Supplemental Information

Automatic Analysis of qNMR Data of Pharmaceutical Compound Libraries

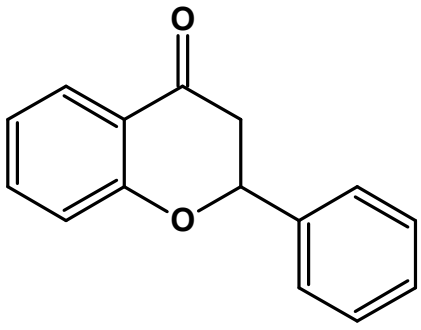
Xuejun Liu[a]; Michael X. Kolpak[b]; Jiejun Wu[a] and Gregory C. Leo[b]*

Janssen Research & Development,

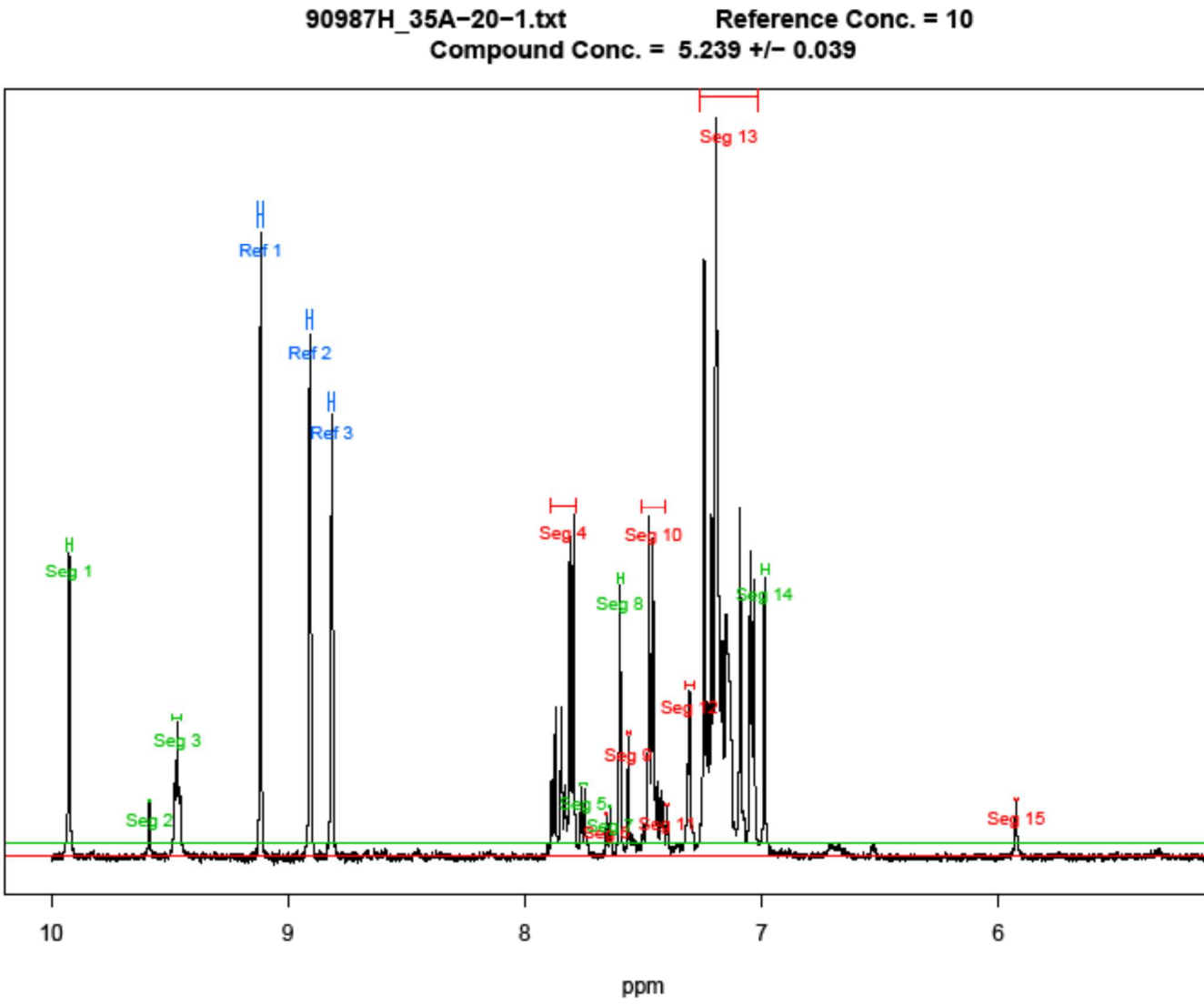
[a]3210 Merryfield Row, San Diego, CA, 92121-1126 USA

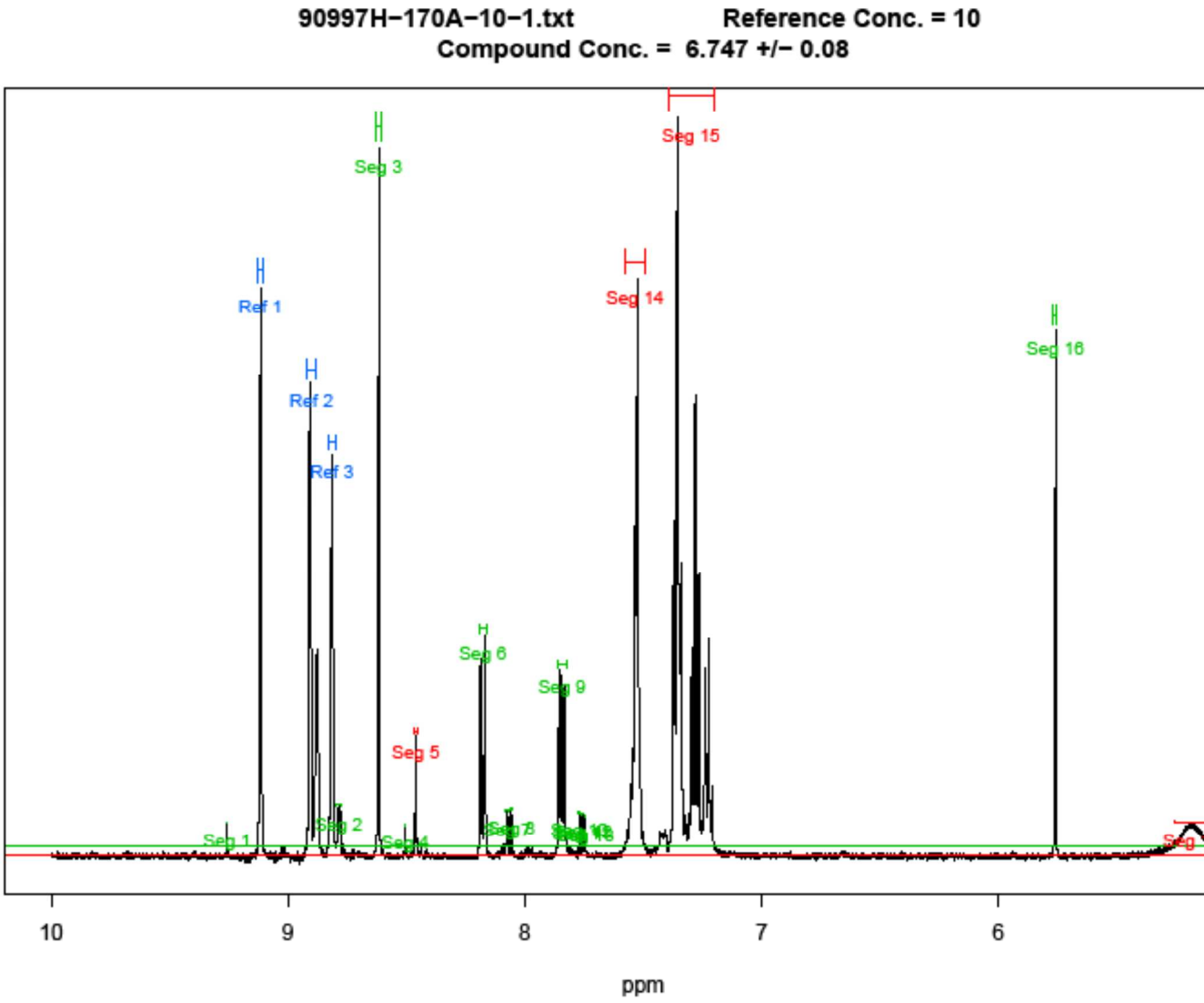
[b]Welsh and McKean Roads, Spring House, PA 19477-0776 USA

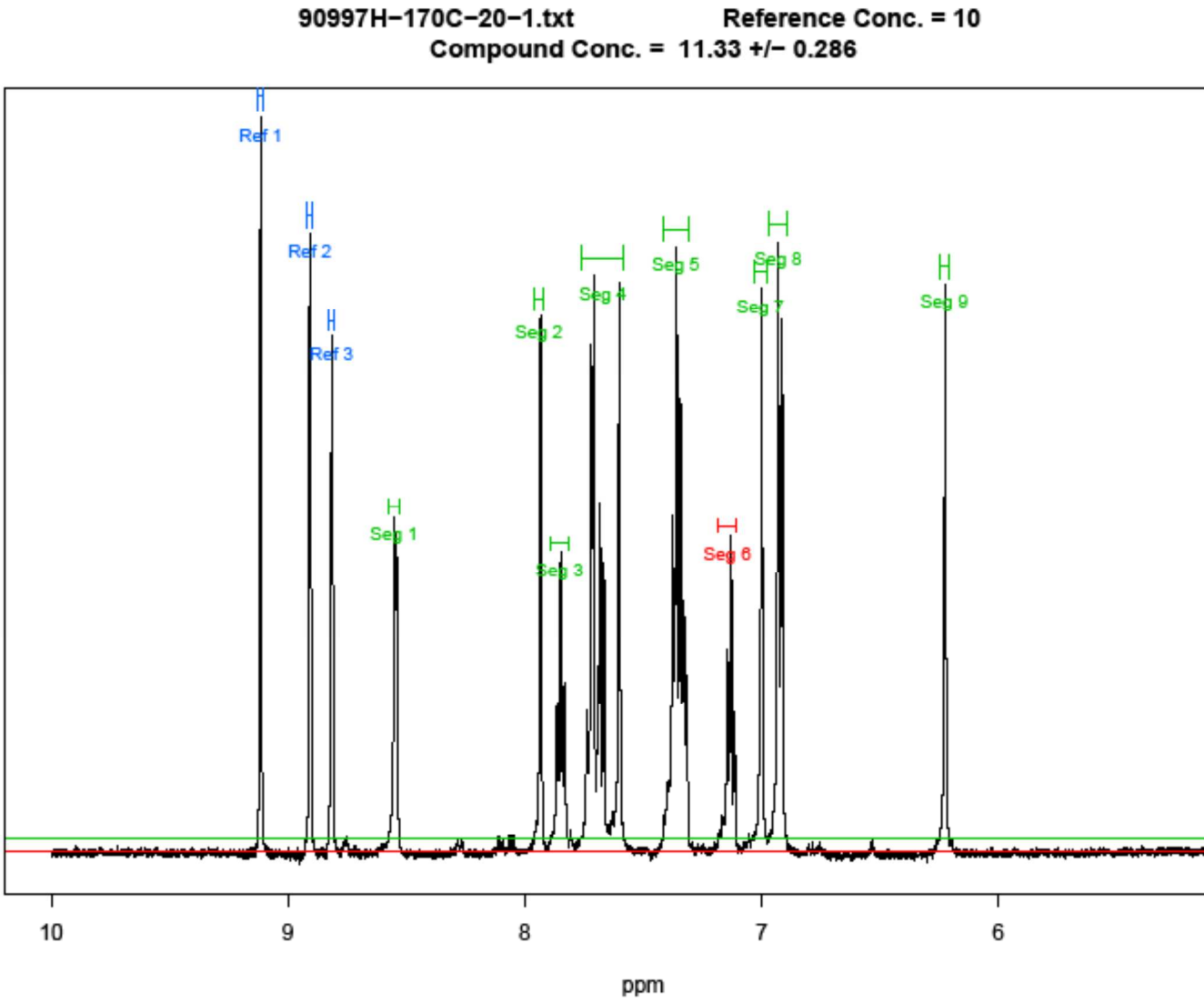
