Calculating Speech Intelligibility Index (SII) using R

Gregory R. Warnes, Ph.D.
Gregory R. Warnes Consulting

August 31, 2026

This document describes the calculation of Speech Intelligibility Index (SII) using R. The core calculations have been encapsulated as an R add-on package named "SII", which, once installed, can be loaded thusly:

```
> library(SII)
```

1 SII constant tables

The R package includes constant tables 1 – 4 from the ANSI/ASA S3.5-1997 (R2024) text. These can be loaded via

```
> ## Table 1: Critical band SII procedure constants
> data("critical")
> head(critical)
```

```
   fi  li  hi     Ii normal raised  loud shout    Xi  Fi
1 150 100 200 0.0103  31.44  34.06 34.21 28.69   1.5 0.6
2 250 200 300 0.0261  34.75  38.98 41.55 42.50  -3.9 1.0
3 350 300 400 0.0419  34.14  38.62 43.68 47.14  -7.2 1.4
4 450 400 510 0.0577  34.58  39.84 44.08 48.46  -8.9 1.4
5 570 510 630 0.0577  33.17  39.44 45.34 50.17 -10.3 1.9
6 700 630 770 0.0577  30.64  37.99 45.22 51.68 -11.4 2.8
```

```
> ## Table 2:Equally contributing (17 band) critical band SII
> ## procedure constants
> data("equal")
> ## Table 3: One-third octave band SII procedure constants
> data("onethird")
> ## Table 4: Octave band SII procedure constants
> data("octave")
> ## Overall SPL constants
> data("overall.spl")
> overall.spl
```

```
normal raised   loud  shout
 62.35  68.34  74.85  82.30
```

It also includes constant tables for alternative transfer functions corresponding to different types of message content, tables B.1 – B.3. These tables have been augmented with the corresponding values for the Connected Speech Test (CST) as originally provided by the author's website. These tables can be loaded via:

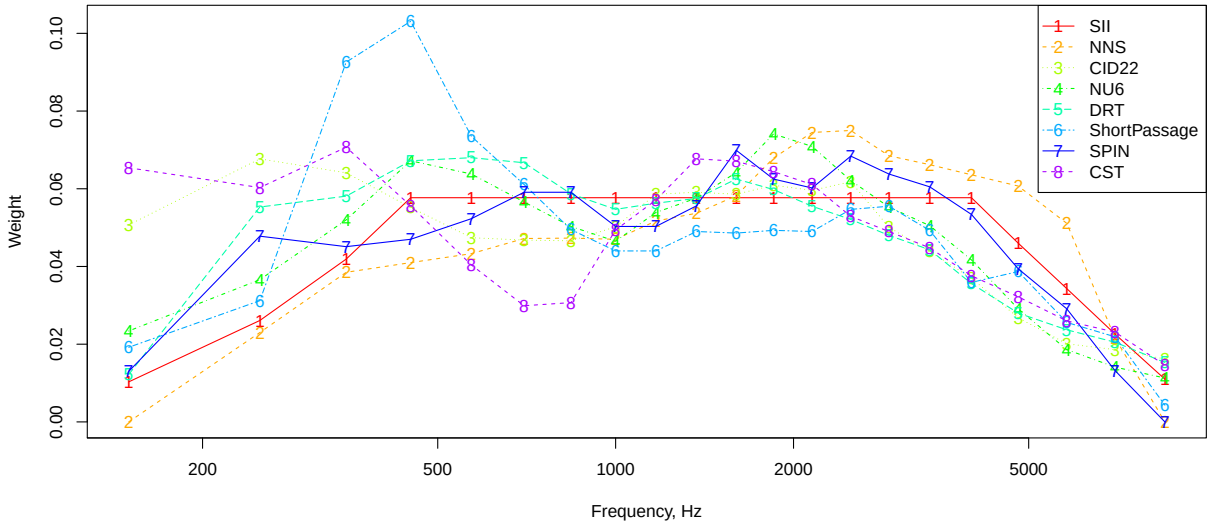
```
> ## Table B.1: Critical band importance functions for various speech tests.
> data(sic.critical)
> head(sic.critical)
```

```
   fi    SII    NNS  CID22    NU6    DRT ShortPassage   SPIN    CST
1 150 0.0103 0.0000 0.0507 0.0234 0.0122       0.0192 0.0130 0.0654
2 250 0.0261 0.0230 0.0677 0.0366 0.0553       0.0312 0.0478 0.0603
3 350 0.0419 0.0385 0.0641 0.0520 0.0581       0.0926 0.0451 0.0707
4 450 0.0577 0.0410 0.0552 0.0672 0.0672       0.1031 0.0470 0.0556
5 570 0.0577 0.0433 0.0474 0.0638 0.0680       0.0735 0.0523 0.0404
6 700 0.0577 0.0472 0.0468 0.0566 0.0667       0.0611 0.0591 0.0299
```

```
> ## Table B.2: One-third octave band importance functions for various speech tests.
> data(sic.onethird)
> ## Table B.3: Octave band importance functions for various speech tests.
> data(sic.octave)
```

With the alternative transfer functions available, it becomes easy to generate a nice graphical comparison of the functions:

```
> data(sic.critical)
> ngroup <- ncol(sic.critical)
> matplot(x=sic.critical[,1], y=sic.critical[,-1],
+         type="o",
+         xlab="Frequency, Hz",
+         ylab="Weight",
+         log="x",
+         lty=1:ngroup,
+         col=rainbow(ngroup)
+ )
> legend(
+        "topright",
+        legend=names(sic.critical)[-1],
+        pch=as.character(1:ngroup),
+        lty=1:ngroup,
+        col=rainbow(ngroup)
+        )
>
```



2 Calculating SII

The `sii` function implements ANSI/ASA S3.5-1997 (R2024) as described in the standard, without any attempt to optimize the performance. The implementation does, however, include the extension for handling conductive hearing loss from Annex A (utilizing the optional `loss` argument, and for utilizing alternative band weights (i.e. transfer function) appropriate for differing message contents (e.g. types of speech) as described in Annex B or user-specified band weights (utilizing the optional argument `importance`).

Further, this implementation provides a mechanism for interpolating/extrapolating available measurements to those required for the specified procedure (via the argument `interpolate=TRUE`). Interpolation is accomplished using linear interpolation (on log-scaled data) to the frequencies required for the specified SII procedure. Interpolation is performed (if necessary) for `speech`, `noise`, `threshold`, and `loss`.

