

# AWB (Anwendungsbeobachtungen, Observational Studies)

This notebook reads in and cleans the received AWB data and shows some basic analysis.

```
In [1]: from __future__ import division
import glob
from datetime import datetime, date, timedelta
import os
import itertools

import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from pandas import DataFrame, Series
import pandas as pd

import seaborn as sns

%matplotlib inline
matplotlib.rcParams['svg.fonttype'] = 'none'

pd.options.display.max_rows = 150
```

```

In [2]: filenames = [('data/AWB KBV Meldungen und Abschlüsse 2004 - 2011.xlsx', range(
2009, 2012)),
('data/AWB KBV Meldungen und Abschlüsse 2012 - 2014.xlsx', (2012,
2013, 2014))]

def read_excel(filename, years, kind='update', needle='Meldungen'):
    xl_file = pd.ExcelFile(filename)
    sheet_names = xl_file.sheet_names

    for year in years:
        sheet_name = [x for x in sheet_names if str(year) in x and needle in x
][0]
        print(filename, sheet_name)
        df = xl_file.parse(sheet_name)
        # Make index into row number column
        df = df.reset_index()
        df['year'] = year
        df['row_type'] = kind
        df = df.rename(columns=dict([(x, x.strip()) for x in df.columns if x.s
trip() != x]))
        # Consolidate column names
        df = df.rename(columns={u'Präparat': u'Präparatname',
u'Präparat/Titel der Anwendung': u'Präparatnam
e',
u'gemeldet am': 'DatumErstanzeige',
u'Datum Erstanzeige': 'DatumErstanzeige',
u'Beobachtungsplan vorliegend': 'Beobachtungspl
an vorliegend',
u'Ärzte gemeldet': u'gemeldete Ärzte',
u'Anzahl teilnehmende Ärzte (wenn angegeben)':
u'gemeldete Ärzte',
u'Anzahl der beobachtenden Ärzte': u'beobachte
nde Ärzte',
'index': 'row_number'
}))
        if 'DatumErstanzeige' not in df.columns:
            df = df.rename(columns={'Eingang': 'DatumErstanzeige'})
        # Fix date columns
        date_cols = list(df.columns[df.columns.str.startswith('Datum')])
        for x in date_cols:
            df['dt_%s' % x] = pd.to_datetime(df[x], errors='coerce')
        df[u'Präparatname'] = df[u'Präparatname'].str.strip()
        # Remove entries with empty drug name
        df = df[df[u'Präparatname'].notnull()]
        yield df

```

First, read in all available update messages.

```

In [3]: num_cols = ['Patienten geplant', 'Patienten beobachtet', u'gemeldete Ärzte', u
'beobachtende Ärzte',
u'Vertragsärzte', u'Aufwandsentschädigung pro Patient']
float_cols = [u'Aufwandsentschädigung pro Patient']

```

```
In [4]: df_updates = pd.concat(itertools.chain(*[read_excel(*args) for args in filenames]))

# To be compatible with Abschluesse column
df_updates['Aufwandsentschädigung gesamt in €'] = None

print('Number of rows', len(df_updates))
df_updates.head()
```

data/AWB KBV Meldungen und Abschlüsse 2004 - 2011.xlsx Meldungen 2009 Gesamt  
 data/AWB KBV Meldungen und Abschlüsse 2004 - 2011.xlsx Meldungen 2010 Gesamt  
 data/AWB KBV Meldungen und Abschlüsse 2004 - 2011.xlsx Meldungen 2011 Gesamt  
 data/AWB KBV Meldungen und Abschlüsse 2012 - 2014.xlsx 2012 Meldungen gesamt  
 data/AWB KBV Meldungen und Abschlüsse 2012 - 2014.xlsx 2013 Meldungen gesamt  
 data/AWB KBV Meldungen und Abschlüsse 2012 - 2014.xlsx 2014 Meldungen gesamt  
 Number of rows 13829

```
Out[4]:
```

	Art der NIS	Auftraggeber	Aufwandsentschädigung Kommentar	Aufwandsentschädigung pro Patient	Beobachtungsplan vorliegend	Beo
0	NaN	NaN	pro vollständig ausgefüllten Dokumentationsbog...	900	NaN	
1	NaN	NaN	pro vollständig dokumentierter Eingangsvisite,...	150	NaN	
2	NaN	NaN	Meldung vor Anzeigenpflicht	NaN	NaN	
3	NaN	NaN	Meldung vor Anzeigenpflicht	NaN	NaN	
4	NaN	NaN	NaN	20	NaN	

