Using FeatureExtraction

Martijn J. Schuemie

2017-10-13

Contents

1 Introduction

The `FeatureExtraction` package can be used to create features for a cohort, using the information stored in the Common Data Model. A cohort is defined a set of persons who satisfy one or more inclusion criteria for a duration of time. Features can for example be diagnoses observed prior to entering the cohort. Some people might also refer to such features as 'baseline characteristics', or to features in general as 'covariates', and we will use those terms interchangeably throughout this vignette.

This vignette describes how features can be contructed using the default covariate definitions embeded in the package. Although these definitions allow quite some customization through predefined parameters, it is possible that someone needs more customization. In this case, the reader is referred to the other vignettes included in this package that deal with constructing completely custom covariates.

This vignette will first describe how to specify which features to construct. In many situations, for example when using `FeatureExtraction` as part of another package such as `CohortMethod` or `PatientLevelPrediction`, that is all one needs to know about the `FeatureExtraction` package, as the actual calling of the package is done by the other package. However, it is also possible to use this package on its own, for example to create a descriptive characterization of a cohort to include in a paper.

2 Covariate settings

Users can specify which covariates to construct in three ways:

1. Choose the default set of covariates.
2. Choose from a set of prespecified analyses.
3. Create a set of custom analyses.

An **analysis** is a process that creates one or more similar covariates. For example, one analysis might create a binary covariate for each condition observed in the condition_occurrence table in the year prior to cohort start, and another analysis might create a single covariate representing the Charlson comorbidity index.

Note that it is always possible to specify a set of concept IDs that can or can't be used to construct features. When choosing the default set (option 1) or the prespecified analysis (option 2) this can only be done across all analysis. When creating custom analyses (option 3) this can be specified per analysis.

For **advanced users**: It is also possible to specify a set of covariate IDs that need to be constructed. A covariate ID identifies a specific covariate, for example the Charlson comorbidity index, or the occurrence of a specific condition concept in a specific time window. A covariate ID is therefore not to be confused with a concept ID. The typical scenario where one might want to specify covariate IDs to construct is when someone already constructed covariates in one population, found a subset of covariates to be of interest, and would like to have only those covariates constructed in another population.

2.1 Using the default set of covariates

Using the default set of covariates is straightforward:

```
settings <- createDefaultCovariateSettings()
```

This will create a wide array of features, ranging from demographics, through conditions and drugs, to several risk scores.

Note that we could specify a set of concepts that should not be used to create covariates, for example:

```
settings <- createDefaultCovariateSettings(excludedCovariateConceptIds = 1124300,
                                           addDescendantsToExclude = TRUE)
```

This will create the default set of covariates, except those derived from concept 1124300 (the ingredient diclofenac) and any of its descendants (ie. all drugs containing the ingredient diclofenac).

2.2 Using prespecified analyses

The function `createCovariateSettings` allow the user to choose from a large set of predefined covariates. Type `?createCovariateSettings` to get an overview of the available options. For example:

```
settings <- createCovariateSettings(useDemographicsGender = TRUE,
                                    useDemographicsAgeGroup = TRUE,
                                    useConditionOccurrenceAnyTimePrior = TRUE)
```

This will create binary covariates for gender, age (in 5 year age groups), and each concept observed in the `condition_occurrence` table any time prior to (and including) the cohort start date.

Many of the prespecified analyses refer to a short, medium, or long term time window. By default, these windows are defined as:

- Long term: 365 days prior up to and including the cohort start date.
- Short term: 180 days prior up to and including the cohort start date.
- Medium term: 30 days prior up to and including the cohort start date.

However, the user can change these values. For example:

```
settings <- createCovariateSettings(useConditionEraLongTerm = TRUE,
                                    useConditionEraShortTerm = TRUE,
                                    useDrugEraLongTerm = TRUE,
                                    useDrugEraShortTerm = TRUE,
                                    longTermStartDays = -180,
```

```
                                    shortTermStartDays = -14,
                                    endDays = -1)
```

This redefines the long term window as 180 days prior up to (but not including) the cohort start date, and redefines the short term window as 14 days prior up to (but not including) the cohort start date.

