FRAUD DETECTION IN R

Introduction & Motivation

Bart Baesens
Professor Data Science at KU Leuven

Instructors

Bart Baesens

Professor Data Science at KU Leuven

Chairholder of BNP Paribas Fortis
Chair in Fraud Analytics

Instructors

Bart Baesens

Professor Data Science at KU Leuven

Chairholder of BNP Paribas Fortis Chair in Fraud Analytics

Tim Verdonck

Professor Data Science at KU Leuven

Chairholder of BNP Paribas Fortis Chair in Fraud Analytics

Instructors

Bart Baesens

Professor Data Science at KU Leuven

Chairholder of BNP Paribas Fortis Chair in Fraud Analytics

Tim Verdonck

Professor Data Science at KU Leuven

Chairholder of BNP Paribas Fortis Chair in Fraud Analytics

Sebastiaan Höppner

PhD researcher in Data Science at KU Leuven

What is fraud?

- Fraud is an uncommon, well-considered, imperceptibly concealed, time-evolving and often carefully organized crime which appears in many types and forms.

Impact of fraud

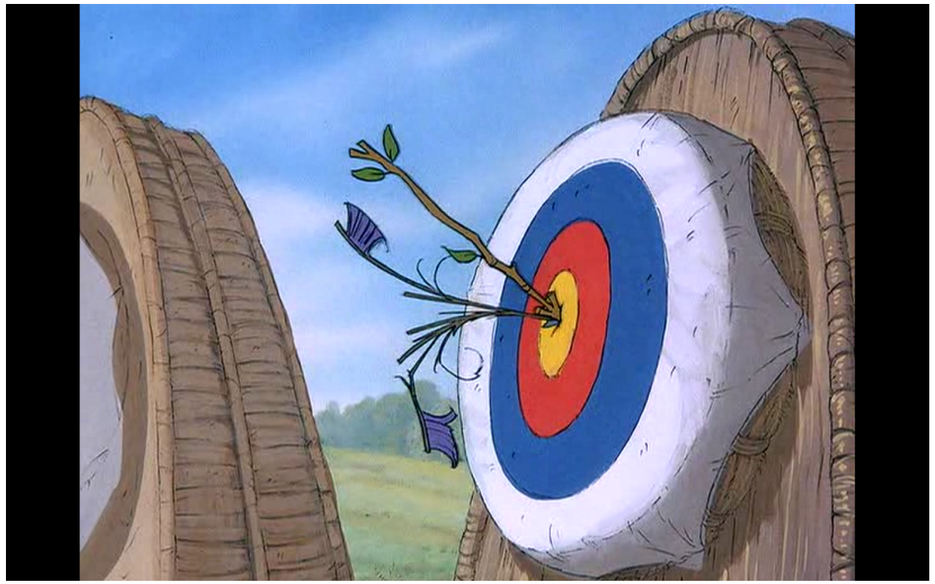
- Fraud is very rare, but cost of not detecting fraud can be huge!
- Examples:
  - Organizations lose **5% of their yearly revenues** to fraud
  - Money lost by businesses to fraud > $3.5 trillion each year
  - **Credit card** companies lose approximately **7 cents per $100** of transactions due to fraud
  - Fraud takes up **5-10% of the claim amounts** paid for **non-life insurance**

Types of fraud

- Anti-money laundering
- Check fraud
- (Credit) card fraud
- Click fraud
- Customs fraud
- Counterfeit
- Identity theft
- Insurance fraud
- Mortgage fraud
- Non-delivery fraud
- Online fraud
- Product warranty fraud
- Tax evasion
- Telecommunication fraud
- Theft of inventory
- Threat
- Ticket fraud
- Transit faud
- Wire fraud
- Workers compensation fraud

Key characteristics of successful fraud analytics models

- Statistical accuracy



Key characteristics of successful fraud analytics models

- Statistical accuracy
- Interpretability

Key characteristics of successful fraud analytics models

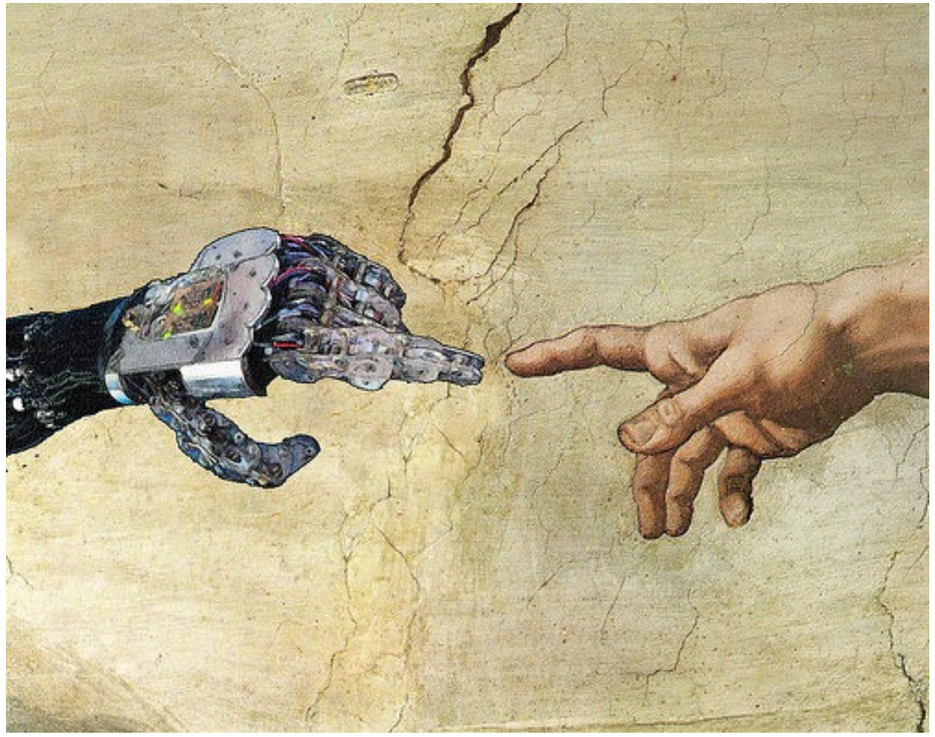
- Statistical accuracy
- Interpretability
- Regulatory compliance

Key characteristics of successful fraud analytics models

- Statistical accuracy
- Interpretability
- Regulatory compliance
- Economical impact

Key characteristics of successful fraud analytics models

- Statistical accuracy
- Interpretability
- Regulatory compliance
- Economical cost
- Complement expert based approaches with data-driven techniques



Challenges of fraud detection model

- Imbalance
  - e.g. in credit card fraud < 0.5% frauds typically

Challenges of fraud detection model

- Imbalance
  - e.g. in credit card fraud < 0.5% frauds typically
- Operational efficiency
  - e.g. in credit card fraud < 8 seconds decision time

Challenges of fraud detection model

- Imbalance
  - e.g. in credit card fraud < 0.5% frauds typically
- Operational efficiency
  - e.g. in credit card fraud < 8 seconds decision time
- Avoid harassing good customers

