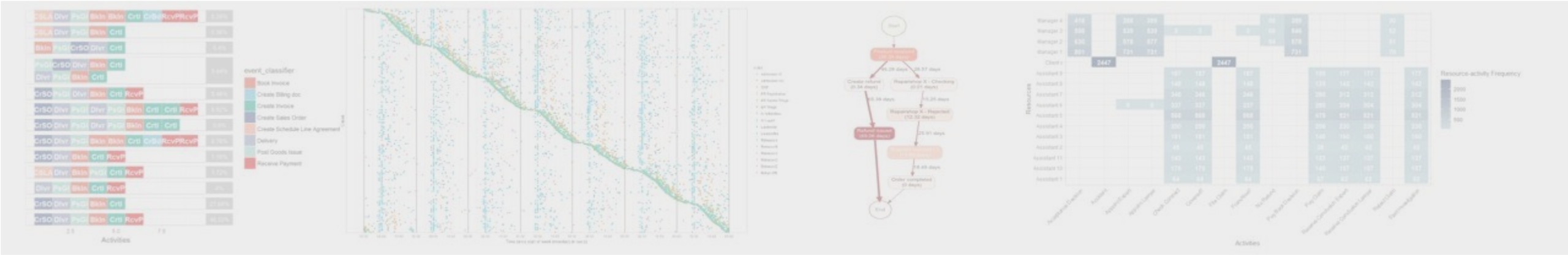


Business Process Analysis in R

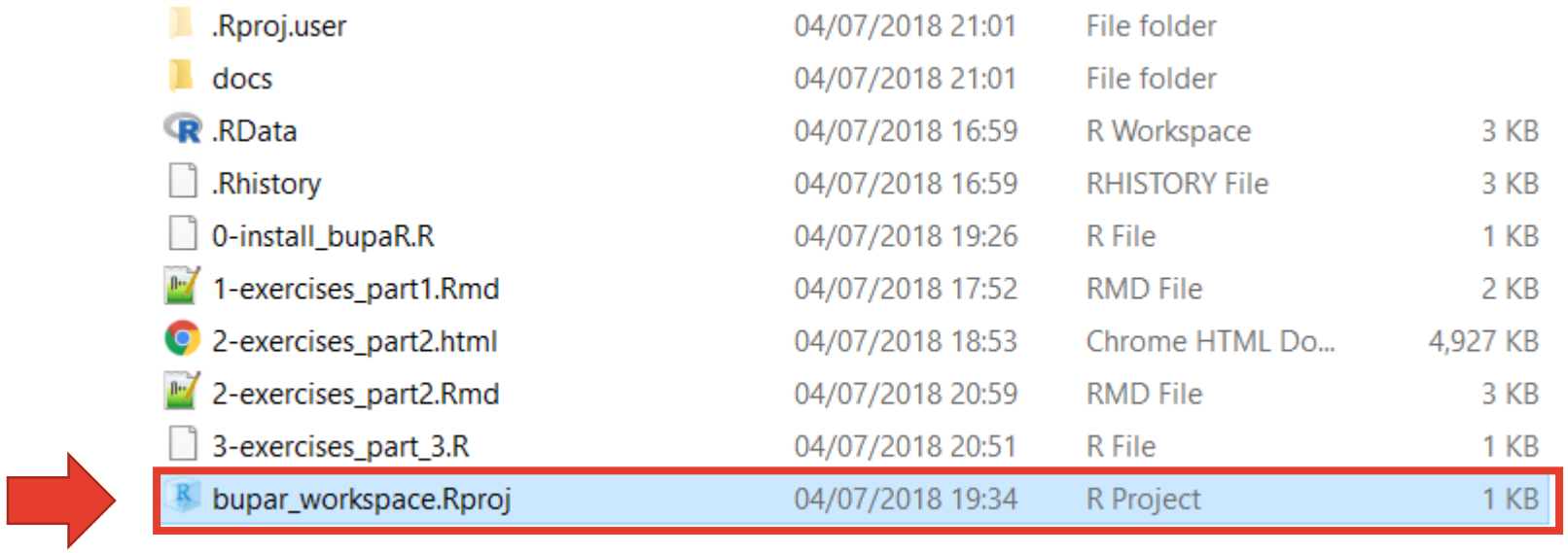
Tutorial

Gert Janssenswillen *@gjanssenswillen @rbupar #bupar*

bupaR

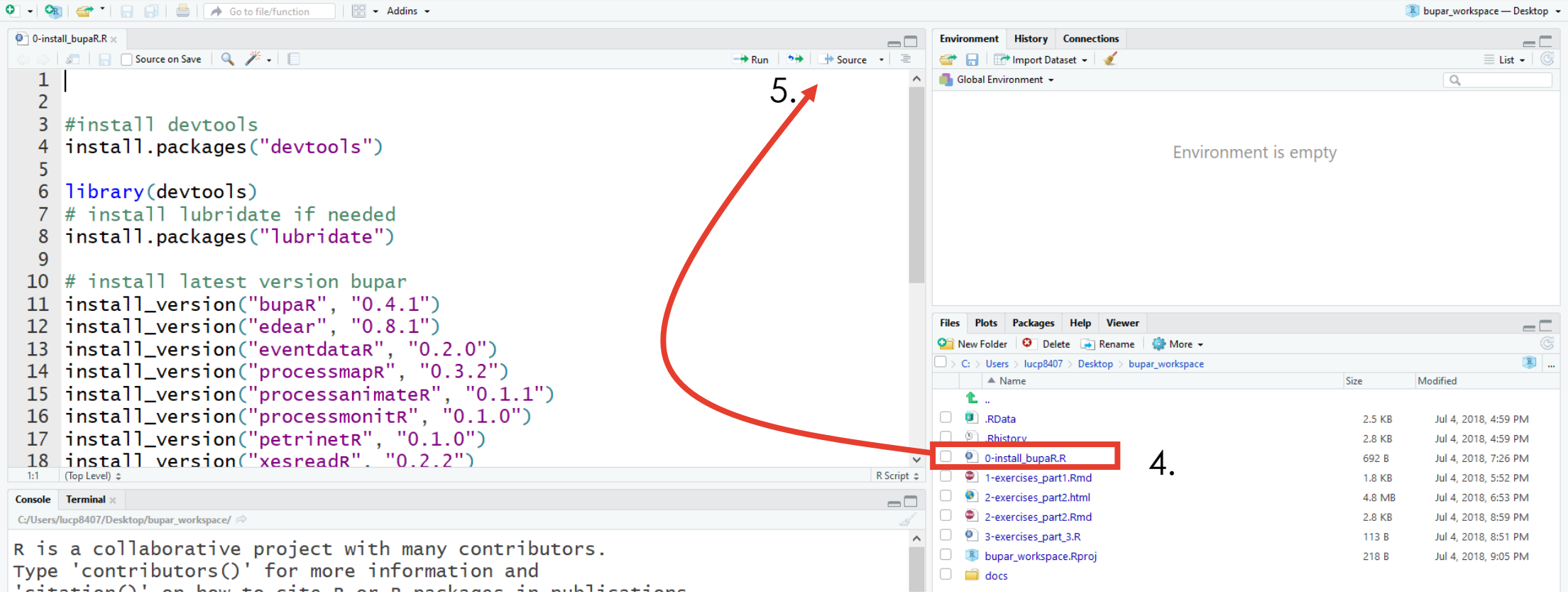
Material

1. Go to **BLACKBOARD** and download `bupar_workspace.zip`
2. Extract the zip-file somewhere on your pc
3. Open the file *bupar_workspace.rproj* with *RStudio*



.Rproj.user	04/07/2018 21:01	File folder	
docs	04/07/2018 21:01	File folder	
.RData	04/07/2018 16:59	R Workspace	3 KB
.Rhistory	04/07/2018 16:59	RHISTORY File	3 KB
0-install_bupaR.R	04/07/2018 19:26	R File	1 KB
1-exercises_part1.Rmd	04/07/2018 17:52	RMD File	2 KB
2-exercises_part2.html	04/07/2018 18:53	Chrome HTML Do...	4,927 KB
2-exercises_part2.Rmd	04/07/2018 20:59	RMD File	3 KB
3-exercises_part_3.R	04/07/2018 20:51	R File	1 KB
bupar_workspace.Rproj	04/07/2018 19:34	R Project	1 KB

Install bupaR



4. Open the file "0-install_bupaR.R" inside the Files pane of Rstudio
5. "Source" the file to run it

- 8 packages released
- 5 packages in development
- +/- 50.000 downloads
- +/- 200 functions

bupaR

#1 A Crash Course in R

Introduction to R

Working with Rstudio

Event data in R

Scripts

Console

Workspace
History

Files
Plots
Help

RStudio

File Edit Code View Project Workspace Plots Tools Help

diamondPricing.R* | formatPlot.R | diamonds

```
library(ggplot2)
source("plots/formatPlot.R")

View(diamonds)
summary(diamonds)

summary(diamonds$price)
aveSize <- round(mean(diamonds$carat), 4)
clarity <- levels(diamonds$clarity)

p <- qplot(carat, price,
           data=diamonds, color=clarity,
           xlab="Carat", ylab="Price",
           main="Diamond Pricing")
```

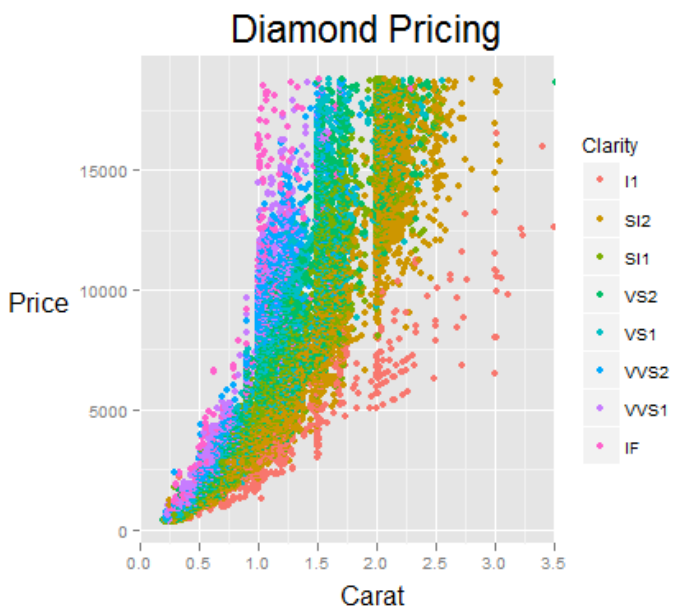
Workspace | History

Data: diamonds — 53940 obs. of 10 variables

Values: aveSize — 0.7979; clarity — character[8]; p — ggplot[8]

Functions: format.plot(plot, size)

Files | Plots | Packages | Help




```
       x                y                z
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
 Median : 5.700   Median : 5.710   Median : 3.530
 Mean   : 5.731   Mean   : 5.735   Mean   : 3.539
 3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040
 Max.   :10.740   Max.   :58.900   Max.   :31.800
> summary(diamonds$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    326     950    2401    3933    5324   18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+            data=diamonds, color=clarity,
+            xlab="Carat", ylab="Price",
+            main="Diamond Pricing")
>
> format.plot(p, size=24)
>
```

Basic R Facts

We use <- for assignment. Or -> for the other way around

```
# Create a character object
> variable <- “value”                or “value” -> variable

# Create a numeric object
> variable2 <- 27

# Create a logical object
> variable3 <- TRUE                  or   variable3 <- T
> variable4 <- FALSE                 or   variable4 <- F
```

Basic R Facts

We use = for function arguments

```
> output <- function(arg1 = input,
                     arg2 = “value1”, …)
```

Basic R Facts

We use function **c** to create a vector

c is for *combining* elements

```
# Create a numeric vector with 2 elements
> interval <- c(2,10)

# Create a character vector with 3 elements
> activities <- c("Check-In","Diagnose","Treatment")
```

Basic R Facts

We start counting from 1

R is a *mathematical* language

```
# Select the 3th element from a vector x
> x[3]
```

Basic R Facts

We are case sensitive

```
> activities <- c("Check-In","Diagnose","Treatment")
> Activities
Error: object 'Activities' not found
> activities == "check-in"
[1] FALSE FALSE FALSE
```

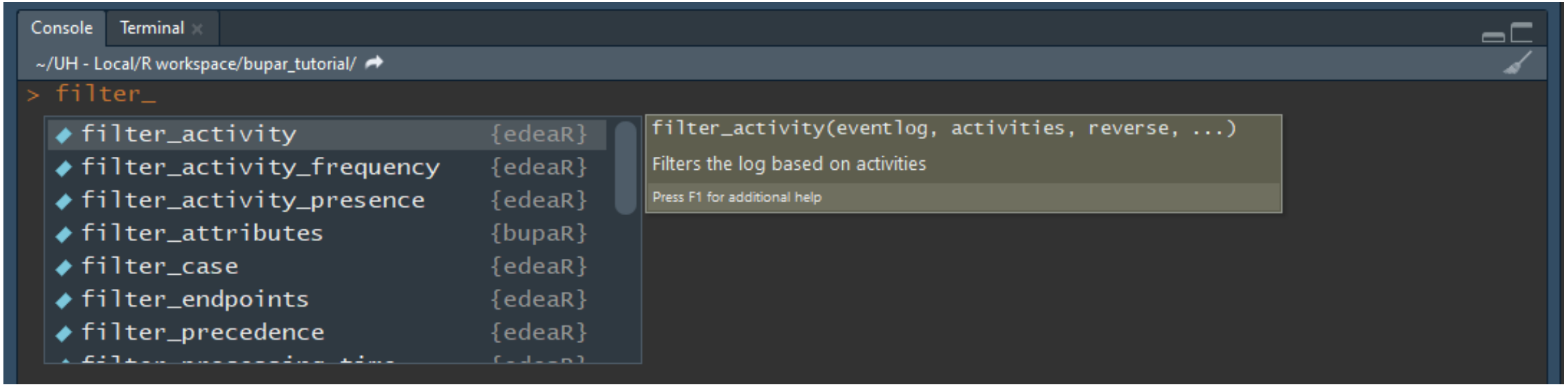
Using R Packages

```
# Install (only once, until new version arrives)
> install.packages("bupaR")

# Load (each session, if needed)
> library("bupaR")          #or      library(bupaR)
```

Getting help - Rstudio features

Auto-complete (Up + Down Arrows + Tab)



Traverse console history (Up + Down Arrows)

Getting help – use ?

Use `?function` to open the helpfile for a function

```
# Open helpfile for n_activities
> ?n_activities
```

Getting help - bupaR

www.bupar.net

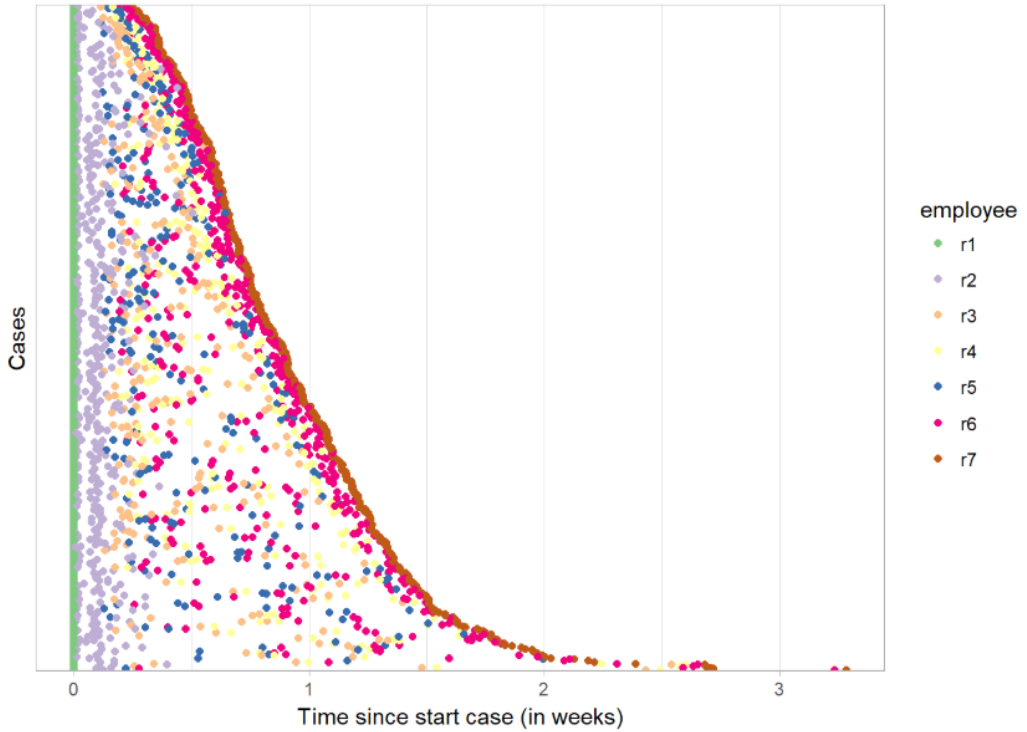
Business Process Analysis in R

Home | About | Get started | I/O | Preprocessing | EDA | Dashboards | Reference

- Static dotted charts
- Interactive dotted charts

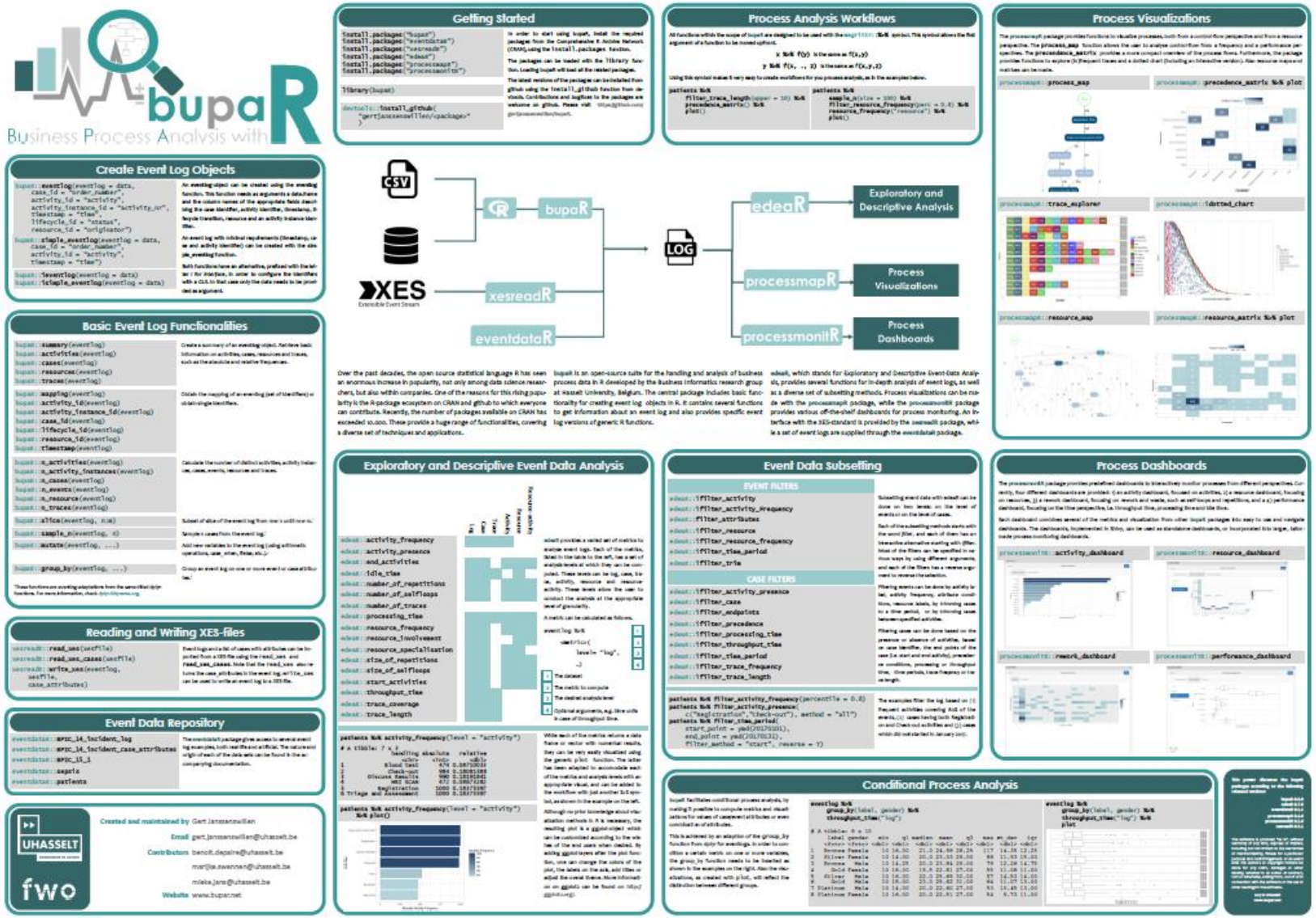
```
patients %>%
    dotted_chart(x = "relative", y = "duration", color = "employee")
```

```
## Warning in deprecated_y_arg(sort, ...): Arguments y is deprecated. Use sort
## instead.
```



Getting help - bupaR

Poster



Event Data in R

```
> library(bupaR)
> patients
Event log consisting of:
5442 events
7 traces
500 cases
7 activities
2721 activity instances

# A tibble: 5,442 x 7
   handling     patient employee handling_id registration_type time                .order
   <fct>        <chr>   <fct>    <chr>       <fct>             <dttm>               <int>
 1 Registration 1       r1       1           start             2017-01-02 11:41:53      1
 2 Registration 2       r1       2           start             2017-01-02 11:41:53      2
 3 Registration 3       r1       3           start             2017-01-04 01:34:05      3
 4 Registration 4       r1       4           start             2017-01-04 01:34:04      4
 5 Registration 5       r1       5           start             2017-01-04 16:07:47      5
 6 Registration 6       r1       6           start             2017-01-04 16:07:47      6
 7 Registration 7       r1       7           start             2017-01-05 04:56:11      7
 8 Registration 8       r1       8           start             2017-01-05 04:56:11      8
 9 Registration 9       r1       9           start             2017-01-06 05:58:54      9
10 Registration 10      r1       10          start             2017-01-06 05:58:54     10
# ... with 5,432 more rows
```

eventlog

An **eventlog** is a special type of **data.frame**

How to test whether object is **eventlog**?

```
> class(patients)
[1] "eventlog"    "tbl_df"      "tbl"
[4] "data.frame"
```

Basic Functions for event data

Counters

- n_activities
- n_activity_instances
- n_cases
- n_events
- n_traces
- n_resources

Labels

- activity_labels
- case_labels
- resource_labels

Details

- activities
- cases
- traces
- resources

Counters

```
> n_activities(patients)
[1] 7
> n_activity_instances(patients)
[1] 2721
> n_cases(patients)
[1] 500
> n_events(patients)
[1] 5442
> n_resources(patients)
[1] 7
> n_traces(patients)
[1] 7
```

