ClearPath_iter_7

July 3, 2024

1 ==========

2 = Iteration #7 =

3 ==========

3.1 Stage 1: Planning

The tasks for this stage are: 1. Compose the project team 2. Set the research question 3. Schedule all review meeting to ensure iterations are time-boxed.

3.1.1 7.1.1. Compose the project team

The project team and Clinical Review Board are CMI and CB.

3.1.2 7.1.2. Set the research question

The research question is:

How does changing the inclusion criteria affect the output of the sequence-mining investigation? Specifically, I only include patient diagnosed before 2000, and I only view their sequences after 2010.

The new requirements specified from the previous iteration are:

1. Change the inclusion criteria to only include patient diagnosed before 2000, and I only view their sequences after 2010.
2. Remove the need for metformin to be the first prescription.

3.1.3 7.1.3. Schedule all review meeting to ensure iterations are time-boxed

CMI and CB will meet in their regular Thursday-morning meeting, which will now be fortnightly.

3.2 Stage 2: Extraction

This tasks for this stage are: 1. Gather knowledge and insight into the processes under study and the data-generating mechanisms. 2. Obtain data for processing

3.2.1 7.2.1. Gather knowledge and insight into the processes under study and the data-generating mechanisms.

Regarding the processes under study, I, the modeller, discuss the project during weekly meetings with an experienced GP who is a professor of primary medical care. Regarding data-generating

mechanisms, I, the modeller, have gathered knowledge from over half a decade experience collaborating with clinicians and patients on research projects about electronic healthcare records.

3.2.2 7.2.2. Obtain data for processing

Data have been obtain by agreement via Connected Bradford. Data are queryable via this Jupyter notebook on the GoogleCloudPlatform using AI Vertex Workbench.

Select which portions of the notebook to run.

Install R packages.

```
Installing package into ‘/home/jupyter/.R/library’
(as ‘lib’ is unspecified)
```

Set and load requisites.

```
Warning message:
"<BigQueryConnection> uses an old dbplyr interface
  Please install a newer version of the package or contact the
maintainer
This warning is displayed once every 8 hours."
```

Define study cohort.

This was trivial in previous iterations because I relied on clinicall-coded diagnoses, only. This time I will also permit diagnosis to be indicated by abnormal HbA1c (i.e. >48 mmol/mol). The justification is that concurrent work by CMI and colleagues shows that clinicall-coded dates of diabetes diagnosis disagree with HbA1c values by more than 10 years. The HbA1c values are surperior indicators but the validity of HbA1c values decreases as we go further back in the record because they weren't used diagnostically until a while after introduction. Therefore, in the early days, we will have to trust the clinically-coded date.

In an email sent 26th July 2024, CB suggested a three-option algorthin for identifying the date of diagnosis: 1. If the clinically-coded diagnosis is before April 2004 AND there are raised HbA1c values after April 2003 (note the difference in years), then use the clinically-coded date. - The actual statement from CB was: if the clinically-coded diagnosis is before April 2004 AND some "recent" raised HbA1c, then use the clinically-coded date, on the assumption that the date is not miscoded. This requries a threshold for "recent" (or does he mean "concurrent"?). 2. If the clinically-coded diagnosis is after April 2004 AND there are raised HbA1c values before April 2003 (note the difference in years), then use the earliest raised HbA1c date. 3. If the clinically-coded diagnosis is after April 2004 AND the first raised HbA1c value is after April 2003 (note the difference in years), then use the clinically-coded date.

My approach to identifying the date of diagnosis will be to start with the original method of using clinical codes, and then only change the date if it satisfies option #2, above.

First, I need to convert historic A1c% values to up-to-date mmol/mol values (Formala taken from https://ebmcalc.com/GlycemicAssessment.htm). Then I will define the cohort of relevant records.

Retrieve dates of prescriptions in the follow-up period.

Retrieve dates of tests in the follow-up period.

Create a variable to indicate the date from which we no longer observe the patient in the record.

Append the prescriptions and tests dataframe logs.

Retrieve indication of diagnoses used in the calculation of comorbidity.

In this iteration, I will use the Mayo Clinic's definition* of two or more of 19 specified conditions, where diabetes is one, separated by 30 days (Rocca et al. (2014)). This means that I only need to identify patients that have at least one of the diagnostic codes in `codes_SNOMED_all_multimorbidity_diagnoses` 30 days before or after their diagnosis for Type 2 Diabetes Mellitus.

*Rocca et al. (2014) used ICD-10 codes but I use SNOMED-CT codes.

3.3 Stage 3: Data processing

This tasks for this stage are: 1. Assess data quality 2. Format data for study

3.3.1 7.3.1. Assess data quality

I will not assess data quality for this illustrative example. Proper project should always assess data quality, perhaps using Weiskopf and Weng's 3X3 DQA.

3.3.2 7.3.2. Format data for study

I will use the `TraMineR` package in R for sequence pattern mining. The data needs to be in the format of state-sequence object. The easiest way to create the state-sequence object is to use the SQL-extracted time series to define a time-series object, which can then be converted into an STS object using `TraMineRextras::TSE_to_STS()`.

Now, I create the simplified dataframe object from the first iteration (i.e. the SQL-extracted dataframe to only focus on a handful of medications because this is only an example).

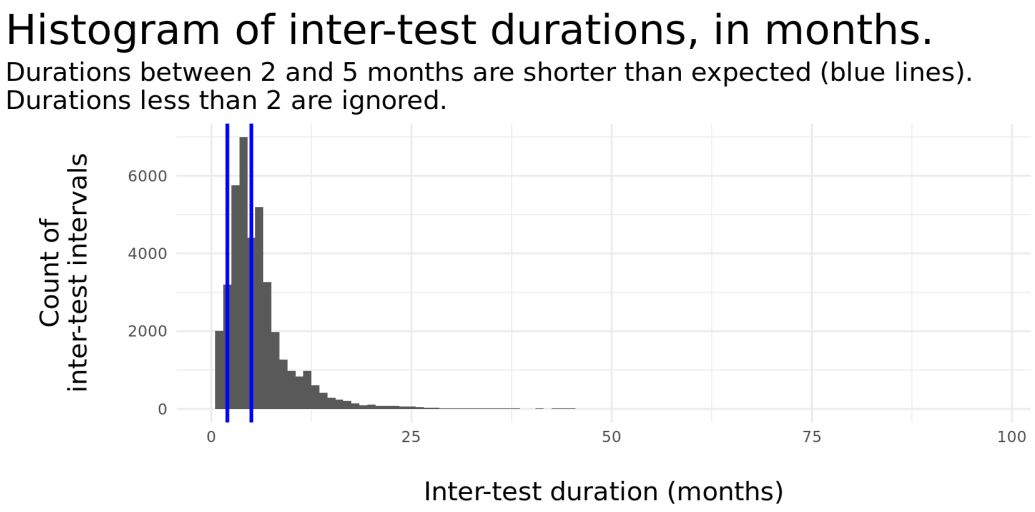
Before I can beging analysing the data, I need to define the strata proposed by the Clinical Review Board. The stratifications were: - H.M.A.: Four strata defined by combinations of {'Expected', 'Shorter-than-expected'} testing intervals and {'No observed change', 'Observed change'} in prescriptions. The H.M.A. acronym derives from the three strata: (0,0)-Hold; (1,0)-Monitor; (0 or 1, 1)-Adjust. - Tests-and-Interventions: Twelve strata defined by combinations of the test statuses and whether the patient is on one, two or three medications simulatneously. - Multimorbidity: Two strata defined by whether or not there is a record for at least one of the multimorbidity diagnositic codes.

The first thing I do is to add a variable that indicates the patient-specific test interval. This will be handy for bounding the variables I need to create.

H.M.A. stratification

The two components of this stratification are the testing interval and the change in prescription. The testing interval requries me to create a variable indicating that the inter-test duration was between `val_testing_interval_LB` and `val_testing_interval_UB`. The change in prescription requires me to create a variable indicating whether the prescriptions before and after an index test event are the same.

The first thing is to calculate the inter-test duration. In this iteration, I will have to ignore the interval between the diagnostic test and the subsequent test because I am deliberately looking at records ten years after the diagnostic test.



Next, I need to make the variable that indicates whether the prescription has changed since the previous interval.

The final step is to combine the two components of the stratification's definition into a single variable called HMA.

HMA and Test Status stratification

First, I add a column indicating the most-recent test value.

Next, I create the variable indicating the combination of HMA category and most-recent test status value.

Finally, I create a reference table indicating the HMA-and-Test Status stratification. The possible values are:

A data.frame: 13 × 3

Value <fct>	HMA.component <chr>	Test.component <chr>
Hold Red	Hold	Test Status = Red
Hold Amber	Hold	Test Status = Amber
Hold Yellow	Hold	Test Status = Yellow
Hold Green	Hold	Test Status = Green
Monitor Red	Monitor	Test Status = Red
Monitor Amber	Monitor	Test Status = Amber
Monitor Yellow	Monitor	Test Status = Yellow
Monitor Green	Monitor	Test Status = Green
Adjust Red	Adjust	Test Status = Red
Adjust Amber	Adjust	Test Status = Amber
Adjust Yellow	Adjust	Test Status = Yellow
Adjust Green	Adjust	Test Status = Green
Unobserved	Unobserved	Unobserved

T-and-I stratification

The two components of this stratification are the test status, T, and the degree of intervention in the previous inter-test interval, I. The test status is already encoded in the `test_status_rollover` variable. The degree of intervention will, in this diabetes case study, require me to count the unique medication names in the `event_value` column that exists between testing events.

Firstly, I add a column that indicates the count of unique medications prescribed in inter-test intervals.

