# Scaling Up Peptide Sequencing ¶

- More Errors
- More Residues

# From Last Time

```
AminoAcid = {
    'A': 'Alanine', 'C': 'Cysteine', 'D': 'Aspartic acid', 'E': 'Glutamic acid',
    'F': 'Phenylalanine', 'G': 'Glycine', 'H': 'Histidine', 'I': 'Isoleucine',
    'K': 'Lysine', 'L': 'Leucine', 'M': 'Methionine', 'N': 'Asparagine',
    'P': 'Proline', 'Q': 'Glutamine', 'R': 'Arginine', 'S': 'Serine',
    'T': 'Theronine', 'V': 'Valine', 'W': 'Tryptophan', 'Y': 'Tyrosine',
    '*': 'STOP'
}

AminoAbbrv = {
    'A': 'Ala', 'C': 'Cys', 'D': 'Asp', 'E': 'Glu',
    'F': 'Phe', 'G': 'Gly', 'H': 'His', 'I': 'Ile',
    'K': 'Lys', 'L': 'Leu', 'M': 'Met', 'N': 'Asn',
    'P': 'Pro', 'Q': 'Gln', 'R': 'Arg', 'S': 'Ser',
    'T': 'Thr', 'V': 'Val', 'W': 'Trp', 'Y': 'Tyr',
    '*': 'STP'
}

# Now it's time to use this dictionary!
Daltons = {
    'A':  71, 'C': 103, 'D': 115, 'E': 129,
    'F': 147, 'G':  57, 'H': 137, 'I': 113,
    'K': 128, 'L': 113, 'M': 131, 'N': 114,
    'P':  97, 'Q': 128, 'R': 156, 'S':  87,
    'T': 101, 'V':  99, 'W': 186, 'Y': 163
}

def TheoreticalSpectrum(peptide):
    # Generate every possible fragment of a peptide
    spectrum = set()
    for fragLength in xrange(1,len(peptide)+1):
        for start in xrange(0,len(peptide)-fragLength+1):
            seq = peptide[start:start+fragLength]
            spectrum.add(sum([Daltons[res] for res in seq]))
    return sorted(spectrum)
```

# Recall our golf tourney approach

```
import itertools

def LeaderboardFindPeptide(noisySpectrum, cutThreshold=0.05):
    # Golf Tournament Heuristic
    spectrum = set(noisySpectrum)
    target = max(noisySpectrum)
    players = [''.join(peptide) for peptide in itertools.product(Daltons.keys(), repeat=2)]
    round = 1
    currentLeader = [0.0, '']
    while True:
        print "%8d Players in round %d [%5.4f]" % (len(players), round, currentLeader[0])
        leaderboard = []
        for prefix in players:
            testSpectrum = set(TheoreticalSpectrum(prefix))
            totalWeight = max(testSpectrum)
            score = len(spectrum & testSpectrum)/float(len(spectrum | testSpectrum))
            if (score > currentLeader[0]):
                currentLeader = [score, prefix]
            elif (score == currentLeader[0]):
                currentLeader += [prefix]
            if (totalWeight < target):
                leaderboard.append((score, prefix))
        remaining = len(leaderboard)
        if (remaining == 0):
            print "Done, no sequences can be extended"
            break
        leaderboard.sort(reverse=True)
        # Prune the larger of the top 5% or the top 5 players
        cut = leaderboard[max(min(5,remaining-1),int(remaining*cutThreshold))][0]
        players = [p+r for s, p in leaderboard if s >= cut for r in Daltons.iterkeys()]
        round += 1
    return currentLeader
```