The `sii` function has the following header:

```
> args(sii)
```

```
function (speech = c("normal", "raised", "loud", "shout"), noise,
    threshold, loss, freq, method = c("critical", "equal-contributing",
        "one-third octave", "octave"), importance = c("SII",
        "NNS", "CID22", "NU6", "DRT", "ShortPassage", "SPIN",
        "CST"), interpolate = FALSE, prescription = NULL, desensitization = FALSE,
    ldl = NULL, gender = "male", experience = "experienced",
    config = "bilateral", coupling = "custom_occluded", module = "standard",
    transducer = "inserts", custom_gain = NULL, measured_wrs = NULL,
    wrs_level = NULL, distortion_category = NULL, nal_ldf = FALSE,
    ...)
NULL
```

Where the arguments are:

speech Speech spectrum level, as a standard level name ("normal", "raised", "loud", or "shout")or a vector of values, in dB

noise Noise spectrum level, in dB

threshold Hearing threshold level, in dB

loss Conductive hearing loss level, in dB

freq Frequencies at which values are provided (required if interpolate=TRUE)

method SII calculation method, one of "critical", "equal-contributing", "one-third octave", "octave".

importance Transfer function (importance weights), as a standard SII measurement name ("SII", "NNS", "CID22", "NU6", "DRT", "ShortPassage", "SPIN", or "CST")

interpolate Flag indicating whether to interpolate from the provide measurement values and frequencies to those required by the specified method

For a detailed description of the arguments see the `sii` manual page in appendix A.

2.1 Example C.1 from ANSI/ASA S3.5-1997 (R2024) Annex C

```
> sii.C1 <- sii(
+               speech   = c(50.0, 40.0, 40.0, 30.0, 20.0,  0.0),
+               noise    = c(70.0, 65.0, 45.0, 25.0,  1.0,-15.0),
+               threshold= c( 0.0,  0.0,  0.0,  0.0,  0.0,  0.0),
+               method="octave"
+               #, importance="SII"
+               #, importance=octave$Ii
+               , importance="CST"
+               )
> round(sii.C1$table[,-c(5:7,13)],2)
```

```
    Fi E'i N'i T'i  Zi    Xi   X'i   Di    Ui   Li   Ki   Ai   Ii IiAi
1  250  50  70   0  70  -3.9  -3.9 70.0 34.75 0.97 0.00 0.00 0.09 0.00
2  500  40  65   0  65  -9.7  -9.7 65.0 34.27 1.00 0.00 0.00 0.19 0.00
3 1000  40  45   0  45 -12.5 -12.5 45.0 25.01 0.97 0.33 0.32 0.16 0.05
4 2000  30  25   0  25 -17.7 -17.7 25.0 17.32 0.98 0.67 0.66 0.30 0.20
5 4000  20   1   0   1 -25.9 -25.9  1.0  9.33 1.00 1.00 1.00 0.18 0.18
6 8000   0 -15   0 -15  -7.1  -7.1 -7.1  1.13 1.00 0.74 0.74 0.08 0.06
```

```
> sii.C1$sii
```

```
[1] 0.4873175
```

The value given in the Standard is 0.504.

2.2 Example C.2 from ANSI/ASA S3.5-1997 (R2024) Annex C

```
> sii.C2 <- sii(
+               speech   = rep(54.0, 18),
+               noise    = c(40.0, 30.0, 20.0, rep(0, 18-3) ),
+               threshold= rep(0.0,  18),
+               method="one-third"
+               )
> sii.C2$table[1:3,1:8]
```

```
   Fi E'i N'i T'i Vi Bi        Ci       Zi
1 160  54  40   0 30 40 -46.58708 40.00000
2 200  54  30   0 30 30 -52.00562 34.65765
3 250  54  20   0 30 30 -51.42416 25.04683
```

```
> sii.C2$sii
```

```
[1] 0.8513749
```

The standard shows the first three rows in the table as

	Fi	E'i	N'i	T'i	Vi	Bi	Ci	Zi
1	160	54.00	40.00	0.00	30.00	40.00	−46.59[1]	40.00
2	200	54.00	30.00	0.00	30.00	30.00	−52.01	34.66
3	250	54.00	20.00	0.00	30.00	30.00	−51.42	25.04

2.3 Interpolation Example

```
> sii.C1 <- sii(
+               speech   = c(50.0, 40.0, 40.0, 30.0, 20.0,  0.0),
+               noise    = c(70.0, 65.0, 45.0, 25.0,  1.0,-15.0),
+               threshold= c( 0.0,  0.0,  0.0,  0.0,  0.0,  0.0),
+               method="octave"
+               #, importance="SII"
+               #, importance=octave$Ii
+               , importance="CST"
+               )
> round(sii.C1$table[,-c(5:7,13)],2)
```

```
    Fi E'i N'i T'i  Zi    Xi   X'i   Di    Ui   Li   Ki   Ai   Ii IiAi
1  250  50  70   0  70  -3.9  -3.9 70.0 34.75 0.97 0.00 0.00 0.09 0.00
2  500  40  65   0  65  -9.7  -9.7 65.0 34.27 1.00 0.00 0.00 0.19 0.00
3 1000  40  45   0  45 -12.5 -12.5 45.0 25.01 0.97 0.33 0.32 0.16 0.05
4 2000  30  25   0  25 -17.7 -17.7 25.0 17.32 0.98 0.67 0.66 0.30 0.20
5 4000  20   1   0   1 -25.9 -25.9  1.0  9.33 1.00 1.00 1.00 0.18 0.18
6 8000   0 -15   0 -15  -7.1  -7.1 -7.1  1.13 1.00 0.74 0.74 0.08 0.06
```

```
> sii.C1$sii
```

```
[1] 0.4873175
```

The value given in the Standard is 0.504.

2.4 Calculating SII for a set of patients

First, we need to load the patient information table

```
> library(readxl)
> patInfo <- read_excel("../AI subject list.xls")
```

Check that we got the data properly read in:

```
> ## Information about variables
> str(patInfo)
```

```
> ## First 6 rows
> head(patInfo)
```

Create some useful variables:

```
> ## measured frequencies
> freq <- c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000)
> ## columns containing frequencies for the right/left ear
> rt.cols <- paste("PTR",freq, sep="")
> lt.cols <- paste("PTL",freq, sep="")
> rt.vals <- patInfo[,rt.cols]
> lt.vals <- patInfo[,lt.cols]
```

Handle missing value encoding

```
> rt.vals[rt.vals==-888] <- NA
> lt.vals[rt.vals==-888] <- NA
```

Now, construct a utility function to handle an individual patient's SII calculation using the arguments we want.

```
> fun <- function(X)
+   {
+     ret <- try(
+                 sii(X,
+                     speech="raised",
+                     threshold=c(15,15,20,25,35,35,45,50),
+                     freq=c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000),
+                     importance="SII",
+                     interpolate=TRUE
```

```
+                   )$sii
+                )
+   if("try-error" %in% class(ret))
+     return(NA)
+   else
+     return(ret)
+   }
> ## Test it
> fun( rt.vals[1,] )
```

Now apply it for the right and left ears:

```
> sii.right <- apply(rt.vals, 1, fun )
> sii.left  <- apply(lt.vals, 1, fun )
```

Now add theses back onto our table:

```
> patInfo$"SII.right" <- sii.right
> patInfo$"SII.left" <- sii.left
> tail(patInfo)
```

