In [2]:
```
# Estimate the fraction of people you follow on Twitter that are female.
# In 2020, 39% of Twitter users are female.
# https://www.statista.com/statistics/828092/distribution-of-users-on-twitter-world
wide-gender/
# To put this in context, estimate the same for people you follow.
# Uses tweepy and gender_guesser packages.
#
# To use the Twitter API, you must create a Developer application:
# https://apps.twitter.com/
# Select the Create New App button and fill out the application information.
# You will ultimately need the following pieces of information:
#
#    API key
#    API secret key
#    Access token
#    Access token secret
#
# These should be stored in a CSV file that looks like:
# API key,XXXXXXXXXXXXXXXXXXXXXXXXX
# API secret key,XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
# Access token,XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
# Access token secret,XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

In [3]:
```
# set up environment
import numpy as np
import tweepy
import matplotlib.pyplot as plt
import gender_guesser.detector as gender
import re
import scipy
import scipy.ndimage as ndimage
import scipy.stats as stats
import scipy.interpolate as interpolate
import matplotlib.colors as mcolors
import csv
```

In [4]:
```
# set up tweepy authentication

# edit authfile to point to CSV file with your Twitter application information
authfile = 'C:/Users/bransonk/.twitter/KristinsGenderRatioAnalysis.csv'
# edit rootusername to be your screen name on Twitter
rootusername = 'kristinmbranson'

authinfo = {}
with open(authfile) as csvfile:
    csvreader = csv.reader(csvfile,delimiter=',')
    for row in csvreader:
        authinfo[row[0]] = row[1]

#print(authinfo)
auth = tweepy.OAuthHandler(authinfo['API key'], authinfo['API secret key'])
auth.set_access_token(authinfo['Access token'],authinfo['Access token secret'])
api = tweepy.API(auth,wait_on_rate_limit=True,
                 wait_on_rate_limit_notify=True)
```

In [5]:
```
# gets the names and ids of followers for an input id/screen name
def get_following_names(id,api,batchsize = 100,ids1=None):
    if ids1 is None:
        ids1 = api.friends_ids(id)
    namescurr = []
    for i0 in range(0,len(ids1),batchsize):
        i1 = min(len(ids1),i0+batchsize)
        user_objs = api.lookup_users(user_ids=ids1[i0:i1])
        namescurr = namescurr + list(map(lambda x: x.name,user_objs))
    namescurr.reverse()
    ids1.reverse()
    return namescurr,ids1
```

In [22]:
```
# classifies gender of names, and computes some statistics
def compute_gender_info(names,ids=None,verbose=False,sigma = 5):
    # "names" that mean this is likely not a name
    badnames = ['Lab','lab','The','the','Stanford','Mag','Club','Adventure','Rock
','Conference','Global','HHMI','Amazon','Google','Facebook','ICLR']

    # gender indicators
    kwfemale = ['female','mostly_female']
    kwmale = ['male','mostly_male']

    # classify gender
    d = gender.Detector()
    gs = []
    firstnames = []
    for name in names:
        nameparts = re.split('\s+',name)
        if not list(filter(lambda x: x in badnames,nameparts)) == []:
            if verbose:
                print('%s contains name we are ignoring'%name)
            gs.append('unknown')
        else:
            gs.append(d.get_gender(nameparts[0].capitalize()))
        if verbose: print('%s -> %s -> %s'%(name,nameparts[0],gs[-1]))

    # get info for names that we could classify
    kwknown = kwfemale + kwmale
    isknown = list(map(lambda x: x in kwknown,gs))
    knowngs = list(filter(lambda x: x in kwknown,gs))
    isfemale = list(map(lambda x: x in kwfemale,knowngs))
    ismale = list(map(lambda x: x in kwmale,knowngs))
    if ids is not None:
        idsknown = []
    else:
        idsknown = None
    namesknown = []
    gsknown = []
    for i in range(len(isknown)):
        if isknown[i]:
            if ids is not None:
                idsknown.append(ids[i])
            namesknown.append(names[i])
            gsknown.append(gs[i])

    # fraction female over time
    num = np.cumsum(np.array(isfemale))
    den = np.arange(len(isfemale))+1
    fracfemale = num/den
    fracfemale_filtered = ndimage.gaussian_filter(fracfemale,sigma,mode='nearest')

    res = {'gs': gs, 'isknown': isknown,'isfemale': isfemale,'ismale': ismale,
           'knowngs': knowngs,'namesknown': namesknown,
           'fracfemale': fracfemale,'fracfemale_filtered': fracfemale_filtered,
           'names': names, 'idsknown': idsknown,'gsknown': gsknown}
    return res
```

In [7]:
```
# chooses random samples from people you follow with at least minnfollowing people
they are following
def choose_samples_helper(idx,myinfo,api,nsample=20,minnfollowing=100,verbose=Fals
e):
    isselected = np.zeros(idx.shape,dtype=bool)
    idxsample = []
    ids1 = []

    while True:
        if np.all(isselected) or len(idxsample) >= nsample:
            break
        idxcurr, = np.nonzero(isselected==False)
        i = np.random.randint(0,len(idxcurr)) # indexes into idxcurr
        i = idxcurr[i] # indexes into isselected
        isselected[i] = True
        i = idx[i] # indexes into myinfo
        id = myinfo['idsknown'][i]
        idscurr = api.friends_ids(id)
        if verbose:
            print('%d: selected %d (%s), nfollowing = %d'%(len(idxsample),id,
                                                            myinfo['namesknown'][i],
                                                            len(idscurr)))
        if len(idscurr) >= minnfollowing:
            print('Adding')
            idxsample.append(i)
            ids1.append(idscurr)
    return (idxsample,ids1)

def choose_samples(myinfo,nsample=20,minnfollowing=100,verbose=False):

    isfemale = np.array(myinfo['isfemale'])
    idxfemale, = np.nonzero(isfemale)
    idxmale, = np.nonzero(isfemale==False)

    idxsample_female,ids1_female = \
        choose_samples_helper(idxfemale,myinfo,api,nsample=nsample,minnfollowing=mi
nnfollowing,verbose=verbose)
    idxsample_male,ids1_male = \
        choose_samples_helper(idxmale,myinfo,api,nsample=nsample,minnfollowing=minn
following,verbose=verbose)

    nsample_female = len(idxsample_female)
    nsample_male = len(idxsample_male)
    ids1 = ids1_female + ids1_male
    idxsample = idxsample_female + idxsample_male

    return idxsample,ids1,nsample_female,nsample_male
```

In [114]:
```
def get_tweets(screen_name,maxntweets=np.Inf):
    statuses = []
    ntweets = 0
    for status in tweepy.Cursor(api.user_timeline,screen_name,include_rts=True).it
ems():
        statuses.append(status)
        ntweets += 1
        if ntweets >= maxntweets:
            break

    statuses.reverse()
    return statuses

def get_retweet_names(statuses,verbose=False):

    retweet_names = []
    retweet_types = []
    isretweet = np.zeros(len(statuses))==0

    for i in range(len(statuses)):
        status = statuses[i]

        if hasattr(status,'is_quote_status') and status.is_quote_status and \
        hasattr(status,'quoted_status'):
            name = status.quoted_status.author.name
            retweet_names.append(name)
            retweet_types.append('quote')
            if verbose:
                print('%d: quote %s'%(i,name))
        elif hasattr(status,'retweeted_status'):
            name = status.retweeted_status.author.name
            retweet_names.append(name)
            retweet_types.append('retweet')
            if verbose:
                print('%d: retweet %s'%(i,name))
        elif hasattr(status,'in_reply_to_screen_name') and \
        status.in_reply_to_screen_name is not None:
            if status.in_reply_to_screen_name.lower() == rootusername.lower():
                if verbose:
                    print('%d: reply to own tweet'%i)
            else:
                in_reply_to_name = None
                for u in status.entities['user_mentions']:
                    if(u['screen_name'].lower() == status.in_reply_to_screen_name.
lower()):
                        in_reply_to_name = u['name']
                        break
                if in_reply_to_name == None:
                    isretweet[i] = False
                    if verbose:
                        print('%d: reply to ??'%i)
                else:
                    retweet_names.append(in_reply_to_name)
                    retweet_types.append('reply')
                    if verbose:
                        print('%d: reply %s'%(i,in_reply_to_name))
        else:
            isretweet[i] = False
            if verbose:
                print('%d: ??'%i)
                print(status.text)

    return (retweet_names,retweet_types,isretweet)
```

In [68]:
```
# load data from file

import pickle
filename = 'C:/Code/TwitterGenderRatio/test20200510.pickle'
fid = open(filename,'rb')
res = pickle.load(fid)
fid.close()

ids = res['ids']
names = res['names']
idxsample = res['idxsample']
names1 = res['names1']
ids1 = res['ids1']
nsample_female = len(idxsample)//2
```

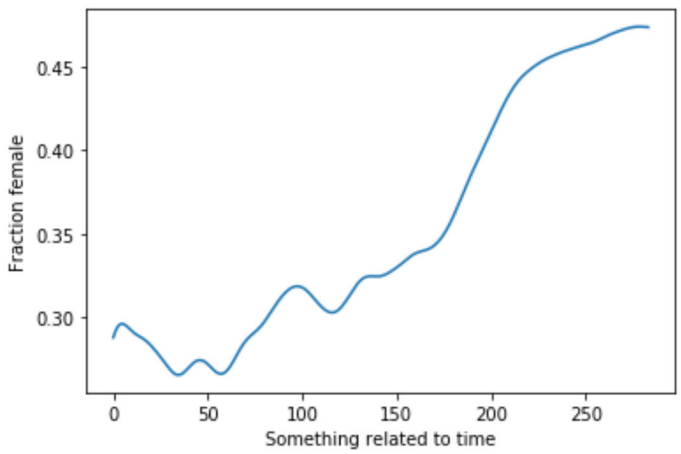
In [9]:
```
# get names & ids of people I follow
names,ids = get_following_names(rootusername,api)
print('Number following: %d'%len(names))
```

```
Number following: 488
```

In [10]:
```
# classify gender based on first name
sigma = 5
myinfo = compute_gender_info(names,ids,verbose=False,sigma=sigma)
print('Fraction of people I follow who are female: %f'%myinfo['fracfemale'][-1])

