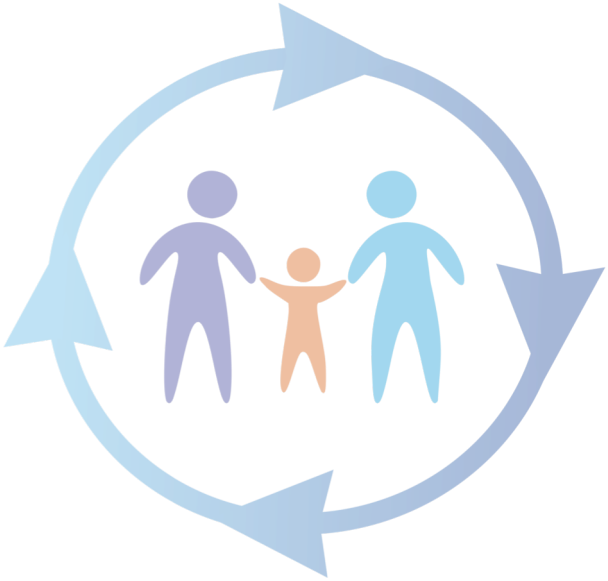
# PEDSnet Data Science Training



July 22, 2019

# Overview

- Web sessions

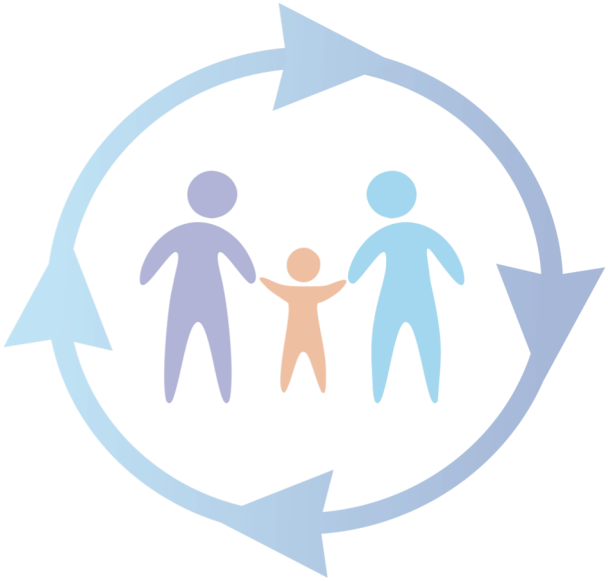
| | |
|---|---|
| Data Model & Vocabularies | April 2019 |
| Data Request Fulfillment | May 2019 |
| Web Application | June 2019 |
| PEDSnet Standard R Framework | July 2019 |

- Hands-on workshop – August 01-02, 2019

- Use of network data and scientific projects

PEDSnet
A Pediatric Learning Health System

# R Data Request Framework



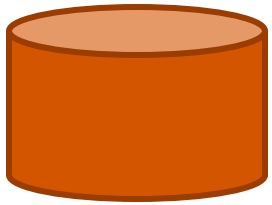
# PEDSnet Data Requests

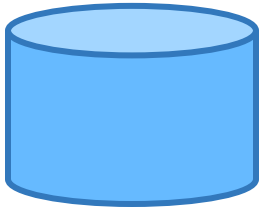
- Count data
  - Often feasibility requests
  - May require low-level computation for screening or aggregation
- Datasets
  - More involved selection process
  - Dataset content
  - ID management
- Analysis Results
  - Frequently an iterative process with requesters
  - Conceptually a dataset + further analysis

PEDSnet
A Pediatric Learning Health System

# Technical requirements

- Efficient computing
  - Take advantage of DBMS' strengths
  - Limit data transfers
- Portability
  - Different database architectures
  - Projects executing in stages across data versions
- Flexible output
  - Database vs flatfile
  - Logging and data
- Availability
  - Open source





PEDSnet
A Pediatric Learning Health System

# SDLC Requirements

- Reusability
  - Overlap among requests
  - Publication requirements
- Documentation
  - Code structure
  - Formal documentation with code
- Standardization
  - Predictable (ideally) behavior
  - Sharing across projects/users
- Improvement
  - Incorporation of bugfixes and new capabilities

PEDSnet
A Pediatric Learning Health System

# Standard Request Framework

- "Shell" within which to situate data fulfillment
  - Common utility code
  - Boilerplate
  - Database interaction
  - Affordances for preferred practices
- Iterative development
  - Continues to incorporate lessons learned and new capability

PEDSnet
A Pediatric Learning Health System

# Implemented in R

- Wide adoption
- Combination of data management and analytic tools
- Acceptance in literature
- Open source
- Documentation and publication tools

PEDSnet
A Pediatric Learning Health System

# “Tidyverse” libraries

- Common tools for data management
- Abstracts database/DBMS details
- Pushes computation to DBMS


- Widely adopted ecosystem
- Areas under construction

- Used as standard for data requests

# Framework structure

- Not a package
  - Intended for “packaged” distribution
  - Evolving rapidly
- Not turnkey development
  - Tools for data management
  - Requires request-specific coding
- Turnkey execution

- Request fulfillment lifecycle reflected in framework

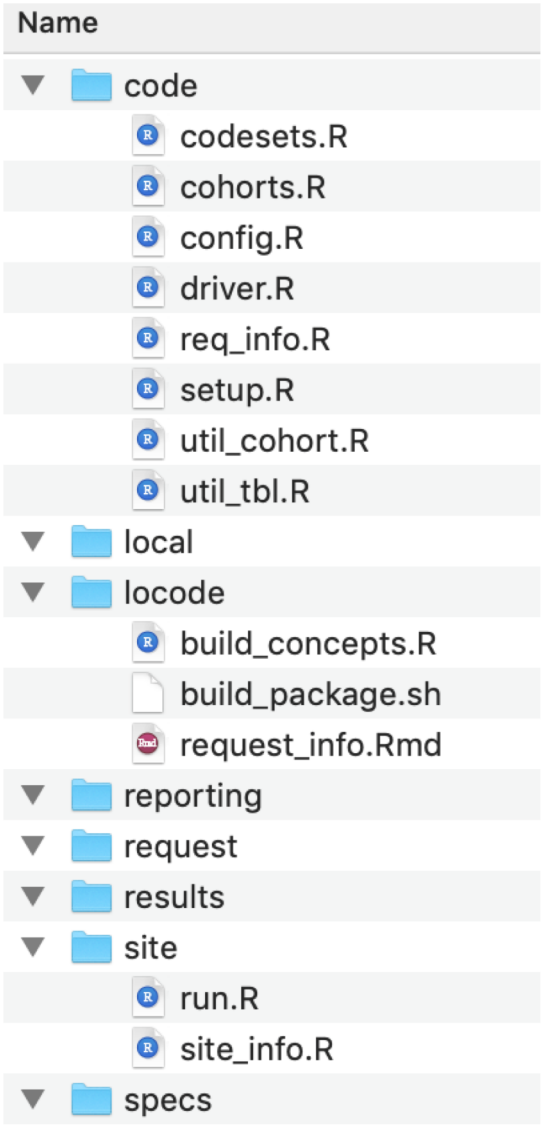
# Standard Framework Overview

| Dir Name | Description |
|---|---|
| code | • contains all the development and execution code for a query<br>• code development for query will occur here |
| specs | • stores codesets, ID mappings, or any other local specification file<br>• *.csv* are the default, with field types as *int* (concept_id), *char* (concept_name), *char* (concept_code), *char* (vocabulary_id), but default can be changed |
| locode | • contains files for producing metadata for packaging data request<br>• contains code for building concepts if developing codesets in the package<br>• stores any other code for local development |
| site | • contains site-specific information for database connections, schema names, study-specific directories, and preferences for logging and how to handle intermediate tables |