Again, we can also specify which concept IDs should or should not be used to construct covariates:

```
settings <- createCovariateSettings(useConditionEraLongTerm = TRUE,
                                    useConditionEraShortTerm = TRUE,
                                    useDrugEraLongTerm = TRUE,
                                    useDrugEraShortTerm = TRUE,
                                    longTermStartDays = -180,
                                    shortTermStartDays = -14,
                                    endDays = -1,
                                    excludedCovariateConceptIds = 1124300,
                                    addDescendantsToExclude = TRUE)
```

2.3 Creating a set of custom covariates

This option should only be used by **advanced users**. It requires one to understand that at the implementation level, an analysis is a combination of a piece of highly parameterized SQL together with a specification of the parameter values. The best way to understand the available options is to take a prespecified analysis as starting point, and convert it to a detailed setting object:

```
settings <- createCovariateSettings(useConditionEraLongTerm = TRUE)
settings2 <- convertPrespecSettingsToDetailedSettings(settings)
settings2$analyses[[1]]
```

```
## $analysisId
## [1] 202
##
## $sqlFileName
## [1] "DomainConcept.sql"
##
## $parameters
## $parameters$analysisId
## [1] 202
##
## $parameters$analysisName
## [1] "ConditionEraLongTerm"
##
## $parameters$startDay
## [1] -365
##
## $parameters$endDay
## [1] 0
##
## $parameters$subType
## [1] "all"
##
## $parameters$domainId
## [1] "Condition"
##
```

```
## $parameters$domainTable
## [1] "condition_era"
##
## $parameters$domainConceptId
## [1] "condition_concept_id"
##
## $parameters$domainStartDate
## [1] "condition_era_start_date"
##
## $parameters$domainEndDate
## [1] "condition_era_end_date"
##
## $parameters$description
## [1] "One covariate per condition in the condition_era table overlapping with any part of the long te
##
##
## $includedCovariateConceptIds
## list()
##
## $includedCovariateIds
## list()
##
## $addDescendantsToInclude
## [1] FALSE
##
## $excludedCovariateConceptIds
## list()
##
## $addDescendantsToExclude
## [1] FALSE
```

One can create a detailed analysis settings object from scratch, and use it to create a detailed settings object:

```
analysisDetails <- createAnalysisDetails(analysisId = 1,
                                         sqlFileName = "DemographicsGender.sql",
                                         parameters = list(analysisId = 1,
                                                           analysisName = "Gender",
                                                           domainId = "Demographics"),
                                         includedCovariateConceptIds = c(),
                                         addDescendantsToInclude = FALSE,
                                         excludedCovariateConceptIds = c(),
                                         addDescendantsToExclude = FALSE,
                                         includedCovariateIds = c())

settings <- createDetailedCovariateSettings(list(analysisDetails))
```

2.4 Temporal covariates

Ordinarily, covariates are created for just a few time windows of interest, for example the short, medium, and long term windows described earlier. However, sometimes a more fine-grained temporal resolution is required, for example creating covariates for each day separately, in the 365 days prior to cohort start. We will refer to this type of covariates as *temporal covariates*. Temporal covariates share the same covariate ID across the time windows, and use a separate time ID to distinguish between time windows. There currently aren't many applications that are able to handle temporal covariats. For example, the `CohortMethod` package will

break when provided with temporal covariates. However, there are some machine learning algorithms in the `PatientLevelPrediction` package that require temporal covariates.

Again, we can just choose to use the default settings:

```
settings <- createDefaultTemporalCovariateSettings()
```

Or, we can choose from a set of prespecified temporal covariates:

```
settings <- createTemporalCovariateSettings(useConditionOccurrence = TRUE,
                                            useMeasurementValue = TRUE)
```

In this case we've chosen to create binary covariates for each concept in the `condition_occurrence` table, and continuous covariates for each measurement - unit combination in the `measurement` table in the CDM. By default, temporal covariates are created for each day separately in the 365 days before (but not including) the cohort start date. Different time windows can also be specified, for example creating 7 day intervals instead:

```
settings <- createTemporalCovariateSettings(useConditionOccurrence = TRUE,
                                            useMeasurementValue = TRUE,
                                            temporalStartDays = seq(-364, -7, by = 7),
                                            temporalEndDays = seq(-358, -1, by = 7))
```

Each time window includes the specified start and end day.

Similar to ordinary covariates, **advanced users** can also define custom analyses:

```
analysisDetails <- createAnalysisDetails(analysisId = 1,
                                         sqlFileName = "MeasurementValue.sql",
                                         parameters = list(analysisId = 1,
                                                           analysisName = "MeasurementValue",
                                                           domainId = "Measurement"),
                                         includedCovariateConceptIds = c(),
                                         addDescendantsToInclude = FALSE,
                                         excludedCovariateConceptIds = c(),
                                         addDescendantsToExclude = FALSE,
                                         includedCovariateIds = c())

settings <-  createDetailedTemporalCovariateSettings(list(analysisDetails))
```

3 Constructing covariates for a cohort of interest

Here we will walk through an example, creating covariates for two cohorts of interest: new users of diclofenac and new users of celecoxib.