# Let's try a Noisier Spectrum

```
# generate a synthetic experimental spectrum with 60% Error
import random
random.seed(1961)

TyrocidineB1 = "VKLFPWFNQY"
print TyrocidineB1
spectrum = TheoreticalSpectrum(TyrocidineB1)
print len(spectrum), spectrum

# Pick around ~60% at random to remove
missingMass = random.sample(spectrum[:-1], 30)
print "Missing Masses = ", missingMass

# Add back another ~10% of false, but actual, peptide masses
falseMass = []
for i in xrange(5):
    fragment = ''.join(random.sample(Daltons.keys(), random.randint(2,len(TyrocidineB1)-2)))
    weight = sum([Daltons[residue] for residue in fragment])
    falseMass.append(weight)
print "False Masses = ", falseMass

experimentalSpectrum = sorted(set([mass for mass in spectrum if mass not in missingMass] + falseMass))

print len(experimentalSpectrum), experimentalSpectrum
```

```
VKLFPWFNQY
51 [97, 99, 113, 114, 128, 147, 163, 186, 227, 241, 242, 244, 260, 261, 283, 291, 333, 340, 357, 388, 389, 405, 430, 447, 485, 487, 543, 544, 55
2, 575, 577, 584, 671, 672, 690, 691, 738, 770, 804, 818, 819, 835, 917, 932, 982, 1031, 1060, 1095, 1159, 1223, 1322]
Missing Masses =  [1159, 114, 691, 186, 819, 357, 291, 543, 1223, 147, 671, 97, 388, 552, 447, 770, 672, 261, 738, 487, 577, 485, 932, 1031, 690,
389, 340, 575, 113, 260]
False Masses =  [356, 160, 572, 879, 244]
25 [99, 128, 160, 163, 227, 241, 242, 244, 283, 333, 356, 405, 430, 544, 572, 584, 804, 818, 835, 879, 917, 982, 1060, 1095, 1322]
```

# Find peptides via the leaderboard approach

```
spectrum = TheoreticalSpectrum(TyrocidineB1)
experimentalSpectrum = [mass for mass in spectrum if mass not in missingMass] + falseMass
%time winners = LeaderboardFindPeptide(experimentalSpectrum)
print winners
print len(winners) - 1, "Candidate residues with", winners[0], 'matches'
print TyrocidineB1, TyrocidineB1 in winners
```

```
     400 Players in round 1 [0.0000]
     440 Players in round 2 [0.1200]
     600 Players in round 3 [0.1481]
    1480 Players in round 4 [0.2069]
    1840 Players in round 5 [0.2121]
    2320 Players in round 6 [0.2222]
    5200 Players in round 7 [0.2619]
    5360 Players in round 8 [0.2826]
    8640 Players in round 9 [0.2826]
    9040 Players in round 10 [0.3220]
    8320 Players in round 11 [0.3220]
     480 Players in round 12 [0.3220]
Done, no sequences can be extended
CPU times: user 3.91 s, sys: 75 ms, total: 3.99 s
Wall time: 3.9 s
[0.3220338983050847, 'VQLDEWFNQY', 'VQLDEWFNKY', 'VQIDEWFNQY', 'VQIDEWFNKY', 'VKLDEWFNQY', 'VKLDEWFNKY', 'VKIDEWFNQY', 'VKIDEWFNKY']
8 Candidate residues with 0.322033898305 matches
VKLFPWFNQY False
```

# Let's try the example in the book

The example on page 66 of the book gives a different answer using our method, because the book does not *normalize* it's similarity metric. While 'VKLFPADFNQY' has one more match, 'VKLFPWFNQY' it also adds 10 unmatched peaks.

```
missingBook = [357,430,543,671,747,778,1031,1061,1225]
falseBook = [115,256,309,330,347,385,435,599,608,653,717,827]
spectrum = TheoreticalSpectrum(TyrocidineB1)
experimentalSpectrum = sorted(set([mass for mass in spectrum if mass not in missingBook] + falseBook))
%time winners = LeaderboardFindPeptide(experimentalSpectrum)
print winners
print len(winners) - 1, "Candidate residues with", winners[0], 'matches'
print TyrocidineB1, TyrocidineB1 in winners
print
x = set(TheoreticalSpectrum('VKLFPADFNQY'))
y = set(TheoreticalSpectrum('VKLFPWFNQY'))
z = set(experimentalSpectrum)
print "%s Matches = %d, Union = %d" % ('VKLFPADFNQY', len(x & z), len(x | z))
print "%s Matches = %d, Union = %d" % ('VKLFPWFNQY', len(y & z), len(y | z))
```

```
     400 Players in round 1 [0.0000]
     760 Players in round 2 [0.0517]
    2680 Players in round 3 [0.1034]
    3800 Players in round 4 [0.1724]
    6000 Players in round 5 [0.2586]
    7320 Players in round 6 [0.3390]
    9160 Players in round 7 [0.4237]
   11280 Players in round 8 [0.5082]
   11440 Players in round 9 [0.5968]
    7840 Players in round 10 [0.7302]
     800 Players in round 11 [0.7302]
Done, no sequences can be extended
CPU times: user 4.54 s, sys: 107 ms, total: 4.65 s
Wall time: 4.53 s
[0.7301587301587301, 'YQNFWPFLQV', 'YQNFWPFLKV', 'YQNFWPFIQV', 'YQNFWPFIKV', 'YKNFWPFLQV', 'YKNFWPFLKV', 'YKNFWPFIQV', 'YKNFWPFIKV', 'VQLFPWFNQ
Y', 'VQLFPWFNKY', 'VQIFPWFNQY', 'VQIFPWFNKY', 'VKLFPWFNQY', 'VKLFPWFNKY', 'VKIFPWFNQY', 'VKIFPWFNKY']
16 Candidate residues with 0.730158730159 matches
```

