CoMap evaluation

January 4, 2016

In [1]:
```
%matplotlib inline
import json
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("paper", font_scale=2)
#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, rev=True):
    palette = sns.color_palette("Blues", n_colors=n_colors, desat=.6)
    if rev:
        palette.reverse()
    return palette

def graded_precision_palette(n_colors, rev=True):
    palette = sns.color_palette("Reds", n_colors=n_colors, desat=.6)
    if rev:
        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:
                    ax = plt.gca()
                    sns.despine(left=True, ax=ax)
                    ax.grid(False, axis='x')
                    if ax.legend_:
                        lgd = ax.legend(loc=2, bbox_to_anchor=(1, 1))
                    else:
                        lgd = None
                    if ax.get_lines():
                        ax.get_lines()[0].set_visible(False)
                    ax.set_ylim(*ylim)
                    plt.xticks(rotation=xrot, ha=ha)
                    if ylabel is not None:
                        ax.yaxis.label(ylabel)
                    if savefig:
                        filename = '{}-{}'.format(PROJECT, savefig)
                        plt.savefig(filename, bbox_extra_artists=[lgd] if lgd else [], bbox_inches=
        return C()

    def draw_lines(ys, palette=None):
        if palette is None:
            palette = sns.color_palette()
        ax = plt.gca()
        ax.axhline(0, 0, 0) # First axhline is not visible??
        for y, color in zip(ys, palette):
            ax.axhline(y, color=color, zorder=-100)
        return ax

    """
    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
```

```
/home/benus/.local/lib/python3.4/site-packages/IPython/html.py:14: ShimWarning: The `IPython.html` packa
  "`IPython.html.widgets` has moved to `ipywidgets`.", ShimWarning)
```

```
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))

    coding_systems = ["ICD-9", "ICD-10", "ICPC-2", "READ-2"]

    def database_label(database):
        #return database
        #return "{} ({})".format(database, databases.coding_system(database))
        return {
            "ICD10CM": "ICD-10",
            "ICD10/CM": "ICD-10",
            "RCD2": "READ-2",
            "ICPC2EENG": "ICPC-2",
            "ICD9CM": "ICD-9",
        }[databases.coding_system(database)]

    def measure_label(measure):
        return {
            "recall": "Sensitivity",
            "precision": "PPV", # "Positive predictive value",
        }[measure]

    def event_label(event):
        return event_names[event]

    def len_if_notnull(x):
        if x != x:
            return 0
        else:
            return len(x)
```

0.1 Load evaluations ev

In [3]:
```
ev = pd.read_csv('../{}.evaluations.csv'.format(PROJECT))
for key in ['generated', 'reference', 'tp', 'fp', 'fn']:
    ev[key] = ev[key].map(lambda x: x if x != x else json.loads(x))
ev['variation event database recall precision'.split()].head()
```

Out[3]:

	variation	event	database	recall	precision
0	baseline0	pc	Medicare	0.250000	1.000000
1	baseline0	pc	IPCI	1.000000	1.000000
2	baseline0	pc	CPRD	0.027523	0.600000
3	baseline0	pc	GePaRD	0.222222	1.000000
4	baseline0	ap	Medicare	1.000000	0.166667

0.2 Mappings

```
In [4]: df = mappings.describe()
        df.index = df.index.map(database_label)
        df.columns = df.columns.map(event_label)
        df['Sum'] = df.iloc[:4,:7].sum(axis=1)
        df['Average'] = df.iloc[:4,:7].mean(axis=1).round(2)
        df.ix['Sum'] = df.iloc[:4, :7].sum()
        df.ix['Average'] = df.iloc[:4, :7].mean().round(2)
        df.ix['Sum']['Sum'] = df['Sum'].sum()
        df.ix['Average']['Average'] = df['Average'].mean()
        df.fillna('-').T
```

Out[4]:

	READ-2	ICD-10	ICPC-2	ICD-9	Sum	Average
Acute Pancreatitis	7	6	1	1	15	3.75
Bladder cancer	91	12	1	12	116	29
Hemorrhagic stroke	36	22	1	3	62	15.5
Ischemic stroke	20	11	2	10	43	10.75
Myocardial Infarction	-	7	1	11	19	6.33
Pancreatic Cancer	109	9	1	8	127	31.75
Ventricular Arrhythmia	27	5	1	5	38	9.5
Sum	290	72	8	50	420	-
Average	48.33	10.29	1.14	7.14	-	16.725

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

Out[5]:

Database	Coding system	Label
Medicare	ICD9CM	ICD-9
IPCI	ICPC2EENG	ICPC-2
CPRD	RCD2	READ-2
GePaRD	ICD10/CM	ICD-10

2.1 Baseline-0

2.1.1 DISO filtering for concepts

```
In [6]: types_distr = pd.DataFrame(json.load(open('../{}.types-distrs.json'.format(PROJECT)))).T

        df = pd.DataFrame()
        df['All'] = types_distr.groupby('group')[['pos', 'neg']].sum().sum()
        df['All %'] =  df['All'] / df['All'].sum()
        df['DISO'] = types_distr.groupby('group')[['pos', 'neg']].sum().ix['DISO']
        df['DISO %'] = df['DISO'] / df['DISO'].sum()
        df
```

Out[6]:

	All	All %	DISO	DISO %
pos	19	0.069597	17	0.309091
neg	254	0.930403	38	0.690909

In [7]:
```
df = ev[ev.variation == 'baseline0'][['event', 'database', 'generated', 'reference', 'tp', 'fp'
for key in 'generated reference tp fp fn'.split():
    df[key] = df[key].map(len_if_notnull)
df['database'] = df['database'].map(database_label)
df.groupby('database').sum()
```

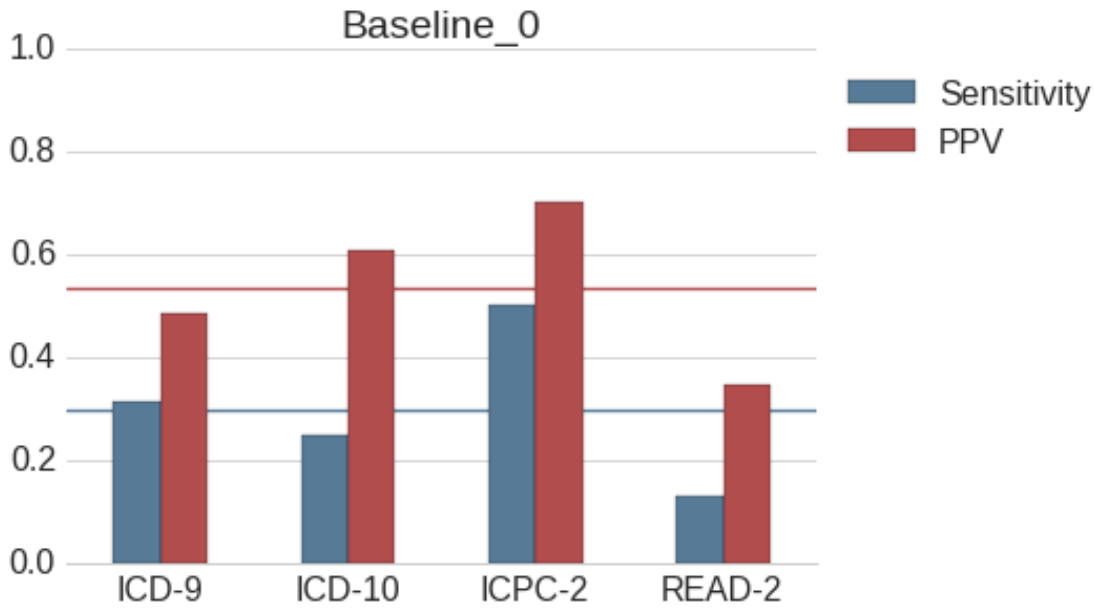
Out[7]:

database	generated	reference	tp	fp	fn
ICD-10	32	72	15	17	57
ICD-9	24	50	10	14	40
ICPC-2	7	8	4	3	4
READ-2	64	290	24	40	266