And save to a file:

```
> write.table(patInfo,
+             file="../AI subject list-SII.xls",
+             row.names=FALSE,
+             sep="\t"
+             )
```

Now define a function to do all of this with a single call

```
> sii.dina <- function(infile, outfile, verbose=TRUE)
+   {
+     if(verbose)
+       cat("\nLoading data file '", infile, "'...\n", sep="")
+     ## Load the data
+     library(readxl)
+     patInfo <- read_excel(infile)
+
+     ## measured frequencies
+     freq <- c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000)
+
+     if(verbose)
+       cat("\nExtracting hearing thresholds...\n")
+
+     ## columns containing frequencies for the right/left ear
+     rt.cols <- paste("PTR",freq, sep="")
+     lt.cols <- paste("PTL",freq, sep="")
+
+     rt.vals <- patInfo[,rt.cols]
```

```
+     lt.vals <- patInfo[,lt.cols]
+
+     ## Handle missing code '-888'
+     rt.vals[rt.vals==-888] <- NA
+     lt.vals[rt.vals==-888] <- NA
+
+     ## define function to compute SII with our defaults
+     fun <- function(X)
+       {
+         ret <- try(
+                    sii(X,
+                        speech="raised",
+                        threshold=c(15,15,20,25,35,35,45,50),
+                        freq=c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000),
+                        importance="SII",
+                        interpolate=TRUE
+                        )$sii #$
+                    )
+       if("try-error" %in% class(ret))
+         return(NA)
+       else
+         return(ret)
+       }
+
+     ## Calculate SII
+     if(verbose)
+       cat("\nCalculating right ear SII...\n")
+     sii.right <- apply(rt.vals, 1, fun )
+     if(verbose)
+       cat("\nCalculating left ear SII...\n")
+     sii.left  <- apply(lt.vals, 1, fun )
+
+     ## Add back onto the table
+     patInfo$"SII.right" <- sii.right
+     patInfo$"SII.left" <- sii.left
+
+     if(verbose)
+       cat("\nWriting new data table as '", outfile, "'...\n", sep="")
+
+     ## Save file
+     write.table(patInfo,
+                 file=outfile,
+                 row.names=FALSE,
+                 sep="\t"
+                 )
+
+     if(verbose)
+       cat("\nDone.\n\n")
+   }
```

Try it out:

```
> sii.dina(infile="../AI subject list.xls",
```

```
+          outfile="../AI subject list-SII.xls")
```

Just for kicks, compare the original AI, and our computed sii values:

```
> library(xtable)
> xt <- xtable(patInfo[,c("AI","SII.right","SII.left")],
+              caption="Comparison of original AI and new right and left SII values \\label{table}",
+              digits=2)
> print(xt)
```

```
> ## put histograms on the diagonal
> panel.hist <- function(x, ...)
+   {
+     usr <- par("usr"); on.exit(par(usr))
+     par(usr = c(usr[1:2], 0, 1.5) )
+     h <- hist(x, plot = FALSE)
+     breaks <- h$breaks; nB <- length(breaks)
+     y <- h$counts; y <- y/max(y)
+     rect(breaks[-nB], 0, breaks[-1], y, col="cyan", ...)
+   }
> pairs.2 <- function(x)
+   pairs(x,
+       panel=panel.smooth,
+       cex = 1.5,
+       pch = 24,
+       bg="light blue",
+       diag.panel=panel.hist,
+       cex.labels = 2,
+       font.labels=2
+       )
> pairs.2(patInfo[,c("AI","SII.right","SII.left")])
```

A SII Help Pages

`SII-package` *Calculate ANSI/ASA S3.5-1997 (R2024) Speech Intelligibility Index*

Description

This package calculates ANSI/ASA S3.5-1997 (R2024) Speech Intelligibility Index (SII), a standard method for computing the intelligibility of speech from acoustical measurements of speech, noise, and hearing thresholds. This package includes data frames corresponding to Tables 1 - 4 in the ANSI standard as well as a function utilizing these tables and user-provided hearing threshold and noise level measurements to compute the SII score. The methods implemented here extend the standard computations to allow calculation of SII when the measured frequencies do not match those required by the standard by applying interpolation to obtain values for the required frequencies.

Author(s)

Gregory R. Warnes `<greg@warnes.net>`

References

ANSI/ASA S3.5-1997 (R2024), "American National Standard Methods for Calculation of the Speech Intelligibility Index" American National Stan- dards Institute, New York.

Other software programs for calculating SII are available from https://sii.to/html/programs.html.

Examples

```
## Example C.1 from ANSI/ASA S3.5-1997 (R2024) Annex C
sii.C1 <- sii(
              speech   = c(50.0, 40.0, 40.0, 30.0, 20.0,  0.0),
              noise    = c(70.0, 65.0, 45.0, 25.0,  1.0,-15.0),
              threshold= c( 0.0,  0.0,  0.0,  0.0,  0.0,  0.0),
              method="octave"
       )
sii.C1                        # rounded to 2 digits by default
print(sii.C1$sii, digits=20)  # full precision
summary(sii.C1)               # full details
plot(sii.C1)                  # plot
## The value given in the Standard is $0.504$.
```

`calculate_binaural_loudness` *Calculate Psychoacoustic Binaural Loudness (Sones)*

Description

Calculates the total perceived binaural loudness based on the Pieper et al. (2021) and Moore et al. (2016) binaural summation models.

Usage

```
calculate_binaural_loudness(x_left, x_right = NULL, alpha_b = -0.25)
```

Arguments

`x_left`	An object of class 'SII' for the left ear.
`x_right`	Optional. An object of class 'SII' for the right ear. If NULL, assumes a perfectly symmetric bilateral fitting ('x_left == x_right').
`alpha_b`	Numeric. The binaural inhibition factor. Default is -0.25, reflecting classical normal-hearing binaural inhibition.

Value

A numeric value representing the total binaural loudness in Sones.

`calculate_loudness` *Calculate Loudness (Sones)*

Description

Calculates the estimated loudness in sones for a given Speech Intelligibility Index (SII) object.

Usage

```
calculate_loudness(x, ohc_proportion = 0.65)
```

Arguments

`x`	An object of class `SII` as returned by `sii`.
`ohc_proportion`	Outer hair cell loss proportion (default: 0.65).

Value

Returns a numeric value representing the total loudness in sones.

Author(s)

Mark Shaver

Examples

```
sii.res <- sii(speech=c(50, 40, 40, 30, 20, 0), threshold=rep(0,6), method="octave")
calculate_loudness(sii.res)
```

`calculate_loudness_bramslow2004` *Calculate Loudness for Cochlear Hearing Loss*

Description

Fast excitation pattern estimation based on Chen et al. 2011 (JASA) "A new model for calculating auditory excitation patterns and loudness for cases of cochlear hearing loss." This implements the Moore & Glasberg (2004) excitation model.

