In [53]:

```
%matplotlib inline
```

Añadir nuevas variables al dataset

1.- Librerias

In [54]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import re
import sklearn.preprocessing as pp

import dateutil
#Hay que istalar esta librería que hace el parseo del user agent
#pip install pyyaml ua-parser user-agents
from user_agents import parse

#Base maps -> mirar como instalarlo en la bibliografía al final del documen
to
#http://gnperdue.github.io/yak-shaving/osx/python/matplotlib/2014/05/01/bas
emap-toolkit.html
from mpl_toolkits.basemap import Basemap
```

2.- Descripcion de los datos

Usuarios

num_columna	Nombre	Descripción	Variable
1	id_usuarios	identificador únco del usuario	discreta
2	nombre	nombre complero del usuario	discreta
3	first_name	nombre del usuario	discreta
4	last_name	apellido del usuario	discreto
5	genero	genero del usuario	discreta
6	email	email del usuario	discreta
7	face_id	identificador de facebook	discreta
8	face_id_clasico	identificador clásico de facebook	discreta
9	googleid	identificador de google	discreta
10	foto	foto de perfil	discreta
11	es_foto_defecto	foto por defecto	discreta
12	fec_incorporacion	fecha de alta	discreta
13	pag_facebook	página de facebook	discreta
14	pag_gplus	página de google plus	discreta
15	locale	idioma del dispositivo	discreta
16	ciudad	ciudad del usuario	discreta
17	timezone	zona horaria del dispositivo	discreta
18	fecha_nacimiento	fecha de nacimiento del usuario	discreta

Visitas

num_columna	Nombre	Descripción	Variable
1	id_visitas	identificador de la visita	discreta
2	id_usuarios	identificador del usuario	discreta
3	id_hotspots	identificador del hotspot	discreta
4	MAC	Mac del dispositivo que se usó en la visita	discreta
5	ip	dirección ip **???**	discreta
6	max_sesion	**??**	
7	like	**??**	
8	post	**??**	
9	fecha_hora	fecha de la visita	discerta
10	agent	user agent del dispositivo	discreta

11	os	sistema operativo	discreta
12	mac_bridge	**?**	
13	via_visita	**?**	

3.- Carga de los datos

Vamos a unir estas dos tablas y definir la variable objetivo

In [55]:

```
usuarios = pd.read_csv('../csv/usuarios.csv')
visitas = pd.read_csv('../csv/visitas.csv')
coord_paises = pd.read_csv('../csv/Country_List_ISO_3166_Codes_Latitude_Lon
gitude.csv')
print usuarios.columns
print
print visitas.columns
```

```
Index([u'id_usuarios', u'nombre', u'first_name', u'last_name', u'genero',
u'email', u'face_id', u'face_id_clasico', u'googleid', u'foto', u'es_foto_d
efecto', u'fec_incorporacion', u'pag_facebook', u'pag_gplus', u'locale',
u'ciudad', u'timezone', u'fecha_nacimiento', u'min_age', u'max_age', u'birt
hday_gplus', u'email_existe'], dtype='object')

Index([u'id_visitas', u'id_usuarios', u'id_hotspots', u'MAC', u'ip', u'ma
x_sesion', u'like', u'post', u'fecha_hora', u'agent', u'os', u'mac_bridge',
u'via_visita'], dtype='object')
```

In [56]:

```
#contamos los usuarios por local y por hora

visita_usuario = visitas.join(usuarios
                              , on='id_usuarios'
                              , how='right'
                              , lsuffix='_left'
                              , rsuffix='_right')
print visita_usuario.head(2)
print visita_usuario.columns
```

```
   id_usuarios  id_visitas  id_usuarios_left  id_hotspots                MA
C  \
0            7           1                 7            2  44-74-6C-3F-83-E
B
1           16           2                16            2  1C-B0-94-36-B3-8
7

    ip  max_sesion  like  post           fecha_hora     ...       \
0  NaN         NaN   NaN   NaN  01/14/2015 20:01:20     ...
1  NaN         NaN   NaN   NaN  01/24/2015 20:57:12     ...