Imbalanced data

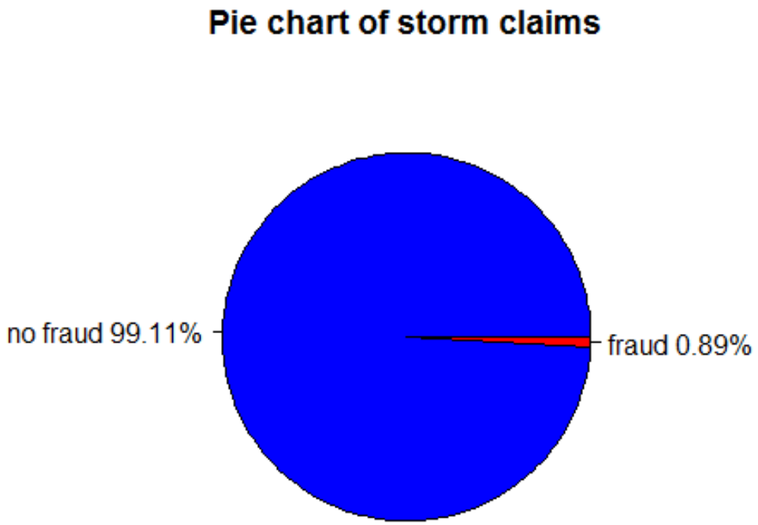
- After a major storm, an insurance company received many claims
  - Fraudulent claims are labeled with 1 and legitimate claims with 0
- The percentage of fraud cases in the data can be determined by using the functions `table()` and `prop.table()`
- `prop.table(table())` to determine percentage of fraud

```
> prop.table(table(fraud_label))

     0      1
0.9911 0.0089
```

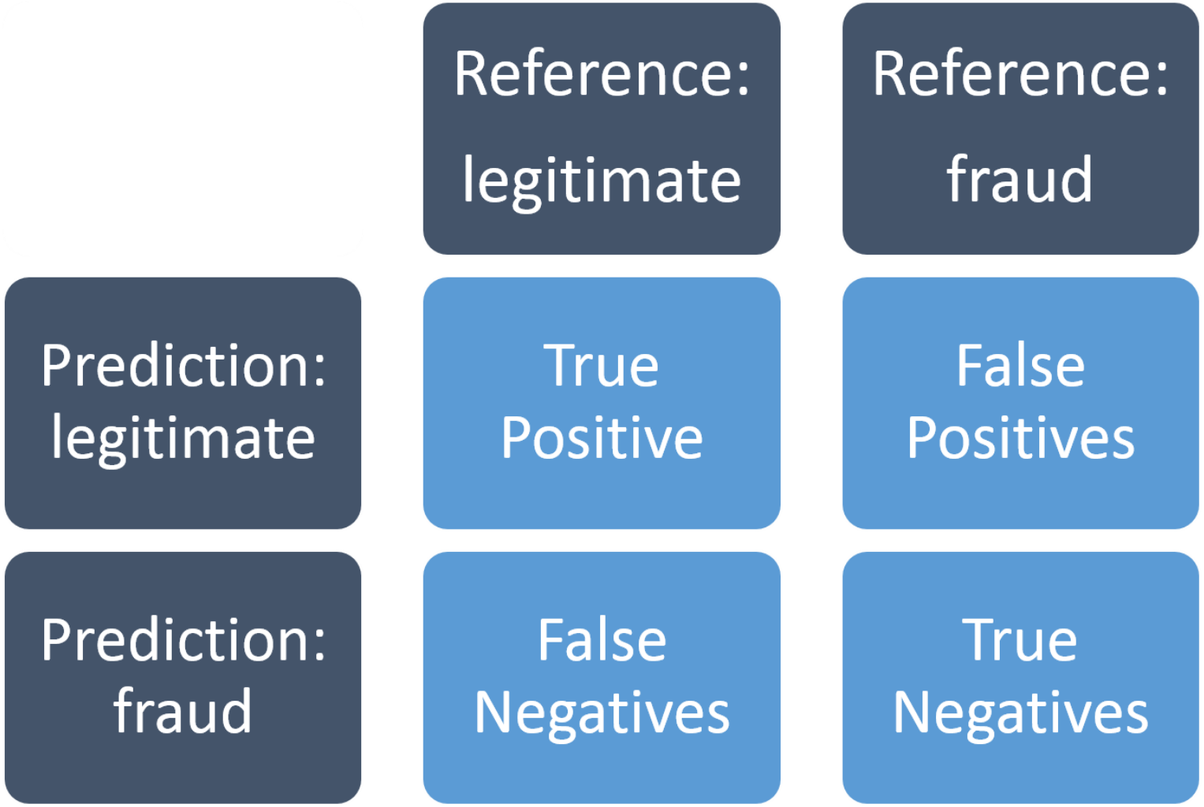
Imbalanced data

- Visualize imbalance with pie chart

```
> labels <- c("no fraud", "fraud")
> labels <- paste(labels, round(100*prop.table(table(fraud_label)), 2))
> labels <- paste0(labels, "%")
> pie(table(fraud_label), labels, col = c("blue", "red"),
      main = "Pie chart of storm claims")
```



Evaluation of supervised method: confusion matrix



Confusion matrix: claims example

- Suppose no detection model is used, so all claims are considered as legitimate:

```
> predictions <- rep.int(0, nrow(claims))
> predictions <- factor(predictions, levels = c("no fraud", "fraud"))
```

- Function confusionMatrix() from package caret:

```
> library(caret)
> confusionMatrix(data = predictions, reference = fraud_label)

 Confusion Matrix and Statistics

             Reference
 Prediction    0   1
            0 614  14
            1   0   0

 Accuracy : 0.9777
```

Total cost of not detecting fraud: claims example

- Total cost of fraud defined as the sum of fraudulent amounts

- Total cost if no fraud is detected:

```
> total_cost <- sum(claim_amount[fraud_label == "fraud"])
> print(total_cost)

[1] 2301508
```

FRAUD DETECTION IN R

Let's practice!

FRAUD DETECTION IN R

Time features

Bart Baesens
Professor Data Science at KU Leuven

Analyzing time

- Certain events are expected to occur at similar moments in time
- Example: customer making transactions at similar hours
- Aim: capture information about the time aspect by meaningful features
- Dealing with time can be tricky
  - 00:00 = 24:00
  - no natural ordering: 23:00 $<$, $>$ 01:00?