Labels

```
> activity_labels(patients)
[1] Registration          Triage and Assessment Blood test
[4] MRI SCAN              X-Ray                 Discuss Results
[7] Check-out
> resource_labels(patients)
[1] r1 r2 r3 r4 r5 r6 r7
```

Details

```
> activities(patients)
# A tibble: 7 x 3
  handling              absolute_frequency relative_frequency
  <fct>                              <int>              <dbl>
1 Registration                         500             0.184
2 Triage and Assessment                500             0.184
3 Discuss Results                      495             0.182
4 Check-out                            492             0.181
5 X-Ray                                261             0.0959
6 Blood test                           237             0.0871
7 MRI SCAN                             236             0.0867
```

Details

```
> resources(patients)
# A tibble: 7 x 3
  employee absolute_frequency relative_frequency
  <fct>                 <int>              <dbl>
1 r1                      500             0.184
2 r2                      500             0.184
3 r6                      495             0.182
4 r7                      492             0.181
5 r5                      261             0.0959
6 r3                      237             0.0871
7 r4                      236             0.0867
```

Details

```
> cases(patients)
# A tibble: 500 x 10
   patient trace_length number_of_activities start_timestamp     complete_timestamp
   <chr>           <int>                 <int> <dttm>              <dttm>
 1 1                   6                     6 2017-01-02 11:41:53 2017-01-09 19:45:45
 2 10                  5                     5 2017-01-06 05:58:54 2017-01-10 15:41:59
 3 100                 5                     5 2017-04-11 16:34:31 2017-04-22 09:58:07
 4 101                 5                     5 2017-04-16 06:38:58 2017-04-23 02:55:23
 5 102                 5                     5 2017-04-16 06:38:58 2017-04-22 10:50:04
 6 103                 6                     6 2017-04-19 20:22:01 2017-04-23 02:36:55
 7 104                 6                     6 2017-04-19 20:22:01 2017-04-23 02:07:20
 8 105                 6                     6 2017-04-21 02:19:09 2017-04-27 01:09:05
 9 106                 6                     6 2017-04-21 02:19:09 2017-05-01 09:54:39
10 107                 5                     5 2017-04-22 18:32:16 2017-04-27 02:45:57
# ... with 490 more rows, and 5 more variables: trace <chr>, trace_id <dbl>,
#   duration_in_days <dbl>, first_activity <fct>, last_activity <fct>
```

Details

```
> traces(patients)
# A tibble: 7 x 3
  trace                                                absolute_frequen~ relative_frequen~
  <chr>                                                            <int>             <dbl>
1 Registration,Triage and Assessment,Blood test,~                    234           0.468
2 Registration,Triage and Assessment,X-Ray,Discu~                    258           0.516
3 Registration,Triage and Assessment,Blood test,~                      2           0.00400
4 Registration,Triage and Assessment,X-Ray,Discu~                      1           0.00200
5 Registration,Triage and Assessment,Blood test                        1           0.00200
6 Registration,Triage and Assessment,X-Ray                             2           0.00400
7 Registration,Triage and Assessment                                   2           0.00400
```

Summary

```
> summary(patients)
Number of events:  5442
Number of cases:  500
Number of traces:  7
Number of distinct activities:  7
Average trace length:  10.884

Start eventlog:  2017-01-02 11:41:53
End eventlog:  2018-05-05 07:16:02

                   handling      patient          employee  handling_id        registration_type      time                          .order
 Blood test            : 474   Length:5442        r1:1000   Length:5442        complete:2721     Min.   :2017-01-02 11:41:53   Min.   :   1
 Check-out             : 984   Class :character   r2:1000   Class :character   start   :2721     1st Qu.:2017-05-06 17:15:18   1st Qu.:1361
 Discuss Results       : 990   Mode  :character   r3: 474   Mode  :character                     Median :2017-09-08 04:16:50   Median :2722
 MRI SCAN              : 472                      r4: 472                                        Mean   :2017-09-02 20:52:34   Mean   :2722
 Registration          :1000                      r5: 522                                        3rd Qu.:2017-12-22 15:44:11   3rd Qu.:4082
 Triage and Assessment:1000                       r6: 990                                        Max.   :2018-05-05 07:16:02   Max.   :5442
 X-Ray                 : 522                      r7: 984
```

Summary

```
> summary(cases(patients))

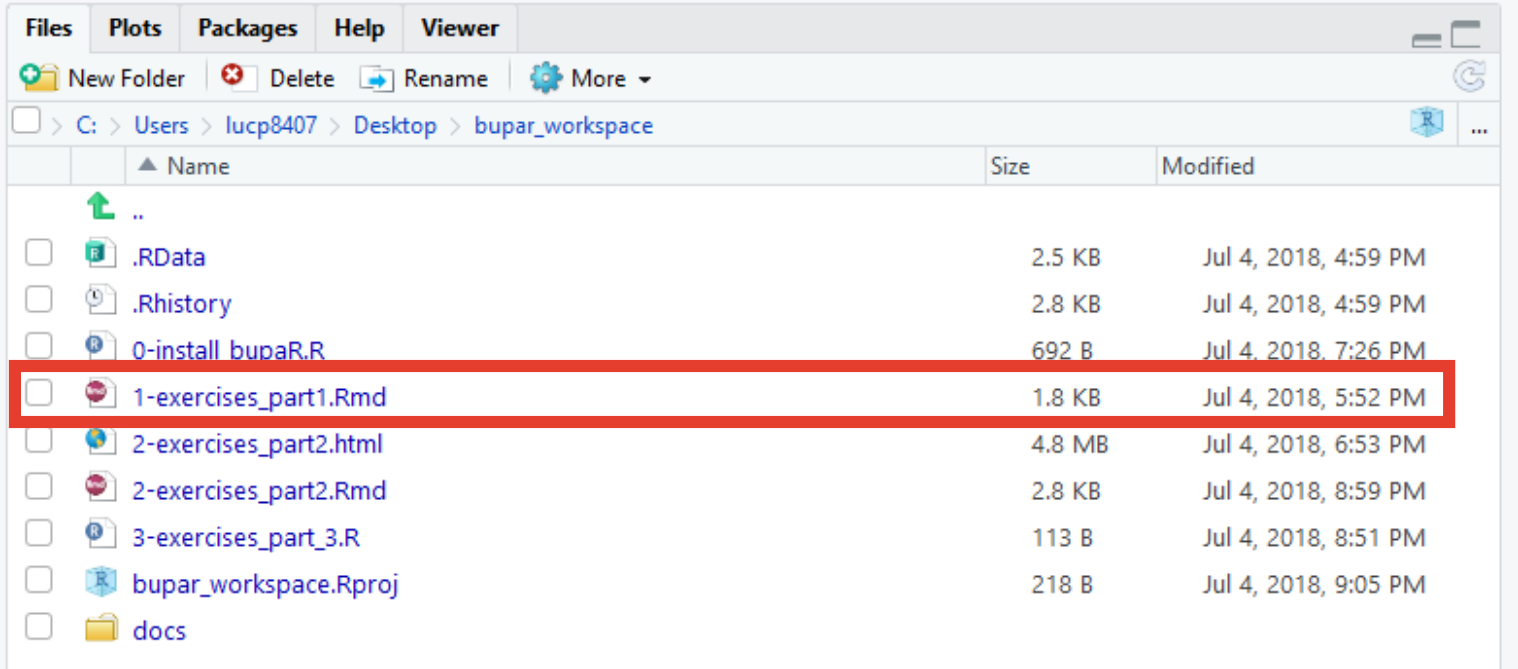
   patient           trace_length   number_of_activities start_timestamp                complete_timestamp
 Length:500         Min.   :2.000   Min.   :2.000        Min.   :2017-01-02 11:41:53   Min.   :2017-01-09 18:37:06
 Class :character   1st Qu.:5.000   1st Qu.:5.000        1st Qu.:2017-05-04 21:29:42   1st Qu.:2017-05-12 21:48:02
 Mode  :character   Median :5.000   Median :5.000        Median :2017-09-06 04:26:19   Median :2017-09-14 00:05:09
                    Mean   :5.442   Mean   :5.442        Mean   :2017-08-31 19:42:13   Mean   :2017-09-07 11:56:06
                    3rd Qu.:6.000   3rd Qu.:6.000        3rd Qu.:2017-12-22 01:11:46   3rd Qu.:2017-12-31 11:28:00
                    Max.   :6.000   Max.   :6.000        Max.   :2018-05-01 22:07:54   Max.   :2018-05-05 07:16:02

   trace              trace_id     duration_in_days      first_activity               last_activity
 Length:500         Min.   :1.000   Min.   : 1.496   Registration:500    Blood test           :  1
 Class :character   1st Qu.:4.000   1st Qu.: 4.314                       Check-out            :492
 Mode  :character   Median :7.000   Median : 6.086                       Discuss Results      :  3
                    Mean   :5.536   Mean   : 6.676                       Triage and Assessment:  2
                    3rd Qu.:7.000   3rd Qu.: 8.587                       X-Ray                :  2
                    Max.   :7.000   Max.   :23.107
```

Exercises Part 1

Exercises part 1

Open *1-exercises_part1.Rmd* in the Files pane of Rstudio



This is an Rmarkdown file, not a regular R script

Rmarkdown?

A document to combine R-code and tekst (markdown)

1-exercises_part1.Rmd

Knit

Knit ~ compile document

Insert

Run

Insert new R chunks

```
output:
  html_document:
---

```{r include = F}
library(bupaR)
library(bupaRtutorial)
```

In this exercise we will be looking at an event log containing recorded eating activities of a group of people. The eventlog has the name `eating_patterns` and can be seen below. Each case is a person and each activity instance is a meal.

```{r}
eating_patterns
```

Try to find the answer to the following questions. The first one has been solved for you as an example.
```

R

Text

Run all chunks above

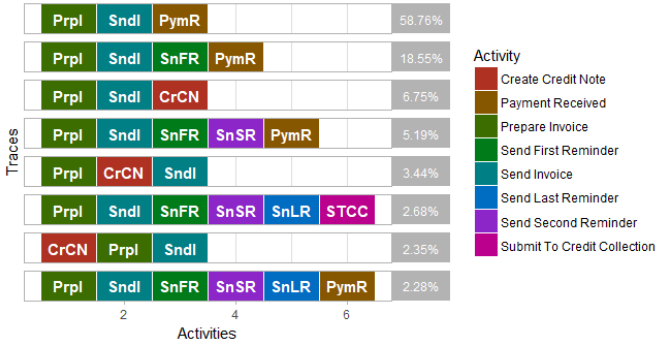
Run this chunk

R "chunks" start with three ``` and end with three ```

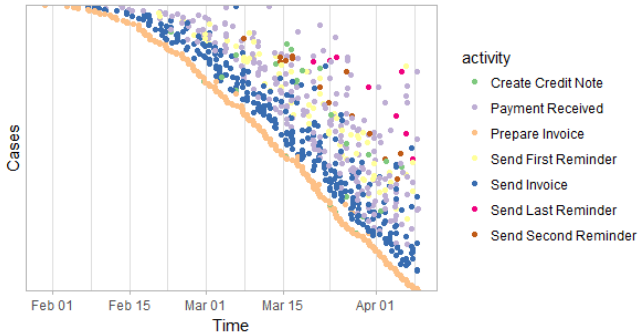
#2 Process Analysis

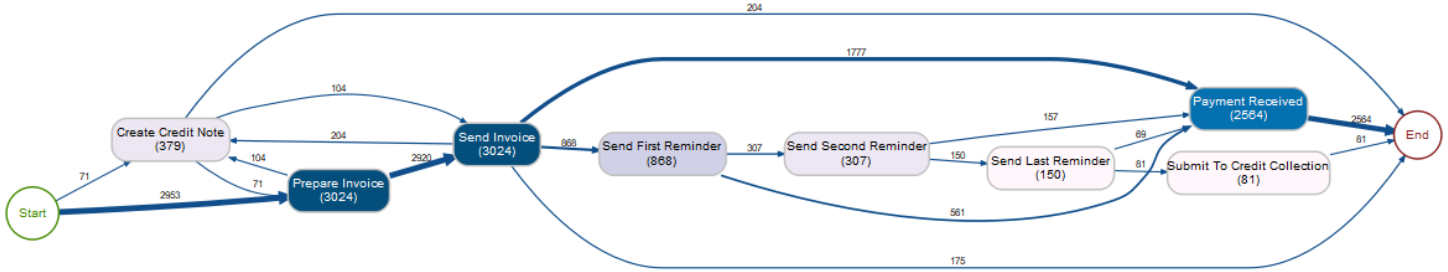
Visualizing Process Data

Visualizing Processes



- Trace explorer
- Dotted chart
- Process map





Trace explorer

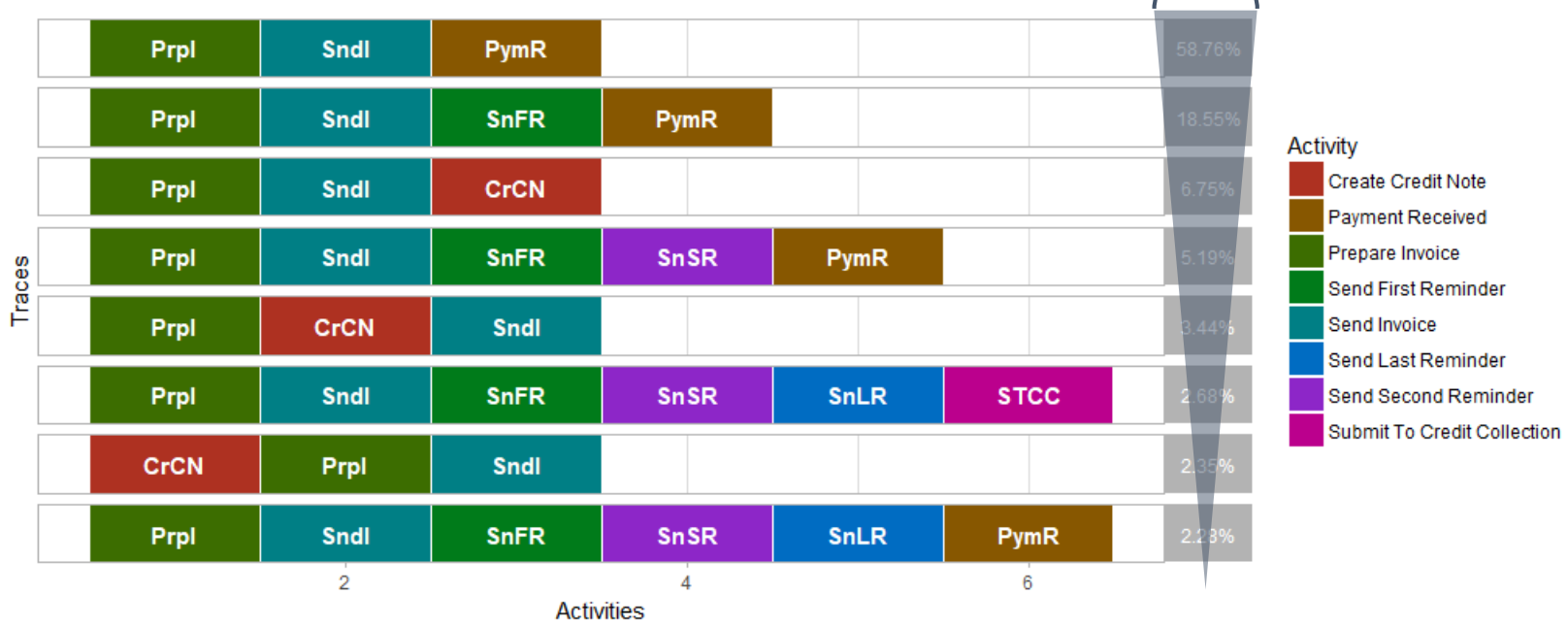
Visualize the **traces**, e.g. variants, in the event log

```
# default call
> trace_explorer(eventlog)
```

Trace explorer

Frequency
Relative number of cases

Traces
(variants)



Trace explorer

Argument	Default	Alternatives
`type`	`“frequent”`	`“infrequent”`
`coverage`	`0.2`	`Value between 0 and 1`

Type defines whether to show the most or least frequents variants first. **Coverage** defines the target percentage of cases to show. The default of 20% anticipates low structuredness in event logs.

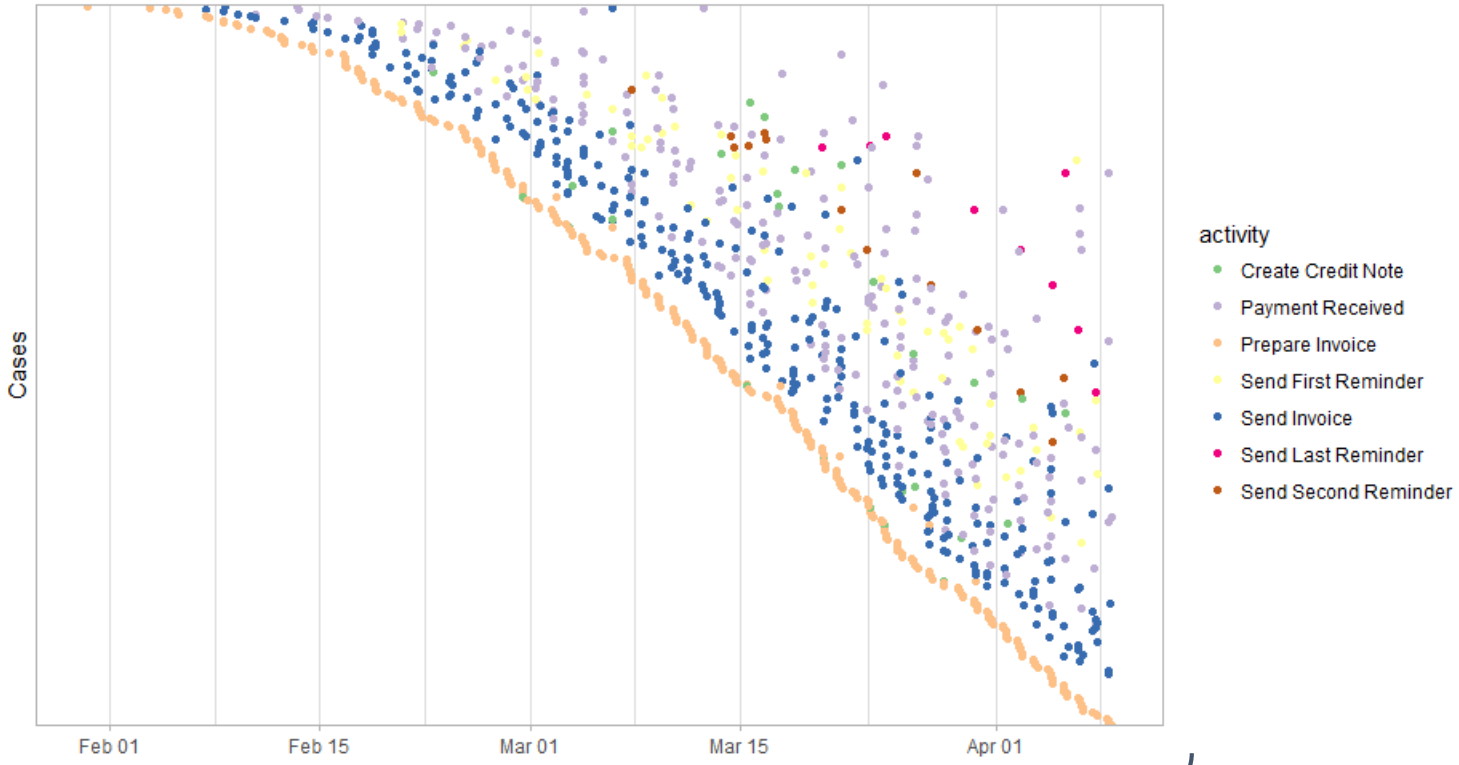
Dotted chart

Graphical overview of different **activities** for different **cases** over **time**.

```
# default call
> dotted_chart(eventlog)
```

