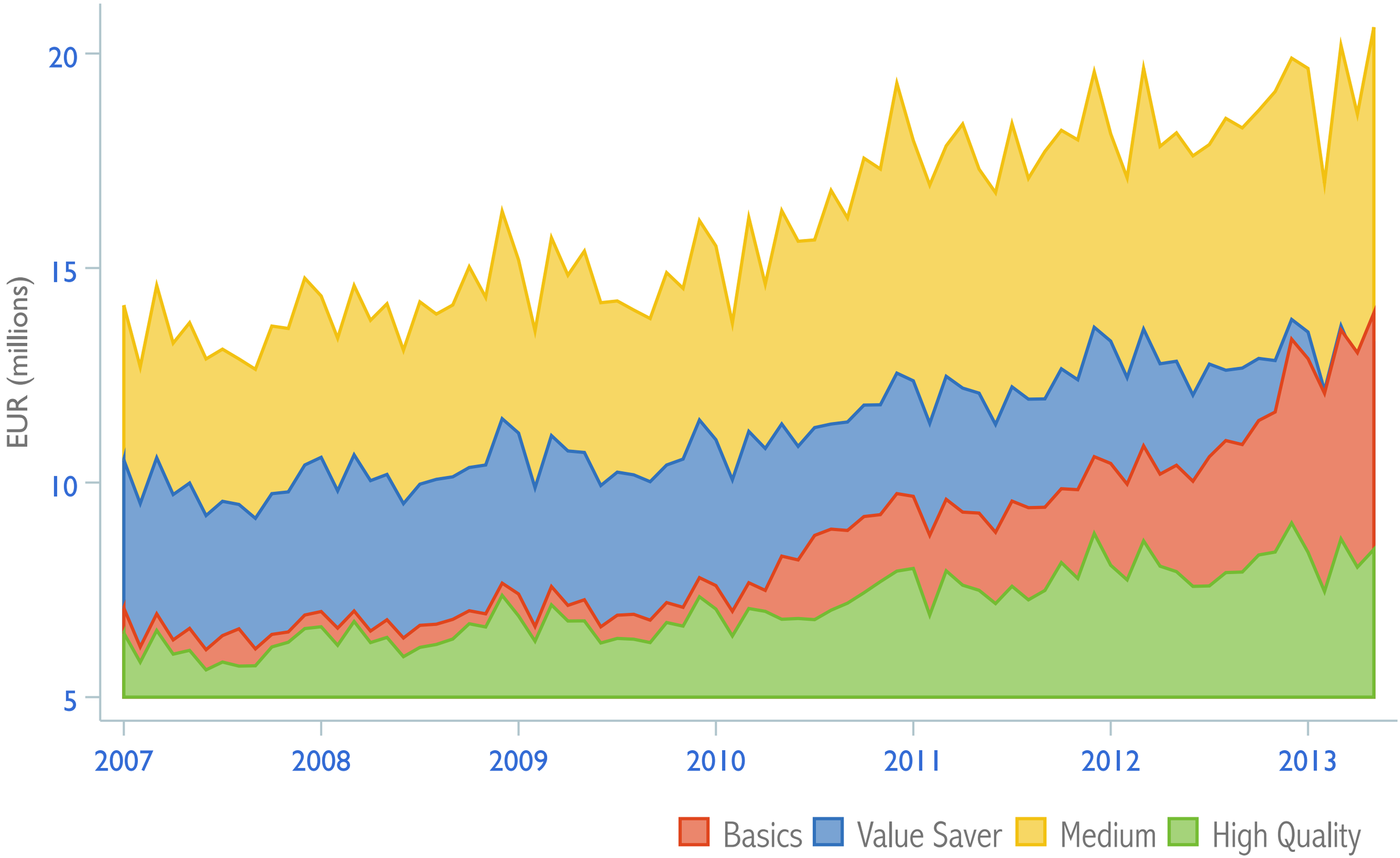


< 75 g  75-100 g  > 100 g

Graphs by Manufacturer

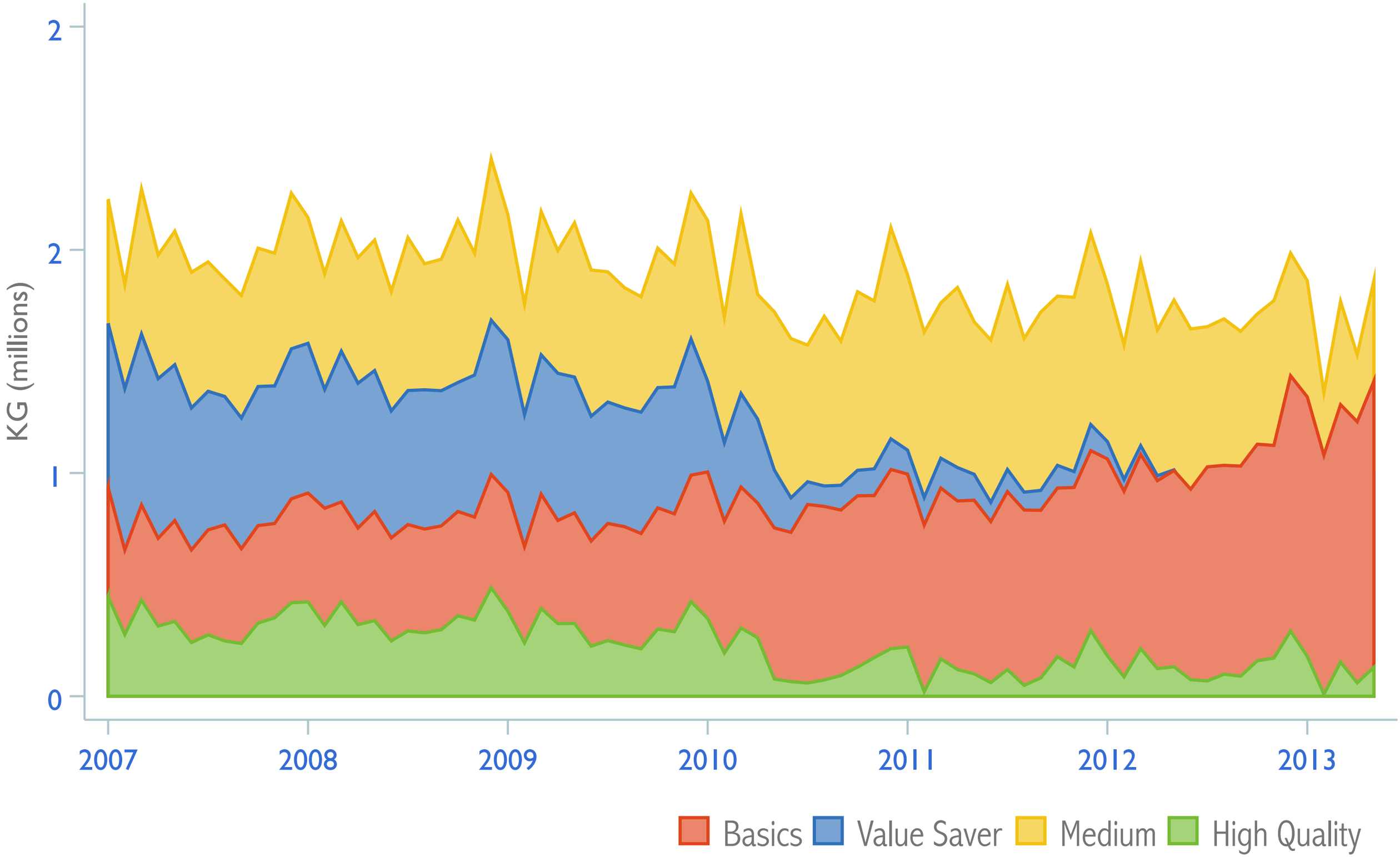
# Total sales by segment

Monthly data (EUR), North West European Market



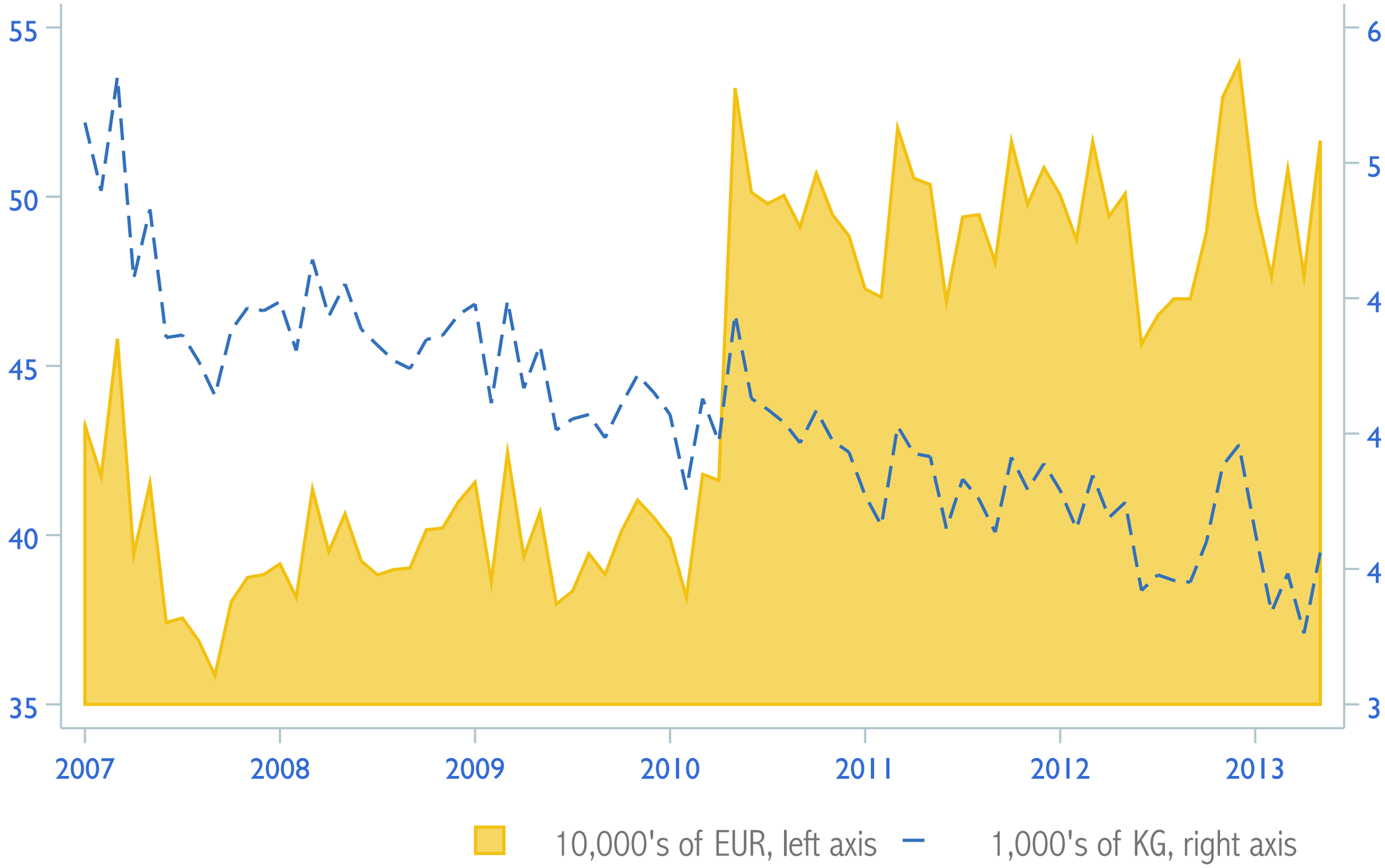
# Total sales by segment

Monthly data (KG), North West European Market



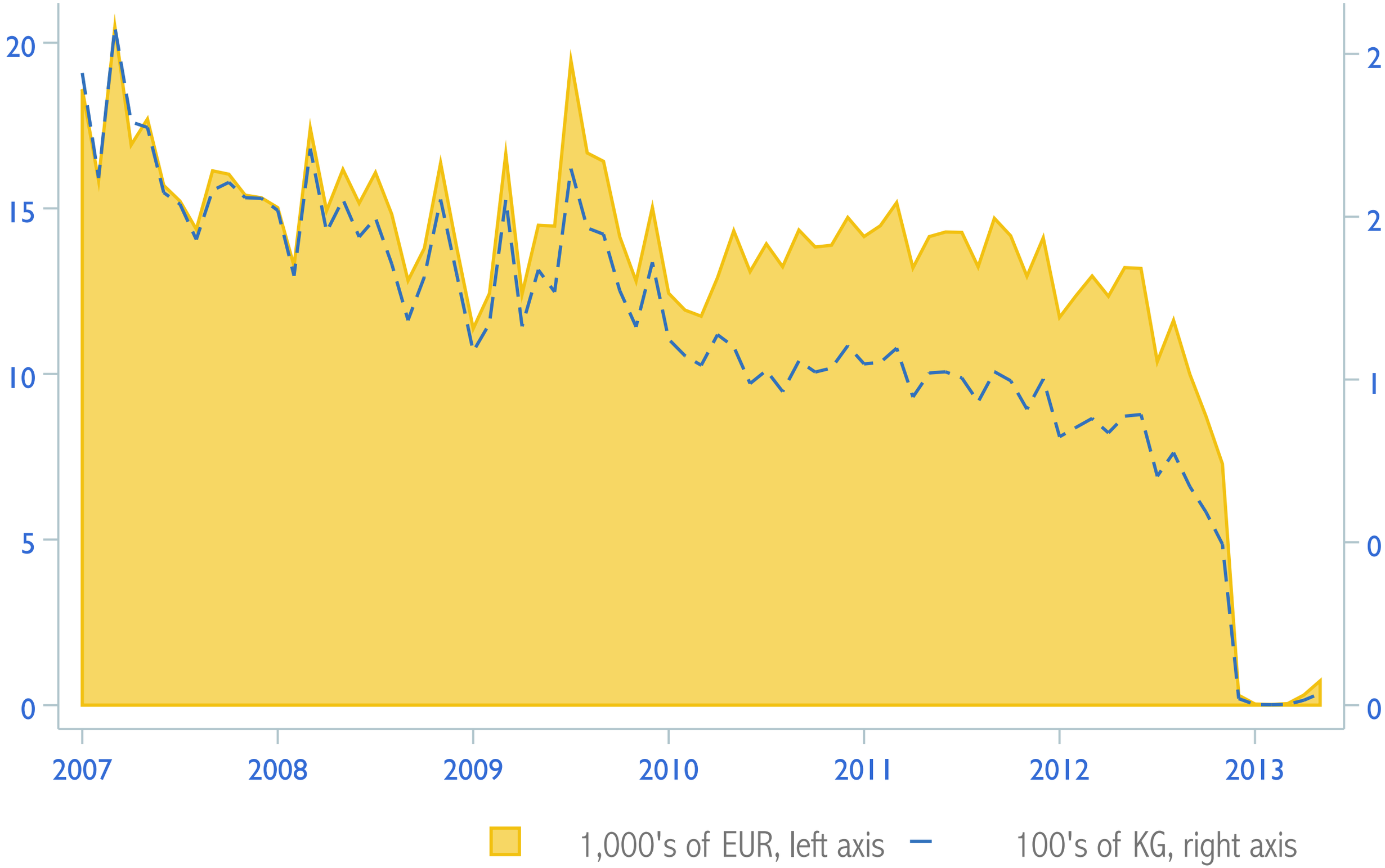
# Total sales Cata

50 grams, North West European Market