# A New Idea

- Maybe we are still not using our spectrum to its fullest extent
- Is there some information about *missing masses* that we can extract?

# Information in the Mass Differences

- Recall the *theoretical spectrum* of "PLAY" is [71, 97, 113, 163, 184, 210, 234, 281, 347, 444]
- Suppose we remove masses 71 and 163, can we get them back?
- Let's generate a table of all pair-wise differences between the observed peaks
- Notice that *interesting* numbers, (***71***, *97*, *113*, 137, ***163***, *234*) are repeated in the table

| | **97** | **113** | **184** | **210** | **234** | **281** | **347** | **444** |
|---|---|---|---|---|---|---|---|---|
| **97** | | 16 | 87 | 113 | 137 | 184 | 250 | 347 |
| **113** | | | 71 | 97 | 121 | 168 | 234 | 331 |
| **184** | | | | 26 | 50 | 97 | 163 | 260 |
| **210** | | | | | 24 | 71 | 137 | 234 |
| **234** | | | | | | 47 | 113 | 210 |
| **281** | | | | | | | 66 | 163 |
| **347** | | | | | | | | 97 |

- Why does this work?
- This table of differences is called a *Spectral Convolution*

# Spectral Convolution

- Spectral Convolution gives us an approach for recovering some missing masses
- Given a noisy experimental spectrum
  1. Compute its spectral convolution
  2. Add frequent masses above some threshold to the spectrum
  3. Infer the peptide sequence

```
def SpectralConvolution(spectrum):
    delta = {}
    for i in xrange(len(spectrum)-1):
        for j in xrange(i+1,len(spectrum)):
            diff = abs(spectrum[j] - spectrum[i])
            delta[diff] = delta.get(diff, 0) + 1
    return delta

spectrum = TheoreticalSpectrum(TyrocidineB1)
experimentalSpectrum = sorted(set([mass for mass in spectrum if mass not in missingMass] + falseMass))
specConv = SpectralConvolution(sorted(experimentalSpectrum))
N = 0
for delta, count in sorted(specConv.iteritems()):
    if (count >= 2) and (delta not in experimentalSpectrum) and (delta > min(Daltons.values())):
        print delta, "appears", count, "times*\t" if delta in missingMass else "times\t",
        experimentalSpectrum.append(delta)
        N += 1
        if (N % 4 == 0):
            print
print sorted(missingMass)
```

```
61 appears 2 times      64 appears 2 times      78 appears 2 times      81 appears 2 times
82 appears 2 times      113 appears 3 times*    114 appears 3 times*    142 appears 2 times
143 appears 2 times     147 appears 2 times*    164 appears 2 times     178 appears 3 times
188 appears 2 times     216 appears 2 times     228 appears 2 times     234 appears 2 times
251 appears 2 times     260 appears 2 times*    277 appears 2 times     291 appears 2 times*
302 appears 2 times     331 appears 2 times     340 appears 2 times*    345 appears 2 times
```