Dotted chart

Cases
Sorted by start

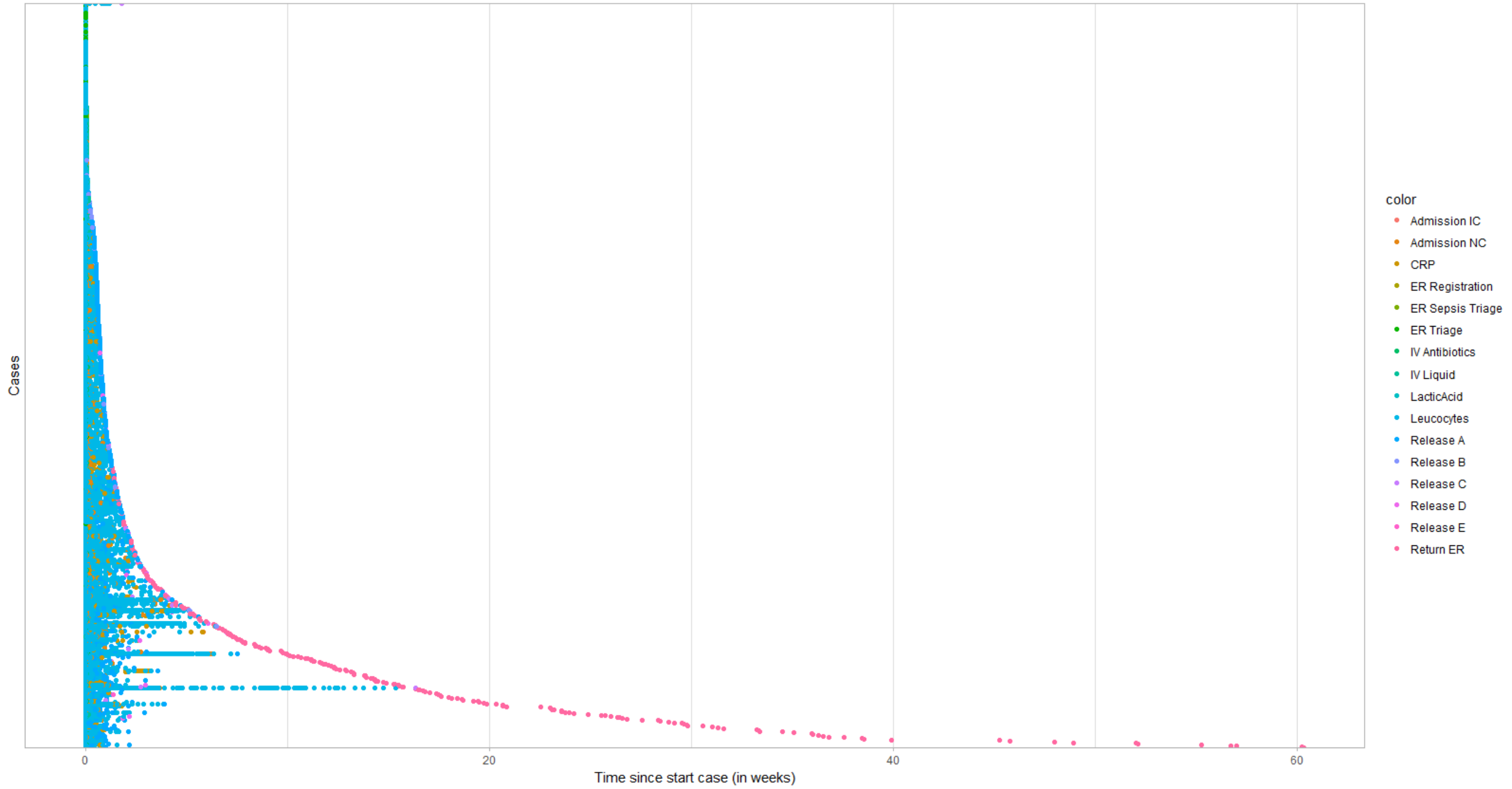


Absolute time

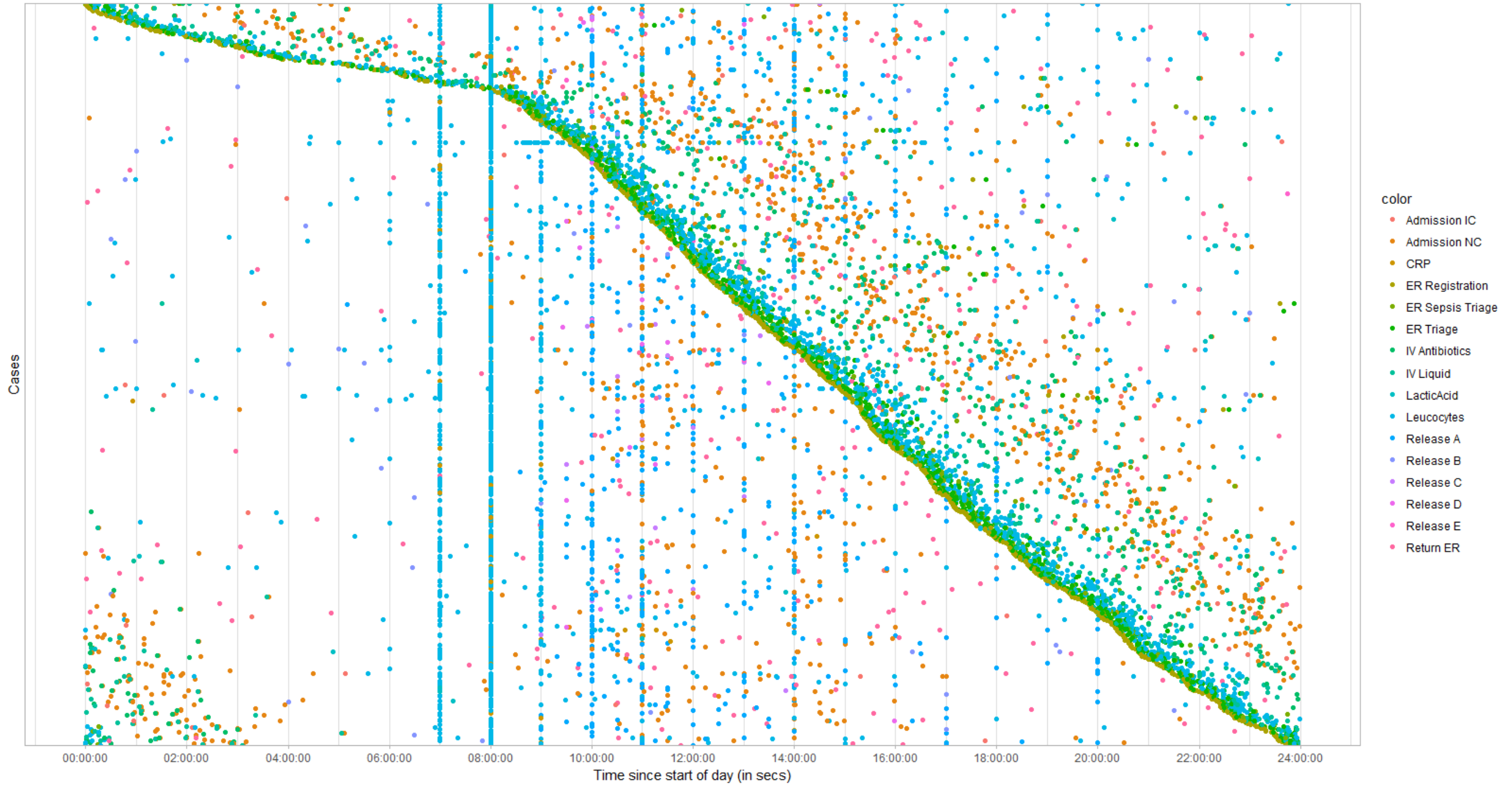
Dotted chart

Argument	Default	Description
x	“absolute”	Time perspective
sort	“start”	Ordering of cases
color	activity_id	Attribute for color
units	“weeks”	Duration units

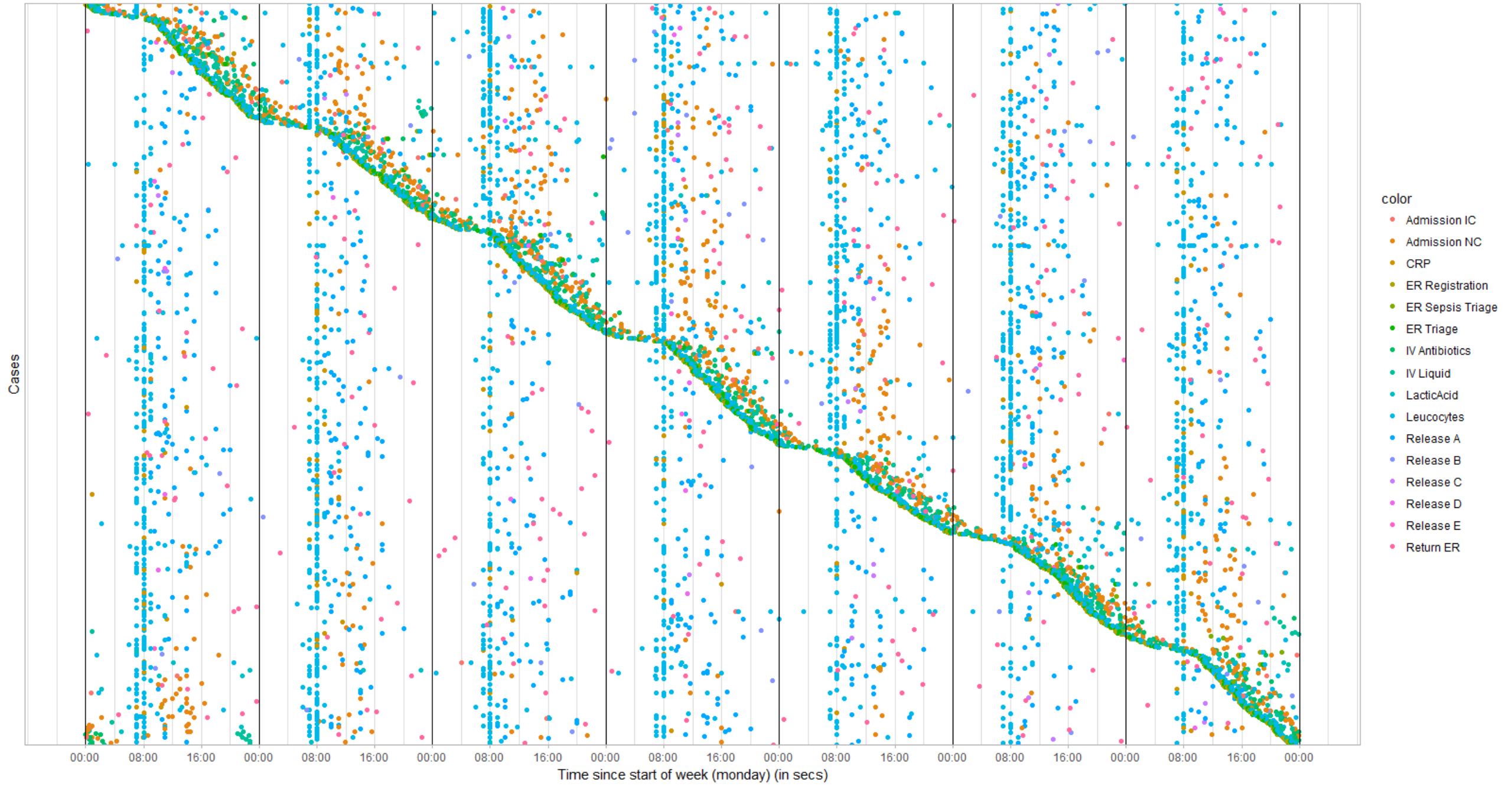
x = “absolute”, sort = “start”



x = “relative”, sort = “duration”



x = “relative_day”, sort = “start_day”



x = “relative_week”, sort = “start_week”

Dotted chart

Variants

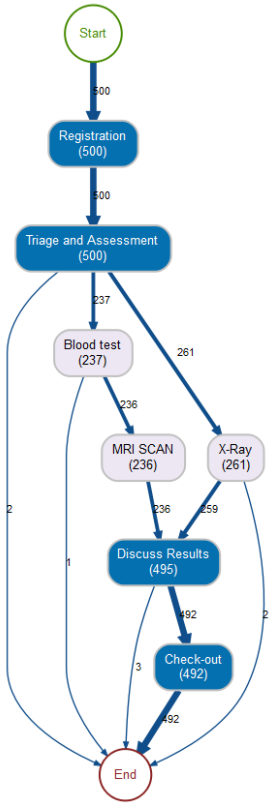
`idotted_chart`	Change settings interactively
`ploty_dotted_chart`	Interact with plot
`iplotly_dotted_chart`	Combination of both

Process maps

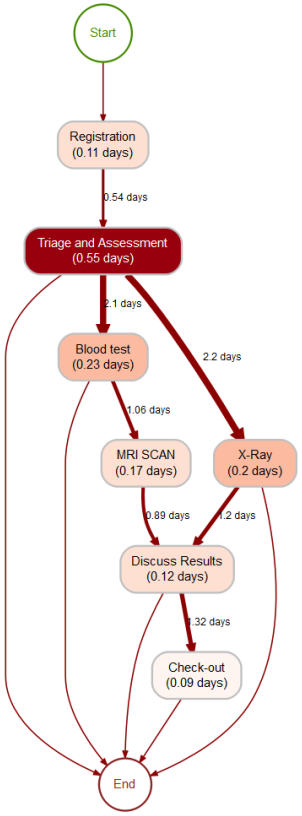
```
# default call
> process_map(eventlog)
```

Process map types

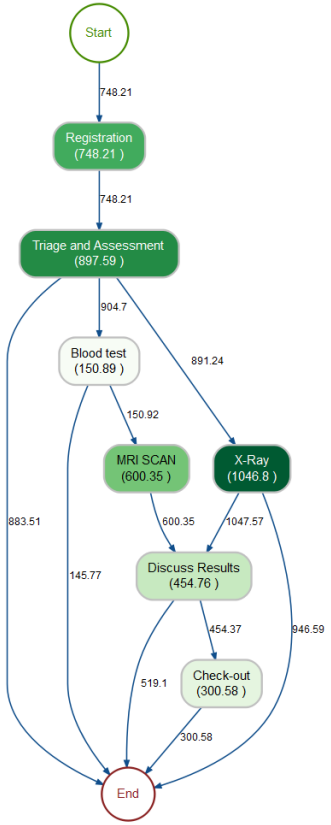
Frequency



Performance



Custom



Process maps

Argument	Default	Other
type	frequency()	performance(),custom()
or		
type_nodes	frequency()	performance(),custom()
type_edges	frequency()	performance(),custom()
rankdir	“lr”	“tb”,”bt”,”rl”

Type can be set for edges and nodes seperately.
Rankdir changes the direction of the map.

Frequency type

Argument	Default	Other

`value`	`“absolute”`	`“relative”`
		`“absolute_case”`
		`“relative_case”`

Primary argument: the frequency value to show on nodes and arcs.

Other arguments: modify color scale

Process maps

```
# A process map with the relative number of cases
> process_map(eventlog, frequency("relative_case"))
```

Performance type

Argument	Default	Other
FUN	mean	Any aggregation function
units	“days”	“hours”,”weeks”,”mins”,…
flow_time	“idle_time”	“inter_start_time”

Any aggregation function (mean, min, max, …), even self-defined, can be used to aggregate performance values

The **flow_time** argument defines the time to put on edges

Process maps

```
# A process map with the relative number of cases
> process_map(eventlog, frequency("relative_case"))
# A process map wit median time in weeks
> process_map(eventlog, performance(median, "weeks"))
```

Custom type

Argument	Default	Other
FUN	mean	Any aggregation function
attribute		Any data attribute
units		Text to indicate units

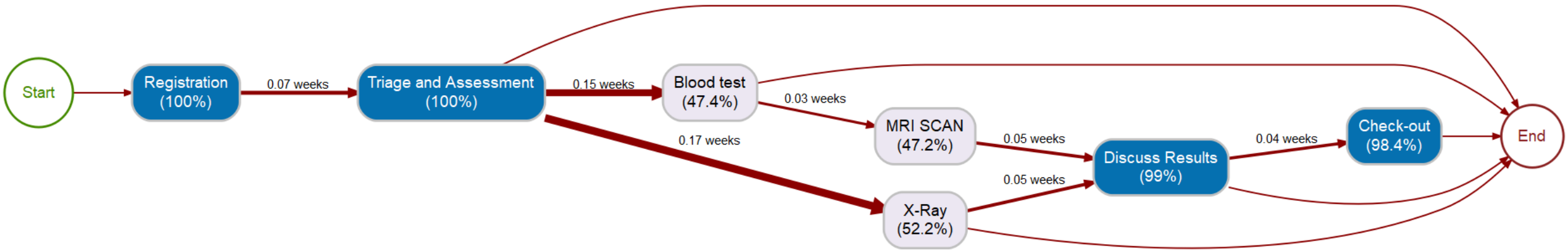
Any aggregation function (mean, min, max, …), even self-defined, can be used to aggregate the values.

Process maps

```
# A process map with the relative number of cases
> process_map(eventlog, frequency("relative_case"))
# A process map wit median time in weeks
> process_map(eventlog, performance(median, "weeks"))
# A process map with average "cost"
> process_map(eventlog, custom(mean, "cost", "USD"))
```

Combining types

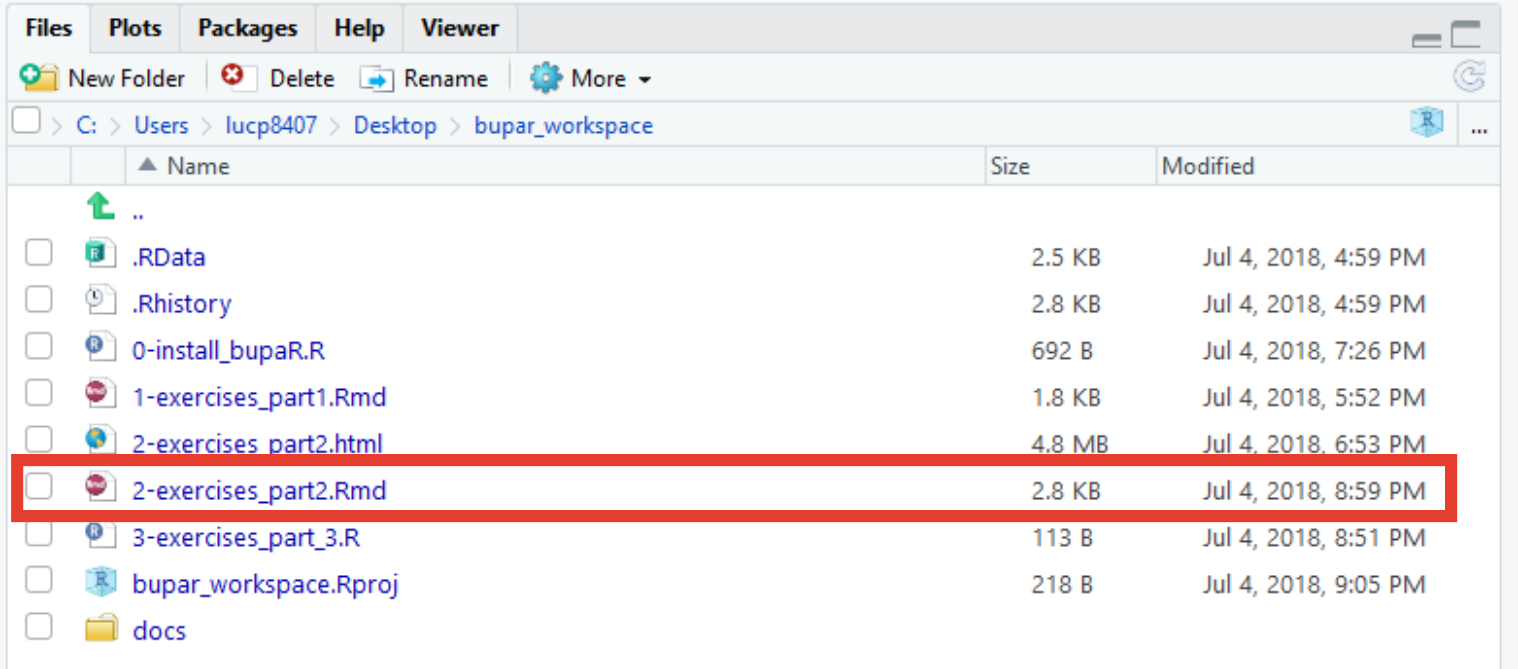
```
> process_map(
        eventlog,
        type_nodes = frequency(“relative_case”),
        type_edges = performance(median, “weeks”)
  )
```


Start
Registration
(100%)
0.07 weeks
Triage and Assessment
(100%)
0.15 weeks
Blood test
(47.4%)
0.03 weeks
MRI SCAN
(47.2%)
0.05 weeks
0.17 weeks
X-Ray
(52.2%)
0.05 weeks
Discuss Results
(99%)
0.04 weeks
Check-out
(98.4%)
End

Exercises Part 2

Exercises part 1

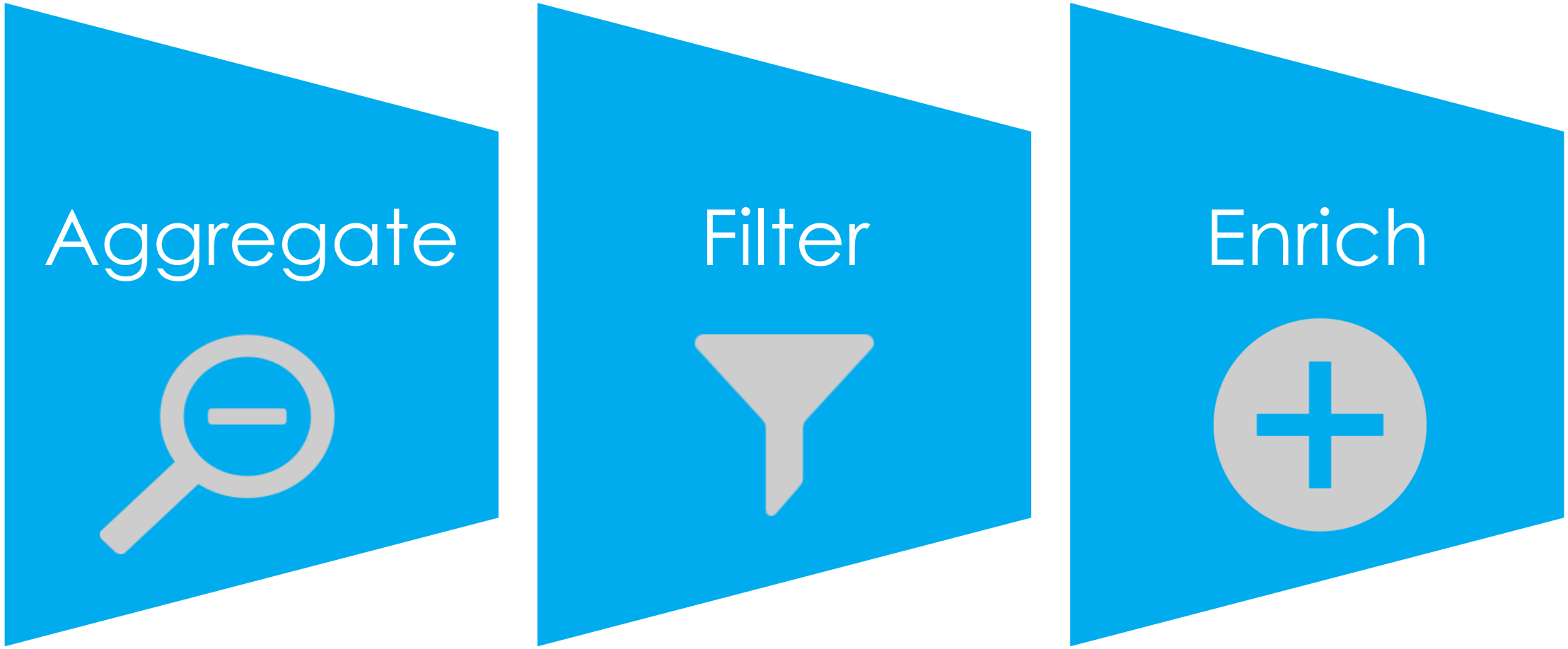
Open *2-exercises_part2.Rmd* in the Files pane of Rstudio