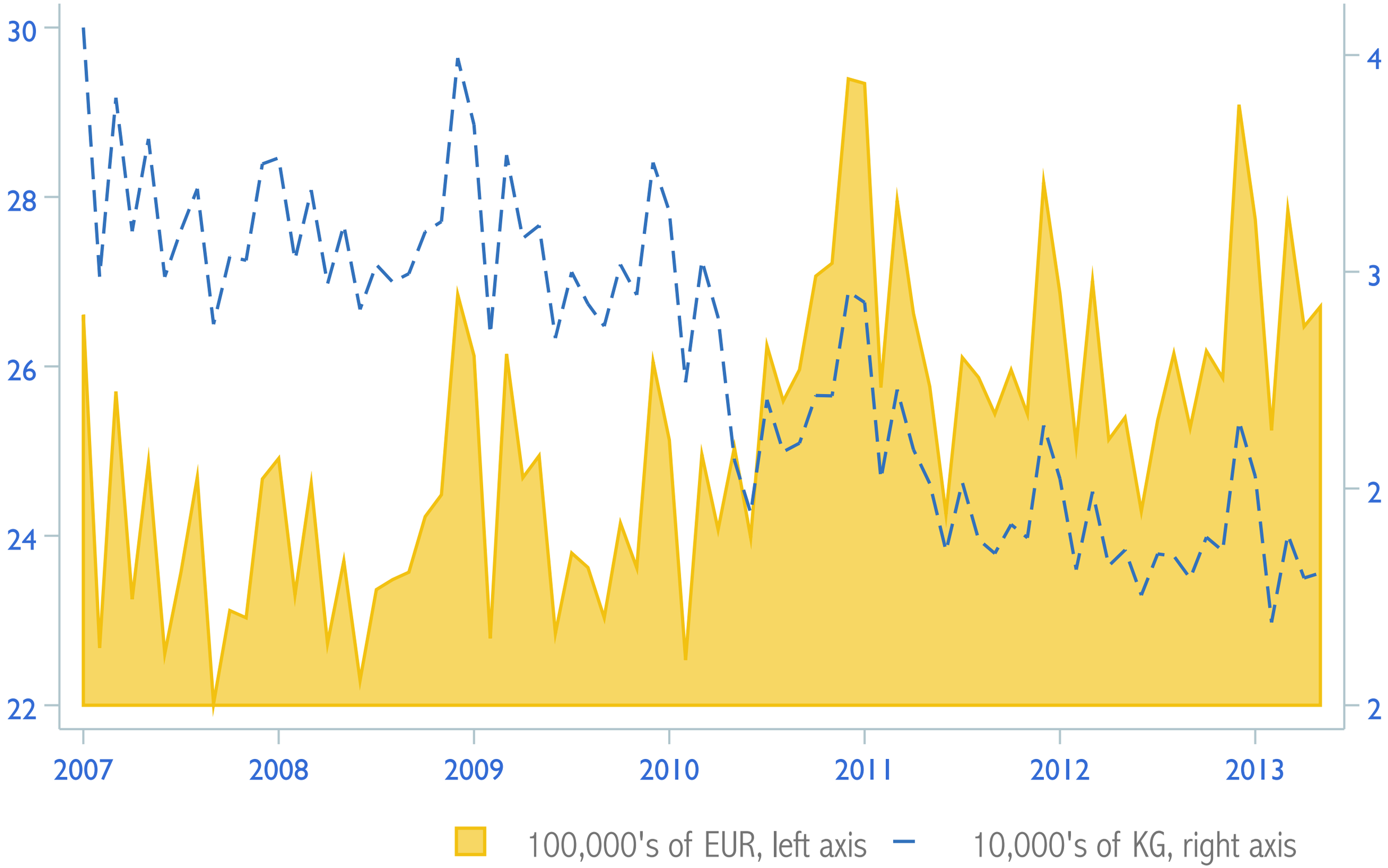
# Total sales Mini Vanilla Bears

50 grams, North West European Market



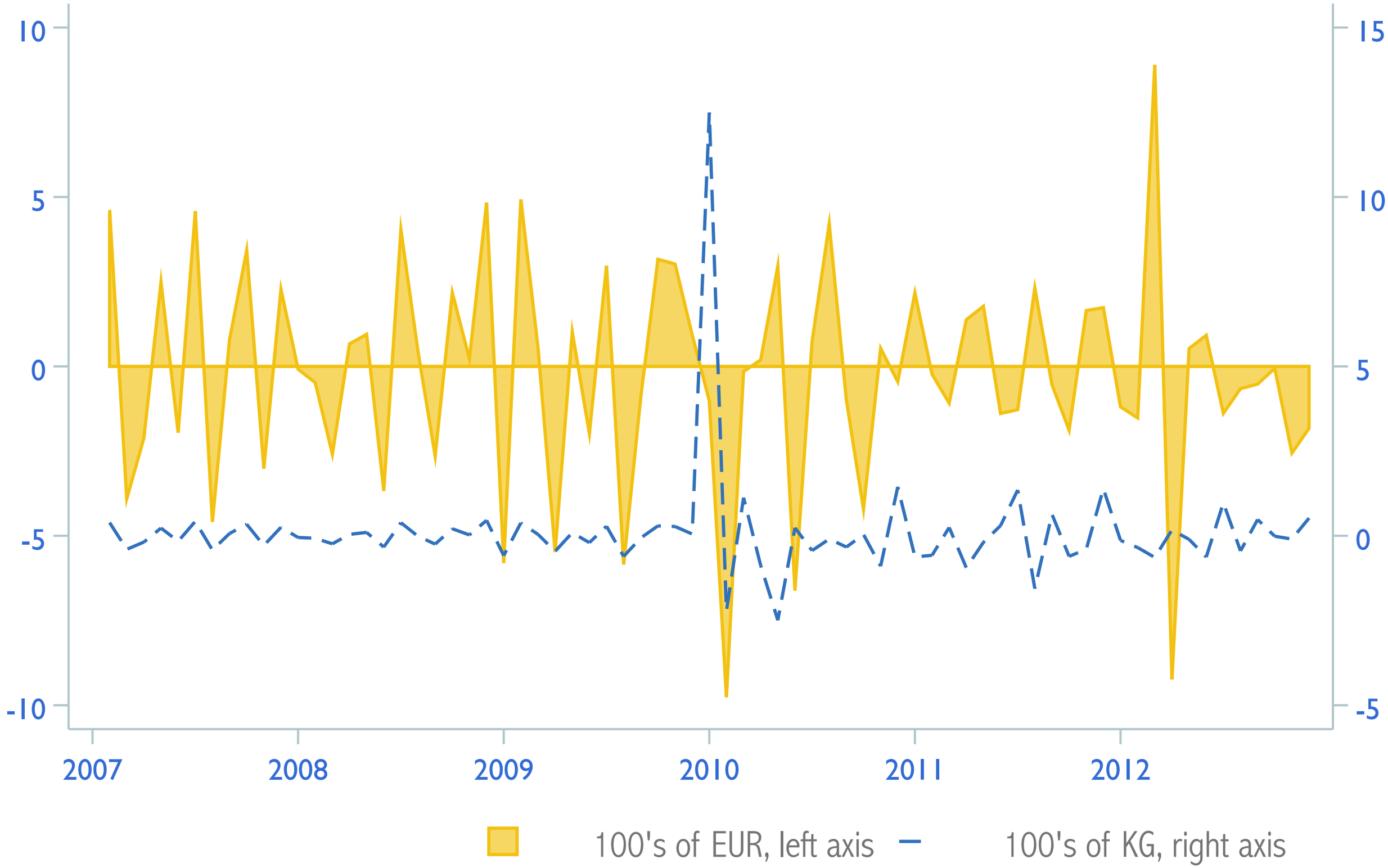
# Total sales Steiff

200 grams, North West European Market



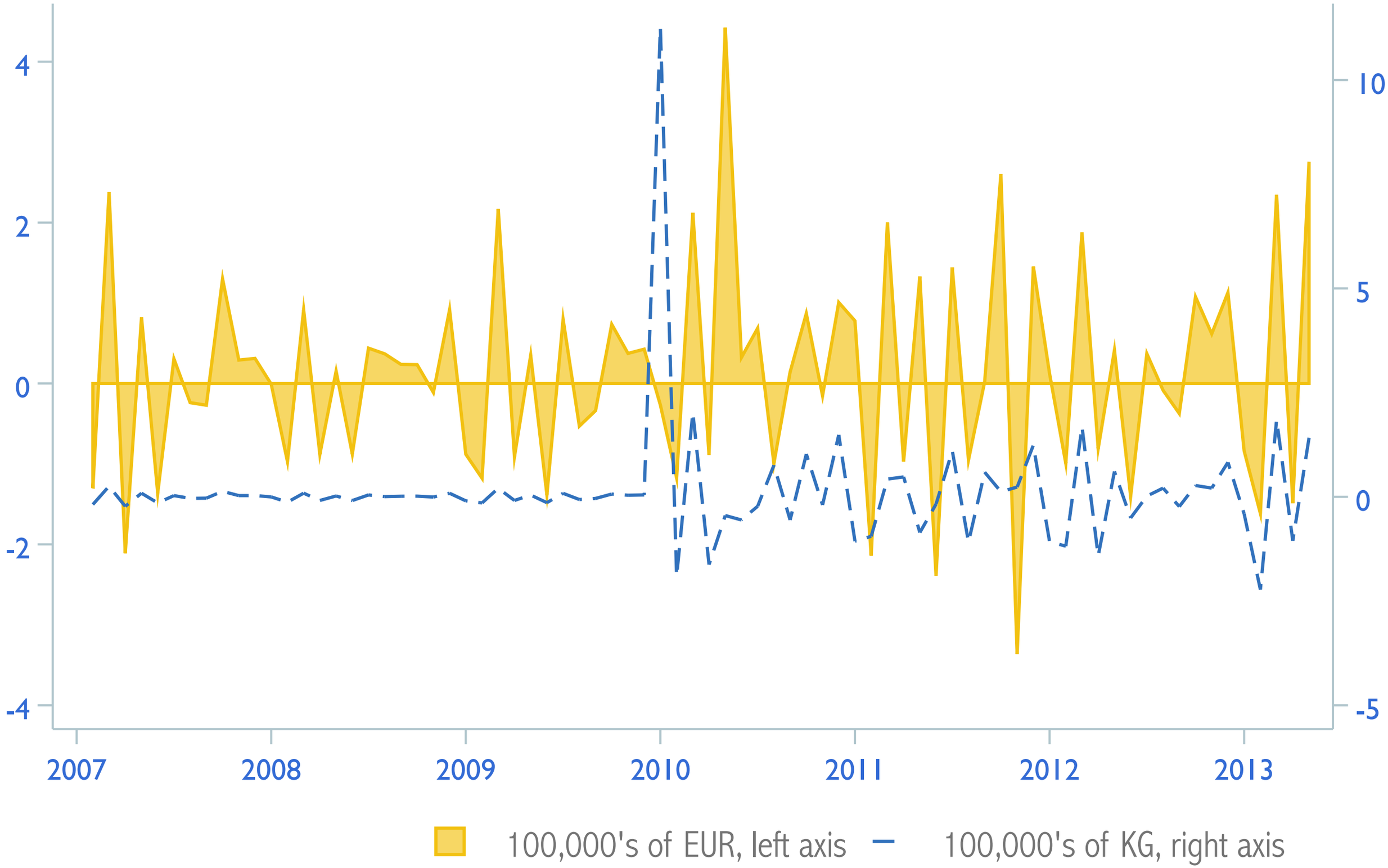
# Total sales Aero

50 grams, North West European Market



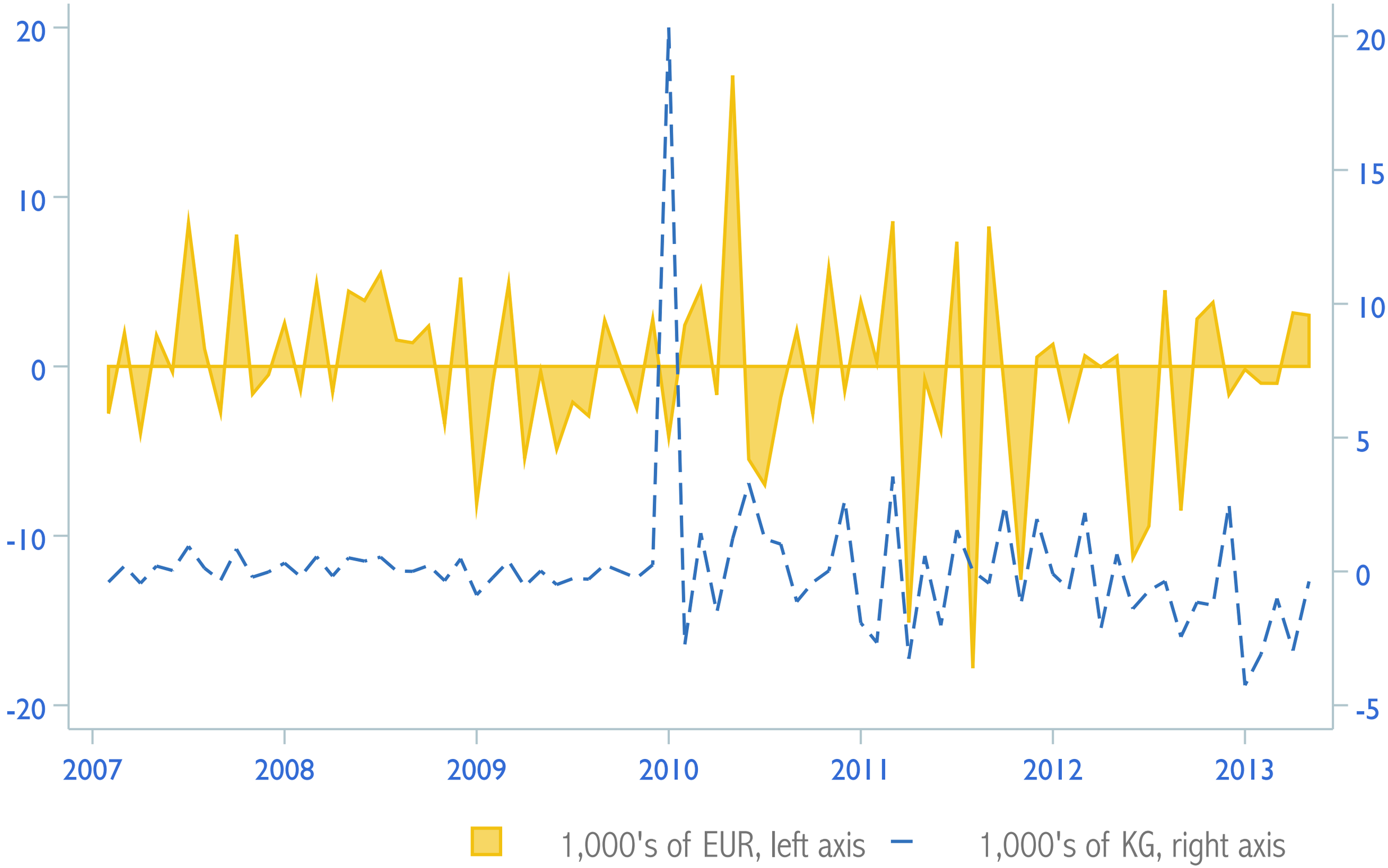
# Total sales Space

50 grams, North West European Market



