CoMap evaluation

October 7, 2015

```python
In [1]: %matplotlib inline
        import os
        import math
        from collections import OrderedDict
        import pandas as pd
        import matplotlib as mpl
        import matplotlib.pyplot as plt
        import seaborn as sns
        import yaml
        from pathlib import Path
        from data import CodesInDbs, Mappings, Databases
        from IPython.display import Latex

        pd.set_option('display.max_colwidth', 100)

        sns.set_style('whitegrid')
        #sns.set_context("poster")
        #plt.rcParams['figure.figsize'] = (4, 3)
        plt.rc("savefig", dpi=150)

        measures_palette = sns.color_palette('Set1', n_colors=2, desat=.5)
        measures_palette.reverse()

        def graded_recall_palette(n_colors):
            palette = sns.color_palette("Blues", n_colors=n_colors, desat=.6)
            palette.reverse()
            return palette

        def graded_precision_palette(n_colors):
            palette = sns.color_palette("Reds", n_colors=n_colors, desat=.6)
            palette.reverse()
            return palette

        def mystyle(palette=None, xrot=0, ha='center', ylim=(0,1), ylabel=None, savefig=None):
            class C:
                def __enter__(self):
                    if palette is not None:
                        palette.__enter__()
                def __exit__(self, exc_type, value, traceback):
                    if palette is not None:
                        palette.__exit__(exc_type, value, traceback)
                    if exc_type is None:
                        sns.despine(left=True)
```

```python
                    plt.grid(False, axis='x')
                    if plt.gca().legend_:
                        lgd=plt.legend(loc=2, bbox_to_anchor=(1, 1))
                    else:
                        lgd=None
                    plt.gca().get_lines()[0].set_visible(False)
                    plt.gca().set_ylim(*ylim)
                    plt.xticks(rotation=xrot, ha=ha)
                    if ylabel is not None:
                        plt.ylabel(ylabel)
                    if savefig:
                        plt.savefig('{}-{}'.format(PROJECT, savefig), bbox_extra_artists=[lgd] if l
            return C()

        pd.set_option('display.notebook_repr_html', True)
        def _repr_latex_(self):
            #return r"\begin{center}%s\end{center}" %
            return self.to_latex()
        pd.DataFrame._repr_latex_ = _repr_latex_   # monkey patch pandas DataFrame

        PROJECT = os.getenv('COMAP_PROJECT')
        print("PROJECT:", PROJECT)

PROJECT: safeguard

In [2]: with open('../projects/{}/variations.yaml'.format(PROJECT)) as f:
            variations = yaml.load(f)

        with open('../projects/{}/config.yaml'.format(PROJECT)) as f:
            config = yaml.load(f)
            databases = Databases.of_config(config)
            coding_systems = config['coding-systems']

        with open('../projects/{}/events.yaml'.format(PROJECT)) as f:
            events = yaml.load(f)
            event_names = {}
            for event in events:
                casedef = yaml.load(open('../projects/{}/case-definitions/{}.yaml'.format(PROJECT, even
                event_names[event] = casedef['name']

        with open('../projects/{}/mappings.yaml'.format(PROJECT)) as f:
            mappings = Mappings.of_raw_data_and_normalize(yaml.load(f), events, databases).normalize(da

        with open('../codes-in-dbs.json') as f:
            codes_in_dbs = CodesInDbs.of_data(json.load(f))

        with open('../{}.code-stats.csv'.format(PROJECT)) as f:
            code_stats = pd.read_csv(f)

        def database_label(database):
            return database
            #return "{} ({})".format(database, databases.coding_system(database))

        def measure_label(measure):
            return {
```

2

```
            "recall": "Sensitivity",
            "precision": "PPV", # "Positive predictive value",
        }[measure]

    def event_label(event):
        return event_names[event]
```

0.1 Load evaluations `ev`

```
In [3]: ev = pd.read_csv('../{}.evaluations.csv'.format(PROJECT))
        ev['variation event database recall precision'.split()].head()
```

Out[3]:

	variation	event	database	recall	precision
0	1-RN-CHD-RB-PAR.expand	is	IPCI	0.666667	1.000000
1	1-RN-CHD-RB-PAR.expand	is	CPRD	0.500000	0.714286
2	1-RN-CHD-RB-PAR.expand	is	GePaRD	0.636364	0.583333
3	1-RN-CHD-RB-PAR.expand	is	Medicare	0.100000	0.200000
4	1-RN-CHD-RB-PAR.expand	hs	IPCI	0.333333	1.000000

1 Notes

Should exclusion codes from the reference be generated? No. Exclusion codes are often added database specifically, where the codes are *not* represented in the case definition.

2 Coding systems

```
In [4]: pd.DataFrame([
            (database, databases.coding_system(database))
            for database in databases.databases()
        ], columns=("Database", "Coding system")).set_index("Database")
```

Out[4]:

|	Coding system
Database	
Medicare	ICD9CM
IPCI	ICPC2EENG
CPRD	RCD2
GePaRD	ICD10CM

3 Baseline

```
In [5]: averages_compare = pd.DataFrame([
            ev[ev.variation == 'baseline'].groupby('database').recall.mean(),
            ev[ev.variation == 'baseline'].groupby('database').precision.mean(),
        ])
        averages_compare.index = averages_compare.index.map(measure_label)
        averages_compare.columns = averages_compare.columns.map(database_label)
        with mystyle(measures_palette, savefig='baseline-performance-by-db.pdf'):
            averages_compare.T.plot(kind='bar', title='Baseline')
```
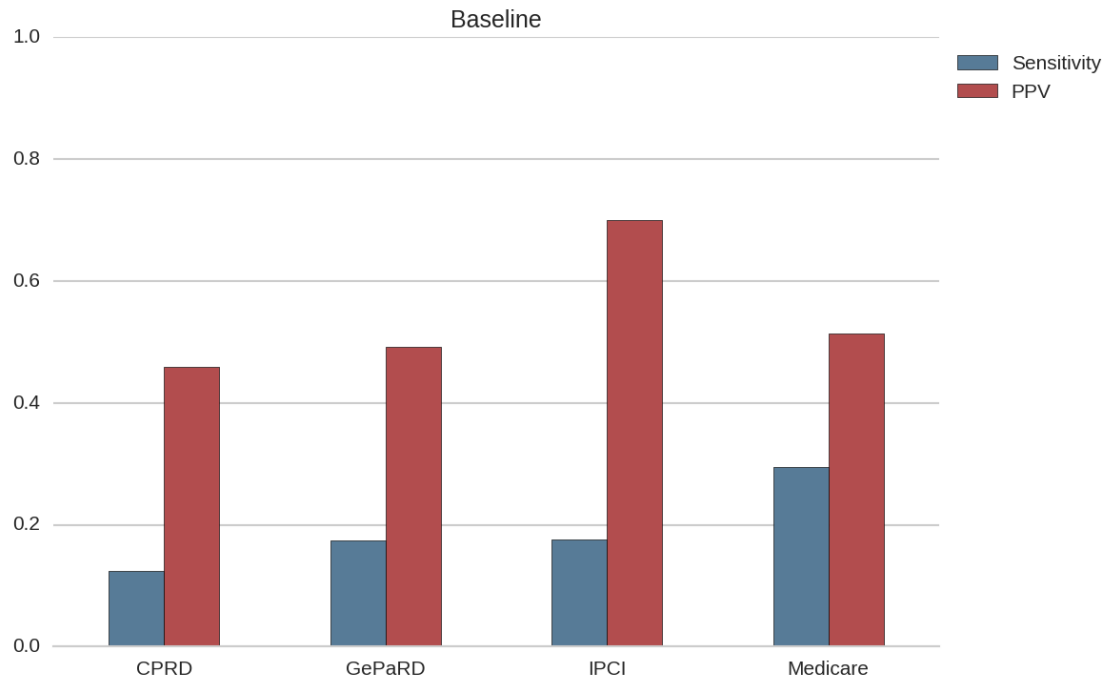
Baseline

```
In [6]: averages_compare = pd.DataFrame([
            ev[ev.variation == 'baseline'].groupby('event').recall.mean(),
            ev[ev.variation == 'baseline'].groupby('event').precision.mean(),
        ])
        averages_compare.index = averages_compare.index.map(measure_label)
        averages_compare.columns = averages_compare.columns.map(event_label)
        with mystyle(measures_palette, xrot=45, ha='right', savefig='baseline-performance-by-event.pdf')
            averages_compare.T.plot(kind='bar', title='Baseline')
```