Mean of timestamps

- Do not use arithmetic mean to compute an average timestamp!
  - Example: transaction made at 01:00, 02:00, 21:00 and 22:00
  - arithmetic mean is 11:30, but no transfer was made close to that time!

```
> data(timestamps)
> head(timestamps)

[1] "20:27:28" "21:08:41" "01:30:16" "00:57:04" "23:12:14" "22:54:16"
```

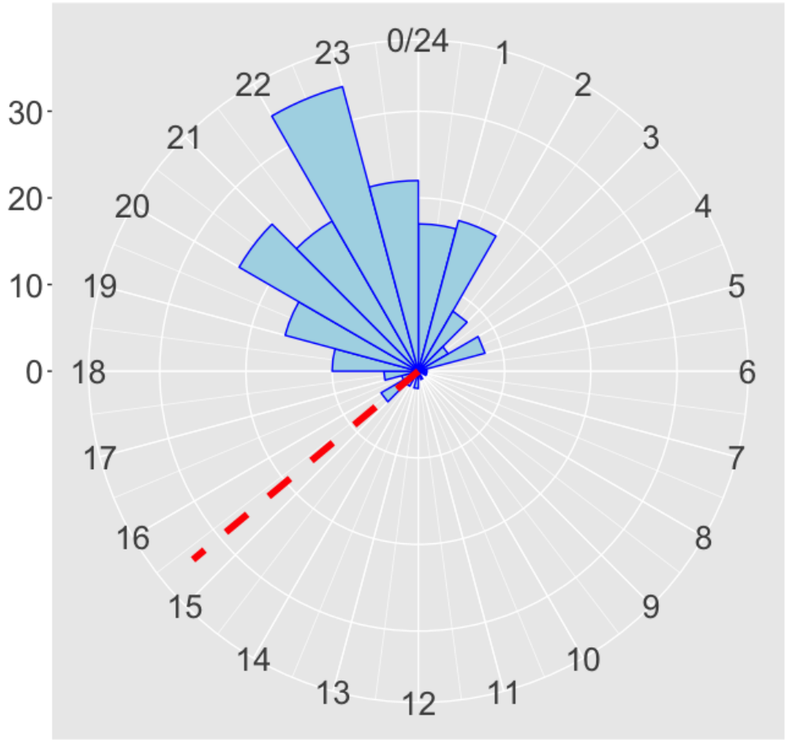
- Convert digital timestamps to decimal format in hours

```
> library(lubridate)
> ts <- as.numeric(hms(timestamps)) / 3600

> head(ts)

[1] 20.4577778 21.1447222  1.5044444  0.9511111 23.2038889 22.9044444
```

Circular histogram

```
> library(ggplot2)

> clock <- ggplot(data.frame(ts), aes(x = ts)) +
  geom_histogram(breaks = seq(0, 24), colour = "blue", fill = "lightblue") +
  coord_polar()

> arithmetic_mean <- mean(ts)

> clock + geom_vline(xintercept = arithmetic_mean,
                     linetype = 2, color = "red", size = 2)
```

Circular histogram with arithmetic mean



von Mises distribution

- Model time as a periodic variable using the von Mises probability distribution (Correa Bahnsen et al., 2016)
- Periodic normal distribution = normal distribution wrapped around a circle
- von Mises distribution of a set of timestamps $D = \{t_1, t_2, \ldots, t_n\}$

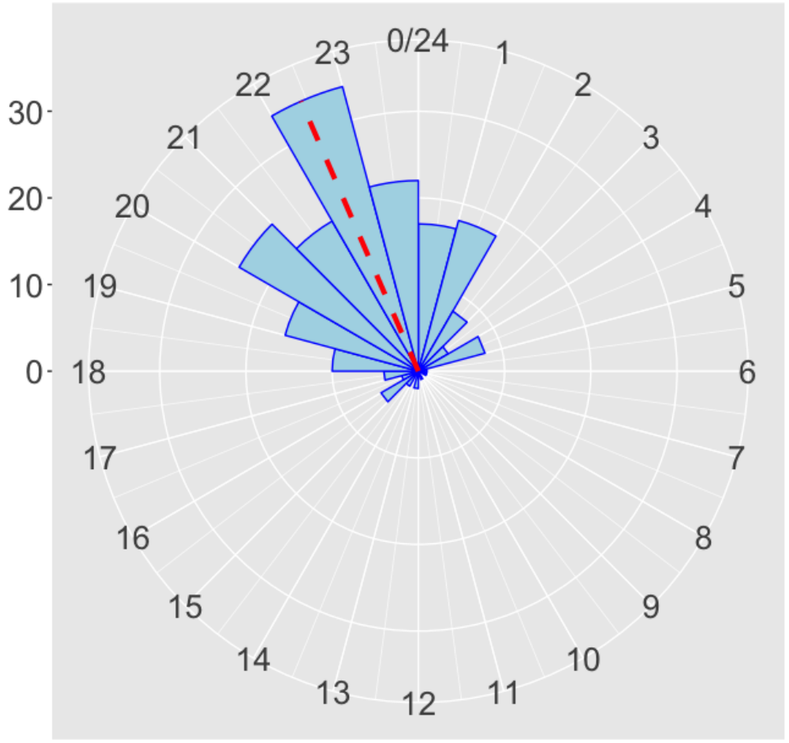
$$D \sim vonMises\left(\mu, \kappa\right)$$

- μ: periodic mean, measure of location, distribution is clustered around μ
- $1/\kappa$: periodic variance; κ is a measure of concentration

Estimating parameters μ and κ

```
# Convert the decimal timestamps to class "circular"
> library(circular)
> ts <- circular(ts, units = "hours", template = "clock24")

> head(ts)

Circular Data:
[1] 20.457889 21.144607  1.504422  0.950982 23.203917  4.904397

> estimates <- mle.vonmises(ts)
> p_mean <- estimates$mu %% 24
> concentration <- estimates$kappa
```

Circular histogram with periodic mean



Confidence interval

- Extract new features: confidence interval for the time of a transaction
- $S = \{x_i^{time} | i = 1, \ldots, n\}$: set of transactions made by the same customer

(1) Estimate $\mu(S)$ and $\kappa(S)$ based on S:

```
> estimates <- mle.vonmises(ts)
> p_mean <- estimates$mu %% 24
> concentration <- estimates$kappa
```

(2) Calculate the density (= likelihood) of the timestamps for the estimated von Mises distribution:

```
> densities <- dvonmises(ts, mu = p_mean, kappa = concentration)
```

Feature extraction

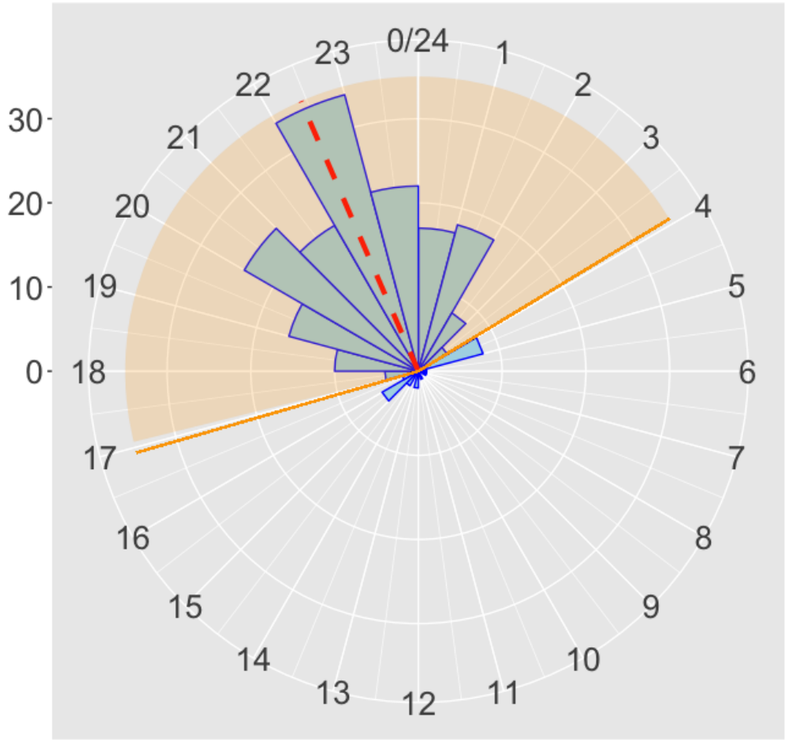
- Binary feature if a new transaction time is within the confidence interval (CI) with probability α (e.g. 0.90, 0.95)
- Timestamp is within 90% CI if its density is larger than the cutoff value:

```
> alpha <- 0.90

> quantile <- qvonmises((1 - alpha)/2,
                        mu = p_mean, kappa = concentration) %% 24

> cutoff <- dvonmises(quantile,
                      mu = p_mean, kappa = concentration)
```