3.1 Configuring the connection to the server

We need to tell R how to connect to the server where the data are. `CohortMethod` uses the `DatabaseConnector` package, which provides the `createConnectionDetails` function. Type `?createConnectionDetails` for the specific settings required for the various database management systems (DBMS). For example, one might connect to a PostgreSQL database using this code:

```
connectionDetails <- createConnectionDetails(dbms = "postgresql",
                                             server = "localhost/ohdsi",
                                             user = "joe",
```

```
                                                password = "supersecret")

cdmDatabaseSchema <- "my_cdm_data"
resultsDatabaseSchema <- "my_results"
```

The last two lines define the `cdmDatabaseSchema` and `resultSchema` variables. We'll use these later to tell R where the data in CDM format live, and where we want to write intermediate and result tables. Note that for Microsoft SQL Server, databaseschemas need to specify both the database and the schema, so for example `cdmDatabaseSchema <- "my_cdm_data.dbo"`.

3.2 Creating a cohort of interest

FeatureExtraction requires the cohorts to be instantiated in the `cohort` table in the Common Data Model, or in a table that has the same structure as the `cohort` table. We could create cohorts using a cohort definition tool, but here we'll just use some simple SQL to find the first drug era per person. Note that because we will be creating covariates based on data before cohort start, we are requiring 365 days of observation before the first exposure. FeatureExtraction will not check if a subject is observed during the specified time windows.

```
/**********************************
File cohortsOfInterest.sql
**********************************/

IF OBJECT_ID('@resultsDatabaseSchema.cohorts_of_interest', 'U') IS NOT NULL
  DROP TABLE @resultsDatabaseSchema.cohorts_of_interest;

SELECT first_use.*
INTO @resultsDatabaseSchema.cohorts_of_interest
FROM (
  SELECT drug_concept_id AS cohort_definition_id,
    MIN(drug_era_start_date) AS cohort_start_date,
    MIN(drug_era_end_date) AS cohort_end_date,
    person_id
  FROM @cdmDatabaseSchema.drug_era
  WHERE drug_concept_id = 1118084-- celecoxib
    OR drug_concept_id = 1124300 --diclofenac
  GROUP BY drug_concept_id,
    person_id
) first_use
INNER JOIN @cdmDatabaseSchema.observation_period
  ON first_use.person_id = observation_period.person_id
  AND cohort_start_date >= observation_period_start_date
  AND cohort_end_date <= observation_period_end_date
WHERE DATEDIFF(DAY, observation_period_start_date, cohort_start_date) >= 365;
```

This is parameterized SQL which can be used by the `SqlRender` package. We use parameterized SQL so we do not have to pre-specify the names of the CDM and result schemas. That way, if we want to run the SQL on a different schema, we only need to change the parameter values; we do not have to change the SQL code. By also making use of translation functionality in `SqlRender`, we can make sure the SQL code can be run in many different environments.

```
library(SqlRender)
sql <- readSql("cohortsOfInterest.sql")
sql <- renderSql(sql,
                 cdmDatabaseSchema = cdmDatabaseSchema,
```

```
                resultsDatabaseSchema = resultsDatabaseSchema)$sql
sql <- translateSql(sql, targetDialect = connectionDetails$dbms)$sql

connection <- connect(connectionDetails)
executeSql(connection, sql)
```

In this code, we first read the SQL from the file into memory. In the next line, we replace the two parameter names with the actual values. We then translate the SQL into the dialect appropriate for the DBMS we already specified in the `connectionDetails`. Next, we connect to the server, and submit the rendered and translated SQL.

If all went well, we now have a table with the cohorts of interest. We can see how many events per type:

```
sql <- paste("SELECT cohort_definition_id, COUNT(*) AS count",
             "FROM @resultsDatabaseSchema.cohorts_of_interest",
             "GROUP BY cohort_definition_id")
sql <- renderSql(sql, resultsDatabaseSchema = resultsDatabaseSchema)$sql
sql <- translateSql(sql, targetDialect = connectionDetails$dbms)$sql

querySql(connection, sql)
```

```
##   cohort_concept_id  count
## 1           1124300 240761
## 2           1118084  47293
```

3.3 Creating per-person covariates for a cohort of interest

We can create per-person covariates for one of the cohorts of interest, for example using the default settings:

```
covariateSettings <- createDefaultCovariateSettings()

covariateData <- getDbCovariateData(connectionDetails = connectionDetails,
                                    cdmDatabaseSchema = cdmDatabaseSchema,
                                    cohortDatabaseSchema = resultsDatabaseSchema,
                                    cohortTable = "cohorts_of_interest",
                                    cohortId = 1118084,
                                    rowIdField = "subject_id",
                                    covariateSettings = covariateSettings)

summary(covariateData)
```

```
## CovariateData object summary
##
## Number of covariates: 41330
## Number of non-zero covariate values: 25892630
```