The structure of flavanone or 4H-1-Benzopyran-4-one, 2,3-dihydro-2-phenyl- that was used in the manuscript:



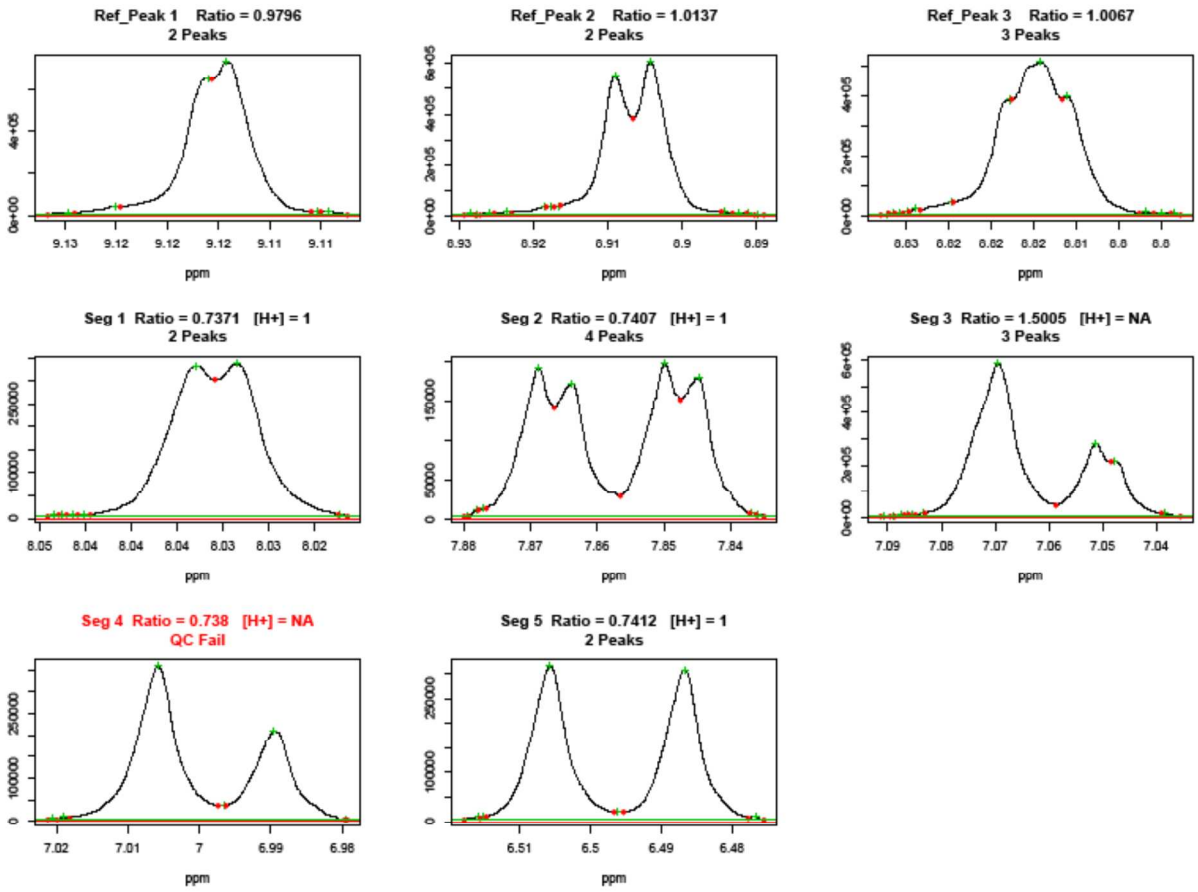
The three spectra below are the output of the automatic analysis for the yellow points in Figure 1.