# plot fraction female
plt.plot(myinfo['fracfemale_filtered'][5*sigma:-5*sigma])
plt.xlabel('Something related to time')
plt.ylabel('Fraction female')
```

```
Fraction of people I follow who are female: 0.485030
```

Out[10]:
```
Text(0, 0.5, 'Fraction female')
```

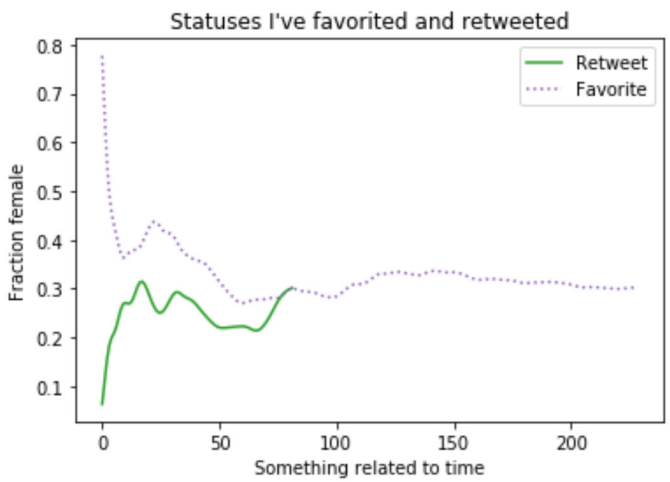


In [147]:
```
# choose some colors
colors = list(mcolors.TABLEAU_COLORS)
colorf = colors[0]
colorm = colors[1]
colorretweet = colors[2]
colorfav = colors[4]
```

In [151]:
```
# look at gender ratio for people I retweet

maxntweets = 400
maxnfavs = 400
mystatuses = get_tweets(rootusername,maxntweets=maxntweets)
myretweetnames,myretweettypes,myisretweet = get_retweet_names(mystatuses,verbose=F
alse)
myretweetinfo = compute_gender_info(myretweetnames,verbose=False,sigma=2)

myfavs = get_favs(screen_name=rootusername,maxnfavs=maxnfavs)
myfavnames = get_fav_names(myfavs)
myfavinfo = compute_gender_info(myfavnames,verbose=False,sigma=2)

print('Fraction of statuses I\'ve retweeted from women: %f'%myretweetinfo['fracfem
ale'][-1])
print('Fraction of statuses I\'ve favorited from women: %f'%myfavinfo['fracfemale
'][-1])

# plot fraction female

plt.plot(myretweetinfo['fracfemale_filtered'],label='Retweet',color=colorretweet)
plt.plot(myfavinfo['fracfemale_filtered'],':',label='Favorite',color=colorfav)

plt.xlabel('Something related to time')
plt.ylabel('Fraction female')
plt.legend()
plt.title('Statuses I\'ve favorited and retweeted')
```

```
Fraction of statuses Ive retweeted from women: 0.304878
Fraction of statuses Ive favorited from women: 0.301310
```

Out[151]:
```
Text(0.5, 1.0, "Statuses I've favorited and retweeted")
```



In [8]:
```
# choose some random following
nsample = 50
minnfollowing = 100
idxsample,ids1,nsample_female,nsample_male = \
    choose_samples(myinfo,nsample=nsample,minnfollowing=minnfollowing,verbose=True)

assert nsample==nsample_female and nsample==nsample_male,'Did not find enough sampl
es with minnfollowing = %d'%minnfollowing
```

```
0: selected 207665930 (Serena Yeung), nfollowing = 120
Adding
1: selected 1359059238 (Dima Damen), nfollowing = 341
Adding
2: selected 19563103 (Gwen Pearson), nfollowing = 841
Adding
3: selected 1890694861 (Rose Yu), nfollowing = 294
Adding
4: selected 731538535795163136 (Sara Hooker), nfollowing = 1682
Adding
5: selected 737478121079906304 (Judith MitraniReiser), nfollowing = 676
Adding
6: selected 178588367 (Karla Kaun), nfollowing = 839
Adding
7: selected 345129453 (Nan Rosemary Ke), nfollowing = 338
Adding
8: selected 35269421 (Ellie Heckscher), nfollowing = 516
Adding
9: selected 2369329526 (Ulrike Boehm 🔬🦠), nfollowing = 286
Adding
10: selected 1135303672219545600 (Marcella Noorman), nfollowing = 45
10: selected 2577596593 (Chelsea Finn), nfollowing = 244
Adding
11: selected 217852227 (Elizabeth C. Gorski), nfollowing = 630
Adding

Rate limit reached. Sleeping for: 871

12: selected 1237734334385344512 (Mai Morimoto), nfollowing = 173
Adding
13: selected 3318332379 (Ilana Witten), nfollowing = 602
Adding
14: selected 1431348835 (Anne Carpenter), nfollowing = 1117
Adding
15: selected 1237147828704575488 (Sara Beery), nfollowing = 759
Adding
16: selected 36819554 (Megan Carey), nfollowing = 933
Adding
17: selected 22385548 (Jane Wang), nfollowing = 339
Adding
18: selected 908902292 (Marta Costa), nfollowing = 207
Adding
19: selected 875432666555965444 (Krystyna Keleman), nfollowing = 213
Adding
20: selected 3072541911 (Stephanie Albin), nfollowing = 498
Adding
21: selected 788467623629500416 (Doris Tsao), nfollowing = 252
Adding
22: selected 2389878942 (Emily Behrman), nfollowing = 340
Adding
23: selected 950751996084150272 (Larissa Heinrich), nfollowing = 86
23: selected 843706252517502977 (Christine Käser-Chen), nfollowing = 184
Adding
24: selected 45675087 (Devi Parikh), nfollowing = 125
Adding
25: selected 2704715387 (Jen Heemstra), nfollowing = 5000
Adding
26: selected 246226577 (Salma Elmalaki), nfollowing = 132
Adding

Rate limit reached. Sleeping for: 897
```

```
27: selected 305092591 (Emma Brunskill), nfollowing = 52
27: selected 43406294 (renan ozturk), nfollowing = 176
Adding
28: selected 19087450 (Edith Zimmerman), nfollowing = 989
Adding
29: selected 215113195 (Naomi Saphra), nfollowing = 1011
Adding
30: selected 979697205664800768 (Elizabeth Hillman), nfollowing = 815
Adding
31: selected 959028649528840192 (Virginie Uhlmann), nfollowing = 159
Adding
32: selected 870108900128903169 (Claire Deo), nfollowing = 285
Adding
33: selected 2405083879 (eugenia chiappe), nfollowing = 422
Adding
34: selected 25320089 (Grace Vesom), nfollowing = 387
Adding
35: selected 16520284 (Alice Oh), nfollowing = 436
Adding
36: selected 748267272939020293 (Laura Leal-Taixe), nfollowing = 60
36: selected 2172505322 (Kathryn Brown), nfollowing = 27
36: selected 535136727 (Dawn Song), nfollowing = 704
Adding
37: selected 842164502422417409 (Adrienne Fairhall), nfollowing = 276
Adding
38: selected 16017475 (Nate Silver), nfollowing = 1288
Adding

Rate limit reached. Sleeping for: 896

39: selected 159315527 (Jewel Burks Solomon), nfollowing = 3858
Adding
40: selected 822090549490499585 (Nadine Gogolla), nfollowing = 749
Adding
41: selected 28912478 (Leslie Vosshall PhD), nfollowing = 497
Adding
42: selected 925800751628279808 (Nan Jiang), nfollowing = 46
42: selected 1235552122957115394 (Carolina Wählby), nfollowing = 0
42: selected 869862586610851840 (Jeannette Bohg), nfollowing = 254
Adding
43: selected 700532262165676033 (Sarah Certel), nfollowing = 247
Adding
44: selected 21828411 (Erin LeDell), nfollowing = 4995
Adding
45: selected 2869101210 (Jenn Wortman Vaughan), nfollowing = 354
Adding
46: selected 1143074659291680768 (Ann Kennedy), nfollowing = 169
Adding
47: selected 2460047754 (Janelle Shane), nfollowing = 885
Adding
48: selected 543919023 (Martha White), nfollowing = 52
48: selected 276643081 (Cori Bargmann), nfollowing = 732
Adding
49: selected 1035389878605885440 (Athena Akrami), nfollowing = 356
Adding

Rate limit reached. Sleeping for: 896
```

```
0: selected 14986849 (Alex Smola), nfollowing = 67
0: selected 828056721750896640 (mark cembrowski), nfollowing = 404
Adding
1: selected 29843511 (Nando de Freitas), nfollowing = 358
Adding
2: selected 261789755 (jeremy freeman), nfollowing = 1572
Adding
3: selected 791306523062497280 (Wyatt Korff), nfollowing = 13
3: selected 234270825 (Ken Jennings), nfollowing = 551
Adding
4: selected 190138220 (Jonathan Pillow), nfollowing = 1153
Adding
5: selected 27648853 (Peter), nfollowing = 1018
Adding
6: selected 433741920 (Marius Pachitariu), nfollowing = 1008
Adding
7: selected 769978990706720768 (Raphael Turcotte), nfollowing = 161
Adding
8: selected 780291008 (karel svoboda), nfollowing = 226
Adding
9: selected 1033383109440356352 (Oisin Mac Aodha), nfollowing = 326
Adding
10: selected 128781736 (Sasha DiGiulian), nfollowing = 620
Adding
11: selected 1014691 (David Cho), nfollowing = 593
Adding
12: selected 1150552125065355264 (Jan Funke), nfollowing = 55
12: selected 18098674 (Brendan Quigley), nfollowing = 79