3.3.1 Per-person covariate output format

The main component of the `covariateData` object is `covariates`:

```
covariateData$covariates
```

```
## ffdf (all open) dim=c(25892630,3), dimorder=c(1,2) row.names=NULL
## ffdf virtual mapping
##                PhysicalName VirtualVmode PhysicalVmode  AsIs VirtualIsMatrix PhysicalIsMatrix Physica
```

```
## rowId                rowId      integer       integer FALSE           FALSE           FALSE
## covariateId       list...1       double        double FALSE           FALSE           FALSE
## covariateValue    list...2       double        double FALSE           FALSE           FALSE
## ffdf data
##                rowId covariateId covariateValue
## 1                  1  4150129212              1
## 2                  2  4211231212              1
## 3                  3  4122924212              1
## 4                  4   436785212              1
## 5                  5  4231363212              1
## 6                  6  4031662212              1
## 7                  7 37016775212              1
## 8                  8  4180170212              1
## :                  :           :              :
## 25892623    25892623        1903              1
## 25892624    25892624        1903              3
## 25892625    25892625        1903              1
## 25892626    25892626        1903              1
## 25892627    25892627        1903              2
## 25892628    25892628        1903              2
## 25892629    25892629        1903              2
## 25892630    25892630        1903              2
```

The columns are defines as follows:

- `rowId` uniquely identifies a cohort entry. When calling `getDbCovariateData` we defined `rowIdField = "subject_id"`, so in this case the `rowId` is the same as the `subject_id` in the cohort table. In cases where a single subject can appear in the cohort more than once it is up to the user to create a field in the cohort table that uniquely identifies each cohort entry, and use that as `rowIdField`.
- `covariateId` identifies the covariate, and definitions of covariates can be found in the `cohortData$covariateRef` object.
- `covariateValue` field provides the value.

3.3.2 Saving the data to file

Creating covariates can take considerable computing time, and it is probably a good idea to save it for future sessions. Because `covariateData` objects use `ff`, we cannot use R's regular save function. Instead, we'll have to use the `saveCovariateData()` function:

```
saveCovariateData(covariateData, "covariates")
```

We can use the `loadCovariateData()` function to load the data in a future session.

3.3.3 Normalizing and removing redundancy

One reason for generating per-person covariates may be to use them in some form of machine learning. In that case it may be necessary to tidy the data before proceeding. The `tidyCovariateData` function can perform two tasks:

1. **Normalization**: scale all covariate values to a value between 0 and 1 (by dividing by the max value for each covariate).
2. **Removal of redundancy**: If every person in the cohort has the same value for a covariate (e.g. a cohort that is restricted to women will have the same gender covariate value for all) that value is redundant. Redundant values may wreak havok with some machine learning algorithms, for example causing a simple regression to no longer have a single solution. Similarly, groups of covariates may be

redundant (e.g. every person will belong to at least one age group, making one group redundant as it can be defined as the absence of the other groups).

```
tidyCovariates <- tidyCovariateData(covariateData,
                                    normalize = TRUE,
                                    removeRedundancy = TRUE)
```

If we want to know which redundant covariates were removed we can query the `metaData` object:

```
deletedCovariateIds <- tidyCovariates$metaData$deletedCovariateIds
deletedCovariateIds
```

```
##  [1]   1118084410   1118084412   1118084413  21603931410  21603931412  21603931413  21603932410  216
## [16] 900000010802      8532001        10003      8527004      2015006         1007
```

If we want to know what these numbers mean, we can use the `covariateRef` object that is part of any `covariateData` object. Because `covariateRef`, like other parts of `covariateData`, uses the `ff` package, it is advised to load the `ffbase` package which allows `ff` objects to be used as data frames:

```
library(ffbase)
covariateData$covariateRef[covariateData$covariateRef$covariateId %in% deletedCovariateIds, ]
```

```
##     covariateId                    covariateName analysisId conceptId
## 1       8527004                     race = White          4      8527
## 2       8532001                   gender = FEMALE          1      8532
## 3       2015006                  index year: 2015          6         0
## 4          1007                    index month: 1          7         0
## 5         10003                  age group: 50-51          3         0
## 6  900000010802 ...tance Abuse Coverage Indicator        802 900000010
## 7   21603933412 ...EUMATIC PRODUCTS, NON-STEROIDS        412  21603933
## 8   21603932412 ...ORY AND ANTIRHEUMATIC PRODUCTS        412  21603932
## 9   21603932410 ...ORY AND ANTIRHEUMATIC PRODUCTS        410  21603932
## 10   1118084412 ...s relative to index: celecoxib        412   1118084
## 11   1118084410 ...s relative to index: celecoxib        410   1118084
## 12  21603931412 ...index: MUSCULO-SKELETAL SYSTEM        412  21603931
## 13  21603991410 ...days relative to index: Coxibs        410  21603991
## 14  21603991412 ...days relative to index: Coxibs        412  21603991
## 15  21603931410 ...index: MUSCULO-SKELETAL SYSTEM        410  21603931
## 16  21603933410 ...EUMATIC PRODUCTS, NON-STEROIDS        410  21603933
## 17  21603933413 ...EUMATIC PRODUCTS, NON-STEROIDS        413  21603933
## 18  21603932413 ...ORY AND ANTIRHEUMATIC PRODUCTS        413  21603932
## 19   1118084413 ...s relative to index: celecoxib        413   1118084
## 20  21603991413 ...days relative to index: Coxibs        413  21603991
## 21  21603931413 ...index: MUSCULO-SKELETAL SYSTEM        413  21603931
```