# Noisey spectrum enhanced by spectral convolution

```
winners = LeaderboardFindPeptide(experimentalSpectrum)
print winners
print len(winners) - 1, "Candidate residues with", winners[0], 'matches'
print TyrocidineB1, TyrocidineB1 in winners
```

```
     400 Players in round 1 [0.0000]
     600 Players in round 2 [0.0469]
    1880 Players in round 3 [0.0938]
    2000 Players in round 4 [0.1385]
    4080 Players in round 5 [0.1791]
    8640 Players in round 6 [0.1972]
   10560 Players in round 7 [0.2676]
   11720 Players in round 8 [0.3243]
   12480 Players in round 9 [0.3636]
   11200 Players in round 10 [0.4375]
    8560 Players in round 11 [0.4419]
    1200 Players in round 12 [0.4419]
     160 Players in round 13 [0.4419]
Done, no sequences can be extended
[0.4418604651162791, 'VQLFPSFVNYK', 'VQLFPSFVNYQ', 'VQIFPSFVNYK', 'VQIFPSFVNYQ', 'VKLFPSFVNYK', 'VKLFPSFVNYQ', 'VKIFPSFVNYK', 'VKIFPSFVNYQ', 'VQL
FPAYVNQY', 'VQLFPAYVNKY', 'VQIFPAYVNQY', 'VQIFPAYVNKY', 'VKLFPAYVNQY', 'VKLFPAYVNKY', 'VKIFPAYVNQY', 'VKIFPAYVNKY', 'VQLFPSFVNQY', 'VQLFPSFVNKY',
'VQIFPSFVNQY', 'VQIFPSFVNKY', 'VKLFPSFVNQY', 'VKLFPSFVNKY', 'VKIFPSFVNQY', 'VKIFPSFVNKY']
24 Candidate residues with 0.441860465116 matches
VKLFPWFNQY False
```

# Some sanity checks

```
x = set(TheoreticalSpectrum('VKLFPAYVNQY'))
y = set(TheoreticalSpectrum('VKLFPWFNQY'))
z = set(experimentalSpectrum)
print "%s Matches = %d, Union = %d" % ('VKLFPAYVNQY', len(x & z), len(x | z))
print "%s Matches = %d, Union = %d" % ('VKLFPWFNQY', len(y & z), len(y | z))
print
print TheoreticalSpectrum('AYV')
print
print experimentalSpectrum
print
print TheoreticalSpectrum(TyrocidineB1)
```

```
VKLFPAYVNQY Matches = 38, Union = 86
VKLFPWFNQY Matches = 35, Union = 80

[71, 99, 163, 234, 262, 333]

[99, 128, 160, 163, 227, 241, 242, 244, 283, 333, 356, 405, 430, 544, 572, 584, 804, 818, 835, 879, 917, 982, 1060, 1095, 1322, 61, 64, 78, 81, 8
2, 113, 114, 142, 143, 147, 164, 178, 188, 216, 228, 234, 251, 260, 277, 291, 302, 331, 340, 345, 485, 487, 523, 552, 577, 591, 655, 675, 676, 69
0, 719, 738, 819, 854, 932]

[97, 99, 113, 114, 128, 147, 163, 186, 227, 241, 242, 244, 260, 261, 283, 291, 333, 340, 357, 388, 389, 405, 430, 447, 485, 487, 543, 544, 552, 5
75, 577, 584, 671, 672, 690, 691, 738, 770, 804, 818, 819, 835, 917, 932, 982, 1031, 1060, 1095, 1159, 1223, 1322]
```

# A More *Realistic* Example

For long sequences the underlying exponential growth becomes more evident.