Rate limit reached. Sleeping for: 896

12: selected 1193222240202035200 (Andrew Saxe), nfollowing = 186
Adding
13: selected 197684961 (Misha Denil), nfollowing = 952
Adding
14: selected 48008938 (Yann LeCun), nfollowing = 282
Adding
15: selected 911297187664949248 (Jeff Dean (@🏠)), nfollowing = 3100
Adding
16: selected 172101003 (Greg Jefferis), nfollowing = 783
Adding
17: selected 1242216846033473537 (Manuel Mohr), nfollowing = 59
17: selected 56786888 (Martin Jones), nfollowing = 3935
Adding
18: selected 636023721 (Adam J Calhoun), nfollowing = 1026
Adding
19: selected 919035620 (Matthieu Louis), nfollowing = 35
19: selected 53514472 (Andrew Fitzgibbon), nfollowing = 461
Adding
20: selected 19301221 (Andrew S. Champion), nfollowing = 248
Adding
21: selected 50393960 (Bill Gates), nfollowing = 218
Adding
22: selected 149895490 (Gonzalo de Polavieja), nfollowing = 1046
Adding
23: selected 3333052551 (Hugo Larochelle), nfollowing = 527
Adding
24: selected 1026931440280391687 (Kaspar Podgorski), nfollowing = 203
Adding

Rate limit reached. Sleeping for: 896
```

```
25: selected 3111733301 (Daniel Gonzales), nfollowing = 926
Adding
26: selected 1400517288 (Sandeep Robert Datta), nfollowing = 802
Adding
27: selected 938416059962609665 (Mike Economo), nfollowing = 578
Adding
28: selected 223734352 (Matt Gritzmacher), nfollowing = 531
Adding
29: selected 16055364 (Il Memming Park), nfollowing = 1612
Adding
30: selected 3192303453 (Andreas Kay), nfollowing = 10
30: selected 51582812 (Stephen Holtz), nfollowing = 827
Adding
31: selected 813286 (Barack Obama), nfollowing = 5000
Adding
32: selected 14348594 (John Hodgman), nfollowing = 3860
Adding
33: selected 2848165007 (John Bogovic), nfollowing = 463
Adding
34: selected 122080635 (Sebastian Seung), nfollowing = 292
Adding
35: selected 15035863 (Noah Snavely), nfollowing = 378
Adding
36: selected 1210596212140892160 (John Langford), nfollowing = 21
36: selected 19767193 (Ed Yong), nfollowing = 1674
Adding

Rate limit reached. Sleeping for: 897

37: selected 314158631 (Erich Jarvis), nfollowing = 371
Adding
38: selected 14230012 (Rex Parker⚾🐘🐘🐘⚾), nfollowing = 359
Adding
39: selected 1173981576046227457 (David E. Clapham), nfollowing = 6
39: selected 930090512 (Stephan Saalfeld), nfollowing = 154
Adding
40: selected 31936449 (Lior Pachter), nfollowing = 1287
Adding
41: selected 33362653 (Trace Henry), nfollowing = 2653
Adding
42: selected 14162415 (Ryan North), nfollowing = 897
Adding
43: selected 221304470 (David Schoppik), nfollowing = 1015
Adding
44: selected 430783446 (Michael Chabon), nfollowing = 38
44: selected 809072402282016768 (Daniel Jiwoong Im), nfollowing = 509
Adding
45: selected 2847954257 (Frank), nfollowing = 145
Adding
46: selected 29905013 (Peter Gordon), nfollowing = 28
46: selected 4558314927 (Sasha Rush), nfollowing = 325
Adding
47: selected 946827254901936130 (Davis Bennett), nfollowing = 77
47: selected 22445339 (Fred "Replace Trump Now" Wolf), nfollowing = 408
Adding
48: selected 2835683058 (Silvio Savarese), nfollowing = 28

Rate limit reached. Sleeping for: 896

48: selected 57663013 (Jesse Marshall), nfollowing = 191
Adding
49: selected 259568572 (Benjamin de Bivort), nfollowing = 1031
Adding
```

In [47]:
```
sampleids = list(map(lambda x: myinfo['idsknown'][x],idxsample))
samplenames = list(map(lambda x: myinfo['namesknown'][x],idxsample))
print('Female samples:')
print(samplenames[:nsample_female])
print('Male samples:')
print(samplenames[nsample_female:])
```

```
Female samples:
['Jeannette Bohg', 'Laura Leal-Taixe', 'Naomi Saphra', 'Bill Gates', 'Nan Rosema
ry Ke', 'Judith MitraniReiser', 'Karla Kaun', 'Martha White', 'Davis Bennett', '
Allan Wong', 'John Hodgman', 'Elizabeth C. Gorski', 'Ellie Heckscher', 'Athena A
krami', 'Anne Carpenter', 'Barack Obama', 'Alex Wild', 'Jane Wang', 'Sara Beery
', 'Adam L. Taylor', 'Chris Potter', 'Sasha Rush', 'Eric Schreiter', 'Alex Smola
', 'Edith Zimmerman', 'Jen Heemstra', 'Elena Rivas', 'Tommy Caldwell', 'David Ch
o', 'Ted Pedersen', 'Elijah Cole', 'Martin Jones', 'Mike Economo', 'Andrew S. Ch
ampion', 'Grace Vesom', 'Rachel Thomas', 'Shakir Mohamed', 'Matt Gritzmacher', '
Erich Jarvis', 'Jewel Burks Solomon', 'Serena Yeung', 'Kay M Tye', 'Silvio Savar
ese', 'Sarah Certel', 'Matthieu Louis', 'Tessa Montague', 'Krystyna Keleman', 'S
ara Hooker', 'Sebastian Seung', 'Noah Snavely']
Male samples:
['Stephanie Albin', 'Nando de Freitas', 'Jim Keeley', 'Doris Tsao', 'Surya Gangu
li', 'karel svoboda', 'Misha Ahrens', 'Marius Pachitariu', 'Nelson Spruston', 'I
lana Witten', 'Nicholas Turner', 'Hanna Wallach', 'Caroline Palavicino-Maggio Ph
D', 'Misha Denil', 'Yann LeCun', 'Jeff Dean (@🏠)', 'Fernanda Viégas', 'Jakob Ma
cke', 'Adam J Calhoun', 'Andrew Fitzgibbon', 'Virginie Uhlmann', 'Andrew Saxe',
'Alice Oh', 'Hugo Larochelle', 'Ed Yong', 'Philipp Hanslovsky', 'Sandeep Robert
Datta', 'mark cembrowski', 'Gonzalo de Polavieja', 'Andrew Straw', 'Andreas Kay
', 'Elizabeth Hillman', 'Christine Käser-Chen', 'John Bogovic', 'Leslie Vosshall
PhD', 'Marcella Noorman', 'Adrienne Fairhall', 'Carlos Ribeiro', 'Rex Parker⚾🐘
🐘🐘⚾', 'Stephan Saalfeld', 'Janelle Shane', 'Jan M Ache', 'John Langford', 'Fr
ed "Replace Trump Now" Wolf', 'Juan Carlos Niebles', 'Rose Yu', 'Lior Pachter',
'David Sussillo 🏠💻👆🤓', 'Ann Kennedy', 'Benjamin de Bivort']
```

In [10]:
```
# get names of their following
batchsize = 100
names1 = []
for samplei in range(len(names1),len(idxsample)):
    sample = idxsample[samplei]
    id = myinfo['idsknown'][sample]
    print('samplei = %d, sample = %d, id = %d'%(samplei,sample,id))
    namescurr,idscurr = get_following_names(id,api,ids1=ids1[samplei])
    names1.append(namescurr)
    ids1.append(idscurr)
```

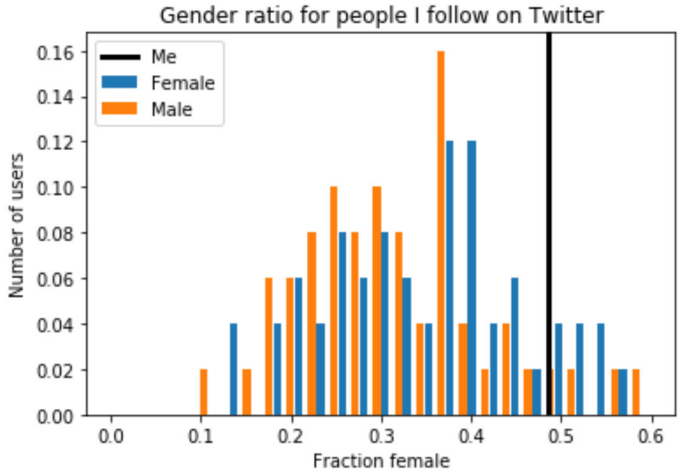
```
samplei = 0, sample = 152, id = 207665930
samplei = 1, sample = 200, id = 1359059238
samplei = 2, sample = 194, id = 19563103
samplei = 3, sample = 167, id = 1890694861
samplei = 4, sample = 182, id = 731538535795163136
samplei = 5, sample = 15, id = 737478121079906304
samplei = 6, sample = 19, id = 178588367
samplei = 7, sample = 183, id = 345129453
samplei = 8, sample = 174, id = 35269421
samplei = 9, sample = 118, id = 2369329526
samplei = 10, sample = 92, id = 2577596593
samplei = 11, sample = 31, id = 217852227
samplei = 12, sample = 173, id = 1237734334385344512
samplei = 13, sample = 87, id = 3318332379
samplei = 14, sample = 4, id = 1431348835
samplei = 15, sample = 156, id = 1237147828704575488
samplei = 16, sample = 192, id = 36819554
samplei = 17, sample = 27, id = 22385548
samplei = 18, sample = 155, id = 908902292
samplei = 19, sample = 73, id = 875432666555965444
samplei = 20, sample = 124, id = 3072541911
samplei = 21, sample = 178, id = 788467623629500416
samplei = 22, sample = 126, id = 2389878942
samplei = 23, sample = 94, id = 843706252517502977
samplei = 24, sample = 105, id = 45675087
samplei = 25, sample = 20, id = 2704715387
samplei = 26, sample = 115, id = 246226577
samplei = 27, sample = 197, id = 43406294
samplei = 28, sample = 106, id = 19087450
samplei = 29, sample = 195, id = 215113195
samplei = 30, sample = 158, id = 979697205664800768
samplei = 31, sample = 52, id = 959028649528840192
samplei = 32, sample = 121, id = 870108900128903169
samplei = 33, sample = 50, id = 2405083879
samplei = 34, sample = 23, id = 25320089
samplei = 35, sample = 146, id = 16520284
samplei = 36, sample = 185, id = 535136727
samplei = 37, sample = 143, id = 842164502422417409
samplei = 38, sample = 83, id = 16017475
samplei = 39, sample = 13, id = 159315527
samplei = 40, sample = 151, id = 822090549490499585
samplei = 41, sample = 67, id = 28912478
samplei = 42, sample = 153, id = 869862586610851840
samplei = 43, sample = 37, id = 700532262165676033
samplei = 44, sample = 148, id = 21828411
samplei = 45, sample = 111, id = 2869101210
samplei = 46, sample = 72, id = 1143074659291680768
samplei = 47, sample = 181, id = 2460047754
samplei = 48, sample = 65, id = 276643081
samplei = 49, sample = 88, id = 1035389878605885440
samplei = 50, sample = 123, id = 828056721750896640
samplei = 51, sample = 36, id = 29843511
samplei = 52, sample = 138, id = 261789755
samplei = 53, sample = 177, id = 234270825
samplei = 54, sample = 56, id = 190138220
samplei = 55, sample = 132, id = 27648853
samplei = 56, sample = 130, id = 433741920
samplei = 57, sample = 129, id = 769978990706720768
samplei = 58, sample = 133, id = 780291008
samplei = 59, sample = 86, id = 1033383109440356352
samplei = 60, sample = 164, id = 128781736
samplei = 61, sample = 107, id = 1014691
samplei = 62, sample = 169, id = 1193222240202035200
samplei = 63, sample = 33, id = 197684961
```

In [49]:
```
# compute gender ratio info
otherinfo = []
for i in range(len(names1)):
    #print('i = %d: %s -> first name = %s'%(i,sample_user_objs[i].name,names1
[i][0]))
    otherinfo.append(compute_gender_info(names1[i],ids=ids1[i],verbose=False,sigma=
sigma))
```

In [150]:
```
# plot a histogram of frac female following for samples and for me
nbins = 25

