Covid-19 isolation and quarantine orders in a district of Berlin, Germany How many, how long, to whom and predictive factors

Jakob Schumacher, Lisa Kühne, Sophie Bruessermann, Benjamin Geisler, Sonja Jäckle

20. Juni 2022

```
# ![]("https://www.horizont.net/news/media/32/Das-neue-Berlin-Logo-315209.jpeg"){width=30%}
# ![]("https://www.ultrasoundsymposium.org/wp-content/uploads/2017/08/fhg-1.gif"){width=30%}
# ![](https://www.lsc-digital-public-health.de/images/partners/leibniz-institut-bips.png){width=30%}


##############################################
# Setting options for knitr, ggplot, fonts
##############################################

# Disabeling scientific notation
options(scipen = 999)

# Create correct figure caption
knitr::opts_knit$set(eval.after = 'fig.cap')

# Linebreaks within code
knitr::opts_chunk$set(tidy = TRUE)

# Adjust the big mark for large numbers
knitr::knit_hooks$set(inline = function(x) { prettyNum(x, big.mark=" ") })
```

About this Repository

The following R-Script calculates all the necessary numbers and figures for a publication. All necessary files to reproduce are available. The analysis is done in R. This project uses Renv. See the file .Rprofile for used packages. This script runs with the package target. The important parts of the script lie in the functions in the code folder. You can check the file _targets.R to see the different steps in their sequential order.

Code

```
create_pairslist <- function(maximumnumber = 84){
  # This function creates a list of numbers which is needed for the other functions
  overlapcheck_pairs <- function(highest = 10){
    mytibble <- tibble(value = 1:highest) %>%
      expand(value, value1 = value) %>%
      filter(value < value1)
    mylist <- map(1:nrow(mytibble), ~c(mytibble$value[.x], mytibble$value1[.x]))
    mylist
```

```
  }

  pairslist <- map(1:maximumnumber, ~overlapcheck_pairs(.x)) # Legt die Pairsliste an
  pairslist
}


#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# De duplication
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# This function checks wheter two entries overlap
overlapcheck <- function(data_input, pair) {
  i <- pair[1]
  j <- pair[2]
  first <- c(data_input$AbsonderungVon[i], data_input$AbsonderungBis[i])
  second <- c(data_input$AbsonderungVon[j], data_input$AbsonderungBis[j])
  if(first %overlaps% second) {
    data_input$AbsonderungVon[i] <- min(c(first, second))
    data_input$AbsonderungBis[i] <- max(c(first, second))
    data_input <- data_input[-c(j),]
  }
  data_input
}


# This function uses the function overlapcheck on the table of one person and gives the result with only
overlapcheck_concise <- function(data_input, testsubject, pair = pairslist) {
  pairslist <- pair
  tdf <- data_input %>% filter(AnonID == testsubject)
  mylist <- pairslist[[nrow(tdf)]]
  allvalues <- map(mylist, ~overlapcheck(tdf, .x))
  table <- bind_rows(allvalues)
  table %>%
    count(rowid) %>%
    filter(n==length(mylist)) %>%
    select(-n) %>%
    left_join(table, by = "rowid") %>%
    distinct() %>%
    group_by(rowid) %>%
    mutate(AbsonderungVon = min(AbsonderungVon)) %>%
    mutate(AbsonderungBis = max(AbsonderungBis)) %>%
    distinct()
}


de_duplication <- function(df, methodslist = methodslist) {
  # To save calculation time the complete dataset is split up. I am sure there is an easier way but this
  kps <- df %>% filter(DatensatzKategorie == "Kontakt-COVID-19")
  faelle <- df %>% filter(DatensatzKategorie == "COVID-19")
  einzelne_kps <- kps %>% count(AnonID) %>% filter(n==1) %>% pull(AnonID)
  doppelte_kps <- kps %>% count(AnonID) %>% filter(n>1) %>% pull(AnonID)