5 rows × 39 columns

```
In [5]: # Catch bogus row that is way down and contains a different header
# Interesting colum "HonorarPlausibilität" (plausibility of fee) which is not
# available in our dataset
bad_series = df_updates[df_updates['Art der NIS'] == 'Art der NIS'].T.iloc[:,0]
# Remove the row before processing further
df_updates = df_updates[~(df_updates['Art der NIS'] == 'Art der NIS')]
bad_series
```

```
Out[5]: Art der NIS
Auftraggeber
Aufwandsentschädigung Kommentar
Aufwandsentschädigung pro Patient
Beobachtungsplan vorliegend
BeobachtungszeitraumKommentar
Brief/Mail/Fax
DatumAbmeldung
DatumBrief
DatumEingang
DatumEnde
DatumErstanzeige
DatumStart
Firma
Kommentar
MeldungsGrund
MeldungsKommentar
Meldungsart
Meldungsinhalt
Patienten beobachtet
Patienten geplant
Präparatname
Titel (Ziel)
Typ
Vertrag vorliegend
Vertragsärzte
Wirkstoff
beobachtende Ärzte
dt_DatumAbmeldung
dt_DatumBrief
dt_DatumEingang
dt_DatumEnde
dt_DatumErstanzeige
dt_DatumStart
gemeldete Ärzte
row_number
row_type
year
Aufwandsentschädigung gesamt in €
Name: 14947, dtype: object

Art der NIS
Auftraggeber
Anzahl der Ärzte Gesamt
AnzahlPatienten
HonorarPlausibilität
BeobachtungszeitraumKommentar
DatumEingang
Meldungsart
Brief/Mail/Fax
ATC
DatumEnde
DatumErstanzeige
DatumStart
Firma
NaN
NaN
NaN
DatumBrief
Meldezeitpunkt: Monat/Quartal/Jahr
Meldungsinhalt
NaN
Präparatname
Titel
Typ
Honorar pro Patient
NaN
Wirkstoff
NaN
NaT
NaT
NaT
NaT
NaT
NaT
DatumAbmeldung
14947
update
2013
None
```

Read in all final notices.

```
In [6]: abschluesse_df = pd.concat(itertools.chain(*[read_excel(*args, needle='Abschl',
, kind='final') for args in filenames]))

abschluesse_df

# Make columns compatible with update notices
columns = u'dt_DatumErstanzeige dt_DatumStart dt_DatumEingang Präparatname
Wirkstoff Firma Patienten beobachtet Patienten geplant beobac
htende Ärzte gemeldete Ärzte Aufwandsentschädigung pro Patient Aufwan
dsentschädigung Kommentar Aufwandsentschädigung gesamt in € Art de
r NIS Auftraggeber Beobachtungsplan vorliegend BeobachtungszeitraumKo
mmentar Brief/Mail/Fax DatumAbmeldung DatumBrief DatumEingang DatumE
nde Kommentar MeldungsGrund MeldungsKommentar Meldungsart
Meldungsinhalt Titel (Ziel) Typ Vertrag vorliegend Vertragsärzte
dt_DatumAbmeldung dt_DatumBrief dt_DatumEnde year'.split('\t')
for c in columns:
    if c not in abschluesse_df:
        abschluesse_df[c] = None

print('Number of rows', len(abschluesse_df))
abschluesse_df.head()
```

```
data/AWB KBV Meldungen und Abschlüsse 2004 - 2011.xlsx Abschlüsse 2009
data/AWB KBV Meldungen und Abschlüsse 2004 - 2011.xlsx Abschlüsse 2010
data/AWB KBV Meldungen und Abschlüsse 2004 - 2011.xlsx Abschlüsse 2011
data/AWB KBV Meldungen und Abschlüsse 2012 - 2014.xlsx 2012 Abschlüsse
data/AWB KBV Meldungen und Abschlüsse 2012 - 2014.xlsx 2013 Abschlüsse
data/AWB KBV Meldungen und Abschlüsse 2012 - 2014.xlsx 2014 Abschlüsse
Number of rows 365
```

Out[6]:

	Art der NIS	Auftraggeber	Aufwandsentschädigung Kommentar	Aufwandsentschädigung gesamt in €	Aufwandsentschädigung pro Patient
0	NaN	NaN	keine Aufwandsentschädigung angegeben	k.A.	NaN
1	NaN	NaN	pro Patient	97743.2	80
2	NaN	NaN	keine Aufwandsentschädigung angegeben	k.A.	NaN
3	NaN	Janssen- Cilag GmbH	pro dokumentierten Patienten	k.A.	75
4	NaN	NaN	pro Patient	k.A.	90

5 rows × 39 columns

```
In [7]: # Same as before, remove bogus header row at line 12607 of final notices in year 2013
bad_series = abschluesse_df[abschluesse_df['Art der NIS'] == 'Art der NIS'].T.
iloc[:,0]
# Remove the row before processing further
abschluesse_df = abschluesse_df[~(abschluesse_df['Art der NIS'] == 'Art der NIS')]
bad_series
```

```
Out[7]: Art der NIS          Art der NIS
Auftraggeber          Auftraggeber
Aufwandsentschädigung Kommentar    Anzahl der Ärzte Gesamt
Aufwandsentschädigung gesamt in €          NaN
Aufwandsentschädigung pro Patient    AnzahlPatienten
Beobachtungsplan vorliegend    HonorarPlausibilität
BeobachtungszeitraumKommentar    BeobachtungszeitraumKommentar
Brief/Mail/Fax          DatumEingang
DatumAbmeldung          Meldungsart
DatumBrief          Brief/Mail/Fax
DatumEingang          ATC
DatumEnde          DatumEnde
DatumErstanzeige    DatumErstanzeige
DatumStart          DatumStart
Firma          Firma
Kommentar          NaN
MeldungsGrund          NaN
MeldungsKommentar    NaN
Meldungsart          DatumBrief
Meldungsinhalt    Meldezeitpunkt: Monat/Quartal/Jahr
Patienten beobachtet    Meldungsinhalt
Patienten geplant    NaN
Präparatname    Präparatname
Titel (Ziel)    Titel
Typ    Typ
Vertrag vorliegend    Honorar pro Patient
Vertragsärzte    NaN
Wirkstoff    Wirkstoff
beobachtende Ärzte    NaN
dt_DatumAbmeldung    NaT
dt_DatumBrief    NaT
dt_DatumEingang    NaT
dt_DatumEnde    NaT
dt_DatumErstanzeige    NaT
dt_DatumStart    NaT
gemeldete Ärzte    DatumAbmeldung
row_number    12605
row_type    final
year    2013
Name: 12605, dtype: object
```