Baseline
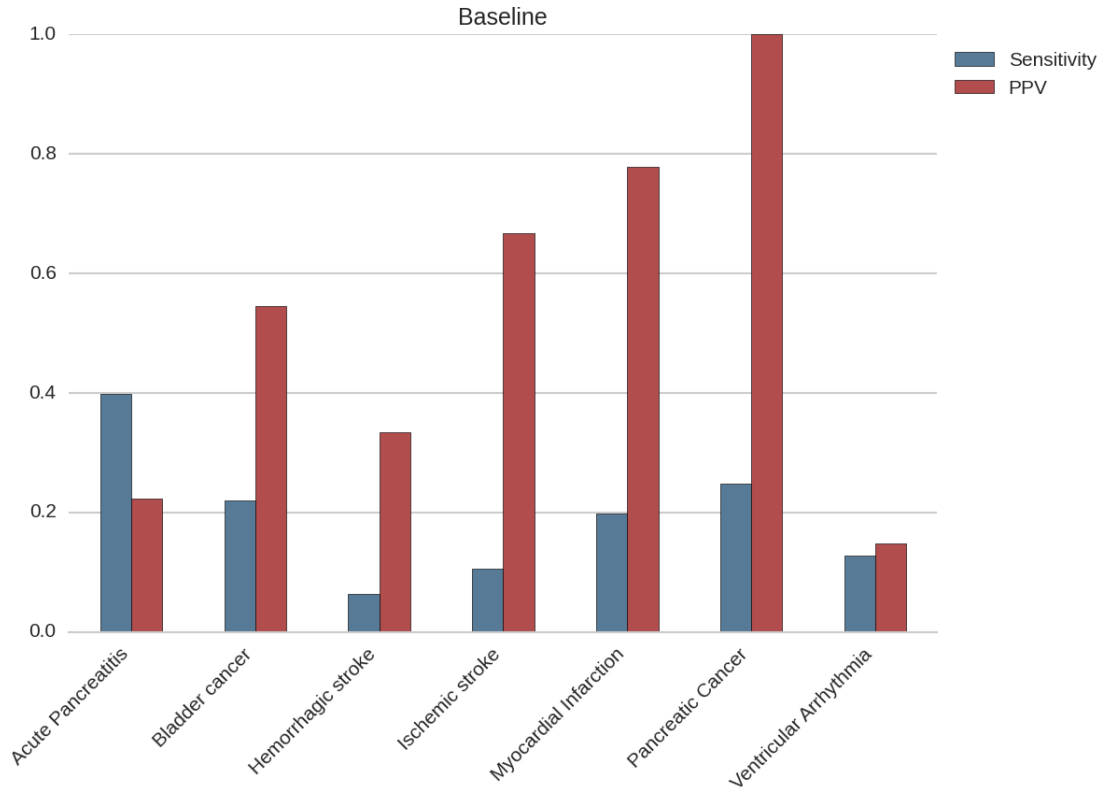
4 Max-recall

```
In [7]: averages_compare = pd.DataFrame([
            ev[ev.variation == 'max-recall'].groupby('database').recall.mean(),
        ])
        averages_compare.index = [measure_label('recall')]
        averages_compare.columns = averages_compare.columns.map(database_label)
        with mystyle(measures_palette, ylim=(.9,1), savefig='max-recall-by-db.pdf'):
            averages_compare.T.plot(kind='bar', legend=False, title='Maximum recall')
            plt.ylabel(measure_label('recall'))
        averages_compare
```

Out[7]:

	CPRD	GePaRD	IPCI	Medicare
Sensitivity	0.929871	0.938033	1	0.988095
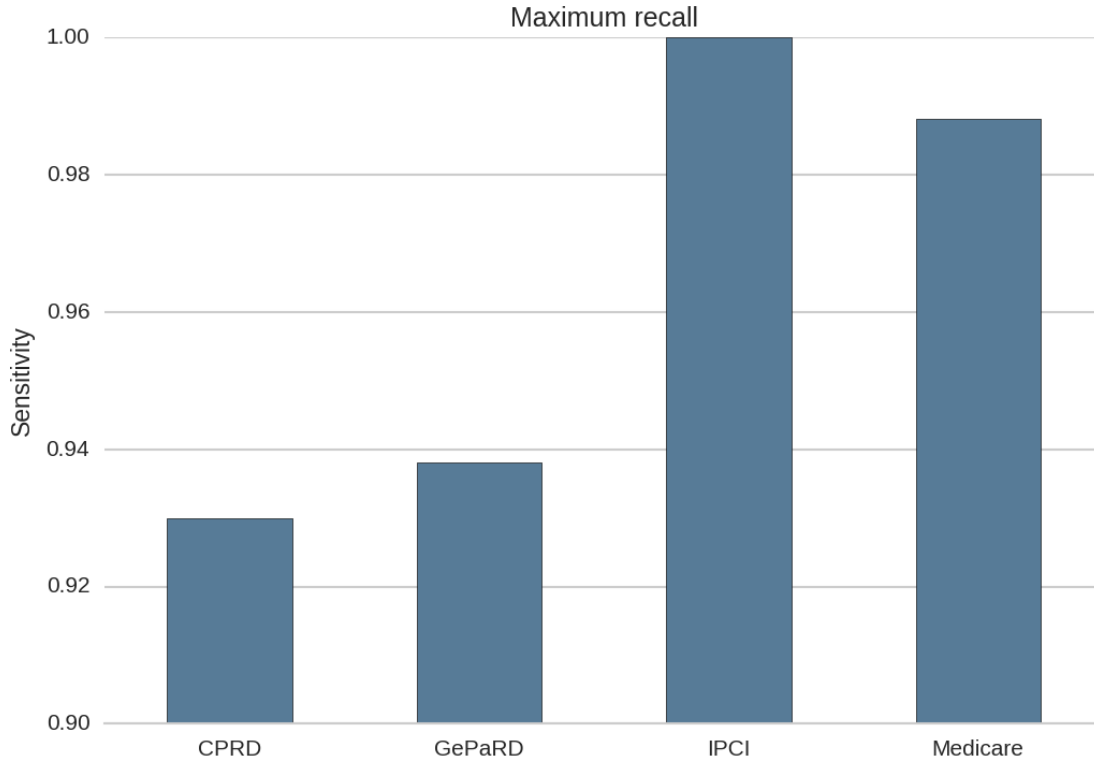
Maximum recall

4.1 Reasons for imperfect sensitivity

```
In [8]: stats = DataFrame()
        stats['In mapping'] = code_stats[code_stats.InMapping]\
            .groupby('Database').Code.count()
        stats['Not in maximum recall'] = code_stats[code_stats.InMapping & ~code_stats.InDnf]\
            .groupby('Database').Code.count()
        stats.fillna(0, inplace=True)
        stats['%'] = (stats['Not in maximum recall'] / stats['In mapping']).map("{:.2%}".format)
        #stats['Not in maximum recall but in database'] = code_stats[code_stats.InMapping & code_stats..
        #    .groupby('Database').Code.count()
        stats.fillna(0, inplace=True)
        stats
```

Out[8]:

	In mapping	Not in maximum recall	%
Database			
CPRD	229	14	6.11%
GePaRD	74	3	4.05%
IPCI	16	0	0.00%
Medicare	53	1	1.89%

```
In [9]: max_recall_fn = ev[(ev.variation == 'max-recall') & (ev.recall < 1)][["database", "fn"]]
        max_recall_fn.database = max_recall_fn.database.map(database_label)
        max_recall_fn.fn = max_recall_fn.fn.fillna('[]').map(json.loads)
```

```python
max_recall_fn = max_recall_fn.groupby('database').fn.sum().to_frame()
max_recall_fn.fn = max_recall_fn.fn.map(set).map(', '.join)
max_recall_fn.index.name = 'Database'
max_recall_fn.columns = ['False negatives of maximum recall']
max_recall_fn
```

Out[9]:

	False negatives of maximum recall
Database	
CPRD	7L1H6, BA0z., ByuE., G64z4, BBd9., 1O0.., BBa.., G61X., G60X., ByuE0, BBaz., 7L1H7, G6W.., G6X..
GePaRD	I64, I46.0, I21.9
Medicare	410.X2

`CPRD`: READ2 codes from the reference are mapped to READ CTV3 codes that are not in UMLS, for example `7L1H6` (READ2) -> `XaM3E, XaPuP, 7L1H6, 7L1h6`.

`GePaRD`: Only 3 codes are missing, but those FN have large influence on sensitivity in mappings with few codes.

```python
In [10]: averages = ev[ev.variation == 'max-recall'].groupby('event').recall.mean()
         averages.name = measure_label('recall')
         averages.index = averages.index.map(event_names.get)
         with mystyle(measures_palette, xrot=45, ha='right', ylim=(0.9,1), savefig='max-recall-by-event
             averages.plot(kind='bar', legend=False, title="Maximum recall")
             plt.ylabel(measure_label('recall'))
         averages.to_frame()
```

Out[10]:

	Sensitivity
Acute Pancreatitis	1.000000
Bladder cancer	0.980769
Hemorrhagic stroke	0.986111
Ischemic stroke	0.939773
Myocardial Infarction	0.924603
Pancreatic Cancer	0.983945
Ventricular Arrhythmia	0.931481