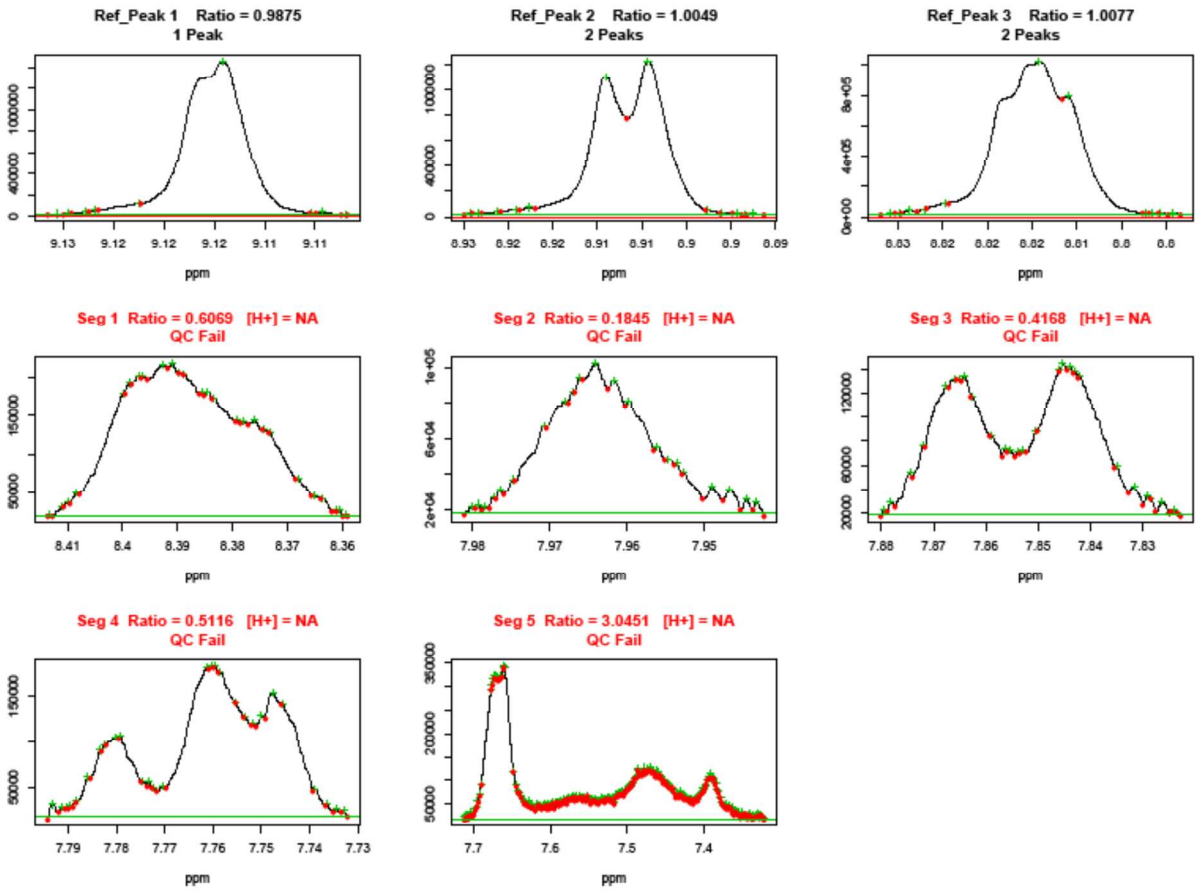

90997H-170A-10-1.txt
Reference Conc. = 10
Compound Conc. = 6.747 +/- 0.08
Seg 15
Seg 3
Ref 1
Seg 14
Seg 16
Ref 2
Ref 3
Seg 6
Seg 9
Seg 5
Seg 1
Seg 2
10
9
8
7
6
ppm


90997H−170C−20−1.txt
Reference Conc. = 10
Compound Conc. = 11.33 +/− 0.286
Ref 1
Ref 2
Ref 3
Seg 1
Seg 2
Seg 3
Seg 4
Seg 5
Seg 6
Seg 7
Seg 8
Seg 9
10
9
8
7
6
ppm

The set of eight panels below are the expansions generated by the R script for the plot shown in Figure 2.



The set of eight panels below are the expansions generated by the R script for the plot shown in Figure 3.



The following is the script described in the paper. It requires that R be installed on your computer.

```
##############################################################################
##############################################################################
########################
# R code for calculation of compound concentration by analyzing qNMR data
#
# written by Xuejun Liu
# Last Update Date: Mon Apr 09 14:54:15 2012

rm( list=ls() )