Get cleaner number representation of total amount.

```
In [8]: import re
import numbers

NUMBER_RE = re.compile('^s*([\d\., ]+)')
NUMBERS_RE = {
    re.compile(r'^([\d\., ]+)(\d{1,2}+)' ): '.',
    re.compile(r'^([\d, ]+)\.(\d{1,2}+)' ): ',',
}

def clean_money(x):
    if isinstance(x, numbers.Number):
        return x
    x = NUMBER_RE.sub('\\1', x)
    for reg, repl in NUMBERS_RE.items():
        m = reg.search(x)
        if m is None:
            continue
        before = int(m.group(1).replace(repl, ''))
        after = int(m.group(2))
        if after < 10:
            after = after / 10.0
        else:
            after = after / 100.0
        return before + after
    return None

abschluesse_df['Aufwandsentschädigung gesamt'] = abschluesse_df['Aufwandsentsc
hädigung gesamt in €'].apply(clean_money)
abschluesse_df[['Aufwandsentschädigung gesamt', 'Aufwandsentschädigung gesamt
in €']].head()
```

```
Out[8]:
```

	Aufwandsentschädigung gesamt	Aufwandsentschädigung gesamt in €
0	NaN	k.A.
1	97743.2	97743.2
2	NaN	k.A.
3	NaN	k.A.
4	NaN	k.A.

```
In [9]: grouper = ['dt_DatumErstanzeige', 'dt_DatumStart']

# Fill missing values in grouping columns with dummy value,
# so it's not silently dropped by pandas groupby
dummy_date = pd.to_datetime(date(1900, 1, 1))
abschluesse_df[grouper] = abschluesse_df[grouper].fillna(dummy_date)
df_updates[grouper] = df_updates[grouper].fillna(dummy_date)

assert not abschluesse_df[grouper].isnull().any().any()
assert not df_updates[grouper].isnull().any().any()
```

Combine update notices and final notices.

```
In [10]: df_all = pd.concat([df_updates, abschlusse_df])
df_all = df_all.reset_index(drop=True)

df_all['row_type'].value_counts()
```

```
Out[10]: update      13828
final         364
Name: row_type, dtype: int64
```

```
In [11]: # Add simpler version of präparatname that might group better later

DRUG_NAME_SPLITTER = re.compile(r'^\w+|\d|_', re.U | re.I)

def clean_praeparat(praeparat):
    name = DRUG_NAME_SPLITTER.split(praeparat)[0].strip().lower()
    if len(name) < 4:
        return praeparat
    return name

df_all['praeparat'] = df_all[u'Präparatname'].apply(clean_praeparat)

print('Original Präparatname Number of Groups', len(df_all[u'Präparatname'].value_counts()))
print('Cleaned Präparatname Number of Groups', len(df_all['praeparat'].value_counts()))
```

```
Original Präparatname Number of Groups 997
Cleaned Präparatname Number of Groups 813
```

```
In [12]: for name in num_cols:
    new_name = 'num_%s' % name
    df_all[new_name] = df_all[name].copy()
    df_all[new_name] = df_all[new_name].apply(str)

    if name not in float_cols:
        df_all[new_name] = (df_all[new_name]
                             .str.replace('(geplante Anzahl *?:|ca\.|max\.|geplant *?:)', '', flags=re.I)
                             .str.strip()
                             .str.replace(r'[ ,\.]', ''))
    df_all[new_name] = (df_all[new_name]
                        .str.replace('^\d+-(\d+)$', '\\1')
                        )
    if name not in float_cols:
        df_all[new_name] = (df_all[new_name]
                             .str.replace(r'^(\d+).*', '\\1', flags=re.I)
                             )
    df_all[new_name] = pd.to_numeric(df_all[new_name], errors='coerce')
```



```
In [40]: # Check if cleaning kind of worked
df_all[df_all['Patienten geplant'].str.contains(' ').fillna(False)][['Patienten geplant', 'num_Patienten geplant']].sample(10)
```

```
Out[40]:
```

	Patienten geplant	num_Patienten geplant
<b>8648</b>	700 (500)	700
<b>3807</b>	25 Patienten	25
<b>2981</b>	36000, vorher 20000	36000
<b>2584</b>	23000 Patienten	23000
<b>10036</b>	584 (weltweit?)	584
<b>5126</b>	15 000	15000
<b>991</b>	36 000	36000
<b>2430</b>	1060 Patienten	1060
<b>1888</b>	14 000	14000
<b>6488</b>	ca. 300, erhöht auf 600	300

## Analysis

Here's some exploratory analysis around the dataset.

