Script 4b

Repeated pain sites

Peter Kamerman

22 January 2020

Contents

Objective

To determine whether a pain site was repeated at the time-point preceding or following the time-point the site registered as the site of worst pain.

Analysis notes

Definitions of missingness

Data were regarded as **missing** when *pain in the last week* data were not present for one or more of weeks 0, 12, 24, 36, 48. Data also were classified as **missing** when there were inconsistencies in the data across the variables collected within a week.

Definition of data inconsistencies

Pain was defined as *pain in the last week* being 'Yes', and *pain at its worst* being > 0. These two measurements were then the "gatekeeper" measurements, such that the two measurements both had to be positive ('Yes' and '> 0', respectively) in order for there to be any entries for *site of pain* and *site of worst pain.* Were the data were inconsistent (e.g., when there was no *pain in the last week* and *pain at its worst* = 0, but there were entries for *site of pain* and *site of worst pain*), then the *site of pain* and *site of worst pain* entries were marked as **inconsistent**.

Data also were considered **inconsistent** when *pain in the last week* = 'Yes', but *site of worst pain* = 'None'.

Lastly, data were considered **inconsistent** when *site of worst pain* was not listed as one of the pain locations for a given measurement week.

For analysis purposes, missing data in the *site of pain* columns were changed to **'No'** (pain not present in the site). This approach was conservative, but we believed that the approach would have the least effect on the outcome, while still retaining as many participants as possible.

Import data

```
df <- read_rds('data-cleaned/data-ADVANCE.rds') %>%
    select(ranid, interval_name, pain_in_the_last_week, pain_worst,
           site_worst, ends_with('_pain'), any_missing, interval_numeric)
```

Quick look

```
head(df)
```

```
## # A tibble: 6 x 20
##   ranid interval_name pain_in_the_las… pain_worst site_worst head_pain
##   <chr> <ord>         <chr>                 <dbl> <chr>      <chr>    
## 1 01-0… 0 weeks       No                        0 None       No       
## 2 01-0… 12 weeks      No                        0 None       No       
## 3 01-0… 24 weeks      No                        0 None       No       
## 4 01-0… 36 weeks      No                        0 None       No       
## 5 01-0… 48 weeks      No                        0 None       No       
## 6 01-0… 0 weeks       No                        0 None       No       
## # … with 14 more variables: cervical_pain <chr>, shoulder_pain <chr>,
## #   arm_pain <chr>, hand_pain <chr>, chest_pain <chr>,
## #   abdominal_pain <chr>, low_back_pain <chr>, buttock_pain <chr>,
## #   hip_groin_pain <chr>, leg_pain <chr>, genital_pain <chr>,
## #   foot_pain <chr>, any_missing <chr>, interval_numeric <dbl>
```

```
glimpse(df)
```

```
## Observations: 5,265
## Variables: 20
## $ ranid                 <chr> "01-0001", "01-0001", "01-0001", "01-0001"…
## $ interval_name         <ord> 0 weeks, 12 weeks, 24 weeks, 36 weeks, 48 …
## $ pain_in_the_last_week <chr> "No", "No", "No", "No", "No", "No", "Yes",…
## $ pain_worst            <dbl> 0, 0, 0, 0, 0, 0, 3, 3, 5, 0, 0, 0, 0, 0, …
## $ site_worst            <chr> "None", "None", "None", "None", "None", "N…
## $ head_pain             <chr> "No", "No", "No", "No", "No", "No", "No", …
## $ cervical_pain         <chr> "No", "No", "No", "No", "No", "No", "No", …
## $ shoulder_pain         <chr> "No", "No", "No", "No", "No", "No", "No", …
## $ arm_pain              <chr> "No", "No", "No", "No", "No", "No", "No", …
## $ hand_pain             <chr> "No", "No", "No", "No", "No", "No", "No", …
## $ chest_pain            <chr> "No", "No", "No", "No", "No", "No", "No", …
## $ abdominal_pain        <chr> "No", "No", "No", "No", "No", "No", "No", …
## $ low_back_pain         <chr> "No", "No", "No", "No", "No", "No", "No", …
## $ buttock_pain          <chr> "No", "No", "No", "No", "No", "No", "No", …
## $ hip_groin_pain        <chr> "No", "No", "No", "No", "No", "No", "Yes",…
## $ leg_pain              <chr> "No", "No", "No", "No", "No", "No", "No", …
## $ genital_pain          <chr> "No", "No", "No", "No", "No", "No", "No", …
## $ foot_pain             <chr> "No", "No", "No", "No", "No", "No", "No", …
## $ any_missing           <chr> "No", "No", "No", "No", "No", "No", "No", …
## $ interval_numeric      <dbl> 0, 12, 24, 36, 48, 0, 12, 24, 36, 48, 0, 1…
```