##############################################################################
##############################################################################
########################
# User Defined Inputs.
# data directory; including comma separated raw data files (in .txt format)
data.dirs = read.table("C:/qNMR/Software/qNMR.Data.Dir.txt", as.is=T)[ ,1]

# Global parameters
# Concentration of reference compound (mM)
ref.conc = 10
# minimal ppm to be considered
min.ppm = 5
# Maximum of allowed ratio between the highest reference peak and the lowest
reference peak
max.ref.ratio = 1.05

# End of user input
##############################################################################
##############################################################################
########################
# functions

f.qNMR.ref.peak = function(dat)
{
# dat: qNMR data, 3 columns (Hz, ppm, Int)
# ppm of 3 ref peaks: 9.12, 8.91, 8.82
      rp.ppm = c(9.12, 8.91, 8.82)
      x = dat$Int
      z = summary(x)
      n = length(x)
      ppm = dat$ppm

#     sn = (1:n)[ x > z[3] ]

      sn = (1:n)[ x > (z[5]+(z[5]-z[2])*1.5) ]


      ss.start = pmax(1, c( sn[1], sn[-1][diff(sn)>1] ) - 1 )
            # modified on 2012-04-05
      ss.end = pmin(n, sn[ c( (2:length(sn))[diff(sn)>1]-1, length(sn) ) ] +
1 )         # modified on 2012-04-05
```

```
      s.start = rep(1, 3)
      s.end = rep(n, 3)
      for(i in 1:3)     {
            sn2 = (1:n)[(ppm>=(rp.ppm[i]-0.03)) & (ppm<=(rp.ppm[i]+0.03)) ]

            sn3 = cbind.data.frame(ss.start, ss.end)[
(ppm[ss.start]>rp.ppm[i]) & (ppm[ss.end]<rp.ppm[i]), ]
            s.start[i] = max( sn3[1,1], min(sn2) )
            s.end[i] = min( sn3[1,2], max(sn2) )
      }
      segment = cbind(start=s.start, end=s.end, s.Hz=dat$Hz[s.start],
e.Hz=dat$Hz[s.end], s.ppm=ppm[s.start], e.ppm=ppm[s.end], peak.Hz=NA,
peak.ppm=NA, peak.Int=NA, sum.Int=NA )

      for(i in 1:nrow(segment) ) {
            s.dat = dat[ segment[i, "start"] : segment[i, "end"], ]
            s.dat = s.dat[order(-s.dat$Int), ]
            segment[i, "peak.Int"] = s.dat$Int[1]
            segment[i, "peak.Hz"] = s.dat$Hz[1]
            segment[i, "peak.ppm"] = s.dat$ppm[1]
            segment[i, "sum.Int"] = sum(as.numeric(s.dat$Int))
      }
      segment = segment[segment[, "peak.Int"] > (z[5]+(z[5]-z[2])*1.5)*3, ]
      segment = cbind(segment, seg.ratio=round(segment[,
"sum.Int"]/min(segment[, "sum.Int"]), d=4) )
      segment
}


f.qNMR.segment.QC = function(seg.dat, bgm, bgh, title="", plot.F=TRUE )
{
# seq.dat: qNMR data, 3 columns (Hz, ppm, Int)
# bgm: median of background intensity; bgh: high background intensity
      x = seg.dat$Int
      n = length(x)
      ppm = seg.dat$ppm

      sm.df = round(15/median(diff(-ppm))/10^4 )
      x.sm = smooth.spline(1:n, x, df=sm.df)

# find troughs and peaks
      x[x<bgh] = bgh
      xd = diff(x)
      tn = (2:(n-1))[ ((xd[1:(n-2)] <= 0) & (xd[2:(n-1)] > 0)) | ((xd[1:(n-
2)] < 0) & (xd[2:(n-1)] >= 0)) ]                # trough
      pn = (2:(n-1))[ ((xd[1:(n-2)] > 0) & (xd[2:(n-1)] <= 0)) | ((xd[1:(n-
2)] >= 0) & (xd[2:(n-1)] < 0)) ]                # peak

# peak recognition and area calculation
      peak = matrix(NA, length(pn), 5)
      colnames(peak) = c("start.tn", "end.tn", "pn", "height", "area.pct" )
      for(i in 1:length(pn) ) {
            peak[i, 1] = max(c(1, tn[tn<pn[i]]))
            peak[i, 2] = min(c(n, tn[tn>pn[i]]))
#           peak[i, 3] = pn[i]
            peak[i, 4] = x[ pn[i] ]
            peak[i, 5] = sum(seg.dat$Int[ peak[i,1]:peak[i,2] ])
```

```
      }
      peak = unique(as.data.frame(peak))
#     peak[,5] = peak[,5] / sum(peak[,5]) * 100
      peak[,5] = peak[,5] / sum(seg.dat$Int) * 100
      for(i in 1:nrow(peak)) {      peak[i, 3] =
round(median(pn[(pn>peak[i,1]) & (pn<peak[i,2])]))    }
      peak = as.matrix(peak[order(-peak[,5], peak[,1]), ])

# Segment QC
      QC = "QC Fail"
      if( sum(peak[,5]>5) < 11 ) {
            if( sum(peak[,5] [ peak[,5]>5 ]) > 90 ) {
                  if( sum(peak[,5]>5) == 1 )    {
                        QC="1 Peak"
                  } else if ( sum(peak[,5]>5) == 2 ) {
                        if(peak[2,4] / peak[1,4] >= 0.70)   QC="2 Peaks"
                  } else if ( sum(peak[,5]>5) == 3 ) {