# Standard Framework Overview

| Dir Name | Description |
|---|---|
| request | • stores any relevant request information (e.g., documents from investigator, scientific specifications, etc) |
| local | • stores output meant to be kept locally and not in the *results* directory |
| results | • stores output from the code that is then sent to DCC |
| reporting | • stores code for summarizing (*for person-level data)* or formally reporting (*for queries with aggregate output)* data<br>• usually one or more R notebook files and supporting code |

# Standard Framework Structure

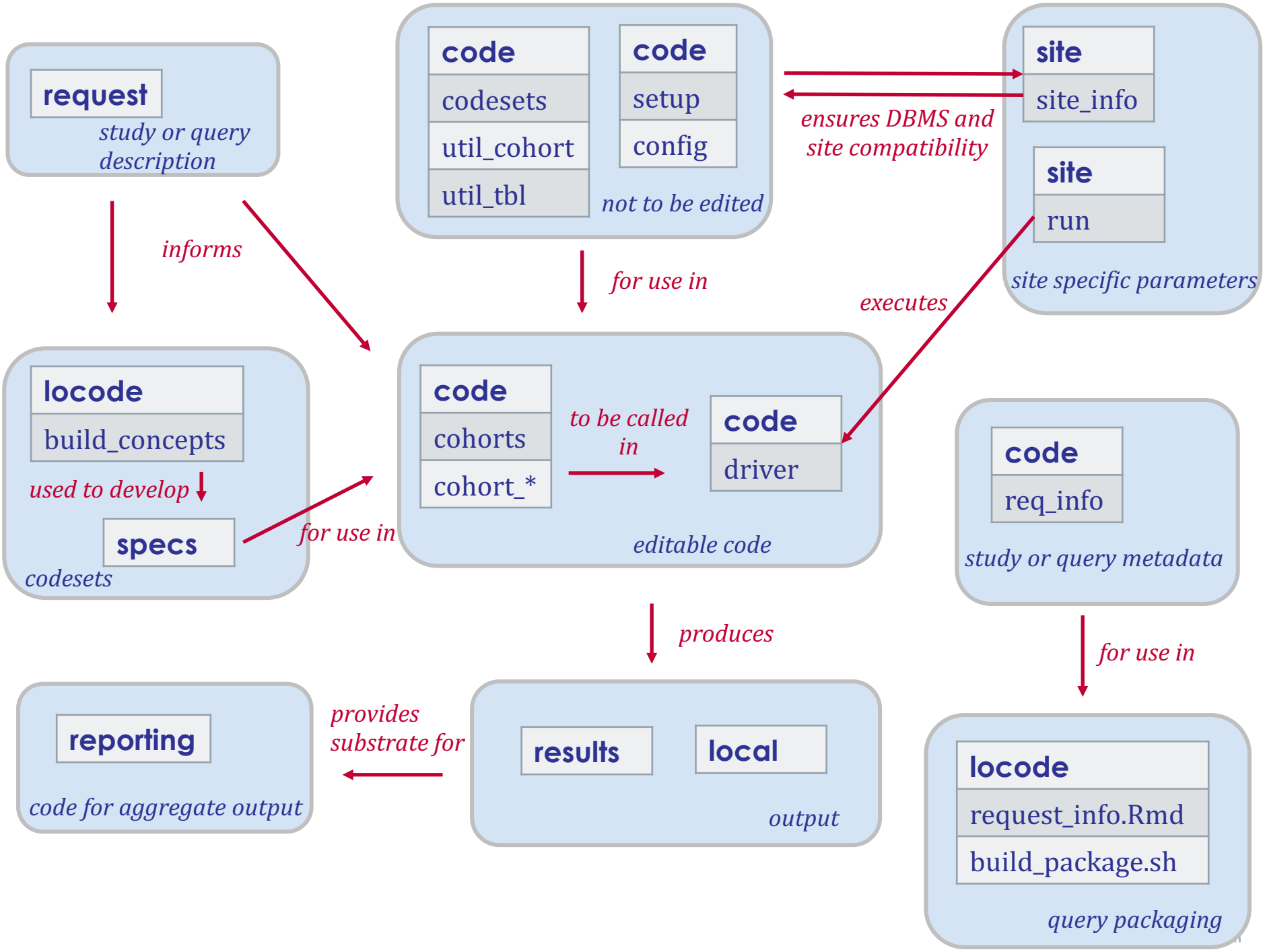


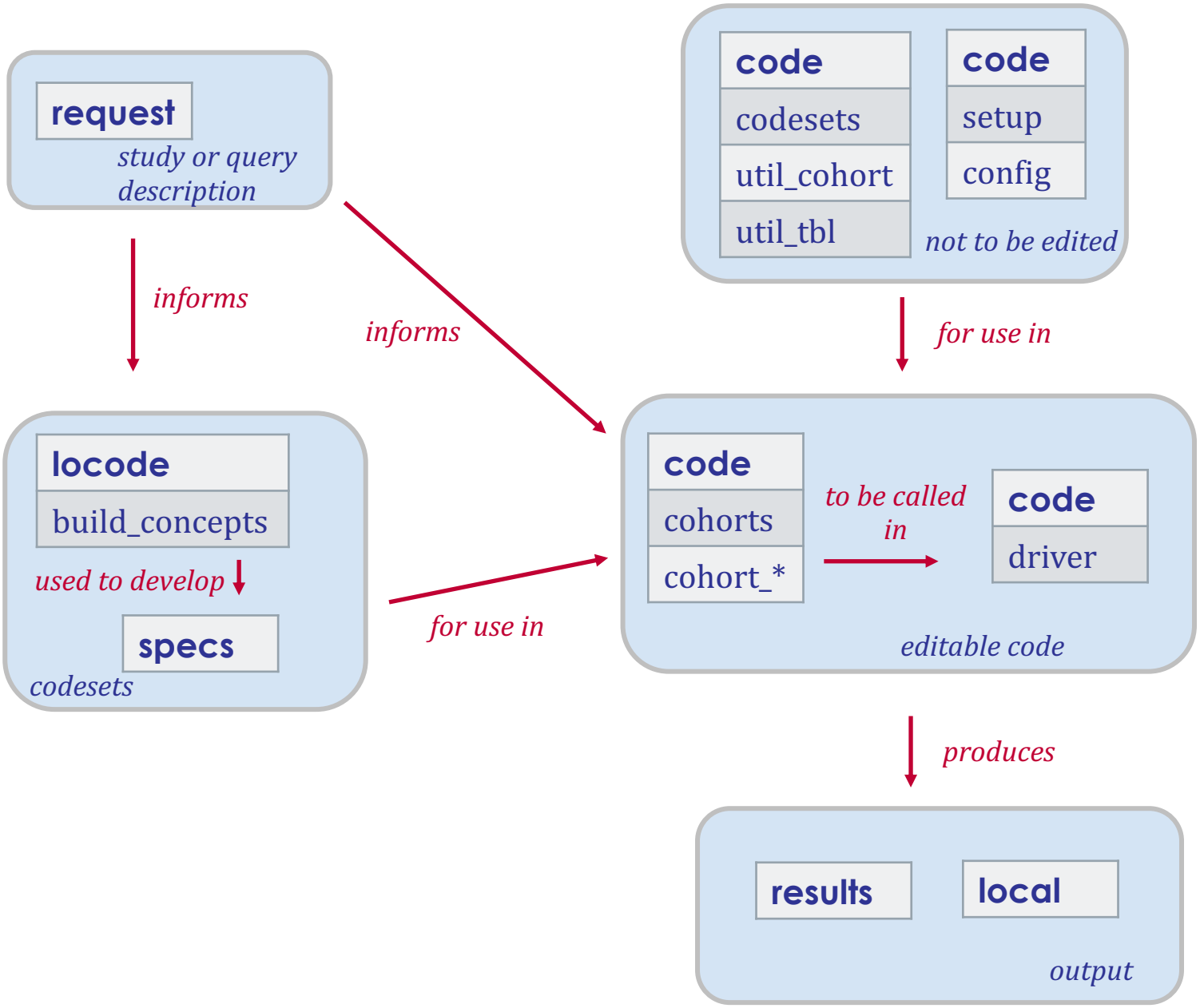

- code for query development and execution
  - ***cohorts, driver:*** edit and write query code
  - ***setup, config:*** required for DMBS interoperability or package standardization (do not edit)
  - ***util_cohort, util_tbl, codesets:*** utility functions to use in cohorts and driver (do not edit)
  - ***req_info:*** package metadata