Maximum recall
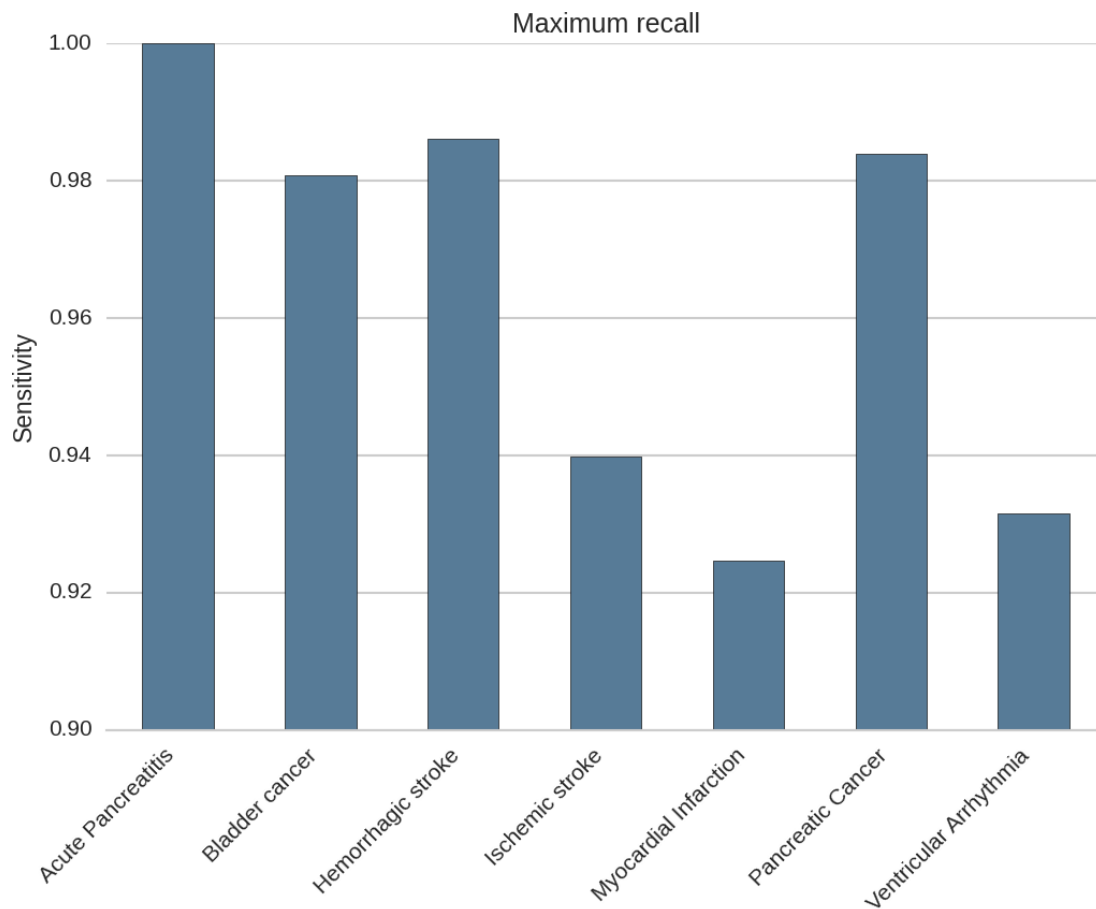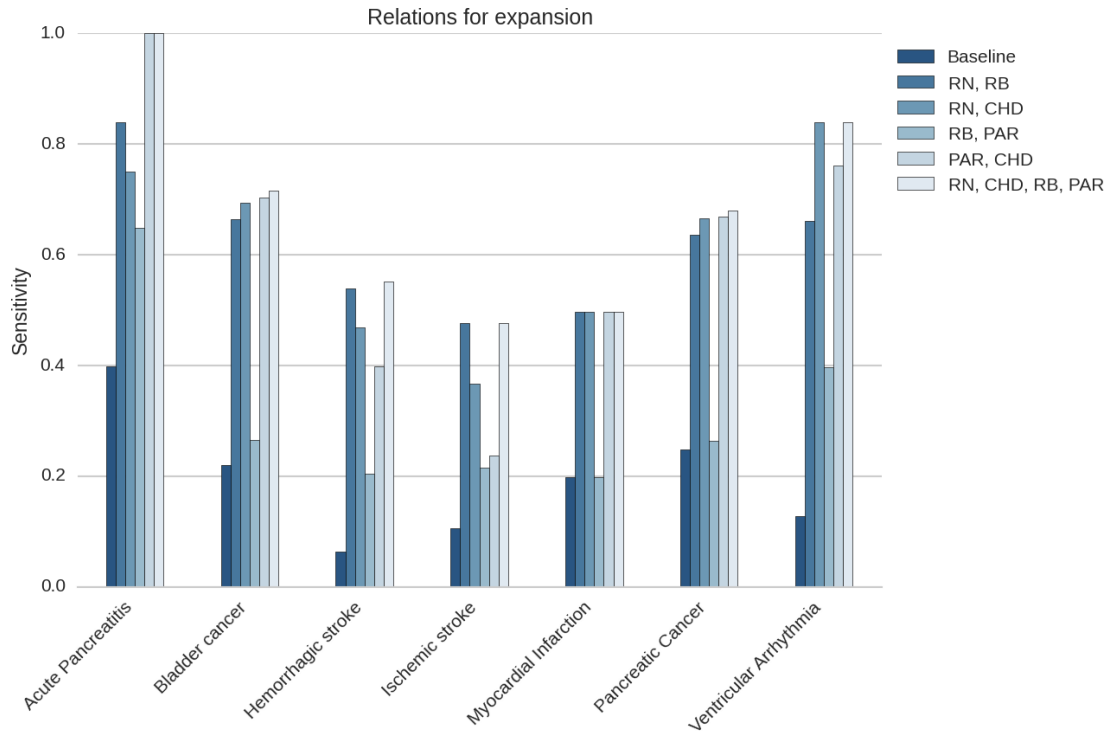
```
In [11]: averages_compare = pd.DataFrame([
             ev[ev.variation == 'max-recall'].groupby('database').recall.mean(),
             ev[ev.variation == 'max-recall'].groupby('database').precision.mean(),
         ])
         averages_compare.index = averages_compare.index.map(measure_label)
         averages_compare.columns = averages_compare.columns.map(database_label)
         with mystyle(measures_palette, savefig='max-recall-performance-by-db.pdf'):
             averages_compare.T.plot(kind='bar', title='Maximum recall')
```
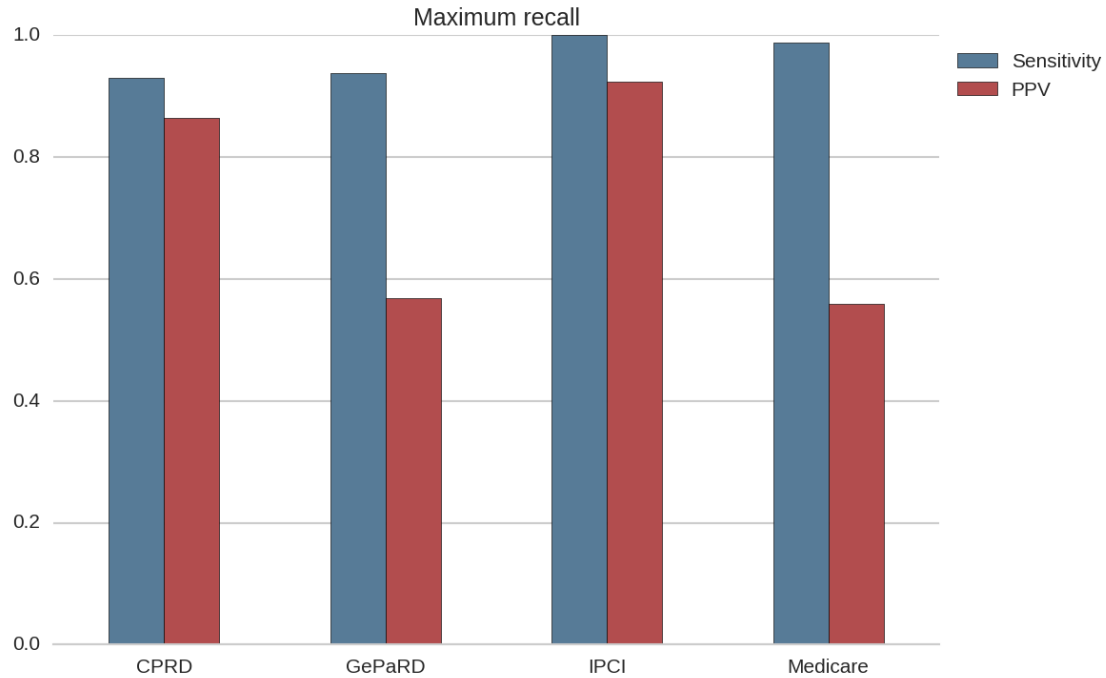
## 5 Compare relations for expansion

```
In [12]: compare_variations = OrderedDict([
             ('baseline', 'Baseline'),
             ('1-RN-RB.expand', 'RN, RB'),
             ('1-RN-CHD.expand', 'RN, CHD'),
             ('1-RB-PAR.expand', 'RB, PAR'),
             ('1-PAR-CHD.expand', 'PAR, CHD'),
             ('1-RN-CHD-RB-PAR.expand', 'RN, CHD, RB, PAR'),
         ])
         averages_compare = pd.DataFrame([
             ev[ev.variation == variation].groupby('event').recall.mean()
             for variation in compare_variations
         ], index=compare_variations)
         averages_compare.columns = averages_compare.columns.map(event_names.get)
         averages_compare.index = compare_variations.values()

         with mystyle(graded_recall_palette(len(compare_variations)), xrot=45, ha='right', savefig='rela
             averages_compare.T.plot(kind='bar', title="Relations for expansion")
             plt.ylabel(measure_label('recall'))
```