                                      pag_facebook pag_gplus  locale  \
0  https://www.facebook.com/profile.php?id=134739...       NaN     NaN
1  https://www.facebook.com/profile.php?id=660429709       NaN     NaN

   ciudad  timezone fecha_nacimiento min_age max_age birthday_gplus  \
0     NaN       NaN              NaN     NaN     NaN            NaN
1     NaN       NaN              NaN     NaN     NaN            NaN

  email_existe
0          NaN
1          NaN

[2 rows x 36 columns]
Index([u'id_usuarios', u'id_visitas', u'id_usuarios_left', u'id_hotspots',
u'MAC', u'ip', u'max_sesion', u'like', u'post', u'fecha_hora', u'agent',
u'os', u'mac_bridge', u'via_visita', u'id_usuarios_right', u'nombre', u'fir
st_name', u'last_name', u'genero', u'email', u'face_id', u'face_id_clasic
o', u'googleid', u'foto', u'es_foto_defecto', u'fec_incorporacion', u'pag_f
acebook', u'pag_gplus', u'locale', u'ciudad', u'timezone', u'fecha_nacimien
to', u'min_age', u'max_age', u'birthday_gplus', u'email_existe'], dtype='ob
ject')
```

In [57]:

```
print visita_usuario.shape
```

```
(26145, 36)
```

Nos quedamos con los datos que vamos a usar y les asignamos su tipo

In [58]:

```
#Estos campos probablemente no los vayamos a usar para nada, pero les ponemos su tipo.
visita_usuario.id_usuarios = visita_usuario.id_usuarios.astype('float64')
visita_usuario.id_visitas = visita_usuario.id_visitas.astype('float64')
visita_usuario.id_hotspots = visita_usuario.id_hotspots.astype('category')
visita_usuario.MAC = visita_usuario.MAC.astype('object')
visita_usuario.ip = visita_usuario.ip.astype('object')
visita_usuario.fecha_hora =  pd.to_datetime(visita_usuario.fecha_hora)
visita_usuario.agent = visita_usuario.agent.astype('object')
visita_usuario.os = visita_usuario.os.astype('category')
visita_usuario.nombre = visita_usuario.nombre.astype('object')
visita_usuario.first_name = visita_usuario.first_name.astype('object')
visita_usuario.last_name = visita_usuario.last_name.astype('object')
visita_usuario.genero = visita_usuario.genero.astype('category')
visita_usuario.email = visita_usuario.email.astype('object')
visita_usuario.fec_incorporacion = pd.to_datetime(visita_usuario.fec_incorporacion)
visita_usuario.locale = visita_usuario.locale.astype('category')
visita_usuario.ciudad = visita_usuario.ciudad.astype('category')
visita_usuario.timezone = visita_usuario.timezone.astype('category')
visita_usuario.fecha_nacimiento = pd.to_datetime(visita_usuario.fecha_nacimiento)
visita_usuario.birthday_gplus = pd.to_datetime(visita_usuario.birthday_gplus)
```

In [59]:

```
print "REGISTROS TOTALES:" , len(visita_usuario)
print
#Vemos los unicos y los nulos
print "USUARIOS UNICOS:"
print "------------"
print "usuarios_unicos:" , len(visita_usuario.id_usuarios.unique())
print "Registros con usuario:" , len(visita_usuario[pd.notnull(visita_usuar
io.id_usuarios)])
print "Registros sin usuario:" , len(visita_usuario[pd.isnull(visita_usuari
o.id_usuarios)])
print
print "HOTSPOSTS"
print "------------"
print "número de hotspots:" , len(visita_usuario.id_hotspots.unique())
print "Registros con HOTSPOT:" , len(visita_usuario[pd.notnull(visita_usuar
io.id_hotspots)])
print "Registros sin HOTSPOT:" , len(visita_usuario[pd.isnull(visita_usuari
o.id_hotspots)])
print
print "MACS:"
print "------------"
print "Número de MACS: " ,len(visita_usuario.MAC.unique())
print "Registros con MAC:" , len(visita_usuario[pd.notnull(visita_usuario.M
AC)])
print "Registros sin MAC:" , len(visita_usuario[pd.isnull(visita_usuario.MA
C)])
print
print "IPS:"
print "------------"
print "Número de IPS: " ,len(visita_usuario.ip.unique())
print "Registros con IPS:" , len(visita_usuario[pd.notnull(visita_usuario.i
p)])
print "Registros sin IPS:" , len(visita_usuario[pd.isnull(visita_usuario.i
p)])
print
print "Fecha_hora:"
print "------------"
print "Número de Fechas: " ,len(visita_usuario.fecha_hora.unique())
print "Registros con Fecha_hora:" , len(visita_usuario[pd.notnull(visita_us
uario.fecha_hora)])
print "Registros sin Fecha_hora:" , len(visita_usuario[pd.isnull(visita_usu
ario.fecha_hora)])
print
print "User_aget:"
print "------------"
print "Número de ua: " ,len(visita_usuario.agent.unique())
print "Registros con ua:" , len(visita_usuario[pd.notnull(visita_usuario.ag
ent)])
print "Registros sin ua:" , len(visita_usuario[pd.isnull(visita_usuario.age
nt)])
print
print "OS:"
print "------------"
print "Número de os: " ,len(visita_usuario.os.unique())
print "Registros con os:" , len(visita_usuario[pd.notnull(visita_usuario.o
```