In [8]:
```
df = pd.DataFrame([
    ev[ev.variation == 'baseline0'].groupby('database').recall.mean(),
    ev[ev.variation == 'baseline0'].groupby('database').precision.mean(),
])
df.index = df.index.map(measure_label)
df.columns = df.columns.map(database_label)
df = df[coding_systems]
df['Average'] = df.mean(axis=1)

with mystyle(measures_palette, savefig='baseline-performance-by-db.pdf'):
    with sns.plotting_context(font_scale=1):
        ax = draw_lines(df['Average'])
        df.iloc[:,:-1].T.plot(kind='bar', title='Baseline_0', ax=ax)

df.round(3)
```

Out[8]:

	ICD-9	ICD-10	ICPC-2	READ-2	Average
Sensitivity	0.316	0.249	0.5	0.132	0.299
PPV	0.486	0.606	0.7	0.347	0.535



3 Baseline

In [9]:
```
df = ev[ev.variation == 'baseline'][['event', 'database', 'generated', 'reference', 'tp', 'fp',
for key in 'generated reference tp fp fn'.split():
    df[key] = df[key].map(len_if_notnull)
df['database'] = df['database'].map(database_label)
df.groupby('database').sum()
```

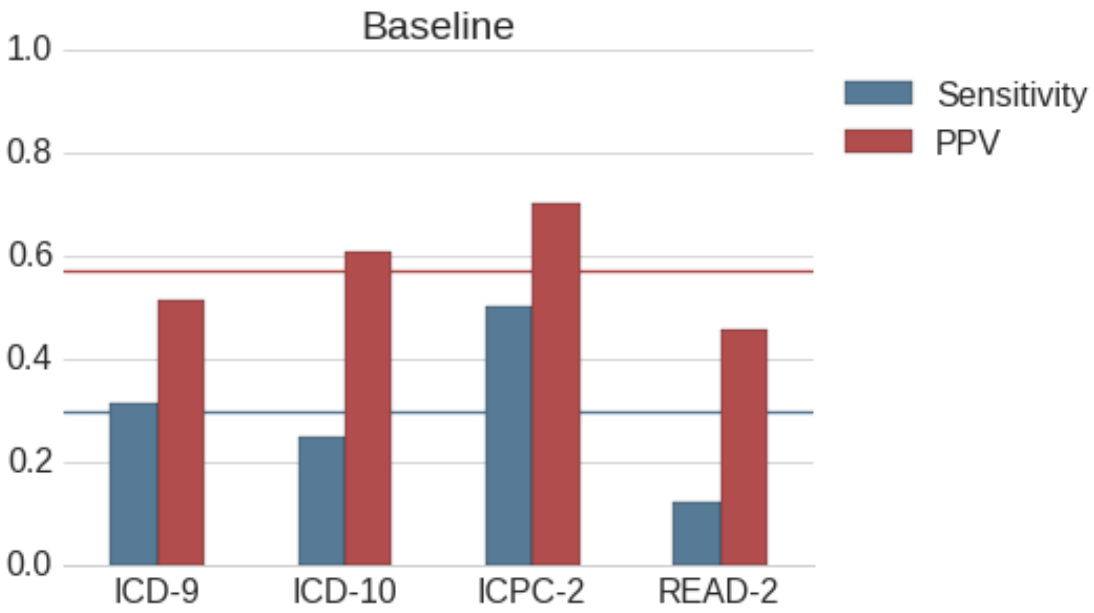
Out[9]:

database	generated	reference	tp	fp	fn
ICD-10	32	72	15	17	57
ICD-9	23	50	10	13	40
ICPC-2	6	8	4	2	4
READ-2	50	290	22	28	268

In [10]:
```
df = pd.DataFrame([
    ev[ev.variation == 'baseline'].groupby('database').recall.mean(),
    ev[ev.variation == 'baseline'].groupby('database').precision.mean(),
])
df.index = df.index.map(measure_label)
df.columns = df.columns.map(database_label)
df = df[coding_systems]
df['Average'] = df.mean(axis=1)

with mystyle(measures_palette, savefig='baseline-performance-by-db.pdf'):
    with sns.plotting_context(font_scale=1):
        ax = draw_lines(df['Average'])
        df.iloc[:,:-1].T.plot(kind='bar', title='Baseline', ax=ax)

df.round(3)
```

Out[10]:

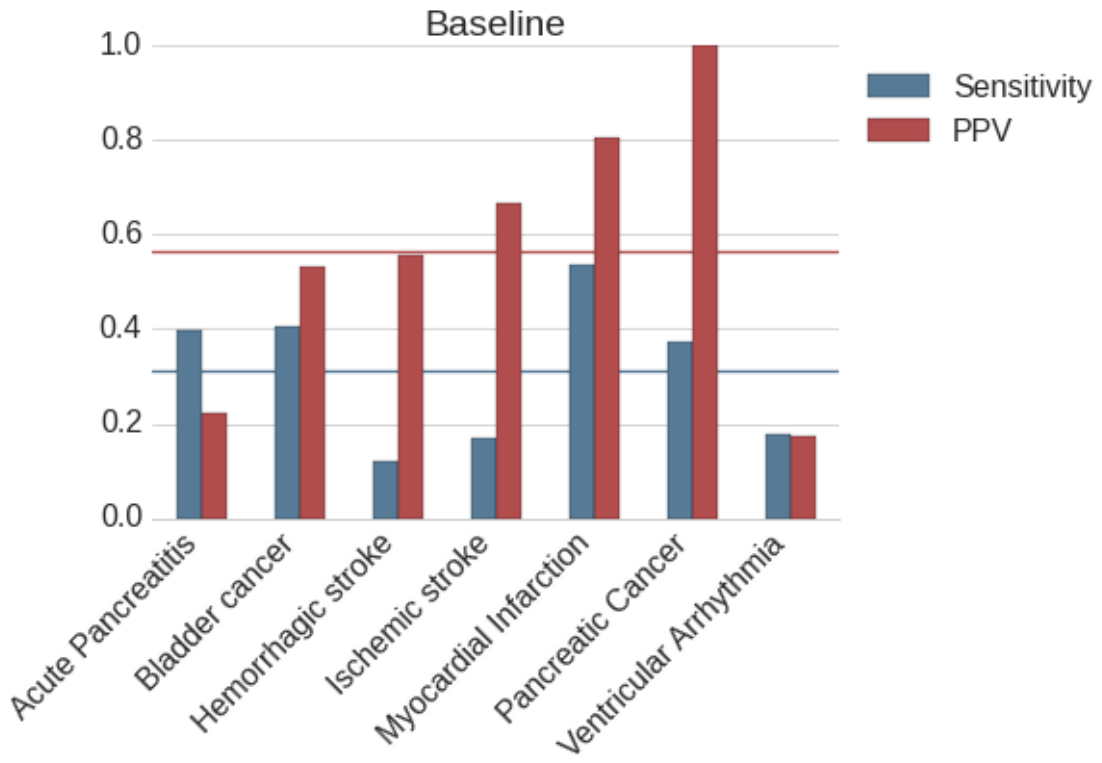
	ICD-9	ICD-10	ICPC-2	READ-2	Average
Sensitivity	0.316	0.249	0.5	0.124	0.297
PPV	0.514	0.606	0.7	0.458	0.570




```
In [11]: df = pd.DataFrame([
             ev[ev.variation == 'baseline'].groupby('event').recall.mean(),
             ev[ev.variation == 'baseline'].groupby('event').precision.mean(),
         ])
         df.index = df.index.map(measure_label)
         df.columns = df.columns.map(event_label)
         df['Average'] = df.mean(axis=1)

         with mystyle(measures_palette, xrot=45, ha='right', savefig='baseline-performance-by-event.pdf
             ax = draw_lines(df['Average'])
             df.iloc[:,:-1].T.plot(kind='bar', title='Baseline', ax=ax)

         df.round(3)
```