                        if(peak[3,4] / peak[1,4] >= 0.35)   QC="3 Peaks"
                  } else if ( sum(peak[,5]>5) == 4 ) {
                        if(peak[4,4] / peak[1,4] >= 0.175)  QC="4 Peaks"
                  } else {
                        QC="multiple Peaks"
                  }
            }
      }

  if(plot.F) {
      plot(-ppm, seg.dat$Int, type="l", xlab="ppm", ylab="", axes=F,
main=paste(title, QC, sep="\n"), col.main= (QC=="QC Fail")+1 )
#     lines(-ppm, x.sm$y, col=4 )
      points(-ppm[ unique(peak[,3]) ], seg.dat$Int[ unique(peak[,3]) ],
pch=3, cex=0.8, col=3)
      points(-ppm[ unique(c(peak[,1:2])) ], seg.dat$Int[
unique(c(peak[,1:2])) ], pch=16, cex=0.8, col=2)
      abline(h=c(bgm, bgh), col=c(2,3) )
      axis(1, axTicks(1), round(-axTicks(1), d=2) )
      axis(2)
      box()
  }
#     peak
      QC
}

f.cpd.conc = function( dat.seg )
{
# calculate compound concentration
# dat.seg: segment data (output from function "f.qNMR.segment")
      rst = rep(NA, 2)
      y = c()
      flag = ""
      if( (sum(dat.seg[[1]][, "QC"] != "QC Fail")>0) & (sum(dat.seg[[2]][,
"QC"] != "QC Fail")>0) ) {
         xx = dat.seg[[2]][, "seg.ratio"] [ dat.seg[[2]][, "QC"] != "QC Fail"
]
         xz = min(xx)
         flag = "Manual Check"
```

```
#        x = dat.seg[[2]][, "seg.ratio"] [ (dat.seg[[2]][, "QC"] != "QC
Fail") & (dat.seg[[2]][, "QC"] != "multiple Peaks") & ( (dat.seg[[2]][,
"s.ppm"]-dat.seg[[2]][, "e.ppm"]) < 0.1 ) ]
#        x = dat.seg[[2]][, "seg.ratio"] [ (dat.seg[[2]][, "QC"] != "QC
Fail") & (dat.seg[[2]][, "QC"] != "multiple Peaks") ]

         x = dat.seg[[2]] [ (dat.seg[[2]][, "QC"] != "QC Fail") &
(dat.seg[[2]][, "QC"] != "multiple Peaks"), ]
         if(nrow(x) > 0)      {
            x = x[, "seg.ratio"] [ (x[,"QC"] == "1 Peak") | ( (x[,"s.ppm"]-
x[,"e.ppm"]) < 0.1 ) ]
            x = x[x >= 0.1]               # added on 2012-04-05, to remove
impurity peaks

            if(length(x) > 0) {

# method 1
#                 x = sort(x)
#                 while(length(x) > 1) {
#                       if( x[length(x)] / x[1] > 2.5) {
#                             x = x[-1]
#                       } else {
#                             break
#                       }
#                 }

# method 2
                  if(length(x) > 2) {
                        xz = log(x)
#                       x = x[ ( xz>=(median(xz)-mad(xz)*3) ) & (
xz<=(median(xz)+mad(xz)*3) ) ]
                        x = x[ ( xz>(median(xz)+log(0.35)) ) & (
xz<(median(xz)+log(1.3)) ) ]
                  }

                  z = rep(100, length(x))
                  z2 = z
                  for(i in 1:length(x)) {
                        y = xx/x[i]
                        z[i] = sum( (y-round(y))^2 )
                        z2[i] = sum( (y*2-round(y*2))^2 )
                  }
#                 z = (1:length(z)) [order(z)][1]
#                 xz = x[z]                              # ratio of base segment
to average of reference peaks
                  xz = x[order(z)][1]                    # ratio of base segment
to average of reference peaks
                  if(xz>1.2)  xz = c(x, x/2)[order(c(z, z2))][1]
      # added on 2012-04-05, to set the base segment with 2 protons

                  flag = ""
            }
         }

         y = round(xx/xz)
         y [xx/xz < 0.8] = 0                            # exclude small
segments in calculation of concentration
```

```
         x = xx[y>0] / y[y>0]

         if(length(x) > 2)    {
            y [ (1:length(y))[y>0] [ ( x<(median(x)-mad(x)*3) ) | (
x>(median(x)+mad(x)*3) ) ] ] = NA
            x = x[ ( x>=(median(x)-mad(x)*3) ) & ( x<=(median(x)+mad(x)*3) )
]
         }
         rst = round( c(cm = mean(x), cse = sd(x) / sqrt(length(x)) ) *
ref.conc, d=3)
      }
      list(rst, y, flag)
}

f.btw.segment = function(segment, int, min.int)
{
      n = nrow(segment)
      s1 = segment[-n, ]
      s2 = segment[-1, ]
      bs = data.frame(e1 = s1$end, s2 = s2$start, ppm.dif = s1$e.ppm -
s2$s.ppm, start=s1$start, end=s2$end, row1=1:(n-1), row2=2:n)
      int.sel = F
      for(i in 1:nrow(bs))    int.sel[i] = min(int[bs[i,1]:bs[i,2]]) >
min.int
      bs = bs[bs$ppm.dif < 0.01 & int.sel, ]

      bs
}

f.segment.merger = function(segment, dat, min.int)
{