  einzelne_faelle <- faelle %>% count(AnonID) %>% filter(n==1) %>% pull(AnonID)
  doppelte_faelle <- faelle %>% count(AnonID) %>% filter(n>1) %>% pull(AnonID)
```

```
  einzelne_kps_df <- kps %>% filter(AnonID %in% einzelne_kps)
  doppelte_kps_df <- kps %>% filter(AnonID %in% doppelte_kps)
  einzelne_faelle_df <- faelle %>% filter(AnonID %in% einzelne_faelle)
  doppelte_faelle_df <- faelle %>% filter(AnonID %in% doppelte_faelle)

  # Create pairslist
  pairslist <- create_pairslist()

  # These functions do the actual work of adjusting the overlapping periods
  doppelte_faelle_df_bereinigt <- bind_rows(map(doppelte_faelle, ~overlapcheck_concise(data_input = dopp
  doppelte_kps_df_bereinigt <- bind_rows(map(doppelte_kps, ~overlapcheck_concise(data_input = doppelte_

  df <- bind_rows(doppelte_faelle_df_bereinigt, doppelte_kps_df_bereinigt, einzelne_faelle_df, einzelne_

}

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# Adjust_overlap
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


# The following variantes are possible
# 1
# KP   |----|
#> Case |----|
#> Filter min(kp)>=min(case)
# result should be:  delete(KP)
# 2
# KP      |----|
#> Case |----|
#> Filter min(kp)>=min(case)
# result should be: delete(kp)
# 4
# KP    |--|
#> Case |----|
#> Filter min(kp)>=min(case)
# result should be:  delete(KP)
# 6
# KP   |--|
#> Case |----|
#> Filter min(kp)>=min(case)
# result should be: delete(KP)
# KP   |----|
#> Case |--|
#> Filter min(kp)>=min(case)
# result should be: delete(kp)
# 8
# KP    |---|
#> Case |----|
#> Filter min(kp)>=min(case)
# result should be: delete(kp)
# 3
# KP   |----|
```

```
#> Case    |----|
#> Filter min(kp)<min(case)
# result should be:  min(KP) - min(Case)
# 5
# KP   |----|
#> Case  |--|
#> Filter min(kp)<min(case)
# result should be: min(KP) - min(case)
# 9
# KP   |----|
#> Case  |---|
#> Filter min(kp)<min(case)
# result should be: min(kp) - min(case)



# This function checks for overlap and adjusts the quarantine if needed
adjustoverlapquarantine <- function(data_input, pair) {
  i <- pair[1]
  j <- pair[2]
  first <- c(data_input$AbsonderungVon[i], data_input$AbsonderungBis[i])
  second <- c(data_input$AbsonderungVon[j], data_input$AbsonderungBis[j])
  firstsequence <- seq(data_input$AbsonderungVon[i], data_input$AbsonderungBis[i], by = 1)
  secondsequence <- seq(data_input$AbsonderungVon[j], data_input$AbsonderungBis[j], by = 1)
  if(first %overlaps% second) {
    if(data_input$DatensatzKategorie[i] == "COVID-19" & data_input$DatensatzKategorie[j] == "Kontakt-COV
      if(min(secondsequence)<min(firstsequence)){
        data_input$AbsonderungVon[j] <- min(secondsequence)
        data_input$AbsonderungBis[j] <- min(firstsequence)
        data_input$overlapadjusted[j] <- "adjusted"
      } else {data_input <- data_input[-c(j),]}

    }  else if(data_input$DatensatzKategorie[i] == "Kontakt-COVID-19" & data_input$DatensatzKategorie[j]
      if(min(firstsequence)<min(secondsequence)){
        data_input$AbsonderungVon[i] <- min(firstsequence)
        data_input$AbsonderungBis[i] <- min(secondsequence)
        data_input$overlapadjusted[i] <- "adjusted"
      } else {data_input <- data_input[-c(i),]}
    }
  }
  data_input
}

# This function applies the adjustoverlapquarantine to every testsubject
adjustoverlapquarantine_concise <- function(data_input, testsubject, pairslist = pairslist) {
  tdf <- data_input %>% filter(AnonID == testsubject)
  mylist <- pairslist[[nrow(tdf)]]
  allvalues <- map(mylist, ~adjustoverlapquarantine(tdf, .x))
  table <- bind_rows(allvalues)
  changedrowids <- table %>% filter(overlapadjusted == "adjusted") %>% distinct()