Usage

```
calculate_loudness_bramslow2004(
  inputF,
  inputLdB,
  HLcf = NULL,
  HLohcdB0 = NULL,
  HLihcdB0 = NULL,
  cambin = 0.1,
  flow = 50,
  fhigh = 15000,
  outerearcorrection = "FreeField"
)
```

Arguments

`inputF`	Vector of dense input frequency values (Hz, typically 1 Hz spaced)
`inputLdB`	Vector of input spectrum levels (dB/Hz)
`HLcf`	Audiogram frequencies
`HLohcdB0`	OHC loss at audiogram frequencies
`HLihcdB0`	IHC loss at audiogram frequencies
`cambin`	Spacing [ERB] between successive auditory filter CFs
`flow`	Lowest center frequency of an auditory filter
`fhigh`	Highest center frequency of an auditory filter
`outerearcorrection`	"FreeField", "PDR10", or "Eardrum"

Value

A list containing Loudness (sones), Excitation, Cams, and CFs.

`calculate_loudness_cpp`

Calculate Canonical Loudness (Native C++ Engine)

Description

Native C++ port of the bramslow2004 canonical engine, corrected per AMToolbox 1.6.0 reference with noted simplifications.

Usage

```
calculate_loudness_cpp(
  inputF,
  inputLdB,
  HLcf,
  HLohcdB0,
  HLihcdB0,
  NoChan = 30L,
  E_Beg = 3,
  E_End = 32,
  Binaural = 0L
)
```

Arguments

`inputF`	Vector of input frequencies (Hz)
`inputLdB`	Vector of input spectrum levels (dB/Hz, free field)
`HLcf`	Audiogram frequencies (Hz)
`HLohcdB0`	OHC loss at audiogram frequencies (dB)
`HLihcdB0`	IHC loss at audiogram frequencies (dB)
`NoChan`	Number of ERB channels (default 30)
`E_Beg`	Lowest ERB rate (default 3.0)
`E_End`	Highest ERB rate (default 32.0)
`Binaural`	Integer indicating whether to compute binaural loudness (default 0L)

Value

A list containing Loudness (sones), Excitation, Cams, and CFs.

`convert_1_3_octave_to_density`
Convert 1/3-octave band levels to 1 Hz spectrum density

Description

Convert 1/3-octave band levels to 1 Hz spectrum density

Usage

```
convert_1_3_octave_to_density(fc, level)
```

Arguments

`fc`	Center frequencies of 1/3 octave bands
`level`	dB SPL in each band

Value

A list containing ‘f‘ (1 Hz frequencies) and ‘l_density‘ (spectrum levels)

`critical` *Constants Tables for ANSI/ASA S3.5-1997 (R2024) Speech Intelligibility Index (SII)*

Description

Tables of constants for ANSI/ASA S3.5-1997 (R2024) Speech Intelligibility Index (SII)

Usage

```
data(critical)
data(equal)
data(onethird)
data(octave)
data(overall.spl)
```

Format

Each data frames has 6-21 observations and a subset of the following variables:

`fi` Center frequency of SII band, Hz

`li` Lower limit of frequency band, Hz

`hi` Upper limit of frequency band, Hz

`Deltai` Band width adjustment, dB

`Ii` Band importance function

`normal, raised, loud and shout` Standard spectrum levels for vocal effort levels "normal", "raised", "loud", and "shout", respectively, dB

`Xi` Spectrum level of internal noise, dB

`Fi` Band importance function (weight)

Details

These data objects provide constant tables 1 – 4 from the ANSI S3.5-1997.

`critical` Table 1: Critical band SII procedure constants

`equal` Table 2:Equally contributing (17 band) critical band SII

`onethird` Table 3: One-third octave band SII procedure constants

`octave` Table 4: Octave band SII procedure constants

`overall.spl` Overall sound pressure level (SPL) for the for vocal effort levels "normal", "raised", "loud", and "shout", in dB

Source

ANSI/ASA S3.5-1997 (R2024), "American National Standard Methods for Calculation of the Speech Intelligibility Index" American National Standards Institute, New York.

References

ANSI/ASA S3.5-1997 (R2024), "American National Standard Methods for Calculation of the Speech Intelligibility Index" American National Standards Institute, New York.

Examples

```
data(critical)
critical # show entire table

data(equal)
names(equal)
equal$fi # extract just the frequency band centers

data(onethird)
barplot(onethird$Ii) # plot band importance function (weights)

data(octave)
round(octave, digits=2) # just 2 digits

data(overall.spl)
overall.spl
```

`export_gains` *Export prescribed insertion gains for 50, 65, and 80 dB SPL input levels*

Description

Export prescribed insertion gains for 50, 65, and 80 dB SPL input levels

Usage

```
export_gains(x)
```

Arguments

`x`	An object of class 'SII'

Value

A data.frame containing frequency and the prescribed insertion gains

`get_specific_loudness` *Calculate Psychoacoustic Loudness (Sones)*

Description

Calculates the total perceived loudness in Sones based on the specific speech spectrum and hearing thresholds of the patient.

Usage

```
get_specific_loudness(x)
```

Arguments

`x`	An object of class `SII`.

Details

The current implementation uses a first-order heuristic model of recruitment based on Stevens' Power Law, mapped over an estimated dynamic range. While it is computationally fast and demonstrates the restoration of loudness conceptually, it is not a full psychoacoustic model (like Moore-Glasberg / CAM2Q), as it does not calculate basilar membrane excitation patterns via RoEx filters.

Mathematical Formula:

1. **Uncomfortable Loudness Level (UCL)** is predicted from the threshold (T_i):

$$UCL_i = 100 + 0.25 \times \max(0, T_i - 20)$$

2. **Dynamic Range (DR)**:

$$DR_i = \max(1, UCL_i - T_i)$$

3. **Sensation Level (SL)** of speech peaks (RMS + 15 dB):

$$SL_i = \max(0, E_i + 15 - T_i)$$

4. **Loudness Level (Phons)** modeling recruitment:

$$Phons_i = \left(\frac{SL_i}{DR_i}\right) \times 100$$

5. **Specific Loudness (Sones)** per band via Stevens' Power Law:

$$Sones_i = 2^{\frac{Phons_i - 40}{10}} \quad \text{for } Phons_i \geq 40$$

$$Sones_i = \left(\frac{Phons_i}{40}\right)^{2.5} \quad \text{for } Phons_i < 40$$

6. **Total Loudness (Sones)**:

$$Sones_{Total} = \sum Sones_i \times \text{calibration\_factor}$$

The calibration factor (0.25 for 21 critical bands) is scaled dynamically based on the number of frequency bands used in the underlying SII method (e.g., 21 for critical, 6 for octave) to ensure equivalent loudness summation across different resolutions.

Value

A numeric value representing the total loudness in Sones.