Basic clean

```
# Clean and process data
df %<>%
    filter(any_missing == 'No') %>%
    select(-any_missing)
```

Quick tabulation

Analysis data set for the period 0 to 48 weeks

```
# Tabulate data
xtabs(~interval_name, data = df)
```

```
## interval_name
##  0 weeks 12 weeks 24 weeks 36 weeks 48 weeks
##      787      787      787      787      787
```

Prepare "pain present" filters

```
# Generate the pain sequences
df_sequence <- df %>%
    select(ranid, interval_name, pain_in_the_last_week) %>%
    mutate(pain_in_the_last_week = as.character(pain_in_the_last_week)) %>%
    pivot_wider(names_from = interval_name,
                values_from = pain_in_the_last_week) %>%
    unite(col = 'sequence', -ranid, sep = ' ') %>%
    mutate(any_pain = str_detect(sequence, pattern = 'Yes'))

# Extract sequences
df_yes <- df_sequence %>%
    mutate(yes_2 = str_detect(sequence, pattern = 'Yes Yes')) %>%
    mutate(yes_3 = str_detect(sequence, pattern = 'Yes Yes Yes')) %>%
    mutate(yes_4 = str_detect(sequence, pattern = 'Yes Yes Yes Yes'))

# Create filters
vec_filter_2yes <- df_yes %>%
    filter(yes_2 == TRUE) %>%
    .$ranid

vec_filter_3yes <- df_yes %>%
    filter(yes_3 == TRUE) %>%
    .$ranid

vec_filter_4yes <- df_yes %>%
```

```r
    filter(yes_4 == TRUE) %>%
    .$ranid
```

Prepare pain sites

```r
# Convert 'Yes' to site name
df_sites <- df %>%
    mutate(head_pain = ifelse(head_pain == 'Yes',
                              yes = 'Head',
                              no = 'No')) %>%
    mutate(cervical_pain = ifelse(cervical_pain == 'Yes',
                                  yes = 'Neck',
                                  no = 'No')) %>%
    mutate(shoulder_pain = ifelse(shoulder_pain == 'Yes',
                                  yes = 'Shoulder',
                                  no = 'No')) %>%
    mutate(arm_pain = ifelse(arm_pain == 'Yes',
                             yes = 'Arm',
                             no = 'No')) %>%
    mutate(hand_pain = ifelse(hand_pain == 'Yes',
                              yes = 'Hand',
                              no = 'No')) %>%
    mutate(chest_pain = ifelse(chest_pain == 'Yes',
                               yes = 'Chest',
                               no = 'No')) %>%
    mutate(abdominal_pain = ifelse(abdominal_pain == 'Yes',
                                   yes = 'Abdomen',
                                   no = 'No')) %>%
    mutate(low_back_pain = ifelse(low_back_pain == 'Yes',
                                  yes = 'Low back',
                                  no = 'No')) %>%
    mutate(buttock_pain = ifelse(buttock_pain == 'Yes',
                                 yes = 'Buttocks',
                                 no = 'No')) %>%
    mutate(hip_groin_pain = ifelse(hip_groin_pain == 'Yes',
                                   yes = 'Hip/groin',
                                   no = 'No')) %>%
    mutate(leg_pain = ifelse(leg_pain == 'Yes',
                             yes = 'Leg',
                             no = 'No')) %>%
    mutate(genital_pain = ifelse(genital_pain == 'Yes',
                                 yes = 'Genitals',
                                 no = 'No')) %>%
    mutate(foot_pain = ifelse(foot_pain == 'Yes',
                              yes = 'Feet',
                              no = 'No')) %>%
    select(-pain_worst, -interval_numeric)

# Unite *_pain per week and then pivot wider based on week
df_united <- df_sites %>%
    unite(col = 'sites' ,
          ends_with('_pain'),
          sep = ' ', ) %>%
    pivot_wider(names_from = interval_name,
                values_from = sites) %>%
    filter(pain_in_the_last_week != 'No') %>%
    group_by(ranid) %>%
    # Fill up and down so that you look at time periods preceding and following
    # the week when a pain site was listed as the worst site of pain
    fill(ends_with('weeks'), .direction = 'updown')
```