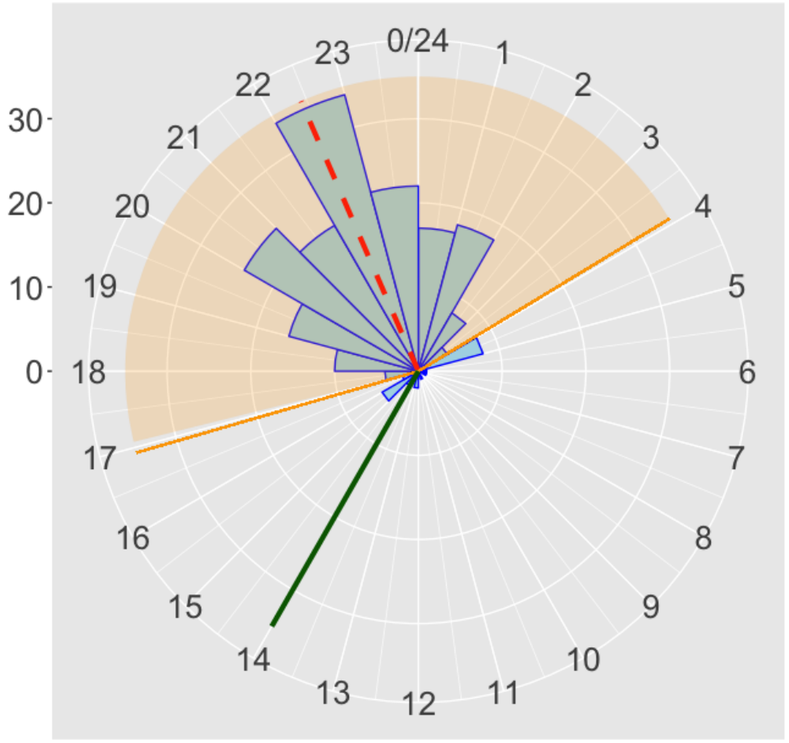
- Binary time feature: `TRUE` if timestamp lies inside CI, `FALSE` otherwise

```
> time_feature <- densities >= cutoff
```

Confidence interval



Confidence interval



Example

Time	Arithmetic mean	Periodic mean	Confidence interval	Periodic feature
18:25	-	-	-	-
20:27	-	-	-	-
20:53	19:26	19:26	17:45 – 21:07	TRUE
00:45	19:55	19:56	18:09 – 21:42	FALSE
19:12	15:08	21:02	17:13 – 00:51	TRUE
23:39	15:56	20:37	17:00 – 00:14	TRUE
06:05	17:14	21:10	17:21 – 00:59	FALSE

Confidence interval with moving time window

```
> print(ts)
[1] 18.42 20.45 20.88  0.75 19.20 23.65  6.08

> time_feature = c(NA, NA)
> for (i in 3:length(ts)) {
    # Previous timestamps
    ts_history <- ts[1:(i-1)]

    # Estimate mu and kappa on historic timestamps
    estimates <- mle.vonmises(ts_history)
    p_mean <- estimates$mu %% 24
    concentration <- estimates$kappa

    # Estimate density of current timestamp
    dens_i <- dvonmises(ts[i], mu = p_mean, kappa = concentration)

    # Check if density is larger than cutoff with confidence level 90%
    alpha <- 0.90
    quantile <- qvonmises((1-alpha)/2, mu=p_mean, kappa=concentration) %% 24
    cutoff <- dvonmises(quantile, mu = p_mean, kappa = concentration)
    time_feature[i] <- dens_i >= cutoff
}

> print(time_feature)
[1]    NA    NA  TRUE FALSE  TRUE  TRUE FALSE
```

FRAUD DETECTION IN R

Let's practice!

FRAUD DETECTION IN R

Frequency features

Tim Verdonck
Professor Data Science at KU Leuven

Need for additional features

Transfers made by Alice & Bob:

```
> trans %>% select(fraud_flag, orig_account_id,
                   benef_country, authentication_cd, channel_cd, amount)

   fraud_flag account_name benef_country authentication_cd channel_cd amount
1           0          Bob         ISO03               AU02       CH07    549
2           0        Alice         ISO03               AU03       CH04     37
3           0          Bob         ISO03               AU04       CH07     25
4           0          Bob         ISO03               AU02       CH06     25
5           0        Alice         ISO03               AU01       CH07     13
6           0          Bob         ISO03               AU02       CH06    785
7           0        Alice         ISO03               AU03       CH04     49
8           0          Bob         ISO03               AU02       CH07     35
...       ...          ...          ...                ...        ...    ...
36          0        Alice         ISO03               AU05       CH04    126
37          0          Bob         ISO03               AU02       CH06     22
38          0        Alice         ISO03               AU03       CH04     41
39          1          Bob         ISO03               AU03       CH05   3779
40          1        Alice         ISO03               AU04       CH05   1531
```

Alice's & Bob's profile

Authentication methods used by Alice:

```
                  fraud_flag
authentication_cd 0 1
             AU01 6 0
             AU02 0 0
             AU03 7 0
             AU04 0 1
             AU05 9 0
```

Alice's & Bob's profile

Authentication methods used by Alice:

```
                  fraud_flag
authentication_cd 0 1
             AU01 6 0
             AU02 0 0
             AU03 7 0
             AU04 0 1
             AU05 9 0
```

Authentication methods used by Bob:

```
                  fraud_flag
authentication_cd 0 1
             AU01 1 0
             AU02 8 0
             AU03 0 1
             AU04 7 0
             AU05 0 0
```

Frequency feature for one account

Arrange the data according to time

```
> library(dplyr)
> trans <- trans %>% arrange(timestamp)
```

Frequency feature for one account

Arrange the data according to time

```
> library(dplyr)
> trans <- trans %>% arrange(timestamp)
```

Alice's data:

```
> trans_Alice <- trans %>% filter(account_name == "Alice")
```

Frequency feature for one account

Arrange the data according to time

```
> library(dplyr)
> trans <- trans %>% arrange(timestamp)
```

Alice's data:

```
> trans_Alice <- trans %>% filter(account_name == "Alice")
```