Out[11]:

	Acute Pancreatitis	Bladder cancer	Hemorrhagic stroke	Ischemic stroke	Myocardial Infarction	Pancreatic Cancer	Ventricular Arrhythmia	Average
Sensitivity	0.399	0.408	0.120	0.170	0.537	0.373	0.178	0.312
PPV	0.222	0.531	0.556	0.667	0.806	1.000	0.175	0.565



4 Max-recall

```
In [12]: df = ev[ev.variation == 'max-recall'][['event', 'database', 'generated', 'reference', 'tp', 'f
         for key in ['generated', 'reference', 'tp', 'fp', 'fn']:
             df[key] = df[key].map(len_if_notnull)
         df['database'] = df['database'].map(database_label)
         df = df.groupby('database').sum()
         df.ix['Overall'] = df.sum()
         df['fn/reference'] = df['fn'] / df['reference']
         #df['tp/generated'] = 1 - (df.tp / df.generated).round(3)
         df.round(3)
```

Out[12]:

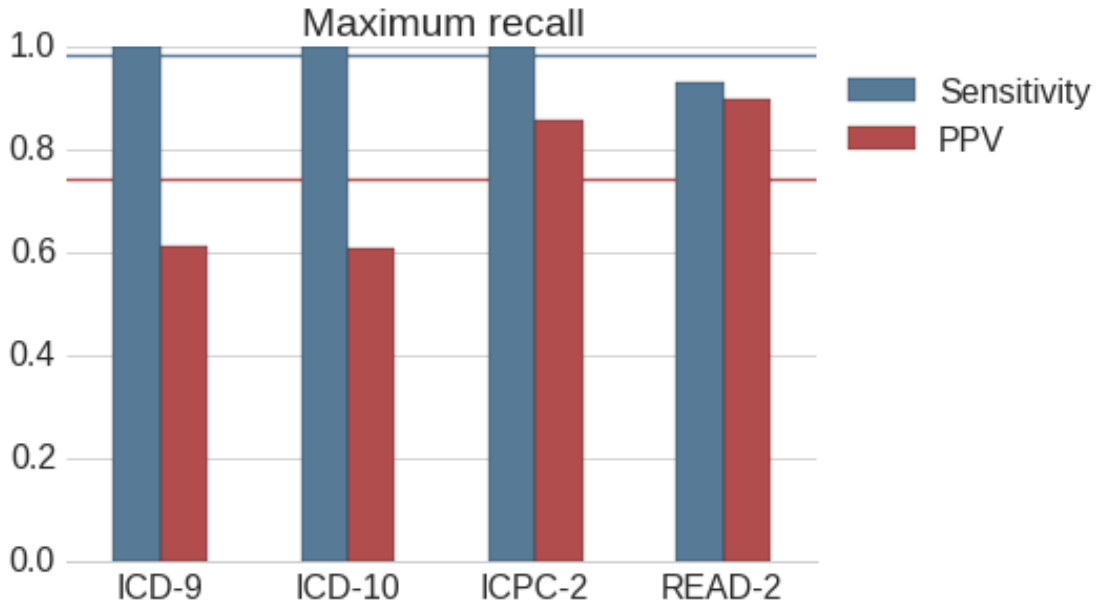
database	generated	reference	tp	fp	fn	fn/reference
ICD-10	148	72	72	76	0	0.000
ICD-9	106	50	50	56	0	0.000
ICPC-2	10	8	8	2	0	0.000
READ-2	296	290	269	27	21	0.072
Overall	560	420	399	161	21	0.050

```
In [13]: df = pd.DataFrame([
             ev[ev.variation == 'max-recall'].groupby('database').recall.mean(),
             ev[ev.variation == 'max-recall'].groupby('database').precision.mean(),
         ])
         df.index = df.index.map(measure_label)
         df.columns = df.columns.map(database_label)
```

```
df = df[coding_systems]
df['Average'] = df.mean(axis=1)

with mystyle(measures_palette, ylim=(0,1), savefig='max-recall-performance-by-db.pdf'):
    ax = draw_lines(df['Average'])
    df.iloc[:,:-1].T.plot(kind='bar', title='Maximum recall', ax=ax)

df.round(3)
```

Out[13]:

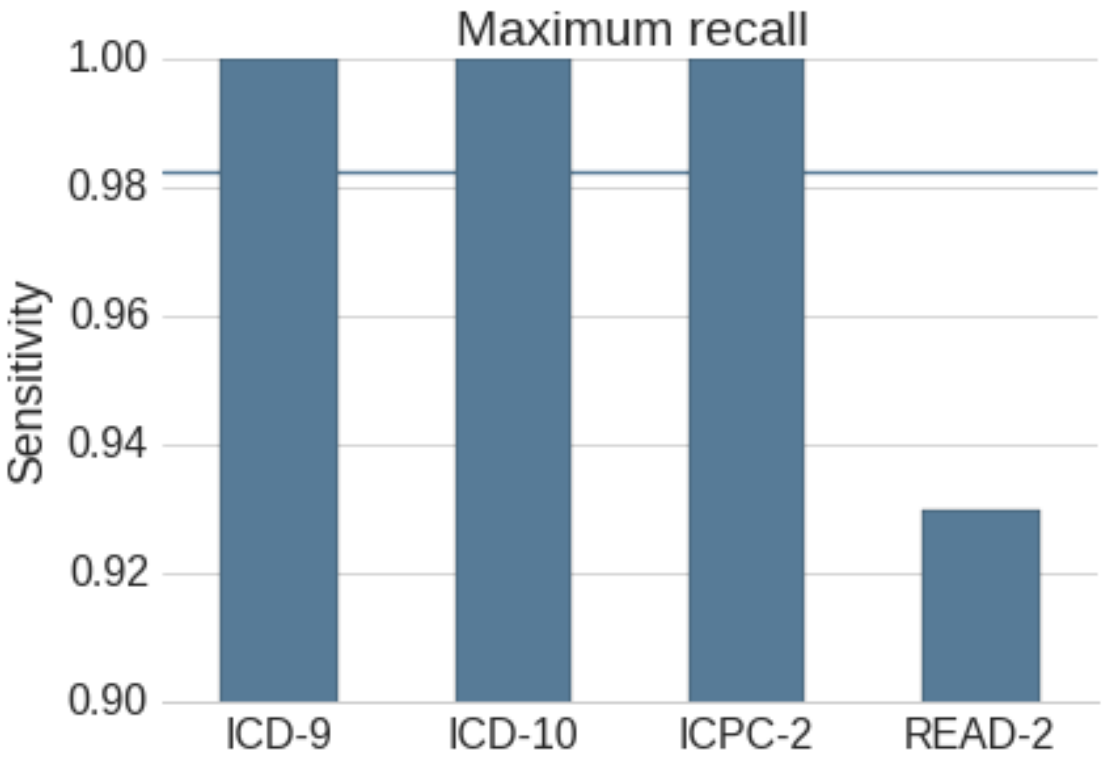
	ICD-9	ICD-10	ICPC-2	READ-2	Average
Sensitivity	1.000	1.000	1.000	0.930	0.982
PPV	0.613	0.607	0.857	0.898	0.744