#3 Preprocessing

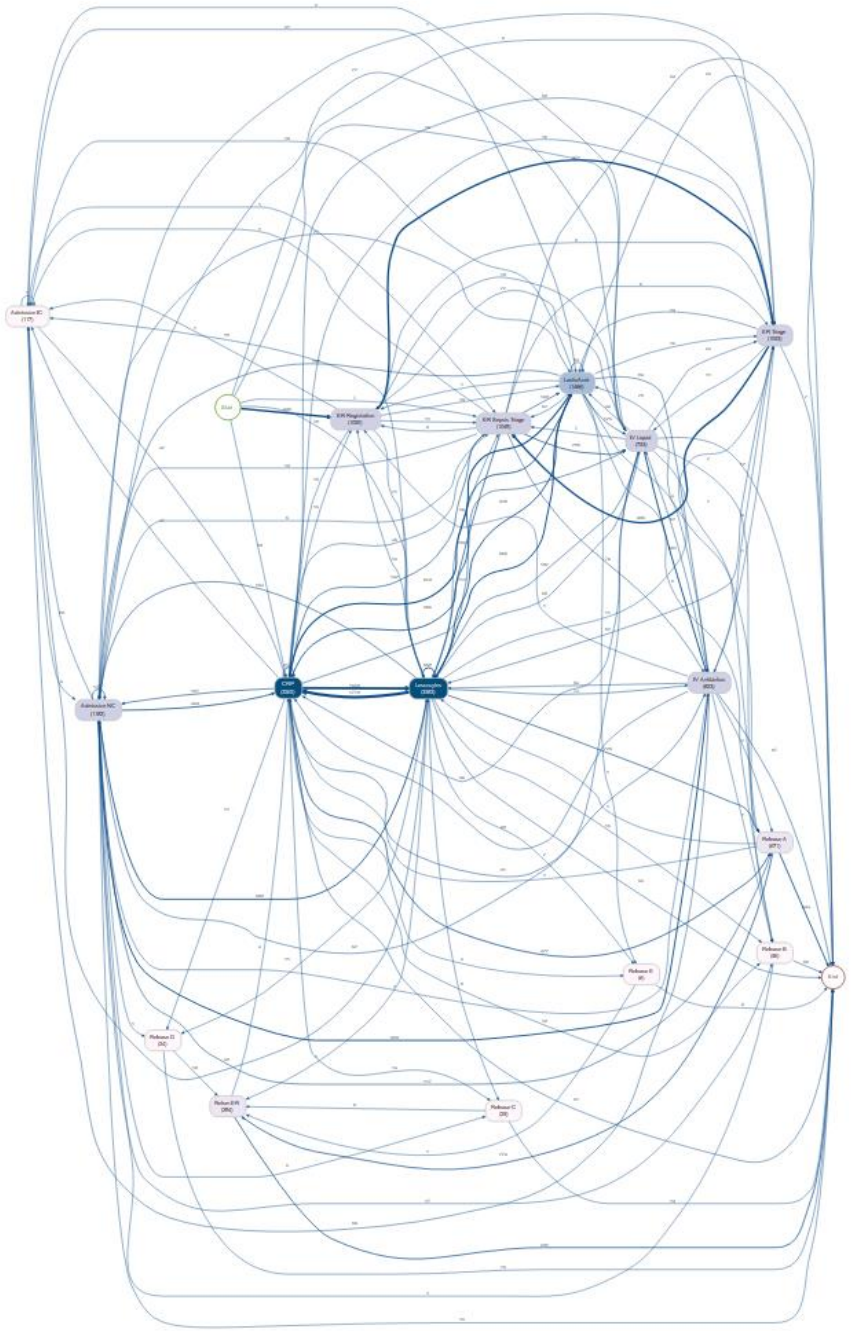
Aggregating

Filtering


Aggregate
Filter
Enrich

Writing Readable code

```
process_map(sepsis)
```



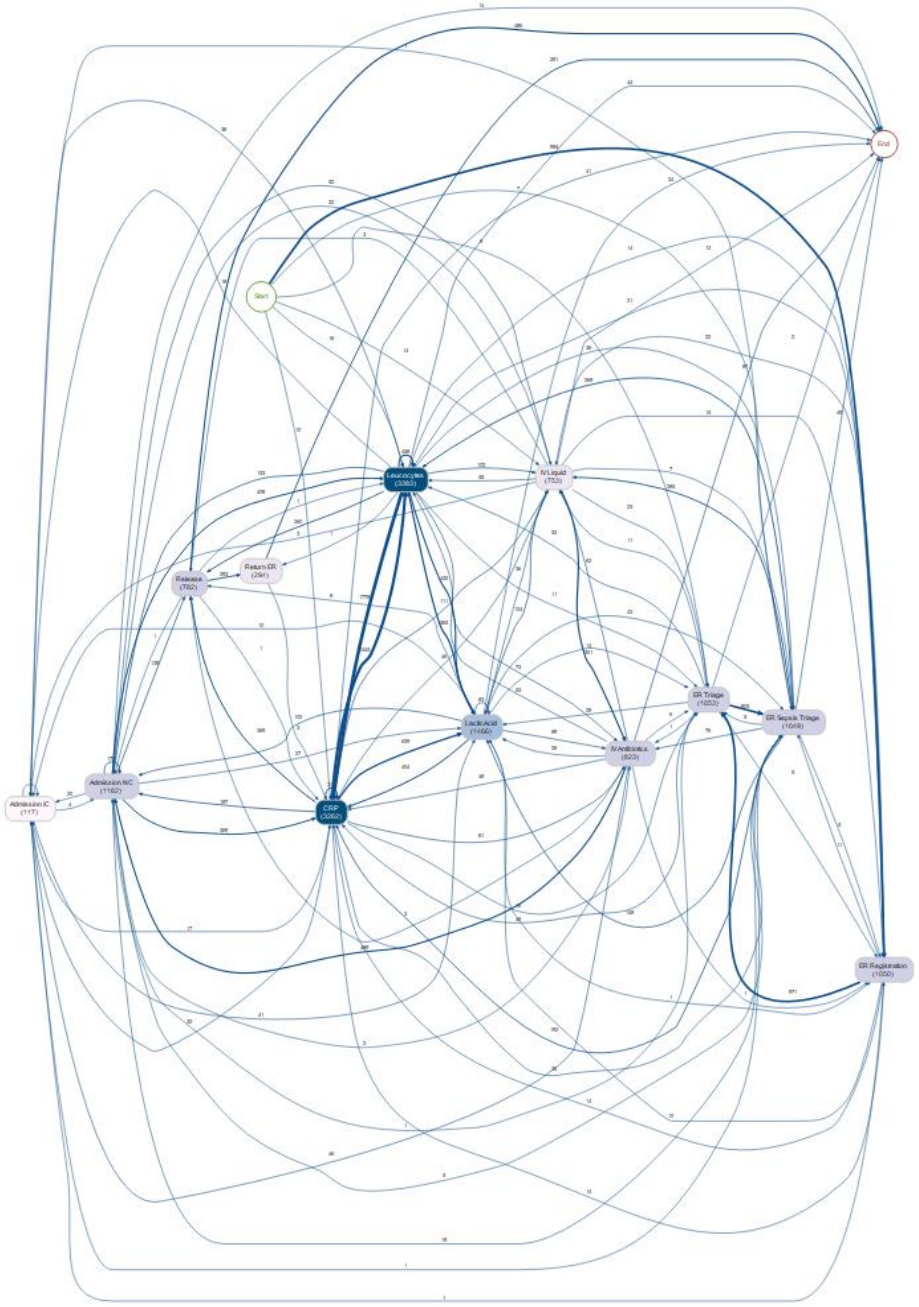
```
> activities(sepsis)
# A tibble: 16 x 3
   activity         absolute_frequency relative_frequency
   <fct>                         <int>              <dbl>
 1 Leucocytes                     3383           0.222
 2 CRP                            3262           0.214
 3 LacticAcid                     1466           0.0964
 4 Admission NC                   1182           0.0777
 5 ER Triage                      1053           0.0692
 6 ER Registration                1050           0.0690
 7 ER Sepsis Triage               1049           0.0689
 8 IV Antibiotics                  823           0.0541
 9 IV Liquid                       753           0.0495
10 Release A                       671           0.0441
11 Return ER                       294           0.0193
12 Admission IC                    117           0.00769
13 Release B                        56           0.00368
14 Release C                        25           0.00164
15 Release D                        24           0.00158
16 Release E                         6           0.000394
```

Writing Readable code

```
process_map(

        act_unite(sepsis,
        "Release" = c("Release A",
                      "Release B",
                      "Release C",
                      "Release D",
                      "Release E")

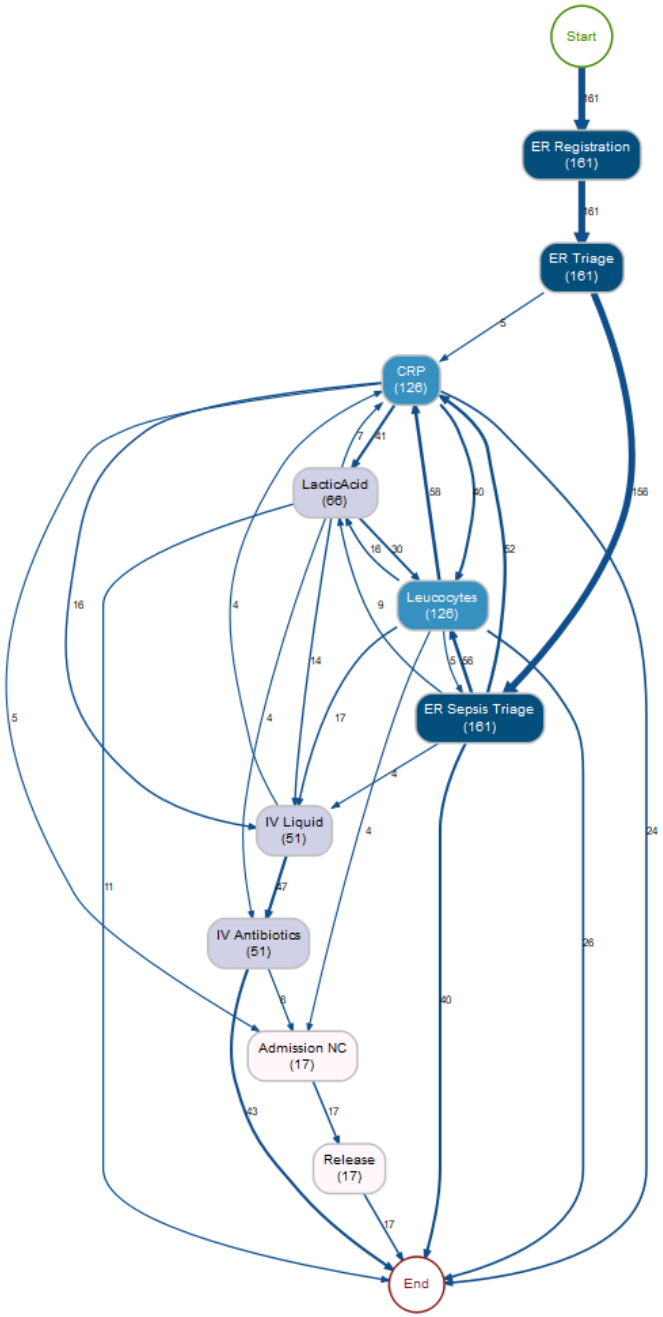
        )
)
```



```
> trace_coverage(sepsis, “trace”)
# A tibble: 842 x 4
   trace                                                  absolute relative cum_sum
   <chr>                                                     <int>    <dbl>   <dbl>
 1 ER Registration,ER Triage,ER Sepsis Triage                   35  0.0333   0.0333
 2 ER Registration,ER Triage,ER Sepsis Triage,Leucocyt~         24  0.0229   0.0562
 3 ER Registration,ER Triage,ER Sepsis Triage,CRP,Leuc~         22  0.0210   0.0771
 4 ER Registration,ER Triage,ER Sepsis Triage,CRP,Lact~         13  0.0124   0.0895
 5 ER Registration,ER Triage,ER Sepsis Triage,Leucocyt~         11  0.0105   0.100
 6 ER Registration,ER Triage,ER Sepsis Triage,Leucocyt~          9  0.00857  0.109
 7 ER Registration,ER Triage,ER Sepsis Triage,Leucocyt~          7  0.00667  0.115
 8 ER Registration,ER Triage,ER Sepsis Triage,Leucocyt~          5  0.00476  0.120
 9 ER Registration,ER Triage,ER Sepsis Triage,LacticAc~          5  0.00476  0.125
10 ER Registration,ER Triage,ER Sepsis Triage,CRP,Leuc~          5  0.00476  0.130
```

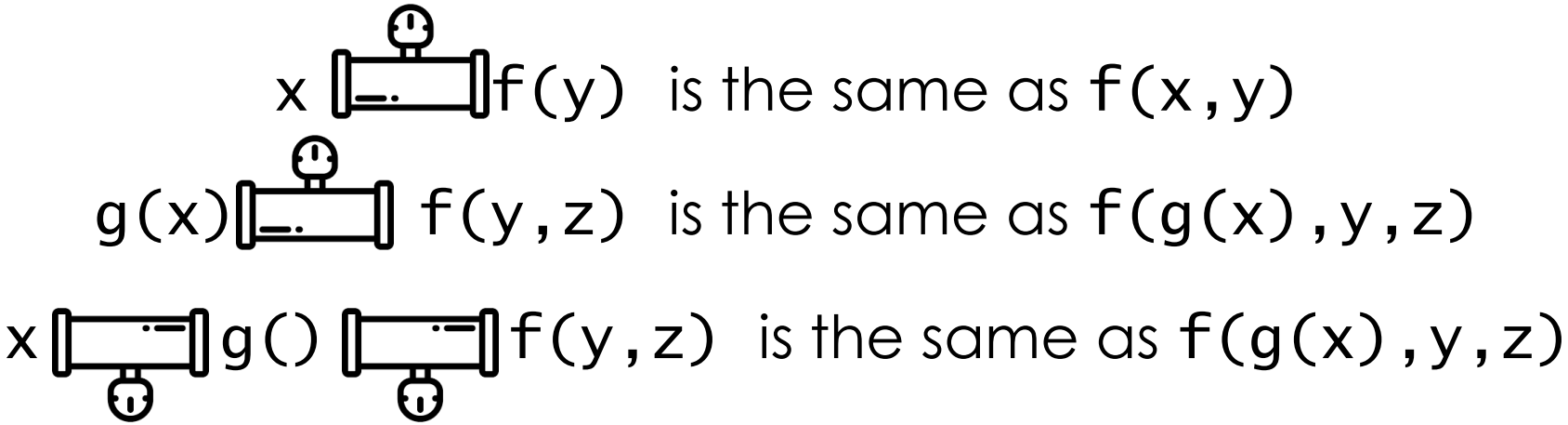
Writing Readable code

```
process_map(
        filter_trace_frequency(
                act_unite(sepsis,
                "Release" = c("Release A",
                              "Release B",
                              "Release C",
                              "Release D",
                              "Release E")
                ),
        percentage = 0.15)
)
```



%>%

This symbol allows you to take the first argument *outside*



It is called the ***piping*** symbol, because it functions as a ***data pipe***

Courtesy of `magrittr` package

Writing Readable code

```
process_map(
        filter_trace_frequency(
                act_unite(sepsis,
                "Release" = c("Release A",
                              "Release B",
                              "Release C",
                              "Release D",
                              "Release E")
                ),
        percentage = 0.15)
)
```

```
filter_trace_frequency(
                act_unite(sepsis,
                "Release" = c("Release A",
                              "Release B",
                              "Release C",
                              "Release D",
                              "Release E")
                ),
        percentage = 0.15) %>%
        process_map()
```

Writing Readable code

```
filter_trace_frequency(
                  act_unite(sepsis,
                  "Release" = c("Release A",
                                "Release B",
                                "Release C",
                                "Release D",
                                "Release E")
                  ),
        percentage = 0.15) %>%
        process_map()
```

```
act_unite(sepsis,
                  "Release" = c("Release A",
                                "Release B",
                                "Release C",
                                "Release D",
                                "Release E")
                  ) %>%
        filter_trace_frequency(
                  percentage = 0.15) %>%
        process_map()
```

Writing Readable code

```
act_unite(sepsis,
                  "Release" = c("Release A",
                                "Release B",
                                "Release C",
                                "Release D",
                                "Release E")
                  ) %>%
        filter_trace_frequency(
                  percentage = 0.15) %>%
        process_map()
```

```
sepsis %>%
        act_unite("Release" = c("Release A",
                                "Release B",
                                "Release C",
                                "Release D",
                                "Release E")
                  ) %>%
        filter_trace_frequency(
                  percentage = 0.15) %>%
        process_map()
```

Writing Readable code

Before

```
process_map(
        filter_trace_frequency(
                act_unite(sepsis,
                "Release" = c("Release A",
                              "Release B",
                              "Release C",
                              "Release D",
                              "Release E")
                ),
        percentage = 0.15)
)
```

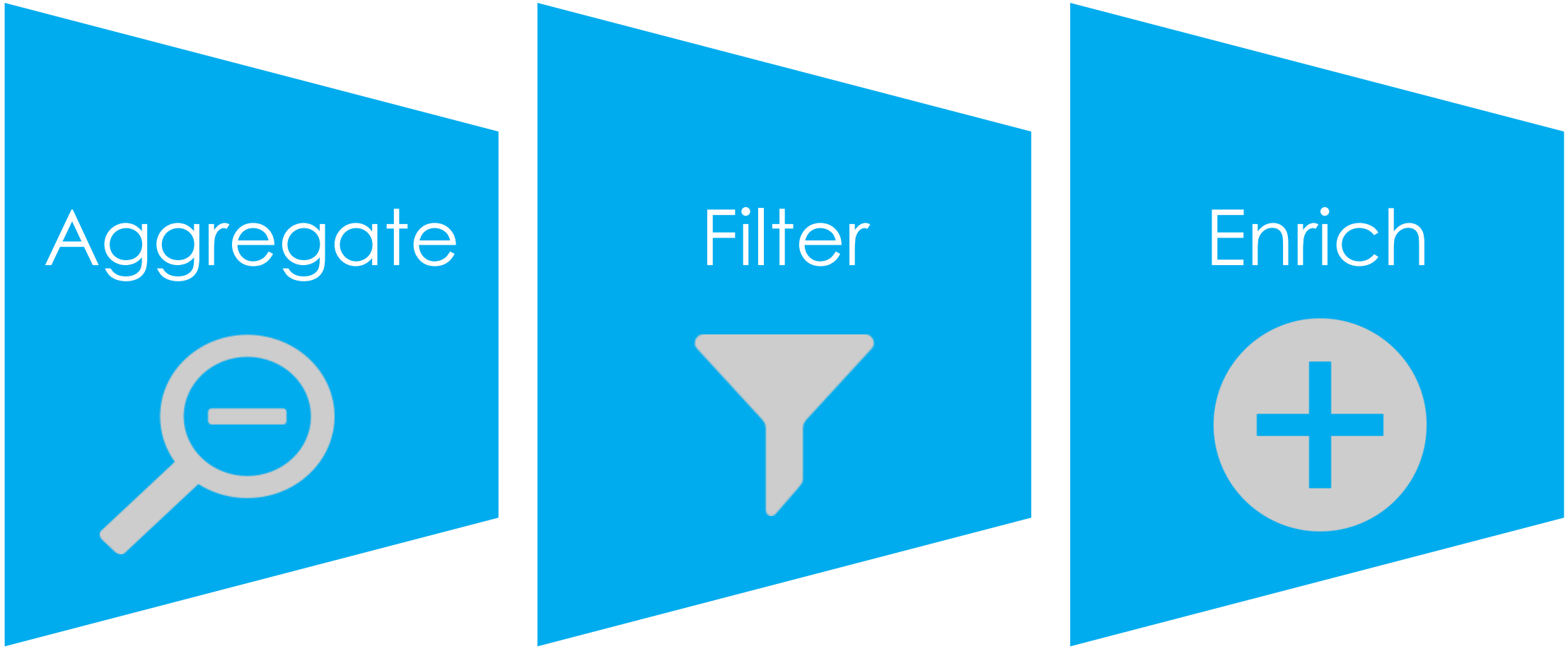
After

```
sepsis %>%
        act_unite("Release" = c("Release A",
                                "Release B",
                                "Release C",
                                "Release D",
                                "Release E")
                ) %>%
        filter_trace_frequency(
                percentage = 0.15) %>%
        process_map()
```

Writing Readable code

1. Take the data
2. Unite activities
3. Filter frequent traces
4. Draw process map
5. Save map as object map

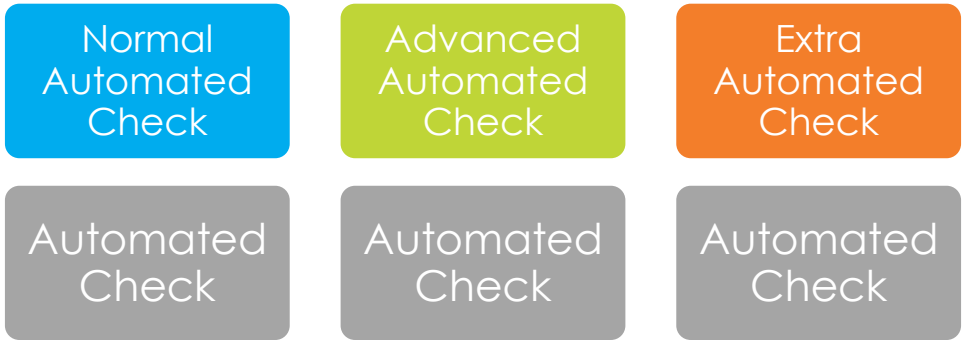
```
sepsis %>%
        act_unite("Release" = c("Release A",
                                "Release B",
                                "Release C",
                                "Release D",
                                "Release E")
                ) %>%
        filter_trace_frequency(
                percentage = 0.15) %>%
        process_map() -> map
```


Aggregate
Filter
Enrich

Aggregating process data

Remove unnecessary information

Is-a



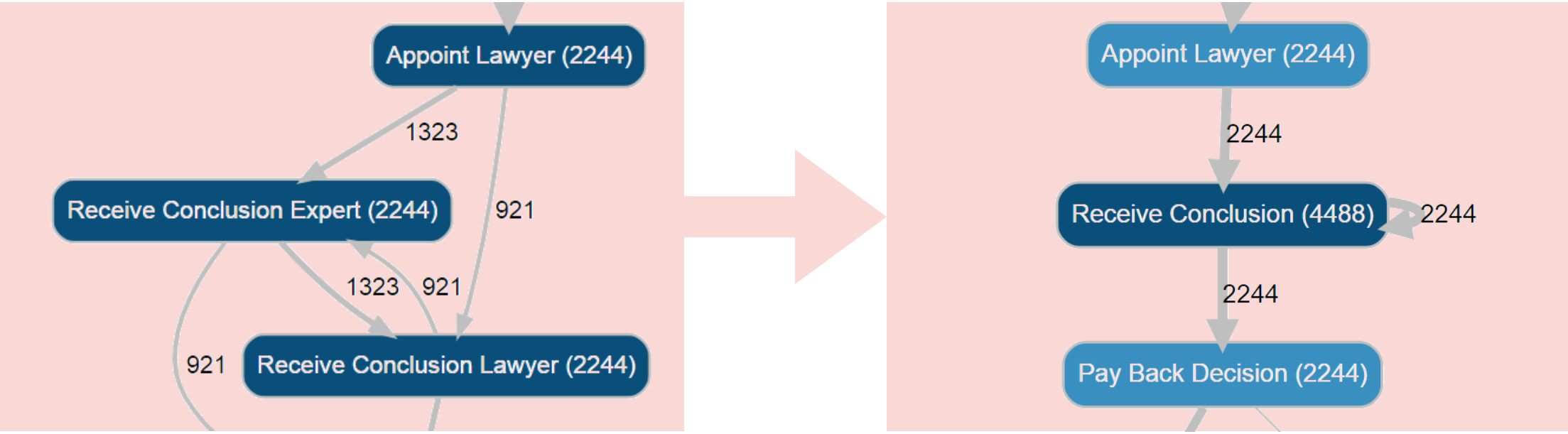
Eliminate differences between similar activity labels