`open_nl` *Open-NL WDRC Gain Prescription*

Description

Calculates experimental prescriptive insertion gain and SSPL90 targets for a given hearing loss, based on a rule-based WDRC heuristic.

Usage

```
open_nl(
  speech = 65,
  threshold,
  freq,
  ...,
  gender = "male",
  experience = "experienced",
  config = "bilateral",
  coupling = "custom_occluded",
  module = "standard",
  ldl = NULL,
  loss = NULL,
  distortion_category = NULL,
  user_cr = NULL,
  optimize = TRUE,
  seed_noise = NULL,
  optim_method = "Nelder-Mead",
  abg_fraction = 0.75,
  enable_severe_booster = FALSE,
  booster_onset = 70,
  disable_sdlfp = FALSE
)

## S3 method for class 'prescription_target'
print(x, ...)

## S3 method for class 'prescription_target'
summary(object, ...)

## S3 method for class 'prescription_target'
plot(x, ...)
```

Arguments

`speech`	Input speech spectrum level at each frequency. If a single number is provided, it's assumed to be the overall broadband SPL.
`threshold`	Hearing threshold level at each frequency.

`freq`	Frequencies at which the thresholds are measured.
`gender`	Gender of the patient ("male", "female").
`experience`	Hearing aid experience ("new", "experienced").
`config`	Fitting configuration ("unilateral", "bilateral").
`coupling`	Acoustic coupling ("custom_occluded", "open_dome", "tulip_dome", "double_dome", "vent_1mm_solid", etc.).
`module`	Fitting module ("standard", "cin").
`ldl`	Loudness Discomfort Levels (optional).
`loss`	Conductive hearing loss component (optional).
`distortion_category`	Distortion category ("Normal", "Low", "Moderate", "High").
`user_cr`	User defined compression ratio (optional).
`optimize`	Optimization flag.
`seed_noise`	Random noise for optimizer seeding.
`optim_method`	Optimization method.
`abg_fraction`	Air-bone gap fraction to compensate.
`enable_severe_booster`	Logical flag to enable severe-loss booster.
`booster_onset`	Threshold for the severe-loss booster (default: 70).
`disable_sdlfp`	Logical flag to disable the Slope-Dependent Low-Frequency Penalty (SD-LFP).
`x`	A `prescription_target` object.
`object`	A `prescription_target` object.
`...`	Additional graphical or printing parameters.

Details

This experimental function generates non-linear prescriptive targets for research and modeling purposes. It is not intended for clinical use.

Value

An object of class `prescription_target`.

Author(s)

Mark Shaver

`prescribe_compression` *Prescribe Compression Settings based on Hearing Loss*

Description

Provides recommendations for compression speed (fast vs slow acting) and compression ratio/release times based on the four-frequency Pure Tone Average (PTA4).

Usage

```
prescribe_compression(freq, threshold, module = "standard")
```

Arguments

`freq`	A numeric vector of frequencies.
`threshold`	A numeric vector of hearing thresholds.
`module`	The operating module ("standard", "cin", "mhl").

Value

A list containing compression recommendations.

`sic.critical` *Alternative ANSI/ASA S3.5-1997 (R2024) SII Transfer Function Weights*

Description

Alternative ANSI/ASA S3.5-1997 (R2024) Speech Intelligibility Index (SII) transfer function weights for for various types of speech material.

Usage

```
data(sic.critical)
data(sic.onethird)
data(sic.octave)
```

Format

Each data frame contains the following 8 variables, each corresponding the the transfer function weights for a specific type of speech material:

`fi` Center frequency, Hz

`SII` Standard SII transfer function (weights)

`NNS` NNS (various nonsense syllable tests where most of the English phonems occur equally often)

`CID22` CID-W22 (PB-words)

`NU6` NU6 monosyllables

`DRT` DRT (Diagnostic Rhyme Test)

`ShortPassage` short passages of easy reading material

`SPIN` SPIN monosyllables

`CST` Connected Speech Test

Details

`sic.critical` provides alternative weights for the critical band SII procedure.

`sic.onethird` provides alternative weights for the one-third octave frequency band SII procedure.

`octave` provides alternative weights for the octave frequency band SII procedure.

note

There is no table of alternative weights for the equally-weighted SII band procedure as the weights for this method are (by definition) constant across all bands.

Source

All values except the `CST` columns are from:

ANSI/ASA S3.5-1997 (R2024), "American National Standard Methods for Calculation of the Speech Intelligibility Index" American National Standards Institute, New York.

Values in the `CST` columns are from the original Connected Speech Test (CST) dataset.

References

ANSI/ASA S3.5-1997 (R2024), "American National Standard Methods for Calculation of the Speech Intelligibility Index" American National Standards Institute, New York.

Examples

```
## Load the alternative weights for the critical band method
data(sic.critical)

## display the weights
round(sic.critical,3)

## draw a comparison plot
ngroup <- ncol(sic.critical)
matplot(x=sic.critical[,1], y=sic.critical[,-1],
        type="o",
        xlab="Frequency, Hz",
        ylab="Weight",
        log="x",
        lty=1:ngroup,
        col=rainbow(ngroup)
)
legend(
       "topright",
       legend=names(sic.critical)[-1],
       pch=as.character(1:ngroup),
       lty=1:ngroup,
       col=rainbow(ngroup)
       )

data(onethird)
data(octave)
```

Description

Compute the Speech Intelligibility Index (SII) described by ANSI specification S3.5-1997, including extensions for conductive hearing loss. Optionally apply interpolation obtain values for the required frequencies.

Usage

```
sii(speech = c("normal", "raised", "loud", "shout"),
    noise, threshold, loss, freq,
    method = c("critical", "equal-contributing",
               "one-third octave", "octave"),
    importance = c("SII", "NNS", "CID22", "NU6", "DRT",
                   "ShortPassage", "SPIN", "CST"),
    interpolate=FALSE,
    prescription=NULL,
    desensitization=FALSE,
    ldl=NULL,
    gender="male",
    experience="experienced",
    config="bilateral",
    coupling="custom_occluded",
    module="standard",
    transducer="inserts",
    custom_gain=NULL,
    measured_wrs=NULL,
    wrs_level=NULL,
    distortion_category=NULL,
    nal_ldf=FALSE, ...)
## S3 method for class 'SII'
print(x, digits=3, ...)
## S3 method for class 'SII'
plot(x, clinical=FALSE, legend=TRUE, legend_only=FALSE, ...)
## S3 method for class 'SII'
summary(object, digits=2, ...)
```

Arguments

`speech`	Either a numeric vector providing E'_i, the equivalent speech spectrum level (in dB) at each frequency, or a character string indicating the stated vocal effort corresponding to one of the standard standard speech spectrum levels ("normal", "raised", "loud", "shout"). Defaults to `speech="normal"` correspoding to the normal level of stated vocal effort.
`noise`	A numeric vector providing N'_i, the equivalent noise spectrum level (in dB) at each frequency. If missing, defaults to -50 dB for each frequency.