      new.seg = segment
      n = nrow(new.seg)
      bs = f.btw.segment(new.seg, dat$Int, min.int)
      while( (n > 1) & (nrow(bs)>0) ) {
            bs = bs[1, ]
            s.start = bs[1, "start"]
            s.end = bs[1, "end"]
            ns1 = cbind.data.frame(start=s.start, end=s.end,
s.Hz=dat$Hz[s.start], e.Hz=dat$Hz[s.end], s.ppm=dat$ppm[s.start],
e.ppm=dat$ppm[s.end], peak.Hz=NA, peak.ppm=NA, peak.Int=NA, sum.Int=NA )
            s.dat = dat[ s.start : s.end, ]
            s.dat = s.dat[order(-s.dat$Int), ]
            ns1[1, "peak.Int"] = s.dat$Int[1]
            ns1[1, "peak.Hz"] = s.dat$Hz[1]
            ns1[1, "peak.ppm"] = s.dat$ppm[1]
            ns1[1, "sum.Int"] = sum(as.numeric(s.dat$Int))

            new.seg = rbind(new.seg, ns1)[-c(bs[1,"row1"], bs[1,"row2"]), ]
            n = nrow(new.seg)
            bs = f.btw.segment(new.seg, dat$Int, min.int)
      }

      new.seg[order(new.seg[,1]), ]
}

f.qNMR.segment = function(dat, title="" )
```

```
{
# dat: qNMR data, 3 columns (Hz, ppm, Int)
      x = dat$Int
      z = summary(x)
      n = length(x)
      ppm = dat$ppm
      QC.flag = "Pass"

# reference peaks
      rp = data.frame(f.qNMR.ref.peak(dat), QC=I("QC Fail") )
      if(nrow(rp)<1)    {
            QC.flag = "No reference peak"
      } else {
            for(i in 1:nrow(rp) ) {
                  rp[i, "QC"] = f.qNMR.segment.QC(dat[ rp[i, "start"] : rp[i,
"end"], ], bgm=z[3], bgh=z[5]+(z[5]-z[2])*1.5, plot.F=F  )
            }
      }
      if( sum(rp[, "QC"]!="QC Fail") > 0 ) {
            rp.sm = mean(rp[ (rp[, "seg.ratio"]<=max.ref.ratio) & (rp[,
"QC"]!="QC Fail"), "sum.Int"])
            rp[, "seg.ratio"] = round( rp[, "sum.Int"] / rp.sm, d=4)
      } else {
            QC.flag = "No reference peak passed QC"
      }

# segment detection
#     sn = (1:n)[ x > z[3] ]

      sn = (1:n)[ x > (z[5]+(z[5]-z[2])*1.5) ]


      sn = setdiff( sn, min(rp[,1]):max(rp[,2]) )                        #
remove segment covering reference peaks
      s.start = pmax(1, c( sn[1], sn[-1][diff(sn)>1] ) - 1 )
            # modified on 2012-04-05
      s.end = pmin(n, sn[ c( (2:length(sn))[diff(sn)>1]-1, length(sn) ) ] + 1
)           # modified on 2012-04-05
      segment = cbind.data.frame(start=s.start, end=s.end,
s.Hz=dat$Hz[s.start], e.Hz=dat$Hz[s.end], s.ppm=ppm[s.start],
e.ppm=ppm[s.end], peak.Hz=NA, peak.ppm=NA, peak.Int=NA, sum.Int=NA )
      for(i in 1:nrow(segment) ) {
            s.dat = dat[ segment[i, "start"] : segment[i, "end"], ]
            s.dat = s.dat[order(-s.dat$Int), ]
            segment[i, "peak.Int"] = s.dat$Int[1]
            segment[i, "peak.Hz"] = s.dat$Hz[1]
            segment[i, "peak.ppm"] = s.dat$ppm[1]
            segment[i, "sum.Int"] = sum(as.numeric(s.dat$Int))
      }

      segment = segment[!is.na(segment[,3]) & !is.na(segment[,4]), ]
      segment = segment[pmax( x[segment[, 1]], x[segment[, 2]] ) <=
(z[5]+(z[5]-z[2])*1.5), ]

      if( sum(segment[, "peak.Int"] > (z[5]+(z[5]-z[2])*1.5)*3) < 1 )   {
            QC.flag = "No compound peak"
      } else {
```

```
            segment = segment[segment[, "peak.Int"] > (z[5]+(z[5]-
z[2])*1.5)*3, ]
            if(nrow(segment)>1)     segment = f.segment.merger(segment, dat,
min.int=max((z[5]+(z[5]-z[2])*1.5)/2, z[3]*15))       # added on 2012-04-09,
merger adjacent segments

            segment = cbind(segment, seg.ratio=round(segment[,
"sum.Int"]/rp.sm, d=4), QC=I("QC Fail") )
            for(i in 1:nrow(segment) ) {
                  segment[i, "QC"] = f.qNMR.segment.QC(dat[ segment[i,
"start"] : segment[i, "end"], ], bgm=z[3], bgh=z[5]+(z[5]-z[2])*1.5, plot.F=F
)
            }
      }

      cpd.flag = ""
      if(QC.flag == "Pass")   {
            cpd.conc = f.cpd.conc ( list("Ref_Peak"=rp, "Segment"=segment) )
            proton.cnt = rep(NA, nrow(segment) )
            proton.cnt[segment[, "QC"] != "QC Fail"] = cpd.conc[[2]]
            cpd.flag = cpd.conc[[3]]
            cpd.conc = cpd.conc[[1]]
            plot(-ppm, x, type="l", xlab="ppm", ylab="", axes=F,
main=paste(title, "                    Reference Conc. =", ref.conc,
"\nCompound Conc. = ", cpd.conc[1], "+/-", cpd.conc[2] ) )
            axis(1, axTicks(1), round(-axTicks(1), d=1) )
            abline(h=c(z[3], z[5]+(z[5]-z[2])*1.5), col=c(2,3))