```
s)])
print "Registros sin os:" , len(visita_usuario[pd.isnull(visita_usuario.o
s)])
print
print "Nombre:"
print "------------"
print "Número de nombre: " ,len(visita_usuario.nombre.unique())
print "Registros con nombre:" , len(visita_usuario[pd.notnull(visita_usuari
o.nombre)])
print "Registros sin nombre:" , len(visita_usuario[pd.isnull(visita_usuari
o.nombre)])
print
print "First Name:"
print "------------"
print "Número de FN: " ,len(visita_usuario.first_name.unique())
print "Registros con FN:" , len(visita_usuario[pd.notnull(visita_usuario.fi
rst_name)])
print "Registros sin FN:" , len(visita_usuario[pd.isnull(visita_usuario.fir
st_name)])
print
print "Last Name:"
print "------------"
print "Número de LN: " ,len(visita_usuario.last_name.unique())
print "Registros con LN:" , len(visita_usuario[pd.notnull(visita_usuario.la
st_name)])
print "Registros sin LN:" , len(visita_usuario[pd.isnull(visita_usuario.las
t_name)])
print
print "Email:"
print "------------"
print "Número de email: " ,len(visita_usuario.email.unique())
print "Registros con email:" , len(visita_usuario[pd.notnull(visita_usuari
o.email)])
print "Registros sin email:" , len(visita_usuario[pd.isnull(visita_usuari
o.email)])
print
print "Fecha alta:"
print "------------"
print "Número de Fechas alta: " ,len(visita_usuario.fec_incorporacion.uniqu
e())
print "Registros con fecha alta:" , len(visita_usuario[pd.notnull(visita_us
uario.fec_incorporacion)])
print "Registros sin fecha_alta:" , len(visita_usuario[pd.isnull(visita_usu
ario.fec_incorporacion)])
print
print "locale:"
print "------------"
print "Número de locale: " ,len(visita_usuario.locale.unique())
print "Registros con locale:" , len(visita_usuario[pd.notnull(visita_usuari
o.locale)])
print "Registros sin locale:" , len(visita_usuario[pd.isnull(visita_usuari
o.locale)])
print
print "Ciudad:"
print "------------"
print "Número de ciudad: " ,len(visita_usuario.ciudad.unique())
print "Registros con ciudad:" , len(visita_usuario[pd.notnull(visita_usuari
```