In [14]:
```
df = pd.DataFrame([
    ev[ev.variation == 'max-recall'].groupby('database').recall.mean(),
])
df.index = df.index.map(measure_label)
df.columns = df.columns.map(database_label)
df = df[coding_systems]
df['Average'] = df.mean(axis=1)

with mystyle(measures_palette, ylim=(.9, 1), savefig='max-recall-recall-by-db.pdf'):
    ax = draw_lines(df['Average'])
    df.iloc[:,:-1].T.plot(kind='bar', legend=False, title='Maximum recall', ax=ax)
    plt.ylabel(measure_label('recall'))

df.round(3)
```

Out[14]:

	ICD-9	ICD-10	ICPC-2	READ-2	Average
Sensitivity	1	1	1	0.93	0.982



4.1 Reasons for imperfect sensitivity

```
In [15]: with open('../{}.code-stats.csv'.format(PROJECT)) as f:
             code_stats = pd.read_csv(f)

         stats = pd.DataFrame()
         stats['Mapping'] = (code_stats[code_stats.InMapping]
                             .groupby('Database')
                             .Code.count())
         stats['Not maximum-recall'] = (code_stats[code_stats.InMapping & ~code_stats.InDnf]
                                        .groupby('Database')
                                        .Code.count())
         stats = stats.fillna(0)
         stats['% of missing'] = (stats['Not maximum-recall'] / stats['Not maximum-recall'].sum()).map(
         stats['% of mapping'] = (stats['Not maximum-recall'] / stats['Mapping']).map("{:.2%}".format)
         stats.index = stats.index.map(database_label)
         stats
```

Out[15]:

	Mapping	Not maximum-recall	% of missing	% of mapping
READ-2	229	14	100.00%	6.11%
ICD-10	72	0	0.00%	0.00%
ICPC-2	7	0	0.00%	0.00%
ICD-9	50	0	0.00%	0.00%

```
In [16]: 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 = max_recall_fn.groupby('database').fn.sum().to_frame()
max_recall_fn['fn'] = max_recall_fn['fn'].map(lambda x: set() if x != x else set(x)).map(', '.
max_recall_fn.index.name = 'Database'
max_recall_fn.columns = ['False negatives of maximum recall']
max_recall_fn
```

Out[16]:

Database	False negatives of ma
READ-2	BBaz., ByuE0, 1O0.., G60X., 7L1H7, G6W.., BBd9., BBa.., 7L1H6, G6X.., G61X., ByuE.,

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.

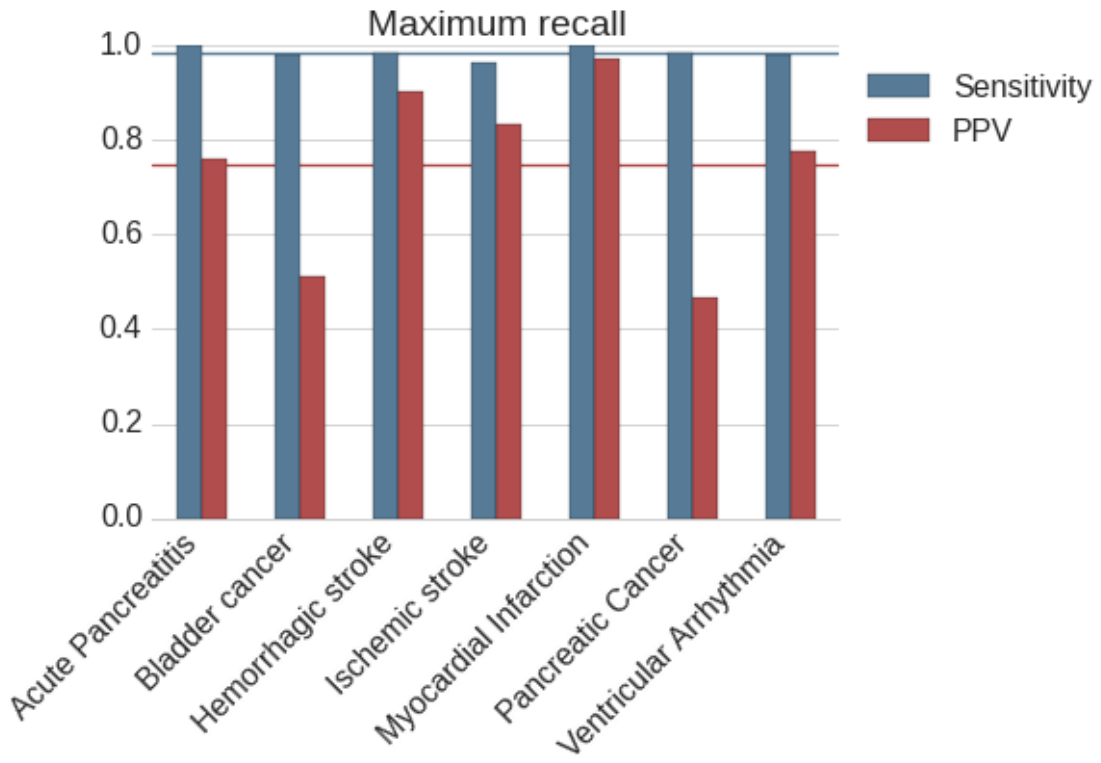
In [17]:
```
averages_compare = pd.DataFrame([
    ev[ev.variation == 'max-recall'].groupby('event').recall.mean(),
    ev[ev.variation == 'max-recall'].groupby('event').precision.mean(),
])
averages_compare.index = averages_compare.index.map(measure_label)
averages_compare.columns = averages_compare.columns.map(event_names.get)
averages_compare['Average'] = averages_compare.mean(axis=1)

with mystyle(measures_palette, xrot=45, ha='right', savefig='max-recall-by-event.pdf'):
    ax = draw_lines(averages_compare['Average'])
    averages_compare.iloc[:,:-1].T.plot(kind='bar', title="Maximum recall", ax=ax)
averages_compare.round(3)
```