  notchangedrowids <- table %>% filter(!rowid %in% changedrowids$rowid) %>% select(rowid) %>% distinct()
  data_output <- bind_rows(changedrowids, notchangedrowids)
  data_output
```

```r
}

adjust_overlap <- function(df_deduplicated){
  df <- df_deduplicated
  # Create empty value
  df$overlapadjusted <- NA
  # To save calculation time the complete dataset is split up. I am sure there is an easier way but thi
  einzelne_anonIDs <- df %>% count(AnonID) %>% filter(n==1) %>% pull(AnonID)
  doppelte_anonIDs <- df %>% count(AnonID) %>% filter(n>1) %>% pull(AnonID)
  einzelne_anonIDs_df <- df %>% filter(AnonID %in% einzelne_anonIDs)
  doppelte_anonIDs_df <- df %>% filter(AnonID %in% doppelte_anonIDs)
  # Create pairslist
  pairslist <- create_pairslist()
  doppelte_anonIDs_df_bereinigt <- bind_rows(map(doppelte_anonIDs, ~adjustoverlapquarantine_concise(data
                                                                                                    test
                                                                                                    pai

  # Saving for the publication
  df <- bind_rows(doppelte_anonIDs_df_bereinigt, einzelne_anonIDs_df) %>% ungroup()

  df
}


#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# Adjoining
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

adjoincheck <- function(data_input, pair) {
  i <- pair[1] # this is number one
  j <- pair[2] # this is number two
  # what counts as adjoining
  adjoiningwhentimedifference <- seq(0,7)
  twoafterone <- data_input$AbsonderungVon[j] - data_input$AbsonderungBis[i]  # if one after two its po
  oneaftertwo <- data_input$AbsonderungVon[i] - data_input$AbsonderungBis[j] # if two after one its pos
  if(data_input$DatensatzKategorie[i] == "COVID-19" & data_input$DatensatzKategorie[j] == "Kontakt-COVI
    if(oneaftertwo %in% adjoiningwhentimedifference){
      data_input$adjoiningQandI[i] <- oneaftertwo
      data_input$adjoiningQandI[j] <- oneaftertwo
    }
  }  else if(data_input$DatensatzKategorie[i] == "Kontakt-COVID-19" & data_input$DatensatzKategorie[j] =
    if(twoafterone %in% adjoiningwhentimedifference){
      data_input$adjoiningQandI[i] <- twoafterone
      data_input$adjoiningQandI[j] <- twoafterone
    }
  }
  data_input
}

adjoincheck_concise <- function(data_input, testsubject, pair = pairslist) {

  # Get pairslist
  pairslist <- pair
```

```
  tdf <- data_input %>% filter(AnonID == testsubject)
  mylist <- pairslist[[nrow(tdf)]]
  allvalues <- map(mylist, ~adjoincheck(tdf, .x))
  table <- bind_rows(allvalues)
  changedrowids <- table %>% filter(!is.na(adjoiningQandI)) %>% distinct()
  notchangedrowids <- table %>% filter(!rowid %in% changedrowids$rowid) %>% select(rowid) %>% distinct()
  data_output <- bind_rows(changedrowids, notchangedrowids)
  data_output
}

find_adjoin <- function(df_overlapped) {

  # Get the dataframe
  df <- df_overlapped

  # Set empty value
  df$adjoiningQandI <- NA

  # Split up the df to save computing time
  einzelne_anonIDs <- df %>% count(AnonID) %>% filter(n==1) %>% pull(AnonID)
  doppelte_anonIDs <- df %>% count(AnonID) %>% filter(n>1) %>% pull(AnonID)
  einzelne_anonIDs_df <- df %>% filter(AnonID %in% einzelne_anonIDs)
  doppelte_anonIDs_df <- df %>% filter(AnonID %in% doppelte_anonIDs)

  # Create pairslist
  pairslist <- create_pairslist()

  # Find the adjoining quarantines and isolations
  doppelte_anonIDs_df_bereinigt <- bind_rows(map(doppelte_anonIDs, ~adjoincheck_concise(data_input = dop
                                                                                        testsubject = .