- can create other files as needed (*in cohort_* format)*

- files for local use
  - ***request_info.Rmd, build_package.sh:*** packages files
  - ***build_concepts:*** explores codesets and vocabulary

- site-specific input:
  - ***site_info:*** database connectivity, schema and table names; *does not* change per query
  - ***run:*** directory for current package; *query specific parameters*

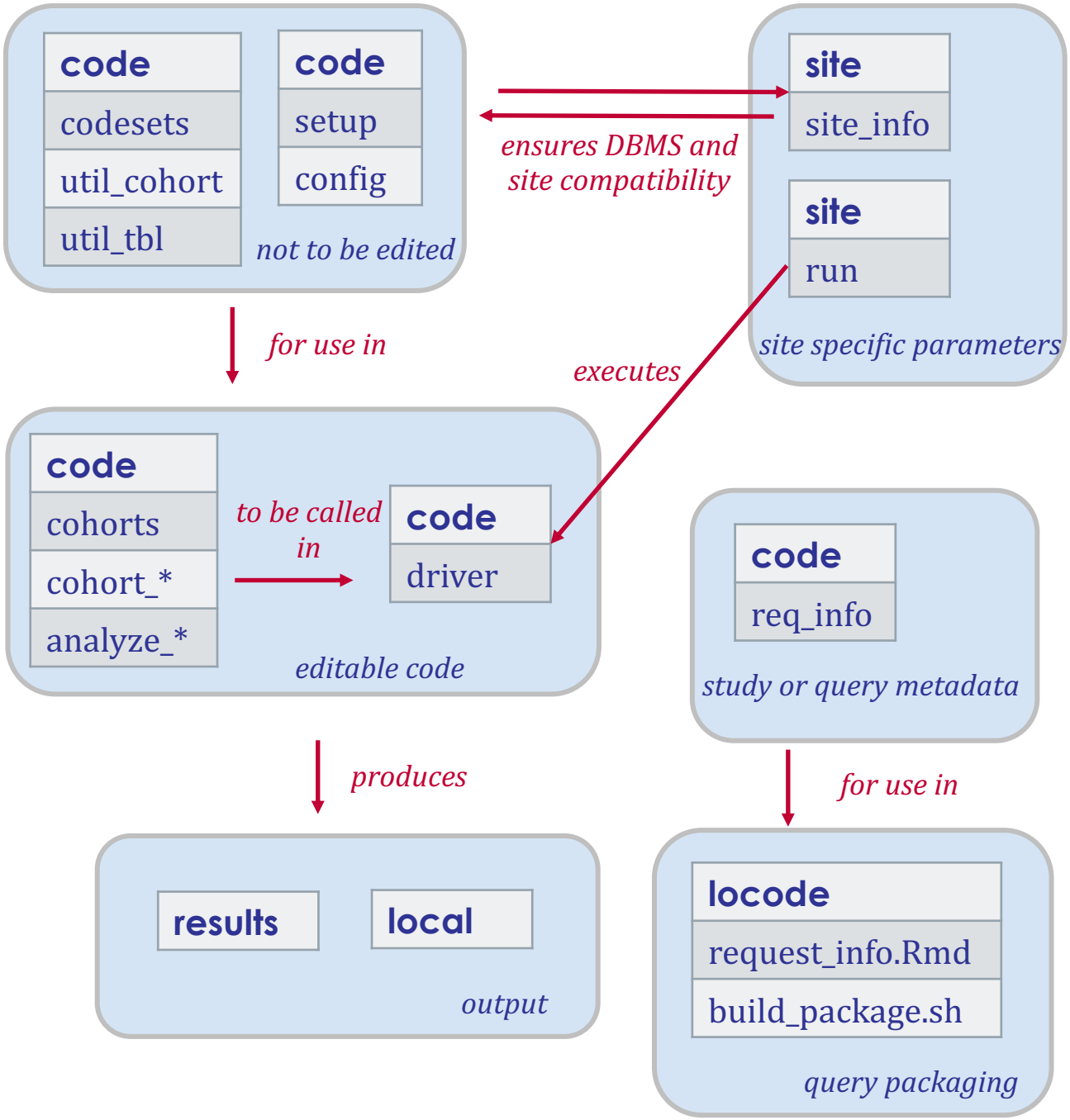
PEDSnet
A Pediatric Learning Health System


request
study or query description
informs
code
codesets
util_cohort
util_tbl
code
setup
config
not to be edited
for use in
site
site_info
ensures DBMS and site compatibility
site
run
site specific parameters
executes
locode
build_concepts
used to develop
specs
codesets
for use in
code
cohorts
cohort_*
to be called in
code
driver
editable code
code
req_info
study or query metadata
for use in
produces
reporting
code for aggregate output
provides substrate for
results
local
output
locode
request_info.Rmd
build_package.sh
query packaging