Out[17]:

	Acute Pancreatitis	Bladder cancer	Hemorrhagic stroke	\
Sensitivity	1.00	0.981	0.986	
PPV	0.76	0.514	0.903	

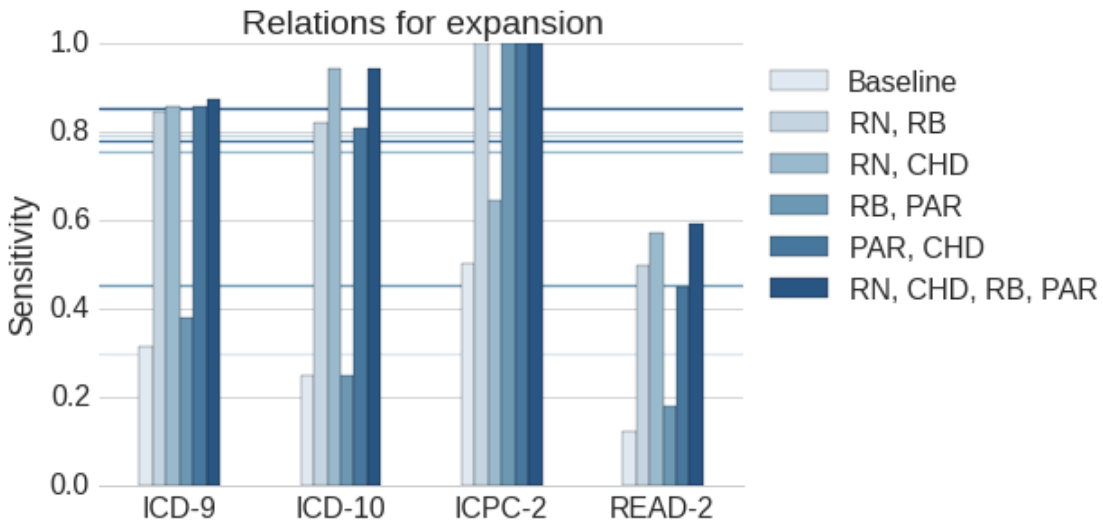
	Ischemic stroke	Myocardial Infarction	Pancreatic Cancer	\
Sensitivity	0.962	1.000	0.984	
PPV	0.833	0.972	0.467	

	Ventricular Arrhythmia	Average
Sensitivity	0.981	0.985
PPV	0.777	0.746

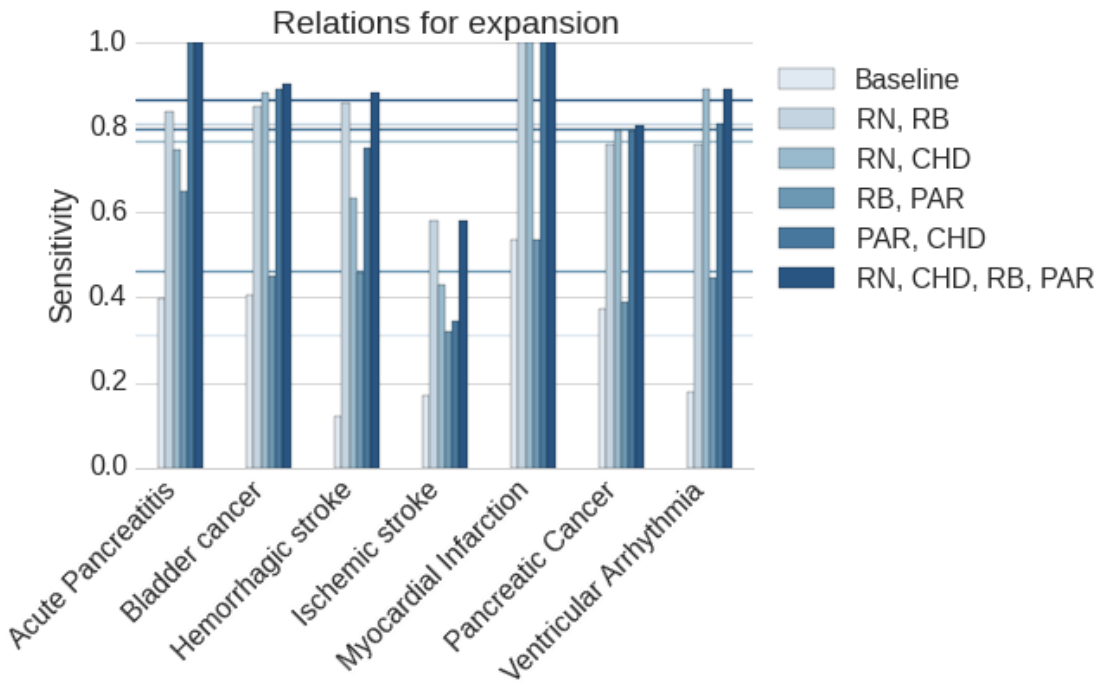


5 Compare relations for expansion

```
In [18]: 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()
         averages_compare = averages_compare[coding_systems]
         averages_compare['Average'] = averages_compare.mean(axis=1)

         with mystyle(graded_recall_palette(len(compare_variations), rev=0), savefig='relations-recall-
             ax = draw_lines(averages_compare['Average'])
             averages_compare.iloc[:, :-1].T.plot(kind='bar', title="Relations for expansion", ax=ax)
             plt.ylabel(measure_label('recall'))
```



In [19]:
```
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()
averages_compare['Average'] = averages_compare.mean(axis=1)

with mystyle(graded_recall_palette(len(compare_variations), rev=0), xrot=45, ha='right', savefi
    ax = draw_lines(averages_compare['Average'])
    averages_compare.iloc[:,:-1].T.plot(kind='bar', title="Relations for expansion", ax=ax)
    plt.ylabel(measure_label('recall'))
```