First step is to group the single update and final notices into observational studies (AWB). We define the identification of one AWB to be the combination of its drug name, its registration date and its start date.

Per group of notices we find maximal numeric values for certain key figures and take the most prominent or last value for other columns.

```

In [14]: def get_best_value(series):
    vc = series.value_counts()
    if len(vc) == 0:
        lvi = series.last_valid_index()
        if lvi is None:
            return None
        return series[lvi]
    return vc.idxmax()

def get_awbs(groups):

    for key, rows in groups:
        # Use maximum number across columns and rows for one AWB
        patient_count = rows[['num_Patienten beobachtet', 'num_Patienten geplamt']].max().max()
        doc_count = rows[['num_beobachtende Ärzte', 'num_gemeldete Ärzte']].max().max()
        fee_per_patient = rows[['num_Aufwandsentschädigung pro Patient']].max().max()

        yield pd.DataFrame([
            'praeparat': key[0],
            'Präparatname': get_best_value(rows['Präparatname']),
            'dt_DatumErstanzeige': key[1],
            'dt_Start': key[2],
            'patient_count': patient_count,
            'doc_count': doc_count,
            'fee_per_patient': fee_per_patient,
            'calculated_total_fee': fee_per_patient * patient_count,
            'fee_comment': get_best_value(rows['Aufwandsentschädigung Kommentar']),
            'final_total_fee': rows['Aufwandsentschädigung gesamt'].max(),
            # Use most used values across AWB rows
            'Auftraggeber': get_best_value(rows['Auftraggeber']),
            'Firma': get_best_value(rows['Firma']),
            'Wirkstoff': get_best_value(rows['Wirkstoff']),
            'dt_DatumEnde': get_best_value(rows['dt_DatumEnde']),
        ])

awb_grouper = ['praeparat', 'dt_DatumErstanzeige', 'dt_DatumStart']
groups = df_all.sort_values(['dt_DatumEingang']).groupby(awb_grouper)
df_awb = pd.concat(get_awbs(groups))
df_awb = df_awb.reset_index(drop=True)
df_awb.head()

```

Out[14]:

	Auftraggeber	Firma	Präparatname	Wirkstoff	calculated_total_fee	do
0	None	LA-SER Europe Ltd.	(diverse Antiarrhythmika)	kein spezieller Wirkstoff	500000	
1	Lilly Deutschland GmbH	ICON plc (CRO)	(diverse Präparate), Alimta	diverse Wirkstoffe	193200	
2	Novartis Pharma GmbH	Lungenforschung	, Ultibro	Glycopyrroniumbromid, Indacaterol	5040000	
3	Beiersdorf AG	GKM Gesellschaft für Therapieforschung mbH	ABC® Wärme-Pflaster Capsicum 11 mg	Dickextrakt aus Cayennepfeffer (4-7:1) entspre...	20000	
4	None	IAS Dr. Jörg Schnitker GmbH	ALK-depot SQ 200	gereinigte Allergene aus verschiedenen Pollen	162000	

Number of extracted AWBs

In [15]: `len(df_awb)`

Out[15]: 1589

## Numbers of patients

In [16]: `df_awb['patient_count'].sum()`

Out[16]: 5270426.0

In [17]: `df_awb['patient_count'].describe()`

```
Out[17]: count      1120.000000
mean       4705.737500
std        51059.828867
min         0.000000
25%        200.000000
50%        500.000000
75%       1500.000000
max      1500000.000000
Name: patient_count, dtype: float64
```

## Numbers of doctors

```
In [18]: df_awb['doc_count'].sum()
```

```
Out[18]: 769844.0
```

```
In [19]: df_awb['doc_count'].describe()
```

```
Out[19]: count      868.000000  
mean      886.917051  
std      2615.096255  
min         0.000000  
25%       50.000000  
50%      215.000000  
75%      822.500000  
max     61500.000000  
Name: doc_count, dtype: float64
```

## Fee per Patient

```
In [20]: df_awb['fee_per_patient'].describe()
```

```
Out[20]: count      1409.000000  
mean      505.490823  
std       686.699514  
min         0.000000  
25%       100.000000  
50%       300.000000  
75%       650.000000  
max      7280.000000  
Name: fee_per_patient, dtype: float64
```

```
In [21]: df_awb['final_total_fee'].describe()
```

```
Out[21]: count         120.000000  
mean      217266.498133  
std      537010.092590  
min         0.000000  
25%      14430.000000  
50%      56212.500000  
75%     185256.675000  
max     4984295.000000  
Name: final_total_fee, dtype: float64
```