Alice her first transaction:

```
steps authentication_cd freq_auth
                  AU03         0
```

Frequency feature for one account (step 1)

Step 1: create function `frequency_fun`

Function counts the number of **previous** transfers with the same authentication method as the current one:

```
> frequency_fun <- function(steps, auth_method) {
      n <- length(steps)
      frequency <- sum(auth_method[1:n] == auth_method[n + 1])
      return(frequency)
  }
```

```
steps authentication_cd freq_auth
                  AU03         0
    1             AU03         1
```

Frequency feature for one account (step 1)

Step 1: create function `frequency_fun`

Function counts the number of **previous** transfers with the same authentication method as the current one:

```
> frequency_fun <- function(steps, auth_method) {
      n <- length(steps)
      frequency <- sum(auth_method[1:n] == auth_method[n + 1])
      return(frequency)
  }
```

```
steps authentication_cd freq_auth
                  AU03         0
    1             AU03         1
    2             AU03         2
```

Frequency feature for one account (step 1)

Step 1: create function `frequency_fun`

Function counts the number of **previous** transfers with the same authentication method as the current one:

```
> frequency_fun <- function(steps, auth_method) {
      n <- length(steps)
      frequency <- sum(auth_method[1:n] == auth_method[n + 1])
      return(frequency)
  }
```

```
steps authentication_cd freq_auth
                   AU03         0
    1              AU03         1
    2              AU03         2
    3              AU01         0
```

Frequency feature for one account (step 1)

Step 1: create function `frequency_fun`

Function counts the number of **previous** transfers with the same authentication method as the current one:

```
> frequency_fun <- function(steps, auth_method) {
      n <- length(steps)
      frequency <- sum(auth_method[1:n] == auth_method[n + 1])
      return(frequency)
  }
```

```
steps authentication_cd freq_auth
                   AU03         0
    1              AU03         1
    2              AU03         2
    3              AU01         0
    4              AU01         1
```

Frequency feature for one account (step 2)

Step 2: use `rollapply` from the package `zoo`

```
> library(zoo)
> freq_auth <- rollapply(trans_Alice$transfer_id,
                         width = list(-1:-length(trans_Alice$transfer_id)),
                         partial = TRUE,
                         FUN = frequency_fun,
                         trans_Alice$authentication_cd)
```

Frequency feature for one account (step 2 & 3)

Step 2: use `rollapply` from the package `zoo`

```
> library(zoo)
> freq_auth <- rollapply(trans_Alice$transfer_id,
                         width = list(-1:-length(trans_Alice$transfer_id)),
                         partial = TRUE,
                         FUN = frequency_fun,
                         trans_Alice$authentication_cd)
```

Step 3: frequency feature starts with a zero

```
> freq_auth <- c(0, freq_auth)
```

Result!

	authentication_cd	freq_auth	fraud_flag
1	AU03	0	0
2	AU03	1	0
3	AU03	2	0
4	AU01	0	0
5	AU01	1	0
6	AU05	0	0
7	AU05	1	0
8	AU05	2	0
9	AU01	2	0
10	AU05	3	0
11	AU05	4	0
12	AU05	5	0
13	AU03	3	0
14	AU05	6	0
15	AU01	3	0
16	AU05	7	0
17	AU03	4	0
18	AU01	4	0
19	AU01	5	0
20	AU03	5	0
21	AU05	8	0
22	AU03	6	0
23	AU04	0	1

For multiple accounts

Step 1: group the data by `account_name`:

```
> trans %>% group_by(account_name)
```

Step 2: use `group_by()` and `mutate()` from `dplyr` package

```
> trans <- trans %>% group_by(account_name) %>%
   mutate(freq_auth = c(0,
                        rollapplyr(transfer_id,
                                   width = list(-1:-length(transfer_id)),
                                   partial = TRUE,
                                   FUN = count_fun, authentication_cd)
                        )
          )
```

Result for multiple accounts

	account_name	authentication_cd	freq_auth	fraud_flag
1	Bob	AU02	0	0
2	Alice	AU03	0	0
3	Bob	AU04	0	0
4	Bob	AU02	1	0
5	Alice	AU01	0	0
6	Bob	AU02	2	0
7	Alice	AU03	1	0
8	Bob	AU02	3	0
9	Alice	AU01	1	0
10	Bob	AU04	1	0
11	Bob	AU02	4	0
12	Alice	AU01	2	0
13	Alice	AU05	0	0
14	Alice	AU05	1	0
15	Alice	AU05	2	0
16	Bob	AU02	5	0
17	Bob	AU04	2	0
18	Bob	AU02	6	0
...	...	...	...	...
37	Bob	AU02	7	0
38	Alice	AU03	5	0
39	Bob	AU03	0	1
40	Alice	AU04	0	1

FRAUD DETECTION IN R

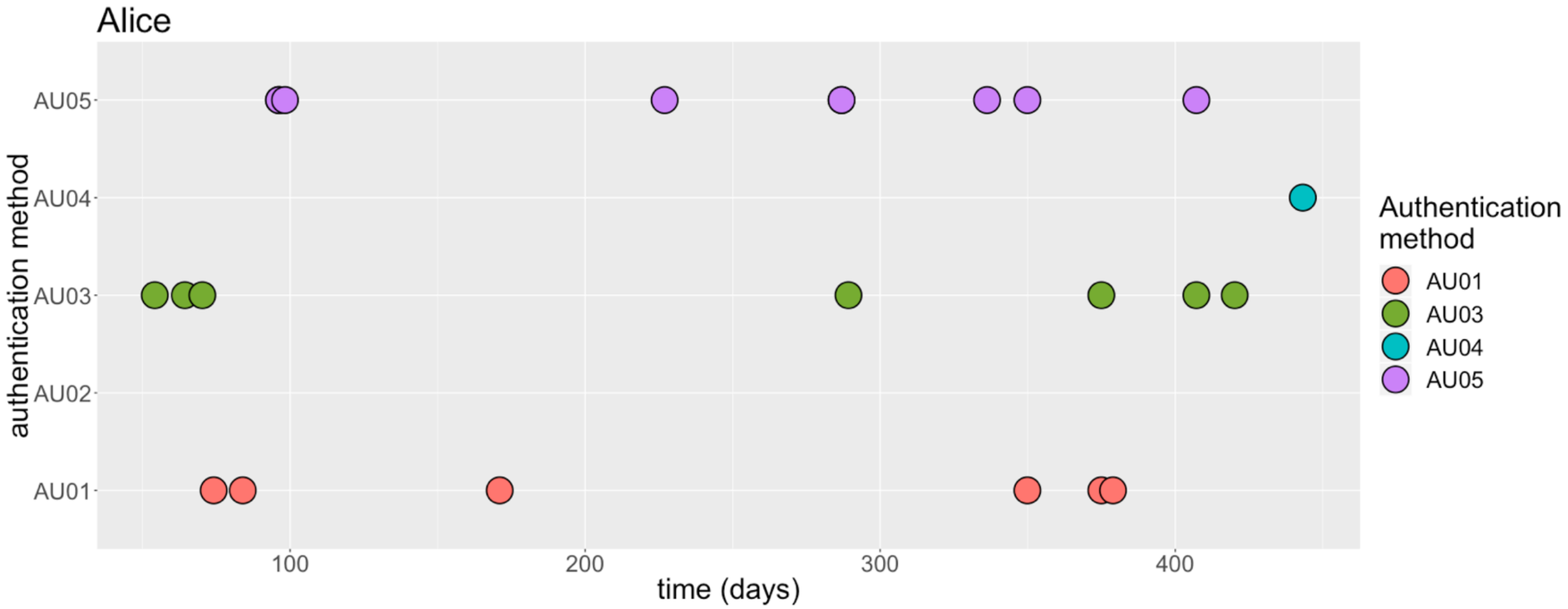
Let's practice!