Part-of

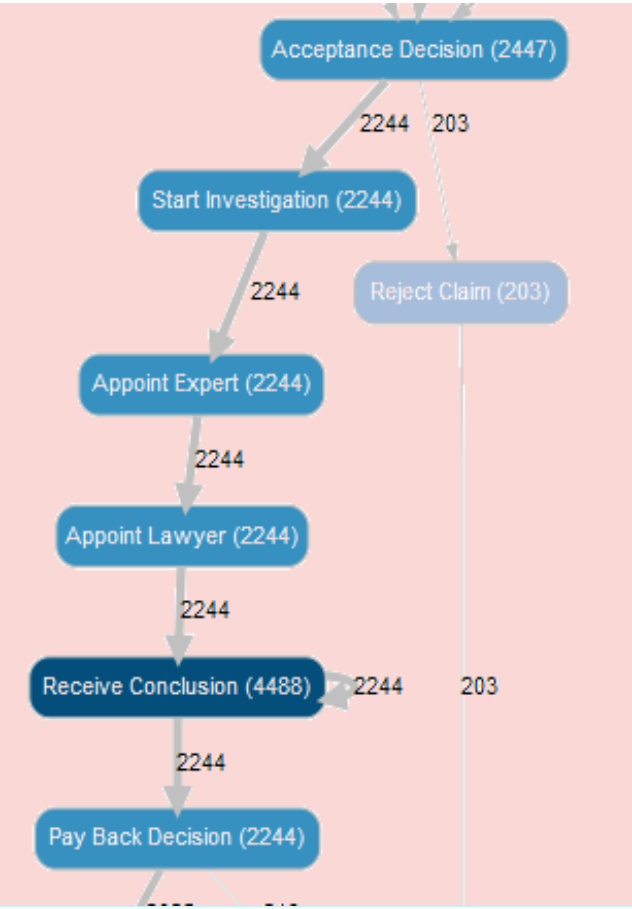
Collapse related activity labels into one label

Is-a aggregation

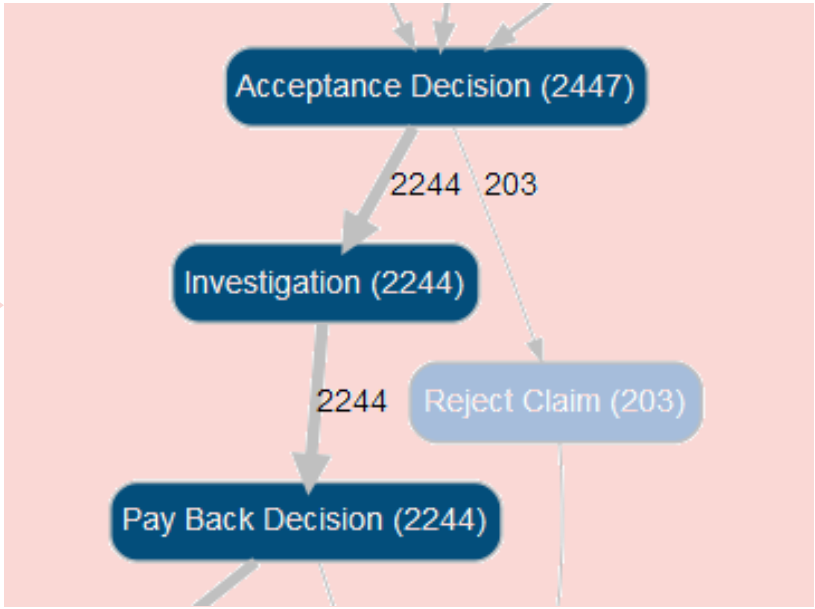
```
log_claims %>%
   act_unite("Receive Conclusion" = c("Receive Conclusion Expert",
                                      "Receive Conclusion Lawyer"))
```



Part-of aggregation



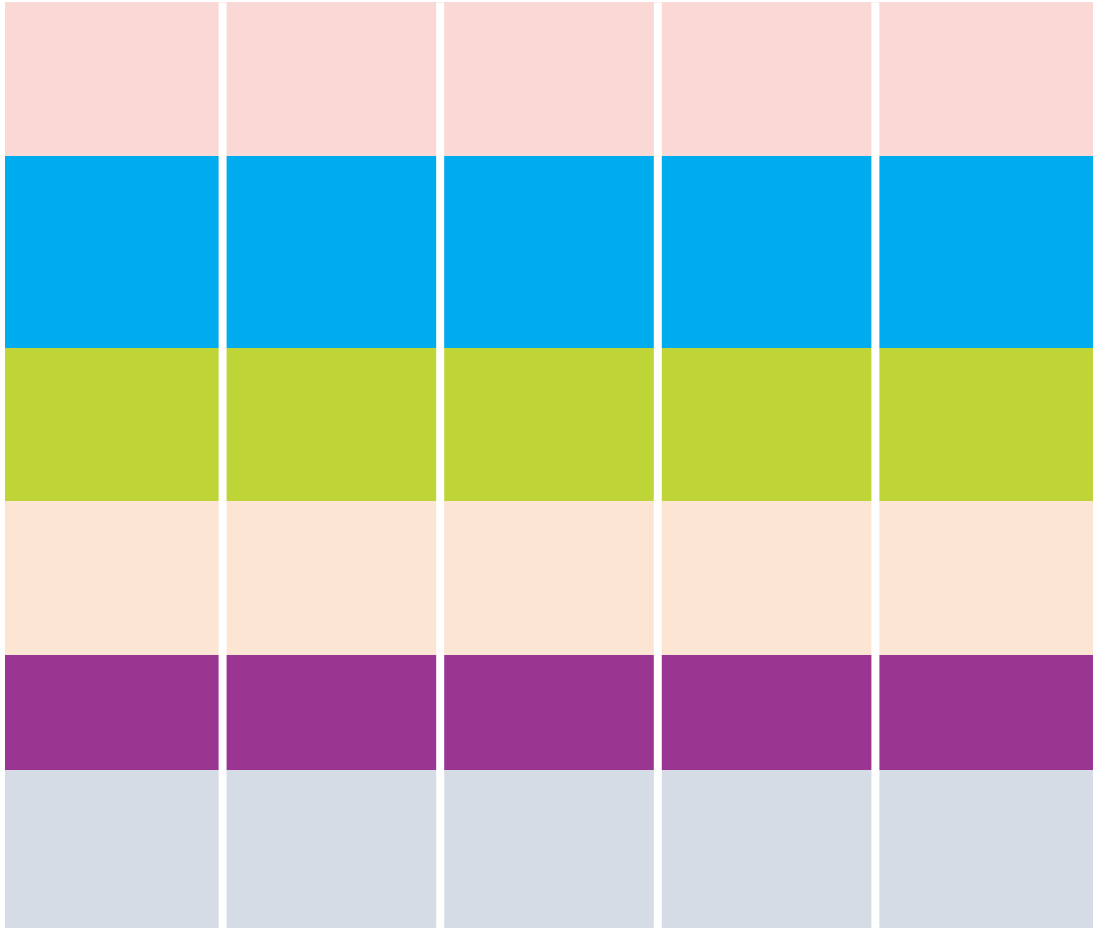

```
log_claims %>%
   act_collapse("Investigation" = c("Start Investigation",
                                    "Appoint Lawyer",
                                    "Appoint Expert",
                                    "Receive Conclusions"))
```



Filter event data

Define the scope of the analysis

Select Cases



Select Events

Case filters

Performance

Processing time	`filter_processing_time`
Throughput time	`filter_throughpout_time`
Idle time	`filter_idle_time`
Trace length	`filter_trace_length`

Control-flow

Activity presence	`filter_activity_presence`
End points	`filter_end_points`
Precendence	`filter_precedence`
Trace frequency	`filter_trace_frequency`

Time

Time period	`filter_time_period`

Other

Case id	`filter_case`

Performance filters

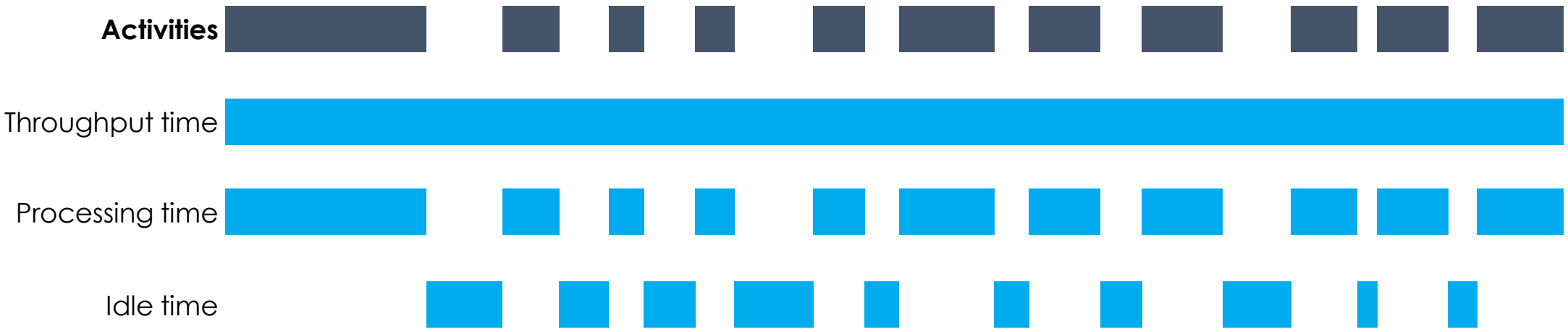
Processing time

Throughput time

Idle time

Trace length

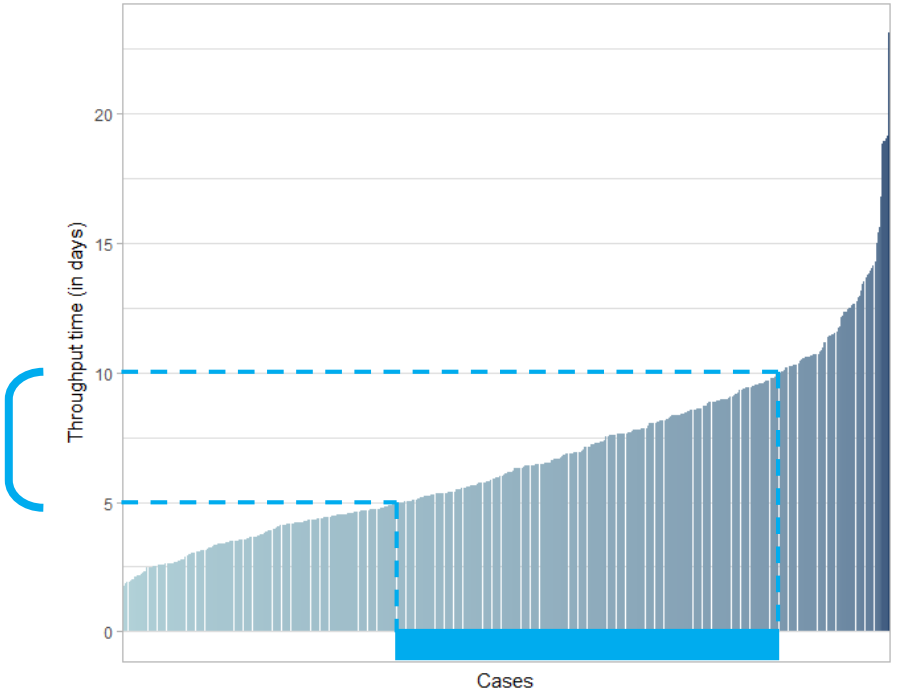
Performance filters



Performance filters

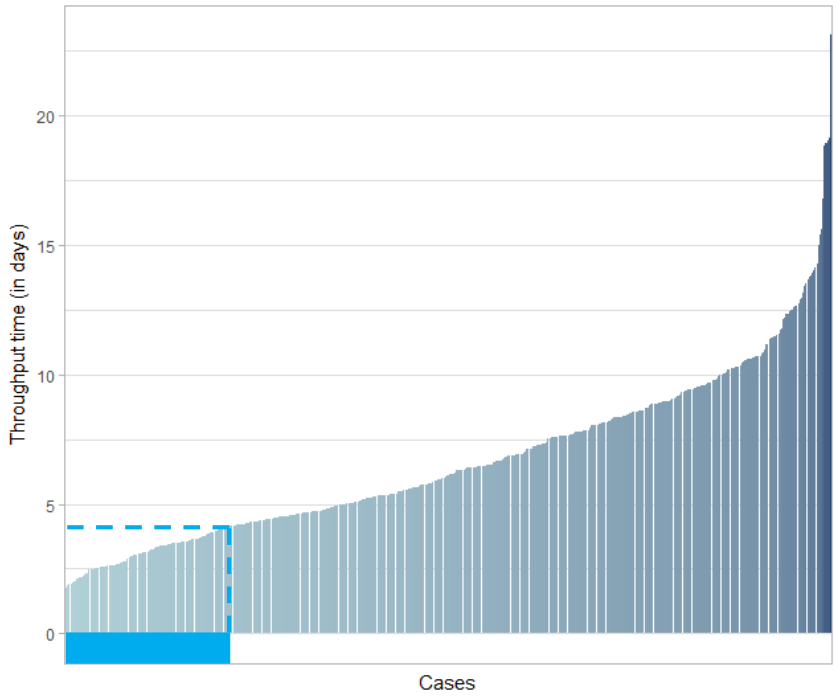
Absolute

E.g. cases with throughput time between 5 and 10 days



Relative

E.g. 20% shortest cases according to throughput time



Performance filters

```
# filter by percentage
> log_claims %>%
      filter_throughput_time(percentage = 0.2)
# filter by interval
> log_claims %>%
      filter_throughput_time(interval = c(5,10), units = “days”)
```

Performance filters

The **reverse** argument can be used to negate filters

```
# filter by percentage
> log_claims %>%
      filter_throughput_time(percentage = 0.2,
                             reverse = T)
# filter by interval
> log_claims %>%
      filter_throughput_time(interval = c(5,10), units = “days”,
                             reverse = T)
```

Performance filters

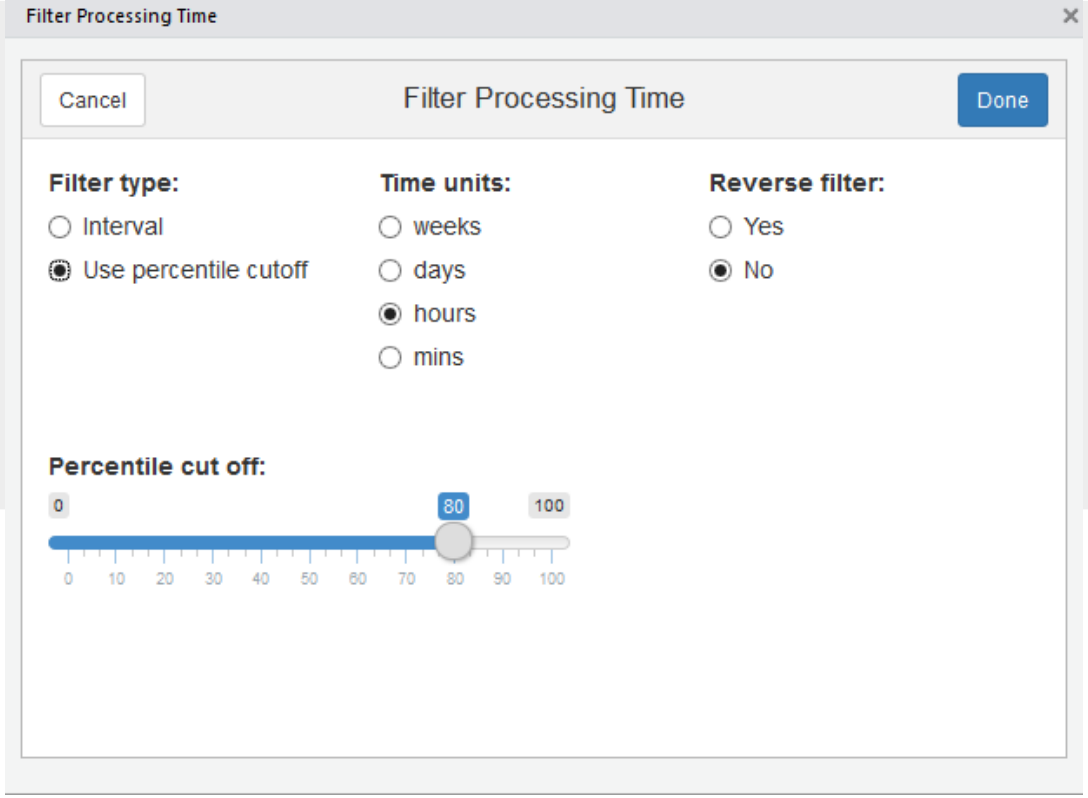
NA can be used for half-open intervals

```
# filter by percentage
> log_claims %>%
      filter_throughput_time(percentage = 0.2,
                             reverse = T)
# filter by interval
> log_claims %>%
      filter_throughput_time(interval = c(5,NA), units = “days”,
                             reverse = T)
```

Performance filters

Adding an i before the filter opens a graphical interface to help

```
log_claims %>%
      ifilter_processing_time()
```



Case filters

Performance

Processing time	`filter_processing_time`
Throughput time	`filter_throughpout_time`
Idle time	`filter_idle_time`
Trace length	`filter_trace_length`

Control-flow

Activity presence	`filter_activity_presence`
End points	`filter_end_points`
Precendence	`filter_precedence`
Trace frequency	`filter_trace_frequency`

Control-flow filters

Activity presence

End points

Precedence

Trace frequency

Activity presence

Filter cases in which a (set of) activities occurs

```
> log_claims %>%
    filter_activity_presence("Reject Claim")
```

Accd	FICl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FICl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FICl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FICl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FICl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FICl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FICl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FICl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FICl	ChcC	Frn?	Cvr?	AccD	RjcC						
Accd	FICl	ChcC	Cvr?	Frn?	AccD	RjcC						
Accd	FICl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICl	Cvr?	Frn?	ChcC	AccD	RjcC						
Accd	FICl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FICl	Frn?	Cvr?	ChcC	AccD	RjcC						
Accd	FICl	Cvr?	ChcC	Frn?	AccD	RjcC						
Accd	FICl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICl	Frn?	ChcC	Cvr?	AccD	RjcC						
Accd	FICl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn

Activity presence

When specifying more than one activity, the `method` can be used.

“all”	each should be present in the case
“one_of”	at least one should be present
“none”	none are allowed to be present.

Activity presence

```
# Keep cases with both activities
> log_claims %>%
      filter_activity_presence(c("Reject Claim","No refund"),
                               method = "all")
# Keep cases with at least one of the activities
> log_claims %>%
      filter_activity_presence(c("Reject Claim","No refund"),
                               method = "one_of")
# Keep cases without the activities
> log_claims %>%
      filter_activity_presence(c("Reject Claim","No refund"),
                               method = "none")
```

Accd	FlCl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	RjcC						
Accd	FlCl	ChcC	Cvr?	Frn?	AccD	RjcC						
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	RjcC						
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	RjcC						
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	RjcC						
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	RjcC						
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn

a11

Accd	FICl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FICl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FICl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FICl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FICl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FICl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FICl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FICl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FICl	ChcC	Frn?	Cvr?	AccD	RjcC						
Accd	FICl	ChcC	Cvr?	Frn?	AccD	RjcC						
Accd	FICl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICl	Cvr?	Frn?	ChcC	AccD	RjcC						
Accd	FICl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FICl	Frn?	Cvr?	ChcC	AccD	RjcC						
Accd	FICl	Cvr?	ChcC	Frn?	AccD	RjcC						
Accd	FICl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICl	Frn?	ChcC	Cvr?	AccD	RjcC						
Accd	FICl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn

one_of

Accd	FICI	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCI
Accd	FICI	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCI
Accd	FICI	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCI
Accd	FICI	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCI
Accd	FICI	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCI
Accd	FICI	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCI
Accd	FICI	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FICI	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCI
Accd	FICI	ChcC	Frn?	Cvr?	AccD	RjcC						
Accd	FICI	ChcC	Cvr?	Frn?	AccD	RjcC						
Accd	FICI	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICI	Cvr?	Frn?	ChcC	AccD	RjcC						
Accd	FICI	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICI	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FICI	Frn?	Cvr?	ChcC	AccD	RjcC						
Accd	FICI	Cvr?	ChcC	Frn?	AccD	RjcC						
Accd	FICI	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICI	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICI	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICI	Frn?	ChcC	Cvr?	AccD	RjcC						
Accd	FICI	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn

none

Accd	FlCl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	RjcC						
Accd	FlCl	ChcC	Cvr?	Frn?	AccD	RjcC						
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	RjcC						
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	RjcC						
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	RjcC						
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	RjcC						
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn

Control-flow filters

Activity presence

End points

Precedence

Trace frequency

End points

Filter cases with a specific start and/or end point.

```
> log_claims %>%
    filter_endpoints(start_activities = c("Accident"),
                     end_activities   = c("Reject Claim"))
```

Accd	FICl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FICl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FICl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FICl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FICl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FICl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FICl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FICl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FICl	ChcC	Frn?	Cvr?	AccD	RjcC						
Accd	FICl	ChcC	Cvr?	Frn?	AccD	RjcC						
Accd	FICl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICl	Cvr?	Frn?	ChcC	AccD	RjcC						
Accd	FICl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FICl	Frn?	Cvr?	ChcC	AccD	RjcC						
Accd	FICl	Cvr?	ChcC	Frn?	AccD	RjcC						
Accd	FICl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FICl	Frn?	ChcC	Cvr?	AccD	RjcC						
Accd	FICl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn

End points

Can also be used with one of the arguments only

```
> log_claims %>%
    filter_endpoints(start_activities = c("Accident"),
                     end_activities = c("Reject Claim"))
> log_claims %>%
    filter_endpoints(end_activities = c("Reject Claim"))
```

End points

Can also be used with more than one activity

```
> log_claims %>%
      filter_endpoints(start_activities = c("Accident"),
                       end_activities = c("Reject Claim"))

> log_claims %>%
      filter_endpoints(end_activities = c("Reject Claim"))

> log_claims %>%
      filter_endpoints(start_activities = c("Accident"),
                       end_activities = c("Reject Claim",
                                          "No refund"))
```

Accd	FlCl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	RjcC						
Accd	FlCl	ChcC	Cvr?	Frn?	AccD	RjcC						
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	RjcC						
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	RjcC						
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	RjcC						
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	RjcC						
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn

Control-flow filters

Activity presence

End points

Precedence

Trace frequency

Precedence

Filter cases in which a two activities are directly, or eventually followed by each other.

```
# Filter cases where “Coverage?” is directly followed by “Franchise?”
> log_claims %>%
            filter_precedence(antecedent = “Coverage?”,
                              consequent = “Franchise?”,
                              precedence_type = “directly_follows”)
```

Accd	FlCl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	RjcC						
Accd	FlCl	ChcC	Cvr?	Frn?	AccD	RjcC						
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	RjcC						
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	RjcC						
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	RjcC						
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	RjcC						
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn

Precedence

Can also be used less strict with argument “eventually_follows”

```
> log_claims %>% filter_precedence(antecedent = “Coverage?”,
                                   consequent = “Franchise?”,
                                   precedence_type = “directly_follows”)
> log_claims %>% filter_precedence(antecedent = “Coverage?”,
                                   consequent = “Franchise?”,
                                   precedence_type = “eventually_follows”)
```