Finally, I create the variable indicating the Tests-and-Interventions stratification by combining the test status with the variable indicating the count of unique medications prescribed in inter-test intervals (i.e. with `n_meds_per_test_interval`). The possible values of the Tests-and-Interventions stratification variable are:

A data.frame: 17 × 3

Value <fct>	Test.component <chr>	Intervention.component <chr>
Red Zero Rx	Test Status = Red	Zero
Red One Rx	Test Status = Red	One
Red Two Rx	Test Status = Red	Two
Red More Rx	Test Status = Red	> Two
Amber Zero Rx	Test Status = Amber	Zero
Amber One Rx	Test Status = Amber	One
Amber Two Rx	Test Status = Amber	Two
Amber More Rx	Test Status = Amber	> Two
Yellow Zero Rx	Test Status = Yellow	Zero
Yellow One Rx	Test Status = Yellow	One
Yellow Two Rx	Test Status = Yellow	Two
Yellow More Rx	Test Status = Yellow	> Two
Green Zero Rx	Test Status = Green	Zero
Green One Rx	Test Status = Green	One
Green Two Rx	Test Status = Green	Two
Green More Rx	Test Status = Green	> Two
Unobserved	Unobserved	Unobserved

Multimorbidity stratification

The strata are defined by whether or not there is a record for at least one of the multimorbidity diagnositic codes. Patients with a record for at least one of the multimorbidity diagnositic codes have already been identified in `qry_log_multimorb_longFormat`. This BigQuery query result needs 'collecting' and joining to `df_log_PandT_longFormat_simplified_StrataLabels`. Then, I need to create the indicator variable with values of `Multimorbid = FALSE` before the `date_multimorb` and `Multimorbid = TRUE` on or after `date_multimorb`.

I also include a stratification called `TandMultiMorb` that, like `TandI`, combines the test status values with the multimorbidity values. This stratification has eight levels with an extra for errors.

Finally, I create the variable indicating the Tests-and-Multimorbidity stratification by combining the test status with the variable indicating multimorbidity. The possible values of the Tests-and-Multimorbidity stratification variable are:

A data.frame: 9×3

Value <fct>	Test.component <chr>	Multimorbidity.component <chr>
Red Multimorbid	Test Status = Red	Multimorbid
Amber Multimorbid	Test Status = Amber	Multimorbid
Yellow Multimorbid	Test Status = Yellow	Multimorbid
Green Multimorbid	Test Status = Green	Multimorbid
Red Not multimorbid	Test Status = Red	Not multimorbid
Amber Not multimorbid	Test Status = Amber	Not multimorbid
Yellow Not multimorbid	Test Status = Yellow	Not multimorbid
Green Not multimorbid	Test Status = Green	Not multimorbid
Unobserved	Unobserved	Unobserved

Create state-sequence objects

Create state-sequence objects for `TraMineR`. In the previous iteration, the Clinical Review Board requested me to separate the test statuses from the prescriptions in the state distribution plot because they help to answer two distinct questions. To achieve this, I will create separate state-sequence objects in addition to the combined one.

Make an test-only state-sequence object for some particular plots.

I actually make two. One contains the "Unobserved" state, which is useful for tracking when sequences stop. The other exlcudes the "Unobserved" state, which is useful for plotting proportions of the remaining states without being distracted by the growing proportion of unobserved events.

Make an intervention-only state-sequence object for some particular plots.

Like the test-only objects, I actually make two. One contains the "Unobserved" state, which is useful for tracking when sequences stop. The other exlcudes the "Unobserved" state, which is useful for plotting proportions of the remaining states without being distracted by the growing proportion of unobserved events.

Make an test-and-intervention status state-sequence object for some particular plots.

Like the test-only objects, I actually make two. One contains the "Unobserved" state, which is useful for tracking when sequences stop. The other exlcudes the "Unobserved" state, which is useful for plotting proportions of the remaining states without being distracted by the growing proportion of unobserved events.

Make an HMA state state-sequence object for some particular plots.

Like the test-only objects, I actually make two. One contains the "Unobserved" state, which is useful for tracking when sequences stop. The other exlcudes the "Unobserved" state, which is useful for plotting proportions of the remaining states without being distracted by the growing proportion of unobserved events.

Make an HMA-and-Test status state-sequence object for some particular plots.

Like the test-only objects, I actually make two. One contains the "Unobserved" state, which is useful for tracking when sequences stop. The other exlcudes the "Unobserved" state, which is useful for plotting proportions of the remaining states without being distracted by the growing proportion of unobserved events.

Make a test-and-multimorbidity status state-sequence object for some particular plots.

Like the test-only objects, I actually make two. One contains the "Unobserved" state, which is useful for tracking when sequences stop. The other exlcudes the "Unobserved" state, which is useful for plotting proportions of the remaining states without being distracted by the growing proportion of unobserved events.

3.4 Stage 4: Mining and analysis

This tasks for this stage are: 1. Discover / Mine process models 2. Build simulation models 3. Design and test model evaluation rig 4. Set up and/or update the evidence template

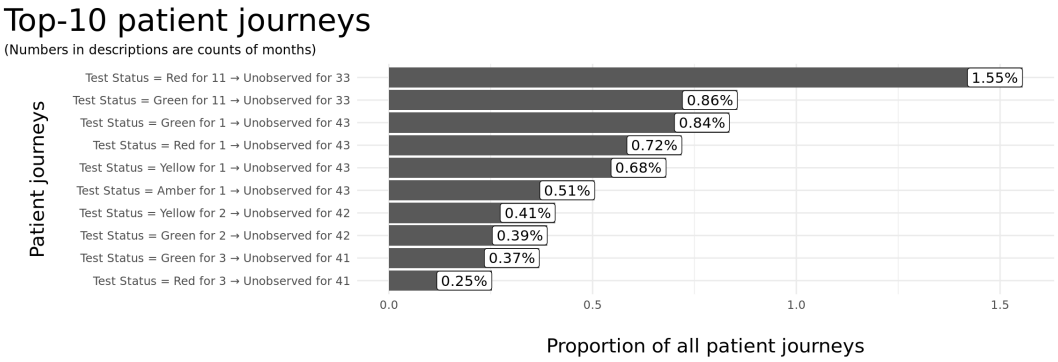
3.4.1 7.4.1 Discover / Mine process models

This iterations analysis will focus on various data-processing steps for 'trajectory' mining without using process-mining or sequence-pattern mining R packages. Instead, I will only use `TraMineR`.

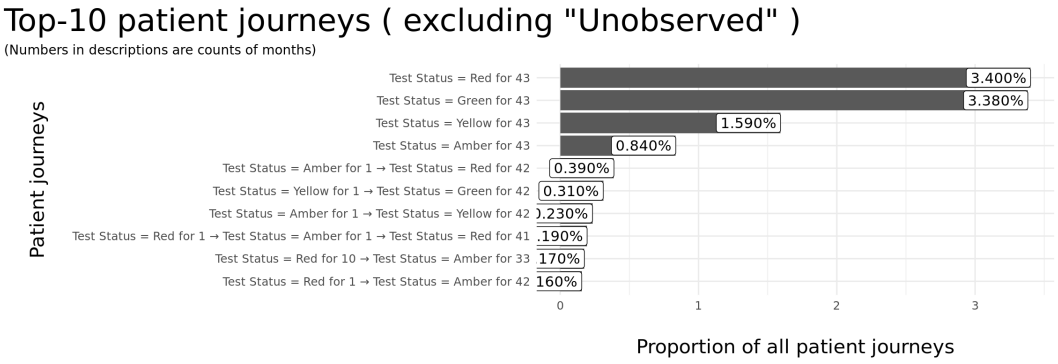
Plot the top-10 most-frequent patient journeys.

A data.frame: 10 × 4

sequence <chr>	Freq <dbl>	Percent <dbl>	cum_sum_percent <dbl>
Test Status = Red for 11 → END	80	1.5549077	1.554908
Test Status = Green for 11 → END	44	0.8551992	2.410107
Test Status = Green for 1 → END	43	0.8357629	3.245870
Test Status = Red for 1 → END	37	0.7191448	3.965015
Test Status = Yellow for 1 → END	35	0.6802721	4.645287
Test Status = Amber for 1 → END	26	0.5053450	5.150632
Test Status = Yellow for 2 → END	21	0.4081633	5.558795
Test Status = Green for 2 → END	20	0.3887269	5.947522
Test Status = Green for 3 → END	19	0.3692906	6.316812
Test Status = Red for 3 → END	13	0.2526725	6.569485