3.4 Creating aggregated covariates for a cohort of interest

Often we do not need to have per-person covariates, but instead we are interested in aggregated statistics instead. For example, we may not need to know which persons are male, but would like to know what proportion of the cohort is male. We can aggregate per-person covariates:

```
covariateData2 <- aggregateCovariates(covariateData)
```

Of course, if all we wanted was aggregated statistics it would have been more efficient to aggregate them during creation:

```
covariateSettings <- createDefaultCovariateSettings()

covariateData2 <- getDbCovariateData(connectionDetails = connectionDetails,
                                     cdmDatabaseSchema = cdmDatabaseSchema,
                                     cohortDatabaseSchema = resultsDatabaseSchema,
                                     cohortTable = "cohorts_of_interest",
                                     cohortId = 1118084,
                                     covariateSettings = covariateSettings,
                                     aggregated = TRUE)
summary(covariateData2)
```

```
## CovariateData object summary
##
## Number of covariates: 41330
## Number of non-zero covariate values: 41330
```

Note that we specified `aggregated = TRUE`. Also, we are no longer required to define a `rowIdField` because we will no longer receive per-person data.

3.4.1 Aggregated covariate output format

The two main components of the aggregated `covariateData` object are `covariates` and `covariatesContinuous`, for binary and continous covariates respectively:

```
covariateData2$covariates
```

```
## ffdf (all open) dim=c(41326,3), dimorder=c(1,2) row.names=NULL
## ffdf virtual mapping
##              PhysicalName VirtualVmode PhysicalVmode  AsIs VirtualIsMatrix PhysicalIsMatrix PhysicalI
## covariateId        list..       double        double FALSE           FALSE            FALSE
## sumValue         list...1       double        double FALSE           FALSE            FALSE
## averageValue     list...2       double        double FALSE           FALSE            FALSE
## ffdf data
##        covariateId     sumValue averageValue
## 1     4.078214e+09 2.000000e+00 4.228956e-05
## 2     4.368402e+08 1.000000e+00 2.114478e-05
## 3     1.388962e+08 1.400000e+01 2.960269e-04
## 4     3.734882e+08 1.000000e+00 2.114478e-05
## 5     4.328982e+08 3.900000e+01 8.246464e-04
## 6     4.340782e+08 1.000000e+00 2.114478e-05
## 7     4.331342e+08 1.000000e+01 2.114478e-04
## 8     4.394172e+08 1.000000e+00 2.114478e-05
## :                :            :            :
## 41319 2.314271e+09 1.970000e+02 4.165521e-03
## 41320 2.314285e+09 1.331000e+03 2.814370e-02
## 41321 4.075713e+10 5.000000e+00 1.057239e-04
## 41322 4.274253e+10 1.000000e+00 2.114478e-05
## 41323 4.481643e+10 2.000000e+00 4.228956e-05
## 41324 4.625748e+10 1.000000e+00 2.114478e-05
## 41325 4.625749e+10 7.600000e+01 1.607003e-03
## 41326 4.625753e+10 1.160000e+02 2.452794e-03
```

```
covariateData2$covariatesContinuous
```

```
## ffdf (all open) dim=c(4,11), dimorder=c(1,2) row.names=NULL
```

```
## ffdf virtual mapping
##                    PhysicalName VirtualVmode PhysicalVmode  AsIs VirtualIsMatrix PhysicalIsMatrix Phys
## covariateId              list..       double        double FALSE           FALSE            FALSE
## countValue             list...1       double        double FALSE           FALSE            FALSE
## minValue               list...2       double        double FALSE           FALSE            FALSE
## maxValue               list...3       double        double FALSE           FALSE            FALSE
## averageValue           list...4       double        double FALSE           FALSE            FALSE
## standardDeviation      list...5       double        double FALSE           FALSE            FALSE
## medianValue            list...6       double        double FALSE           FALSE            FALSE
## p10Value               list...7       double        double FALSE           FALSE            FALSE
## p25Value               list...8       double        double FALSE           FALSE            FALSE
## p75Value               list...9       double        double FALSE           FALSE            FALSE
## p90Value              list...10       double        double FALSE           FALSE            FALSE
## ffdf data
##    covariateId   countValue     minValue     maxValue averageValue standardDeviation  medianValue
## 1  1901.000000 33013.000000     0.000000    21.000000     2.321168          2.657008     1.000000
## 2  1904.000000 31301.000000     0.000000     6.000000     1.307128          1.144543     1.000000
## 3  1902.000000 27951.000000     0.000000    13.000000     2.646671          2.723938     1.000000
## 4  1903.000000 30898.000000     0.000000     5.000000     1.234157          1.055174     1.000000
```