Relations for expansion
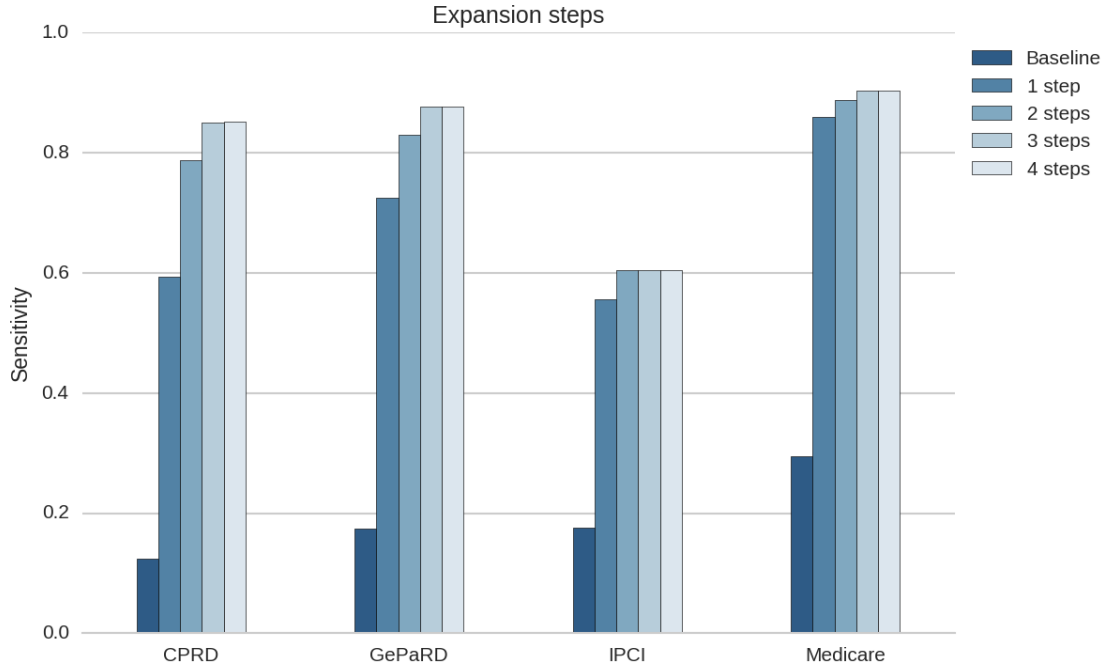
```
In [13]: compare_variations = OrderedDict([
            ('baseline', 'Baseline'),
            ('1-RN-RB.expand', 'RN, RB'),
            ('1-RN-CHD.expand', 'RN, CHD'),
            ('1-RB-PAR.expand', 'RB, PAR'),
            ('1-PAR-CHD.expand', 'PAR, CHD'),
            ('1-RN-CHD-RB-PAR.expand', 'RN, CHD, RB, PAR'),
        ])
        averages_compare = pd.DataFrame([
            ev[ev.variation == variation].groupby('database').recall.mean()
            for variation in compare_variations
        ], index=compare_variations)
        averages_compare.columns = averages_compare.columns.map(database_label)
        averages_compare.index = compare_variations.values()
        with mystyle(graded_recall_palette(len(compare_variations)), savefig='relations-recall-by-db.p
            averages_compare.T.plot(kind='bar', title="Relations for expansion")
            plt.ylabel(measure_label('recall'))
```

Relations for expansion
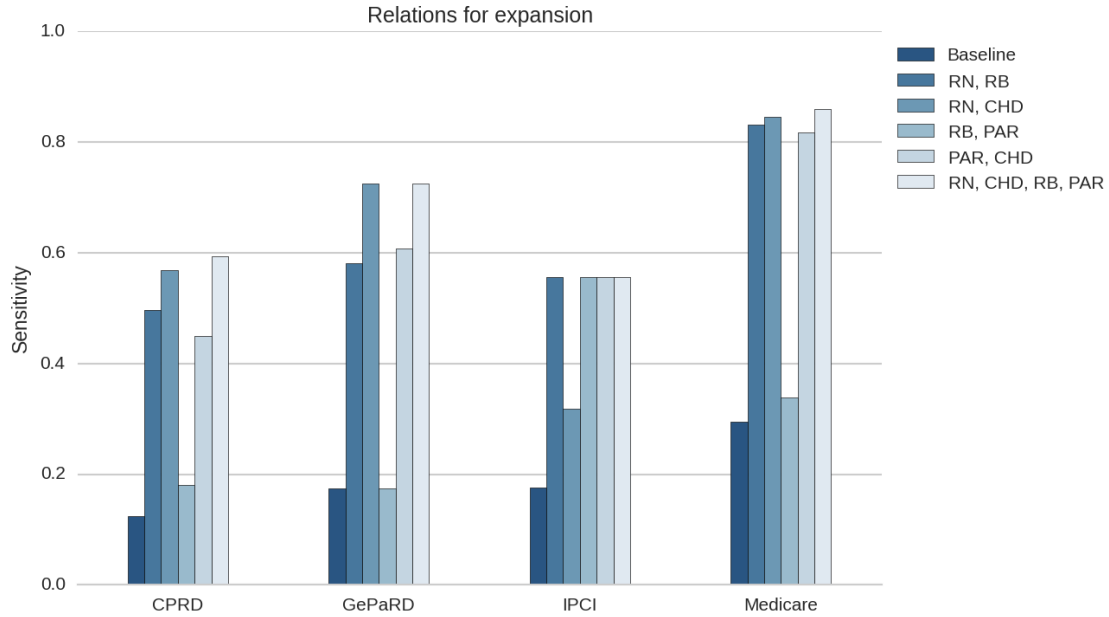
6 Increasing sensitivity with more expansion steps

```
In [14]: compare_variations = OrderedDict([
             ('baseline', 'Baseline'),
             ('1-RN-CHD-RB-PAR.expand', '1 step'),
             ('2-RN-CHD-RB-PAR.expand', '2 steps'),
             ('3-RN-CHD-RB-PAR.expand', '3 steps'),
             ('4-RN-CHD-RB-PAR.expand', '4 steps'),
         ])
         averages_compare = pd.DataFrame([
             ev[ev.variation == variation].groupby('database').recall.mean()
             for variation in compare_variations
         ], index=compare_variations)
         averages_compare.columns = averages_compare.columns.map(database_label)
         averages_compare.index = compare_variations.values()

         with mystyle(graded_recall_palette(len(compare_variations)), savefig='steps-recall-by-db.pdf')
             averages_compare.T.plot(kind='bar', title="Expansion steps")
             plt.ylabel(measure_label('recall'))
```

Expansion steps

## 6.1 Reasons for low performance in IPCI when including exclusion codes

Exclusion codes are not in the evaluation any more. See note above.

The IPCI mapping contains *very* broad codes that are refined with additional terms. For example