# Total sales Tether White

50 grams, North West European Market



# Managing data

- **Decide what you want.**
- **Start easy** and don't plan too much.
- **Use a single do-file** that only import's merges, appends, reshapes and cleans the data.
- **Don't alter the raw data** or accidentally save over it.
- **Test your do-file** often and move slowly.
- **Check your work** with Excel.

Source: Social Science Computing Cooperative, University of Wisconsin, Madison

# Import & clean the data

- **Import** from Excel
- **Drop** everything superfluous
- **Label** the dataset
- **Destring** `barsize`.
- **Encode** `segment`.
- **Compress & save** the dataset.

# Import from Excel

```
import excel "Value - 2007-09.xlsx", ///
    cellrange(A3)                     ///
    firstrow                          ///
    case(lower)                       ///
    sheet("Sheet1")                   ///
    clear
```

# Drop what’s superfluous

```
drop f h i j au-ce
drop in 1/147
```

# Label the dataset

```
label data "North west European Market"
```

# Destring `barsize`

```
destring barsize, ignore("g") replace
```

# Encode segment.

```
label define quality ///
    1 "Basics"        ///
    2 "Value Saver"   ///
    3 "Medium"        ///
    4 "High Quality"
encode segment, gen(tmp) label(quality)
drop segment
rename tmp segment
```