Extract and analyse data

2 adjacent yes' ONLY

```
# 2 yes's in a row ONLY
df_2yes <- df_united %>%
    filter(!ranid %in% vec_filter_4yes) %>%
    filter(!ranid %in% vec_filter_3yes) %>%
    filter(ranid %in% vec_filter_2yes)

# Find repeats
df_2duplicates <- df_2yes %>%
    # Replace <NA> with explicit NA (weeks when there is no pain)
    mutate_at(vars(ends_with('weeks')),
              str_replace_na) %>%
    # Unite the weeks columns, separating with a '-'
    unite(col = 'united',
          ends_with('weeks'),
          sep = '-') %>%
    # Now separate the column based on NAs
    # This will result in contiguous 'pain' blocks seperated by weeks of no pain
    separate(col = united,
             into = c('block_1', 'block_2', 'block_3', 'block_4', 'block_5'),
             sep = 'NA') %>%
    # Within each block, count how many times a worst pain site occurs
    # (even if it is not listed as a worst pain site on the other occasions)
    mutate(count_1 = str_count(block_1, site_worst),
           count_2 = str_count(block_2, site_worst),
           count_3 = str_count(block_3, site_worst),
           count_4 = str_count(block_4, site_worst),
           count_5 = str_count(block_5, site_worst)) %>%
    filter(count_1 > 1 | count_2 > 1 | count_3 > 1 | count_4 > 1 | count_5 > 1) %>%
    select(ranid, site_worst, starts_with('count'))
```

```
## Warning: Expected 5 pieces. Missing pieces filled with `NA` in 147 rows [1,
## 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

```
knitr::kable(df_2duplicates,
             caption = 'Participants with 2 sequential occurences of a pain')
```

Table 1: Participants with 2 sequential occurences of a pain

ranid	site_worst	count_1	count_2	count_3	count_4	count_5
01-0019	Neck	2	0	NA	NA	NA
01-0019	Head	1	2	NA	NA	NA
01-0105	Chest	0	0	2	0	NA
01-0142	Leg	0	2	0	0	NA
01-0159	Head	2	1	NA	NA	NA
01-0174	Feet	0	0	2	0	NA

ranid	site_worst	count_1	count_2	count_3	count_4	count_5
01-0183	Hip/groin	0	2	0	0	NA
01-0197	Abdomen	1	0	2	NA	NA
01-0231	Head	0	2	NA	NA	NA
01-0243	Chest	2	0	1	NA	NA
01-0285	Leg	2	0	NA	NA	NA
01-0322	Feet	2	0	0	0	NA
01-0387	Chest	2	0	0	0	NA
01-0389	Abdomen	2	0	0	0	NA
01-0415	Low back	0	0	0	2	NA
01-0424	Feet	1	2	0	NA	NA
01-0436	Shoulder	2	0	0	0	NA
01-0454	Low back	2	0	0	0	NA
01-0521	Genitals	1	2	NA	NA	NA
01-0639	Leg	1	2	0	NA	NA
01-0689	Low back	0	2	0	0	NA
01-0693	Hip/groin	2	0	0	0	NA
01-0721	Head	0	0	0	2	NA
01-0735	Head	2	0	0	0	NA
01-0826	Abdomen	0	0	2	0	NA
01-0882	Shoulder	0	0	2	0	NA
01-0929	Shoulder	2	0	0	0	NA
01-1000	Feet	0	0	0	2	NA
01-1037	Feet	0	0	2	0	NA