- `K24` (Fear of heart attack)
- `K90` (stroke)
- `K93` (Pulmonary embolism)
- `D70` (Dementia) OR "dementia" AND "infarct"
- `U14` (Kidney symptom/complaint) OR "nier" AND "infarct"
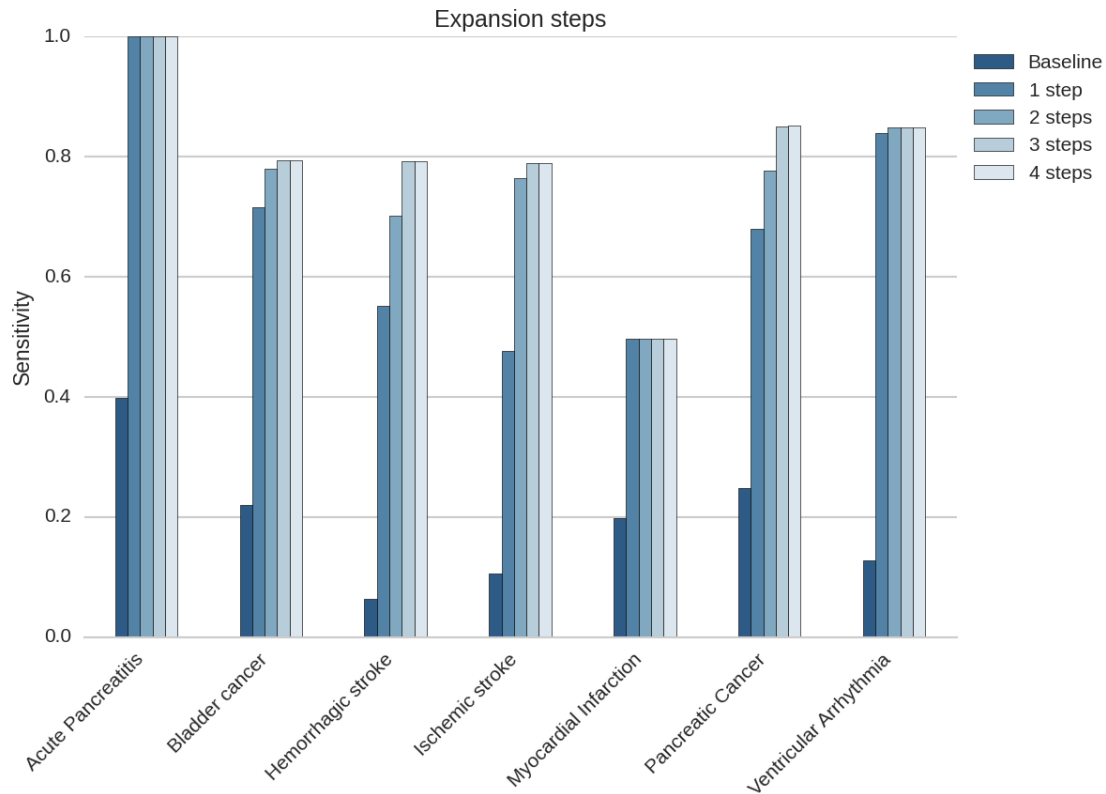
```
In [15]: compare_variations = OrderedDict([
             ('baseline', 'Baseline'),
             ('1-RN-CHD-RB-PAR.expand', '1 step'),
             ('2-RN-CHD-RB-PAR.expand', '2 steps'),
             ('3-RN-CHD-RB-PAR.expand', '3 steps'),
             ('4-RN-CHD-RB-PAR.expand', '4 steps'),
         ])
         averages_compare = pd.DataFrame([
             ev[ev.variation == variation].groupby('database').precision.mean()
             for variation in compare_variations
         ], index=compare_variations)
         averages_compare.columns = averages_compare.columns.map(database_label)
         averages_compare.index = compare_variations.values()

         with mystyle(graded_precision_palette(len(compare_variations)), savefig='steps-precision-by-db
             averages_compare.T.plot(kind='bar', title="Expansion steps")
             plt.ylabel(measure_label('precision'))
```
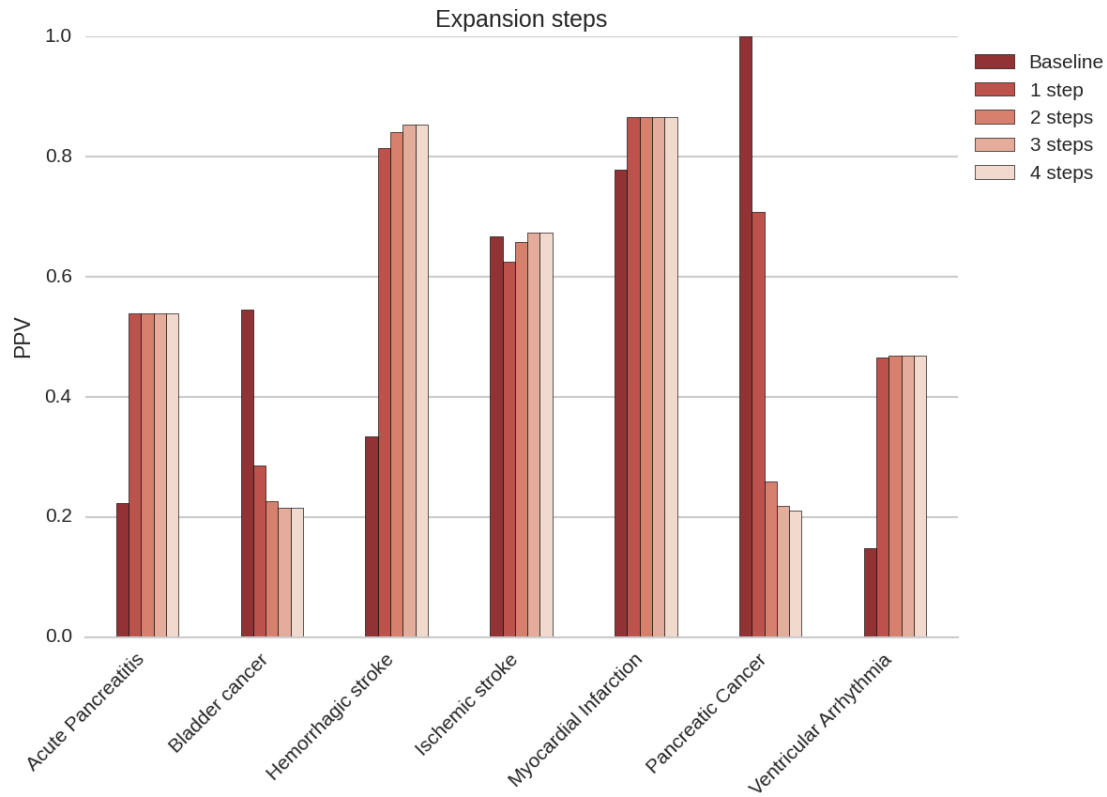
Expansion steps
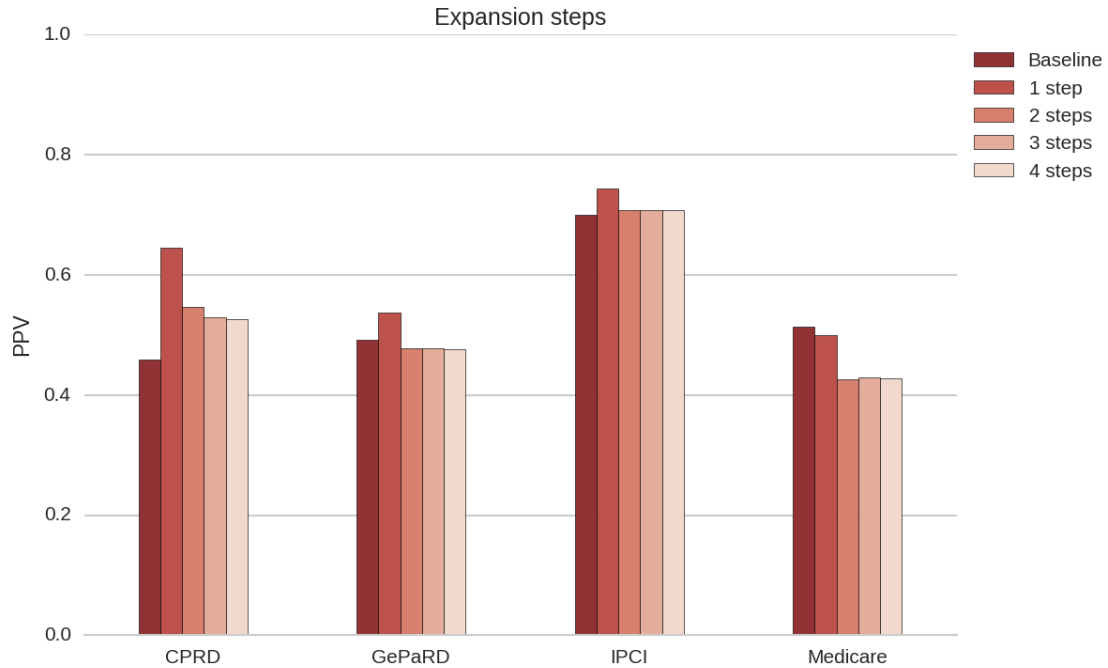
```
In [16]: compare_variations = OrderedDict([
            ('baseline', 'Baseline'),
            ('1-RN-CHD-RB-PAR.expand', '1 step'),
            ('2-RN-CHD-RB-PAR.expand', '2 steps'),
            ('3-RN-CHD-RB-PAR.expand', '3 steps'),
            ('4-RN-CHD-RB-PAR.expand', '4 steps'),
        ])
        averages_compare = pd.DataFrame([
            ev[ev.variation == variation].groupby('event').recall.mean()
            for variation in compare_variations
        ], index=compare_variations)
        averages_compare.columns = averages_compare.columns.map(event_names.get)
        averages_compare.index = compare_variations.values()

        with mystyle(graded_recall_palette(len(compare_variations)), xrot=45, ha='right', savefig='step
            averages_compare.T.plot(kind='bar', title="Expansion steps")
            plt.ylabel(measure_label('recall'))
```

Expansion steps

(Bar chart titled "Expansion steps" with y-axis "Sensitivity" from 0.0 to 1.0, x-axis categories: Acute Pancreatitis, Bladder cancer, Hemorrhagic stroke, Ischemic stroke, Myocardial Infarction, Pancreatic Cancer, Ventricular Arrhythmia. Legend: Baseline, 1 step, 2 steps, 3 steps, 4 steps)
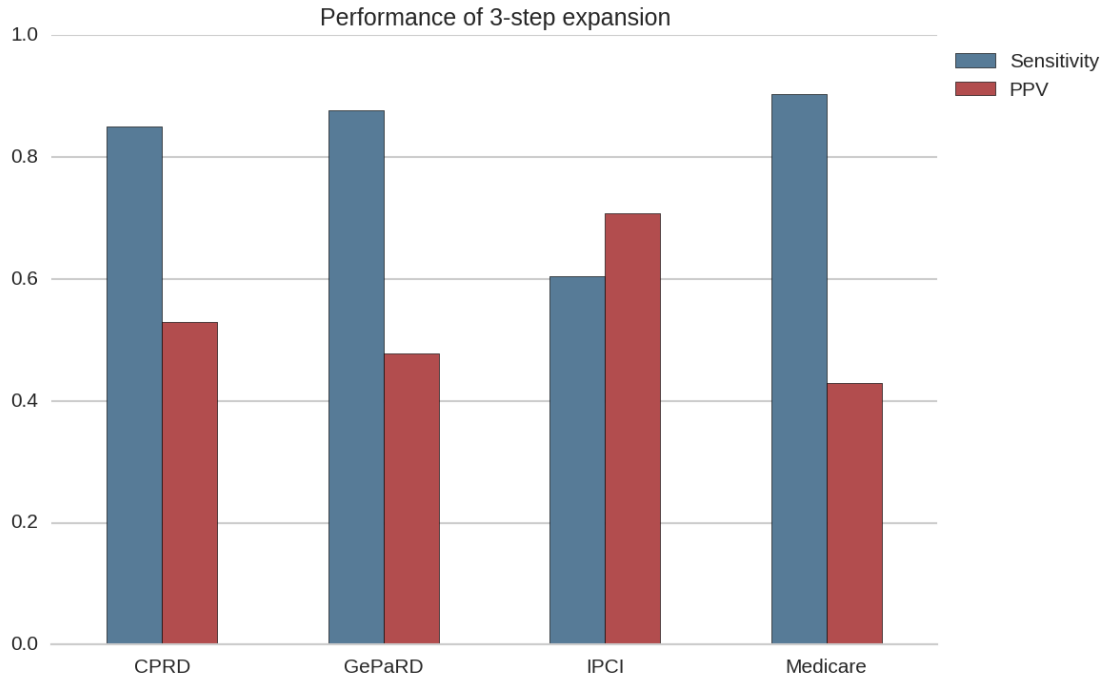
```
In [17]: compare_variations = OrderedDict([
            ('baseline', 'Baseline'),
            ('1-RN-CHD-RB-PAR.expand', '1 step'),
            ('2-RN-CHD-RB-PAR.expand', '2 steps'),
            ('3-RN-CHD-RB-PAR.expand', '3 steps'),
            ('4-RN-CHD-RB-PAR.expand', '4 steps'),
         ])
         averages_compare = pd.DataFrame([
             ev[ev.variation == variation].groupby('event').precision.mean()
             for variation in compare_variations
         ], index=compare_variations)
         averages_compare.columns = averages_compare.columns.map(event_names.get)
         averages_compare.index = compare_variations.values()

         with mystyle(graded_precision_palette(len(compare_variations)), xrot=45, ha='right', savefig='
             averages_compare.T.plot(kind='bar', title="Expansion steps")
             plt.ylabel(measure_label('precision'))
```
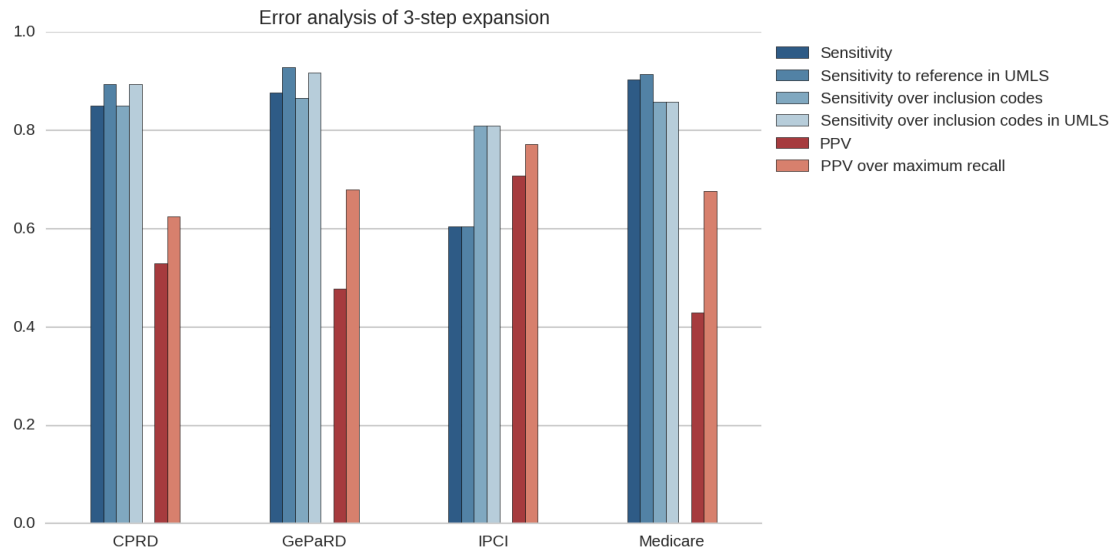
Expansion steps

```
In [18]: measures = ['recall', 'precision']
         averages_compare = pd.DataFrame([
             ev[ev.variation == '3-RN-CHD-RB-PAR.expand'].groupby('database')[measure].mean()
             for measure in measures
         ], index=map(measure_label, measures))
         averages_compare.columns = averages_compare.columns.map(database_label)

         with mystyle(measures_palette, savefig='expansion3-performance-by-db.pdf'):
             averages_compare.T.plot(kind='bar', title="Performance of 3-step expansion")
```