FRAUD DETECTION IN R

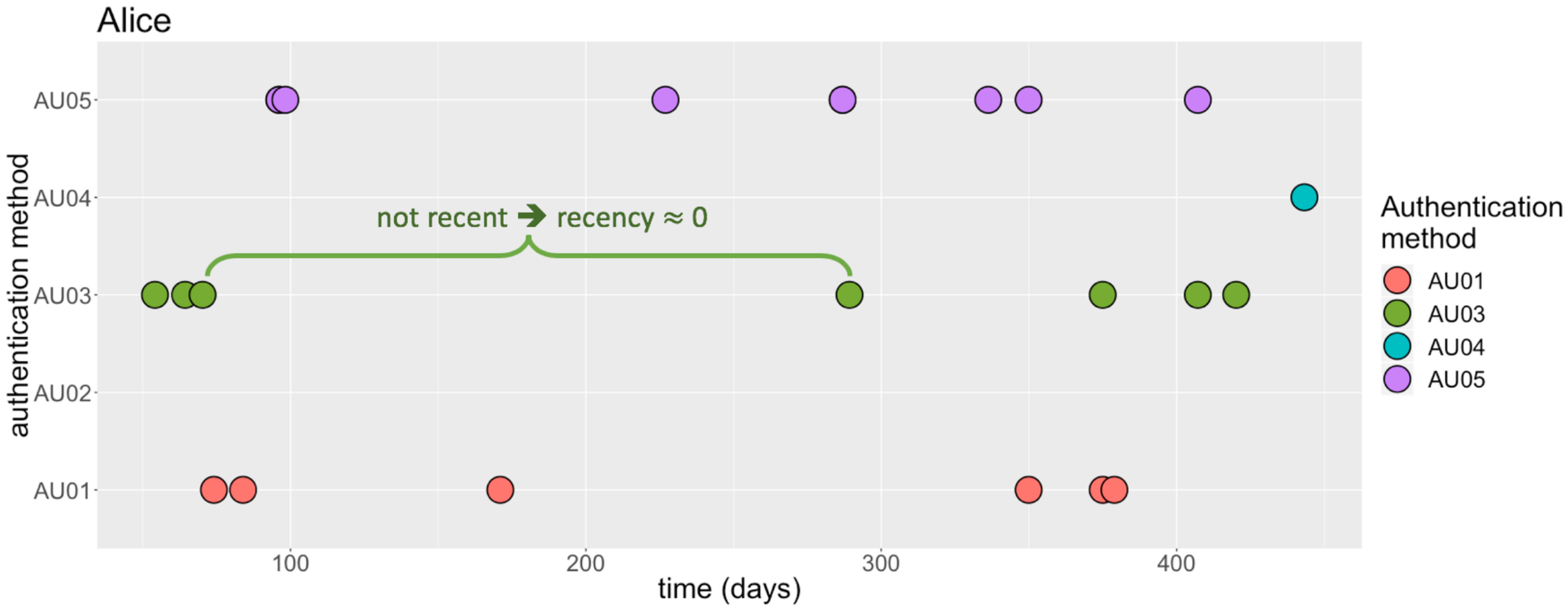
Recency features

Tim Verdonck
Professor Data Science at KU Leuven

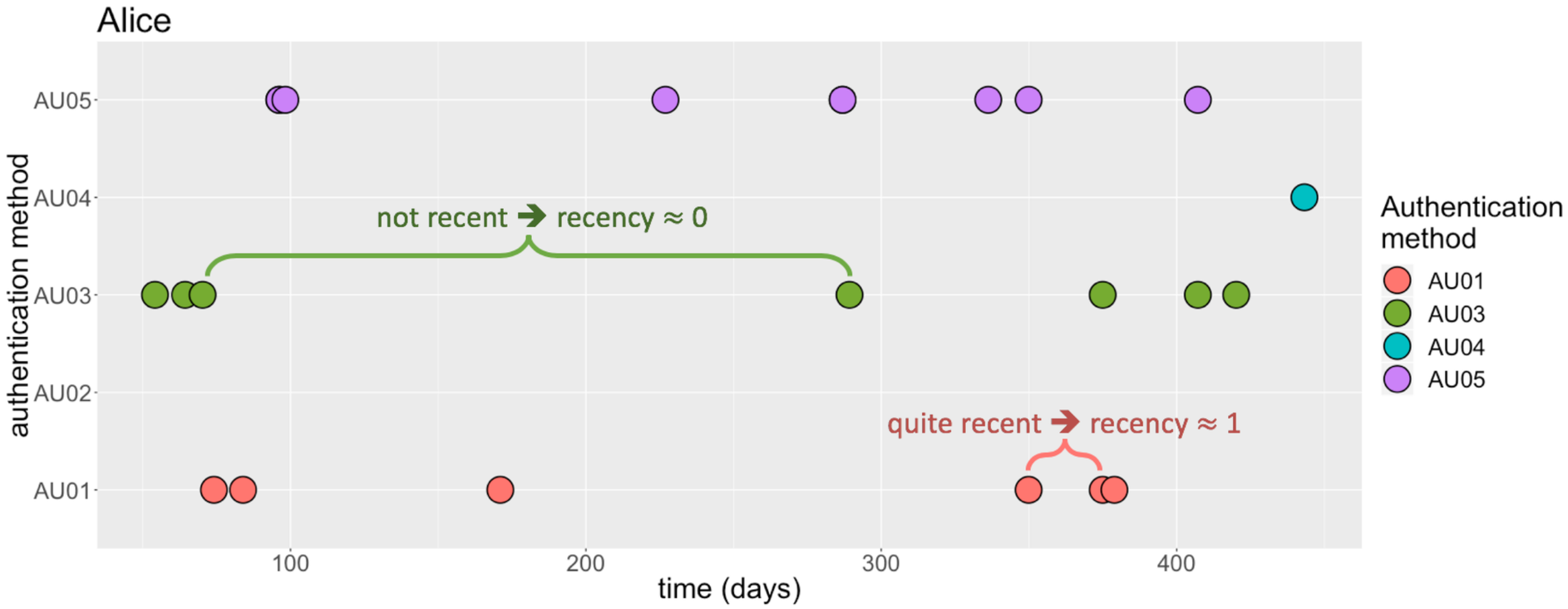
Authentication method vs time



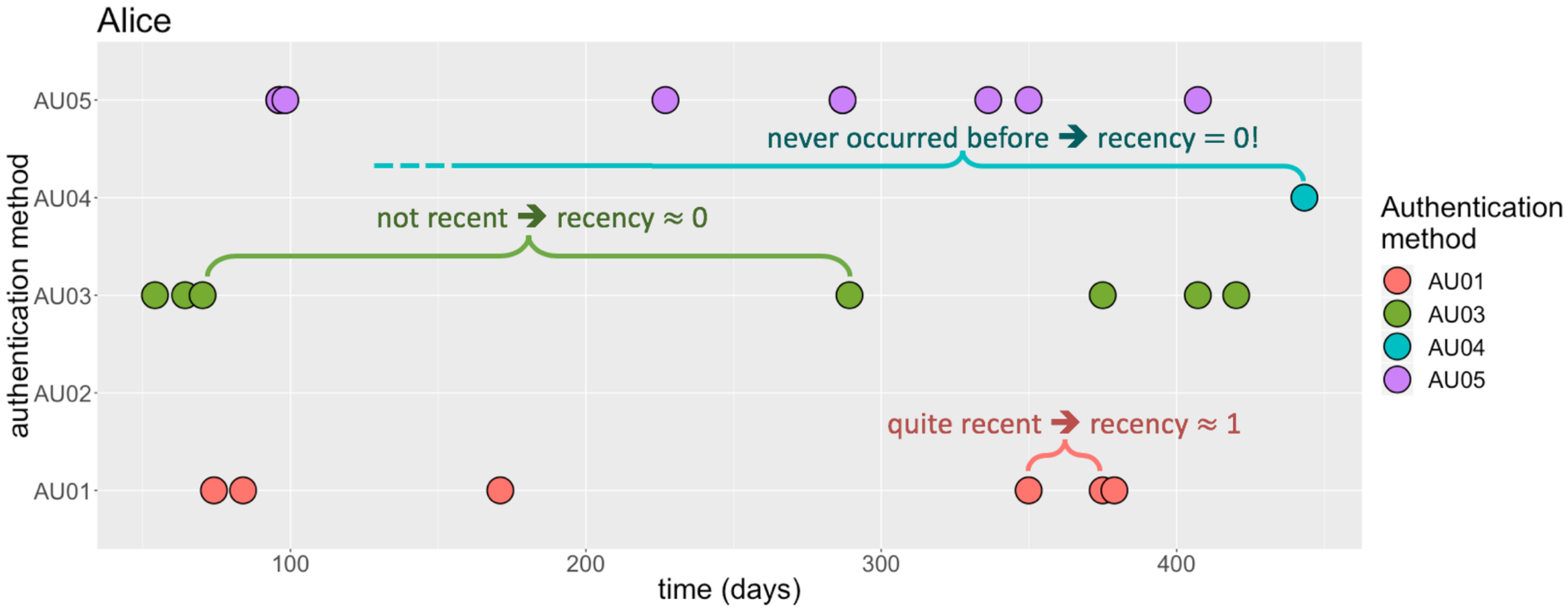
Large time interval



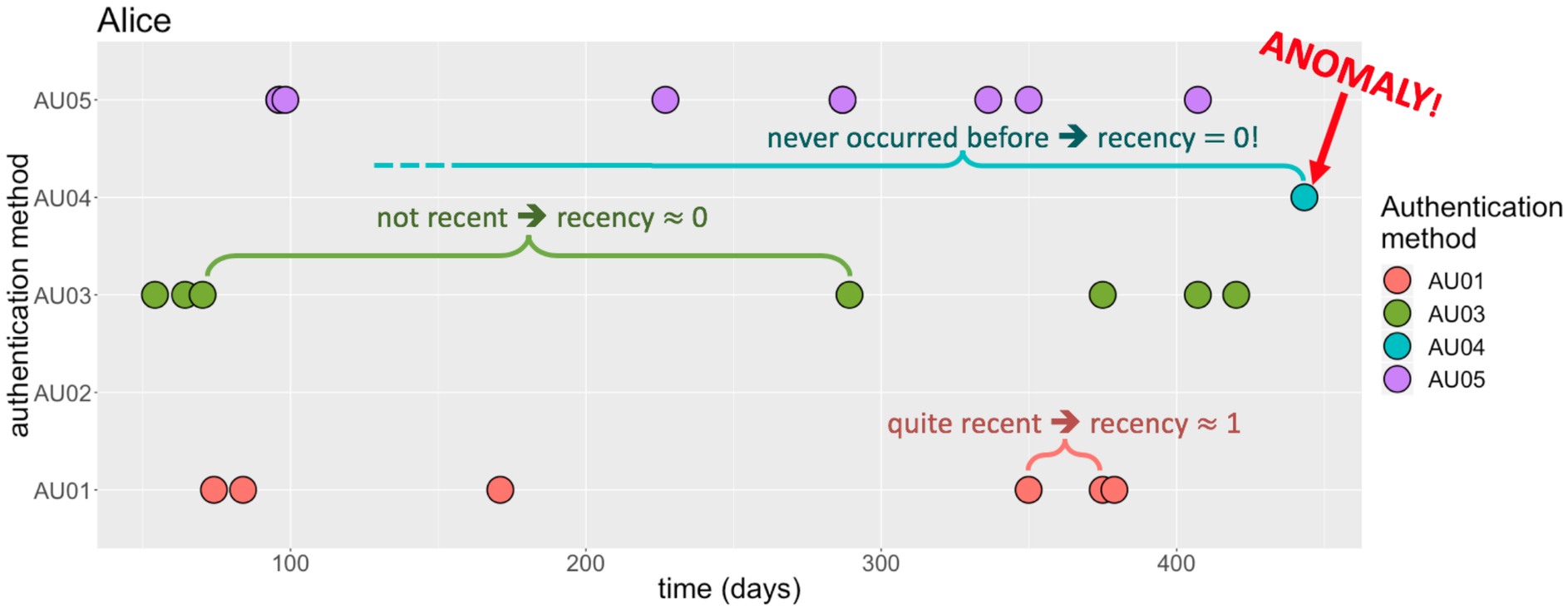
Small time interval



Zero recency



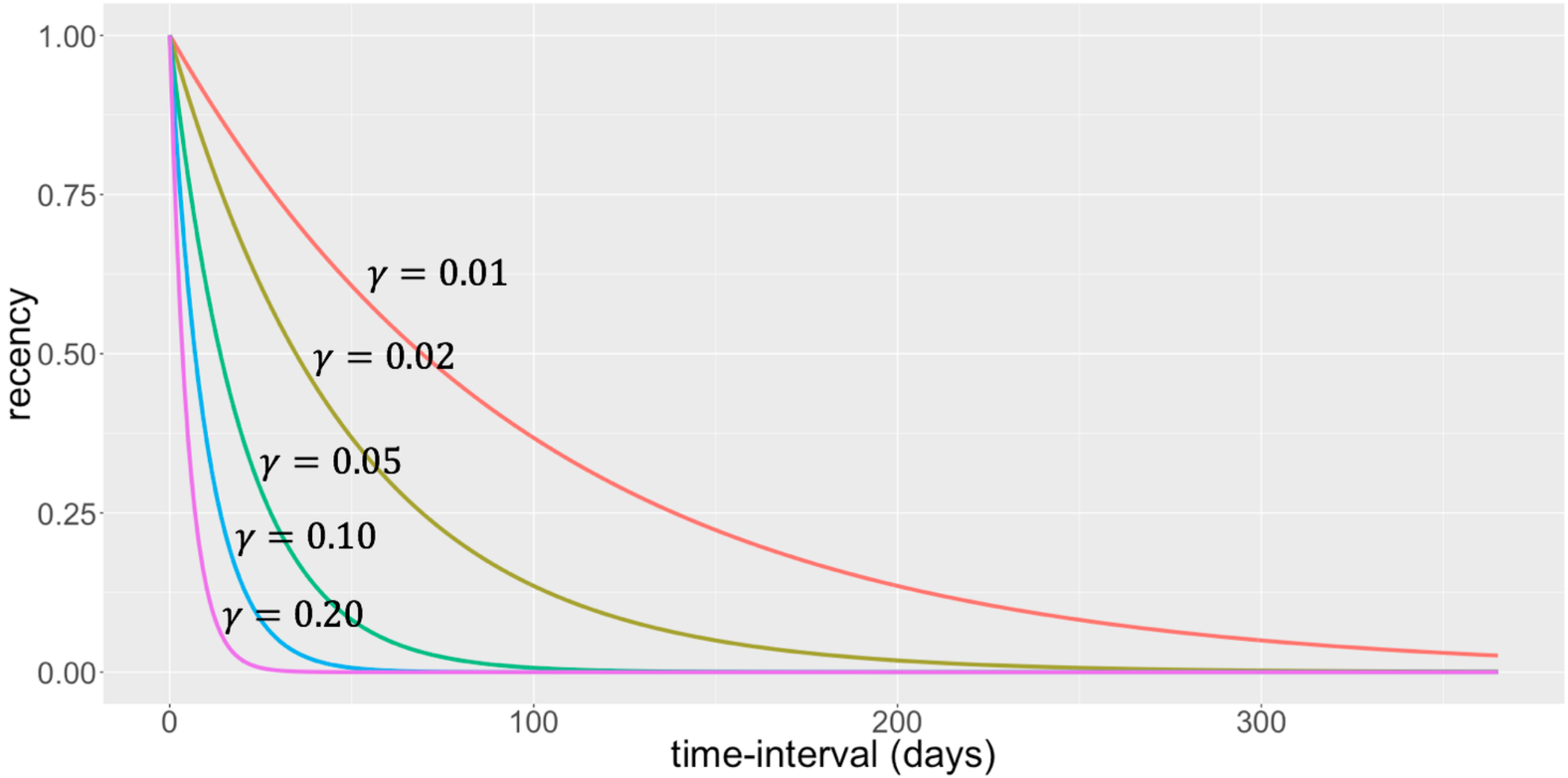
Anomalous behavior



Definition

- $recency = exp(-\gamma \cdot t) = e^{-\gamma t}$
  - t = time-interval between two consecutive events of the same type
  - γ = tuning parameter, typically close to 0 (e.g. 0.01, 0.02, 0.05)
  - $0 \leq recency \leq 1$

Recency vs time



How to choose parameter γ?

- (1) choose when recency is small (e.g. 0.01) after certain amount (t) of time
- (2) calculate $\gamma = -log(recency)/t$
- **Example**: set γ such that recency = 0.01 after t = 180 days

```
> gamma <- -log(0.01)/180
> gamma

[1] 0.02558428
```

Recency feature in R (step 1)

```
recency_fun <- function(t, gamma, auth_cd, freq_auth) {