```
o.ciudad)])
print "Registros sin ciudad:" , len(visita_usuario[pd.isnull(visita_usuari
o.ciudad)])
print
print "tmezone:"
print "------------"
print "Número de timezone: " ,len(visita_usuario.timezone.unique())
print "Registros con timeZone:" , len(visita_usuario[pd.notnull(visita_usua
rio.timezone)])
print "Registros sin timezone:" , len(visita_usuario[pd.isnull(visita_usuar
io.timezone)])
print
print "Fechanacimiento:"
print "------------"
print "Número de fechas nacimiento: " ,len(visita_usuario.fecha_nacimient
o.unique())
print "Registros con fecha nacimiento:" , len(visita_usuario[pd.notnull(vis
ita_usuario.fecha_nacimiento)])
print "Registros sin fecha nacimiento:" , len(visita_usuario[pd.isnull(visi
ta_usuario.fecha_nacimiento)])
print
print "Fecha nacimiento google:"
print "------------"
print "Número de FNG: " ,len(visita_usuario.birthday_gplus.unique())
print "Registros FNG:" , len(visita_usuario[pd.notnull(visita_usuario.birth
day_gplus)])
print "Registros FNG:" , len(visita_usuario[pd.isnull(visita_usuario.birthd
ay_gplus)])
print
```

REGISTROS TOTALES: 26145

USUARIOS UNICOS:
---
usuarios_unicos: 5645
Registros con usuario: 26145
Registros sin usuario: 0

HOTSPOSTS
---
número de hotspots: 35
Registros con HOTSPOT: 26106
Registros sin HOTSPOT: 39

MACS:

Número de MACS: 5924
Registros con MAC: 26106
Registros sin MAC: 39

IPS:

Número de IPS: 633
Registros con IPS: 25970
Registros sin IPS: 175

Fecha_hora:

Número de Fechas: 25555
Registros con Fecha_hora: 26106
Registros sin Fecha_hora: 39

User_aget:

Número de ua: 2264
Registros con ua: 26106
Registros sin ua: 39

OS:

Número de os: 16
Registros con os: 25970
Registros sin os: 175

Nombre:

Número de nombre: 5576
Registros con nombre: 26123
Registros sin nombre: 22

First Name:

Número de FN: 1652
Registros con FN: 14447
Registros sin FN: 11698

Last Name:

```
------------
Número de LN:  2652
Registros con LN: 14443
Registros sin LN: 11702

Email:
------------
Número de email:  5382
Registros con email: 24757
Registros sin email: 1388

Fecha alta:
------------
Número de Fechas alta:  5623
Registros con fecha alta: 26123
Registros sin fecha_alta: 22

locale:
------------
Número de locale:  66
Registros con locale: 14673
Registros sin locale: 11472

Ciudad:
------------
Número de ciudad:  363
Registros con ciudad: 4411
Registros sin ciudad: 21734

tmezone:
------------
Número de timezone:  21
Registros con timeZone: 12106
Registros sin timezone: 14039

Fechanacimiento:
------------
Número de fechas nacimiento:  1830
Registros con fecha nacimiento: 10910
Registros sin fecha nacimiento: 15235

Fecha nacimiento google:
------------
Número de FNG:  5
Registros FNG: 7
Registros FNG: 26138
```

In [60]:

```
# Usamos los siguientes datos de la tabla inicialmente
valid_data =  visita_usuario[[ u'fecha_hora', u'agent'
                            , u'os', u'nombre', u'first_name', u'last_nam
e', u'genero'
                            , u'email', u'fec_incorporacion', u'pag_gplus',
u'locale'
                             , u'ciudad', u'timezone', u'fecha_nacimient
o',u'id_hotspots'
                            ]]
```

In [62]:

```
print valid_data.columns
```

```
Index([u'fecha_hora', u'agent', u'os', u'nombre', u'first_name', u'last_nam
e', u'genero', u'email', u'fec_incorporacion', u'pag_gplus', u'locale', u'c
iudad', u'timezone', u'fecha_nacimiento', u'id_hotspots'], dtype='object')
```

User agent - idoma y pais - hora de la visita

Aprovechando que se hace un bucle sobre todo el data frame, obtenemos los siguientes campos:

- Separamos el user agent en los campos que contiene, y los añadimos al dataset
- Nos quedamos con el servidor de email
- Obtenemos el pais del visitante
- Obtenemos la hora de la visita
- Nombre (hay que revisar si el nombre conicide con el first name y decidir con cual nos quedamos)

In [39]:

```
#Definimos un par de funciones que extraen el idioma del user agent
def is_number(s):
    try:
        float(s)
        return True
    except ValueError:
        return False

def get_lang(ua):
    for m in re.finditer('-', ua):
        #print ("-" , m.start(), m.end())
    #print pos
        pos = m.start()
        if pos > 0:
            lang = ua[pos-2:pos+3]
            es_lang = True
            for letra in lang:
                if is_number(letra):
                    es_lang = False
                    break
        else:
            lang = ""
        if es_lang:
            return lang

def rango_horario(hora):
    if hora >=7 and hora <12:
        return "morning"
    elif hora >=12 and hora < 15:
        return "noon"
    elif hora >=15 and hora < 19 :
        return "afternoon"
    elif hora >=19 and hora < 21 :
        return "evening"
    elif hora >= 21:
        return "night"
    elif hora >= 0 and hora < 7:
        return "early_morning"

def calculate_age(visita, born):

    try:
        birthday = born.replace(year=visita.year)
    except ValueError: # raised when birth date is February 29 and the curr
ent year is not a leap year
        birthday = born.replace(year=visita.year, day=born.day-1)
    if birthday > visita:
        return visita.year - born.year - 1
    else:
        return visita.year - born.year
```

In [63]:

```
i= 0
new_data = []

for index, data  in valid_data.iterrows():
    data = data.copy()
    hay_lang = True
    try:
        #Usamos la función para obener los datos del user agent
        user_agent = parse(data.agent)
        #Obtenemos el idioma del user agent y de ahí el país
        idioma = get_lang(data.agent)
        if idioma == None:
            idioma = np.nan
        data["ua_idioma"] = idioma

        if pd.notnull(data.locale):
            data["lang_country"] = data.locale
            idioma_pais = data.locale.split("_")
        else:
            if pd.notnull(idioma):
                data["lang_country"] = idioma
                idioma_pais = idioma.split("-")
            else:
                idioma_pais = np.nan
                hay_lang = False

        if not hay_lang:
            data["idioma"] = np.nan
            data["pais"] = np.nan
        else:
            if len(idioma_pais)>1:
                data["idioma"] = idioma_pais[0].upper()
                data["pais"] = idioma_pais[1].upper()
            else:
                data["idioma"] = idioma_pais[0].upper()
                data["pais"] = idioma_pais[0].upper()

        data["ua_browser_family"] = user_agent.browser.family
        data["ua_os_family"] = user_agent.os.family
        data["ua_device_family"] = user_agent.device.family

        #Dividimos device para quedarnos con la marca únicamente:
        marca = user_agent.device.family.split(" ")
        data["ua_device"] = marca[0]

        data["ua_is_movile"] = user_agent.is_mobile
        data["ua_is_tablet"] =user_agent.is_tablet
        data["ua_is_tounch_capable"] = user_agent.is_touch_capable
        data["ua_is_pc"] = user_agent.is_pc
        data["ua_is_bot"] = user_agent.is_bot

        #Ahora vamos a coger el servidor al que pertenece el email
        if pd.notnull(data.email):
            email = data.email.split('@')
            data["email_server"]  = email[1]
```

```
        else:
            data["email_server"]= np.nan

        #A continuación obtenemos la hora de la visita
        data["hora_visita"] = data.fecha_hora.hour
        data["rango_horario"] = rango_horario(data.fecha_hora.hour)

        #A continuacion obtenemos el día de la semana de la visita y si es
weekend
        data["weekday"] = data.fecha_hora.weekday()
        if data.fecha_hora.weekday() == 5 or data.fecha_hora.weekday() ==
6:
            data["is_weekend"] = True
        else:
            data["is_weekend"] = False

        #A continicion nos quedamos con el nombre:
        f_nombre = data.first_name
        if data.nombre != None:
            nombre = data.nombre.split(" ")
            nombre_final = nombre[0]
        else:
            if f_nombre == None:
                nombre_final = np.nan
            else:
                nombre_final = f_nombre
        data["nombre_final"] = nombre_final

        #A continuación calculamos la edad en el momento de la visita:
        if pd.notnull(data.fecha_nacimiento):
            data["edad"] = calculate_age(data.fecha_hora, data.fecha_nacimi
ento)
        else:
            data["edad"] = np.nan

        new_data.append(data)
    except:
        #Eliminamos los registros sin user agent porque por aquí no se añad
e nada a new_data

        i+=1

df = pd.DataFrame(new_data)
```

In [41]:

```
#Contamos los registros y los vemos un poco
print "sin user agent:",i
print "total registros" , len(df)

print df.columns
print df.pais.unique()
print pd.value_counts(df.hora_visita)
```

```
sin user agent: 67
total registros 26078
Index([u'agent', u'ciudad', u'edad', u'email', u'email_server', u'fec_incor
poracion', u'fecha_hora', u'fecha_nacimiento', u'first_name', u'genero',
u'hora_visita', u'id_hotspots', u'idioma', u'is_weekend', u'lang_country',
u'last_name', u'locale', u'nombre', u'nombre_final', u'os', u'pag_gplus',
u'pais', u'rango_horario', u'timezone', u'ua_browser_family', u'ua_device',
u'ua_device_family', u'ua_idioma', u'ua_is_bot', u'ua_is_movile', u'ua_is_p
c', u'ua_is_tablet', u'ua_is_tounch_capable', u'ua_os_family', u'weekday'],
dtype='object')
[nan 'NL' 'GB' 'IE' 'ES' 'US' 'CZ' 'LA' 'PT' 'DE' 'IT' 'FR' 'RU' 'DG' 'RO'
 'TR' 'PI' 'FI' 'GR' 'CA' 'BR' 'CO' 'GT' 'EN' 'HK' 'HR' 'PL' 'SM' 'BG' 'NO'
 'CN' 'SE' 'SI' 'BE' 'GL' 'CH' 'UD' 'AR' 'LT' 'SK' 'UK' 'JP' 'IL' 'DK'
 '419' 'CS' 'EE' 'SV' 'DA' 'KR']
20    1987
19    1886
18    1818
21    1803
22    1682
17    1576
23    1484
14    1429
16    1413
15    1389
13    1371
11    1322
12    1285
10    1275
9     1103
0     1068
8      789
1      528
2      292
7      269
3      110
4       79
6       68
5       52
dtype: int64
```

nos quedamos con las columnas que necesitamos

In [42]:

```
print df.columns
print df[['locale','ua_idioma','pais','idioma']].head()
```

```
Index([u'agent', u'ciudad', u'edad', u'email', u'email_server', u'fec_incor
poracion', u'fecha_hora', u'fecha_nacimiento', u'first_name', u'genero',
u'hora_visita', u'id_hotspots', u'idioma', u'is_weekend', u'lang_country',
u'last_name', u'locale', u'nombre', u'nombre_final', u'os', u'pag_gplus',
u'pais', u'rango_horario', u'timezone', u'ua_browser_family', u'ua_device',
u'ua_device_family', u'ua_idioma', u'ua_is_bot', u'ua_is_movile', u'ua_is_p
c', u'ua_is_tablet', u'ua_is_tounch_capable', u'ua_os_family', u'weekday'],
dtype='object')
     locale ua_idioma pais idioma
0       NaN       NaN  NaN    NaN
1       NaN     nl-nl   NL     NL
2       NaN     en-gb   GB     EN
8465    NaN     en-ie   IE     EN
3        ES       NaN   ES     ES
```

In [43]:

```
print "DEVICE:"
print pd.value_counts(df.ua_device).head()
print
print "BROWSER FAMILY:"
print pd.value_counts(df.ua_browser_family).head()

paises_freq = pd.value_counts(df.pais)
email_server_freq = pd.value_counts(df.email_server)
rango_horario_freq = pd.value_counts(df.rango_horario)
dia_semana_freq = pd.value_counts(df.weekday)
fines_seman_freq = pd.value_counts(df.is_weekend)
nombres_freq = pd.value_counts(df.nombre_final)
print
print "RANGO HORARIO:"
print rango_horario_freq
print
print "Dia de la semana:"
print dia_semana_freq
print
print "fin de semana:"
print fines_seman_freq
```

```
DEVICE:
iPhone     12214
Samsung     3830
Other       2771
iPad        1543
Nexus        548
dtype: int64

BROWSER FAMILY:
Mobile Safari    13886
Chrome Mobile     5832
Android           2633
Chrome            1763
AppleMail          656
dtype: int64

RANGO HORARIO:
afternoon        6196
night            4969
morning          4758
noon             4085
evening          3873
early_morning    2197
dtype: int64