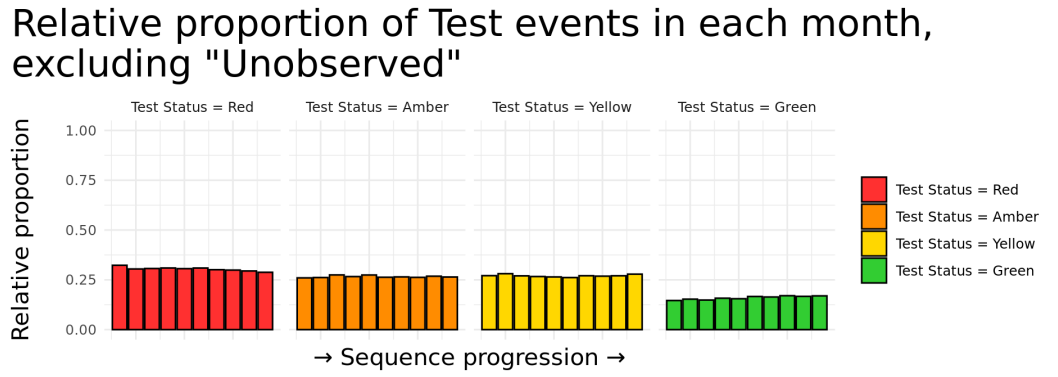


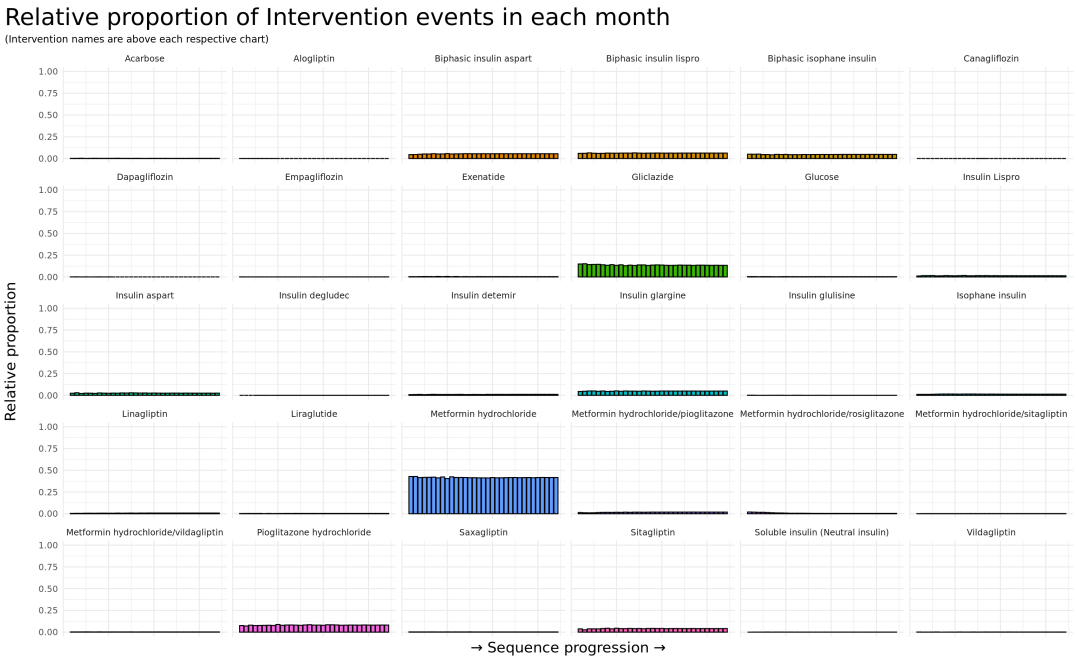
Note: The top-10 patient journeys only account for approximately 7% of all sequences.



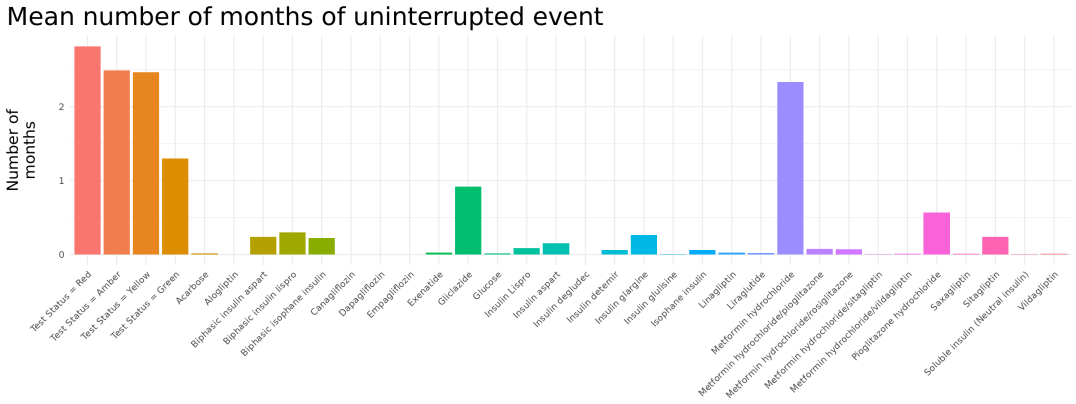
Note: The top-10 patient journeys only account for approximately 11% of all sequences.

Below are the month-by-month relative propotions of each event.





Average duration that a patient spends in a "state" without changing.



Some initial observations of the process-mining investigation: - **The most-common sequence patterns (when the "Unobserved" values are ignored) are dominated by long stretches of the same test-status values.** - Evidenced by the percentage contribution of patient journeys in the top 10. - **Unlike when we studied sequences from diagnosis, the counts of test-status values stay steady over time** - Evidenced by the level counts of test status values.

Burden of Treatment / Turbulence / Complexity of care Calculate statistics that summarise and represent the complicatedness / complexity of patients' sequences, then plot as histograms.

Component

- The **Transitions Count** statistic is the count of times a patient changes state during their sequence. It has an unbounded range and does not take the length of the sequence into account (i.e. low counts can indicate short sequences with little time to change state, or can indicate a long and stable sequence).
- The $\mathbf{log_2}(SubsequenceCount)$ statistic is the log_2 of the count of 'distinct successive states' during a patient's sequence. It has an unbounded range with a sequence-specific maximum reached when the sequence cycles between all its states. Low values can indicate either few changes or a small number of unique states observed in the sequence.
- The **Longitudinal Entropy** statistic quantifies the entropy of the distribution of durations spent in each state observed in the sequence. It is a measure of diversity of states within a sequence. It's range is between 0 and 1 because each sequence's value is scaled to the theoretical maximum of log_a, where a is the count of unique states across all sequences, even those not observed in the sequence (i.e. its alphabet).

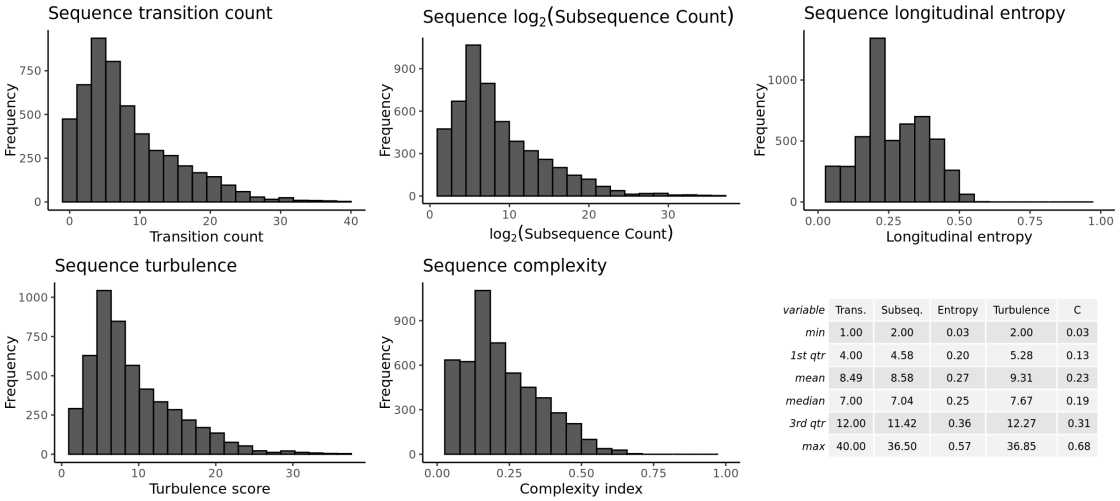
Composite

- The **Turbulence** statistic is the log_2 of the product of the count of subsequences and a sequence-specific scaling parameter (Elzinga et al. 2007). The sequence-specific scaling parameter is based on the variance of observed state durations (which we know to be inversely proportional to the notion of turbulent sequences). Larger values are associated with sequences that have many states, changes, and similar durations in each state. This statistic differs from the $\mathbf{log_2}(SubsequenceCount)$ only by the sequence-specific scaling parameter within the log_2 operator.
- The **Complexity Index** statistic is the geometric mean of a scaled count of transitions in a sequence and the sequence's scaled longitudinal entropy (Gabadinho et al. 2010). The count of transitions is scaled to the length of the sequence, and the longitundinal entropy is scaled to the theoretical maximum of log_a, where a is the count of unique states across all sequences, even those not observed in the sequence (i.e. its alphabet). It's range is between 0 and 1.

Below I plot the distributions of each statistics, for various sets of states (e.g. tests only or interventions only). Upon request from the Clinical Review Board, I also provide plots stratifed where hte disrtibutinos are stratified by patient records' multimorbidity status (Green = Not multimorbidd; Red = Multimorbid).

Distributions of sequence-complexity statistics, across all patient records, using all states.

