UBS

Public

Przetwarzanie i wizualizacja danych w R

Adam Wróbel
Risk Modelling & Analytics Specialist

28 March 2017

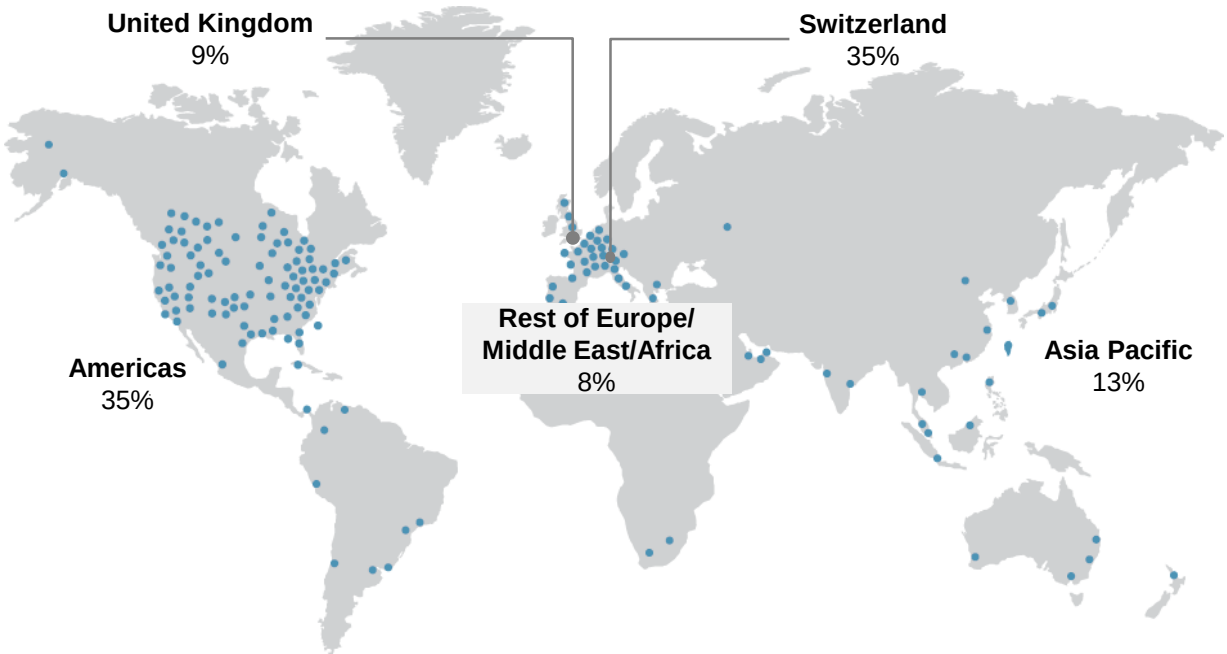
Get to know us

We are a premiere global financial services company.

We are well-established and thoroughly global

We're:

- a client-focused financial services firm with a 150-year history
- headquartered in Switzerland
- over 60,000 people in more than 50 countries
- committed to our wealth management businesses and our universal bank in Switzerland, along with our asset management and investment banking businesses.



UBS as an employer

Globally

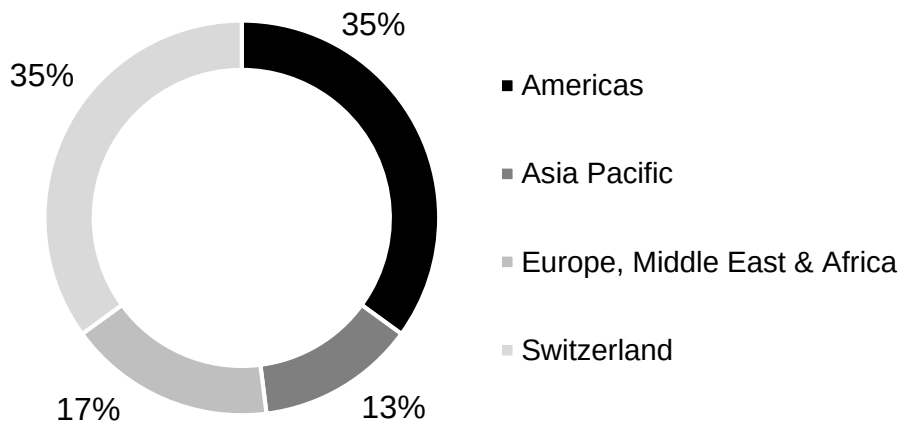
~60,000	147	~130	893	in 56	~9 years
employees	nationalities	languages	offices	countries	average tenure at the firm