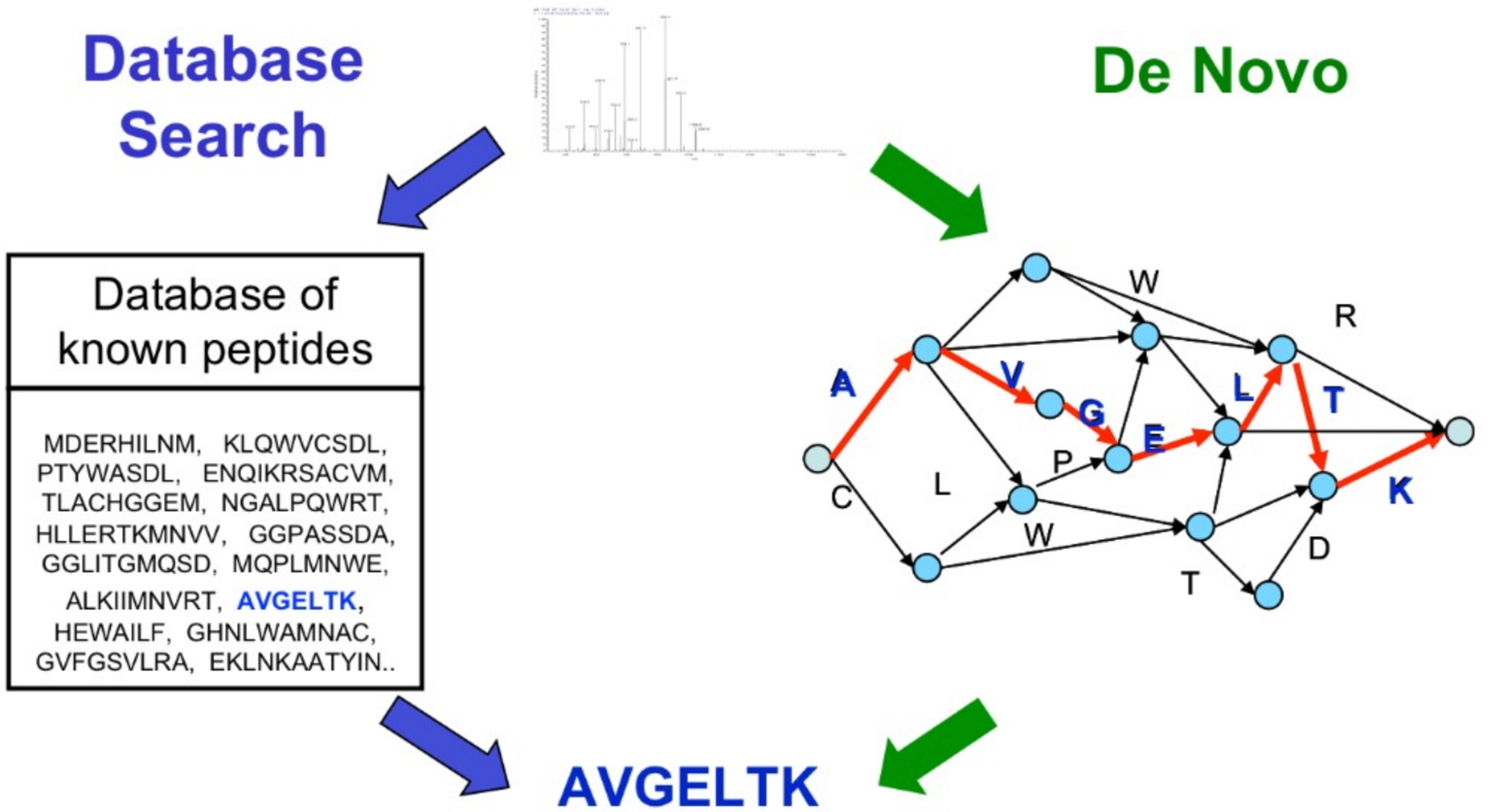
```
Insulin = "MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQGSLQPLALEGSLQKRGIVEQCCTSICSLYQLENYCN"
spectrum = TheoreticalSpectrum(Insulin)
print len(spectrum)
missingMass = random.sample(spectrum[:-1], 100)
experimentalSpectrum = sorted([mass for mass in spectrum if mass not in missingMass])
print len(experimentalSpectrum)
# See slide for the following *hack*
del Daltons['I']
del Daltons['K']
%time winners = LeaderboardFindPeptide(experimentalSpectrum, cutThreshold=0.01)
print winners
print len(winners) - 1, "Candidate residues with", winners[0], 'matches'
print Insulin, Insulin in winners
# See slide for the following *hack*
Daltons['I'] = Daltons['L']
Daltons['K'] = Daltons['Q']
```