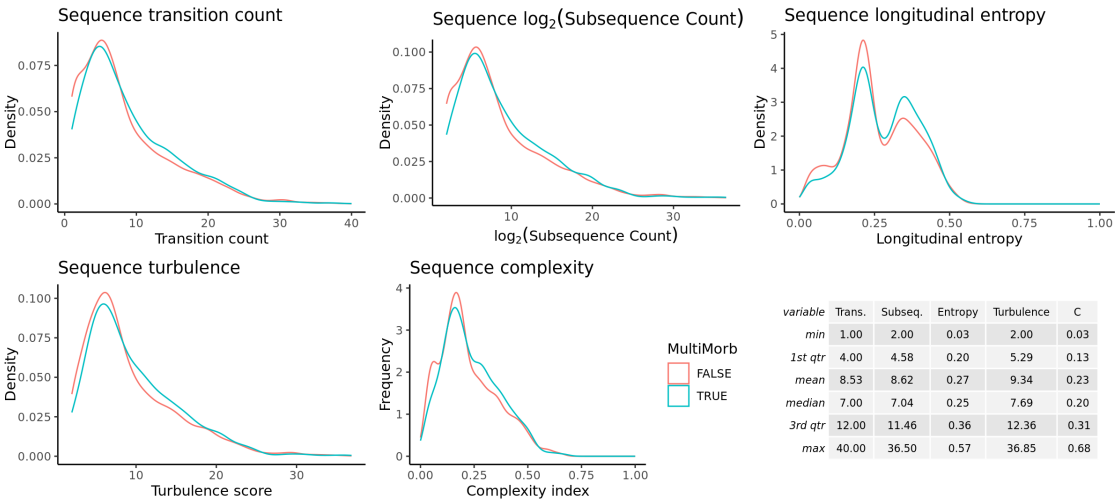
The state-sequence alphabet is made of 35 states.



variable	Trans.	Subseq.	Entropy	Turbulence	C
min	1.00	2.00	0.03	2.00	0.03
1st qtr	4.00	4.58	0.20	5.28	0.13
mean	8.49	8.58	0.27	9.31	0.23
median	7.00	7.04	0.25	7.67	0.19
3rd qtr	12.00	11.42	0.36	12.27	0.31
max	40.00	36.50	0.57	36.85	0.68

Smoothed densities of sequence-complexity statistics, across all patient records, using all states, stratified by multimorbidity status.

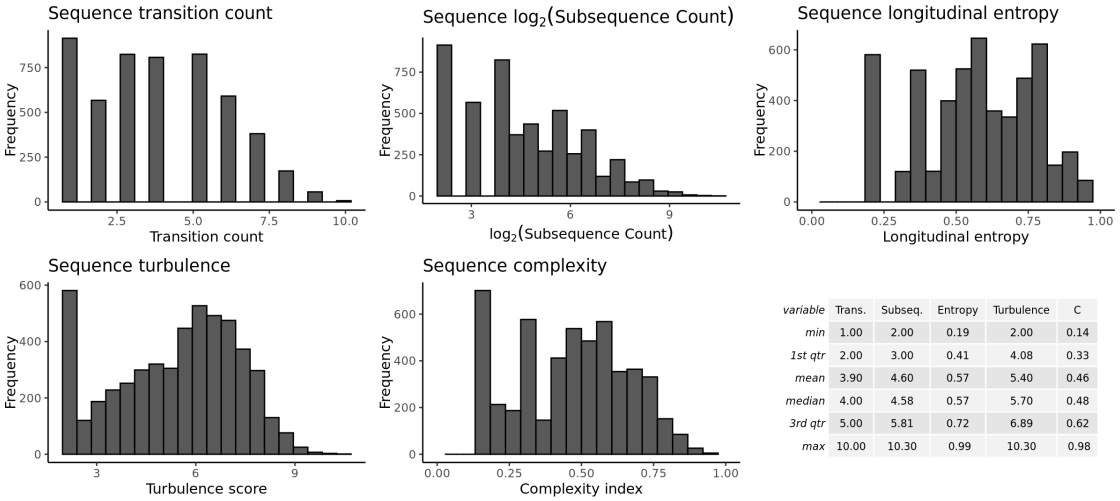
The state-sequence alphabet is made of 35 states.



variable	Trans.	Subseq.	Entropy	Turbulence	C
min	1.00	2.00	0.03	2.00	0.03
1st qtr	4.00	4.58	0.20	5.29	0.13
mean	8.53	8.62	0.27	9.34	0.23
median	7.00	7.04	0.25	7.69	0.20
3rd qtr	12.00	11.46	0.36	12.36	0.31
max	40.00	36.50	0.57	36.85	0.68

Distribution of sequence-complexity statistics, across all patient records, using test states only.

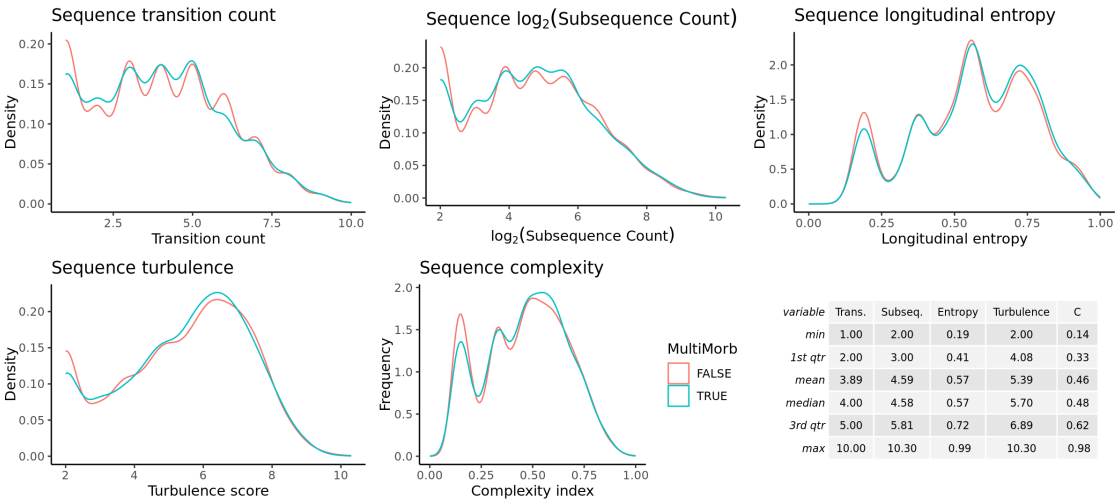
The state-sequence alphabet is made of 5 states.



variable	Trans.	Subseq.	Entropy	Turbulence	C
min	1.00	2.00	0.19	2.00	0.14
1st qtr	2.00	3.00	0.41	4.08	0.33
mean	3.90	4.60	0.57	5.40	0.46
median	4.00	4.58	0.57	5.70	0.48
3rd qtr	5.00	5.81	0.72	6.89	0.62
max	10.00	10.30	0.99	10.30	0.98

Smoothed densities of sequence-complexity statistics, across all patient records, using test states only, stratified by multimorbidity status.

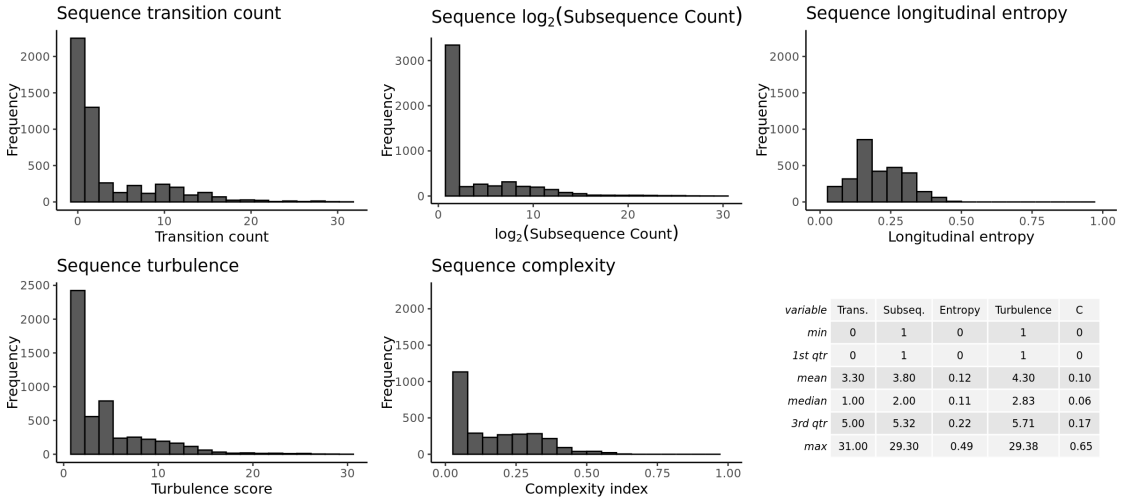
The state-sequence alphabet is made of 5 states.



variable	Trans.	Subseq.	Entropy	Turbulence	C
min	1.00	2.00	0.19	2.00	0.14
1st qtr	2.00	3.00	0.41	4.08	0.33
mean	3.89	4.59	0.57	5.39	0.46
median	4.00	4.58	0.57	5.70	0.48
3rd qtr	5.00	5.81	0.72	6.89	0.62
max	10.00	10.30	0.99	10.30	0.98

Distributions of sequence-complexity statistics, across all patient records, using intervention states only.

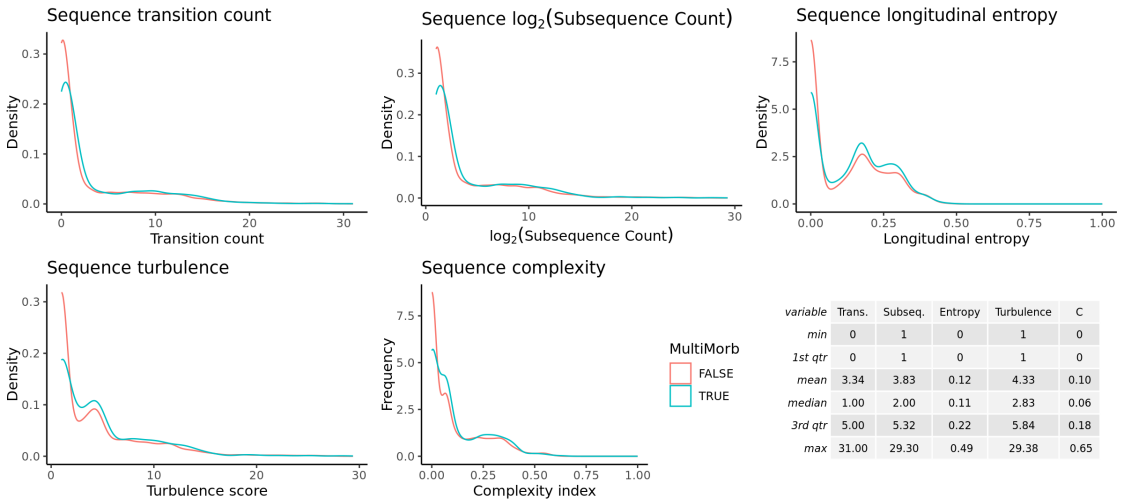
The state-sequence alphabet is made of 31 states.



variable	Trans.	Subseq.	Entropy	Turbulence	C
min	0	1	0	1	0
1st qtr	0	1	0	1	0
mean	3.30	3.80	0.12	4.30	0.10
median	1.00	2.00	0.11	2.83	0.06
3rd qtr	5.00	5.32	0.22	5.71	0.17
max	31.00	29.30	0.49	29.38	0.65

Smoothed densities of sequence-complexity statistics, across all patient records, using intervention states only, stratified by multimorbidity status.