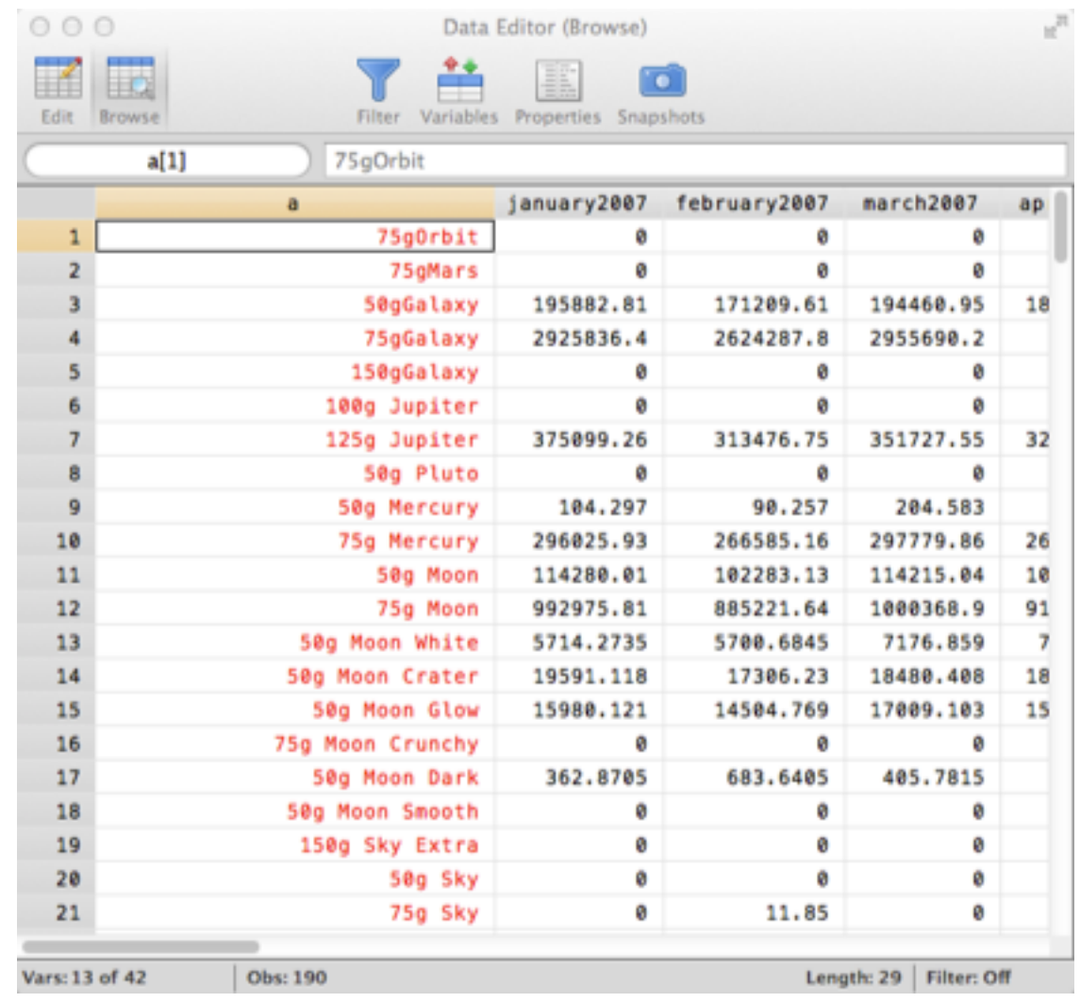
# Compress & save

```
compress
save value2007.dta
```

# Reshaping data

# Wide data

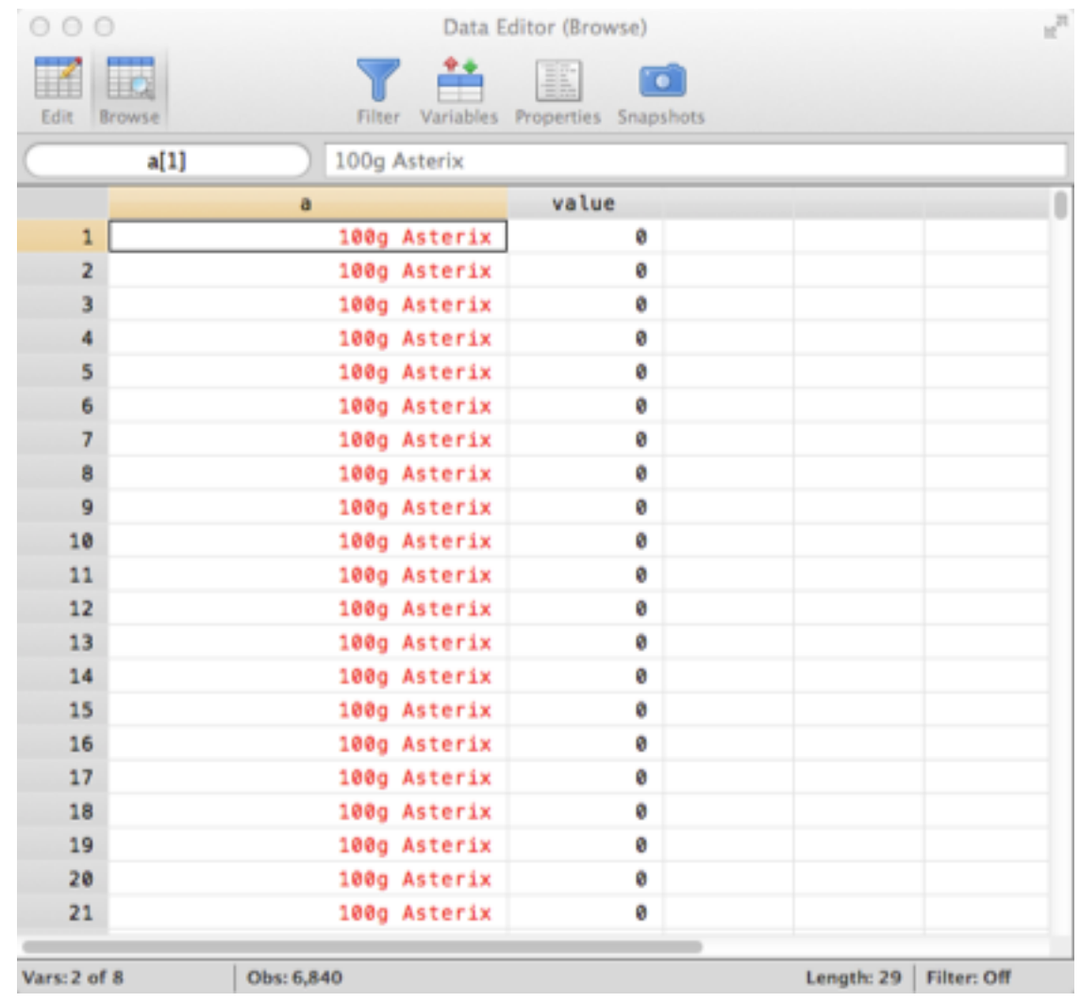
- Few observations.
- Many variables.
- One variable uniquely identifies each observation.
- Many variables contain data across one dimension.