fracfemale_f = np.zeros(nsample_female)
fracfemale_m = np.zeros(nsample_female)
for i in range(nsample_female):
    if len(otherinfo[i]['fracfemale']) == 0:
        fracfemale_f[i] = np.nan
        print('Bad sample %d'%i)
    else:
        fracfemale_f[i] = otherinfo[i]['fracfemale'][-1]
    if len(otherinfo[i+nsample_female]['fracfemale']) == 0:
        fracfemale_m[i] = np.nan
        print('Bad sample %d'%i+nsample_female)
    else:
        fracfemale_m[i] = otherinfo[i+nsample_female]['fracfemale'][-1]

myprctile_f = np.count_nonzero(fracfemale_f<=myinfo['fracfemale'][-1])/nsample_fem
ale
myprctile_m = np.count_nonzero(fracfemale_m<=myinfo['fracfemale'][-1])/nsample_fem
ale
print('My percentile among women I follow: %f'%(myprctile_f*100.))
print('My percentile among men I follow: %f'%(myprctile_m*100.))
print('My percentile among people I follow: %f'%((myprctile_f+myprctile_m)*50.))

counts_f,edges = np.histogram(fracfemale_f,bins=25,range=(0,.6))
counts_m,edges = np.histogram(fracfemale_m,bins=25,range=(0,.6))
ctrs = (edges[1:]+edges[:-1])/2.
width = edges[1]-edges[0]
normcounts_f = counts_f / np.sum(counts_f)
normcounts_m = counts_m / np.sum(counts_m)
plt.bar(ctrs+width/5.,normcounts_f,width=width*.35,color=colorf,label='Female')
plt.bar(ctrs-width/5.,normcounts_m,width=width*.35,color=colorm,label='Male')
ylim = plt.gca().get_ylim()
plt.plot([myinfo['fracfemale'][-1]]*2,ylim,'k-',lw=3,label='Me')
plt.gca().set_ylim(ylim)
plt.legend()
plt.xlabel('Fraction female')
plt.ylabel('Number of users')
plt.title('Gender ratio for people I follow on Twitter')

stats.mannwhitneyu(fracfemale_f,fracfemale_m,alternative='greater')
```

```
My percentile among women I follow: 86.000000
My percentile among men I follow: 92.000000
My percentile among people I follow: 89.000000
```

Out[150]:
```
MannwhitneyuResult(statistic=1463.0, pvalue=0.07146909115821613)
```



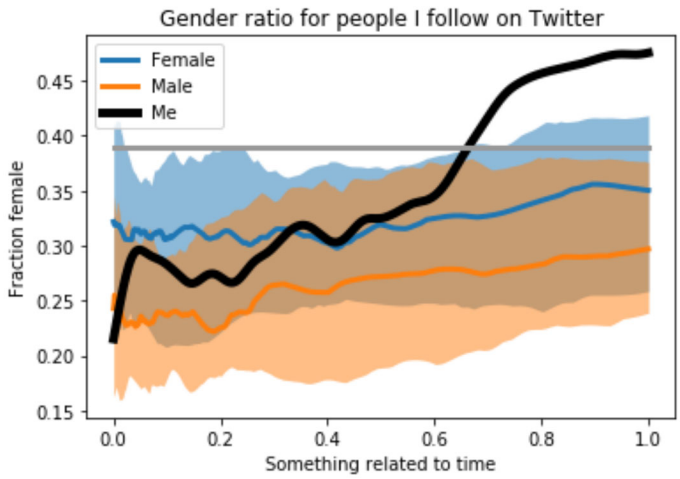
In [154]:
```
# get info about following
anonymize = True
samplenames = list(map(lambda x: myinfo['namesknown'][x],idxsample))
samplegenders = list(map(lambda x: myinfo['gsknown'][x],idxsample))

for i in range(len(samplenames)):
    if anonymize:
        if len(otherinfo[i]['fracfemale']) == 0:
            print('anonymous %s: not following anyone'%(samplegenders[i]))
        else:
            print('anonymous %s: %f'%(samplegenders[i],otherinfo[i]['fracfemale
'][-1]))
    else:
        if len(otherinfo[i]['fracfemale']) == 0:
            print('%s (%s): not following anyone'%(samplenames[i],samplegenders
[i]))
        else:
            print('%s (%s): %f'%(samplenames[i],samplegenders[i],otherinfo[i]['fra
cfemale'][-1]))
```

```
anonymous female: 0.125000
anonymous female: 0.225000
anonymous female: 0.511530
anonymous male: 0.215789
anonymous mostly_female: 0.378406
anonymous female: 0.501348
anonymous female: 0.416149
anonymous female: 0.138889
anonymous male: 0.441558
anonymous male: 0.353933
anonymous male: 0.207547
anonymous female: 0.392573
anonymous female: 0.371681
anonymous female: 0.390187
anonymous female: 0.384189
anonymous male: 0.428571
anonymous male: 0.403200
anonymous female: 0.187739
anonymous female: 0.315789
anonymous male: 0.244275
anonymous mostly_male: 0.436782
anonymous mostly_male: 0.314286
anonymous male: 0.375000
anonymous male: 0.301887
anonymous female: 0.246377
anonymous female: 0.529982
anonymous female: 0.373134
anonymous male: 0.211765
anonymous male: 0.466387
anonymous male: 0.286614
anonymous male: 0.367188
anonymous male: 0.272727
anonymous male: 0.298701
anonymous male: 0.298319
anonymous female: 0.560000
anonymous female: 0.266272
anonymous male: 0.189555
anonymous male: 0.320000
anonymous male: 0.220106
anonymous mostly_female: 0.397608
anonymous female: 0.437624
anonymous male: 0.485623
anonymous male: 0.375887
anonymous female: 0.337349
anonymous male: 0.508062
anonymous female: 0.396761
anonymous female: 0.261905
anonymous female: 0.529750
anonymous male: 0.250564
anonymous male: 0.303419
anonymous female: 0.384354
anonymous male: 0.175573
anonymous male: 0.365979
anonymous female: 0.290488
anonymous male: 0.374847
anonymous male: 0.334895
anonymous male: 0.294118
anonymous male: 0.260870
anonymous male: 0.242038
anonymous female: 0.220000
anonymous male: 0.406528
anonymous female: 0.354430
anonymous female: 0.206667
anonymous male: 0.147766
```

In [58]:
```
# plot stats of samples and me
maxl = 1000
dointerp = True # whether to plot absolute following number or fraction of followin
g
doplotprctiles = True
allusers_fracfemale = 0.39 # https://www.statista.com/statistics/828092/distributio
n-of-users-on-twitter-worldwide-gender/

Y = np.zeros((len(otherinfo),maxl))
Y[:] = np.nan
counts = np.zeros((1,maxl))
xinterp = np.linspace(0, 1, maxl)
for i in range(0,len(otherinfo)):
    y = otherinfo[i]['fracfemale_filtered'][3*sigma:-3*sigma]
    if dointerp:
        x = np.linspace(0,1,y.shape[0])
        f = interpolate.interp1d(x,y,axis=0)
        yinterp = f(xinterp)
        Y[i,:] = yinterp
    else:
        l = min(len(y),maxl)
        Y[i,:l] = y[:l]

idxfemale = np.arange(0,nsample_female)
idxmale = np.arange(nsample_female,2*nsample_female)

prctiles_compute = np.array([25,50,75])
middlei, = np.where(prctiles_compute==50)
middlei = middlei[0]
prctiles_female = np.percentile(Y[idxfemale,:],prctiles_compute,axis=0)
prctiles_male = np.percentile(Y[idxmale,:],prctiles_compute,axis=0)

mu_female = np.nanmean(Y[idxfemale,:],axis=0)
mu_male = np.nanmean(Y[idxmale,:],axis=0)
sig_female = np.nanstd(Y[idxfemale,:],axis=0)
sig_male = np.nanstd(Y[idxmale,:],axis=0)
counts_female = np.sum(np.isnan(Y[idxfemale,:])==False,axis=0)
counts_male = np.sum(np.isnan(Y[idxmale,:])==False,axis=0)
stderr_female = sig_female / np.sqrt(counts_female)
stderr_male = sig_male / np.sqrt(counts_male)
if dointerp:
    x = xinterp
else:
    x = np.arange(maxl)

if doplotprctiles:
    alpha = (1.-np.abs(50.-np.array(prctiles_compute))/50.)
    print(alpha)
    for i in range(prctiles_compute.shape[0]//2):
        plt.fill_between(x, prctiles_female[i,:], prctiles_female[-i-1,:],alpha=alp
ha[i],lw=0,color=colorf)
        plt.fill_between(x, prctiles_male[i,:], prctiles_male[-i-1,:],alpha=alpha
[i],lw=0,color=colorm)
    hf, = plt.plot(x,prctiles_female[middlei,:],'-',linewidth=3,color=colorf,label=
'Female')
    hm, = plt.plot(x,prctiles_male[middlei,:],'-',linewidth=3,color=colorm,label='M
ale')
else:

    # plot mean and standard error
    plt.plot(x,mu_female-sig_female,'-',color=colorf)
    plt.plot(x,mu_female+sig_female,'-',color=colorf)
    hf, = plt.plot(x,mu_female,'-',color=colorf,linewidth=3,label='Female')
    plt.plot(x,mu_male-sig_male,'-',color=colorm)
```

```
[0.5 1.  0.5]
```

Out[58]: <matplotlib.legend.Legend at 0x12ffa264a08>



In [69]:
```
# who do people i follow retweet?

otherretweetnames = []

for samplei in range(len(otherretweetnames),len(idxsample)):
    sample = idxsample[samplei]
    id = myinfo['idsknown'][sample]
    print('samplei = %d, name = %s, sample = %d, id = %d'%(samplei,myinfo['namesknown'][sample],sample,id))
    statusescurr = get_tweets(id,maxntweets=maxntweets)
    namescurr,retweettypescurr,isretweetcurr = get_retweet_names(statusescurr,verbose=False)
    #print(namescurr)
    otherretweetnames.append(namescurr)
```

```
samplei = 0, name = Jeannette Bohg, sample = 152, id = 869862586610851840
samplei = 1, name = Laura Leal-Taixe, sample = 200, id = 748267272939020293
samplei = 2, name = Naomi Saphra, sample = 194, id = 215113195
samplei = 3, name = Bill Gates, sample = 167, id = 50393960
samplei = 4, name = Nan Rosemary Ke, sample = 182, id = 345129453
samplei = 5, name = Judith MitraniReiser, sample = 15, id = 737478121079906304
samplei = 6, name = Karla Kaun, sample = 19, id = 178588367
samplei = 7, name = Martha White, sample = 183, id = 543919023
samplei = 8, name = Davis Bennett, sample = 174, id = 946827254901936130
samplei = 9, name = Allan Wong, sample = 118, id = 49224993
samplei = 10, name = John Hodgman, sample = 92, id = 14348594
samplei = 11, name = Elizabeth C. Gorski, sample = 31, id = 217852227
samplei = 12, name = Ellie Heckscher, sample = 173, id = 35269421
samplei = 13, name = Athena Akrami, sample = 87, id = 1035389878605885440
samplei = 14, name = Anne Carpenter, sample = 4, id = 1431348835
samplei = 15, name = Barack Obama, sample = 156, id = 813286
samplei = 16, name = Alex Wild, sample = 192, id = 166022406
samplei = 17, name = Jane Wang, sample = 27, id = 22385548
samplei = 18, name = Sara Beery, sample = 155, id = 1237147828704575488
samplei = 19, name = Adam L. Taylor, sample = 73, id = 14840926
samplei = 20, name = Chris Potter, sample = 124, id = 1043667150
samplei = 21, name = Sasha Rush, sample = 178, id = 4558314927
samplei = 22, name = Eric Schreiter, sample = 126, id = 3797818101
samplei = 23, name = Alex Smola, sample = 94, id = 14986849
samplei = 24, name = Edith Zimmerman, sample = 105, id = 19087450
samplei = 25, name = Jen Heemstra, sample = 20, id = 2704715387
samplei = 26, name = Elena Rivas, sample = 115, id = 2789742375
samplei = 27, name = Tommy Caldwell, sample = 197, id = 246124588
samplei = 28, name = David Cho, sample = 106, id = 1014691
samplei = 29, name = Ted Pedersen, sample = 195, id = 1002638800575782912
samplei = 30, name = Elijah Cole, sample = 158, id = 1192003193921654784
samplei = 31, name = Martin Jones, sample = 52, id = 56786888
samplei = 32, name = Mike Economo, sample = 121, id = 938416059962609665
samplei = 33, name = Andrew S. Champion, sample = 50, id = 19301221
samplei = 34, name = Grace Vesom, sample = 23, id = 25320089
samplei = 35, name = Rachel Thomas, sample = 146, id = 1408142352
samplei = 36, name = Shakir Mohamed, sample = 185, id = 476582730
samplei = 37, name = Matt Gritzmacher, sample = 143, id = 223734352
samplei = 38, name = Erich Jarvis, sample = 83, id = 314158631
samplei = 39, name = Jewel Burks Solomon, sample = 13, id = 159315527

Rate limit reached. Sleeping for: 280
```

```
samplei = 40, name = Serena Yeung, sample = 151, id = 207665930
samplei = 41, name = Kay M Tye, sample = 67, id = 498542680
samplei = 42, name = Silvio Savarese, sample = 153, id = 2835683058
samplei = 43, name = Sarah Certel, sample = 37, id = 700532262165676033
samplei = 44, name = Matthieu Louis, sample = 148, id = 919035620
samplei = 45, name = Tessa Montague, sample = 111, id = 985586833479356417
samplei = 46, name = Krystyna Keleman, sample = 72, id = 875432666555965444
samplei = 47, name = Sara Hooker, sample = 181, id = 731538535795163136
samplei = 48, name = Sebastian Seung, sample = 65, id = 122080635
samplei = 49, name = Noah Snavely, sample = 88, id = 15035863
samplei = 50, name = Stephanie Albin, sample = 123, id = 3072541911
samplei = 51, name = Nando de Freitas, sample = 36, id = 29843511
samplei = 52, name = Jim Keeley, sample = 138, id = 13537162
samplei = 53, name = Doris Tsao, sample = 177, id = 788467623629500416
samplei = 54, name = Surya Ganguli, sample = 56, id = 2235411914
samplei = 55, name = karel svoboda, sample = 132, id = 780291008
samplei = 56, name = Misha Ahrens, sample = 130, id = 1410832886
samplei = 57, name = Marius Pachitariu, sample = 129, id = 433741920
samplei = 58, name = Nelson Spruston, sample = 133, id = 288578245
samplei = 59, name = Ilana Witten, sample = 86, id = 3318332379
samplei = 60, name = Nicholas Turner, sample = 164, id = 951320423597211653
samplei = 61, name = Hanna Wallach, sample = 107, id = 823957466
samplei = 62, name = Caroline Palavicino-Maggio PhD, sample = 169, id = 47241596
samplei = 63, name = Misha Denil, sample = 33, id = 197684961
samplei = 64, name = Yann LeCun, sample = 0, id = 48008938
samplei = 65, name = Jeff Dean (@🏠), sample = 10, id = 911297187664949248
samplei = 66, name = Fernanda Viégas, sample = 60, id = 19920203
samplei = 67, name = Jakob Macke, sample = 53, id = 3147637105
samplei = 68, name = Adam J Calhoun, sample = 22, id = 636023721
samplei = 69, name = Andrew Fitzgibbon, sample = 12, id = 53514472
samplei = 70, name = Virginie Uhlmann, sample = 51, id = 959028649528840192
samplei = 71, name = Andrew Saxe, sample = 168, id = 1193222240202035200
samplei = 72, name = Alice Oh, sample = 145, id = 16520284
samplei = 73, name = Hugo Larochelle, sample = 35, id = 3333052551
samplei = 74, name = Ed Yong, sample = 141, id = 19767193
samplei = 75, name = Philipp Hanslovsky, sample = 101, id = 3131559501
samplei = 76, name = Sandeep Robert Datta, sample = 41, id = 1400517288
samplei = 77, name = mark cembrowski, sample = 122, id = 828056721750896640
samplei = 78, name = Gonzalo de Polavieja, sample = 144, id = 149895490
samplei = 79, name = Andrew Straw, sample = 58, id = 324367317
samplei = 80, name = Andreas Kay, sample = 188, id = 3192303453
samplei = 81, name = Elizabeth Hillman, sample = 157, id = 979697205664800768
samplei = 82, name = Christine Käser-Chen, sample = 93, id = 843706252517502977
samplei = 83, name = John Bogovic, sample = 43, id = 2848165007
samplei = 84, name = Leslie Vosshall PhD, sample = 66, id = 28912478
samplei = 85, name = Marcella Noorman, sample = 89, id = 1135303672219545600
samplei = 86, name = Adrienne Fairhall, sample = 142, id = 842164502422417409
samplei = 87, name = Carlos Ribeiro, sample = 84, id = 429146012
samplei = 88, name = Rex Parker⚾🐘🐘🐘⚾, sample = 30, id = 14230012
samplei = 89, name = Stephan Saalfeld, sample = 38, id = 930090512
samplei = 90, name = Janelle Shane, sample = 180, id = 2460047754
samplei = 91, name = Jan M Ache, sample = 69, id = 1091165441723166720
samplei = 92, name = John Langford, sample = 79, id = 1210596212140892160
samplei = 93, name = Fred "Replace Trump Now" Wolf, sample = 76, id = 22445339
samplei = 94, name = Juan Carlos Niebles, sample = 190, id = 559207602
samplei = 95, name = Rose Yu, sample = 166, id = 1890694861

Rate limit reached. Sleeping for: 585

samplei = 96, name = Lior Pachter, sample = 179, id = 31936449
samplei = 97, name = David Sussillo 🏠💻🤞🤪, sample = 77, id = 8000673177028812
80
samplei = 98, name = Ann Kennedy, sample = 71, id = 1143074659291680768
samplei = 99, name = Benjamin de Bivort, sample = 25, id = 259568572
```

In [119]:
```
# who do people i follow favorite?
otherfavnames = []

for samplei in range(len(otherfavnames),len(idxsample)):
    sample = idxsample[samplei]
    id = myinfo['idsknown'][sample]
    print('samplei = %d, name = %s, sample = %d, id = %d'%(samplei,myinfo['namesknown'][sample],sample,id))
    favscurr = get_favs(user_id=id,maxnfavs=maxnfavs)
    namescurr = get_fav_names(favscurr,verbose=False)
    #print(namescurr)
    otherfavnames.append(namescurr)
```