The state-sequence alphabet is made of 31 states.



variable	Trans.	Subseq.	Entropy	Turbulence	C
min	0	1	0	1	0
1st qtr	0	1	0	1	0
mean	3.34	3.83	0.12	4.33	0.10
median	1.00	2.00	0.11	2.83	0.06
3rd qtr	5.00	5.32	0.22	5.84	0.18
max	31.00	29.30	0.49	29.38	0.65

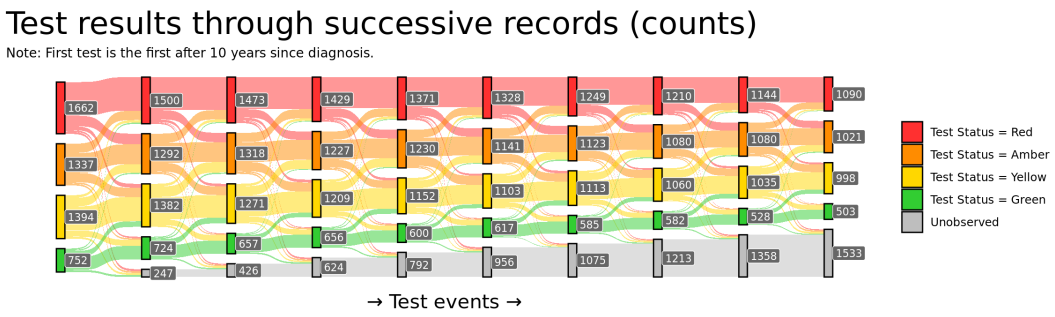
Some initial observations of the distribution plots: - Compared to the previous iteration where we looked at sequences from diagnosis, these sequences 10 years after diagnosis show less complexity. - Evidenced by a left shift in all histograms, for the plots pretaining to all events. - Compared to the previous iteration where we looked at sequences from diagnosis, the complexity of test-status sequences is spread throughout the range, showing greater complexity, on average. - Evidenced

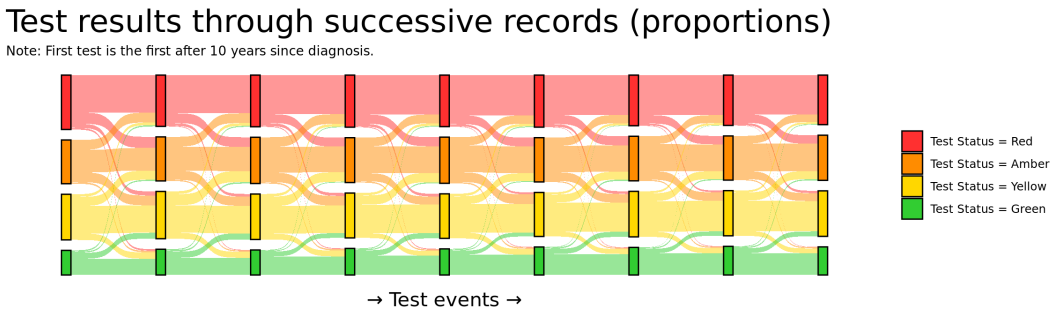
by a right shift in all histograms, for the plots pretaining to test-status events. - Compared to the previous iteration where we looked at sequences from diagnosis, the complexity of intervention sequences is similar.

3.4.2 Sankey plots

Note: The Sankey plots show data that are in STate Sequence format. This means that: 1. the number of sequence steps is decided by the maximum number of sequence steps observed across the dataset, *and* 2. a state remains unchanged in the sequence by default, rather than records being lost to follow-up.

Test Statuses First, I show a basic Sankey plot showing patients' test results at each testing event and visualise the proportion of patient records that switch between test statuses.

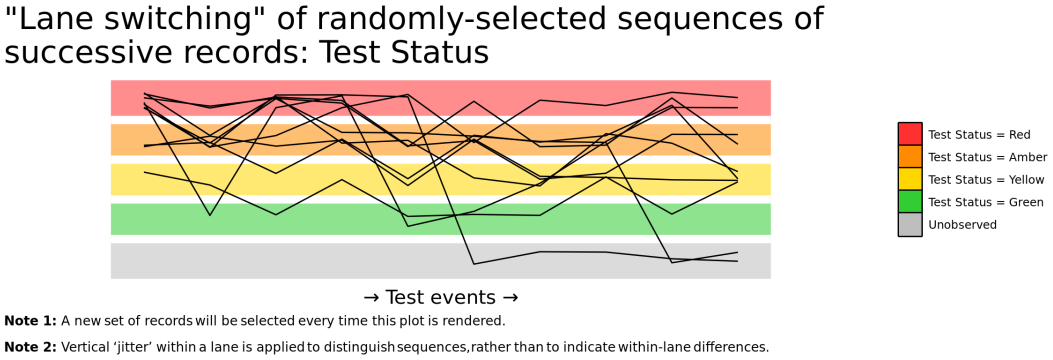




Some initial observations from the Test Statuses Sankey diagram: 1. **The overall count of patient records in each level of Test Status is approximately stable.** - Evidenced by little variation in the height of the vertical bars, over time. 2. **Patient records indicating a particular Test Status level predominantly stay in that Test Status level.** - Evidenced by the largest ribbon joining subsequent vertical bars almost exclusively comes from the the same Test Status level. 3. **The Test Status levels that change between every test event rarely change by more than one level.** - Evidenced by many ribbons linking vertically-adjacent bars at subsequent

test points. 4. **Unlike the previous iteration where we looked at sequences from diagosis, these sequences from 10 years after diagnosis show an unchanging probability of changing test-status values between tests.** - Evidenced by a consistent size of ribbons joining different colours, as we move from left to right in the graphic.

In the meeting of the wider project team at the end of the previous iteration, there was a request to view how individual records progress between the strata of the variable (a.k.a. the lanes) and across the events. The plot below shows the progression of a randomly-selected group of records. The random selection is taken every time the plot is rendered, and the count of records being selected can be set with the `n_records_to_sample` pareameter at the start of the code block. Note that he veritcal 'jitter' within a lane is applied to distinguish sequences, rather than to indicate within-lane differences.



Note 1: A new set of records will be selected every time this plot is rendered.

Note 2: Vertical 'jitter' within a lane is applied to distinguish sequences, rather than to indicate within-lane differences.

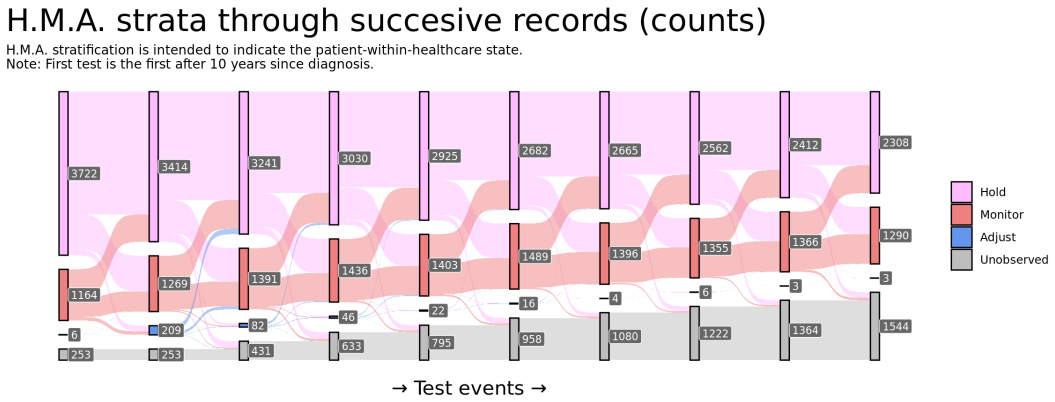
A data.frame: 10 × 4

sequence <chr>	Freq <dbl>	Percent <dbl>	cu <d
Test Status = Red for 10 → END	228	4.4314869	4.4
Test Status = Green for 10 → END	74	1.4382896	5.8
Test Status = Red for 1 → END	67	1.3022352	7.1
Test Status = Green for 1 → END	63	1.2244898	8.3
Test Status = Yellow for 1 → END	63	1.2244898	9.6
Test Status = Amber for 1 → END	54	1.0495627	10.
Test Status = Red for 9 → Test Status = Amber for 1 → END	37	0.7191448	11.
Test Status = Yellow for 2 → END	32	0.6219631	12.
Test Status = Red for 3 → END	29	0.5636540	12.
Test Status = Green for 2 → END	28	0.5442177	13.