Performance of 3-step expansion

	Sensitivity	PPV

Bar chart with x-axis categories: CPRD, GePaRD, IPCI, Medicare. Y-axis from 0.0 to 1.0.

```
In [19]: measures = OrderedDict([
             ('recall', measure_label('recall')),
             ('recall_in_umls', '{} to reference in UMLS'.format(measure_label('recall'))),
             ('recall_without_exclusions', '{} over inclusion codes'.format(measure_label('recall'))),
             ('recall_without_exclusions_in_umls', '{} over inclusion codes in UMLS'.format(measure_lab
             ('', ''),
             ('precision', measure_label('precision')),
             ('precision_over_dnf', '{} over maximum recall'.format(measure_label('precision'))),
         ])
         averages_compare = pd.DataFrame([
             ev[ev.variation == '3-RN-CHD-RB-PAR.expand'].groupby('database')[measure].mean()\
                 if measure else\
                 pd.Series([0] * len(ev.database.unique()), index=ev.database.unique())
             for measure in measures
         ], index=measures.values())
         averages_compare.columns = averages_compare.columns.map(database_label)

         p = sns.color_palette(graded_recall_palette(5)[:-1] + [(1,1,1)] + graded_precision_palette(3)[
         with mystyle(p, savefig='expansion3-error-analysis-by-db.pdf'):
             averages_compare.T.plot(kind='bar', title="Error analysis of 3-step expansion")
```

Error analysis of 3-step expansion

```
In [20]: averages_compare = pd.DataFrame([
             ev[ev.variation == '3-RN-CHD-RB-PAR.expand'].groupby('event')[measure].mean()\
                 if measure else\
                 pd.Series([0] * len(ev.event.unique()), index=ev.event.unique())
             for measure in measures
         ], index=measures.values())
         averages_compare.columns = averages_compare.columns.map(event_label)

         p = sns.color_palette(graded_recall_palette(5)[:-1] + [(1,1,1)] + graded_precision_palette(3)[
         with mystyle(p, savefig='expansion3-error-analysis-by-db.pdf', xrot=45, ha='right'):
             averages_compare.T.plot(kind='bar', title="Error analysis of 3-step expansion")
```


Error analysis of 3-step expansion
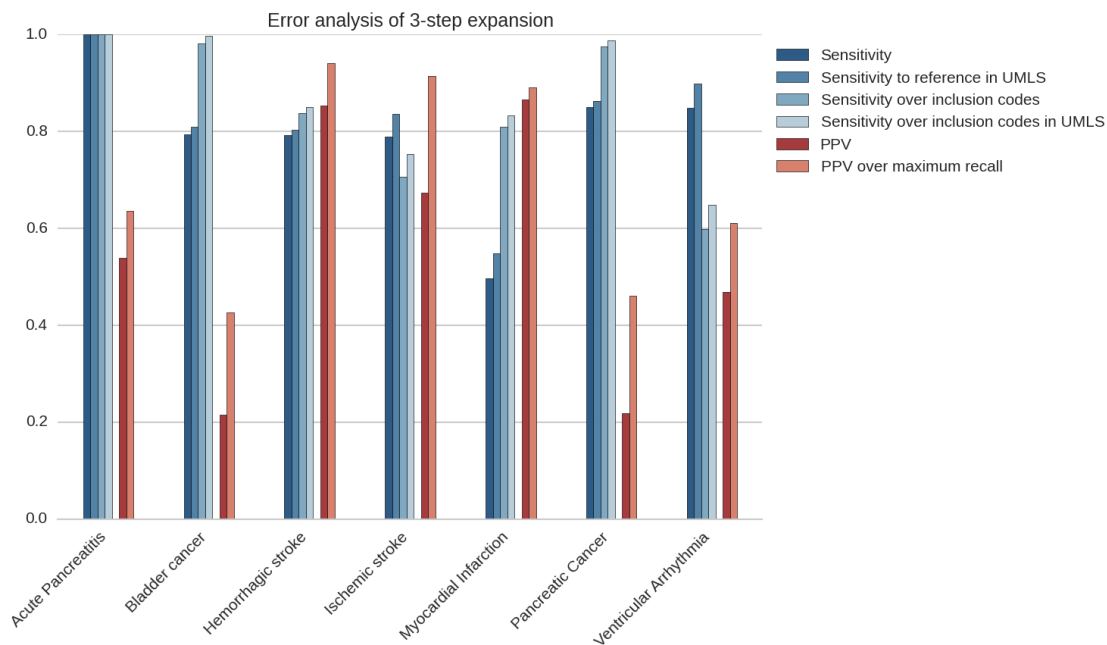
```
In [21]: residuals = ev[ev.variation == '3-RN-CHD-RB-PAR.expand']

         residuals.fn_inclusions_in_umls = residuals.fn_inclusions_in_umls\
             .fillna('NaN').map(json.loads)
         def get_missed(row):
             if math.isnan(row.recall_without_exclusions_in_umls):
                 return ''
             else:
                 reference = set(json.loads(row.reference_inclusions_in_umls))
                 return "{}/{}".format(len(row.fn_inclusions_in_umls), len(reference))

         residuals['missed'] = residuals.apply(get_missed, axis=1)

         residuals.fn_inclusions_in_umls = residuals.fn_inclusions_in_umls\
             .map(lambda s: ', '.join(s) if type(s) == list else 'N/A')
         residuals.database = residuals.database.map(database_label)
         residuals.event = residuals.event.map(event_label)
         residuals.recall_without_exclusions_in_umls = residuals.recall_without_exclusions_in_umls\
             .map('{:.2f}'.format)
         residuals = residuals.sort_index(by=['database', 'event']).reset_index(drop=True)

         residuals = residuals[['database', 'event', 'recall_without_exclusions_in_umls', 'missed', 'fn_
         residuals.columns = ["Database", "Event", "Recall", "Missed", "Residual FNs"]

         #residuals = residuals.set_index(['Database', 'Event'])["Residual FNs"].unstack()
         residuals

/usr/lib/python3/dist-packages/pandas/core/generic.py:1858: SettingWithCopyWarning: A value is trying to
Try using .loc[row_index,col_indexer] = value instead
  self[name] = value

Out[21]:
```

	Database	Event	Recall	Missed	Residual FNs
0	CPRD	Acute Pancreatitis	1.00	0/7	
1	CPRD	Bladder cancer	0.99	1/85	1O0..
2	CPRD	Hemorrhagic stroke	0.88	4/34	70041, 70042, 70043, 70082
3	CPRD	Ischemic stroke	0.94	1/18	G64z4
4	CPRD	Myocardial Infarction	nan		N/A
5	CPRD	Pancreatic Cancer	0.95	5/103	1O0.., B8086, BB44., BB5C., BB5Cz
6	CPRD	Ventricular Arrhythmia	0.59	11/27	3283., 7L1H., 7L1H4, 7L1H6, 7L1H7, 7L1Hy, 7L1Hz, 853
7	GePaRD	Acute Pancreatitis	1.00	0/6	
8	GePaRD	Bladder cancer	1.00	0/12	
9	GePaRD	Hemorrhagic stroke	0.92	0/24	
10	GePaRD	Ischemic stroke	1.00	0/10	
11	GePaRD	Myocardial Infarction	0.50	3/6	I21.0, I21.1, I21.2
12	GePaRD	Pancreatic Cancer	1.00	0/9	
13	GePaRD	Ventricular Arrhythmia	1.00	0/4	
14	IPCI	Acute Pancreatitis	1.00	0/1	
15	IPCI	Bladder cancer	1.00	0/1	
16	IPCI	Hemorrhagic stroke	1.00	0/1	
17	IPCI	Ischemic stroke	0.67	0/3	
18	IPCI	Myocardial Infarction	1.00	0/1	
19	IPCI	Pancreatic Cancer	1.00	0/1	
20	IPCI	Ventricular Arrhythmia	0.00	0/1	
21	Medicare	Acute Pancreatitis	1.00	0/1	
22	Medicare	Bladder cancer	1.00	0/12	
23	Medicare	Hemorrhagic stroke	0.60	0/5	
24	Medicare	Ischemic stroke	0.40	6/10	433.01, 433.11, 433.21, 433.31, 433.81, 434.01
25	Medicare	Myocardial Infarction	1.00	0/11	
26	Medicare	Pancreatic Cancer	1.00	0/8	
27	Medicare	Ventricular Arrhythmia	1.00	0/5	