Accd	FlCl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	RjcC						
Accd	FlCl	ChcC	Cvr?	Frn?	AccD	RjcC						
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	RjcC						
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	RjcC						
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	RjcC						
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	RjcC						
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn

Precedence

Can also be used with multiple activity pairs

```
# Directly follows
> log_claims %>% filter_precedence(antecedent = “Coverage?”,
                                   consequent = “Franchise?”,
                                   precedence_type = “directly_follows”)
# Eventually follows
> log_claims %>% filter_precedence(antecedent = “Coverage?”,
                                   consequent = “Franchise?”,
                                   precedence_type = “eventually_follows”)
# Multiple pairs
> log_claims %>% filter_precedence(antecedent = “Coverage?”,
                                   consequent = c(“Franchise?”,
                                                  “Check contract”),
                                   precedence_type = “eventually_follows”,
                                   filter_method = “all/one_of/none”)
```

Control-flow filters

Activity presence

End points

Precedence

Trace frequency

Trace frequency

Similar to performance filters

```
# filter by percentage
> log_claims %>%
      filter_trace_frequency(percentage = 0.2)


# filter by interval
> log_claims %>%
      filter_trace_frequency(interval = c(5,10), units = “days”)
```

Case filters

Performance

Processing time	`filter_processing_time`
Throughput time	`filter_throughpout_time`
Idle time	`filter_idle_time`
Trace length	`filter_trace_length`

Control-flow

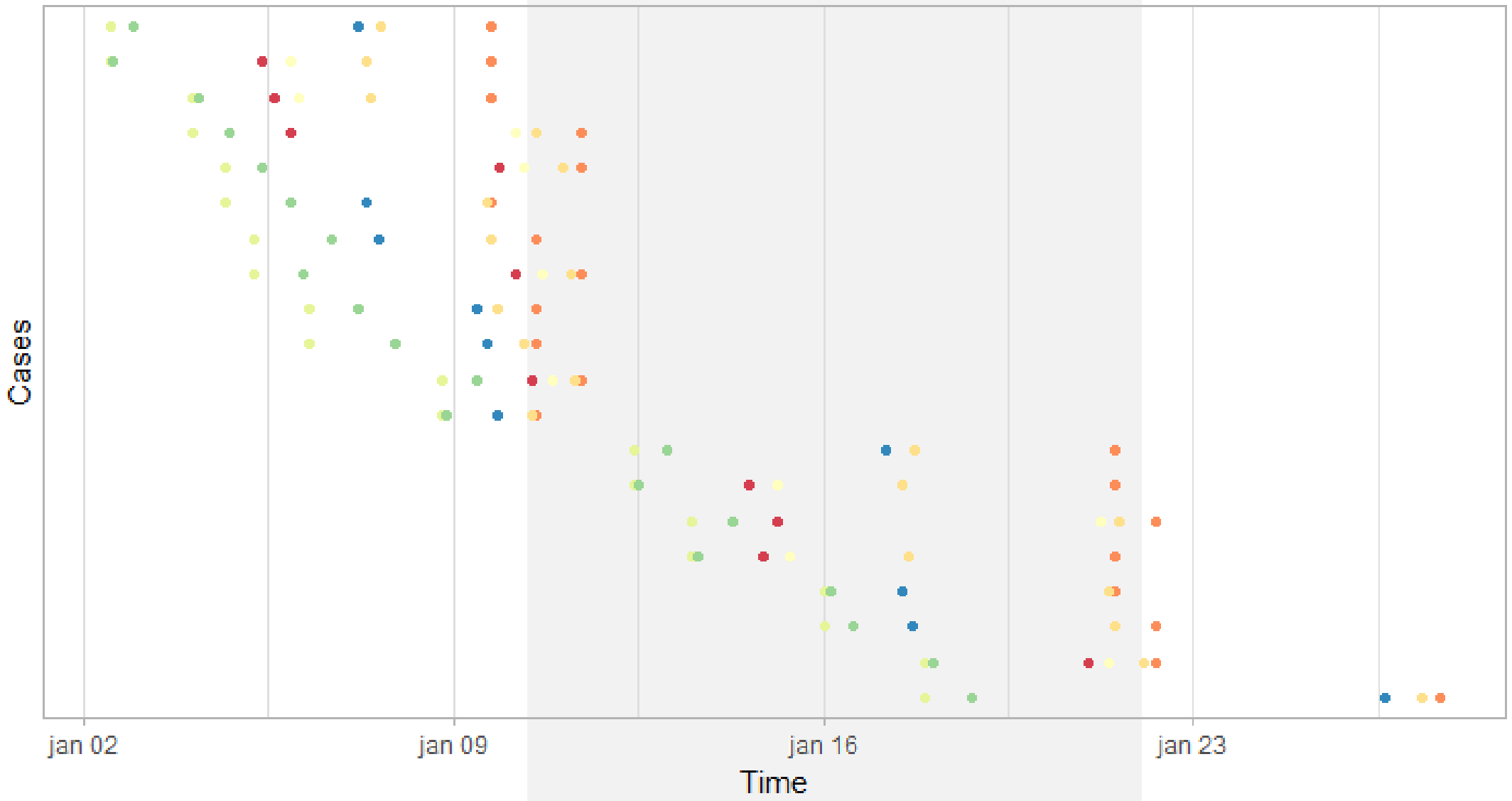
Activity presence	`filter_activity_presence`
End points	`filter_end_points`
Precendence	`filter_precedence`
Trace frequency	`filter_trace_frequency`

Time

Time period	`filter_time_period`

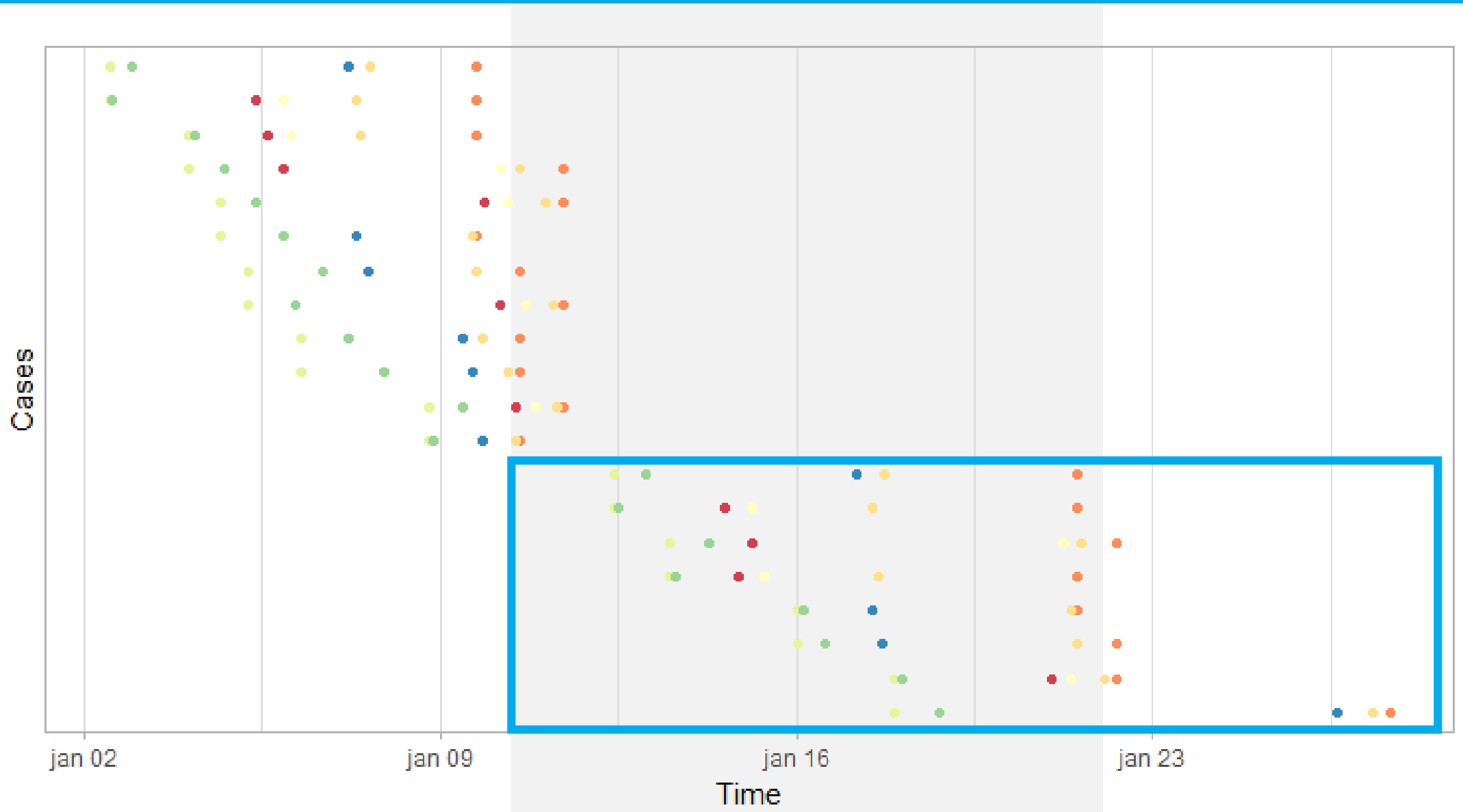
Time


```
> log_claims %>%
      filter_time_period(interval = ymd(c(20160101, 20160131)),
                         filter_method = “start/complete/
                                          contained/intersecting”)
```



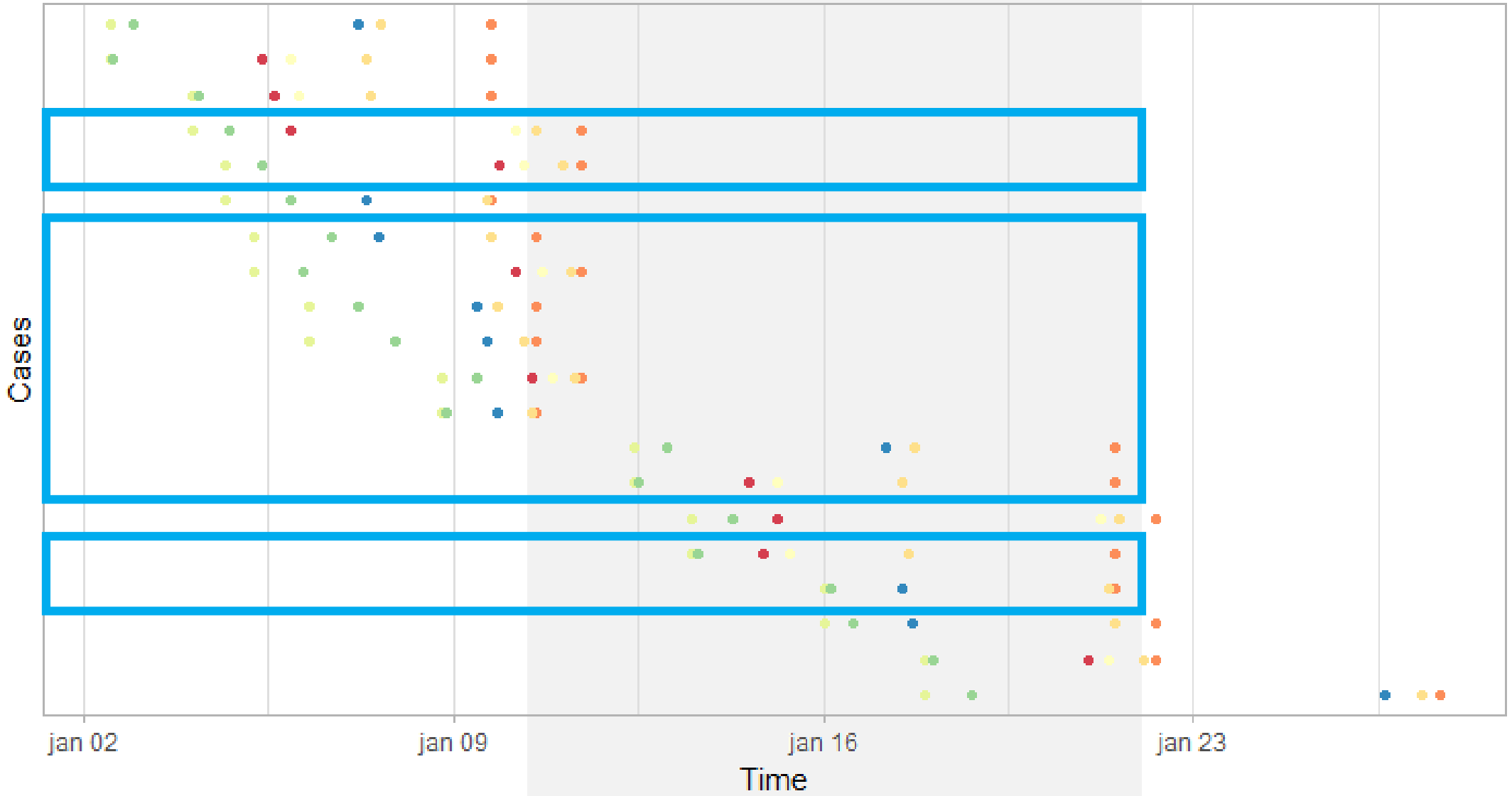
Time interval

filter_method = “start”



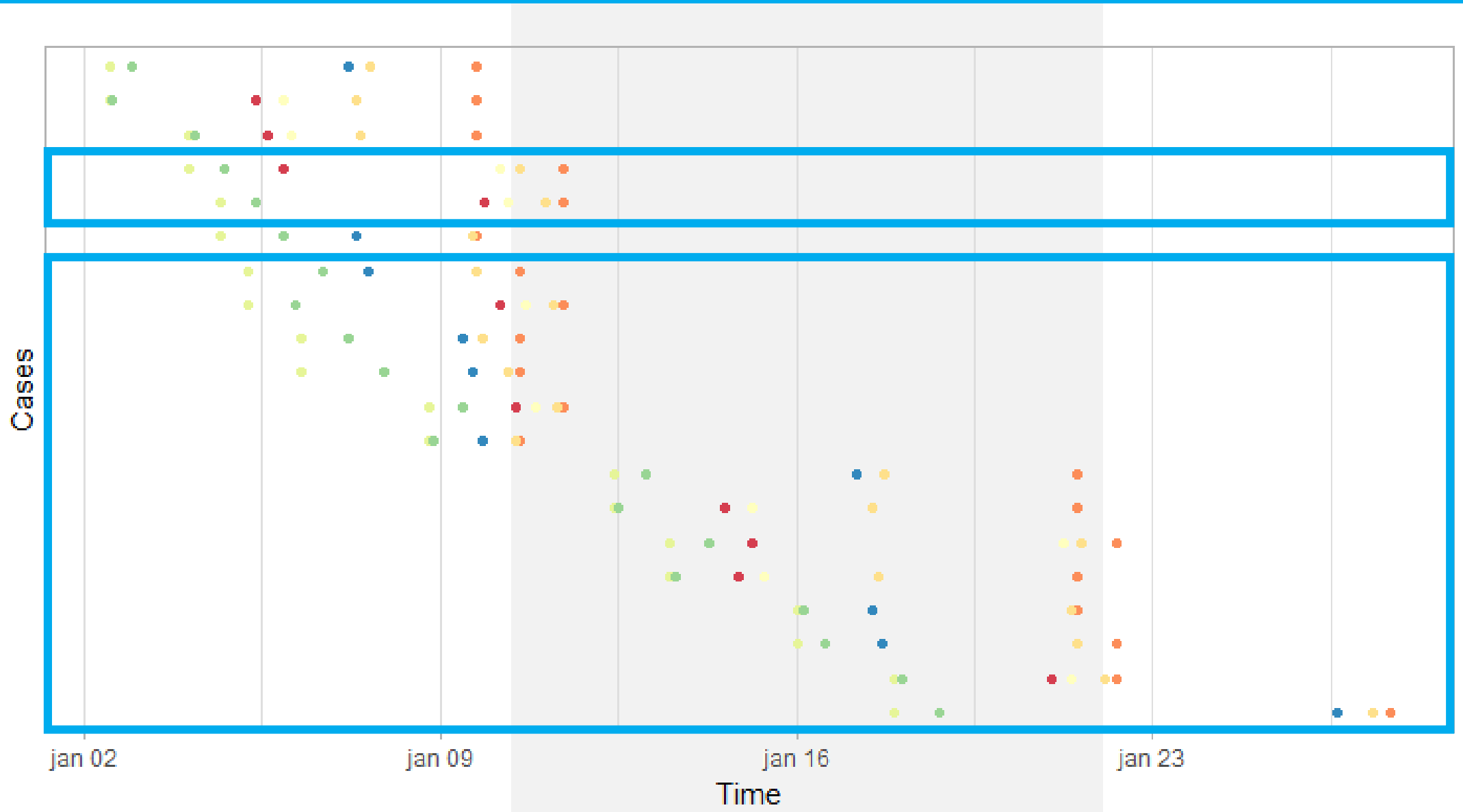
Time interval

filter_method = “complete”



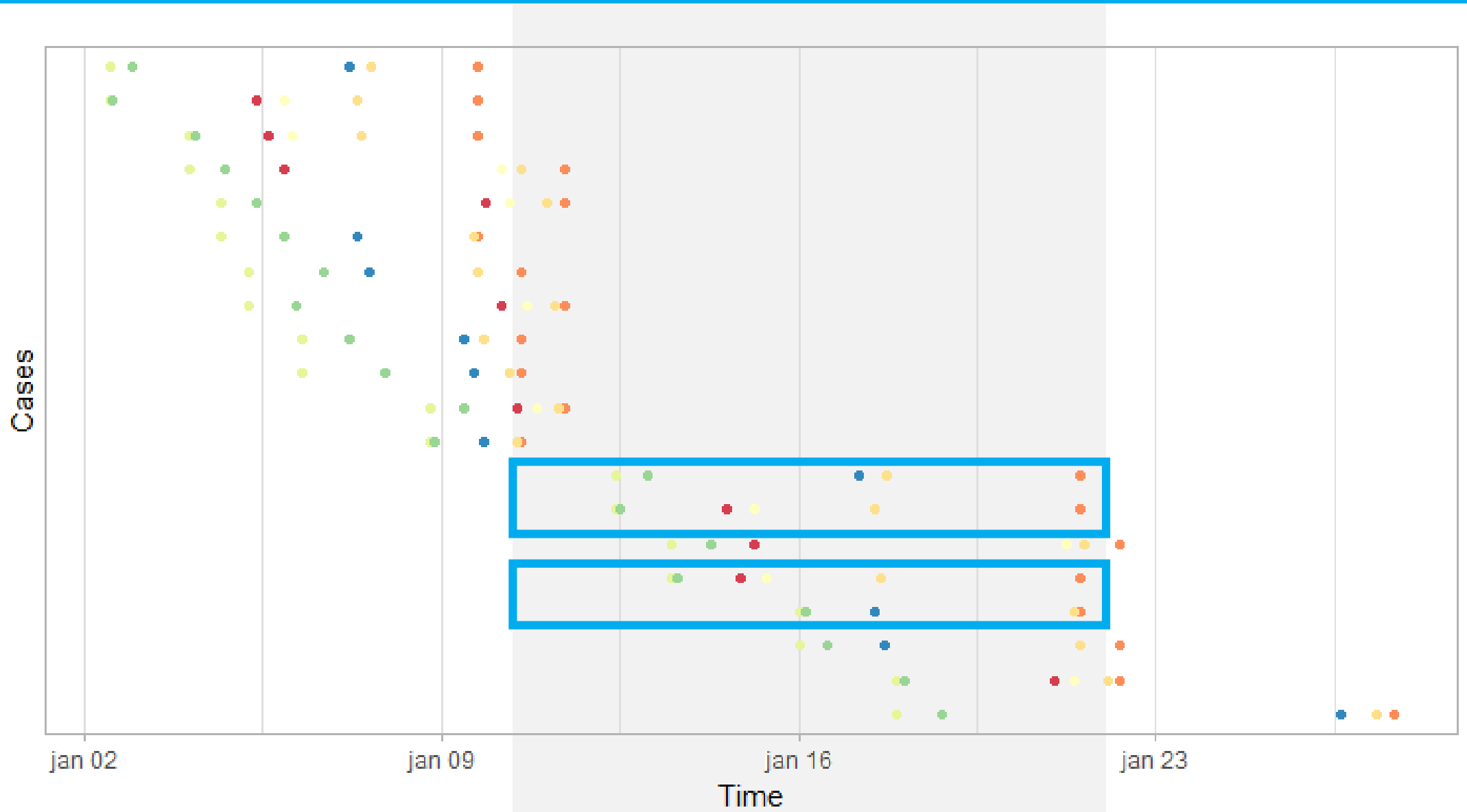
Time interval

filter_method = “intersecting”



Time interval

filter_method = “contained”



Time interval

Time

Also can be used with `reverse` or half-open intervals

```
log_claims %>%
      filter_time_period(interval = ymd(20160101, 20160131),
                         filter_method = “start”)
log_claims %>%
      filter_time_period(interval = ymd(20160101, 20160131),
                        filter_method = “start”,
                        reverse = TRUE)
log_claims %>%
      filter_time_period(interval = ymd(NA, 20160131),
                         filter_method = “start”)
```

Case filters

Performance	
Processing time	`filter_processing_time`
Throughput time	`filter_throughpout_time`
Idle time	`filter_idle_time`
Trace length	`filter_trace_length`
Control-flow	
Activity presence	`filter_activity_presence`
End points	`filter_end_points`
Precendence	`filter_precedence`
Trace frequency	`filter_trace_frequency`
Time	
Time period	`filter_time_period`
Other	
Case id	`filter_case`

Event filters

Trim

Trim to time period	`filter_time_period`
Trim to end points	`filter_trim`

By conditions

Attributes	`filter`

By label frequency

Activity frequency	`filter_activity_frequency`
Resource frequency	`filter_resource_frequency`