## How many AWBs over the years?

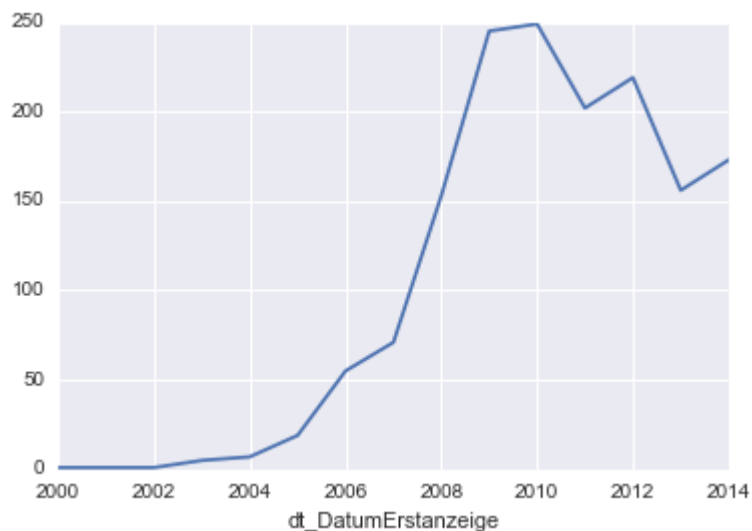
```
In [22]: awbs_per_year = df_awb.groupby(df_awb.dt_DatumErstanzeige.dt.year).size()

# Drop stupid values
awbs_per_year = awbs_per_year.drop([1900, 1905])
awbs_per_year
```

```
Out[22]: dt_DatumErstanzeige
2000      1
2001      1
2002      1
2003      5
2004      7
2005     19
2006     55
2007     71
2008    153
2009    245
2010    249
2011    202
2012    219
2013    156
2014    173
dtype: int64
```

```
In [23]: awbs_per_year.plot()
```

```
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x10de3cb38>
```



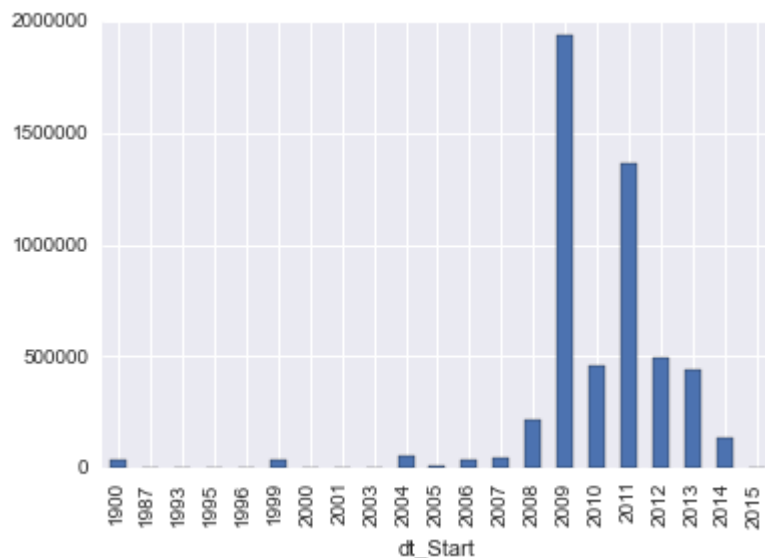
**How many patients were in the studies over the years?**

```
In [24]: patients_per_year = df_awb.groupby(df_awb.dt_Start.dt.year)['patient_count'].sum()
patients_per_year
```

```
Out[24]: dt_Start
1900      34978
1987         NaN
1993         NaN
1995      1000
1996         NaN
1999     36000
2000      3402
2001       400
2003        36
2004     52009
2005     14397
2006     33757
2007     45020
2008     216901
2009    1939378
2010     456893
2011    1365986
2012     494402
2013     442075
2014     133187
2015         605
Name: patient_count, dtype: float64
```

```
In [25]: patients_per_year.plot(kind='bar')
```

```
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x10d5a2f28>
```