7 Removing unused codes
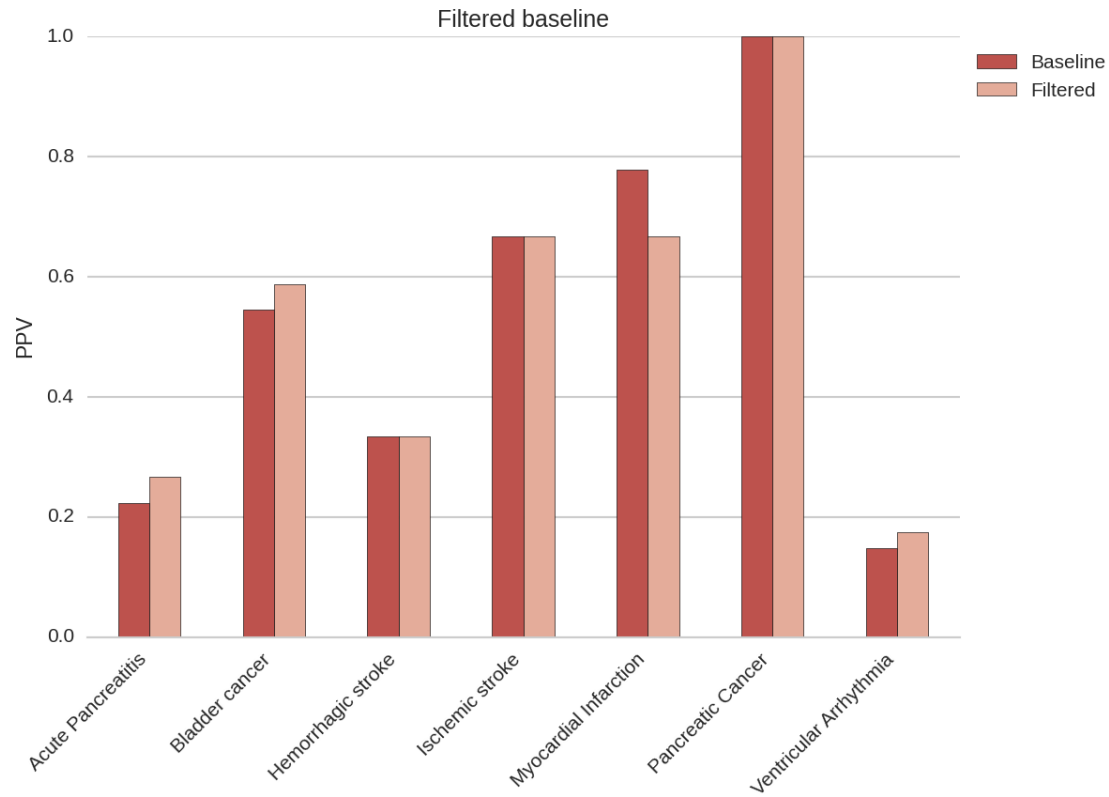
```
In [22]: compare_variations = OrderedDict([
             ('baseline', 'Baseline'),
             ('baseline.filter-gen', 'Filtered'),
         ])
         averages_compare = pd.DataFrame([
             ev[ev.variation == variation].groupby('database').precision.mean()
             for variation in compare_variations
         ], index = compare_variations.values())
         averages_compare.columns = averages_compare.columns.map(database_label)

         with mystyle(graded_precision_palette(len(compare_variations)), savefig='filtered-baseline-pre
             averages_compare.T.plot(kind='bar', title="Filtered baseline")
             plt.ylabel(measure_label('precision'))
```

Filtered baseline

```
In [23]: compare_variations = OrderedDict([
             ('baseline', 'Baseline'),
             ('baseline.filter-gen', 'Filtered'),
         ])
         averages_compare = pd.DataFrame([
             ev[ev.variation == variation].groupby('event').precision.mean()
             for variation in compare_variations
         ], index = compare_variations.values())
         averages_compare.columns = averages_compare.columns.map(event_label)

         with mystyle(graded_precision_palette(len(compare_variations)), xrot=45, ha='right', savefig=':
             averages_compare.T.plot(kind='bar', title="Filtered baseline")
             plt.ylabel(measure_label('precision'))
```

Filtered baseline

PPV

Acute Pancreatitis, Bladder cancer, Hemorrhagic stroke, Ischemic stroke, Myocardial Infarction, Pancreatic Cancer, Ventricular Arrhythmia

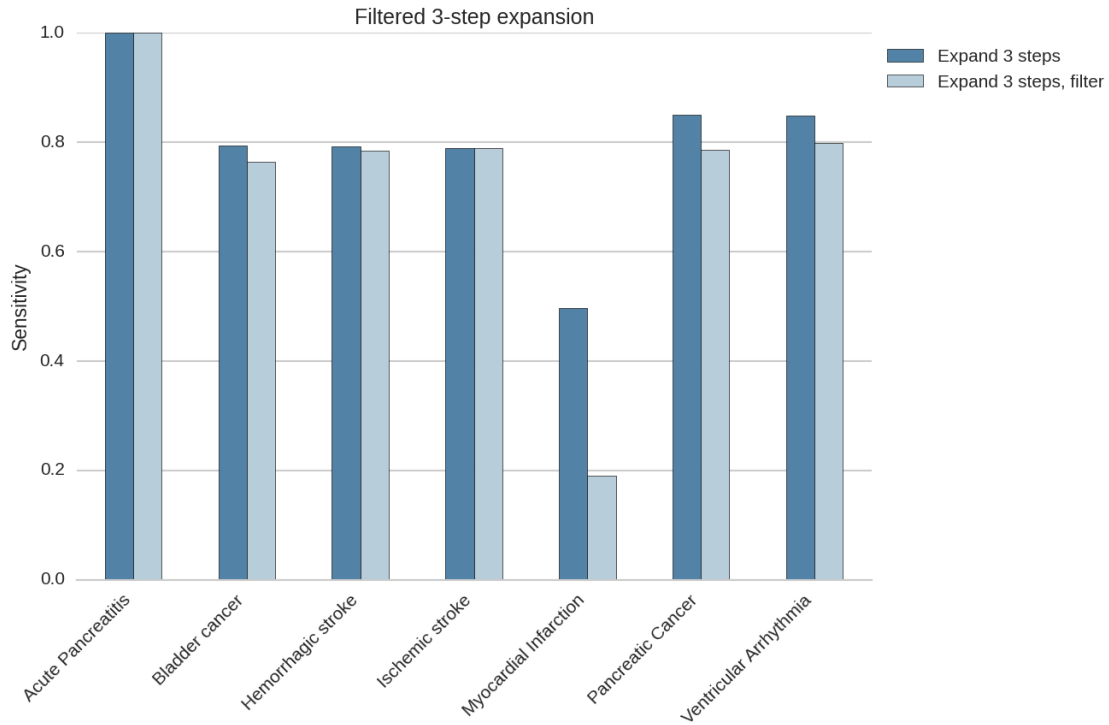Baseline, Filtered

```
In [24]: compare_variations = OrderedDict([
             ('3-RN-CHD-RB-PAR.expand', 'Expand 3 steps'),
             ('3-RN-CHD-RB-PAR.expand.filter-gen', 'Expand 3 steps, filter'),
         ])
         averages_compare = pd.DataFrame([
             ev[ev.variation == variation].groupby('database').precision.mean()
             for variation in compare_variations
         ], index = compare_variations.values())
         averages_compare.columns = averages_compare.columns.map(database_label)

         with mystyle(graded_precision_palette(len(compare_variations)), savefig='filtered-expansion3-p
             averages_compare.T.plot(kind='bar', title="Filtered expansion")
             plt.ylabel(measure_label('precision'))
```
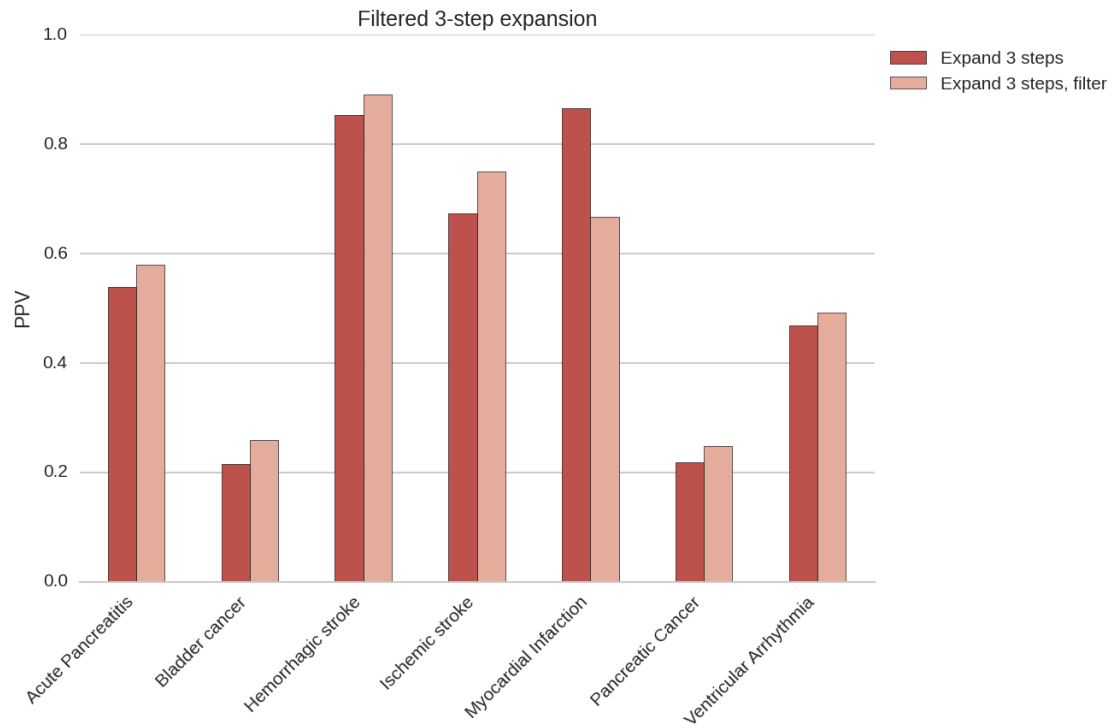
Filtered expansion
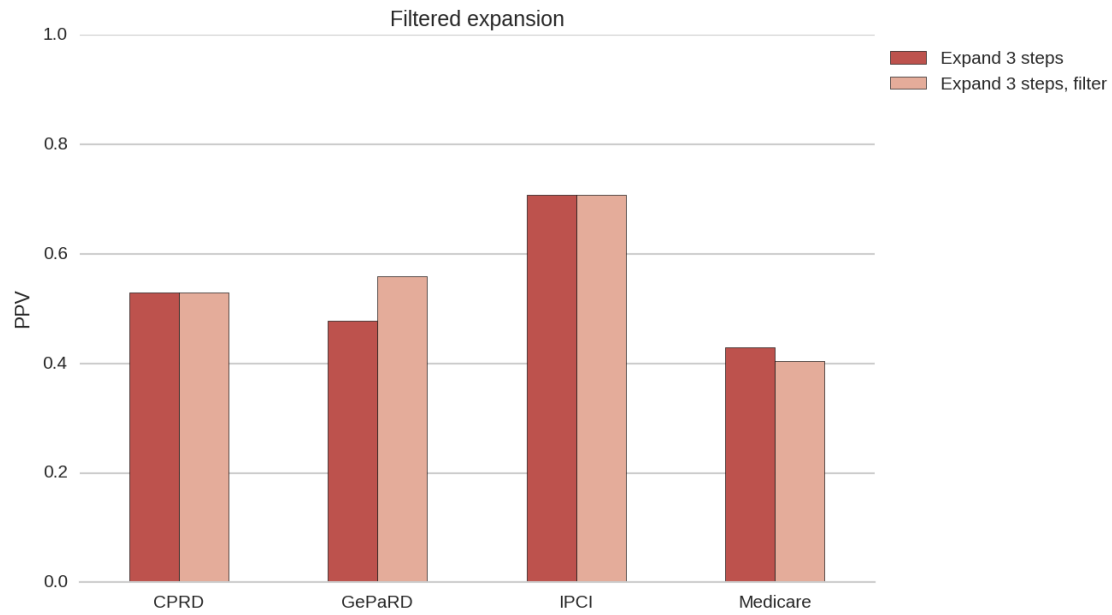
```
In [25]: compare_variations = OrderedDict([
             ('3-RN-CHD-RB-PAR.expand', 'Expand 3 steps'),
             ('3-RN-CHD-RB-PAR.expand.filter-gen', 'Expand 3 steps, filter'),
         ])
         averages_compare = pd.DataFrame([
             ev[ev.variation == variation].groupby('event').recall.mean()
             for variation in compare_variations
         ], index = compare_variations.values())
         averages_compare.columns = averages_compare.columns.map(event_names.get)

         with mystyle(graded_recall_palette(len(compare_variations)), xrot=45, ha='right', savefig='fil
             averages_compare.T.plot(kind='bar', title="Filtered 3-step expansion")
             plt.ylabel(measure_label('recall'))
```