H.M.A. strata In the next plot, the Sankey plot visualises the proportions of patient records moving between strata of the H.M.A. stratification.

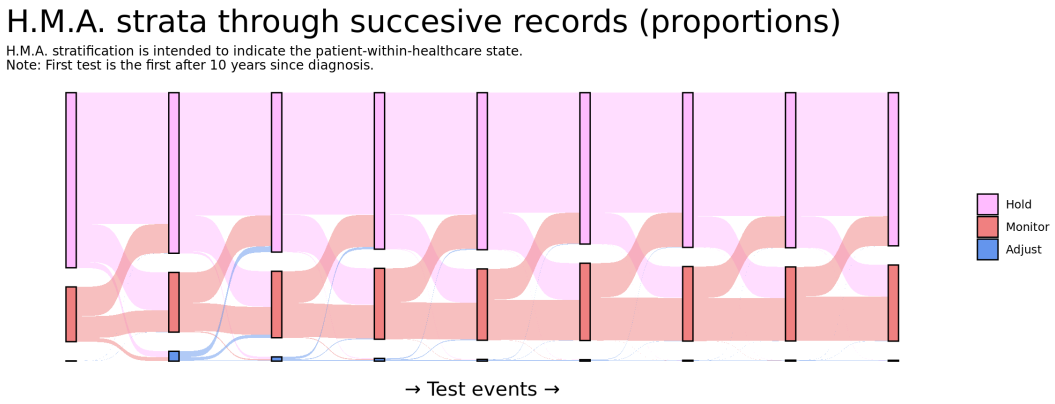
H.M.A. stratification is something CB proposed. It has four strata infomed by two components: the testing interval since the previous test {'Expected', 'Shorter-than-expected'}, and the change in prescriptions compared with the previous testing interval {'No observed change', 'Observed change'}. The stratification gets its name from the four strata: (0,0)-Run; (1,0)-Monitor; (0 or 1,1)-Adjust.

The first task is to create a new state-sequence object that tracks H.M.A. strata.



The H.M.A. strata are:

- **Hold**, for which the testing interval since the previous test is 'Expected' and there has been 'No observed change' in prescriptions.
- **Monitor**, for which the testing interval since the previous test is 'Shorter than expected' and there has been 'No observed change' in prescriptions.
- **Adjust**, for which the testing interval since the previous test is 'Expected' and there has been an 'Observed change' in prescriptions.

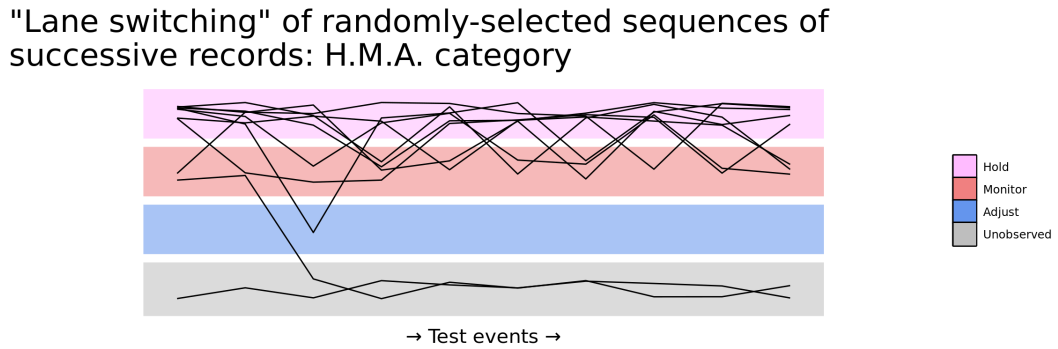


The H.M.A. strata are:

- **Hold**, for which the testing interval since the previous test is 'Expected' and there has been 'No observed change' in prescriptions.
- **Monitor**, for which the testing interval since the previous test is 'Shorter than expected' and there has been 'No observed change' in prescriptions.
- **Adjust**, for which the testing interval since the previous test is 'Expected' and there has been an 'Observed change' in prescriptions.

Some initial observations from the H.M.A. Sankey diagram: 1. **Most patient records indicate a 'Hold' strategy at first, but the 'Monitor' strategy prevails for those records that are longer.** - Evidenced by the 'Hold' vertical bars being the largest, initially, but the 'Monitor' vertical bars being largest, at the end. 2. **Very few patient records indicate the undesirable category of 'Adjust'.** - Evidenced by the barely visible vertical bars that represent the counts of patients in the 'Adjust' state. 3. **Unlike the previous iteration where we looked at sequences from diagosis, these sequences from 10 years after diagnosis show an unchanging probability of changing HMA-status values between tests.** - Evidenced by a consistent size of ribbons joining different colours, as we move from left to right in the graphic.

In the meeting of the wider project team at the end of the previous iteration, there was a request to view how individual records progress between the strata of the variable (a.k.a. the lanes) and across the events. The plot below shows the progression of a randomly-selected group of records. The random selection is taken every time the plot is rendered, and the count of records being selected can be set with the `n_records_to_sample` pareameter at the start of the code block. Note that he veritcal 'jitter' within a lane is applied to distinguish sequences, rather than to indicate within-lane differences.



Note 1: A new set of records will be selected every time this plot is rendered.

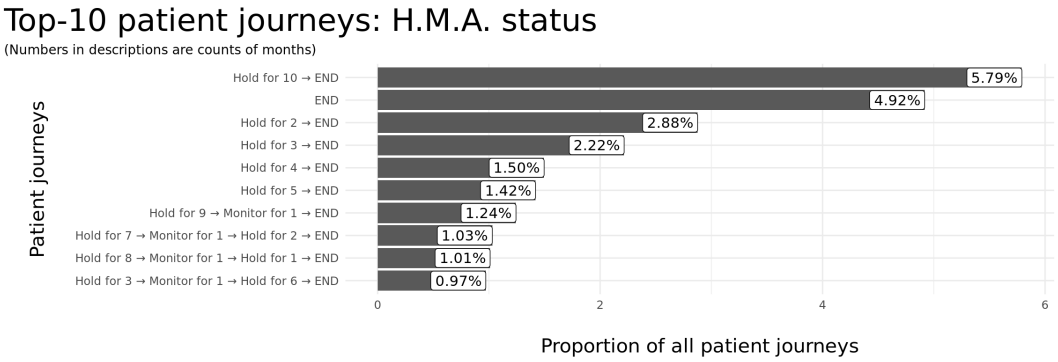
Note 2: Vertical 'jitter' within a lane is applied to distinguish sequences, rather than to indicate within-lane differences.

The H.M.A. strata are:

- **Hold**, for which the testing interval since the previous test is 'Expected' and there has been 'No observed change' in prescriptions.
- **Monitor**, for which the testing interval since the previous test is 'Shorter than expected' and there has been 'No observed change' in prescriptions.
- **Adjust**, for which the testing interval since the previous test is 'Expected' and there has been an 'Observed change' in prescriptions.

A data.frame: 10 × 4

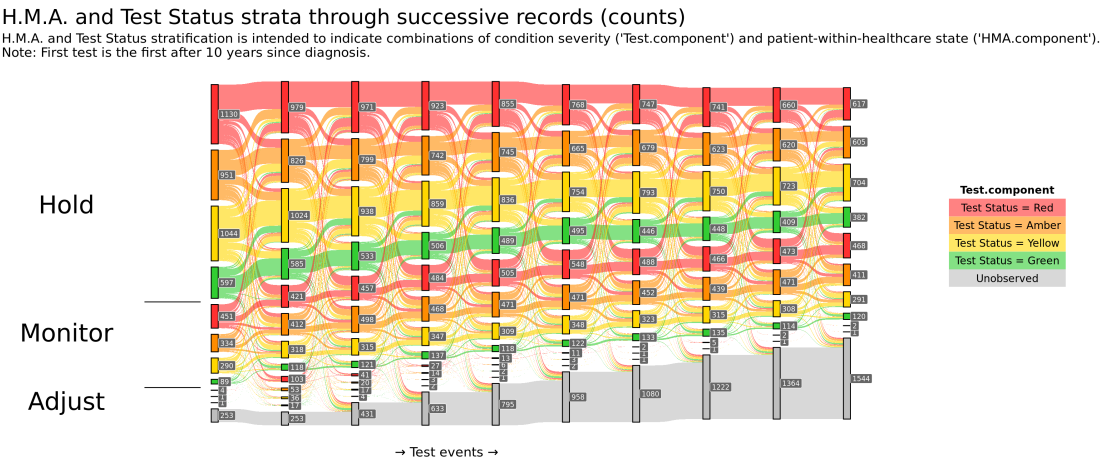
sequence <chr>	Freq <dbl>	Percent <dbl>	cum_sum_perce <dbl>
Hold for 10 → END	298	5.7920311	5.792031
END	253	4.9173955	10.709427
Hold for 2 → END	148	2.8765792	13.586006
Hold for 3 → END	114	2.2157434	15.801749
Hold for 4 → END	77	1.4965986	17.298348
Hold for 5 → END	73	1.4188533	18.717201
Hold for 9 → Monitor for 1 → END	64	1.2439261	19.961127
Hold for 7 → Monitor for 1 → Hold for 2 → END	53	1.0301263	20.991254
Hold for 8 → Monitor for 1 → Hold for 1 → END	52	1.0106900	22.001944
Hold for 3 → Monitor for 1 → Hold for 6 → END	50	0.9718173	22.973761