Some of the reasons that things blow up:

1. The search space got large fast
2. There must be a *LOT* of ties
3. Algorithm tends to keep all (N-k+1) subpeptides as k approaches the sequence's size (k is related to our *round*)
4. The I/L and K/Q ambiguities lead to exponential number of ties, hence the *hack*
5. Reversed sequences are doubling our leaderboard size

There are bandaids to fix problems 3 and 4, but the problem remains

# How are peptide sequences identified?



# Peptide Identification Problem

**Goal:** Find a peptide from the database with maximal match between an experimental and theoretical spectrum.

**Input:**

- S: experimental spectrum
- database of peptides
- Δ: set of possible ion types
- m: parent mass

**Output:**

- A peptide of mass m from the database whose theoretical spectrum matches the experimental S spectrum the best

# Mass Spec Database Searches

How do you get a database?

1. Compute theoretical spectrums for all peptides from length $N$ to $M$
2. More commonly, store theoretical spectrums for known peptide sequences

- Database searches are very effective in identfying *known* or *closely related* proteins.
- Experimental spectrums are compared with spectra of database peptides to find the best fit (ex. SEQUEST, Yates et al., 1995)
- But reliable algorithms for identification of new proteins is a more difficult problem.

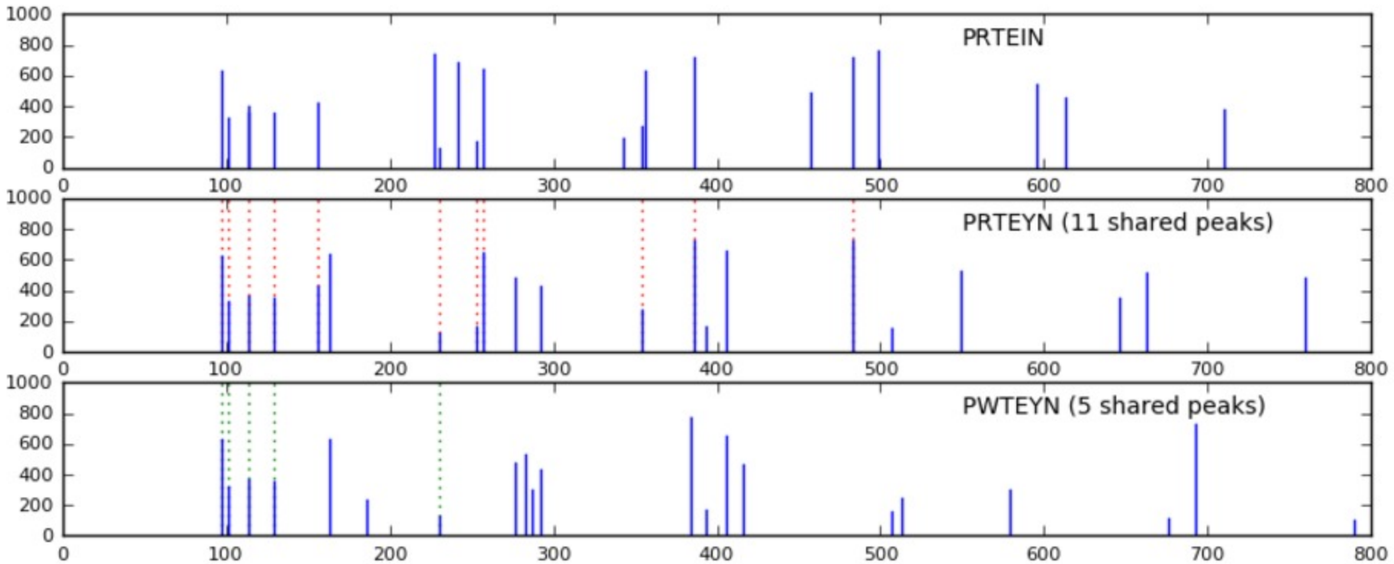
**Essence of the Database Search**

- We need a notion of *spectral similarity* that correlates well with the sequence similarity.
- If peptides are a few mutations/modifications apart, the spectral similarity between their spectra should be high.
- **Simplest measure:** *Shared Peak Counts (SPC)*
  - Very similar to the scoring function used in our *De novo* approach.