The columns are defines as follows:

- `covariateId` identifies the covariate, and definitions of covariates can be found in the `cohortData$covariateRef` object.
- `sumValue` is the sum of the covariate values. Because these are binary features, this is equivalent to the number of people that have the covariate with a value of 1.
- `averageValue` is the average covariate value. Because these are binary features, this is equivalent to the proportion of people that have the covariate with a value of 1.
- `countValue` is the number of people that have a value (for continous variables).
- `minValue`, `maxValue`, `averageValue`, `standardDeviation`, `medianValue`, `p10Value`, `p25Value`, `p75Value`, and `p90Value` all inform on the distribution of covariate values. Note that for some covariates (such as the Charlson comorbidity index) a value of 0 is interpreted as the value 0, while for other covariates (Such as blood pressure) 0 is interpreted as missing, and the distribution statistics are only computed over non-missing values. To learn which continuous covariates fall into which category one can consult the `missingMeansZero` field in the `covariateData$analysisRef` object.

3.5 Creating a table 1

One task supported by the `FeatureExtraction` package is creating a table of overall study population characteristics that can be include in a paper. Since this is typically the first table in a paper we refer to such a table as 'table 1'. A default table 1 is available in the `FeatureExtraction` package:

```
result <- createTable1(covariateData2)
print(result, row.names = FALSE, right = FALSE)
```

```
 Characteristic                      % (n = 47,293) Characteristic                                    % (n = 47,293)
 Age group                                            Heart failure                                    7
   0-1                                0               Ischemic heart disease                           7
   5-6                                0               Peripheral vascular disease                     13
   10-11                              1               Pulmonary embolism                               1
   15-16                              2               Venous thrombosis                                3
   20-21                              2             Medical history: Neoplasms
   25-26                              4               Hematologic neoplasm                           1
   30-31                              6               Malignant lymphoma                             0
   35-36                              7               Malignant neoplasm of anorectum                0
   40-41                              9               Malignant neoplastic disease                   6
   45-46                             11               Malignant tumor of breast                      2
   50-51                             14               Malignant tumor of colon                       0
   55-56                             13               Malignant tumor of lung                        0
   60-61                             11               Malignant tumor of urinary bladder             0
   65-66                              6               Primary malignant neoplasm of prostate         0
   70-71                              4             Medication use
   75-76                              3               Agents acting on the renin-angiotensin system 47
   80-81                              3               Antibacterials for systemic use               83
   85-86                              2               Antidepressants                               64
```

90-91	0
Gender: female	72
Race	
race = Black or African American	19
race = Other Race	19
race = White	62
Ethnicity	
ethnicity = Hispanic or Latino	2
ethnicity = Not Hispanic or Latino	17
Medical history: General	
Acute respiratory disease	37
Attention deficit hyperactivity disorder	2
Chronic liver disease	5
Chronic obstructive lung disease	19
Crohn's disease	1
Dementia	2
Depressive disorder	34
Diabetes mellitus	25
Gastroesophageal reflux disease	25
Gastrointestinal hemorrhage	4
Human immunodeficiency virus infection	1
Hyperlipidemia	36
Hypertensive disorder	50
Lesion of liver	1
Obesity	17
Osteoarthritis	45
Pneumonia	6
Psoriasis	1
Renal impairment	5
Rheumatoid arthritis	5
Schizophrenia	4
Ulcerative colitis	0
Urinary tract infectious disease	15
Viral hepatitis C	4
Visual system disorder	36
Medical history: Cardiovascular disease	
Atrial fibrillation	3
Cerebrovascular disease	6
Coronary arteriosclerosis	10
Heart disease	26
Antiepileptics	77
Antiinflammatory and antirheumatic products	100
Antineoplastic agents	27
Antipsoriatics	9
Antithrombotic agents	22
Beta blocking agents	54
Calcium channel blockers	48
Diuretics	63
Drugs for acid related disorders	83
Drugs for obstructive airway diseases	60
Drugs used in diabetes	31
Immunosuppressants	4
Lipid modifying agents	55
Opioids	80
Psycholeptics	86
Psychostimulants, agents used for adhd and nootropics	78