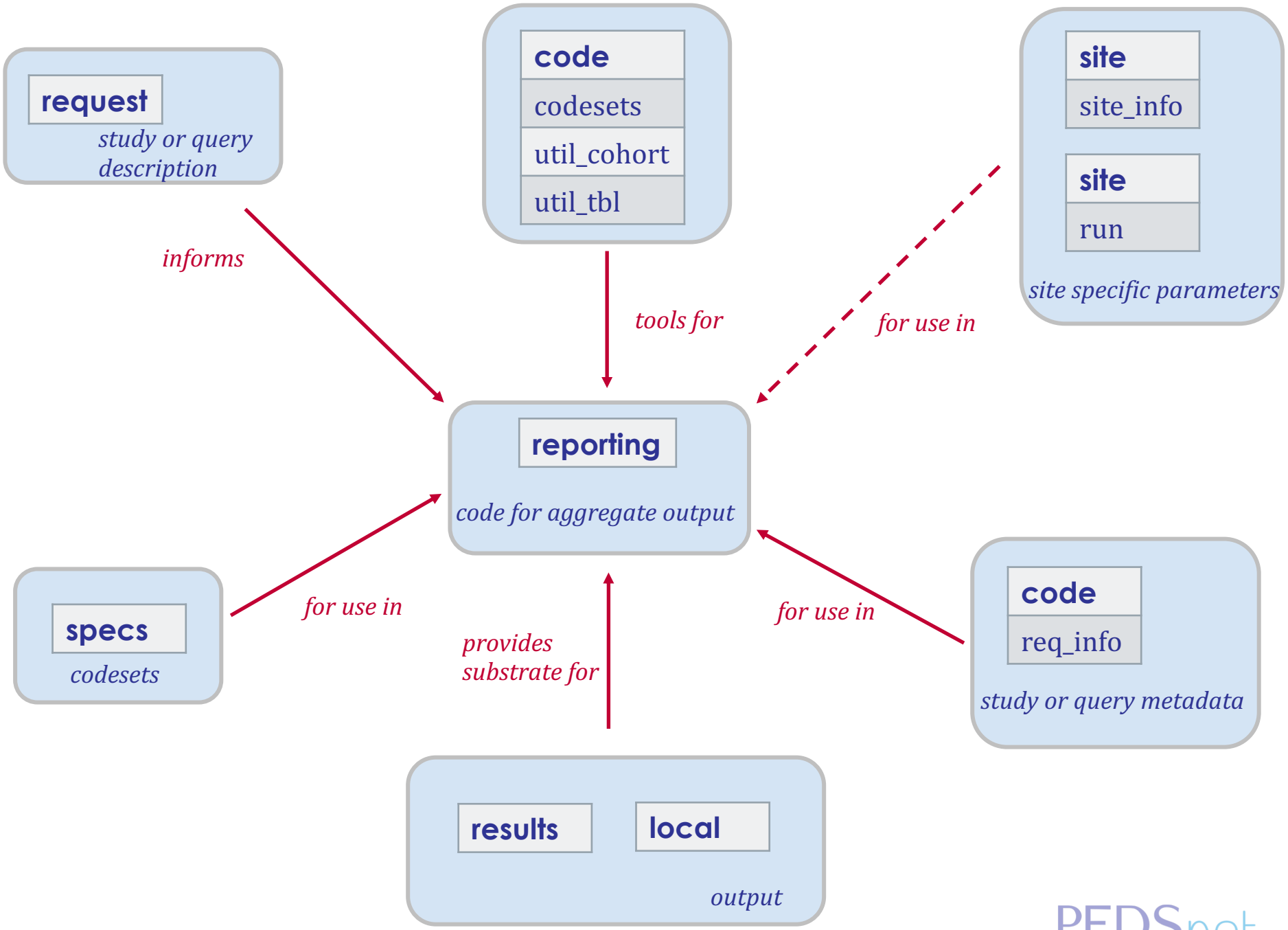



query packaging

PEDSnet
A Pediatric Learning Health System



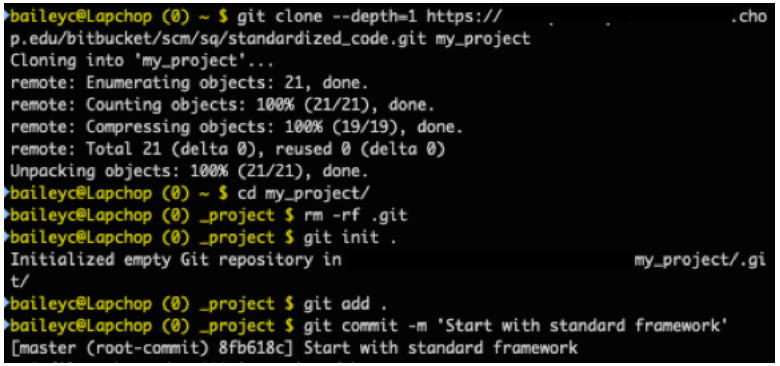

PEDSnet
A Pediatric Learning Health System




PEDSnet
A Pediatric Learning Health System

# Getting Started

1. Use as a repository template – easier to set up; doesn't track changes to standard framework
   - Clone the current repo
   - Delete the .git directory tree
   - Git init a new repo
   - Add and commit framework files
   - Begin development




2. Use as a Git submodule – tracks the framework repo, but requires a higher degree of git expertise

PEDSnet
A Pediatric Learning Health System

# Using `config()`

- Reduces dependence on global variables
- Should be used for all non-parameter request data

- Simple key-value store
  - `config('item')` returns current value (invisibly)
  - `config('item', value)` sets current value, then returns it
- Missing item returns NULL

# Specifying request-specific information

| Location | Tasks |
|---|---|
| `code/req_info.R` | • Define discrete metadata about request (*e.g.* provenance, CDM type, version) |
| `site/site_info.R` | • Define site-specific database architecture<br>• Set defaults for request output<br>• Same file may be used across many requests |
| `site/run.R` | • Define request-specific location<br>• Define output options |
| `locode/req_info.Rmd` | • Narrative description of request<br>• Instructions for execution<br>• Dependencies |

PEDSnet
A Pediatric Learning Health System

# Accessing database tables

- Framework presumes one database connection, though others can be added
- Use utility functions, rather than direct names

| | |
|---|---|
| `cdm_tbl('name')` | Connect to a CDM data table |
| `vocabulary_tbl('name')` | Connect to a CDM vocabulary table |
| `results_tbl('name')` | Connect to a results table (existing) |
| `intermed_name('name', temporary)`<br>`results_name('name')`<br>`temp_name('name')` | Return the full name of a results table<br>• Possibly temporary<br>• Defaults to random name<br>• Need not exist |

PEDSnet
A Pediatric Learning Health System

# Creating database tables

- Use `_new` utility functions, rather than base dplyr functions

| | |
|---|---|
| `compute_new(…)` | Materialize a table from a query expression<br>• Let temporary status be set by config<br>• Specify any needed indices |
| `copy_to_new(…)` | Copy local data to database<br>• Let temporary status be set by config<br>• Specify any needed indices<br>• Best to use named args and . |
| `collect_new(…)` | Retrieve data from query to local store |