`threshold` A numeric vector providing T_i', the equivalent hearing threshold level (in dB) at each frequency. If missing, defaults to 0 dB for each frequency.

`loss` A numeric vector providing J_i', the conductive hearing loss level (in dB) at each frequency. If missing, defaults to 0 dB for each frequency.

`freq` Vector of frequencies for which `speech`, `noise`, `threshold`, and/or `loss` are specified. If `interpolate=TRUE`, `freq` must be specified. Otherwise, it must either match the required value for SII calculation method given by argument `method`, or be missing, in which case it will default to the values required for the specified method.

`method` A character string specifying the SII calculation method ("critical", "one-third octave", "equal-contributing", "octave")

`importance` Either a numeric vector providing F_i, the transfer function (importance weights) at each frequency, or a character string indicating which transfer function to employ ("SII", "NNS", "CID22", "NU6", "DRT", "ShortPassage", "SPIN", "CST"). Defaults to the standard SII transfer function, `importance="SII"`.

`interpolate` Logical flag indicating whether to interpolate from the provide measurement values and frequencies to those required by the specified method via linear interpolation on the log scale.

`prescription` A character string (e.g. `"NAL-R"`, `"Open-NL"`). If provided, the function will automatically calculate hearing aid insertion gain based on the specified prescriptive fitting rationale and add it to the speech and noise spectrum, calculating an Aided SII.

`desensitization`
Logical flag. If `TRUE`, applies the hearing loss desensitization factor (Ching et al., 2011; Johnson, 2013) to the calculation to account for suprathreshold distortion in impaired ears. Defaults to `FALSE` (Traditional ANSI S3.5 calculation).

`ldl` Numeric vector specifying Loudness Discomfort Levels (LDL) in dB HL for each frequency. If `NULL`, limits are estimated based on hearing thresholds.

`gender` Character string specifying patient gender ("male" or "female"). Used for prescriptive algorithms. Defaults to "male".

`experience` Character string specifying user hearing aid experience ("experienced", "new"). Defaults to "experienced".

`config` Character string for fitting configuration ("unilateral", "bilateral"). Bilateral fittings prescribe slightly less gain (~3 dB) to account for binaural loudness summation.

`coupling` Character string specifying the acoustic coupling / vent ("custom_occluded", "double_dome", "tulip_dome", "open_dome", "vent_1mm_solid", "vent_2mm_solid", "vent_3mm_solid", "vent_1mm_hollow", "vent_2mm_hollow", "vent_3mm_hollow"). Modifies low-frequency leakage and insertion gain targets.

`module` Character string for hearing aid module ("standard" or other specific module constraints).

`transducer` Character string specifying the audiometric transducer used for testing ("inserts" or "supra_aural"). Determines RETSPL and RECD corrections for SPLogram plots.

`custom_gain` Numeric vector specifying a custom insertion gain array (in dB) for each frequency. If provided along with `prescription="Custom"`, this gain is explicitly applied.

`measured_wrs` Numeric value specifying the clinical Word Recognition Score (WRS) in percent (0-100). When provided, invokes the Margolis (2025) speech distortion model.

`wrs_level` Numeric value specifying the presentation level (in dB HL or SPL) of the clinical WRS test.

`distortion_category`
Character string ("Mild", "Moderate", or "Severe") specifying the degree of speech distortion. If `NULL`, it is estimated based on `measured_wrs` and `wrs_level`.

`nal_ldf`	Logical flag indicating whether to use NAL-NL2 loudness discomfort factors (optional).
`object`	SII object
`x`	SII object
`digits`	Number of digits to display
`clinical`	Logical flag. If `TRUE`, plots a patient-friendly clinical SPLogram highlighting the audible speech area. If `FALSE` (the default), the behavior depends on whether the object is aided or unaided. For unaided objects, it plots a diagnostic interpolation graph. For aided objects (when a prescription was used), it automatically generates a 3-line Insertion Gain plot for 55, 65, and 75 dB SPL input levels (dynamically loading and scaling the true Normal and Loud LTASS spectra).
`legend`	Logical flag. If `TRUE` (the default), draws a legend on the plot.
`legend_only`	Logical flag. If `TRUE`, suppresses the main plot and only draws the legend in the center of the plotting window. Useful for multi-plot layouts.
`...`	Optional arguments to `print`, `summary`, and `plot` methods

Details

American National Standard ANSI/ASA S3.5-1997 (R2024) ("Methods for Calculation of the Speech Intelligibility Index") defines a method for computing a physical measure that is highly correlated with the intelligibility of speech as evaluated by speech perception tests given a group of talkers and listeners. This measure is called the Speech Intelligibility Index, or SII. The SII is calculated from acoustical measurements of speech and noise.

The `sii` function implements ANSI/ASA S3.5-1997 (R2024) as described in the standard, without any attempt to optimize the performance. The implementation does, however, include the extension for handling conductive hearing loss from Annex A (utilizing the optional `loss` argument), and for utilizing alternative band weights (i.e. transfer function) appropriate for differing message contents (e.g. types of speech) as described in Annex B or user-specified band weights (utilizing the optional argument `importance`).

Further, this implementation provides a mechanism for interpolating/extrapolating available measurements to those required for the specified calculation procedure. When `interpolate=TRUE`, required values for `speech`, `noise`, `threshold`, and `loss` will be computed using linear interpolation (of the log-scaled data). In this case, missing values may be provided and will be appropriately interpolated.

Prescriptive Fitting Rationales: If the `prescription` argument is provided, the function will dynamically calculate a frequency-specific hearing aid insertion gain and apply it to the speech and noise spectrum, calculating an Aided SII. The following rationales are supported:

- `"NAL-R"`: A classic linear fitting rationale designed by the National Acoustic Laboratories to maximize speech intelligibility for mild-to-moderate losses. Gain is a linear function of the pure-tone average (PTA) and frequency-specific thresholds.
- `"Open-NL"`: A completely transparent, open-source non-linear fitting algorithm developed specifically for this package. It was designed to serve as an open-source alternative to proprietary modern non-linear clinical targets (such as NAL-NL2 and DSL v5.0). The algorithm calculates insertion gain using the following explicit mathematical steps:
  1. **Conductive Component Separation**: If an Air-Bone Gap is present, the purely sensorineural component is isolated for WDRC compression. Linear gain representing 75% of the conductive loss is added at the end (consistent with NAL-NL2/Johnson).
  2. **Dynamic Base Gain**: The base gain relies on a dynamic multiplier (ranging from 0.43 to 0.48) determined by user experience, combined with empirical NAL-R shaping constants. Severe losses receive a booster, while steep slopes receive a low-frequency penalty to avoid upward spread of masking.