  n_t <- length(t)

  if (freq_auth[n_t] == 0) {
    recency <- 0 # recency = 0 when frequency = 0

  } else {
    time_diff <- t[1] - max(t[2:n_t][auth_cd[(n_t-1):1] == auth_cd[n_t]])
    # time-interval = current timestamp
    #                 - timestamp of previous transfer with same auth_cd

    recency <- exp(-gamma * time_diff)
  }
  return(recency)

}
```

Recency feature in R (step 2)

(1) Choose value for γ

```
> gamma <- -log(0.01)/180 # = 0.0256
```

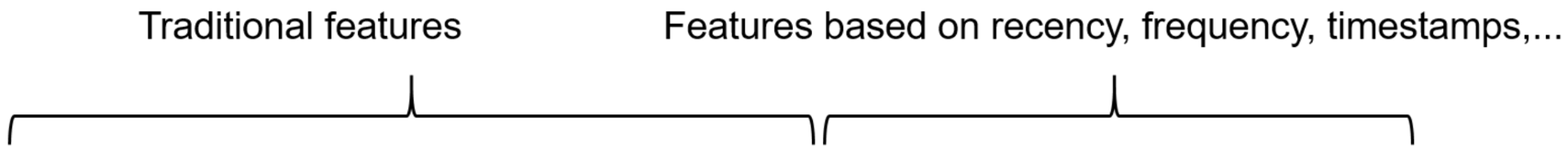
(2) Use `rollapply()`, `group_by()`, and `mutate()`

```
> library(dplyr) # needed for group_by() and mutate()
> library(zoo) # needed for rollapply()

> trans <- trans %>% group_by(account_name) %>%
    mutate(rec_auth = rollapply(timestamp,
                                width = list(0:-length(transfer_id)),
                                partial = TRUE,
                                FUN = recency_fun,
                                gamma, authentication_cd, freq_auth))
```

Result!