# Using codesets

- Static codesets live in `specs` directory
- Follow standard structure unless you need other information for execution/documentation

| | |
|---|---|
| `read_codeset('name')` | Read codeset from a CSV file into local variable<br>• More common for reporting |
| `load_codeset('name')` | Read codeset from a CSV file into (possibly temporary) database table<br>• More common for query execution<br>• Given codeset loaded only once<br>• Default index on `concept_id` |

```
find_exposed <- function(meds = load_codeset('criterion_1_rx_concapts'),
                         drug_tbl = cdm_tbl('drug_exposure')) {

  scdfs <- meds %>%
    inner_join(vocabulary_tbl('concept_ancestor'),
               by = c('drug_concept_id' = 'descendant_concept_id')) %>%
    inner_join(vocaulary_tbl('concept'),
               by = c('ancestor_concept_id' = 'concept_id')) %>%
    filter(concept_class_id == 'Clinical Drug Form') %>%
    select(descendant_concept_id, scdf_concept_id = ancestor_concept_id,
           scdf_name = concept_name) %>%
    compute_new(indexes = list('descendant_concept_id'))

  inner_join(drug_tbl, meds,
             by = c('drug_concept_id' = 'concept_id')) %>%
    select(drug_exposure_id, person_id, drug_concept_id, drug_start_date) %>%
    left_join(scdfs, by = c('drug_concept_id' = 'descendant_concept_id')) %>%
    group_by('scdf_concept_id', 'scdf_name') %>%
    summarize(exposures = n(), pts = n_distinct(person_id))
}
```

# Writing results

- Use `output_tbl()` for nearly all purposes

| | |
|---|---|
| `name` | Name (unqualified) of output. Defaults to name of data variable. |
| `local` | TRUE: write results to local holding dir<br>FALSE (default): write results to returned data<br>(Not used for database output) |
| `file` | If TRUE, write results to CSV file. Defaults to true in production mode, FALSE otherwise. |
| `db` | If TRUE, write results to a (non-temporary) database table. Defaults to opposite of `file`. |

# Query fulfillment code

- Functions to retrieve and characterize data developed in `cohort_*.R`
  - May only need `cohorts.R` for simpler request
- Functions to perform more complex analyses in `analyze_*.R`
  - Analyses that are part of query, not reporting
- Load request-specific libraries as needed here

# Query writing

- Develop functions that manipulate data expressions
  - One task per function
  - Use parameters to allow subsetting and debugging
- Few restrictions on scope or approach
  - Use utility functions
  - Document
- Materialize intermediates and use indices as needed

# Cohort helpers

- Qualifying data

| | |
|---|---|
| `add_site(data)` | Adds a site name column to data.<br>• Do not assume database has this |
| `lookback_facts(data, index_dates, lookback)` | Find subset of `data` within `lookback` days of a person's index date<br>• Useful for subsetting data based on cohort entry |

```r
exp_baseline_dx <- function(med_tbl = cdm_tbl('drug_exposure'),
                            incl_meds = load_codeset('cohort_meds')
                            dx_tbl = cdm_tbl('condition_occurrence'),
                            baseline = 365L) {
  exposed <- med_tbl %>%
    inner_join(incl_meds, by = ('drug_concept_id' = 'concept_id')) %>%
    group_by(drug_concept_id, person_id) %>%
    summarize(cohort_start_date = min(drug_exposure_start_date))

  exposed %>%
    lookback_facts(dx_tbl, exposed, lookback = baseline) %>%
    add_site()
}
```

# Simple privacy risk reduction

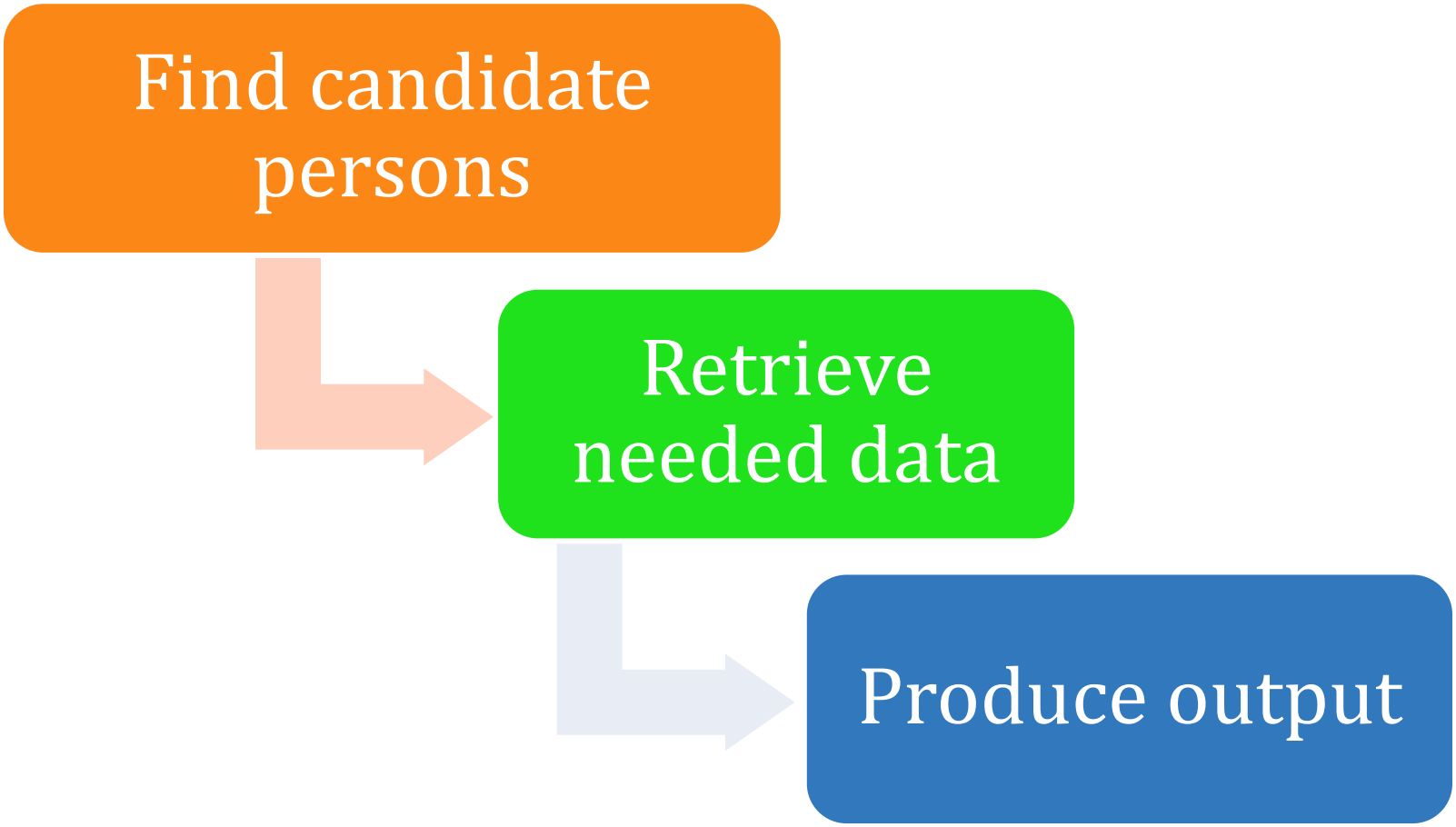
| | |
|---|---|
| `gen_xwalk(data, id)` | Create nonce replacements for an ID |
| `new_id(data, xwalk)` | Add or replace an ID with a new one |
| `dates_to_ages(data)` | Replace `_date` columns with ages in days<br>Also deletes `_datetime` columns |
| `scrub_person_info(data)` | All of the above |