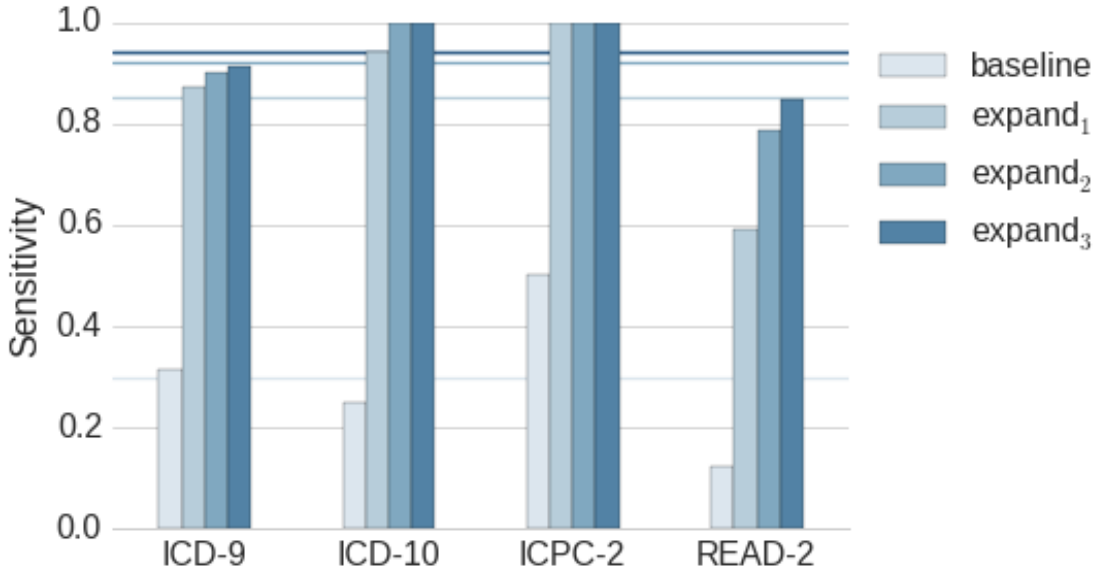
6 Increasing sensitivity with more expansion steps

```
In [20]: variations_names = OrderedDict([
             ('baseline', 'baseline'),
             ('1-RN-CHD-RB-PAR.expand', 'expand$_1$'),
             ('2-RN-CHD-RB-PAR.expand', 'expand$_2$'),
             ('3-RN-CHD-RB-PAR.expand', 'expand$_3$'),
             ('4-RN-CHD-RB-PAR.expand', 'expand$_4$'),
         ])
         df = pd.DataFrame({
             name: ev[ev.variation == variation].groupby('database').recall.mean()
             for variation, name in variations_names.items()
         }).T
         df.columns = df.columns.map(database_label)
         df = df[coding_systems]
         df['Average'] = df.mean(axis=1)

         with mystyle(graded_recall_palette(len(variations_names), rev=0), savefig='steps-recall-by-db.p
             ax = draw_lines(df['Average'])
             df.iloc[:-1,:-1].T.plot(kind='bar', ax=ax)
             plt.ylabel(measure_label('recall'))

         df.round(3)
```

Out[20]:

	ICD-9	ICD-10	ICPC-2	READ-2	Average
baseline	0.316	0.249	0.5	0.124	0.297
expand$_1$	0.871	0.942	1.0	0.593	0.851
expand$_2$	0.900	1.000	1.0	0.787	0.922

expand$_3$	0.914	1.000	1.0	0.850	0.941
expand$_4$	0.914	1.000	1.0	0.851	0.941



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

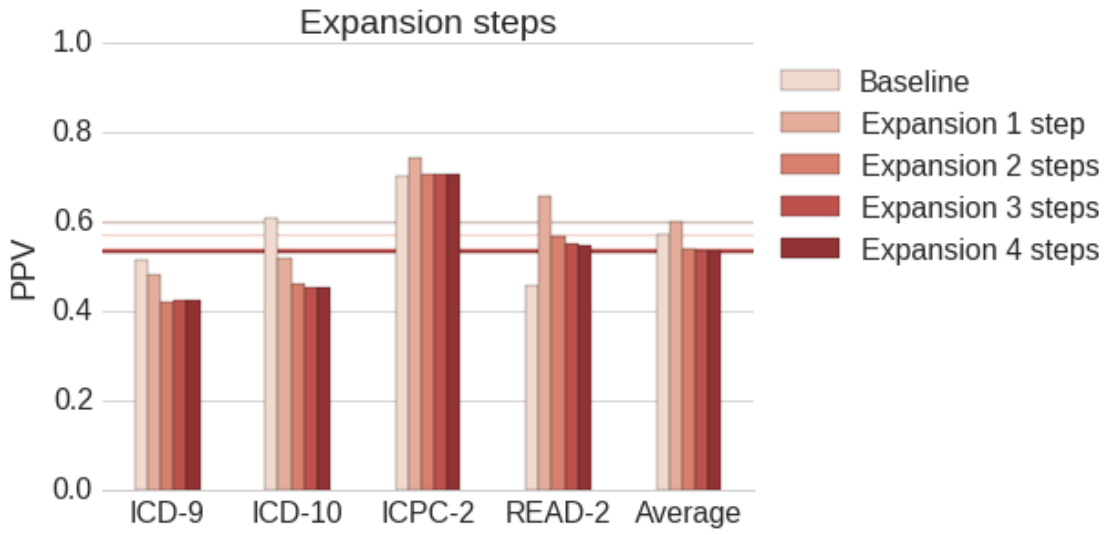
- K24 (Fear of heart attack)
- K90 (stroke)
- K93 (Pulmonary embolism)
- D70 (Dementia) OR "dementia" AND "infarct"
- U14 (Kidney symptom/complaint) OR "nier" AND "infarct"

```
In [21]: compare_variations = OrderedDict([
             ('baseline', 'Baseline'),
             ('1-RN-CHD-RB-PAR.expand', 'Expansion 1 step'),
             ('2-RN-CHD-RB-PAR.expand', 'Expansion 2 steps'),
             ('3-RN-CHD-RB-PAR.expand', 'Expansion 3 steps'),
             ('4-RN-CHD-RB-PAR.expand', 'Expansion 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()
         averages_compare = averages_compare[coding_systems]
         averages_compare['Average'] = averages_compare.mean(axis=1)

         with mystyle(graded_precision_palette(len(compare_variations), rev=0), savefig='steps-precisio
             ax = draw_lines(averages_compare['Average'])
```

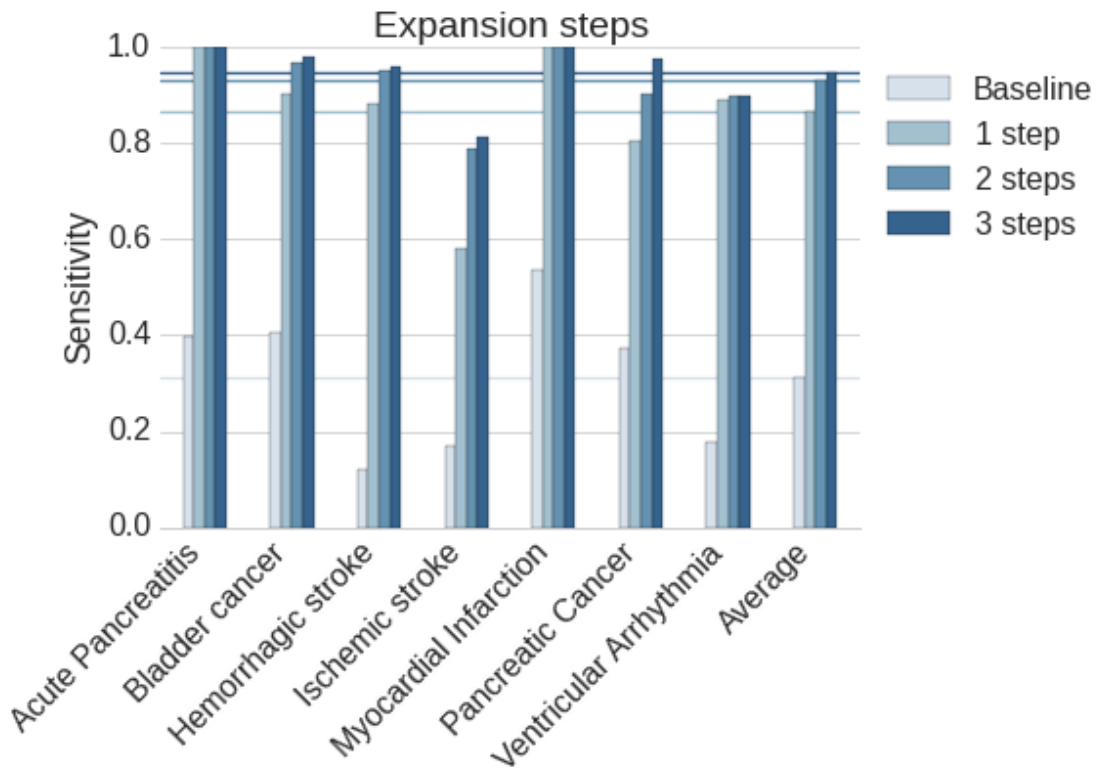
```
        averages_compare.T.plot(kind='bar', title="Expansion steps", ax=ax)
        plt.ylabel(measure_label('precision'))
    averages_compare.round(3)
```