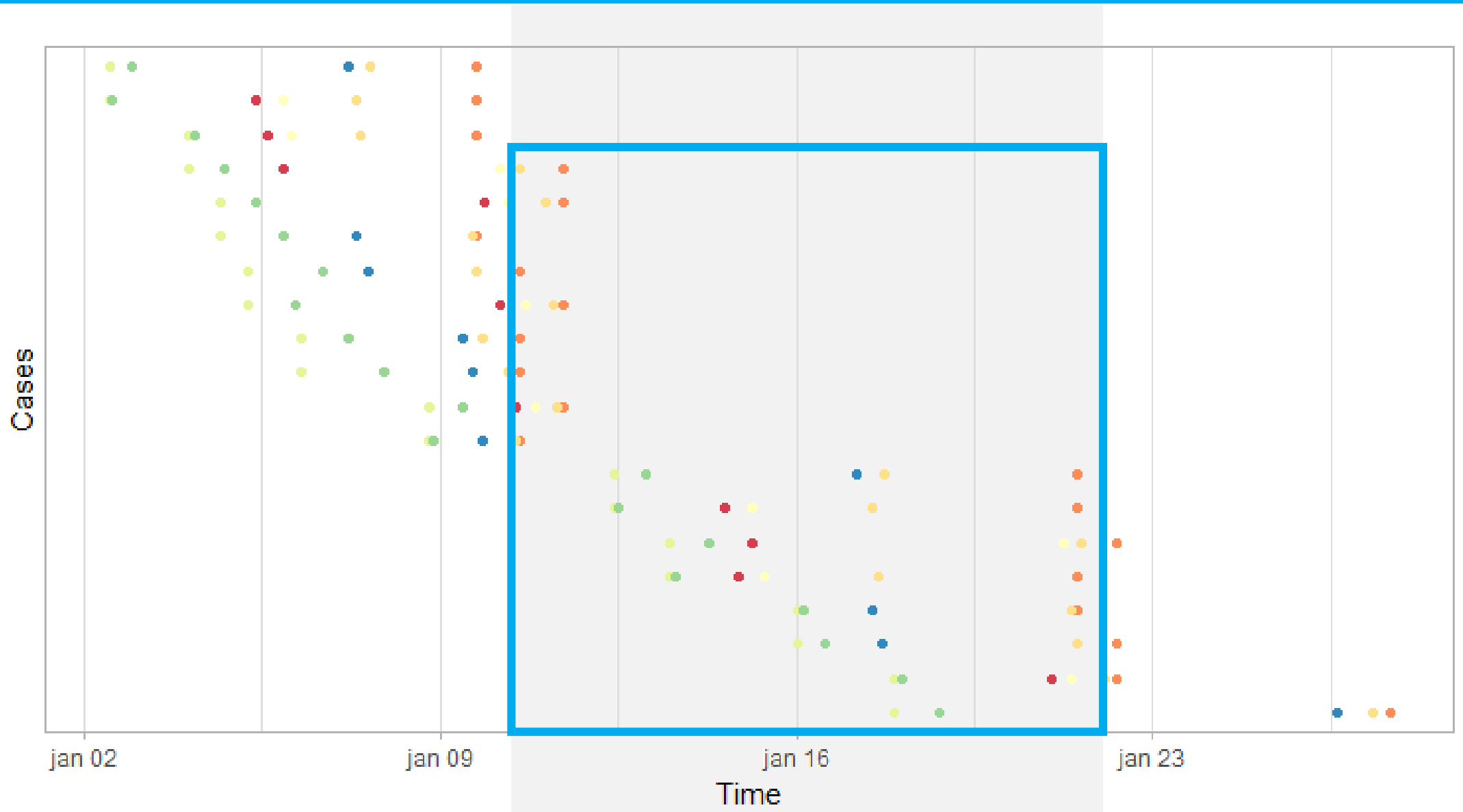
By label

Activity id	`filter_activity`
Resource id	`filter_resource`

Trim to Time Period


```
> log_claims %>%
    filter_time_period(interval = ymd(c(20160101, 20160131)),
                       filter_method = “trim”)
```

filter_method = “trim”



Time interval

Trim cases

Trim head and/or tails of cases

```
> log_claims %>% filter_trim(start_activities = “Check contract”,
                             end_activities = c(“Reject Claim”,
                                                “Appoint Lawyer”))
```

Not to be confused with `filter_endpoints`, which filters entire traces based on first/last activity

Accd	FlCl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	PyCl
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Cvr?	Frn?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	PyCl
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	RjcC						
Accd	FlCl	ChcC	Cvr?	Frn?	AccD	RjcC						
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	RjcC						
Accd	FlCl	Cvr?	Frn?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	ChcC	Frn?	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	RjcC						
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	RjcC						
Accd	FlCl	Cvr?	ChcC	Frn?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	Cvr?	ChcC	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCE	RcCL	PyBD	NRfn
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	RjcC						
Accd	FlCl	Frn?	ChcC	Cvr?	AccD	Strl	AppE	AppL	RcCL	RcCE	PyBD	NRfn

Others

Attributes	`filter`	Data conditions
Activity frequency	`filter_activity_frequency`	Interval/percentage
Resource frequency	`filter_resource_frequency`	Interval/percentage
Activity label	`filter_activity`	Activity labels
Resource label	`filter_resource`	Resource labels

Exercises Part 3

Let's play a game

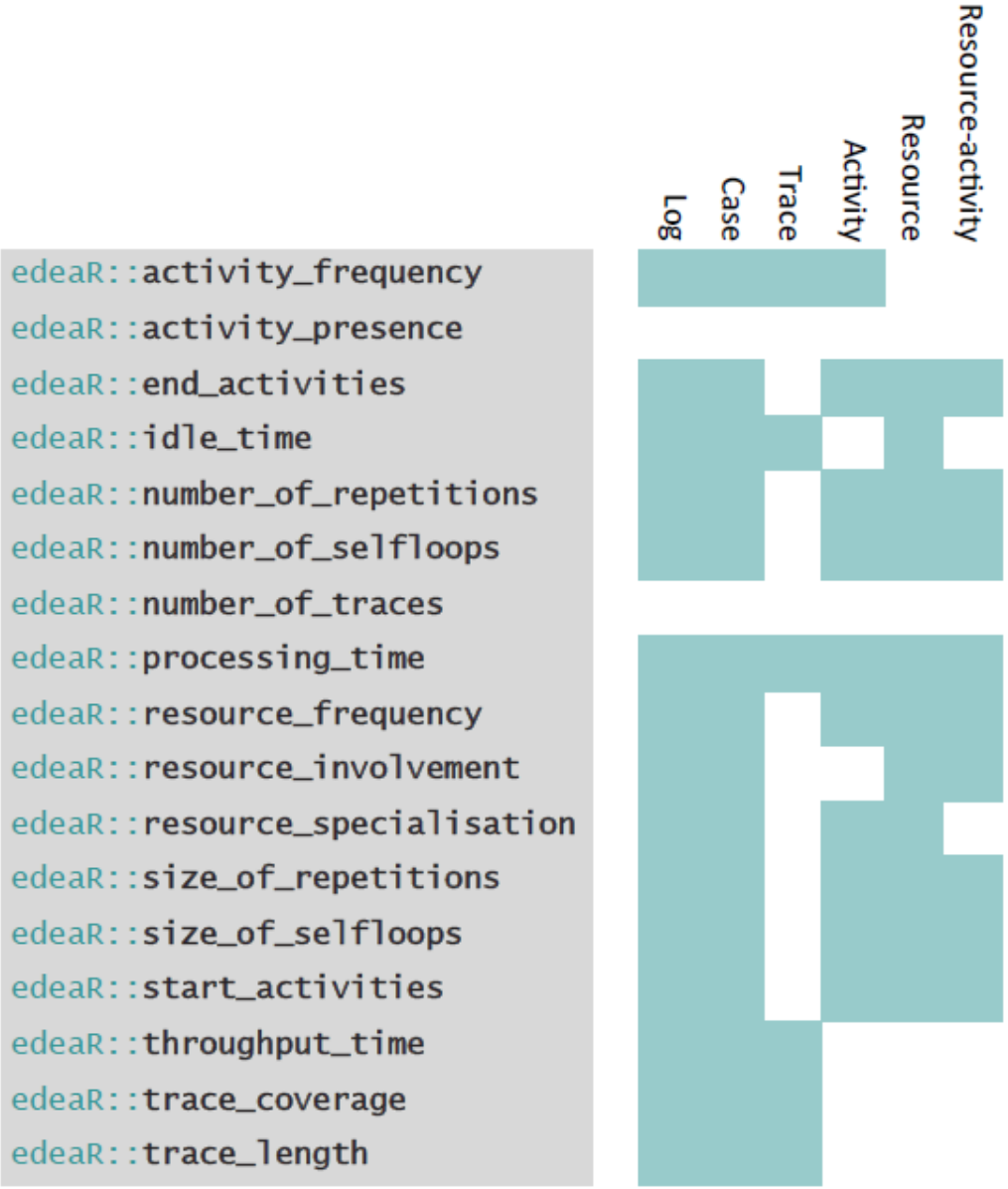
1. Go to **www.socrative.com**
2. Click "**Student Login**"
3. Enter room name "**BUPAR**"

#4 Advanced

Constructing Event logs
Exploratory and Descriptive Metrics
Advanced Data Wrangling
Enriching

Exploratory and Descriptive Metrics

Metrics



Each metric has different levels of granularity

- Log
- Case
- Trace
- Activity
- Resource
- Resource-activity

Trace length

```
sepsis %>% trace_length()
```

```
    min       q1   median     mean       q3      max   st_dev      iqr
  3.000    9.000   13.000   14.490   16.000  185.000   11.476    7.000
```

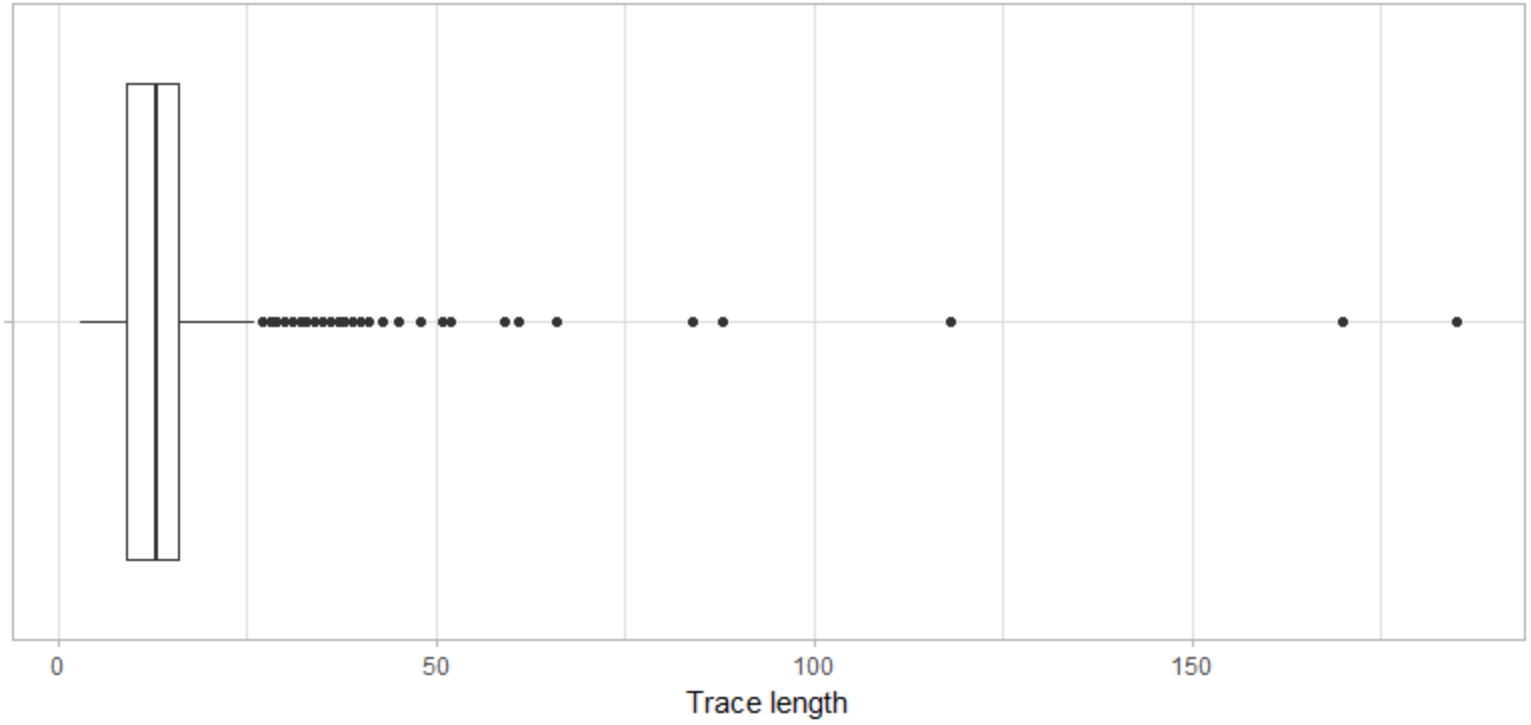
Trace length

```
sepsis %>% trace_length("case")
```

```
# A tibble: 1,050 x 2
   Case_ID trace_length
     <chr>        <int>
 1       A           22
 2      AA            8
 3     AAA           11
 4      AB            8
 5     ABA           17
 6      AC           13
 7     ACA            8
 8      AD           29
 9     ADA           24
10      AE           18
# ... with 1,040 more rows
```

Trace length

```
sepsis %>% trace_length() %>% plot
```



Processing time

```
patients %>% processing_time(units = “hours”)
```

```
   min     q1 median   mean     q3    max st_dev    iqr
10.718 24.950 27.727 27.749 30.737 38.246  4.175  5.787
```

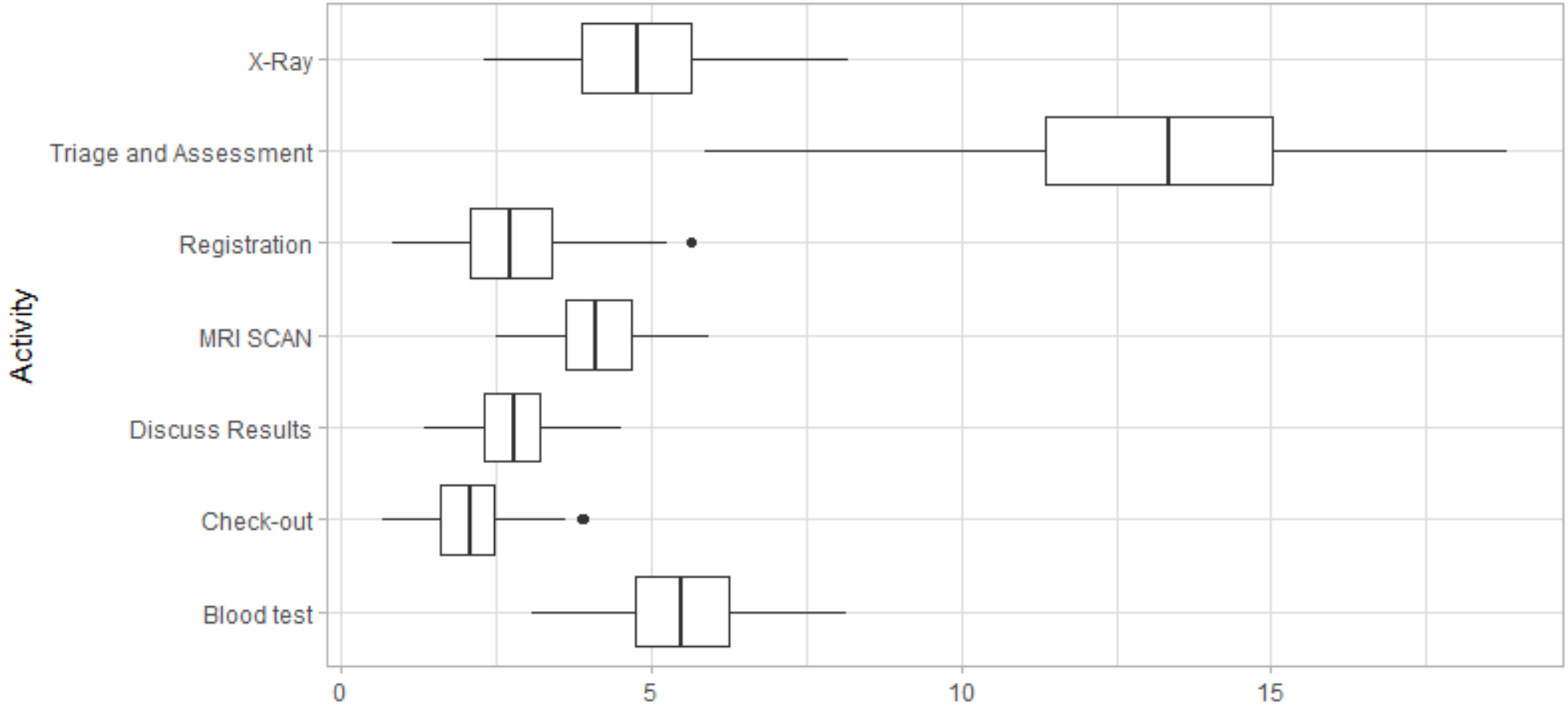
Processing time

```
patients %>% processing_time(level = “activity”, units = “hours”)
```

```
# A tibble: 7 x 11
               handling relative_frequency   min     q1 median   mean     q3    max st_dev   iqr      tot
                  <chr>              <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <dbl>    <dbl>
1          Registration              0.184 0.829  2.070  2.714  2.754  3.402  5.635  0.954 1.332 1376.900
2 Triage and Assessment              0.184 5.868 11.352 13.341 13.105 15.031 18.802  2.759 3.680 6552.749
3       Discuss Results              0.182 1.334  2.314  2.772  2.776  3.220  4.536  0.628 0.906 1374.001
4             Check-out              0.181 0.668  1.612  2.072  2.063  2.472  3.896  0.620 0.860 1014.944
5                 X-Ray              0.096 2.294  3.886  4.792  4.846  5.649  8.169  1.281 1.763 1264.815
6            Blood test              0.087 3.089  4.756  5.468  5.536  6.267  8.138  1.060 1.511 1311.979
7              MRI SCAN              0.087 2.489  3.607  4.090  4.150  4.697  5.924  0.735 1.090  979.346
```

Processing time

```
patients %>% processing_time(level = “activity”) %>% plot
```



Advanced Data Wrangling

And Enriching

dplyr verbs

`filter`	Filter event log
`mutate`	Add new column to event log
`select`	Select columns **but always retains mapping**
`arrange`	Arrange event log
`group_by`	Create `grouped_eventlog`
`group_by_case, …`	`Shortcuts for grouping`

Easy generic filtering and sampling

`slice`	Take a slice of cases
`slice_activities`	Take a slice of activity instances
`slice_events`	Take a slice of events
`first_n, last_n`	Take the first/last n activity instances
`sample_n`	Take a sample of n cases

Changing the view on the go

`set_case_id`	Change case identifier
`set_activity_id`	Change activity identifier
`re_map`	Reuse a mapping saved earlier

Enrich with Calculated variables

```
# Was a refund made for a case?
> log_claims %>%
      group_by_case %>%
      mutate(refund_made = any(str_detect(activity, “Pay Claim”)))
# What is the total cost, by summing the cost of activities
> log_claims %>%
      group_by_case %>%
      mutate(total_cost = sum(cost)) -> log_claims
# What is the cost range?
> log_claims %>%
      mutate(impact = case_when(total_cost > 100000 ~ “High”,
                                total_cost > 25000 ~ “Medium”,
                                TRUE ~ “Low”))
```

Enrich with Descriptive Metrics

```
# Add the throughput time of each case as a case attribute
> log_claims %>%
      throughput_time(level = "case", append = TRUE)
```

```
Event log consisting of:
35499 events
21 traces
2447 cases
15 activities
30593 activity instances

# A tibble: 35,499 x 11
   claim_id activity_instance          activity status       date    resource result  type value agent throughput_time_case
      <chr>             <chr>            <fctr> <fctr>     <date>      <fctr>  <chr> <chr> <int> <chr>                <dbl>
 1        1             10002    Check Contract  start 2008-01-12 Assistant 1     OK  <NA>    NA  <NA>                  127
 2        1             10011 Pay Back Decision  start 2008-03-22   Manager 2   <NA>  <NA>    NA  <NA>                  127
 3        2             10015    Check Contract  start 2008-01-13 Assistant 6     OK  <NA>    NA  <NA>                  127
 4        2             10024 Pay Back Decision  start 2008-03-23   Manager 2   <NA>  <NA>    NA  <NA>                  127
 5        3             10028    Check Contract  start 2008-01-14 Assistant 7     OK  <NA>    NA  <NA>                   22
 6        4             10035    Check Contract  start 2008-01-15 Assistant 7     OK  <NA>    NA  <NA>                  127
 7        4             10044 Pay Back Decision  start 2008-03-25   Manager 2   <NA>  <NA>    NA  <NA>                  127
 8        5             10048    Check Contract  start 2008-01-16   Manager 3     OK  <NA>    NA  <NA>                  127
 9        5             10057 Pay Back Decision  start 2008-03-26   Manager 3   <NA>  <NA>    NA  <NA>                  127
10        6             10061    Check Contract  start 2008-01-17   Manager 3     OK  <NA>    NA  <NA>                  127
# ... with 35,489 more rows
```

Enrich with Descriptive Metrics

```
# Add the throughput time of each case as a case attribute
> sepsis %>%
      throughput_time(level = “case”, units = “days”, append = TRUE) %>%

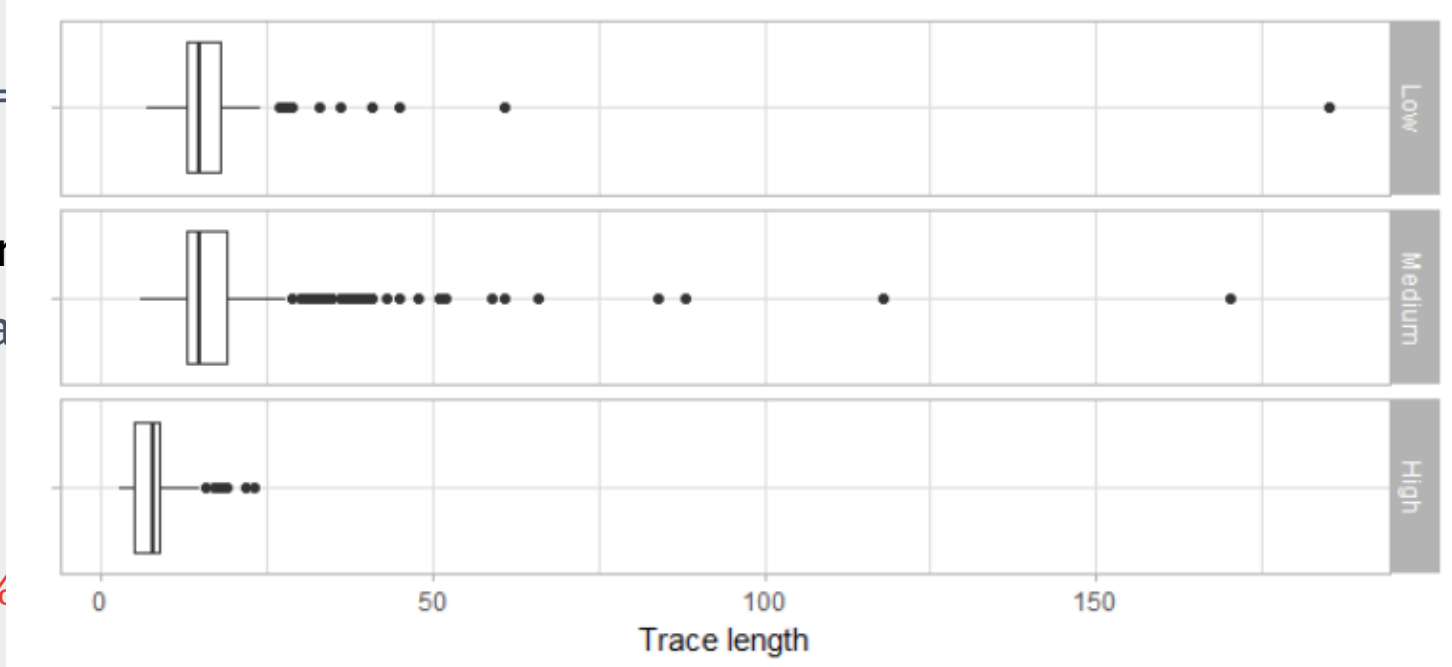

# Create a label for performance
      mutate(performance = case_when(throughput_time_case > 100 ~ “Low”,
                                     throughput_time_case > 25 ~ “Medium”,
                                     TRUE ~ “High”))
```