- Used for simple regulatory requirements during data extraction
- More complete scrubbing of datasets done at a later step via PEDSnet::Lessidentify

PEDSnet
A Pediatric Learning Health System

```
pers_xwalk <- gen_xwalk(union(result_0, result_1),
                        id_col = 'person_id')
vis_xwalk <- gen_xwalk(result_1,
                       id_col = 'visit_occurrence_id')

scrub_0 <- scrub_person_info(result_0
                             person_xwalk = pers_xwalk)
scrub_1 <- new_id(result_1,
                  id_col = 'visit_occurrence_id',
                  xwalk = vis_xwalk,
                  replace = TRUE) %>%
  dates_to_ages() %>%
  new_id(id_col = 'person_id',
         xwalk = pwes_xwalk,
         replace = TRUE)
```

# Putting it all together - `.run()`



# Logging

- Simple utility functions to track steps in execution
  - Timing data (CPU and wall)
  - Additional user-defined data – must be consistent across each step

| | |
|---|---|
| `init_sum(…)` | Start the list<br>• Define custom columns to include<br>• Specify any needed indices |
| `append_sum(…)` | Add an event |
| `output_sum()` | Write the list – follows data output config |

- Warnings/errors and SQL trace always written to console
  - File in production mode, standard message output otherwise

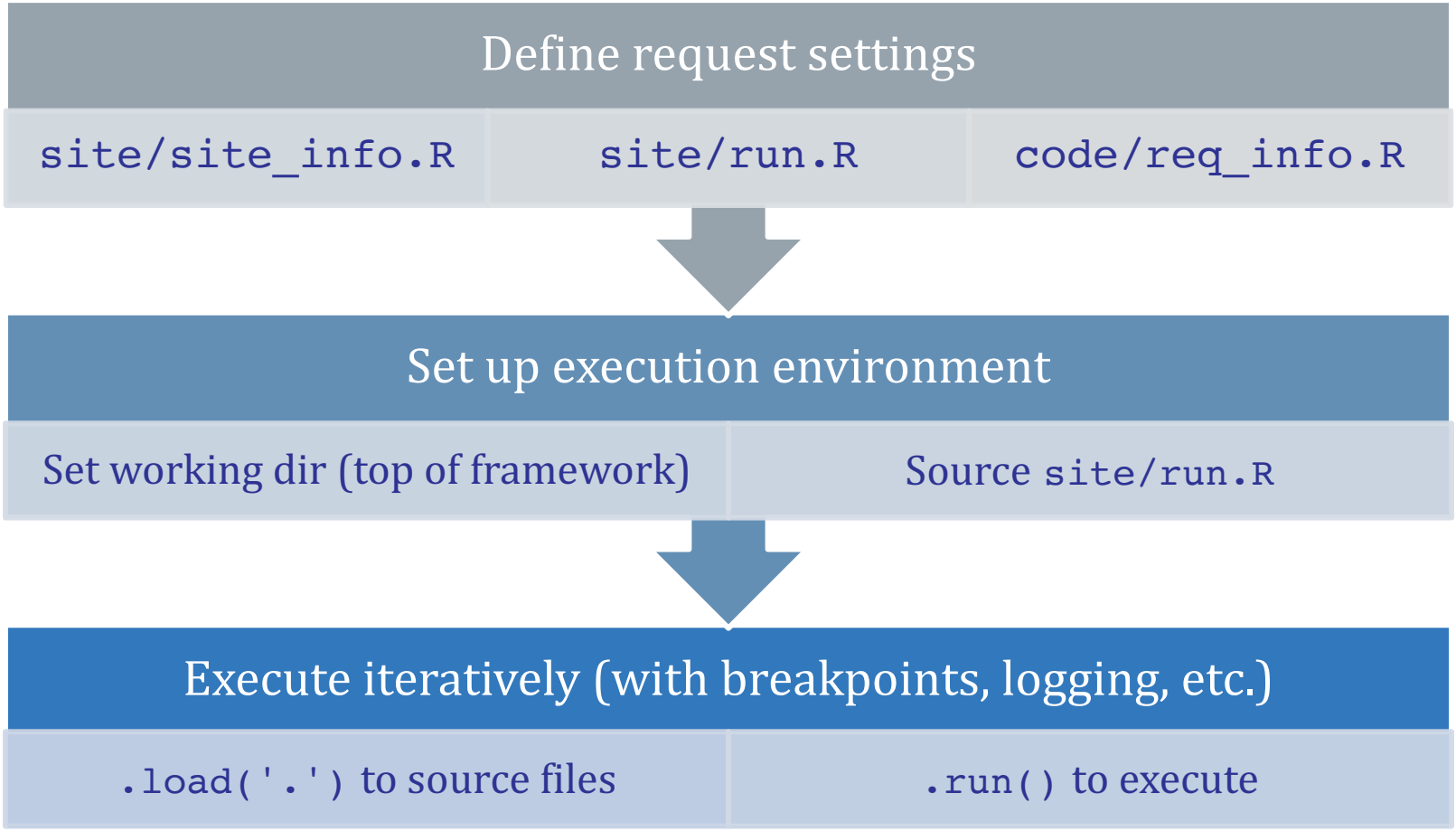
# Typical `.run()` flow

```
init_sum(stage = 'Start', persons = 0L)

result_0 <- cohort_step0(load_codeset('criterion_0'))

append_sum(stage = 'Dx',
      persons = distinct_ct(result_0))

result_1 <- cohort_step1(base = result_0,
      load_codeset('criterion_1'))

append_sum(stage = 'Rx',
      persons = distinct_ct(result_1))

…
```

# Typical `.run()` flow

```
append_sum(stage = 'Crosswalk', persons = 0L)
pers_xwalk <- gen_xwalk(union(result_0, result_1))
append_sum(stage = 'Writing output', persons = 0L)
output_tbl(new_id(result_0, 'person_id', pers_xwalk))
output_tbl(new_id(result_1, 'person_id', pers_xwalk))
…
append_sum(stage = 'Done', persons = 0L)
output_sum()
```