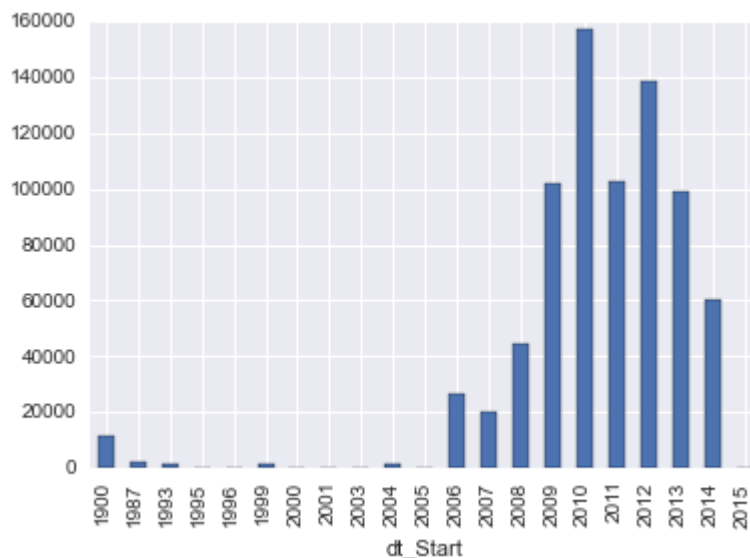
**How many doctors participated over the years?**

```
In [26]: doctors_per_year = df_awb.groupby(df_awb.dt_Start.dt.year)['doc_count'].sum()
doctors_per_year
```

```
Out[26]: dt_Start
1900      11385
1987       2020
1993       1240
1995        NaN
1996        NaN
1999       1380
2000        440
2001         10
2003        370
2004       1694
2005        241
2006      27014
2007      19940
2008      44914
2009     101600
2010     157077
2011     102957
2012     138402
2013      98724
2014     60236
2015         200
Name: doc_count, dtype: float64
```

```
In [27]: doctors_per_year.plot(kind='bar')
```

```
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x10d587dd8>
```



```
In [28]: awbv = ['Auftraggeber', 'Firma', 'Präparatname', 'doc_count', 'patient_count',
'final_total_fee', 'fee_per_patient',
'fee_comment', 'calculated_total_fee', 'praeparat', 'Wirkstoff', 'dt_D
atumEnde', 'dt_DatumErstanzeige',
'dt_Start']
```

## Highest patient count

In [29]: `df_awb.sort_values('patient_count', ascending=False).head(10)[awbv]`

Out[29]:

	Auftraggeber	Firma	Präparatname	doc_count	patient_count	final_total_fee	fee_per_
1533	None	Guerbet GmbH	Xenetix 250/300/350	1830	1500000	1136800	
329	None	Guerbet GmbH	Dotarem	64	750000	450110	
1246	None	Daiichi Sankyo	Sevikar	NaN	153610	NaN	
826	None	Agfa HealthCare Imaging Agents GmbH	Magnegita	0	150000	NaN	
1248	Daiichi Sankyo Deutschland GmbH	INC Research, vorher Kendle GmbH	Sevikar HCT	18920	130000	525560	
947	CHDI Foundation, Inc.	Quintiles GmbH	nicht zutreffend	NaN	100000	NaN	
331	None	Guerbet GmbH	Dotarem	150	100000	30480	
69	Chiesi Farmaceutici S.p.A.	Pierrel Research	alle zugelassenen Anti-Asthma Therapien	240	81500	NaN	
1249	Daiichi Sankyo Deutschland GmbH	CFC - Stolberg	Sevikar HCT	1436	65000	447500	
351	None	Takeda Pharma GmbH	Edarbi	4380	50000	409550	

## Highest fee per patient



```
In [30]: df_awb.sort_values('fee_per_patient', ascending=False).head(10)[awbv]
```

```
Out[30]:
```

	Auftraggeber	Firma	Präparatname	doc_count	patient_count	final_total_fee
<b>253</b>	Merck Serono SA-Geneva	Outcome Europe Sarl (CRO)	Cladribine	NaN	2000	NaN
<b>66</b>	Pfizer Pharma GmbH	Parexel International GmbH	alle mit Latanoprost sowie nicht- Latanoprost-h...	10	14	NaN
<b>928</b>	Schwarz Pharma Deutschland GmbH	UCB Biosciences GmbH	Neupro transdermales Pflaster	100	NaN	NaN
<b>1384</b>	Pfizer Pharma GmbH	Winicker Norimed GmbH	Torisel, Sutent, Inlyta	1890	1600	NaN
<b>626</b>	None	AbbVie Deutschland GmbH & Co. KG	HUMIRA	110	NaN	NaN
<b>625</b>	None	Abbott GmbH & Co. KG	Humira	NaN	NaN	NaN
<b>942</b>	Chiesi	CROMSOURCE GmbH	nicht- medikamentöse Beobachtungsstudie	30	89	NaN
<b>645</b>	Alimera Sciences Limited	medicomp	Iluvien	80	800	NaN
<b>636</b>	Baxter Innovations GmbH/Wien	Baxter Innovations GmbH/Wien	HyQvia	110	42	NaN
<b>239</b>	Baxter Healthcare	Dabio (MAPI S.A.S)	Ceprotrin	60	40	NaN

**Highest calculated total fee (fee per patient times number of patients)**

```
In [31]: df_awb.sort_values('calculated_total_fee', ascending=False).head(10)[awbv]
```

```
Out[31]:
```

	Auftraggeber	Firma	Präparatname	doc_count	patient_count	final_total_fee	fee_
<b>947</b>	CHDI Foundation, Inc.	Quintiles GmbH	nicht zutreffend	NaN	100000	NaN	
<b>804</b>	Novartis Pharma GmbH	clinicalmonitor (Quintiles)	Lucentis	200	30000	NaN	
<b>805</b>	None	Novartis Pharma GmbH	Lucentis	210	30000	NaN	
<b>809</b>	Novartis Pharma GmbH	Kantar Health GmbH	Lucentis	NaN	20000	NaN	
<b>1069</b>	None	Roche Pharma AG	Pegasys	6000	9790	NaN	
<b>69</b>	Chiesi Farmaceutici S.p.A.	Pierrel Research	alle zugelassenen Anti-Asthma Therapien	240	81500	NaN	
<b>999</b>	Thrombosis Research Institute (UK)	Parexel International GmbH	orale + andere Antikoagulanzen	NaN	50000	NaN	
<b>1533</b>	None	Guerbet GmbH	Xenetix 250/300/350	1830	1500000	1136800	
<b>943</b>	Thrombosis Research Institute (UK)	Parexel International GmbH	nicht zutreffend	NaN	50000	NaN	
<b>253</b>	Merck Serono SA-Geneva	Outcome Europe Sarl (CRO)	Cladribine	NaN	2000	NaN	