            box()
            for(i in 1:nrow(rp) ) {
                  lines( -rep(rp[i, "s.ppm"], 2), rp[i, "peak.Int"]*c(1.01,
1.05), col=(rp[i, "QC"]!="QC Fail")*2 + 2 )
                  lines( -rep(rp[i, "e.ppm"], 2), rp[i, "peak.Int"]*c(1.01,
1.05), col=(rp[i, "QC"]!="QC Fail")*2 + 2 )
                  lines( -c(rp[i, "s.ppm"], rp[i, "e.ppm"]), rp[i,
"peak.Int"]*c(1.03, 1.03), col=(rp[i, "QC"]!="QC Fail")*2 + 2 )
                  text( -(rp[i, "s.ppm"]+rp[i, "e.ppm"])/2, rp[i,
"peak.Int"]*1, paste("Ref", i), cex=0.8, pos=1, col=(rp[i, "QC"]!="QC
Fail")*2 + 2 )
            }
            for(i in 1:nrow(segment) ) {
                  lines( -rep(segment[i, "s.ppm"], 2), segment[i,
"peak.Int"]*c(1.01, 1.05), col=(segment[i, "QC"]!="QC Fail") + 2 )
                  lines( -rep(segment[i, "e.ppm"], 2), segment[i,
"peak.Int"]*c(1.01, 1.05), col=(segment[i, "QC"]!="QC Fail") + 2 )
                  lines( -c(segment[i, "s.ppm"], segment[i, "e.ppm"]),
segment[i, "peak.Int"]*c(1.03, 1.03), col=(segment[i, "QC"]!="QC Fail") + 2 )

                  text( -(segment[i, "s.ppm"]+segment[i, "e.ppm"])/2,
segment[i, "peak.Int"]*1, paste("Seg", i), cex=0.8, pos=1, col=(segment[i,
"QC"]!="QC Fail") + 2 )
            }
            par(mfrow=c(3,3) )
            for(i in 1:nrow(rp) ) {
                  f.qNMR.segment.QC(dat[ rp[i, "start"] : rp[i, "end"], ],
bgm=z[3], bgh=z[5]+(z[5]-z[2])*1.5, title=paste("Ref_Peak", i, "   Ratio =",
rp[i, "seg.ratio"]) )
            }
```

```
            for(i in 1:nrow(segment) ) {
                  f.qNMR.segment.QC(dat[ segment[i, "start"] : segment[i,
"end"], ], bgm=z[3], bgh=z[5]+(z[5]-z[2])*1.5, title=paste("Seg", i, " Ratio
=", segment[i, "seg.ratio"], "  [H+] =", proton.cnt[i]) )
            }
      } else {
            cpd.conc = rep(NA, 2)
            plot(-ppm, x, type="l", xlab="ppm", ylab="", axes=F,
main=paste(title, "\n", QC.flag) )
            axis(1, axTicks(1), round(-axTicks(1), d=1) )
            abline(h=c(z[3], z[5]+(z[5]-z[2])*1.5), col=c(2,3))
            box()
      }

      list("Ref_Peak"=rp, "Segment"=segment, "Cpd_Conc"=cpd.conc,
"Cpd.flag"=cpd.flag)
}

#     dat.seg = f.qNMR.segment(dat, title=fn )

# end of functions
##############################################################################
##############################################################################
########################
# Data processing
setwd("C:")
dir.create("C:/qNMR")
setwd("C:/qNMR")
dir.create("C:/qNMR/Process.Log")

#     pdf(file="C:/Data.analysis/JieJun.Wu/qNMR/Test_Segment7.pdf", width=11,
height=8)

for(data.dir in data.dirs) {
      cat("Processing data in", data.dir, "\n")
      setwd(data.dir)
#     dir.create("Process.Log")
      fns = list.files(pattern="txt")
      cpd.conc = matrix(NA, length(fns), 2)
      dimnames(cpd.conc) = list( fns, c("Conc.Avg", "Conc.SE") )
      flag=rep("", length(fns))
      names(flag) = fns

      for(fn in fns )   {
cat(fn, "\n")
            pdf(file=paste("C:/qNMR/Process.Log/", fn, ".pdf", sep=""),
width=11, height=8)
            dat = read.delim(file=fn, header=F, sep=",")
            colnames(dat) = c("Hz", "ppm", "Int")
            dat$Int = as.double(dat$Int)
            dat = dat[dat$ppm>=min.ppm, ]
            par(mfrow=c(1,1))
            dat.seg = try( f.qNMR.segment(dat, title=fn ) )
            if( class(dat.seg) == "list" )      {
                  cpd.conc[fn, ] = dat.seg[[3]]
                  flag[fn] = dat.seg[[4]]
            }
```

```
            dev.off()
      }

      dd = gsub("\\\\", "_", data.dir)
      dd = gsub("\\/", "_", dd)
      dd = gsub(":", "", dd)
      dd = gsub("__", "_", dd)
      while(substr(dd,1,1) == "_" ) {     dd = substr(dd, 2, nchar(dd)) }

      flag[ (cpd.conc[,1]>12) | (cpd.conc[,1]<3) | is.na(cpd.conc[,2]) |
(cpd.conc[,2]>0.5) ] = "Manual Check"
      write.table(data.frame(FileName=fns, cpd.conc, Flag=flag),
file=paste("C:/qNMR/Process.Log/", dd, "_qNMR.csv", sep=""),  sep=",",
row.names=F, col.names=T)
}

#     dev.off()

setwd("C:/qNMR")

##############################################################################
##############################################################################
########################
# end of the entire program
```