| | a | january2007 | february2007 | march2007 | ap |
|---|---|---|---|---|---|
| 1 | 75gOrbit | 0 | 0 | 0 | |
| 2 | 75gMars | 0 | 0 | 0 | |
| 3 | 50gGalaxy | 195882.81 | 171209.61 | 194460.95 | 18 |
| 4 | 75gGalaxy | 2925836.4 | 2624287.8 | 2955690.2 | |
| 5 | 150gGalaxy | 0 | 0 | 0 | |
| 6 | 100g Jupiter | 0 | 0 | 0 | |
| 7 | 125g Jupiter | 375099.26 | 313476.75 | 351727.55 | 32 |
| 8 | 50g Pluto | 0 | 0 | 0 | |
| 9 | 50g Mercury | 104.297 | 90.257 | 204.583 | |
| 10 | 75g Mercury | 296025.93 | 266585.16 | 297779.86 | 26 |
| 11 | 50g Moon | 114280.01 | 102283.13 | 114215.04 | 10 |
| 12 | 75g Moon | 992975.81 | 885221.64 | 1000368.9 | 91 |
| 13 | 50g Moon White | 5714.2735 | 5700.6845 | 7176.859 | 7 |
| 14 | 50g Moon Crater | 19591.118 | 17306.23 | 18480.408 | 18 |
| 15 | 50g Moon Glow | 15980.121 | 14504.769 | 17009.103 | 15 |
| 16 | 75g Moon Crunchy | 0 | 0 | 0 | |
| 17 | 50g Moon Dark | 362.8705 | 683.6405 | 405.7815 | |
| 18 | 50g Moon Smooth | 0 | 0 | 0 | |
| 19 | 150g Sky Extra | 0 | 0 | 0 | |
| 20 | 50g Sky | 0 | 0 | 0 | |
| 21 | 75g Sky | 0 | 11.85 | 0 | |

# Long data

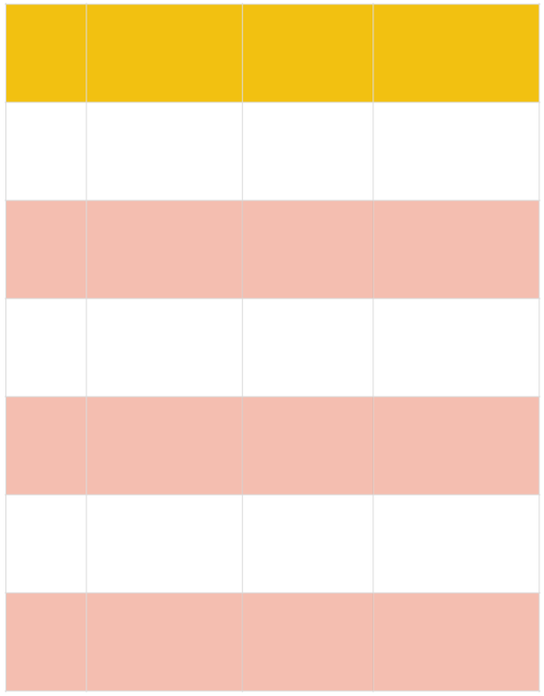
- Many observations.
- Few variables.
- No variable uniquely identifies each observation.
- Each variable contains data over multiple dimensions.




| | a | value |
|---|---|---|
| 1 | 100g Asterix | 0 |
| 2 | 100g Asterix | 0 |
| 3 | 100g Asterix | 0 |
| 4 | 100g Asterix | 0 |
| 5 | 100g Asterix | 0 |
| 6 | 100g Asterix | 0 |
| 7 | 100g Asterix | 0 |
| 8 | 100g Asterix | 0 |
| 9 | 100g Asterix | 0 |
| 10 | 100g Asterix | 0 |
| 11 | 100g Asterix | 0 |
| 12 | 100g Asterix | 0 |
| 13 | 100g Asterix | 0 |
| 14 | 100g Asterix | 0 |
| 15 | 100g Asterix | 0 |
| 16 | 100g Asterix | 0 |
| 17 | 100g Asterix | 0 |
| 18 | 100g Asterix | 0 |
| 19 | 100g Asterix | 0 |
| 20 | 100g Asterix | 0 |
| 21 | 100g Asterix | 0 |

# From wide to long

| id | sex | inc80 | inc81 |
|---|---|---|---|
| 1 | 0 | 5000 | 5500 |
| 2 | 1 | 2000 | 2200 |
| 3 | 0 | 3000 | 2000 |



# From wide to long

| id | sex | inc80 | inc81 |
|---|---|---|---|
| 1 | 0 | 5000 | 5500 |
| 2 | 1 | 2000 | 2200 |
| 3 | 0 | 3000 | 2000 |

| id | year | sex | inc |
|---|---|---|---|
| 1 | 80 | 0 | 5000 |
| 1 | 81 | 0 | 5500 |
| 2 | 80 | 1 | 2000 |
| 2 | 81 | 1 | 2200 |
| 3 | 80 | 0 | 3000 |
| 3 | 81 | 0 | 2000 |

# Syntax: wide to long

$x_{ij}$

i stub$_j$ stub$_j$

| id | sex | inc80 | inc81 |
|---|---|---|---|
| 1 | 0 | 5000 | 5500 |
| 2 | 1 | 2000 | 2200 |
| 3 | 0 | 3000 | 2000 |

j

reshape

i j stub

| id | year | sex | inc |
|---|---|---|---|
| 1 | 80 | 0 | 5000 |
| 1 | 81 | 0 | 5500 |
| 2 | 80 | 1 | 2000 |
| 2 | 81 | 1 | 2200 |
| 3 | 80 | 0 | 3000 |
| 3 | 81 | 0 | 2000 |

```
reshape long stub, i(i) j(j)
```

# Exercise

- Load `reshape1` (using `webuse`) and drop `ue80`, `ue81` and `ue82`.

```
webuse reshape1, clear
drop ue*
```

- Is the data long or wide? Convert to the other form.

```
reshape long inc, i(id) j(year)
```

- Use a shortcut to convert the data back again.

```
reshape wide
```

# Exercise

- Load `reshape1` again, but don't drop anything.

```
webuse reshape1
```

- Reshape from wide to long.

```
reshape long inc ue, i(id) j(year)
```

- Use a shortcut to convert it back to long.

```
reshape long inc ue, i(id) j(year)
```

# Exercise

- Load `reshape2` from the web.

```
webuse reshape2
```

- Try to reshape from wide to long.

```
reshape long inc ue, i(id) j(year)
```

- Why did you get an error?

# Exercise

- Load `reshape1` from the web and drop `ue81`.

```
webuse reshape1
drop ue81
```

- Reshape from wide to long.

```
reshape long inc ue, i(id) j(year)
```

- How did `reshape` handle the missing `ue81`?

- Convert the data back again. What happens to `ue81`?

```
reshape wide
```

# Exercise

- Load `reshape3` from the web.

```
webuse reshape3
```

- Reshape from wide to long.

```
reshape long inc@r ue, i(id) j(year)
```

# Exercise

- Load `reshape4` from the web.