Characteristic	Value
Charlson comorbidity index	
Mean	2
Minimum	0
25th percentile	0
Median	1
75th percentile	3
Maximum	21
CHADS2Vasc	
Mean	1
Minimum	0
25th percentile	0
Median	1
75th percentile	2
Maximum	6
DCSI	
Mean	3
Minimum	0
25th percentile	0
Median	1
75th percentile	5
Maximum	13

Where applicable, these characteristics are drawn from analyses pertaining the 'long-term' windows, so concepts observed in the 365 days before up to and included the cohort start date.

The `createTable1` function requires a simple specification of what variables to include in the table. The default specifications included in the package can be reviewed by calling the `getDefaultTable1Specifications` function. The specification reference analysis IDs and covariate IDs, and in the default specification these IDs refer to those in the default covariate settings. It is possible to create custom table 1 specifications and use those instead.

Here we based table 1 on a `covariateData` object containing all default covariates, even though only a small fraction of covariates are used in the table. If we only want to extract those covariates needed for the table, we can use the `createTable1CovariateSettings` function:

```
covariateSettings <- createTable1CovariateSettings()

covariateData2b <- getDbCovariateData(connectionDetails = connectionDetails,
                                      cdmDatabaseSchema = cdmDatabaseSchema,
                                      cohortDatabaseSchema = resultsDatabaseSchema,
                                      cohortTable = "cohorts_of_interest",
                                      cohortId = 1118084,
                                      covariateSettings = covariateSettings,
                                      aggregated = TRUE)
summary(covariateData2b)
```

```
## CovariateData object summary
##
## Number of covariates: 90
## Number of non-zero covariate values: 90
```

3.6 Comparing two cohorts

Another task supported by the `FeatureExtraction` package is comparing two cohorts of interest. Suppose we want to compare two cohorts only on the variables included in the default table 1:

```
settings <- createTable1CovariateSettings(excludedCovariateConceptIds = c(1118084, 1124300),
                                           addDescendantsToExclude = TRUE)

covCelecoxib <- getDbCovariateData(connectionDetails = connectionDetails,
                                   cdmDatabaseSchema = cdmDatabaseSchema,
                                   cohortDatabaseSchema = resultsDatabaseSchema,
                                   cohortTable = "cohorts_of_interest",
                                   cohortId = 1118084,
                                   covariateSettings = settings,
                                   aggregated = TRUE)

covDiclofenac <- getDbCovariateData(connectionDetails = connectionDetails,
                                    cdmDatabaseSchema = cdmDatabaseSchema,
                                    cohortDatabaseSchema = resultsDatabaseSchema,
                                    cohortTable = "cohorts_of_interest",
                                    cohortId = 1124300,
                                    covariateSettings = settings,
                                    aggregated = TRUE)

std <- computeStandardizedDifference(covCelecoxib, covDiclofenac)
```

In this example we have chosen to exclude any covariates derived from the two concepts that were used to define the two cohorts: celecoxib (1118084), and diclofenac (1124300). We compute the standardized difference between the remaining covariates.

```
head(std)
```

```
##    covariateId      mean1       sd1     mean2       sd2        sd    stdDiff                     covariateName
## 90        1904 1.30712790 1.1445434 0.9100477 1.0777444 1.5721045 -0.2525787                      CHADS2VASc
## 4         3003 0.02404161 0.1550552 0.1183165 0.3439724 0.3773051  0.2498638                  age group: 15-16
## 24    80180210 0.44879792 0.6699309 0.2619693 0.5118305 0.8430764 -0.2216034 ...ative to index: Osteoarthritis
## 89        1902 2.64667075 2.7239376 1.8278085 2.6827256 3.8231993 -0.2141825 ...orbidity Severity Index (DCSI)
## 88        1901 2.32116804 2.6570083 1.6216871 2.5198808 3.6618974 -0.1910160 ...lson index - Romano adaptation
## 74 21601853410 0.48421542 0.6958632 0.3182368 0.5641259 0.8958034 -0.1852847 ... index: LIPID MODIFYING AGENTS
```

The `stdDiff` column contains the standardized difference. By default the data is ranked in descending order of the absolute value of the standardized difference, showing the covariate with the largest difference first.