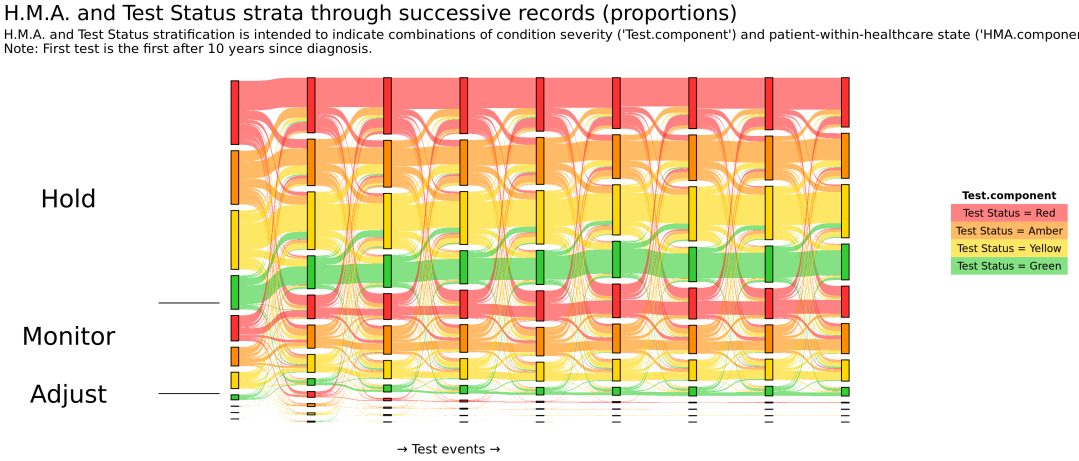


Note: The top-10 patient journeys only account for approximately 23% of all H.M.A. sequences.

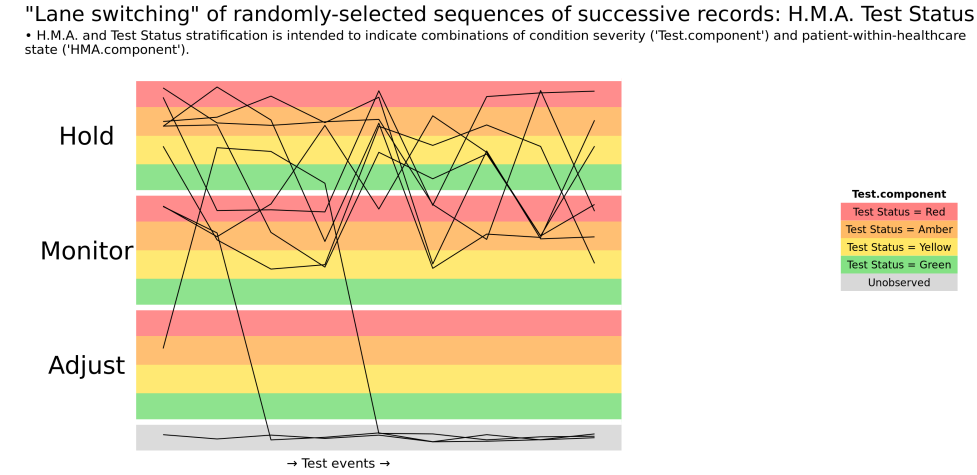
H.M.A. and Test Status strata In the next plot, the Sankey plot visualises the proportions of patient records moving through test statues and H.M.A. category.

The first task is to create a new state-sequence object that tracks `HMAandTestStatus` strata.





Some initial observations from the H.M.A. and Test Status Sankey diagram: 1. **Most patient records indicating 'Hold' or 'Monitor' have a Yellow test status.** - Evidenced by the largest bars in the 'Hold' and 'Monitor' states being yelloe, throughout. 2. **Very few patient records indicate the undesirable category of 'Adjust'.** - Evidenced by the barely visible vertical bars that represent the counts of patients in the 'Adjust' state. 3. **Unlike the previous iteration where we looked at sequences from diagosis, these sequences from 10 years after diagnosis show an unchanging probability of changing test-status values between tests.** - Evidenced by a consistent size of ribbons joining different colours, as we move from left to right in the graphic.



Note 1: A new set of records will be selected every time this plot is rendered.

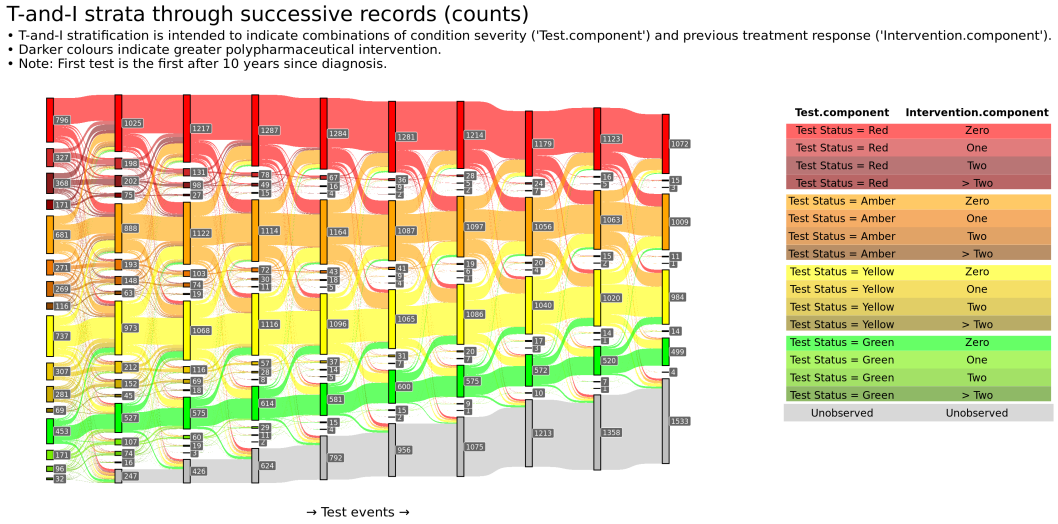
Note 2: Vertical 'jitter' within a lane is applied to distinguish sequences, rather than to indicate within-lane differences.

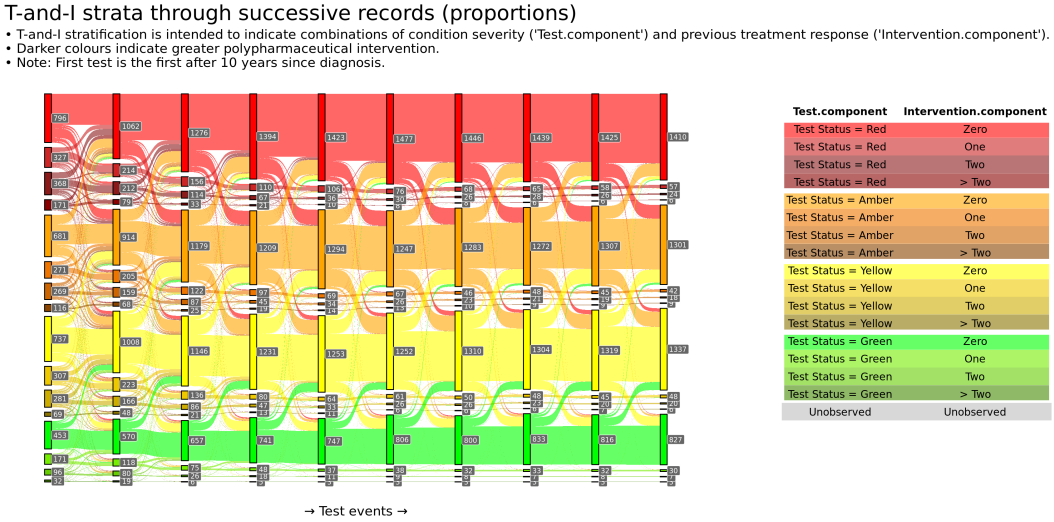
A data.frame: 10 × 4

sequence <chr>	Freq <dbl>	Percent <dbl>	cum_sum_percent <dbl>
END	253	4.9173955	4.917396
Hold Green for 10 → END	32	0.6219631	5.539359
Hold Green for 2 → END	26	0.5053450	6.044704
Hold Yellow for 2 → END	24	0.4664723	6.511176
Hold Green for 3 → END	20	0.3887269	6.899903
Hold Red for 2 → END	16	0.3109815	7.210884
Hold Amber for 2 → END	15	0.2915452	7.502430
Hold Yellow for 1 → Hold Green for 1 → END	13	0.2526725	7.755102
Hold Red for 3 → END	11	0.2137998	7.968902
Hold Amber for 1 → Hold Red for 1 → END	10	0.1943635	8.163265

T-and-I strata In the next plot, the Sankey plot visualises the proportions of patient records moving between strata of the T-and-I stratification.

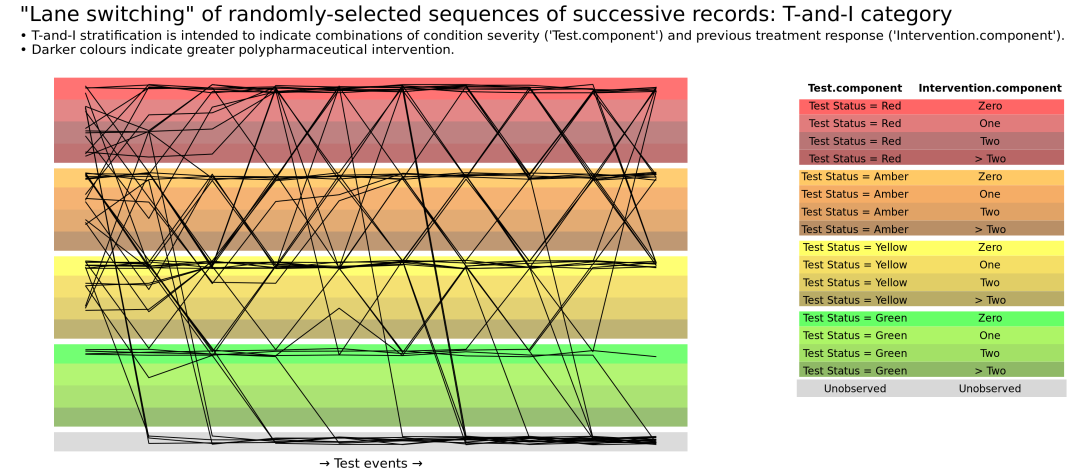
The T-and-I stratification is something CB proposed. It has 17 strata derived from combining the test status, T, {Test status = Red, Test status = Amber, Test status = Yellow, Test status = Green} with a variable indicating the count of unique medications prescribed in previous inter-test intervals, P, {0, 1, 2, 3}, i.e. poly-pharmacy. One additional strata is designated for where errors arise in the data.