                                                                                        pair = pairslis

  df <- bind_rows(doppelte_anonIDs_df_bereinigt, einzelne_anonIDs_df)
}


# Final cleaning ------------------------------------------------------------


final_cleaning <- function(df_adjoined, externalinput){
  df_adjoined %>%
    left_join(externalinput$zeiten, by = c("AbsonderungVon" = "dates")) %>%
    mutate(Meldemonat = paste(year(AbsonderungVon), format.Date(AbsonderungVon, "%m"), sep = "_")) %>%
    mutate(Meldewoche = paste(year(AbsonderungVon), format.Date(AbsonderungVon, "%W"), sep = "_")) %>%
    mutate(dauer = as.numeric(AbsonderungBis - AbsonderungVon)) %>%
    select(-Q_Def_value, -Q_Def_url, -I_Duration_value, -I_Duration_url, -Q_Duration_value, -Q_Duration_
    mutate(result = NA) %>%
    mutate(result = ifelse(adjoiningQandI == 0, "I_correct_after_Q", result)) %>%
    mutate(result = ifelse(adjoiningQandI > 0, "I_too_long_after_Q", result)) %>%
    mutate(result = ifelse(is.na(adjoiningQandI), "No_I_after_Q", result)) %>%
    mutate(result = ifelse(AbsonderungVon < externalinput$StartDateKP | AbsonderungVon > externalinput$
}
```

Results

```
df <- tar_read(df)
demographiedaten <- tar_read(demographiedaten)
zeiten <- tar_read(externalinput)$zeiten
externalinput <- tar_read(externalinput)
resultslist <- tar_read(results)
```

We extracted 109 087 datasets from SurvNet. 73 220 entries fullfilled the definition (11 215 had missing dates, 108 entries had an IDs that did not lead to an existing person and 24 563 separation orders did not begin in the study period). We removed 371 entries because they had a presumed typing error in one of the dates. We also removed 30 duplicated isolations and 2 497 duplicated quarantines. For 3 484 quarantines we reduced the length by the overlap with a following isolation period. In the demographic data we found 266 123 inhabitants registered in Berlin Reinickendorf.

Results of statistical measures

- *Analysis of quantity of isolation and quarantines:* The local public health agency ordered $n_{\text{i}} = 24\ 433$ isolations and $n_{\text{q}} = 45\ 335$ quarantines ($n_{\text{i-p100}} = 9.2$ isolations and $n_{\text{q-p100}} = 17$ quarantines per 100 inhabitants). The number of quarantines and isolations by age group and recommendation period can bee seen in @ref(tab:agegrouptable)). The number of quarantines per 100 inhabitants $n_{\text{q-p100}}$ was 50.6 for the kindergarten-hildren and 64.9 in the school children compared to 10.5 in adults or 3.2 in the elderly. 46 817 (81.5 %) of persons had one spearation order (quarantine or isolation), 9 061 (15.8 %) had two spearation orders, 1 359 (2.4 %) had three spearation orders, 163 (0.3 %) had four spearation orders and 20 had five spearation orders - the maximum.
- *Analysis of the duration of isolation quarantines:* The median duration for isolations was $\tilde{d}_{\text{i}} = 10$ days (interquartile range 8 - 13). The duration did change in between different periods of recommendations. The median of the duration during the recommendation periods were: 14 days for the period No. 1, 8 days for the period No. 2 and 12 days for the period No. 3. The overall median duration for quarantines was $\tilde{d}_{\text{q}} = 8$ days (interquartile range 6 - 11). The median duration did differ between periods of different recommendations and age groups. The median of the duration during the recommendation periods were: 9 days for the period No. 1, 9 days for the period No. 2, 10 days for the period No. 3 and 4 days for the period No. 4. See Fig @ref(fig:duration). All together the public health agency ordered 684 years of isolations and 1 031 years of quarantine or 1 714 years in total.
- *Analysis of the ratio of contact persons per case:* The overal ratio of contact persons was $r_{\text{qi}} = 1.89$. In the period of the contact person defintion no. 1 the ratio was 2.88 in the period no. 2 the ratio was 1.96 and in the period no. 3 the ratio was: 0.95.