We can also show the comparison as a standard table 1:

```
result <- createTable1(covCelecoxib, covDiclofenac)
print(result, row.names = FALSE, right = FALSE)
```

Characteristic	% (n = 47,293)	% (n = 240,761)	Std.Diff
Age group			
0-1	0	0	-0.01
5-6	0	0	0.00
10-11	1	3	0.11
15-16	2	12	0.25
20-21	2	6	0.14
25-26	4	9	0.14
30-31	6	10	0.11
35-36	7	10	0.07
40-41	9	9	0.00
45-46	11	9	-0.05
50-51	14	10	-0.09
55-56	13	9	-0.09
60-61	11	6	-0.13
65-66	6	2	-0.12
70-71	4	2	-0.11
75-76	3	1	-0.10
80-81	3	1	-0.10
85-86	2	1	-0.07
90-91	0	0	-0.01
Gender: female	72	73	0.00
Race			
race = Black or African American	19	28	0.14
race = Other Race	19	15	-0.06
race = White	62	57	-0.05
Ethnicity			
ethnicity = Hispanic or Latino	2	2	0.03
ethnicity = Not Hispanic or Latino	17	13	-0.08
Medical history: General			
Acute respiratory disease	37	39	0.03
Attention deficit hyperactivity disorder	2	4	0.07
Chronic liver disease	5	4	-0.05
Chronic obstructive lung disease	19	11	-0.14
Crohn's disease	1	0	-0.02
Dementia	2	1	-0.06
Depressive disorder	34	29	-0.06
Diabetes mellitus	25	17	-0.12
Heart failure	7	4	-0.09
Ischemic heart disease	7	4	-0.09
Peripheral vascular disease	13	8	-0.12
Pulmonary embolism	1	1	-0.05
Venous thrombosis	3	2	-0.06
Medical history: Neoplasms			
Hematologic neoplasm	1	1	-0.04
Malignant lymphoma	0	0	-0.02
Malignant neoplasm of anorectum	0	0	-0.02
Malignant neoplastic disease	6	4	-0.08
Malignant tumor of breast	2	1	-0.04
Malignant tumor of colon	0	0	-0.02
Malignant tumor of lung	0	0	-0.03
Malignant tumor of urinary bladder	0	0	-0.02
Primary malignant neoplasm of prostate	0	0	-0.02
Medication use			
Agents acting on the renin-angiotensin system	47	33	-0.16
Antibacterials for systemic use	81	80	0.00
Antidepressants	64	53	-0.10
Antiepileptics	77	74	-0.03
Antiinflammatory and antirheumatic products	73	72	-0.01
Antineoplastic agents	17	13	-0.08
Antipsoriatics	9	7	-0.05
Antithrombotic agents	22	11	-0.18
Beta blocking agents	54	40	-0.15
Calcium channel blockers	48	32	-0.17
Diuretics	63	48	-0.14
Drugs for acid related disorders	83	80	-0.03
Drugs for obstructive airway diseases	60	57	-0.02
Drugs used in diabetes	31	21	-0.14
Immunosuppressants	4	2	-0.07
Lipid modifying agents	48	32	-0.19
Opioids	80	79	0.00
Psycholeptics	86	81	-0.04
Psychostimulants, agents used for adhd and nootropics	78	78	0.00

Characteristic	Value	Value	Std.Diff

Gastroesophageal reflux disease	25	19	-0.09
Gastrointestinal hemorrhage	4	3	-0.04
Human immunodeficiency virus infection	1	1	0.01
Hyperlipidemia	36	25	-0.14
Hypertensive disorder	50	37	-0.14
Lesion of liver	1	0	-0.02
Obesity	17	17	-0.01
Osteoarthritis	45	26	-0.22
Pneumonia	6	4	-0.07
Psoriasis	1	1	-0.02
Renal impairment	5	3	-0.05
Rheumatoid arthritis	5	2	-0.09
Schizophrenia	4	3	-0.04
Ulcerative colitis	0	0	-0.02
Urinary tract infectious disease	15	16	0.01
Viral hepatitis C	4	3	-0.04
Visual system disorder	36	31	-0.06
Medical history: Cardiovascular disease			
Atrial fibrillation	3	1	-0.06
Cerebrovascular disease	6	4	-0.06
Coronary arteriosclerosis	10	5	-0.11
Heart disease	26	17	-0.15
Charlson comorbidity index			
Mean	2	2	-0.19
Minimum	0	0	
25th percentile	0	0	
Median	1	1	
75th percentile	3	2	
Maximum	21	27	
CHADS2Vasc			
Mean	1	1	-0.25
Minimum	0	0	
25th percentile	0	0	
Median	1	0	
75th percentile	2	1	
Maximum	6	6	
DCSI			
Mean	3	2	-0.21
Minimum	0	0	
25th percentile	0	0	
Median	1	0	
75th percentile	5	3	
Maximum	13	13	