Out[21]:

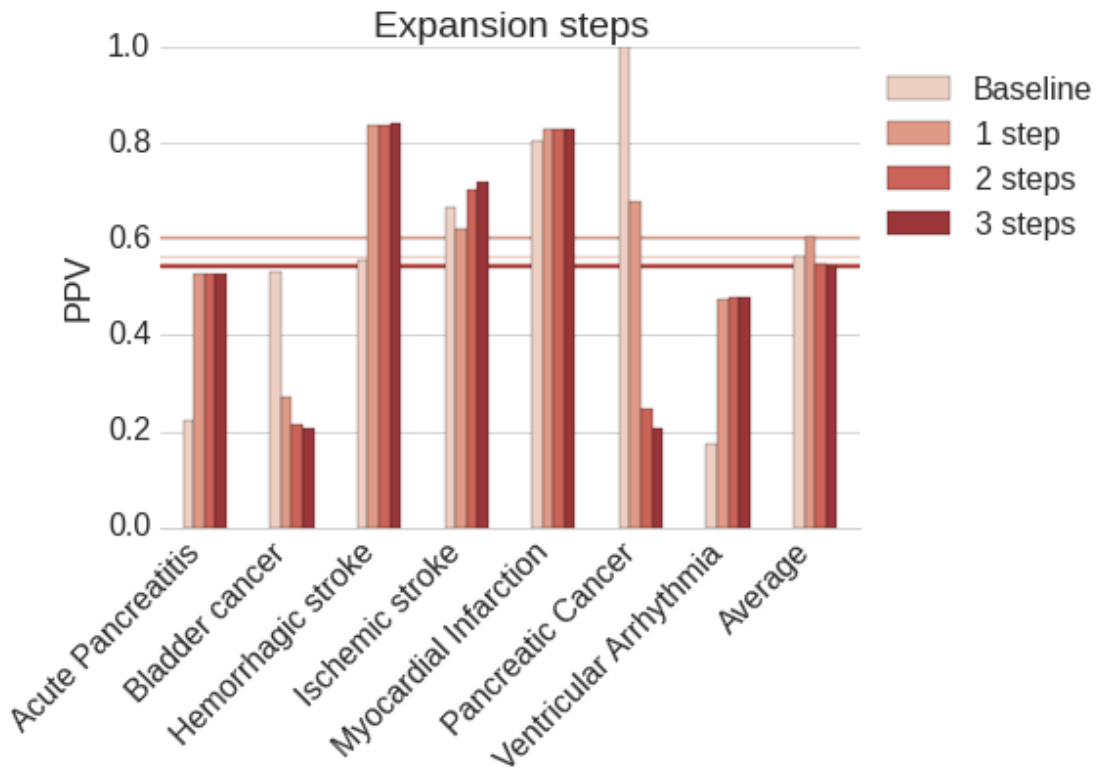
	ICD-9	ICD-10	ICPC-2	READ-2	Average
Baseline	0.514	0.606	0.700	0.458	0.570
Expansion 1 step	0.483	0.518	0.743	0.656	0.600
Expansion 2 steps	0.422	0.461	0.707	0.568	0.539
Expansion 3 steps	0.426	0.455	0.707	0.550	0.534
Expansion 4 steps	0.425	0.454	0.707	0.547	0.533



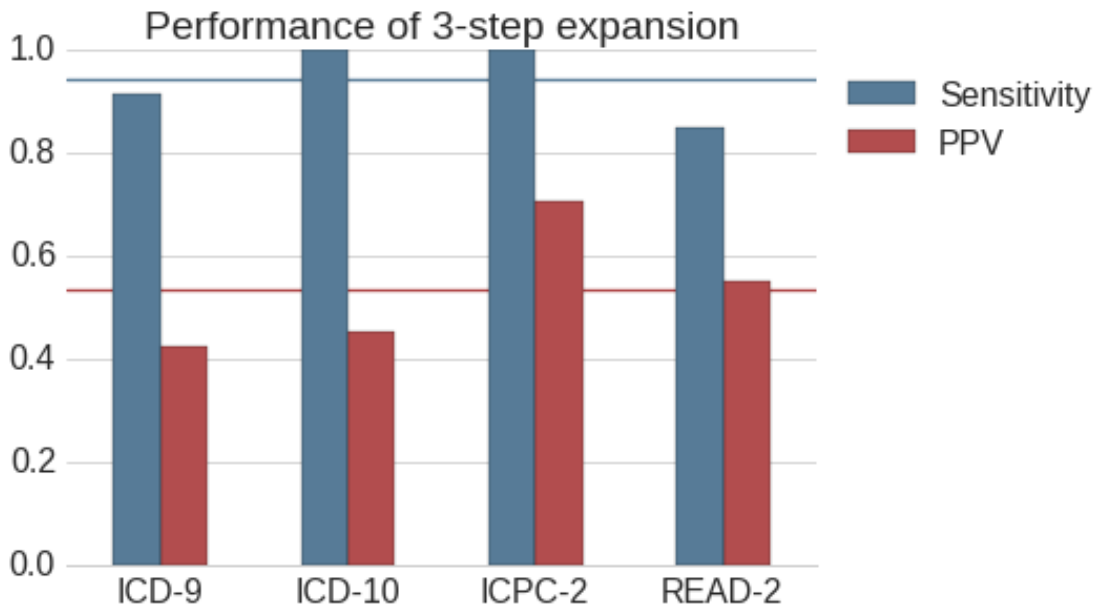
In [22]:
```
compare_variations = OrderedDict([
    ('baseline', 'Baseline'),
    ('1-RN-CHD-RB-PAR.expand', '1 step'),
    ('2-RN-CHD-RB-PAR.expand', '2 steps'),
    ('3-RN-CHD-RB-PAR.expand', '3 steps'),
#    ('4-RN-CHD-RB-PAR.expand', '4 steps'),
])
averages_compare = pd.DataFrame([
    ev[ev.variation == 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()
averages_compare['Average'] = averages_compare.mean(axis=1)

with mystyle(graded_recall_palette(len(compare_variations), rev=0), xrot=45, ha='right', savefi
    ax = draw_lines(averages_compare['Average'])
    averages_compare.T.plot(kind='bar', title="Expansion steps", ax=ax)
    plt.ylabel(measure_label('recall'))
```




```
In [23]: 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()
         averages_compare['Average'] = averages_compare.mean(axis=1)

         with mystyle(graded_precision_palette(len(compare_variations), rev=0), xrot=45, ha='right', sa
             ax = draw_lines(averages_compare['Average'])
             averages_compare.T.plot(kind='bar', title="Expansion steps", ax=ax)
             plt.ylabel(measure_label('precision'))
```




```
In [24]: 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)
         averages_compare = averages_compare[coding_systems]
         averages_compare['Average'] = averages_compare.mean(axis=1)

         name = 'expansion3-performance-by-db'
         with mystyle(measures_palette, savefig=name+'.pdf'):
             ax = draw_lines(averages_compare['Average'])
             averages_compare.iloc[:,:-1].T.plot(kind='bar', title="Performance of 3-step expansion", a
         averages_compare.to_csv(name+'.csv')
```