# SPC Diminishes Quickly

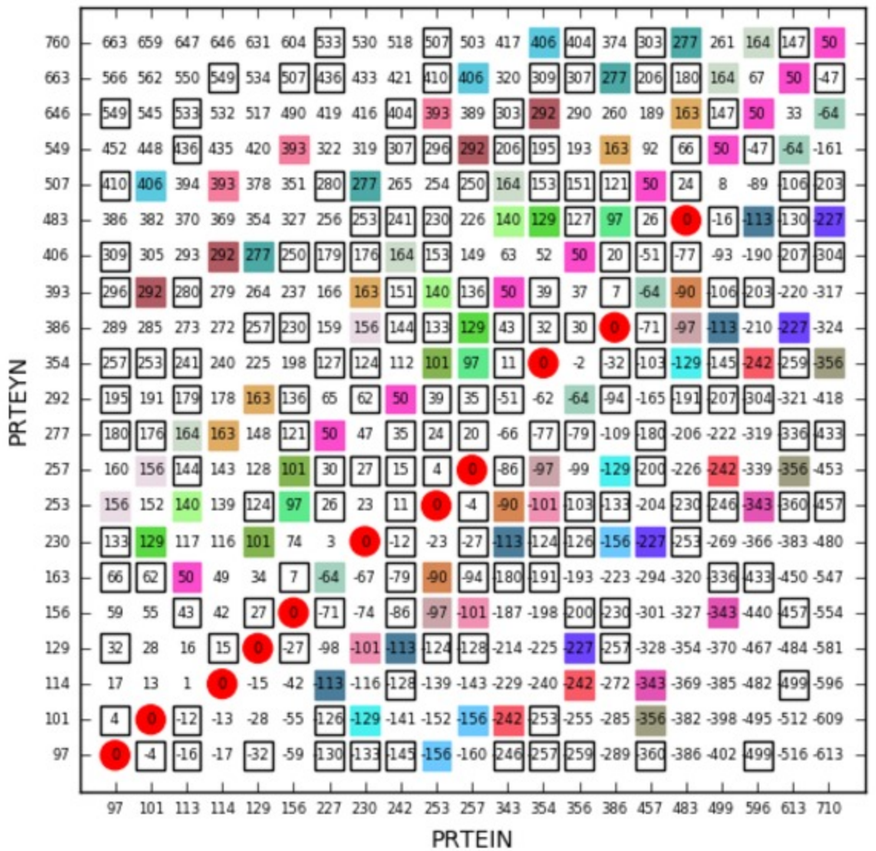
```
print TheoreticalSpectrum('PRTEIN')
print TheoreticalSpectrum('PRTEYN')
print TheoreticalSpectrum('PWTEYN')

print set(TheoreticalSpectrum('PRTEIN')) & set(TheoreticalSpectrum('PRTEYN'))
print set(TheoreticalSpectrum('PRTEIN')) & set(TheoreticalSpectrum('PWTEYN'))
```

```
[97, 101, 113, 114, 129, 156, 227, 230, 242, 253, 257, 343, 354, 356, 386, 457, 483, 499, 596, 613, 710]
[97, 101, 114, 129, 156, 163, 230, 253, 257, 277, 292, 354, 386, 393, 406, 483, 507, 549, 646, 663, 760]
[97, 101, 114, 129, 163, 186, 230, 277, 283, 287, 292, 384, 393, 406, 416, 507, 513, 579, 676, 693, 790]
set([97, 354, 483, 101, 230, 257, 129, 386, 114, 156, 253])
set([97, 114, 101, 230, 129])
```



# Spectral Convolution to the Rescue!

Difference matrix of spectrums. The elements with multiplicity > 2 are shown in colored boxes. The black outlined boxes enclose elements with multiplicity = 2. The SPC takes into account consideres only the zero entries in red circles.



# Spectral Convolution to the Rescue!

Difference matrix of spectrums. The elements with multiplicity > 2 are shown in colored boxes. The black outlined boxes enclose elements with multiplicity = 2. The SPC takes into account consideres only the zero entries in red circles.