```
samplei = 0, name = Jeannette Bohg, sample = 152, id = 869862586610851840
samplei = 1, name = Laura Leal-Taixe, sample = 200, id = 748267272939020293
samplei = 2, name = Naomi Saphra, sample = 194, id = 215113195
samplei = 3, name = Bill Gates, sample = 167, id = 50393960

Rate limit reached. Sleeping for: 757

samplei = 4, name = Nan Rosemary Ke, sample = 182, id = 345129453
samplei = 5, name = Judith MitraniReiser, sample = 15, id = 737478121079906304
samplei = 6, name = Karla Kaun, sample = 19, id = 178588367
samplei = 7, name = Martha White, sample = 183, id = 543919023
samplei = 8, name = Davis Bennett, sample = 174, id = 946827254901936130

Rate limit reached. Sleeping for: 867

samplei = 9, name = Allan Wong, sample = 118, id = 49224993
samplei = 10, name = John Hodgman, sample = 92, id = 14348594
samplei = 11, name = Elizabeth C. Gorski, sample = 31, id = 217852227
samplei = 12, name = Ellie Heckscher, sample = 173, id = 35269421
samplei = 13, name = Athena Akrami, sample = 87, id = 1035389878605885440

Rate limit reached. Sleeping for: 869

samplei = 14, name = Anne Carpenter, sample = 4, id = 1431348835
samplei = 15, name = Barack Obama, sample = 156, id = 813286
samplei = 16, name = Alex Wild, sample = 192, id = 166022406
samplei = 17, name = Jane Wang, sample = 27, id = 22385548

Rate limit reached. Sleeping for: 869

samplei = 18, name = Sara Beery, sample = 155, id = 1237147828704575488
samplei = 19, name = Adam L. Taylor, sample = 73, id = 14840926
samplei = 20, name = Chris Potter, sample = 124, id = 1043667150
samplei = 21, name = Sasha Rush, sample = 178, id = 4558314927

Rate limit reached. Sleeping for: 868

samplei = 22, name = Eric Schreiter, sample = 126, id = 3797818101
samplei = 23, name = Alex Smola, sample = 94, id = 14986849
samplei = 24, name = Edith Zimmerman, sample = 105, id = 19087450
samplei = 25, name = Jen Heemstra, sample = 20, id = 2704715387
samplei = 26, name = Elena Rivas, sample = 115, id = 2789742375

Rate limit reached. Sleeping for: 868

samplei = 27, name = Tommy Caldwell, sample = 197, id = 246124588
samplei = 28, name = David Cho, sample = 106, id = 1014691
samplei = 29, name = Ted Pedersen, sample = 195, id = 1002638800575782912
samplei = 30, name = Elijah Cole, sample = 158, id = 1192003193921654784
samplei = 31, name = Martin Jones, sample = 52, id = 56786888

Rate limit reached. Sleeping for: 873

samplei = 32, name = Mike Economo, sample = 121, id = 938416059962609665
samplei = 33, name = Andrew S. Champion, sample = 50, id = 19301221
samplei = 34, name = Grace Vesom, sample = 23, id = 25320089
samplei = 35, name = Rachel Thomas, sample = 146, id = 1408142352

Rate limit reached. Sleeping for: 872

samplei = 36, name = Shakir Mohamed, sample = 185, id = 476582730
samplei = 37, name = Matt Gritzmacher, sample = 143, id = 223734352
samplei = 38, name = Erich Jarvis, sample = 83, id = 314158631
samplei = 39, name = Jewel Burks Solomon, sample = 13, id = 159315527

Rate limit reached. Sleeping for: 867
```

```
samplei = 40, name = Serena Yeung, sample = 151, id = 207665930
samplei = 41, name = Kay M Tye, sample = 67, id = 498542680
samplei = 42, name = Silvio Savarese, sample = 153, id = 2835683058
samplei = 43, name = Sarah Certel, sample = 37, id = 700532262165676033
samplei = 44, name = Matthieu Louis, sample = 148, id = 919035620
samplei = 45, name = Tessa Montague, sample = 111, id = 985586833479356417

Rate limit reached. Sleeping for: 865

samplei = 46, name = Krystyna Keleman, sample = 72, id = 875432666555965444
samplei = 47, name = Sara Hooker, sample = 181, id = 731538535795163136
samplei = 48, name = Sebastian Seung, sample = 65, id = 122080635
samplei = 49, name = Noah Snavely, sample = 88, id = 15035863

Rate limit reached. Sleeping for: 872

samplei = 50, name = Stephanie Albin, sample = 123, id = 3072541911
samplei = 51, name = Nando de Freitas, sample = 36, id = 29843511
samplei = 52, name = Jim Keeley, sample = 138, id = 13537162
samplei = 53, name = Doris Tsao, sample = 177, id = 788467623629500416

Rate limit reached. Sleeping for: 872

samplei = 54, name = Surya Ganguli, sample = 56, id = 2235411914
samplei = 55, name = karel svoboda, sample = 132, id = 780291008
samplei = 56, name = Misha Ahrens, sample = 130, id = 1410832886
samplei = 57, name = Marius Pachitariu, sample = 129, id = 433741920

Rate limit reached. Sleeping for: 872

samplei = 58, name = Nelson Spruston, sample = 133, id = 288578245
samplei = 59, name = Ilana Witten, sample = 86, id = 3318332379
samplei = 60, name = Nicholas Turner, sample = 164, id = 951320423597211653
samplei = 61, name = Hanna Wallach, sample = 107, id = 823957466

Rate limit reached. Sleeping for: 872

samplei = 62, name = Caroline Palavicino-Maggio PhD, sample = 169, id = 47241596
samplei = 63, name = Misha Denil, sample = 33, id = 197684961
samplei = 64, name = Yann LeCun, sample = 0, id = 48008938
samplei = 65, name = Jeff Dean (@🏠), sample = 10, id = 911297187664949248

Rate limit reached. Sleeping for: 872

samplei = 66, name = Fernanda Viégas, sample = 60, id = 19920203
samplei = 67, name = Jakob Macke, sample = 53, id = 3147637105
samplei = 68, name = Adam J Calhoun, sample = 22, id = 636023721
samplei = 69, name = Andrew Fitzgibbon, sample = 12, id = 53514472

Rate limit reached. Sleeping for: 873

samplei = 70, name = Virginie Uhlmann, sample = 51, id = 959028649528840192
samplei = 71, name = Andrew Saxe, sample = 168, id = 1193222240202035200
samplei = 72, name = Alice Oh, sample = 145, id = 16520284
samplei = 73, name = Hugo Larochelle, sample = 35, id = 3333052551

Rate limit reached. Sleeping for: 874

samplei = 74, name = Ed Yong, sample = 141, id = 19767193
samplei = 75, name = Philipp Hanslovsky, sample = 101, id = 3131559501
samplei = 76, name = Sandeep Robert Datta, sample = 41, id = 1400517288
samplei = 77, name = mark cembrowski, sample = 122, id = 828056721750896640

Rate limit reached. Sleeping for: 873
```

```
samplei = 78, name = Gonzalo de Polavieja, sample = 144, id = 149895490
samplei = 79, name = Andrew Straw, sample = 58, id = 324367317
samplei = 80, name = Andreas Kay, sample = 188, id = 3192303453
samplei = 81, name = Elizabeth Hillman, sample = 157, id = 979697205664800768

Rate limit reached. Sleeping for: 874

samplei = 82, name = Christine Käser-Chen, sample = 93, id = 843706252517502977
samplei = 83, name = John Bogovic, sample = 43, id = 2848165007
samplei = 84, name = Leslie Vosshall PhD, sample = 66, id = 28912478
samplei = 85, name = Marcella Noorman, sample = 89, id = 1135303672219545600
samplei = 86, name = Adrienne Fairhall, sample = 142, id = 842164502422417409
samplei = 87, name = Carlos Ribeiro, sample = 84, id = 429146012

Rate limit reached. Sleeping for: 872

samplei = 88, name = Rex Parker⚾🐘🐘🐘⚾, sample = 30, id = 14230012
samplei = 89, name = Stephan Saalfeld, sample = 38, id = 930090512
samplei = 90, name = Janelle Shane, sample = 180, id = 2460047754

Rate limit reached. Sleeping for: 868

samplei = 91, name = Jan M Ache, sample = 69, id = 1091165441723166720
samplei = 92, name = John Langford, sample = 79, id = 1210596212140892160
samplei = 93, name = Fred "Replace Trump Now" Wolf, sample = 76, id = 22445339
samplei = 94, name = Juan Carlos Niebles, sample = 190, id = 559207602
samplei = 95, name = Rose Yu, sample = 166, id = 1890694861

Rate limit reached. Sleeping for: 872

samplei = 96, name = Lior Pachter, sample = 179, id = 31936449
samplei = 97, name = David Sussillo 🏠💻👆🤓, sample = 77, id = 8000673177028812
80
samplei = 98, name = Ann Kennedy, sample = 71, id = 1143074659291680768
samplei = 99, name = Benjamin de Bivort, sample = 25, id = 259568572
```

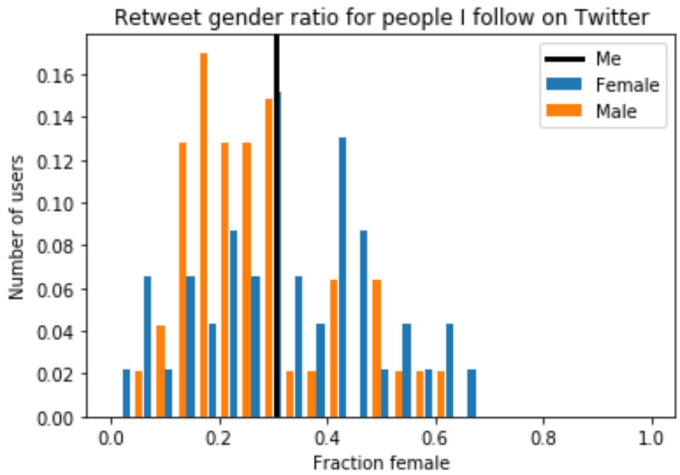
In [124]:
```
# compute gender ratio info
otherretweetinfo = []
othernretweets = []
otherfavinfo = []
othernfavs = []
for i in range(len(names1)):
    #print('i = %d: %s -> first name = %s'%(i,sample_user_objs[i].name,names1
[i][0]))
    sample = idxsample[i]
    namescurr = otherretweetnames[i]
    otherretweetinfo.append(compute_gender_info(namescurr,verbose=False,sigma=2))
    ncurr = len(otherretweetinfo[-1]['knowngs'])
    othernretweets.append(ncurr)
    namescurr = otherfavnames[i]
    otherfavinfo.append(compute_gender_info(namescurr,verbose=False,sigma=2))
    nfavcurr = len(otherfavinfo[-1]['knowngs'])
    othernfavs.append(nfavcurr)
    if ncurr > 0:
        ffr = otherretweetinfo[-1]['fracfemale'][-1]
    else:
        ffr = np.nan
    if nfavcurr > 0:
        fff = otherfavinfo[-1]['fracfemale'][-1]
    else:
        fff = np.nan
    print('i = %d, %s retweet: %f (n = %d), fav: %f (n = %d))'%(i,myinfo['namesknow
wn'][sample],ffr,ncurr,fff,nfavcurr))
```