# Testing/Debugging




Tip: `.env_cleanup()` before exiting

PEDSnet
A Pediatric Learning Health System

# Packaging

1. Edit settings as needed
   - `config('execution_mode', 'production')`
   - Auto-detect location
   - Typically do not retain intermediates
   - Appropriate values for other settings in `site/site_info.R` and `site/run.R`
2. Run `locode/build_package.sh` on clean HEAD
   - Requires tools to render PDF from Rmd
3. Test package on model of target environment
4. Execute

# Reporting

- Development in `reporting` directory
- R notebook preferred format for report formatting
- Use functions for repeated processing
  - Complex processing better in its own .R file(s)
- Typically will `source()` files from framework to get to data
  - Remember working dir is notebook dir during knitting
  - If you are `source()`ing many file, consider `.load()`

# Query Example: Testing Algorithms to Identify Patients with Lupus

- Objective
  - Feasibility / pre-study request to determine how much overlap there is in the PEDSnet database between two different algorithms that identify patients with lupus

- Structure of Query:
  - *code*:
    - *cohorts.R*:
      - functions to identify patients with lupus and definitions for the different definitions of lupus
    - *driver.R*:
      - calls functions from *cohorts.R*
      - outputs to database
    - *req_info.R*
      - study-specific metadata
  - *specs*:
    - *lupus_inclusion.csv:*
      - codeset for lupus patients
  - *site*:
    - site-specific info to run code
  - *reporting*:
    - *Report.Rmd* for report generation

PEDSnet
A Pediatric Learning Health System

# Lupus Query Example: Function in Cohorts.R

```r
#' Get initial cohort of all person_ids with diagnoses
#' specified in codeset and associated condition_start_dates
#'
#' @param codeset_name String which specifies the name of the codeset with diagnosis codes
#' Defaults to 'lupus_inclusion'
#' @param data_tbl Table in which to find all person_ids with diagnoses.
#' specified in codeset and associated condition_start_dates.
#' Defaults to cdm_tbl('condition_occurrence).
#' @param visits_inc a vector that narrows down visits of interest based on
#' visit_concept_id. Defaults to c(9201,9202,9203,2000000048,2000000088).
#'
#' @return tbl with initial cohort (all person_ids with diagnoses
#' specified in codeset and associated condition_start_dates)
get_initial_cohort <- function(codeset_name = 'lupus_inclusion',
                               data_tbl = cdm_tbl('condition_occurrence'),
                               visits_inc = c(9201,9202,9203,2000000048,2000000088)){

  codeset <- load_codeset(codeset_name)

  lupus_pts <-
    data_tbl %>%
    inner_join(codeset,
               by = c("condition_concept_id" = "concept_id")) %>%
    distinct(person_id, visit_occurrence_id, condition_concept_id,
             condition_type_concept_id, condition_start_date)

  filtered_visits <-
    select(cdm_tbl('visit_occurrence'), visit_occurrence_id,
           visit_start_date, visit_concept_id) %>%
    inner_join(lupus_pts, by = 'visit_occurrence_id') %>%
    filter(visit_concept_id %in% visits_inc)
}
```

*function from util_tbl to refer to CDM database table*

*function from codeset.R reading a csv file from specs*

Code Documentation:
- objective stated
- parameters named
- output defined

Intermediate Table
- identifies all visits in database with any lupus diagnosis

PEDSnet
A Pediatric Learning Health System

# Lupus Query Example: Part of .run() function in driver.R

```
.run  <- function(base_dir = config('base_dir')) {

  lupus_summary <- init_sum(cohort = 'Start', persons = 0)

  #' PREPARING BROAD COHORT
  message('Finding all patients with a lupus inclusion code')
  lupus_pts_all <- get_initial_cohort() %>%
    output_tbl(name = 'init_pts',
               file = NA,
               indexes = list('person_id',
                              'condition_concept_id'))

  #' PREPARING SPECIFIC COHORTS
  message('Computing alg_a1_setup')
  alg_a1_setup <- get_alg_a1(filtered_visits = lupus_pts_all,
                             setup_tbl = TRUE) %>%
    output_tbl(name = 'alg_a1_setup',
               file = NA,
               indexes = list('person_id'))

  lupus_summary <- append_sum(cohort = 'Algorithm A Setup with no filters',
                              persons = distinct_ct(alg_a1_setup))

  message('Computing alg_a1_cohort')
  alg_a1_cohort <- get_alg_a1(filtered_visits = lupus_pts_all,
                              setup_tbl = FALSE) %>%
    output_tbl(name = 'alg_a1_cohort',
               file = NA,
               indexes = list('person_id'))

  lupus_summary <- append_sum(cohort = 'Algorithm A Cohort Table',
                              persons = distinct_ct(alg_a1_cohort))
```

*function from util_cohort to begin computing times for each step in query*

*function from cohorts.R (intermediate table)*

*messages to output (or log) to track progress*

*function from util_tbl.R to output data to database*

PEDSnet
A Pediatric Learning Health System

# Lupus Query Example: Codeset Example

lupus_inclusion.csv

```
concept_id,concept_code,concept_name,vocabulary_id
4055640,196138005,Lung disease with systemic lupus erythematosus,SNOMED
4057084,19682006,Lupus hepatitis,SNOMED
4344158,239887007,Systemic lupus erythematosus with organ/system involvement,SNOMED
4346976,239889005,Bullous systemic lupus erythematosus,SNOMED
4344400,239890001,Systemic lupus erythematosus with multisystem involvement,SNOMED
4101469,25380002,Pericarditis secondary to systemic lupus erythematosus,SNOMED
4149913,309762007,Systemic lupus erythematosus with pericarditis,SNOMED
4269448,36471000,Dilated cardiomyopathy secondary to systemic lupus erythematosus,SNOMED
4295179,403486000,Acute systemic lupus erythematosus,SNOMED
4296502,403487009,Fulminating systemic lupus erythematosus,SNOMED
4301051,403488004,Systemic lupus erythematosus of childhood,SNOMED
4217054,417303004,Retinal vasculitis due to systemic lupus erythematosus,SNOMED
44784527,698694005,Systemic lupus erythematosus in remission,SNOMED
```

- *named lupus_inclusion.csv and called in load_codeset()*
- *contains default column types (iccc)*