3. **Dead Region Roll-offs**: High-frequency or low-frequency dead regions trigger a steep 30 dB/octave penalty beyond the viable boundaries to prevent acoustic distortion and feedback.
4. **Wide Dynamic Range Compression (WDRC)**: A dynamic compression ratio (ranging from 1.0 to 3.0) is applied symmetrically around a 65 dB SPL pivot point. The ratio scales dynamically based on the sensorineural threshold and measured Loudness Discomfort Levels (LDLs).
5. **Bandwidth Roll-off**: Empirical bandwidth roll-offs are applied to limit unnecessary low/high frequency amplification, with separate curves for adults vs. infants.
6. **Acoustic Coupling**: Adjustments are applied to the low frequencies depending on the specified venting, dome type, and acoustic seal.
7. **MPO Limits**: Output is capped by predictive NAL-SSPL90 limits, adjusting for conductive attenuation, with an absolute safety hard cap at 120 dB SPL at the cochlea and a hardware output limit of 135 dB SPL.

Value

The return value is an object of class SII, containing the following components:

`call` Function call used to generate the SII object

`orig` List containing original (pre-extrapolation) values for `freq`, `speech`, `noise`, `threshold`, and `loss`.

`speech`, `noise`, `threshold`, `loss`, and `freq`
Values used in calculations (extrapolated if necessary)

`unaided_speech`
Original speech array before applying any prescriptive gain

`vocal_effort` String representing the stated vocal effort

`gain` Insertion gain array added to speech and noise when a prescription is used

`prescription` The fitting rationale used (if any)

`unaided_sii` Calculated SII value before applying the prescriptive gain (if a prescription was used)

`table` SII calculation worksheet, containing columns corresponding to both Table C.1 and C.2 in Annex C of the standard. Table columns are

Fi Center frequency of SII band, Hz
E'i Spectrum level of equivalent speech, dB
N'i Spectrum level of equivalent noise, dB
T'i Equivalent hearing threshold level, dB
Vi Spectrum level for self-speech masking, dB
Bi Larger of the specrum levels for equivalent noise and self-speech masking, dB
Ci Slope per octave (doubling of frequency) of the upward spread of masking, dB/octave
Zi Spectrum level for equivalent masking, dB
Xi Spectrum level of internal noise, dB
X'i Spectrum level of equivalent internal noise, dB
Di Spectrum level for equivalent disturbance, dB
Ui Spectrum level of standard speech for normal vocal effort, dB
Ji Equivalent hearing threshold due to conductive hearing loss, dB
Li Speech level distortion factor, dB
Ki Temporary variable used in the calculation of the band auditability function
Ai Band auditability function
Ii Band importance function
IiAi Product of the band importance function (Ii), and band auditability function(Ai)

`sii` Calculated SII value

Open-NL Prescription Rationale

DISCLAIMER: `Open-NL` is an untested, experimental fitting rationale. It is designed to mimic aspects of other generic non-linear fitting rationales (such as NAL-NL2 and DSL v5.0) strictly to promote open research, algorithmic transparency, and rapid iteration within the audiology community. It is not intended for clinical use.

When `prescription = "Open-NL"` is selected, the function calculates and applies this WDRC-optimized rationale to maximize the aided Speech Intelligibility Index.

The mathematical algorithm consists of:

1. **Minimal Hearing Loss (MHL) Bypass:** If `module == "mhl"` and $PTA_{.5,1,2,4k} \leq 25$ dB HL, WDRC is bypassed. Applies flat linear gain interpolated from $(f, G) = \{(250, 0), (500, 0), (1k, 3), (2k, 5), (4k, 5), (8k, 5)\}$ dB, with a 1.5 compression ratio for loud inputs.
2. **Base Anchor (65 dB SPL Input):** $g_{65} = \max(0, m \times HTL + C)$, where $m \in \{0.40, 0.45, 0.50\}$ depending on `experience`, and C interpolates arrays such as $\{-8, -1, +3, +1, 0, 0, 0, 0\}$ dB evaluated at standard audiometric frequencies.
3. **Audiometric Profile Corrections:**
   - *Steep Slope Knee:* For slopes > 30 dB/octave, penalizes the "knee" (500-1500 Hz) by up to 6 dB, and boosts frequencies ≥ 2000 Hz by up to 6 dB.
   - *Severe-Loss Booster (SLB):* Adds $\min(15, \max(0, HTL - 60) \times 0.5)$ dB. Tapered heavily in mid-frequencies and disabled in dead regions.
4. **High-Frequency Desensitization:** To prevent distortion, excess gain above a limit is compressed at a 2:1 ratio.
   - Limit $= 45 + 1.0 \times \max(0, HTL - 60)$
5. **Roll-offs and Dead Regions:**
   - *Bandwidth Roll-off:* Multiplies g_{65}. Interpolates $\{(250Hz, 0.7), (500, 1.0), ..., (6k, 0.8), (8k, 0.5)\}$.
   - *Dead Regions:* Identifies HF Dead Regions ($HTL \geq 90$ at $f \geq 1k$) and LF Dead Regions ($HTL \geq 80$ at $f \leq 1k$). Applies a 30 dB/octave penalty beyond viable boundaries ($1.7 \times f_e$ for HF, $0.57 \times f_e$ for LF).
6. **Bi-directional WDRC:**
   - *Compression Ratio (CR):* $CR_{base} = 1 + \max(0, HTL - 20)/40$. For $HTL > 65$, CR reduces toward 1.0 in low frequencies. Overall bounded strictly to ≤ 1.5 for low frequencies (≤ 500 Hz) and up to 2.4 for high frequencies (≥ 3000 Hz).
   - *Compression Threshold (CT):* Interpolated from (HTL, CT): $\{(20HL, 30SPL), ..., (100, 45)\}$.
   - *Aggressive MPO Defense:* For steeply sloping losses (> 15 dB difference) with low LDLs (< 100 dB SPL), the formula proactively defends against MPO collision. The CT is aggressively lowered by up to 10 dB, and the CR is forced up by an additional 0.05 per dB of LDL penalty.
   - *I/O Computation:* Calculates gain at CT (G_{CT}) scaling back from 65 dB SPL pivot. Applies linear gain below CT, and WDRC above CT.
7. **Empirical Adjustments and Smoothing:**
   - *Demographic Boosts:* Gender: Female $= -1.5$ dB. Config: Unilateral $= +3.0$ dB. Experience: New users with $PTA > 40$ get up to -6.0 dB penalty.
   - *LDL Dynamic Range Mapping:* If `ldl` is provided, dynamic range is evaluated. For every 1 dB the measured LDL is lower than predicted, g_{65} is reduced by 0.2 dB and CR_{base} is increased by 0.02.
   - *Acoustic Coupling:* Subtracts vent leakage (e.g., Open Dome $= \{-35, -28, -15, -2, 0, 0\}$ dB).
   - Applies a 3-point moving average to the final gain array.
8. **SSPL90 MPO Limiting:**
   - $MPO_{heuristic} = 100 + 0.5 \times \max(0, HTL - 40)$.