## Highest total final costs per AWB

```
In [32]: df_awb.sort_values('final_total_fee', ascending=False).head(10)[awbv]
```

```
Out[32]:
```

	Auftraggeber	Firma	Präparatname	doc_count	patient_count	final_total_fee	fee_per
<b>611</b>	None	Abbott GmbH & Co. KG	HUMIRA 40 mg	3510	5000	4984295.00	
<b>101</b>	None	AMGEN GmbH	Aranesp	620	950	2016638.00	
<b>1352</b>	None	Sanofi-Aventis Deutschland GmbH	Taxotere	4530	40000	1425312.75	
<b>203</b>	None	Roche Pharma AG	Bonviva , Fosamax , diverse Alendronat-Generika	2000	6009	1299733.75	
<b>1533</b>	None	Guerbet GmbH	Xenetix 250/300/350	1830	1500000	1136800.00	
<b>814</b>	None	Roche Pharma AG	MabThera	2160	5330	1073488.20	
<b>1517</b>	Shire Deutschland GmbH & Co. KG	POI Pharm-Olam International Deutschland GmbH	Xagrid	190	3647	790058.51	
<b>868</b>	None	Roche Pharma AG	Mircera	350	1500	678961.71	
<b>259</b>	None	Sanofi-Aventis Deutschland GmbH	Clexane 20 mg, 40 mg, 100 mg	530	816	614823.50	
<b>176</b>	CSL Behring GmbH	Chiltern International Ltd.	Berinert	5	113	600129.62	

## MabThera AWB

```
In [33]: v = ['dt_DatumEingang', 'Präparatname', 'Auftraggeber', 'Patienten beobachtet',
            'Patienten geplant',
            'Aufwandsentschädigung pro Patient', 'Aufwandsentschädigung gesamt in €',
            'row_type', 'year', 'row_number']
df_all[(df_all['Präparatname'] == 'MabThera') & (df_all['dt_DatumErstanzeige']
.dt.year == 2009) & (df_all['dt_DatumErstanzeige'].dt.month == 5)][v]
```

```
Out[33]:
```

	dt_DatumEingang	Präparatname	Auftraggeber	Patienten beobachtet	Patienten geplant	Aufwandsentschädi pro P
<b>960</b>	2009-06-08	MabThera	NaN	NaN	NaN	
<b>1159</b>	2009-07-10	MabThera	NaN	NaN	NaN	
<b>1346</b>	2009-08-17	MabThera	NaN	NaN	NaN	
<b>3603</b>	2010-08-16	MabThera	NaN	NaN	NaN	
<b>3864</b>	2010-09-30	MabThera	NaN	229	NaN	
<b>13879</b>	2010-09-30	MabThera	NaN	229	NaN	

```
In [34]: df_awb[df_awb['Präparatname'] == 'MabThera'][awbv]
```

```
Out[34]:
```

	Auftraggeber	Firma	Präparatname	doc_count	patient_count	final_total_fee	fee_per_
<b>814</b>	None	Roche Pharma AG	MabThera	2160	5330	1073488.2	
<b>815</b>	None	Roche Pharma AG	MabThera	460	229	65742.5	
<b>816</b>	Roche Pharma AG	proinnovera	MabThera	1380	NaN	NaN	
<b>817</b>	None	Roche Pharma AG	MabThera	NaN	NaN	NaN	
<b>818</b>	None	Roche Pharma AG	MabThera	90	NaN	NaN	
<b>819</b>	None	Roche Pharma AG	MabThera	NaN	NaN	NaN	
<b>820</b>	None	Roche Pharma AG	MabThera	2640	NaN	NaN	
<b>822</b>	None	Roche Pharma AG	MabThera	100	700	NaN	
<b>823</b>	Roche Pharma AG	AMS Advanced Medical Services GmbH	MabThera	1390	1076	NaN	

## Difference between calculated total fee and final total fee

Top 10 where final total is higher than calculated total costs.

```
In [42]: df_awb['final_calculated_diff'] = df_awb['calculated_total_fee'] - df_awb['final_total_fee']
df_awb.sort_values('final_calculated_diff').head(10)[['final_calculated_diff']
+ awbv]
```