- *Analysis of isolations following quarantines:* In the time period from the start of the recording of quarantines 3 483 of 23 892 isolations had a directly preceeding quarantine and 532 a preceeding quarantine in the 1 to 7 days before the isolations. 3 483 of 45 272 quarantines in that time period had a directly following isolation (contained case) and 535 a isolation following in the days 1 to 7 after the quarantine (non-contained case). This did differ between different periods and recommendations see Fig @ref(fig:adjoining-quarantines-and-isolation).
- *Reduction of the reproductive number:* Assuming a total prevention of transmission by the quarantine order we calculated a reduction of 0.15 of the reproductive Number by quarantine orders.
- *Analysis of timeliness:* Our approximation of the median time period between the last contact and the beginning of the quarantine order was $\tilde{d}_{\text{delay}} = 4$ (interquartile range 1 - 6) during the time periods when 14 days were recommended as a quarantine duration.

All results

```
resultslist
```

```
## $queried
## [1] 109087
##
## $emptydates
## [1] 11215
##
## $wrongid
## [1] 108
##
## $outofrange
## [1] 24563
##
## $definitionfullfilled
## [1] 73220
##
## $typingerror
## [1] 371
##
## $deleted_duplicates_table
##  DatensatzKategorie deleted_duplicates
##            COVID-19                 30
##    Kontakt-COVID-19               2497
##
## $deleted_duplicates_quarantines
## [1] 2497
##
## $deleted_duplicates_isolations
## [1] 30
##
## $adjustedQuarantines
## [1] 3484
##
## $N
## [1] 266123
##
## $N_0_6
## [1] 18084
##
## $N_7_17
## [1] 27001
##
## $N_18_64
## [1] 158199
##
## $N_65_110
## [1] 62839
##
## $I_n
## [1] 24433
##
```

```
## $Q_n
## [1] 45335
##
## $I_p
## [1] 9.2
##
## $Q_p
## [1] 17
##
## $totaltime_groups
## # A tibble: 8 x 7
##   DatensatzKategor~ AgeGroup completeduratio~ completeduratio~  value percentage
##   <chr>             <chr>               <dbl>            <dbl>  <dbl>      <dbl>
## 1 COVID-19          0 to 6              15491             42.4  18084        6.2
## 2 COVID-19          7 to 17             44356            122.   27001       17.8
## 3 COVID-19          18 to 64           161215            442.  158199       64.6
## 4 COVID-19          65 to 1~            28445             77.9  62839       11.4
## 5 Kontakt-COVID-19  0 to 6              74977            205.   18084       19.9
## 6 Kontakt-COVID-19  7 to 17            139069            381.   27001       37
## 7 Kontakt-COVID-19  18 to 64           145456            399.  158199       38.7
## 8 Kontakt-COVID-19  65 to 1~            16712             45.8  62839        4.4
## # ... with 1 more variable: completeduration_person <dbl>
##
## $QundIproPerson_table
## # A tibble: 5 x 3
##   number     n     p
##    <int> <int> <dbl>
## 1      1 46817  81.5
## 2      2  9061  15.8
## 3      3  1359   2.4
## 4      4   163   0.3
## 5      5    20   0
##
## $QundIproPerson_1_order_n
## [1] 46817
##
## $QundIproPerson_1_order_p
## [1] 81.5
##
## $QundIproPerson_2_order_n
## [1] 9061
##
## $QundIproPerson_2_order_p
## [1] 15.8
##
## $QundIproPerson_3_order_n
## [1] 1359
##
## $QundIproPerson_3_order_p
## [1] 2.4
##
## $QundIproPerson_4_order_n
## [1] 163
##
```