We

have employees in all career stages, from apprentices, interns and trainees to highly experienced professionals, specialists and industry leaders

continuously improve through individual and business training – including advisory and leadership courses and a wide range of eLearning options

Where we work



UBS Business Solutions Centers

We

- are integral part of the bank
- are part of a global network of services
- leverage the synergies of pooled expertise

And this is to:

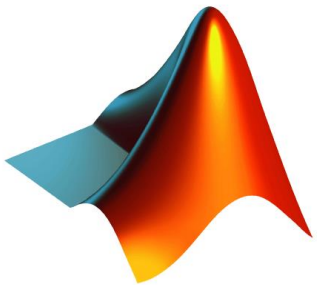
- improve service
- increase efficiency
- reduce risk

UBS

Narzędzia

Jakie narzędzia są wykorzystywane przez quantów w bankach

- Prototypowanie modeli odbywa się najczęściej w R, SAS, Matlab
- Produkcyjnie wersje modeli są najczęściej pisane w C++ oraz C#
- Cześć modeli (głównie statystycznych) ma produkcyjne implementacje w R, SAS, Matlab
- Inne środowiska/języki, które również są popularne to Python i VBA
- Moim zdaniem najbardziej przyszłościowe są:
  - R
  - Python




Cykl życia modelu w banku

Kroki:

- Zdiagnozowanie potrzeby zbudowania nowego modelu – np. w wyniku regulacji
- Zdobycie danych
- Wstępne przetwarzanie danych
- Poznanie danych:
  - statystyki/agregaty
  - wizualizacja
  - testy
- Przetwarzanie danych, aby zdefiniować na nich model
- Budowa modelu
  - ocena i wybór modelu
  - dokumentacja modelu
- Walidacja modelu

Przetwarzanie danych z pakietem dplyr

dplyr

- input %>% funkcja
  - jest tożsame z funkcja(input)
- Funkcje:

  %>% select

  %>% filter

  %>% mutate

  %>% group_by

  %>% summarize

- Skrypt: *przetwarzanie_danych_dplyr.R*

dplyr - wprowadzenie

```
# load library

library(dplyr)


# pipe notation: %>%

output <- input %>% function()


# example

mean_vector <- vector %>% mean()
```

```
> vector
[1] 1 2 3 4 5
> mean_vector
[1] 3
```

UBS

dplyr - select i filter

```
# select - selecting columns/variables

input %>% select(column1, column2)

iris %>% select(Petal.Length, Petal.Width, Species)
```

```
    Petal.Length Petal.Width    Species
1            1.4         0.2     setosa
2            1.4         0.2     setosa
3            1.3         0.2     setosa
```

```
# filter - filtering on some condition

input %>% filter(type == "type A")

iris %>% filter(Species == "versicolor")
```

```
   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1           7.0         3.2          4.7         1.4 versicolor
2           6.4         3.2          4.5         1.5 versicolor
3           6.9         3.1          4.9         1.5 versicolor
4           5.5         2.3          4.0         1.3 versicolor
```

dplyr - arrange i mutate

```
# arrange - sorting on given column

input %>% arrange(column1)

iris %>% arrange(Petal.Length)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	4.6	3.6	1.0	0.2	setosa
2	4.3	3.0	1.1	0.1	setosa
3	5.8	4.0	1.2	0.2	setosa