Out[42]:	final_calculated_diff	Auftraggeber	Firma	Präparatname	doc_count	patient_co
<b>611</b>	-4234295.00	None	Abbott GmbH & Co. KG	HUMIRA 40 mg	3510	51
<b>1352</b>	-825312.75	None	Sanofi-Aventis Deutschland GmbH	Taxotere	4530	401
<b>176</b>	-534589.62	CSL Behring GmbH	Chiltern International Ltd.	Berinert	5	
<b>1350</b>	-186760.50	None	Sanofi-Aventis Deutschland GmbH	Taxotere 20mg/80mg Infusionslösung (Arzneimittel)	1450	,
<b>865</b>	-20196.63	None	Roche Pharma AG	Mircera	400	:
<b>200</b>	-10720.50	None	Takeda Pharma GmbH	Blopress forte 32 mg Plus 25	10110	41
<b>170</b>	-9156.00	GlaxoSmithKline	Kantar Health GmbH	Benlysta	210	.
<b>1560</b>	-8950.00	None	Actelion Pharmaceuticals Deutschland GmbH	Zavesca 100 mg Hartkapseln (Arzneimittel)	70	
<b>815</b>	-8492.50	None	Roche Pharma AG	MabThera	460	:
<b>821</b>	-7400.00	None	Roche Pharma AG	MabThera, Enbrel, Humira, Remicade	14	

Top 10 where final total is lower than calculated total costs.

```
In [43]: df_awb.sort_values('final_calculated_diff', ascending=False).head(10)[['final_calculated_diff'] + awbv]
```

Out[43]:

	final_calculated_diff	Auftraggeber	Firma	Präparatname	doc_count	patient_
<b>1533</b>	13863200.00	None	Guerbet GmbH	Xenetix 250/300/350	1830	15
<b>926</b>	13463449.87	UCB Pharma GmbH	PRA: Pharmaceutical Research Associates GmbH	Neupro	450	
<b>1517</b>	12878897.49	Shire Deutschland GmbH & Co. KG	POI Pharm-Olam International Deutschland GmbH	Xagrid	190	
<b>1248</b>	11174440.00	Daiichi Sankyo Deutschland GmbH	INC Research, vorher Kendle GmbH	Sevikar HCT	18920	1
<b>777</b>	7873880.00	None	Sanofi-Aventis Deutschland GmbH	Lantus	536	
<b>351</b>	7090450.00	None	Takeda Pharma GmbH	Edarbi	4380	
<b>329</b>	7049890.00	None	Guerbet GmbH	Dotarem	64	7
<b>1249</b>	5402500.00	Daiichi Sankyo Deutschland GmbH	CFC - Stolberg	Sevikar HCT	1436	
<b>1333</b>	4017745.20	Mundipharma gmbH	IfEG: Institut für Empirische Gesundheitsökonomie	Targin 10/20 Retard	133	
<b>569</b>	2352700.00	ALK-Abelló Arzneimittel GmbH	IAS Dr. Jörg Schnitker GmbH	Grazax, Avanz, ALK- depot SQ	4430	

Rough Top 10 final total fee by Firma

```
In [47]: df_awb.groupby('Firma')['final_total_fee'].sum().sort_values(ascending=False).head(10)
```

```
Out[47]: Firma
Abbott GmbH & Co. KG                4984295.00
Roche Pharma AG                    4573017.96
Sanofi-Aventis Deutschland GmbH    3297441.75
AMGEN GmbH                        2170981.22
Guerbet GmbH                      1819630.00
Takeda Pharma GmbH                1164780.50
POI Pharm-Olam International GmbH   790058.51
Chiltern International Ltd.         689886.62
Medidata GmbH                     590910.50
INC Research, vorher Kendle GmbH    525560.00
Name: final_total_fee, dtype: float64
```

## Analysis of fee comments

"Patient independent payments" and other interesting bits can be found.

```
In [35]: pd.set_option('max_colwidth', 120)
v = ['Präparatname', 'Auftraggeber', 'Patienten geplant', 'Aufwandsentschädigung pro Patient', 'Aufwandsentschädigung gesamt in €']
abschluesse_df[abschluesse_df['Aufwandsentschädigung gesamt in €'].str.contains('unabh').fillna(False)][v]
```

```
Out[35]:
```

	Präparatname	Auftraggeber	Patienten geplant	Aufwandsentschädigung pro Patient	Aufwandsentschädigung gesamt in €
<b>20</b>	Lantus	NaN	1600	100	149.465 incl. 20 mal 75€ für Qualitätsüberprüfung (patientenunabhängig)
<b>91</b>	Revlimid, Velcade	Celgene International Sàrl	NaN	743.75	16.734,39 (inklusive patientenunabhängig 743,75€ pro Arzt)

```
In [36]: df_all[df_all['Typ'] == 'Nahrungsergänzungsmittel'][v]
```

```
Out[36]:
```

	Präparatname	Auftraggeber	Patienten geplant	Aufwandsentschädigung pro Patient	Aufwandsentschädigung gesamt in €
<b>12748</b>	Cefamagar Tabletten	Cefak KG	350	150	None
<b>13361</b>	Cefamagar Tabletten	Cefak KG	494	150	None
<b>13565</b>	Cefamagar Tabletten	Cefak KG	494	150	None
<b>13706</b>	Cefamagar Tabletten	Cefak KG	493	150	None