```
## $QundIproPerson_4_order_p
## [1] 0.3
##
## $QundIproPerson_5_order_n
## [1] 20
##
## $QundIproPerson_5_order_p
## [1] 0
##
## $MedianeDauerI
##   0%  25%  50%  75% 100%
##    1    8   10   13   30
##
## $MedianeDauerI_Rec
## # A tibble: 3 x 2
##   I_Duration   quint
##   <chr>        <dbl>
## 1 I_Duration_1    14
## 2 I_Duration_2     8
## 3 I_Duration_3    12
##
## $MedianeDauerI_Rec_1
## 50%
##  14
##
## $MedianeDauerI_Rec_2
## 50%
##   8
##
## $MedianeDauerI_Rec_3
## 50%
##  12
##
## $MedianeDauerQ
##   0%  25%  50%  75% 100%
##    1    6    8   11   28
##
## $MedianeDauerQ_Rec
## # A tibble: 4 x 2
##   Q_Duration   quint
##   <chr>        <dbl>
## 1 Q_Duration_1     9
## 2 Q_Duration_2     9
## 3 Q_Duration_3    10
## 4 Q_Duration_4     4
##
## $MedianeDauerQ_Rec_1
## 50%
##   9
##
## $MedianeDauerQ_Rec_2
## 50%
##   9
##
```

```
## $MedianeDauerQ_Rec_3
## 50%
##  10
##
## $MedianeDauerQ_Rec_4
## 50%
##   4
##
## $qi_d
## [1] 625721
##
## $qi_d_in_y
## [1] 1714
##
## $q_d_in_y
## [1] 1031
##
## $i_d_in_y
## [1] 684
##
## $K_F_Verhaeltnis
## [1] 1.89
##
## $K_F_Verhaeltnis_QDef
## # A tibble: 3 x 4
##   q_def   covid_19 kontakt_covid_19 verhaeltnis
##   <chr>      <int>            <int>       <dbl>
## 1 Q_Def_1    10402            29965        2.88
## 2 Q_Def_2     2446             4791        1.96
## 3 Q_Def_3    11044            10516        0.95
##
## $K_F_Verhaeltnis_QDef_1
## [1] 2.88
##
## $K_F_Verhaeltnis_QDef_2
## [1] 1.96
##
## $K_F_Verhaeltnis_QDef_3
## [1] 0.95
##
## $I_after_Q
## $I_after_Q$I_correct_after_Q
## [1] 3483
##
## $I_after_Q$I_too_long_after_Q
## [1] 532
##
## $I_after_Q$No_I_after_Q
## [1] 19877
##
##
## $I_n_kptime
## [1] 23892
##
```

```
## $Q_with_I_after
## $Q_with_I_after$I_correct_after_Q
## [1] 3483
##
## $Q_with_I_after$I_too_long_after_Q
## [1] 535
##
## $Q_with_I_after$No_I_after_Q
## [1] 41254
##
##
## $Q_n_kptime
## [1] 45272
##
## $r
## [1] 0.15
##
## $Q_n_by_QDef
## # A tibble: 3 x 2
##   Q_Def       n
##   <chr>   <int>
## 1 Q_Def_1 29965
## 2 Q_Def_2  4791
## 3 Q_Def_3 10516
##
## $Q_with_correct_I_by_QDef_table
## # A tibble: 3 x 5
## # Groups:   Q_Def [3]
##   Q_Def   result                n     N percentage
##   <chr>   <chr>             <int> <int>      <dbl>
## 1 Q_Def_1 I_correct_after_Q  1802 29965          6
## 2 Q_Def_2 I_correct_after_Q   658  4791         14
## 3 Q_Def_3 I_correct_after_Q  1024 10516         10
##
## $Q_with_too_late_I_by_QDef_table
## # A tibble: 3 x 5
## # Groups:   Q_Def [3]
##   Q_Def   result                  n     N percentage
##   <chr>   <chr>               <int> <int>      <dbl>
## 1 Q_Def_1 I_too_long_after_Q   205 29965        0.7
## 2 Q_Def_2 I_too_long_after_Q    52  4791        1.1
## 3 Q_Def_3 I_too_long_after_Q   278 10516        2.6
##
## $Q_n_by_AgeGroup
## # A tibble: 4 x 2
##   AgeGroup      n
##   <ord>     <int>
## 1 0 to 6     9149
## 2 7 to 17   17528
## 3 18 to 64  16678
## 4 65 to 110  1980
##
## $Q_with_correct_I_by_Agegroup_table
## # A tibble: 4 x 5
```

```