Enrich with Descriptive Metrics

```
# Add the throughput time of each case as a case attribute
> sepsis %>%
      throughput_time(level =

# Create a label for performan
      mutate(performance = ca

# Use the new variable
      group_by(performance) %
      trace_length() %>%
      plot()
```



Constructing Event Logs

Creating an `eventlog`

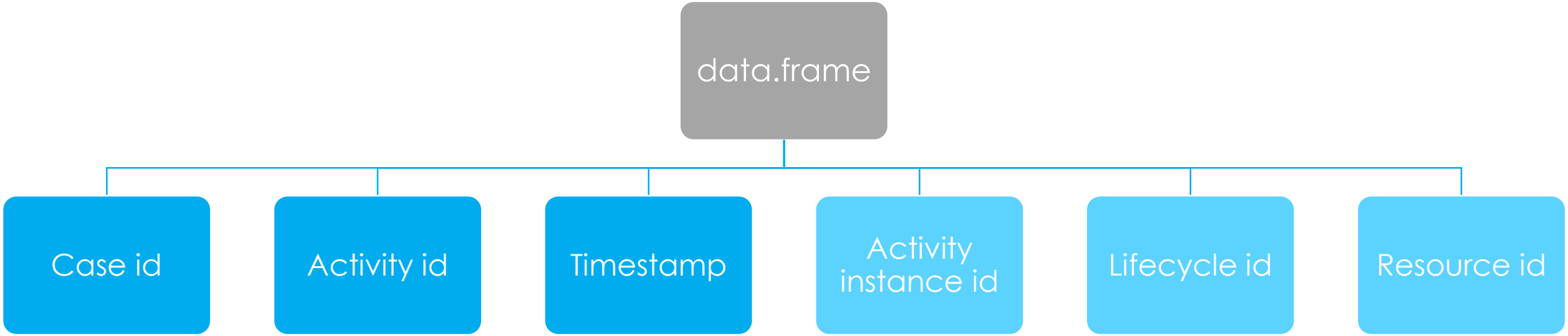
1. Get the data in the right format
2. Define a mapping

Get the data in the right format

A data.frame where each row is a different event

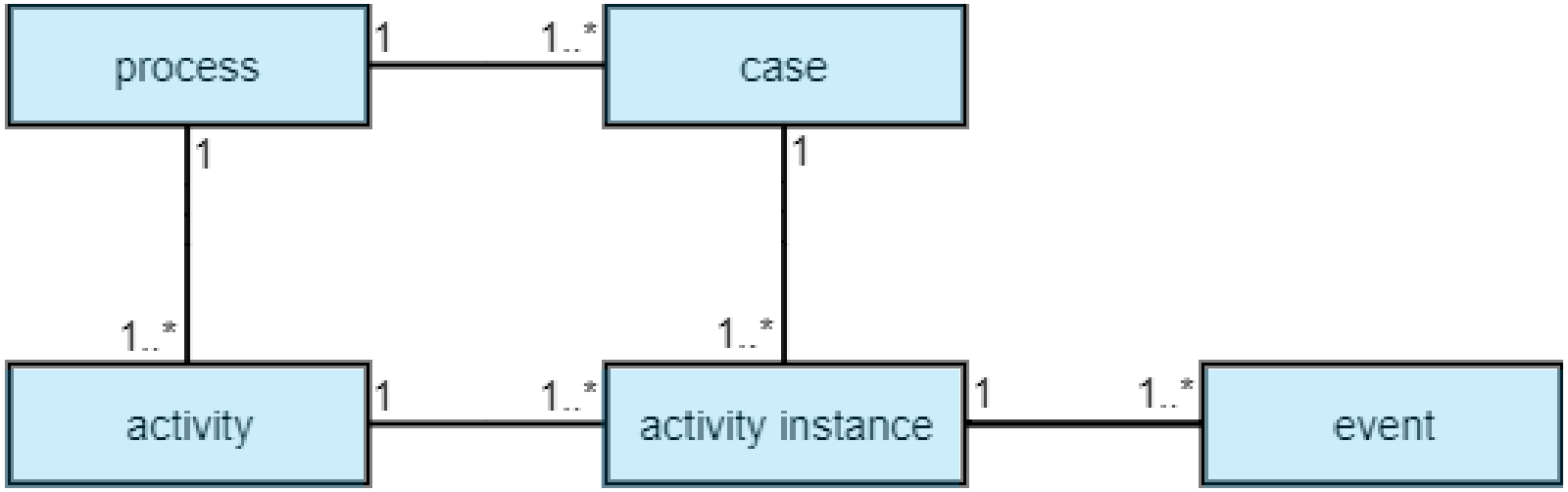
Claim id	Activity instance	Activity	Status	Date	Resource	Result	Type	Value	Agent
1	10001	File Claim	complete	2008-01-09	Client x	NA	Car	600	Agent A
1	10002	Check Contract	start	2008-01-12	Assistant 1	OK	NA	NA	NA
1	10002	Check Contract	complete	2008-01-13	Assistant 1	OK	NA	NA	NA
1	10003	Franchise?	complete	2008-01-13	Assistant 1	By client	NA	NA	NA
1	10004	Covered?	complete	2008-01-15	Assistant 1	OK	NA	NA	NA
1	10005	Acceptance Decision	complete	2008-01-20	Manager 2	NA	NA	NA	NA
1	10006	Start Investigation	complete	2008-02-01	Assistant 1	NA	NA	NA	NA
1	10011	Pay Back Decision	start	2008-03-22	Manager 2	NA	NA	NA	NA
1	10011	Pay Back Decision	complete	2008-03-23	Manager 2	NA	NA	NA	NA

Define a mapping



Which column describes which aspect of the events?

Data Model


process
case
activity
activity instance
event
1
1..*
1
1
1..*
1..*
1
1..*
1
1..*

Define a mapping

Cases	Activities								
Claim id	Activity instance	Activity	Status	Date	Resource	Result	Type	Value	Agent
1	10001	File Claim	complete	2008-01-09	Client x	NA	Car	600	Agent A
1	10002	Check Contract	start	2008-01-12	Assistant 1	OK	NA	NA	NA
1	10002	Check Contract	complete	2008-01-13	Assistant 1	OK	NA	NA	NA
1	10003	Franchise?	complete	2008-01-13	Assistant 1	By client	NA	NA	NA
1	10004	Covered?	complete	2008-01-15	Assistant 1	OK	NA	NA	NA
1	10005	Acceptance Decision	complete	2008-01-20	Manager 2	NA	NA	NA	NA
1	10006	Start Investigation	complete	2008-02-01	Assistant 1	NA	NA	NA	NA
1	10011	Pay Back Decision	start	2008-03-22	Manager 2	NA	NA	NA	NA
1	10011	Pay Back Decision	complete	2008-03-23	Manager 2	NA	NA	NA	NA

Creating an eventlog

```
claims %>%
      eventlog(case_id = “Claim id”,
               activity_id = “Activity”,
               activity_instance_id = “Activity Instance”,
               timestamp = “Date”,
               lifecycle_id = “Status”,
               resource_id = “Resource”) -> log_claims
```

But

"I don't have resources, lifecycle information and activity instances,

just **cases, activities and time.**"

Try ‘simple_eventlog’

Claim id	Activity	Date
1	File Claim	2008-01-09
1	Check Contract	2008-01-12
1	Franchise?	2008-01-13
1	Covered?	2008-01-15
1	Acceptance Decision	2008-01-20
1	Start Investigation	2008-02-01
1	Pay Back Decision	2008-03-22

The result of `simple_eventlog` will behave the same as a normal event log. It is just a wrapper that transforms the data with artificial resources, lifecycle statuses and activity instances.

You will be able to use all the functions, though not everything will be insightfull. E.g. processing time will be zero.

```
claims %>%
       simple_eventlog(case_id = “Claim id”,
                       activity_id = “Activity”,
                       timestamp = “Date”) -> log_claims
```

But

"My data has **multiple events on one row**, with different timestamps."

Try ‘activities_to_eventlog’

claim_id	activity	resource	result	type	value	agent	start	complete
1	Check Contract	Assistant 1	OK	NA	NA	NA	2008-01-12	2008-01-13
1	Pay Back Decision	Manager 2	NA	NA	NA	NA	2008-03-22	2008-03-23
1	Accident	Client x	NA	NA	NA	NA	NA	2008-01-01
1	File Claim	Client x	NA	Car	600	Agent A	NA	2008-01-09
1	Franchise?	Assistant 1	By client	NA	NA	NA	NA	2008-01-13
1	Covered?	Assistant 1	OK	NA	NA	NA	NA	2008-01-15
1	Acceptance Decision	Manager 2	NA	NA	NA	NA	NA	2008-01-20
1	Start Investigation	Assistant 1	NA	NA	NA	NA	NA	2008-02-01
1	Appoint Lawyer	Manager 2	NA	NA	NA	NA	NA	2008-02-10
1	Appoint Expert	Manager 2	NA	NA	NA	NA	NA	2008-02-10

```
claims %>%
      activities_to_eventlog(case_id      = “Claim id”,
                             activity_id  = “Activity”,
                             timestamps   = c(“start”,”complete”)) -> log_claims
```

But

"I have yet another problem with my data."

Try learning R

Data Import : : CHEAT SHEET

R's **tidyverse** is built around **tidy data** stored in **tibbles**, which are enhanced data frames.

The front side of this sheet shows how to read text files into R with **readr**.

The reverse side shows how to create tibbles with **tibble** and to layout tidy data with **tidyr**.

OTHER TYPES OF DATA

Try one of the following packages to import other types of files

- **haven** - SPSS, Stata, and SAS files
- **readxl** - excel files (.xls and .xlsx)
- **DBI** - databases
- **jsonlite** - json
- **xml2** - XML
- **httr** - Web APIs
- **rvest** - HTML (Web Scraping)

Save Data

Save **x**, an R object, to **path**, a file path, as:

Comma delimited file
write_csv(x, path, na = "NA", append = FALSE, col_names = !append)

File with arbitrary delimiter
write_delim(x, path, delim = " ", na = "NA", append = FALSE, col_names = !append)

CSV for excel
write_excel_csv(x, path, na = "NA", append = FALSE, col_names = !append)

String to file
write_file(x, path, append = FALSE)

String vector to file, one element per line
write_lines(x,path, na = "NA", append = FALSE)

Object to RDS file
write_rds(x, path, compress = c("none", "gz", "bz2", "xz"), ...)

Tab delimited files
write_tsv(x, path, na = "NA", append = FALSE, col_names = !append)

Read Tabular Data - These functions share the common arguments:

read_*(file, col_names = TRUE, col_types = NULL, locale = default_locale(), na = c("", "NA"), quoted_na = TRUE, comment = "", trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(1000, n_max), progress = interactive()**)**

Comma Delimited Files
read_csv("file.csv")
To make file.csv run:
write_file(x = "a,b,c\n1,2,3\n4,5,NA", path = "file.csv")

Semi-colon Delimited Files
read_csv2("file2.csv")
write_file(x = "a;b;c\n1;2;3\n4;5;NA", path = "file2.csv")

Files with Any Delimiter
read_delim("file.txt", delim = "\|")
write_file(x = "a\|b\|c\n1\|2\|3\n4\|5\|NA", path = "file.txt")

Fixed Width Files
read_fwf("file.fwf", col_positions = c(1, 3, 5))
write_file(x = "a b c\n1 2 3\n4 5 NA", path = "file.fwf")

Tab Delimited Files
read_tsv("file.tsv") Also **read_table()**.
write_file(x = "a\tb\tc\n1\t2\t3\n4\t5\tNA", path = "file.tsv")

USEFUL ARGUMENTS

Example file
write_file("a,b,c\n1,2,3\n4,5,NA","file.csv")
f <- "file.csv"

No header
read_csv(f, **col_names = FALSE**)

Provide header
read_csv(f, **col_names = c("x", "y", "z")**)

Skip lines
read_csv(f, **skip = 1**)

Read in a subset
read_csv(f, **n_max = 1**)

Missing Values
read_csv(f, **na = c("1", ".")**)

Read Non-Tabular Data

Read a file into a single string
read_file(file, locale = default_locale())

Read each line into its own string
read_lines(file, skip = 0, n_max = -1L, na = character(), locale = default_locale(), progress = interactive())

Read Apache style log files
read_log(file, col_names = FALSE, col_types = NULL, skip = 0, n_max = -1, progress = interactive())

Read a file into a raw vector
read_file_raw(file)

Read each line into a raw vector
read_lines_raw(file, skip = 0, n_max = -1L, progress = interactive())

Data types

readr functions guess the types of each column and convert types when appropriate (but will NOT convert strings to factors automatically).

A message shows the type of each column in the result.

```
## Parsed with column specification:
## cols(
##   age = col_integer(),
##   sex = col_character(),
##   earn = col_double()
## )
```

age is an integer; sex is a character; earn is a double (numeric)

1. Use **problems()** to diagnose problems
x <- read_csv("file.csv"); problems(x)

2. Use a col_ function to guide parsing
- **col_guess()** - the default
- **col_character()**
- **col_double()**, **col_euro_double()**
- **col_datetime**(format = "") Also **col_date**(format = ""), **col_time**(format = "")
- **col_factor**(levels, ordered = FALSE)
- **col_integer()**
- **col_logical()**
- **col_number()**, **col_numeric()**
- **col_skip()**

x <- read_csv("file.csv", col_types = cols(
A = col_double(),
B = col_logical(),
C = col_factor()))

3. Else, read in as character vectors then parse with a parse_ function.
- **parse_guess()**
- **parse_character()**
- **parse_datetime()** Also **parse_date()** and **parse_time()**
- **parse_double()**
- **parse_factor()**
- **parse_integer()**
- **parse_logical()**
- **parse_number()**

x$A <- parse_number(x$A)

Data Transformation with dplyr : : CHEAT SHEET

dplyr functions work with pipes and expect **tidy data**. In tidy data:

Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own **row**

pipes
x %>% f(y) becomes **f(x, y)**

Summarise Cases

These apply **summary functions** to columns to create a new table. Summary functions take vectors as input and return one value (see back).

summary function

summarise(.data, ...)
Compute table of summaries. Also **summarise_()**.
summarise(mtcars, avg = mean(mpg))

count(x, ..., wt = NULL, sort = FALSE)
Count number of rows in each group defined by the variables in ... Also **tally()**.
count(iris, Species)

VARIATIONS

summarise_all() - Apply funs to every column.
summarise_at() - Apply funs to specific columns.
summarise_if() - Apply funs to all cols of one type.

Group Cases

Use **group_by()** to created a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))

group_by(.data, ..., add = FALSE)
Returns copy of table grouped by ...
g_iris <- group_by(iris, Species)

ungroup(x, ...)
Returns ungrouped copy of table.
ungroup(g_iris)

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table. Use a variant that ends in _ for non-standard evaluation friendly code.

filter(.data, ...) Extract rows that meet logical criteria. Also **filter_()**. *filter(iris, Sepal.Length > 7)*

distinct(.data, ..., .keep_all = FALSE) Remove rows with duplicate values. Also **distinct_()**.
distinct(iris, Species)

sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select fraction of rows.
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select size rows. *sample_n(iris, 10, replace = TRUE)*

slice(.data, ...) Select rows by position. Also **slice_()**. *slice(iris, 10:15)*

top_n(x, n, wt) Select and order top n entries (by group if grouped data). *top_n(iris, 5, Sepal.Width)*

Logical and boolean operators to use with filter()

<	<=	is.na()	%in%	|	xor()
>	>=	!is.na()	!	&	

See **?base::logic** and **?Comparison** for help.

ARRANGE CASES

arrange(.data, ...)
Order rows by values of a column (low to high), use with **desc()** to order from high to low.
arrange(mtcars, mpg)
arrange(mtcars, desc(mpg))

ADD CASES

add_row(.data, ..., .before = NULL, .after = NULL**)**
Add one or more rows to a table.
add_row(faithful, eruptions = 1, waiting = 1)

Column functions return a set of columns as a new table. Use a variant that ends in _ for non-standard evaluation friendly code.

select(.data, ...)
Extract columns by name. Also **select_if()**
select(iris, Sepal.Length, Species)

Use these helpers with select (),
e.g. select(iris, starts_with("Sepal"))

contains(match) **num_range**(prefix, range) **:**, e.g. mpg:cyl
ends_with(match) **one_of**(...) **-**, e.g. -Species
matches(match) **starts_with**(match)

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

vectorized function

mutate(.data, ...)
Compute new column(s).
mutate(mtcars, gpm = 1/mpg)

transmute(.data, ...)
Compute new column(s), drop others.
transmute(mtcars, gpm = 1/mpg)

mutate_all(.tbl, .funs, ...**)** Apply funs to every column. Use with **funs()**.
mutate_all(faithful, funs(log(.), log2(.)))

mutate_at(.tbl, .cols, .funs, ...**)** Apply funs to specific columns. Use with **funs()**, **vars()** and the helper functions for select().
mutate_at(iris, vars(-Species), funs(log(.)))

mutate_if(.tbl, .predicate, .funs, ...**)**
Apply funs to all columns of one type. Use with **funs()**.
mutate_if(iris, is.numeric, funs(log(.)))

add_column(.data, ..., .before = NULL, .after = NULL**)** Add new column(s).
add_column(mtcars, new = 1:32)

rename(.data, ...) Rename columns.
rename(iris, Length = Sepal.Length)

- Import
- Metrics
- Visuals
- Granularity levels
- Aggregations
- Filters
- Calculated variables
- Groupings
- …

Combining the building blocks

```
claims_eventlog %>%
    filter_activity_frequency(percentage = 0.9) %>%
    group_by_case() %>%
    mutate(refund = any(str_detect(activity, "Refund made"))) %>%
    group_by(refund) %>%
    throughput_time() %>%
    plot()
```

“Standing on the shoulder of giants”

+ R

What makes bupaR unique?

Flexibility

- Combining different functionalities
- Easy to work with custom data attributes
- Granularity levels
- Endless configurations
- …

What makes bupaR unique?

Extensibility

gertjanssenswillen / bupaR

Unwatch 4 | Star 2 | Fork 3

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

Branch: master | bupaR / R / | Create new file | Upload files | Find file | History

gertjanssenswillen Export activity to eventlogé — Latest commit b3db818 14 days ago

..		
act_collapse.R	Code cleanup	2 months ago
act_recode.R	Code cleanup	2 months ago
act_unite.R	Code cleanup	2 months ago
activities.R	Add label functions	2 months ago
activities_to_eventlog.R	Export activity to eventlogé	14 days ago
activity_id.R	Add label functions	2 months ago
activity_instance_id.R	v0.3.1.9000	2 months ago
arrange.eventlog.R	v0.3.1.9000	2 months ago
bupar.R	v0.3.1.9000	2 months ago
case_id.r	v0.3.1.9000	2 months ago
case_list.R	Export case_list	2 months ago
cases.r	Add label functions	2 months ago
durations.R	v0.3.1.9000	2 months ago
eventlog.R	Remove resource restriction	2 months ago
filter.R	Change onAttach	a month ago

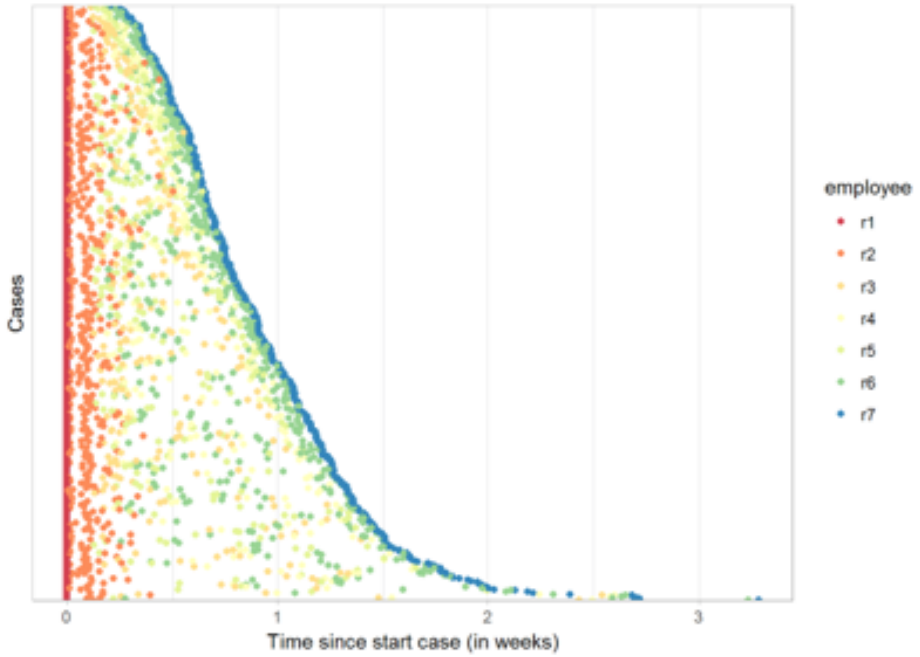
What makes bupaR unique?

Scripts

```
claims_eventlog %>%
   filter_activity_frequency(percentage = 0.9) %>%
   group_by_case() %>%
   mutate(refund = any(str_detect(activity, "Refun
   group_by(refund) %>%
   throughput_time() %>%
   plot()
```

RMarkdown

```
patients %>%
    dotted_chart(x = "relative", y = "duration", color = "employee")
```



Reproducibility

bupaR

I want more of this

bupar.net

Introduction to R

Tidyverse

dplyr

ggplot2

tidyr

Data Import

Process model support

Petri Nets

BPMN

Process Trees

Process discovery

Discovery algorithms

Build Statistical Models

Conformance checking

Rule-based conformance checking

Model-based conformance checking

Scalability

Convert back-end to C++

dbupaR: database version of bupaR

I want to help

Thank you!

Gert Janssenswillen
gert.janssenswillen@uhasselt.be
FWO PhD Fellow
Research Group Business Informatics

T +32(0)11 26 86 39

www.uhasselt.be - www.businessinformatics.be
Universiteit Hasselt - Campus Diepenbeek
Agoralaan Gebouw D- B-3590 Diepenbeek