	account_name	timestamp	authentication_cd	rec_auth	fraud_flag
1	Bob	44.25	AU02	0.000	0
2	Alice	54.12	AU03	0.000	0
3	Bob	57.45	AU04	0.000	0
4	Bob	64.29	AU02	0.599	0
5	Alice	64.29	AU03	0.771	0
6	Bob	64.29	AU02	1.000	0
7	Alice	70.25	AU03	0.859	0
8	Bob	70.25	AU02	0.859	0
9	Alice	74.08	AU01	0.000	0
10	Bob	74.08	AU04	0.653	0
11	Bob	74.08	AU02	0.907	0
12	Alice	83.93	AU01	0.777	0
13	Alice	96.21	AU05	0.000	0
14	Alice	96.21	AU05	1.000	0
15	Alice	98.25	AU05	0.949	0
16	Bob	109.27	AU02	0.406	0
17	Bob	123.89	AU04	0.280	0
18	Bob	155.95	AU02	0.303	0
...	...	...	...	...	...
37	Bob	407.17	AU02	0.002	0
38	Alice	420.17	AU03	0.717	0
39	Bob	441.34	AU03	0.000	1
40	Alice	443.24	AU04	0.000	1

Features based on time, frequency and recency



	Country Beneficiary	Age Beneficiary	Payment channel	...	Amount	Freq_auth	Rec_auth	...	Time_feat	Fraud
1	Belgium	42	Mobile		102	3	0.02		0	No
2	Greece	35	ATM		125	1	0.59		0	No
3	Germany	27	Web		1067	0	0.86		1	No
...										
n	Belgium	56	Mobile		1039	2	0.12		0	No

FRAUD DETECTION IN R

Let's practice!