```r
# Number of unique participants with a sequence of 2 sequential occurences of a pain
length(unique(df_2duplicates$ranid))
```

```
## [1] 28
```

```r
# Number of unique participants with ONLY 2 yes' in a row
length(unique(df_2yes$ranid))
```

```
## [1] 79
```

```r
# Proportion of participants with worst pain repeated in an adjacent time block
## Irrespective of whether the worst pain was the worst pain again.
round(length(unique(df_2duplicates$ranid)) / length(unique(df_2yes$ranid)), 2)
```

```
## [1] 0.35
```

3 adjacent yes' ONLY

```r
# At least 3 yes's in a row
df_3yes <- df_united %>%
    filter(!ranid %in% vec_filter_4yes) %>%
    filter(ranid %in% vec_filter_3yes)

# Find repeats
df_3duplicates <- df_3yes %>%
    # Replace <NA> with explicit NA (weeks when there is no pain)
    mutate_at(vars(ends_with('weeks')),
              str_replace_na) %>%
    # Unite the weeks columns, separating with a '-'
    unite(col = 'united',
          ends_with('weeks'),
          sep = '-') %>%
    # Now separate the column based on NAs
    # This will result in contiguous 'pain' blocks seperated by weeks of no pain
    separate(col = united,
             into = c('block_1', 'block_2', 'block_3', 'block_4', 'block_5'),
             sep = 'NA') %>%
    # Within each block, count how many times a worst pain site occurs
    # (even if it is not listed as a worst pain site on the other occasions)
    mutate(count_1 = str_count(block_1, site_worst),
           count_2 = str_count(block_2, site_worst),
           count_3 = str_count(block_3, site_worst),
           count_4 = str_count(block_4, site_worst),
           count_5 = str_count(block_5, site_worst)) %>%
    filter(count_1 > 1 | count_2 > 1 | count_3 > 1 | count_4 > 1 | count_5 > 1) %>%
    select(ranid, site_worst, starts_with('count'))
```

```
## Warning: Expected 5 pieces. Missing pieces filled with `NA` in 47 rows [1,
## 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

```
knitr::kable(df_3duplicates,
             caption = 'Participants with 2 or 3 sequential occurences of a pain')
```

Table 2: Participants with 2 or 3 sequential occurences of a pain

ranid	site_worst	count_1	count_2	count_3	count_4	count_5
01-0002	Low back	0	2	0	NA	NA
01-0060	Head	0	2	0	NA	NA
01-0077	Chest	3	0	0	NA	NA
01-0189	Abdomen	2	0	NA	NA	NA
01-0229	Feet	2	0	NA	NA	NA
01-0233	Chest	1	2	NA	NA	NA
01-0248	Chest	2	0	0	NA	NA
01-0339	Abdomen	0	2	0	NA	NA
01-0670	Leg	0	0	3	NA	NA
01-0750	Low back	0	0	3	NA	NA
01-0751	Abdomen	2	1	NA	NA	NA
01-0841	Head	0	2	NA	NA	NA
01-0866	Head	0	3	0	NA	NA
01-0935	Head	2	0	0	NA	NA
01-0947	Low back	0	3	0	NA	NA
01-0978	Low back	0	2	0	NA	NA
01-1016	Abdomen	0	0	3	NA	NA

```
# Number of unique participants with a sequence of 2 sequential occurences of a pain
length(unique(df_3duplicates$ranid))
```

```
## [1] 17
```

```
# Number of unique participants with ONLY 3 yes' in a row
length(unique(df_3yes$ranid))
```

```
## [1] 20
```

```
# Proportion of participants with worst pain repeated in an adjacent time block
## Irrespective of whether the worst pain was the worst pain again.
round(length(unique(df_3duplicates$ranid)) / length(unique(df_3yes$ranid)), 2)
```

```
## [1] 0.85
```