```
i = 0, Jeannette Bohg retweet: 0.361809 (n = 199), fav: 0.418367 (n = 196))
i = 1, Laura Leal-Taixe retweet: 0.078947 (n = 76), fav: 0.136882 (n = 263))
i = 2, Naomi Saphra retweet: 0.413043 (n = 184), fav: 0.362903 (n = 248))
i = 3, Bill Gates retweet: 0.448276 (n = 29), fav: 0.348837 (n = 86))
i = 4, Nan Rosemary Ke retweet: 0.152778 (n = 72), fav: 0.216450 (n = 231))
i = 5, Judith MitraniReiser retweet: 0.427184 (n = 206), fav: 0.483108 (n = 29
6))
i = 6, Karla Kaun retweet: 0.475113 (n = 221), fav: 0.523466 (n = 277))
i = 7, Martha White retweet: 0.000000 (n = 2), fav: nan (n = 0))
i = 8, Davis Bennett retweet: 0.144828 (n = 145), fav: 0.208000 (n = 250))
i = 9, Allan Wong retweet: 1.000000 (n = 1), fav: 0.500000 (n = 2))
i = 10, John Hodgman retweet: 0.313830 (n = 188), fav: 0.338028 (n = 284))
i = 11, Elizabeth C. Gorski retweet: 0.262712 (n = 118), fav: 0.384615 (n = 10
4))
i = 12, Ellie Heckscher retweet: 0.266234 (n = 154), fav: 0.350000 (n = 280))
i = 13, Athena Akrami retweet: 0.292000 (n = 250), fav: 0.344156 (n = 308))
i = 14, Anne Carpenter retweet: 0.350806 (n = 248), fav: 0.425856 (n = 263))
i = 15, Barack Obama retweet: 0.607143 (n = 28), fav: 0.600000 (n = 5))
i = 16, Alex Wild retweet: 0.298343 (n = 181), fav: 0.279167 (n = 240))
i = 17, Jane Wang retweet: 0.194313 (n = 211), fav: 0.255245 (n = 286))
i = 18, Sara Beery retweet: 0.416667 (n = 84), fav: 0.537815 (n = 238))
i = 19, Adam L. Taylor retweet: 0.214575 (n = 247), fav: 0.850829 (n = 362))
i = 20, Chris Potter retweet: 0.292237 (n = 219), fav: 0.329060 (n = 234))
i = 21, Sasha Rush retweet: 0.102941 (n = 204), fav: 0.189189 (n = 259))
i = 22, Eric Schreiter retweet: 0.214286 (n = 42), fav: 0.188172 (n = 186))
i = 23, Alex Smola retweet: 0.075758 (n = 66), fav: 0.100000 (n = 80))
i = 24, Edith Zimmerman retweet: 0.671569 (n = 204), fav: 0.613559 (n = 295))
i = 25, Jen Heemstra retweet: 0.526946 (n = 167), fav: 0.581481 (n = 270))
i = 26, Elena Rivas retweet: 0.400000 (n = 105), fav: 0.461794 (n = 301))
i = 27, Tommy Caldwell retweet: 0.333333 (n = 3), fav: 0.200000 (n = 5))
i = 28, David Cho retweet: 0.306452 (n = 248), fav: 0.666667 (n = 3))
i = 29, Ted Pedersen retweet: 0.457831 (n = 83), fav: 0.625000 (n = 256))
i = 30, Elijah Cole retweet: 0.388889 (n = 18), fav: 0.247012 (n = 251))
i = 31, Martin Jones retweet: 0.325843 (n = 178), fav: 0.415525 (n = 219))
i = 32, Mike Economo retweet: 0.169231 (n = 65), fav: 0.161765 (n = 68))
i = 33, Andrew S. Champion retweet: 0.046729 (n = 107), fav: 0.090580 (n = 276))
i = 34, Grace Vesom retweet: 0.562500 (n = 128), fav: 0.656827 (n = 271))
i = 35, Rachel Thomas retweet: 0.632850 (n = 207), fav: 0.469534 (n = 279))
i = 36, Shakir Mohamed retweet: 0.335260 (n = 173), fav: 0.360731 (n = 219))
i = 37, Matt Gritzmacher retweet: 0.309524 (n = 210), fav: 0.381579 (n = 304))
i = 38, Erich Jarvis retweet: 0.551913 (n = 183), fav: 0.710317 (n = 252))
i = 39, Jewel Burks Solomon retweet: 0.462500 (n = 160), fav: 0.427350 (n = 23
4))
i = 40, Serena Yeung retweet: 0.214286 (n = 14), fav: 0.202899 (n = 69))
i = 41, Kay M Tye retweet: 0.510823 (n = 231), fav: 0.525253 (n = 297))
i = 42, Silvio Savarese retweet: 0.029412 (n = 34), fav: 0.100000 (n = 20))
i = 43, Sarah Certel retweet: 0.407407 (n = 27), fav: 0.388704 (n = 301))
i = 44, Matthieu Louis retweet: 0.431818 (n = 44), fav: 0.516129 (n = 31))
i = 45, Tessa Montague retweet: 0.280374 (n = 214), fav: 0.428571 (n = 266))
i = 46, Krystyna Keleman retweet: 0.000000 (n = 2), fav: 0.388889 (n = 18))
i = 47, Sara Hooker retweet: 0.258278 (n = 151), fav: 0.390244 (n = 246))
i = 48, Sebastian Seung retweet: 0.224490 (n = 98), fav: 0.242574 (n = 202))
i = 49, Noah Snavely retweet: 0.153846 (n = 104), fav: 0.248175 (n = 274))
i = 50, Stephanie Albin retweet: 0.581395 (n = 129), fav: 0.639344 (n = 244))
i = 51, Nando de Freitas retweet: 0.188482 (n = 191), fav: 0.152344 (n = 256))
i = 52, Jim Keeley retweet: 0.197368 (n = 76), fav: 0.271605 (n = 162))
i = 53, Doris Tsao retweet: 0.288591 (n = 149), fav: 0.227723 (n = 303))
i = 54, Surya Ganguli retweet: 0.185000 (n = 200), fav: 0.263158 (n = 285))
i = 55, karel svoboda retweet: 0.215190 (n = 79), fav: 0.313725 (n = 153))
i = 56, Misha Ahrens retweet: 0.172727 (n = 220), fav: 0.222973 (n = 296))
i = 57, Marius Pachitariu retweet: 0.207843 (n = 255), fav: 0.213483 (n = 267))
i = 58, Nelson Spruston retweet: 0.300000 (n = 10), fav: 0.250000 (n = 4))
i = 59, Ilana Witten retweet: 0.400735 (n = 272), fav: 0.446541 (n = 318))
i = 60, Nicholas Turner retweet: 0.160000 (n = 25), fav: 0.250825 (n = 303))
```

In [153]:
```
# plot a histogram of frac female retweet for samples and for me

fracfemale_retweet_f = np.zeros(nsample_female)
fracfemale_retweet_m = np.zeros(nsample_female)
nsample_retweet_f = 0
nsample_retweet_m = 0
minnretweets = 10
retweet_hist_range = (0.,1.)
nbins = 25

for i in range(nsample_female):
    if othernretweets[i] < minnretweets:
        fracfemale_retweet_f[i] = np.nan
        print('Bad sample %d'%i)
    else:
        fracfemale_retweet_f[i] = otherretweetinfo[i]['fracfemale'][-1]
        nsample_retweet_f += 1
    if othernretweets[i+nsample_female] < minnretweets:
        fracfemale_retweet_m[i] = np.nan
        print('Bad sample %d'%(i+nsample_female))
    else:
        fracfemale_retweet_m[i] = otherretweetinfo[i+nsample_female]['fracfemale
'][-1]
        nsample_retweet_m += 1


idxgood_f = np.isnan(fracfemale_retweet_f) == False
idxgood_m = np.isnan(fracfemale_retweet_m) == False

myprctile_f = np.count_nonzero(fracfemale_retweet_f[idxgood_f]<=myretweetinfo['fra
cfemale'][-1])/nsample_retweet_f
myprctile_m = np.count_nonzero(fracfemale_retweet_m[idxgood_m]<=myretweetinfo['fra
cfemale'][-1])/nsample_retweet_m
print('My percentile among women I follow: %f'%(myprctile_f*100.))
print('My percentile among men I follow: %f'%(myprctile_m*100.))
print('My percentile among people I follow: %f'%((myprctile_f+myprctile_m)*50.))