```
# mutate - creating new variables

input %>% mutate(column3 = pmax(column1, column2))

iris %>% mutate(Petal.Length.squared = Petal.Length^2)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species	Petal.Length.squared
1	5.1	3.5	1.4	0.2	setosa	1.96
2	4.9	3.0	1.4	0.2	setosa	1.96
3	4.7	3.2	1.3	0.2	setosa	1.69

dplyr - group_by

```
# group_by & mutate

input %>% group_by(type) %>%

mutate(mean_per_type = mean(column1))


# iris example

iris %>% group_by(Species) %>%

mutate(mean_PL_per_Species = mean(Petal.Length))
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species	mean_PL_per_Species
1	5.1	3.5	1.4	0.2	setosa	1.462
2	4.9	3.0	1.4	0.2	setosa	1.462
3	4.7	3.2	1.3	0.2	setosa	1.462
50	5.0	3.3	1.4	0.2	setosa	1.462
51	7.0	3.2	4.7	1.4	versicolor	4.260
52	6.4	3.2	4.5	1.5	versicolor	4.260

dplyr - summarize

```
# group_by & summarize

input %>% group_by(type) %>%

summarize(mean_per_type = mean(column1))


# iris example

iris %>% group_by(Species) %>%

summarize(mean_PL_per_Species = mean(Petal.Length))
```

```
     Species mean_PL_per_Species
1     setosa                1.462
2 versicolor                4.260
3  virginica                5.552
```

dplyr - połączenie wszystkich elementów

```
iris %>% head
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
```

```
iris %>% mutate(Petal.Surface = Petal.Length*Petal.Width)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species Petal.Surface
1          5.1         3.5          1.4         0.2  setosa          0.28
2          4.9         3.0          1.4         0.2  setosa          0.28
3          4.7         3.2          1.3         0.2  setosa          0.26
```

```
iris %>% mutate(Petal.Surface = Petal.Length*Petal.Width) %>%
filter(Petal.Surface > 0.5)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species Petal.Surface
1          5.4         3.9          1.7         0.4  setosa          0.68
2          5.7         4.4          1.5         0.4  setosa          0.60
3          5.4         3.9          1.3         0.4  setosa          0.52
```

```
iris %>% mutate(Petal.Surface = Petal.Length*Petal.Width) %>%
filter(Petal.Surface > 0.5) %>% group_by(Species) %>%
summarize(mean_Petal.Surface = mean(Petal.Surface))
```

```
     Species mean_Petal.Surface
1     setosa             0.6720
2 versicolor             5.7204
3  virginica            11.2962
```

UBS

Wizualizacja danych z pakietem ggplot2

ggplot2

- Definiowanie dynamicznych elementów obiektu:
  - ggplot(dane, aes(x = date, y = price, group_by = index_name, colour = index_name)
- Definiowanie typu obiektu
  + geom_line()
  + geom_point()
  + geom_density()
- Definiowanie dodatkowych statycznych parametrów
  + geom_line(lty = „dotted”)
  + geom_point(colour = „red”)
  + geom_density(alpha = 0.7)
- Skrypt: *wizualizacja_danych_ggplot2.R*

Model do zbudowania

Regionalizacja indeksu na potrzeby testów stresu

- FED w ramach scenariuszy ekonomicznych, które publikuje podaje indeks cen nieruchomości na poziomie Stanów Zjednoczonych
- Ze względu na to, że nasz wysymulowany portfel hipotek nie jest równomiernie rozłożony w całych stanach chcemy wyznaczyć poziom indeksów regionalnych
- Zrobimy to korzystając z historycznej zależności między indeksem na poziomie całych stanów a indeksami z poszczególnych miast, które nas interesują

- Skypt: *regionalizacja_indeksu.R*

Polecana literatura/materiały

Materiały z wykładu: Programowanie w R :

https://github.com/AdamWrobel/Workshops/tree/master/AGH_matematyki_finansowej_2017

- "Przewodnik po pakiecie R", Przemysław Biecek, 2017
- "*R for Data Science*", Hadley Wickham, Garrett Grolemund, 2017
- *datacamp.com*
- *r-bloggers.com*

Informacje kontaktowe

Adam Wróbel

UBS
Risk Modelling & Analytics Specialist
Email: adam.wrobel@ubs.com

ubs.com/polandcareers