## # Groups:   AgeGroup [4]
##   AgeGroup  result                n     N percentage
##   <ord>     <chr>             <int> <int>      <dbl>
## 1 0 to 6    I_correct_after_Q   434  9149        4.7
## 2 7 to 17   I_correct_after_Q   867 17528        4.9
## 3 18 to 64  I_correct_after_Q  1838 16678       11
## 4 65 to 110 I_correct_after_Q   345  1980       17.4
##
## $Q_with_too_late_I_by_Agegroup_table
## # A tibble: 4 x 5
## # Groups:   AgeGroup [4]
##   AgeGroup  result                 n     N percentage
##   <ord>     <chr>              <int> <int>      <dbl>
## 1 0 to 6    I_too_long_after_Q    97  9149        1.1
## 2 7 to 17   I_too_long_after_Q   194 17528        1.1
## 3 18 to 64  I_too_long_after_Q   210 16678        1.3
## 4 65 to 110 I_too_long_after_Q    34  1980        1.7
##
## $before_after_q_duration_no_4
## # A tibble: 4 x 4
## # Groups:   before_after_q_duration_no_4 [2]
##   before_after_q_duration_no_4 result                n      p
##   <chr>                        <chr>             <int>  <dbl>
## 1 no1_3                        I_correct_after_Q   5673 0.108
## 2 no1_3                        I_too_long_after_Q   543 0.0103
## 3 no4                          I_correct_after_Q   1295 0.0755
## 4 no4                          I_too_long_after_Q   527 0.0307
##
## $q_timeliness_median
##   0%  25%  50%  75% 100%
##    0    1    4    6   12
##
## $q_timeliness_mean
## # A tibble: 2 x 2
##   Q_Duration    mean
##   <chr>        <dbl>
## 1 Q_Duration_1  4.32
## 2 Q_Duration_3  4.01
##
## $q_timeliness_median_D1
##   0%  25%  50%  75% 100%
##    0    1    4    7   12
##
## $q_timeliness_median_D3
##   0%  25%  50%  75% 100%
##    0    2    3    6   12
##
## $agegroup_table
## # A tibble: 4 x 16
##   AgeGroup       N   q_n   i_n   q_p   i_p   q_d   i_d q_sum_in_y i_sum_in_y
##   <chr>      <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl>      <dbl>      <dbl>
## 1 0 to 6     18084  9149  1383  50.6   7.6   8.2  11.2      205.        42.4
## 2 7 to 17    27001 17528  3983  64.9  14.8   7.9  11.1      381        122.
## 3 18 to 64  158199 16678 16041  10.5  10.1   8.7  10.1      398.       442.
```

```
## 4 65 to 110  62839  1980  3026   3.2   4.8   8.4   9.4       45.8       77.9
## # ... with 6 more variables: q_sum_in_d_per_p <dbl>, i_sum_in_d_per_p <dbl>,
## #   contained <int>, containedp <dbl>, toolate <int>, toolatep <dbl>
##
## $qdef_table
## # A tibble: 3 x 16
##   Q_Def        N   q_n   i_n   q_p   i_p   q_d   i_d q_sum_in_y i_sum_in_y
##   <chr>    <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl>      <dbl>      <dbl>
## 1 Q_Def_1 266123 29973 10876  11.3   4.1   8.9   8.3       734.      248.
## 2 Q_Def_2 266123  4791  2446   1.8   0.9   9.5  11.4       124.       76.2
## 3 Q_Def_3 266123 10571 11111   4     4.2   5.9  11.8       172.      360.
## # ... with 6 more variables: q_sum_in_d_per_p <dbl>, i_sum_in_d_per_p <dbl>,
## #   contained <int>, containedp <dbl>, toolate <int>, toolatep <dbl>
##
## $total_table
## # A tibble: 1 x 16
##   total      N   q_n   i_n   q_p   i_p   q_d   i_d q_sum_in_y i_sum_in_y
##   <chr>  <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl>      <dbl>      <dbl>
## 1 total 266123 45335 24433    17   9.2   8.3  10.2      1031.       684.
## # ... with 6 more variables: q_sum_in_d_per_p <dbl>, i_sum_in_d_per_p <dbl>,
## #   contained <int>, containedp <dbl>, toolate <int>, toolatep <dbl>
```