counts_f,edges = np.histogram(fracfemale_retweet_f[idxgood_f],bins=nbins,range=ret
weet_hist_range)
counts_m,edges = np.histogram(fracfemale_retweet_m[idxgood_m],bins=nbins,range=ret
weet_hist_range)
ctrs_retweet = (edges[1:]+edges[:-1])/2.
width_retweet = edges[1]-edges[0]

normcounts_f = counts_f / np.sum(counts_f)
normcounts_m = counts_m / np.sum(counts_m)
plt.bar(ctrs_retweet+width_retweet/5.,normcounts_f,width=width_retweet*.35,color=c
olorf,label='Female')
plt.bar(ctrs_retweet-width_retweet/5.,normcounts_m,width=width_retweet*.35,color=c
olorm,label='Male')
ylim = plt.gca().get_ylim()
plt.plot([myretweetinfo['fracfemale'][-1]]*2,ylim,'k-',lw=3,label='Me')
plt.gca().set_ylim(ylim)
plt.legend()
plt.xlabel('Fraction female')
plt.ylabel('Number of users')
plt.title('Retweet gender ratio for people I follow on Twitter')

print('Mean, median fraction of retweeted statuses from women for women I follow:
%f, %f'%(np.mean(fracfemale_retweet_f[idxgood_f]),np.median(fracfemale_retweet_f[i
dxgood_f])))
print('Mean, median fraction of retweeted statuses from women for men I follow: %
f, %f'%(np.mean(fracfemale_retweet_m[idxgood_m]),np.median(fracfemale_retweet_m[id
xgood_m])))
```

```
Bad sample 7
Bad sample 9
Bad sample 27
Bad sample 80
Bad sample 85
Bad sample 92
Bad sample 46
My percentile among women I follow: 45.652174
My percentile among men I follow: 72.340426
My percentile among people I follow: 58.996300
Mean, median fraction of retweeted statuses from women for women I follow: 0.327
448, 0.311677
Mean, median fraction of retweeted statuses from women for men I follow: 0.26946
7, 0.244019
```

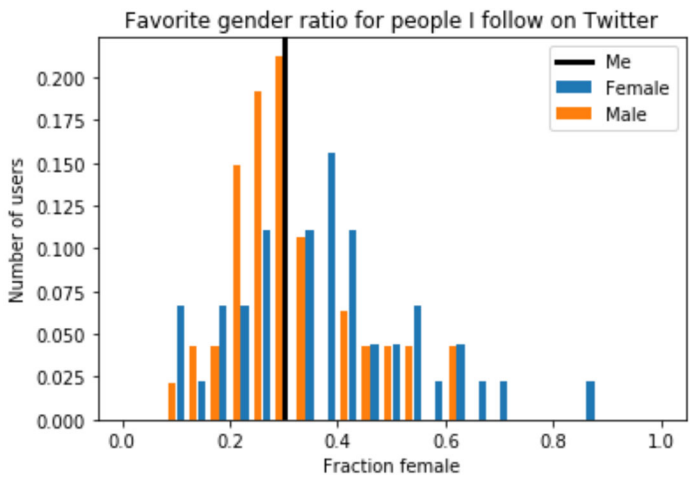
Out[153]: `MannwhitneyuResult(statistic=1347.0, pvalue=0.020666258736301783)`



In [152]:
```
# plot histogram of frac female favorite for samples and for me

fracfemale_fav_f = np.zeros(nsample_female)
fracfemale_fav_m = np.zeros(nsample_female)
nsample_fav_f = 0
nsample_fav_m = 0
minnfavs = 10
for i in range(nsample_female):
    if othernfavs[i] < minnfavs:
        fracfemale_fav_f[i] = np.nan
        print('Bad sample %d'%i)
    else:
        fracfemale_fav_f[i] = otherfavinfo[i]['fracfemale'][-1]
        nsample_fav_f += 1
    if othernfavs[i+nsample_female] < minnfavs:
        fracfemale_fav_m[i] = np.nan
        print('Bad sample %d'%(i+nsample_female))
    else:
        fracfemale_fav_m[i] = otherfavinfo[i+nsample_female]['fracfemale'][-1]
        nsample_fav_m += 1


idxgood_f = np.isnan(fracfemale_fav_f) == False
idxgood_m = np.isnan(fracfemale_fav_m) == False

myprctile_f = np.count_nonzero(fracfemale_fav_f[idxgood_f]<=myfavinfo['fracfemale
'][-1])/nsample_fav_f
myprctile_m = np.count_nonzero(fracfemale_fav_m[idxgood_m]<=myfavinfo['fracfemale
'][-1])/nsample_fav_m
print('My percentile among women I follow: %f'%(myprctile_f*100.))
print('My percentile among men I follow: %f'%(myprctile_m*100.))
print('My percentile among people I follow: %f'%((myprctile_f+myprctile_m)*50.))

counts_f,edges = np.histogram(fracfemale_fav_f[idxgood_f],bins=nbins,range=retweet
_hist_range)
counts_m,edges = np.histogram(fracfemale_fav_m[idxgood_m],bins=nbins,range=retweet
_hist_range)
ctrs_fav = (edges[1:]+edges[:-1])/2.
width_fav = edges[1]-edges[0]
normcounts_f = counts_f / np.sum(counts_f)
normcounts_m = counts_m / np.sum(counts_m)
plt.bar(ctrs_fav+width_fav/5.,normcounts_f,width=width_fav*.35,color=colorf,label=
'Female')
plt.bar(ctrs_fav-width_fav/5.,normcounts_m,width=width_fav*.35,color=colorm,label=
'Male')
ylim = plt.gca().get_ylim()
plt.plot([myfavinfo['fracfemale'][-1]]*2,ylim,'k-',lw=3,label='Me')
plt.gca().set_ylim(ylim)
plt.legend()
plt.xlabel('Fraction female')
plt.ylabel('Number of users')
plt.title('Favorite gender ratio for people I follow on Twitter')

print('Mean, median fraction of favorited statuses from women for women I follow:
%f, %f'%(np.mean(fracfemale_fav_f[idxgood_f]),np.median(fracfemale_fav_f[idxgood_
f])))
print('Mean, median fraction of favorited statuses from women for men I follow: %
f, %f'%(np.mean(fracfemale_fav_m[idxgood_m]),np.median(fracfemale_fav_m[idxgood_
m])))

stats.mannwhitneyu(fracfemale_fav_f[idxgood_f],fracfemale_fav_m[idxgood_m],alterna
tive='greater')
```

```
Bad sample 7
Bad sample 58
Bad sample 9
Bad sample 15
Bad sample 27
Bad sample 28
Bad sample 80
Bad sample 92
My percentile among women I follow: 33.333333
My percentile among men I follow: 53.191489
My percentile among people I follow: 43.262411
Mean, median fraction of favorited statuses from women for women I follow: 0.375
659, 0.381579
Mean, median fraction of favorited statuses from women for men I follow: 0.31780
3, 0.300654
```

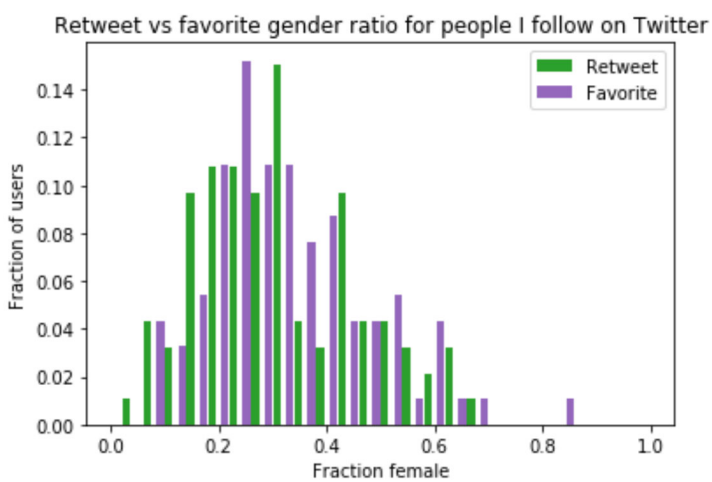
Out[152]: `MannwhitneyuResult(statistic=1282.0, pvalue=0.04009198093587479)`



In [149]:
```
# plot histogram for retweets vs favorites

ff_retweet_x = np.concatenate((fracfemale_retweet_f,fracfemale_retweet_m),0)
ff_fav_x = np.concatenate((fracfemale_fav_f,fracfemale_fav_m),0)

idxgood_retweet = np.isnan(ff_retweet_x) == False
idxgood_fav = np.isnan(ff_fav_x) == False

counts_retweet,edges = np.histogram(ff_retweet_x[idxgood_retweet],bins=nbins,range
=retweet_hist_range)
counts_fav,edges = np.histogram(ff_fav_x[idxgood_fav],bins=nbins,range=retweet_his
t_range)
ctrs = (edges[1:]+edges[:-1])/2.
width = edges[1]-edges[0]
normcounts_retweet = counts_retweet / np.sum(counts_retweet)
normcounts_fav = counts_fav / np.sum(counts_fav)
plt.bar(ctrs+width/5.,normcounts_retweet,width=width*.35,color=colorretweet,label=
'Retweet')
plt.bar(ctrs-width/5.,normcounts_fav,width=width*.35,color=colorfav,label='Favorit
e')
# ylim = plt.gca().get_ylim()
# plt.plot([myretweetinfo['fracfemale'][-1]]*2,ylim,'k-',lw=3,label='Me')
# plt.gca().set_ylim(ylim)
plt.legend()
plt.xlabel('Fraction female')
plt.ylabel('Fraction of users')
plt.title('Retweet vs favorite gender ratio for people I follow on Twitter')

print('Mean, median fraction of retweeted statuses from women: %f, %f'%(np.mean(ff
_retweet_x[idxgood_retweet]),np.median(ff_retweet_x[idxgood_retweet])))
print('Mean, median fraction of favorited statuses from women: %f, %f'%(np.mean(ff
_fav_x[idxgood_fav]),np.median(ff_fav_x[idxgood_fav])))

stats.mannwhitneyu(ff_retweet_x[idxgood_retweet],ff_fav_x[idxgood_fav])
```

```
Mean, median fraction of retweeted statuses from women: 0.298145, 0.280374
Mean, median fraction of favorited statuses from women: 0.346102, 0.321776
```

Out[149]:
```
MannwhitneyuResult(statistic=3464.5, pvalue=0.012791159375702027)
```



In [122]:
```
import pickle
filename = 'C:/Code/TwitterGenderRatio/test20200517.pickle'
fid = open(filename,'wb')
pickle.dump({'ids': ids, 'names': names,'idxsample': idxsample, 'names1': names1,
'ids1': ids1,
            'myretweetnames': myretweetnames,'myretweettypes': myretweettypes,'myi
sretweet': myisretweet,
            'myfavnames': myfavnames,'otherretweetnames': otherretweetnames,'other
favnames': otherfavnames},fid)
fid.close()
```