Filtered 3-step expansion

```
In [26]: compare_variations = OrderedDict([
             ('3-RN-CHD-RB-PAR.expand', 'Expand 3 steps'),
             ('3-RN-CHD-RB-PAR.expand.filter-gen', 'Expand 3 steps, filter'),
         ])
         averages_compare = pd.DataFrame([
             ev[ev.variation == variation].groupby('event').precision.mean()
             for variation in compare_variations
         ], index = compare_variations.values())
         averages_compare.columns = averages_compare.columns.map(event_names.get)

         with mystyle(graded_precision_palette(len(compare_variations)), xrot=45, ha='right', savefig=':
             averages_compare.T.plot(kind='bar', title="Filtered 3-step expansion")
             plt.ylabel(measure_label('precision'))
```

Filtered 3-step expansion

Legend:
- Expand 3 steps
- Expand 3 steps, filter

Y-axis: PPV

X-axis categories: Acute Pancreatitis, Bladder cancer, Hemorrhagic stroke, Ischemic stroke, Myocardial Infarction, Pancreatic Cancer, Ventricular Arrhythmia

```
In [27]: measures = ['recall', 'precision']
         averages_compare = pd.DataFrame([
             ev[ev.variation == '3-RN-CHD-RB-PAR.expand.filter-gen'].groupby('database')[measure].mean()
             for measure in measures
         ], index=map(measure_label, measures))
         averages_compare.columns = averages_compare.columns.map(database_label)
         #averages_compare.index = compare_variations.values()

         with mystyle(measures_palette, savefig='filtered-expansion3-performance-by-db.pdf'):
             averages_compare.T.plot(kind='bar', title="Performance of filtered 3-step expansion")
```

Performance of filtered 3-step expansion
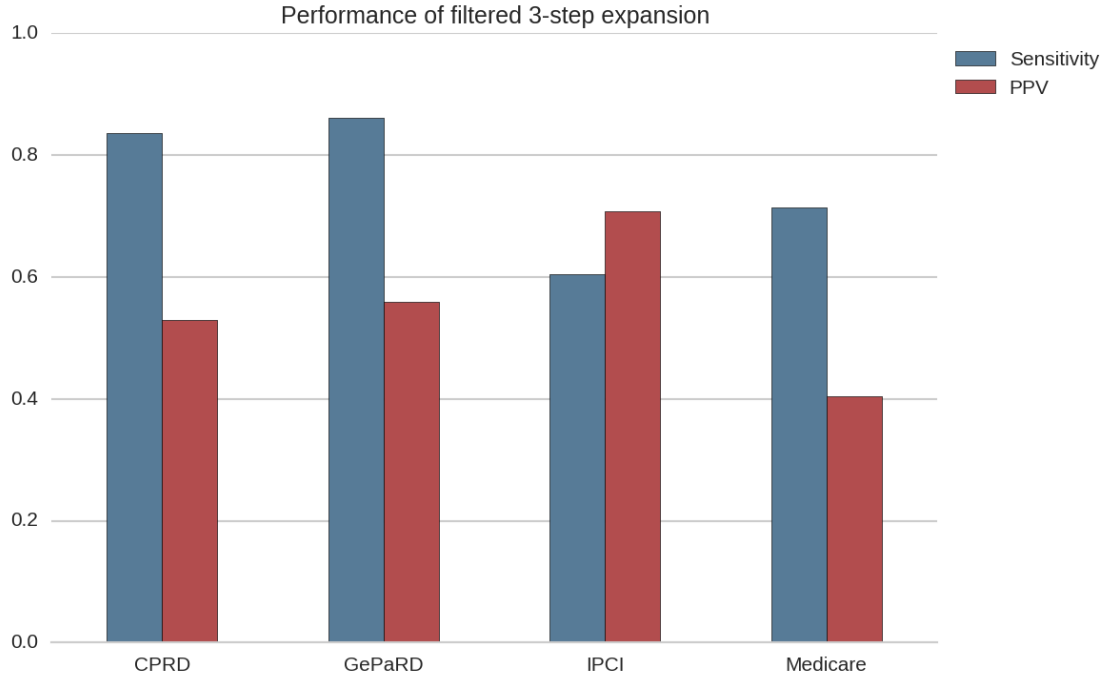
The drop in PPV for Myocardial infarction is caused by the mapping to codes `410.*` (Acute myocardial infarction) in Medicare which is *not* used in the ARS database.

8 Codes in reference mappings, not in databases

Codes that might be removed from the TP when filtering.

```
In [28]: stats = DataFrame()
         stats['In ref'] = code_stats[code_stats.InMapping]\
             .groupby('Database').Code.count()
         stats['Not in DB'] = code_stats[code_stats.InMapping & ~code_stats.InDatabase]\
             .groupby('Database').Code.count()
         stats.fillna(0, inplace=True)
         stats['%'] = (stats['Not in DB'] / stats['In ref']).map("{:.2%}".format)
         stats['Codes'] = code_stats[code_stats.InMapping & ~code_stats.InDatabase]\
             .groupby('Database').Code.aggregate(lambda vs: ', '.join(set(vs)))
         stats
```

Out[28]:

	In ref	Not in DB	%	Codes
Database				
CPRD	229	5	2.18%	8531., BB46., ByuF., B91zz, Gyu60
GePaRD	74	1	1.35%	C25
IPCI	16	0	0.00%	NaN
Medicare	53	15	28.30%	410.0, 410.5, 410.1, 427.4, 410.2, 410.4, 410.3, 410.8, 157, 410.6, 410.X2, 410

```
In [29]:
```