Some initial observations from the T-and-I Sankey diagram: 1. **Unlike the previous iteration where we looked at sequences from diagosis, these sequences from 10 years after diagnosis show an increase the proportion of records indicating a given test status with no prescription.** - Evidenced by a gradual increase in the size of ribbons joining the same colours, as we move from left to right in the graphic.

In the meeting of the wider project team at the end of the previous iteration, there was a request to view how individual records progress between the strata of the variable (a.k.a. the lanes) and across the events. The plot below shows the progression of a randomly-selected group of records. The random selection is taken every time the plot is rendered, and the count of records being selected can be set with the `n_records_to_sample` pareameter at the start of the code block. Note that he veritcal 'jitter' within a lane is applied to distinguish sequences, rather than to indicate within-lane differences.



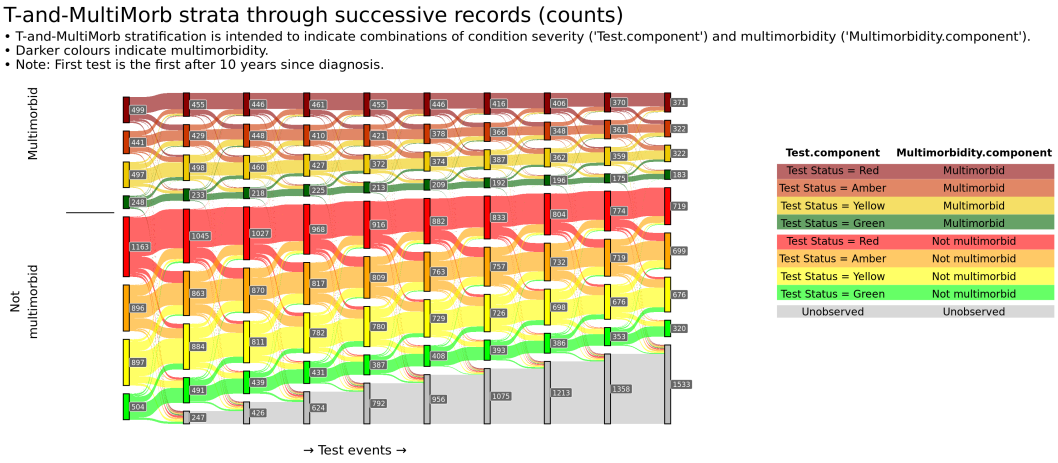
Note 1: A new set of records will be selected every time this plot is rendered.

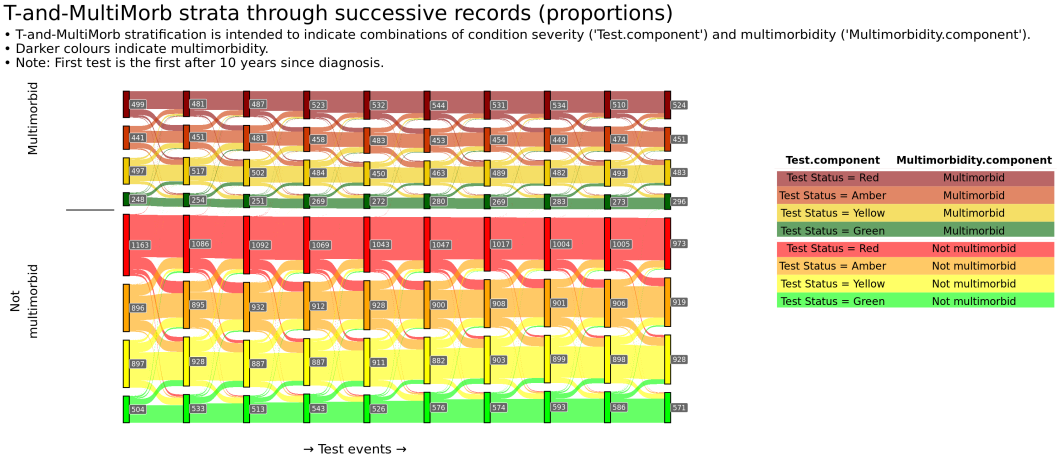
Note 2: Vertical 'jitter' within a lane is applied to distinguish sequences, rather than to indicate within-lane differences.

A data.frame: 10 × 4

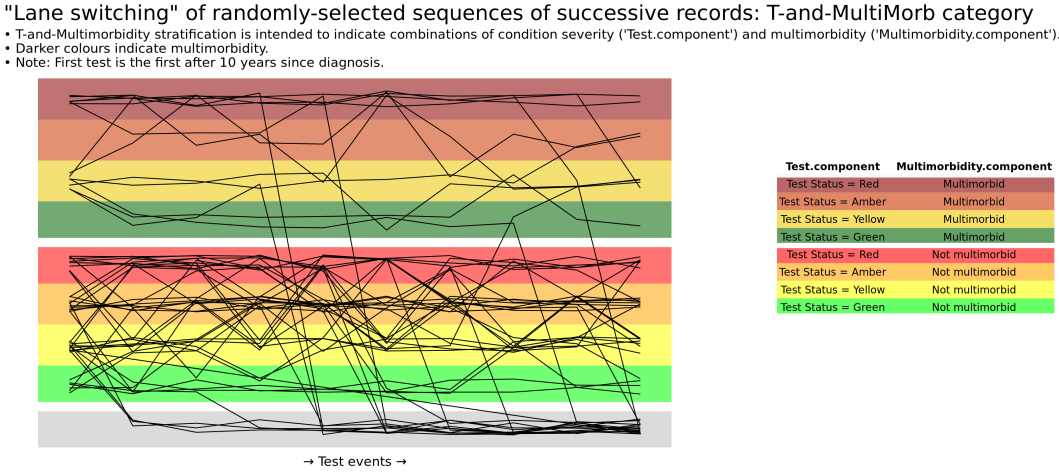
sequence <chr>	Freq <dbl>	Percent <dbl>	cum_sum_perce <dbl>
Red Zero Rx for 10 → END	95	1.8464529	1.846453
Green Zero Rx for 10 → END	52	1.0106900	2.857143
Green Zero Rx for 1 → END	43	0.8357629	3.692906
Red Zero Rx for 1 → END	37	0.7191448	4.412051
Yellow Zero Rx for 1 → END	35	0.6802721	5.092323
Amber Zero Rx for 1 → END	26	0.5053450	5.597668
Red Two Rx for 1 → Red Zero Rx for 9 → END	25	0.4859086	6.083576
Red More Rx for 1 → Red Zero Rx for 9 → END	24	0.4664723	6.550049
Yellow Zero Rx for 2 → END	21	0.4081633	6.958212
Green Zero Rx for 2 → END	20	0.3887269	7.346939

Multimorbidity strata In the next plot, the Sankey plot visualises the proportions of patient records moving from no multimorbidity to multimorbidity.





Some initial observations from the T-and-I Sankey diagram: 1. **Unlike the previous iteration where we looked at sequences from diagosis, these sequences from 10 years after diagnosis show the greatest proportion of records have a Red test status while not being multimorbid (as opposed to yellow and multimorbid).**



Note 1: A new set of records will be selected every time this plot is rendered.

Note 2: Vertical 'jitter' within a lane is applied to distinguish sequences, rather than to indicate within-lane differences.

A data.frame: 10 × 4

sequence <chr>	Freq <dbl>	Percent <dbl>	cum_sum_percent <dbl>
Red Not multimorbid for 10 → END	167	3.2458698	3.245870
Red Multimorbid for 10 → END	55	1.0689990	4.314869
Green Not multimorbid for 10 → END	53	1.0301263	5.344995
Yellow Not multimorbid for 1 → END	44	0.8551992	6.200194
Green Not multimorbid for 1 → END	42	0.8163265	7.016521
Red Not multimorbid for 1 → END	41	0.7968902	7.813411
Amber Not multimorbid for 1 → END	32	0.6219631	8.435374
Red Multimorbid for 1 → END	26	0.5053450	8.940719
Amber Multimorbid for 1 → END	22	0.4275996	9.368319
Green Not multimorbid for 2 → END	22	0.4275996	9.795918

3.4.3 7.4.2 Build simulation models, if applicable

Not applicable for this iteration.

3.4.4 7.4.3 Design and test model evaluation rig

Not applicable for this iteration.

3.4.5 7.4.4 Set up and/or update the evidence template

Not applicable for this iteration.

3.5 Stage 5: Evaluation

This tasks for this stage are: 1. Meet with Clinical Review Board to assess validity. 2. Set requirements for next interation of stages 1-5.

3.5.1 7.5.1. Meet with Clinical Review Board to assess validity

Notes from meeting of the Clinical Review Board on Thursday 6th June: - …

3.5.2 7.5.2 Set requirements for next interation of stages 1-5.

Requirements for the next iteration are: 1. …