PEDSnet
A Pediatric Learning Health System

# Lupus Query Example: site_info.R

```r
#' Please alter to reflect your site name.
config('site', 'DCC')

#' Code to establish a database connection at your site.
#' @md
#'
#' The connection must be able to reach CDM data, the vocabularies,
#' and any result schemata needed.  The connection may be either a
#' dplyr-style src_foo() object (as below), or a DBI-style dbConnect()
#' object (e.g. dbConnect(MySQL(), host = ...)).
config('db_src', {
  # If using Oracle, the following are required before loading
  # ROracle, if these are not set in the global environment:
  # Sys.setenv(TZ=Sys.timezone())
  # Sys.setenv(ORA_SDTZ=Sys.timezone())
  require(Argos);
  require(DBI);
  db <- src_argos('argos_v32');
          (db, 'set role          ')
  db
})

#' Name of the schema, if any, to be prepended to CDM fact table names.
#' @md
#'
#' If `NA`, no schema qualifier is added.
config('cdm_schema', 'dcc_pedsnet')

#' Name of the schema, if any, to be prepended to vocabulary tables.
#' @md
#'
#' If `NA`, no schema qualifier is added.
config('vocabulary_schema', 'vocabulary')
```

site information

- connection information
- uses Argos package
- uses v32 of CDM

- cdm and vocabulary schemas specified

PEDSnet
A Pediatric Learning Health System

# Lupus Query Example: run.R

```r
#' Path to the top-level directory of this request package
#' @md
#'
#' This path should point to the top-level directory of the unzipped
#' data request package.  It will be the parent directory of the
#' directory containing this file.  You can replace the value below
#' with a simple path string.  If you don't the program will make a
#' guess about where the package is located, and will stop with an
#' error if it can't.
base_dir <- '~/lupus_nephrology/lupus_algorithm_refinement'
```

location of package locally

```r
#' This Boolean value can be used to override the `default_retain_intermediates`
#' setting from site_info.R.  If it is NA, the default is not overridden.
config('retain_intermediates', FALSE)

#' Request-specific results schema
#' @md
#'
#' Specific schema to which you want intermediate tables to be written
#' during processing of the request.
#' If it is `NA`, the default value from site_info.R is used.
config('results_schema', 'lupus_alg_hr')
```

- does not retain intermediate (temporary) tables
- specifies output schema

```r
################################################################
#
# End of site-specific code.  Please do not edit below this point
#
################################################################

#' Request-specific debug output for database operations
#' @md
#'
#' This Boolean value specifies whether the query log should include
#' detailed information about execution of SQL queries in the database
config('db_trace', FALSE)
```

- turn off debug output for database queries

PEDSnet
A Pediatric Learning Health System

# Lupus Query Example: Report generation setup

```r
```{r setup, include = FALSE}
knitr::opts_chunk$set(echo = TRUE)
require(Argos)
require(dplyr)
require(knitr)
require(ggplot2)
library(VennDiagram)
#library(tidyverse)
library(knitr)

source('../code/config.R')
source('../code/req_info.R')
source('../site/site_info.R')
source('../code/setup.R')
source('../code/codesets.R')
source('../code/util_cohort.R')
source('../code/util_tbl.R')
source('../code/cohorts.R')

base_dir <- '..'
```
```

- packages to run query
- sources all the functions needed to work with the database and convenience functions
- does not require running the entire algorithm (i.e., omits driver.R and run.R)
- working directory is the reporting directory

```r
```{r collect_data, echo = FALSE, message = FALSE}
alg_a1 <- results_tbl('alg_a1_cohort        ') %>%
                      collect() %>%
  distinct(person_id)

alg_b <- results_tbl('alg_b_cohort        ') %>%
  collect() %>%
  distinct(person_id)
```
```

- database contains output
- collects tables and works locally

PEDSnet
A Pediatric Learning Health System

# Lupus Query Example: Report generation markdown example

````
## Comparison of ALGORITHM A1 and ALGORITHM B

The table and the venn diagram below provide a summary of the number of unique patients who meet the definition of lupus according to the two algorithms as well as how these groups of patients overlap.

**Name** | **Total number of unique patients**
------- | ---------------
ALGORITHM A1 |  `r nrow(alg_a1)`
ALGORITHM B |  `r nrow(alg_b)`
ALGORITHM A1 **and** B | `r nrow(alg_a1_alg_b_overlap)`
ALGORITHM A1 **or** B | `r nrow(alg_a1_alg_b_overlap) + (nrow(alg_a1) - nrow(alg_a1_alg_b_overlap)) + (nrow(alg_b) - nrow(alg_a1_alg_b_overlap))`
ALGORITHM A1 **and not** B | `r nrow(alg_a1) - nrow(alg_a1_alg_b_overlap)`
ALGORITHM B **and not** A1 | `r nrow(alg_b) - nrow(alg_a1_alg_b_overlap)`

```{r venn_diagram_a1_b, echo = FALSE, message = FALSE, fig.width = 3, fig.height = 3, fig.align = 'center'}
venn <- draw.pairwise.venn(nrow(alg_a1), nrow(alg_b), nrow(alg_a1_alg_b_overlap),
                           category = c("ALGORITHM A1",
                                        "ALGORITHM B"),
                           cex = rep(0.8, 3),
                           lwd = 0.5,
                           lty = rep("solid", 1),
                           fontfamily = rep("sans", 3),
                           fill = c("blue", "red"),
                           alpha = rep(0.2, 2),
                           cat.dist = c(0.05, 0.05),
                           cat.pos = c(-25, 25),
                           cat.cex = rep(0.7, 2),
                           cat.fontfamily = rep("sans", 2)
                           )
grid.draw(venn)
```
````

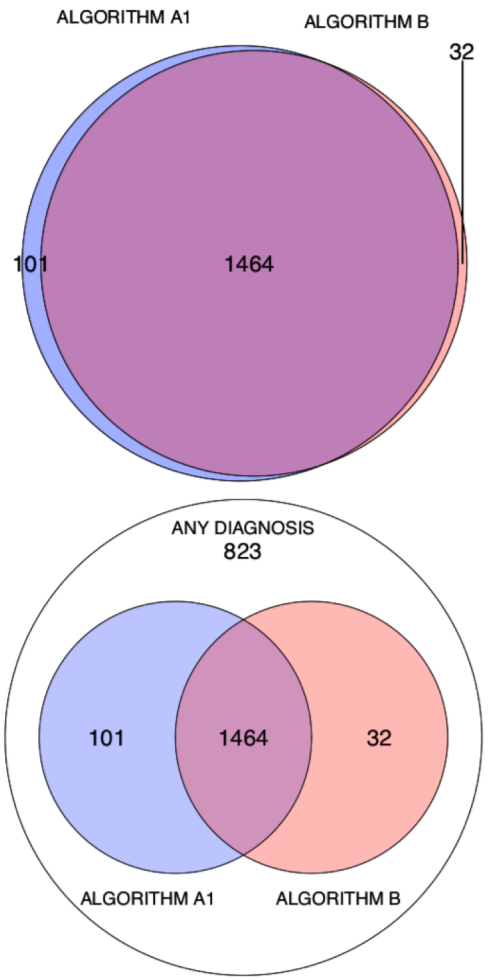
PEDSnet
A Pediatric Learning Health System

# Lupus Query Example: Part of Report Output

### Comparison of ALGORITHM A1 and ALGORITHM B

The table and the venn diagram below provide a summary of the number of unique patients who meet the definition of lupus according to the two algorithms as well as how these groups of patients overlap.

| Name | Total number of unique patients |
|---|---|
| ALGORITHM A1 | 1565 |
| ALGORITHM B | 1496 |
| ALGORITHM A1 **and** B | 1464 |
| ALGORITHM A1 **or** B | 1597 |
| ALGORITHM A1 **and not** B | 101 |
| ALGORITHM B **and not** A1 | 32 |




PEDSnet
A Pediatric Learning Health System