4 adjacent yes' ONLY

```r
# 4 yes's in a row ONLY
df_4yes <- df_united %>%
    filter(ranid %in% vec_filter_4yes)

# Find repeats
df_4duplicates <- df_4yes %>%
    # Replace <NA> with explicit NA (weeks when there is no pain)
    mutate_at(vars(ends_with('weeks')),
              str_replace_na) %>%
    # Unite the weeks columns, separating with a '-'
    unite(col = 'united',
          ends_with('weeks'),
          sep = '-') %>%
    # Now separate the column based on NAs
    # This will result in contiguous 'pain' blocks seperated by weeks of no pain
    separate(col = united,
             into = c('block_1', 'block_2', 'block_3', 'block_4', 'block_5'),
             sep = 'NA') %>%
    # Within each block, count how many times a worst pain site occurs
    # (even if it is not listed as a worst pain site on the other occasions)
    mutate(count_1 = str_count(block_1, site_worst),
           count_2 = str_count(block_2, site_worst),
           count_3 = str_count(block_3, site_worst),
           count_4 = str_count(block_4, site_worst),
           count_5 = str_count(block_5, site_worst)) %>%
    filter(count_1 > 1 | count_2 > 1 | count_3 > 1 | count_4 > 1 | count_5 > 1) %>%
    select(ranid, site_worst, starts_with('count'))
```

```
## Warning: Expected 5 pieces. Missing pieces filled with `NA` in 14 rows [1,
## 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14].
```

```r
knitr::kable(df_4duplicates,
             caption = 'Participants with 2 or more sequential occurences of a pain')
```

Table 3: Participants with 2 or more sequential occurences of a pain

ranid	site_worst	count_1	count_2	count_3	count_4	count_5
01-0014	Head	3	0	NA	NA	NA
01-0014	Low back	2	0	NA	NA	NA
01-0107	Head	3	0	NA	NA	NA
01-0154	Leg	5	NA	NA	NA	NA
01-0168	Low back	0	2	NA	NA	NA
01-0168	Shoulder	0	2	NA	NA	NA
01-0857	Leg	0	4	NA	NA	NA
01-0857	Feet	0	2	NA	NA	NA

```r
# Number of unique participants with a sequence of 2 sequential occurences of a pain
length(unique(df_4duplicates$ranid))
```

```
## [1] 5
```

```r
# Number of unique participants with 4 or 5 yes' in a row
length(unique(df_4yes$ranid))
```

```
## [1] 5
```

```r
# Proportion of participants with worst pain repeated in an adjacent time block
## Irrespective of whether the worst pain was the worst pain again.
round(length(unique(df_4duplicates$ranid)) / length(unique(df_4yes$ranid)), 2)
```

```
## [1] 1
```

Session information

```r
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Mojave 10.14.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] magrittr_1.5    forcats_0.4.0   stringr_1.4.0   dplyr_0.8.3
##  [5] purrr_0.3.3     readr_1.3.1     tidyr_1.0.0     tibble_2.1.3
##  [9] ggplot2_3.2.1   tidyverse_1.2.1
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_0.2.5 xfun_0.10        haven_2.1.1      lattice_0.20-38
##  [5] colorspace_1.4-1 vctrs_0.2.0      generics_0.0.2   htmltools_0.4.0
##  [9] yaml_2.2.0       utf8_1.1.4       rlang_0.4.2      pillar_1.4.2
## [13] glue_1.3.1       withr_2.1.2      modelr_0.1.5     readxl_1.3.1
## [17] lifecycle_0.1.0  munsell_0.5.0    gtable_0.3.0     cellranger_1.1.0
## [21] rvest_0.3.4      evaluate_0.14    knitr_1.25       fansi_0.4.0
## [25] highr_0.8        broom_0.5.2      Rcpp_1.0.3       scales_1.0.0
## [29] backports_1.1.5  jsonlite_1.6     hms_0.5.1        digest_0.6.23
## [33] stringi_1.4.3    grid_3.6.1       cli_2.0.0        tools_3.6.1
## [37] lazyeval_0.2.2   crayon_1.3.4     pkgconfig_2.0.3  zeallot_0.1.0
## [41] ellipsis_0.3.0   xml2_1.2.2       lubridate_1.7.4  assertthat_0.2.1
## [45] rmarkdown_1.16   httr_1.4.1       rstudioapi_0.10  R6_2.4.1
## [49] nlme_3.1-141     compiler_3.6.1
```