Dia de la semana:
5    4236
3    4111
4    4106
2    3828
6    3752
1    3400
0    2645
dtype: int64

fin de semana:
False    18090
True      7988
dtype: int64
```

Mapa de las visitas de estranjeros que usan la wifi

In [44]:

```
print paises_freq.head()
```

```
ES    7649
LA    2247
GB    1388
US    1366
DE     957
dtype: int64
```

In [45]:

```
latitude= []
longitude=[]
freq=[]
pais_origen = []
i= 0
for pais in paises_freq.index:
    #print coord_paises[]
    try:
        #quito españa para ver los extranjeros
        if pais != 'ES' and pais!='LA' and paises_freq[pais] > 1:
            coordenada = coord_paises.ix[coord_paises["Alpha-2 code"]==pai
s]
            pais_name = coordenada["Country"].values[0]
            latitud = coordenada["Latitude (average)"].values[0]
            longitud = coordenada["Longitude (average)"].values[0]
            latitude.append(latitud)
            longitude.append(longitud)
            pais_origen.append(pais_name )
            paises_freq.add(paises_freq[pais])
    except:
        print "no existen este pais: ", pais

print pais_origen
```

```
no existen este pais:  EN
no existen este pais:  CS
no existen este pais:  PI
no existen este pais:  419
no existen este pais:  DG
['United Kingdom', 'United States', 'Germany', 'Italy', 'Netherlands', 'Fra
nce', 'Czech Republic', 'Portugal', 'Sweden', 'Russian Federation', 'Belgiu
m', 'Slovenia', 'Slovakia', 'Norway', 'Finland', 'Poland', 'Brazil', 'Greec
e', 'Greenland', 'Romania', 'China', 'Turkey', 'Bulgaria', 'Argentina', 'Co
lombia', 'Canada', 'El Salvador', 'Japan', 'Lithuania', 'Estonia', 'Israe
l', 'Croatia', 'Ireland', 'Denmark', 'Switzerland']
```

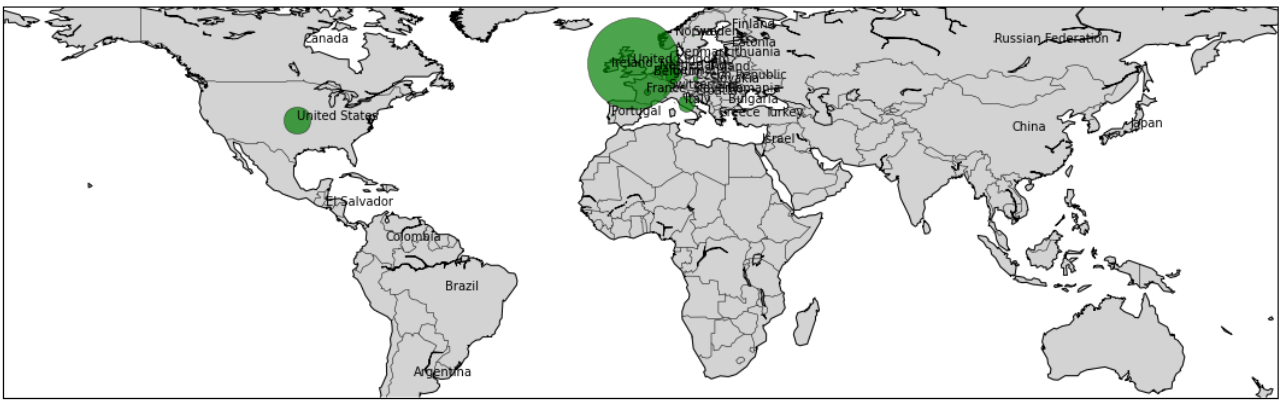
In [46]:

```
#pinto el mapa, he quitado españa y  oara
fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 15
fig_size[1] = 8

map = Basemap(llcrnrlat=-40,urcrnrlat=70, llcrnrlon=-180,urcrnrlon=180,reso
lution='c')
map.drawcoastlines()
map.drawcountries()
map.fillcontinents(color = 'lightgrey')
map.drawmapboundary()

min_marker_size = 100
for lon, lat, pf, nombre_pais in zip(longitude, latitude, paises_freq, pai
s_origen):
    x,y = map(lon, lat)
    msize = pf / min_marker_size
    print nombre_pais, pf
    map.plot(x, y, 'go', markersize=msize,alpha=0.7)
    plt.annotate(nombre_pais, xy=(lon, lat))
plt.tight_layout()
plt.rcParams["figure.figsize"] = fig_size
plt.savefig('users_country.pdf')
plt.show()
```

```
United Kingdom 7649
United States 2247
Germany 1388
Italy 1366
Netherlands 957
France 589
Czech Republic 477
Portugal 299
Sweden 268
Russian Federation 192
Belgium 172
Slovenia 116
Slovakia 87
Norway 70
Finland 57
Poland 56
Brazil 52
Greece 47
Greenland 28
Romania 26
China 25
Turkey 25
Bulgaria 18
Argentina 18
Colombia 16
Canada 13
El Salvador 8
Japan 7
Lithuania 6
Estonia 5
Israel 5
Croatia 4
Ireland 4
Denmark 4
Switzerland 3
```



4.- Tratamos los nulos

Eliminiamos los registros que no tienen user_agent y ponemos "vacio" en aquellos registros de los que nos disponemos de información

In [66]:

```
# Primero quitamos los datos de caracter personal (email, nombre completo,
MAC del dispositivo,
#                                                ids de redes sociales)
# Nos quedamos con el nombre porque puede servir para identificar el género
#
print df.columns
```

```
Index([u'agent', u'ciudad', u'edad', u'email', u'email_server', u'fec_incor
poracion', u'fecha_hora', u'fecha_nacimiento', u'first_name', u'genero',
u'hora_visita', u'id_hotspots', u'idioma', u'is_weekend', u'lang_country',
u'last_name', u'locale', u'nombre', u'nombre_final', u'os', u'pag_gplus',
u'pais', u'rango_horario', u'timezone', u'ua_browser_family', u'ua_device',
u'ua_device_family', u'ua_idioma', u'ua_is_bot', u'ua_is_movile', u'ua_is_p
c', u'ua_is_tablet', u'ua_is_tounch_capable', u'ua_os_family', u'weekday'],
dtype='object')
```

In [67]:

```
valid_colums = [u'ciudad', u'email_server', u'edad', u'genero',
       u'hora_visita', u'idioma', u'is_weekend',u'nombre_final', u'os', u'p
ais',
       u'rango_horario', u'timezone', u'ua_browser_family', u'ua_device',
       u'ua_device_family', u'ua_is_bot', u'ua_is_movile',
       u'ua_is_pc', u'ua_is_tablet', u'ua_is_tounch_capable', u'ua_os_famil
y',
       u'weekday',u'id_hotspots']
df = df[valid_colums]
print df.head(2)

#el nombre final lo guardo porque voy a intentar usarlo para predecir el se
xo, con nltk
```

```
  ciudad email_server  edad genero  hora_visita idioma is_weekend  \
0    NaN      msn.com   NaN    NaN           20    NaN      False
1    NaN  hotmail.com   NaN    NaN           20     NL       True

  nombre_final   os pais     ...            ua_device  ua_device_family  \
0         Abel  NaN  NaN     ...                D6503             D6503
1         Gert  NaN   NL     ...      Sensation_Z710e   Sensation_Z710e

  ua_is_bot ua_is_movile ua_is_pc ua_is_tablet ua_is_tounch_capable  \
0     False         True    False        False                 True
1     False         True    False        False                 True

  ua_os_family weekday id_hotspots
0      Android       2           2
1      Android       5           2

[2 rows x 23 columns]
```

In [68]:

```
for col in df.columns:
    if (col!= 'edad' and col != 'genero'):
        df[col] = df[col].fillna('vacio')
```

5.- Guardamos el dataset resultante para poder operar con el con facilidad

In [69]:

```
df.to_csv('../csv/datos_limpios.csv')
```

Bibliografía:

MAPAS:

https://peak5390.wordpress.com/2012/12/08/mapping-global-earthquake-activity-a-matplotlib-basemap-tutorial/