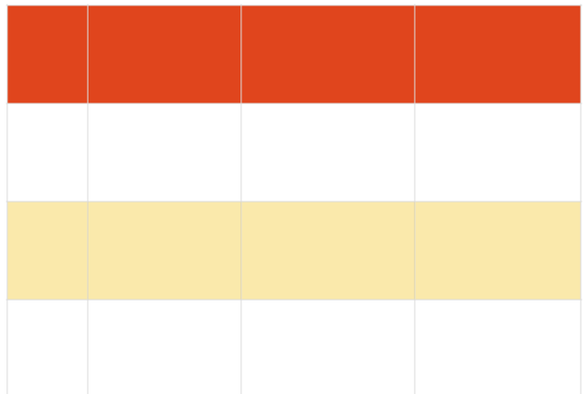
```
webuse reshape4
```

- Reshape from wide to long.

```
reshape long inc, i(id) j(sex) string
```

# From long to wide

| id | sex | kids | inc |
|---|---|---|---|
| 1 | f | 0 | 9000 |
| 1 | m | 0 | 2000 |
| 2 | f | 1 | 7000 |
| 2 | m | 1 | 1000 |
| 3 | f | 2 | 3000 |
| 3 | m | 2 | 8000 |



# From long to wide

| id | sex | kids | inc |
|---|---|---|---|
| 1 | f | 0 | 9000 |
| 1 | m | 0 | 2000 |
| 2 | f | 1 | 7000 |
| 2 | m | 1 | 1000 |
| 3 | f | 2 | 3000 |
| 3 | m | 2 | 8000 |

| id | kids | incm | incf |
|---|---|---|---|
| 1 | 0 | 2000 | 9000 |
| 2 | 1 | 1000 | 7000 |
| 3 | 2 | 8000 | 3000 |

# Syntax: long to wide

i j stub

| id | sex | kids | inc |
|---|---|---|---|
| 1 | f | 0 | 9000 |
| 1 | m | 0 | 2000 |
| 2 | f | 1 | 7000 |
| 2 | m | 1 | 1000 |
| 3 | f | 2 | 3000 |
| 3 | m | 2 | 8000 |

reshape

$x_{ij}$

i $stub_j$ $stub_j$

| id | kids | incm | incf |
|---|---|---|---|
| 1 | 0 | 2000 | 9000 |
| 2 | 1 | 1000 | 7000 |
| 3 | 2 | 8000 | 3000 |

j

```
reshape wide stub, i(i) j(j)
```

# Exercise

- Load `reshape6` from the web.

```
webuse reshape6
```

- Reshape from long to wide.

```
reshape wide inc ue, i(id) j(year)
```

- Why did you get an error?

# Exercise

- Load `reshapexp1` from the web.

```
webuse reshapexp1
```

- Try to reshape from long to wide.

```
reshape wide inc ue, i(id) j(year)
```

- Why did you get an error?

# `reshape` isn't working…

- Wide to long: does `i` uniquely identify every observation?

```
tabulate i
return list
```

# `reshape` isn't working…

- Long to wide: within each `i`, is there only one `j`?

```
egen unique = group(id year)
tabulate unique
return list
```

# `reshape` isn't working…

- Long to wide: do you mention all variables which vary within `i`?
- Either way: are `i` or `j` string variables?
- Type `reshape error`.

# Application to your data…

# Exercise

- Load `value2007.dta`.

```
use value2007, clear
```

- Reshape data from wide to long, where the new variable `value` contains all data from `january2007`, `february2007`, `march2007`, …

```
rename *20* value*20*
reshape long value, i(a) j(date) string
```

- Destring `date` and put it in month-year format.

```
generate tempdate = monthly(date,"MY")
drop date
rename tempdate date
format date %tm
```

# Exercise

- At the top of your do-file, change `"value - 2007-09.xlsx"` to `"volume - 2007-09.xlsx"`. Does it work?

- Using a `foreach` loop, import, clean, reshape and save `value - 2007-09.xlsx` *and* `volume - 2007-09.xlsx`.

```
foreach item in value volume {
  …
}
```

  - ➡ Be sure to replace `value` with `` `item' `` everywhere!

- Merge `value2007.dta` with `volume2007.dta`. Save the data as `data2007.dta`.

```
merge 1:1 a date using value2007
save data2007, replace
```

# Exercise

- Using a *separate* do-file, replicate everything for `value - 2010-13.xlsx` and `volume - 2010-13.xlsx`. Name the new dataset `data2013.dta`.

- Append `data2007.dta` to `data2013.dta`. Name the new dataset `data.dta`.

```
append using data2007
save data, replace
```

- Within a single loop, import, clean, reshape and save `value - 2007-09.xlsx`, `volume - 2007-09.xlsx` , `value - 2010-13.xlsx`, and `volume - 2010-13.xlsx`.

# Collapsing data

# Why do we want to do this?

- Collapsing data is Stata's version of pivot tables.
- It's a quick and dirty way to make graphs and tables.

# Exercise

- Create a dataset with the mean `volume` for each `date`.

```
collapse volume, by(date)
```

- Create a dataset with the mean `volume` and `value` for each `date`.

```
collapse volume value, by(date)
```

- Create a dataset with total `volume` and `value` for each `date` and `manufacturer`.

```
collapse volume value, by(date manufacturer)
```

- Create a dataset with the median `value` per `segment`.

```
replace value = . if value == 0
collapse (median) value, by(segment)
```

# Exercise

- Create a dataset with the count of `value` and `volume` by `year` and `barsize`.

```
recode date                ///
  (564/575 = 2007)         ///
  (576/587 = 2008)         ///
  (588/599 = 2009)         ///
  (600/611 = 2010)         ///
  (612/623 = 2011)         ///
  (624/635 = 2012)         ///
  (nonmissing = 2013), ///
  generate(year)
collapse (count) volume value, by(barsize year)
```

- Create a dataset with the standard deviation of `volume` and minimum of `value` for each brand per `year`; retain the `manufacturer` variable.

```
collapse (first) manufacturer (sd) value, by(brand year)
```

# Schemes

# What is a scheme?

- Schemes define the overall look of a graph.
- Within a scheme file, define graph colours, text sizes, backgrounds, etc.
- Stata's default schemes are ugly, but we can change that

# How do I make a scheme?

- Create a new file called `myscheme-scheme.scheme` and save it in your personal ado folder.
- Each entry in a scheme file specifies how a particular attribute of a graph element looks.
- First line should always be `#include s2color`.
- `help scheme` describes how to create your own schemes.
- `help scheme entries` lists all possible definitions to include in `myscheme-scheme.scheme`.

# Exercise

- Colour graph titles blue.

```
color heading blue
```

- Make graph titles very large.

```
gsize heading large
```

- Colour graph subtitles grey and put them in the north-east corner.

```
color subheading gs10
clockdir subtitle_position 1
```

- Colour the first plot orange.

```
color p1 orange
```

# Exercise

- Colour the background black.

```
color background black
```

- Colour grid-lines as RGB 200 200 200.

```
color major_grid “200 200 200"
```

- Make x-axis labels horizontal.

```
anglestyle vertical_tick horizontal
```

- Place graph legends in the south-east corner.

```
clockdir legend_position 4
```