7 FN error-analysis

In [25]:
```
variation = '3-RN-CHD-RB-PAR.expand'

with open("../{}.{}.error-analyses.yaml".format(PROJECT, variation)) as f:
    error_analyses = yaml.load(f)

def get_category(fn_or_fp, database, event, code):
    if database in error_analyses[fn_or_fp] and event in error_analyses[fn_or_fp][database]:
        return error_analyses[fn_or_fp][database][event]['code-categories'].get(code) or '?'
    else:
        return '??'

evs = ev[(ev.variation == variation) & ev.fn.notnull()][['event', 'database', 'fn', 'fp']]

fn = evs.apply(lambda row: pd.Series(row.fn), axis=1).stack().reset_index(level=1, drop=True)
fn.name = 'code'
# fns : | event | database | code |
fns = evs.drop(['fn', 'fp'], axis=1).join(fn, how='inner').drop_duplicates()
fns['category'] = fns.apply(lambda r: get_category('fn', r.database, r.event, r.code), axis=1)

fp = evs.apply(lambda row: pd.Series(row.fp), axis=1).stack().reset_index(level=1, drop=True)
fp.name = 'code'
# fps : | event | database | code |
fps = evs.drop(['fn', 'fp'], axis=1).join(fp, how='inner').drop_duplicates()
fps['category'] = fps.apply(lambda r: get_category('fp', r.database, r.event, r.code), axis=1)

fns.groupby(['category', 'database']).code.aggregate(lambda s: set(s)).map(', '.join).to_frame
```

Out[25]:

category	database

```
database-specific CPRD            853Z., 8532., 70042, 7L1H., 853.., 8531., 70041, 70082, 7L1
                  Medicare                                                          433.81, 434.01, 4
next-expansion    CPRD
not-in-dnf        CPRD      BBaz., ByuE0, 1O0.., G60X., 7L1H7, G6W.., BBd9., BBa.., 7L1H6, G6X
```

In [26]:
```
fps.groupby(['category', 'database']).code.aggregate(lambda s: set(s)).map(', '.join).to_frame
```

Out[26]:
```
category database
in-dnf   CPRD      BBJz., BBP.., BBR.., B151., K3..., B8301, BB90., B1512, BBcC., B717z, BBcD.
         GePaRD    D09.9, D01.9, D09.8, I46, C80, C49.3, C75, C90.3, C4A, I47.9, C75.9, I49.01
         IPCI
         Medicare  184.9, 234.8, 236.9, 209, 235.5, 149.0, 155.2, 238.8, 159.0, 236, 434.9, 19
other-fp CPRD      J67z., 1969., R1043, BB52., R090., R090z, R0003, R050., B7D3., Ryu06, J67..
         GePaRD    C64.-C68.9, D00.-D09, C73.-C75, K90.-K93.9, C45.-C49.9, D30.3, K86.9, C73.-C
         IPCI
         Medicare  789.0, 789.00, 427, 190.-199.99, 577.9, 427.9, 430.-438.99, 577, 140.-239.9
```

In [27]:
```
code_counts = pd.Series({
    database: len(set(mappings.all_codes(database)))
    for database in databases.databases()
})
code_counts.ix['All'] = code_counts.sum()
code_counts.index.name = 'database'

def category_label(category):
    return {
        # FN
        'not-in-dnf': 'Not in UMLS',
        'database-specific': 'DB specific',
        'next-expansion': 'expansion_{4}',
        'isolated': 'Isolated',
        # FP
        'in-dnf': 'Cosynonym',
        'other-fp': 'Indexing FP',
    }.get(category, category)

def counts(code_categories, FN_or_FP):
    "code_categories : | code | category |"
    # (database, category) | int
    s1 = code_categories.groupby('database').category.value_counts()
    # category | int
    s2 = code_categories.category.value_counts()
    s2.index = pd.MultiIndex.from_product([['Overall'], s2.index])
    res = pd.concat([s1, s2]).to_frame('count')
    res['%'] = (res['count'] / s2.sum()).map('{:.1%}'.format)
    #res['% (mapping)'] = (res['count'] / code_counts).map('{:.1%}'.format)

    res = res.rename(columns={'count': '{} category'.format(FN_or_FP)}).reset_index()
    res['category'] = res['category'].map(category_label)
    res['database'] = res['database'].map(lambda db: db if db == 'Overall' else database_label
    return res

counts(fps, 'FP')
```

Out[27]:

	database	category	FP category	%
0	READ-2	Cosynonym	394	53.5%
1	READ-2	Indexing FP	20	2.7%
2	ICD-10	Cosynonym	152	20.6%
3	ICD-10	Indexing FP	47	6.4%
4	ICPC-2	Cosynonym	7	0.9%
5	ICPC-2	Indexing FP	1	0.1%
6	ICD-9	Cosynonym	91	12.3%
7	ICD-9	Indexing FP	25	3.4%
8	Overall	Cosynonym	644	87.4%
9	Overall	Indexing FP	93	12.6%

In [28]:
```
counts(fns, 'FN')
```

Out[28]:

	database	category	FN category	%
0	READ-2	Not in UMLS	21	47.7%
1	READ-2	DB specific	13	29.5%
2	READ-2	expansion_{4}	4	9.1%
3	ICD-9	DB specific	6	13.6%
4	Overall	Not in UMLS	21	47.7%
5	Overall	DB specific	19	43.2%
6	Overall	expansion_{4}	4	9.1%

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

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)[:-1])
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")
```

```
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_inclusio
residuals.columns = ["Database", "Event", "Recall", "Missed", "Residual FNs"]

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

8 Removing unused codes

```
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-precision-by
    averages_compare.T.plot(kind='bar', title="Filtered baseline")
    plt.ylabel(measure_label('precision'))
```

```
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='filtered-
    averages_compare.T.plot(kind='bar', title="Filtered baseline")
    plt.ylabel(measure_label('precision'))

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-precision-
    averages_compare.T.plot(kind='bar', title="Filtered expansion")
    plt.ylabel(measure_label('precision'))

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='filtered-expa
    averages_compare.T.plot(kind='bar', title="Filtered 3-step expansion")
    plt.ylabel(measure_label('recall'))

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='filtered-
    averages_compare.T.plot(kind='bar', title="Filtered 3-step expansion")
    plt.ylabel(measure_label('precision'))

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")
```

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.

9 Codes in reference mappings, not in databases

Codes that might be removed from the TP when filtering.

```
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
```