- $MPO_{safe} = LDL_{spl} - 5$.
- $PTS_{safe\_limit} = 105 + 0.5 \times \max(0, HTL - 50)$.
- $MPO_{final} = \min(120, MPO_{heuristic}, MPO_{safe}, PTS_{safe\_limit})$.

Author(s)

Gregory R. Warnes <greg@warnes.net>

References

ANSI/ASA S3.5-1997 (R2024), "American National Standard Methods for Calculation of the Speech Intelligibility Index" American National Standards Institute, New York.

Other software programs for calculating SII are available from https://sii.to/html/programs.html.

See Also

SII Constants: critical, and sic.critical

Examples

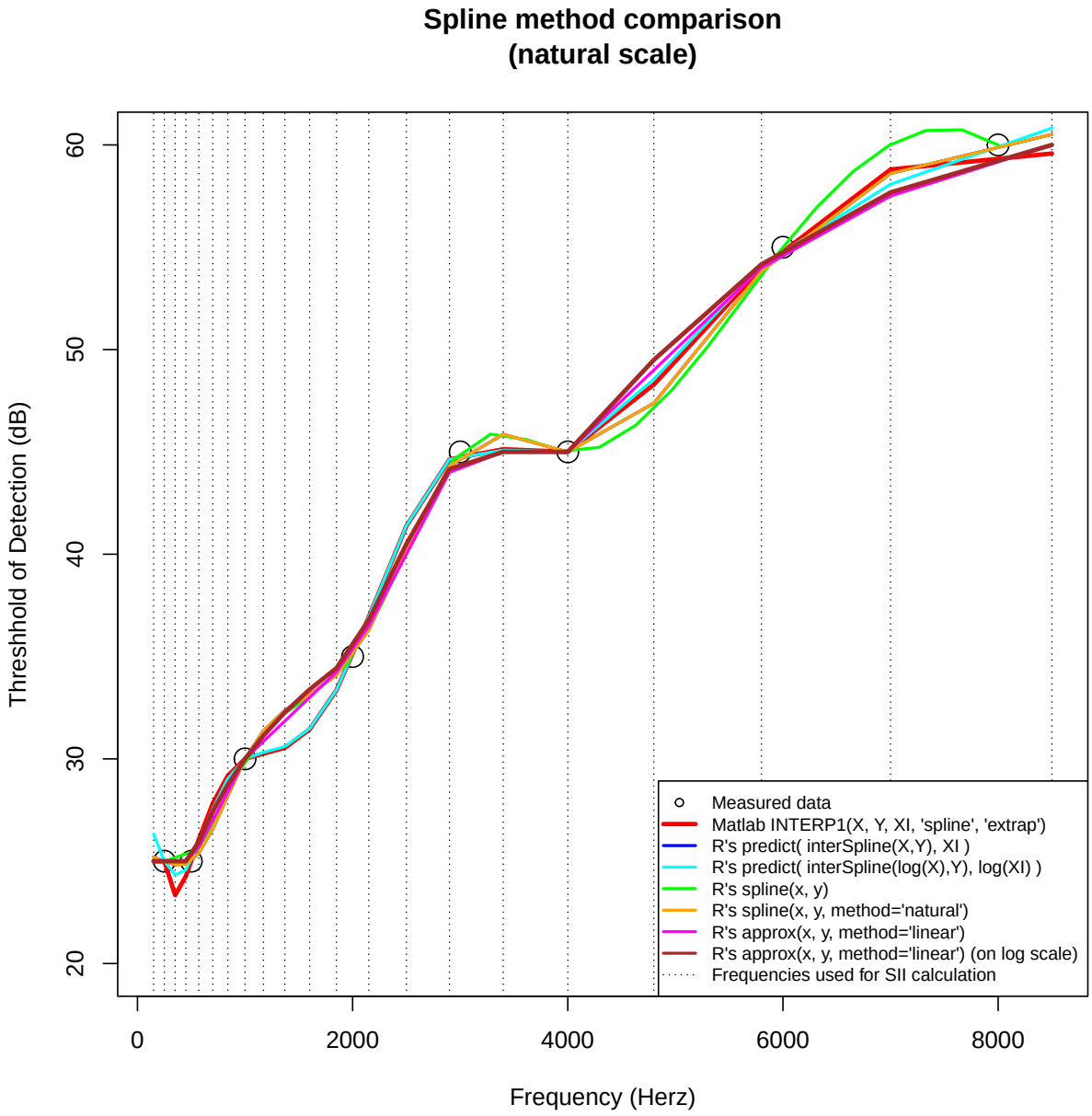
```
## Example C.1 from ANSI/ASA S3.5-1997 (R2024) Annex C
sii.C1 <- sii(
              speech   = c(50.0, 40.0, 40.0, 30.0, 20.0,  0.0),
              noise    = c(70.0, 65.0, 45.0, 25.0,  1.0,-15.0),
              threshold= c( 0.0,  0.0,  0.0,  0.0,  0.0,  0.0),
              method="octave"
       )
sii.C1                        # rounded to 2 digits by default
print(sii.C1$sii, digits=20)  # full precision
summary(sii.C1)               # full details
plot(sii.C1)                  # plot
plot(sii.C1, clinical=TRUE)   # clinical SPLogram plot
## The value given in the Standard is $0.504$.



## Same calculation, but manually specify the frequencies
## and importance function, and use default for threshold

sii.C1 <- sii(
              speech   = c(50.0, 40.0, 40.0, 30.0, 20.0,  0.0),
              noise    = c(70.0, 65.0, 45.0, 25.0,  1.0,-15.0),
              method="octave",
              freq=c(250, 500, 1000, 2000, 4000, 8000),
       importance=c(0.0617, 0.1671, 0.2373, 0.2648, 0.2142, 0.0549)
       )
sii.C1

## Now perform the calculation using frequency weights for the Connected
## Speech Test (CST)
sii.CST <- sii(
               speech   = c(50.0, 40.0, 40.0, 30.0, 20.0,  0.0),
```

```
               noise    = c(70.0, 65.0, 45.0, 25.0,  1.0,-15.0),
               method="octave",
        importance="CST"
       )
round(sii.CST$table[,-c(5:7,13)],2)
sii.CST$sii

## Example C.2 from ANSI/ASA S3.5-1997 (R2024) Annex C

sii.C2 <- sii(
              speech   = rep(54.0, 18),
              noise    = c(40.0, 30.0, 20.0, rep(0, 18-3) ),
              threshold= rep(0.0,  18),
              method="one-third"
              )
sii.C2$table[1:3,1:8]
sii.C2

## Interpolation example, for 8 frequencies using NU6 importance
## weight, default values for noise.
sii.left <- sii(
                speech="raised",
                threshold=c(25,25,30,35,45,45,55,60),
                freq=c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000),
                method="critical",
                importance="NU6",
                interpolate=TRUE
                )
sii.left
```

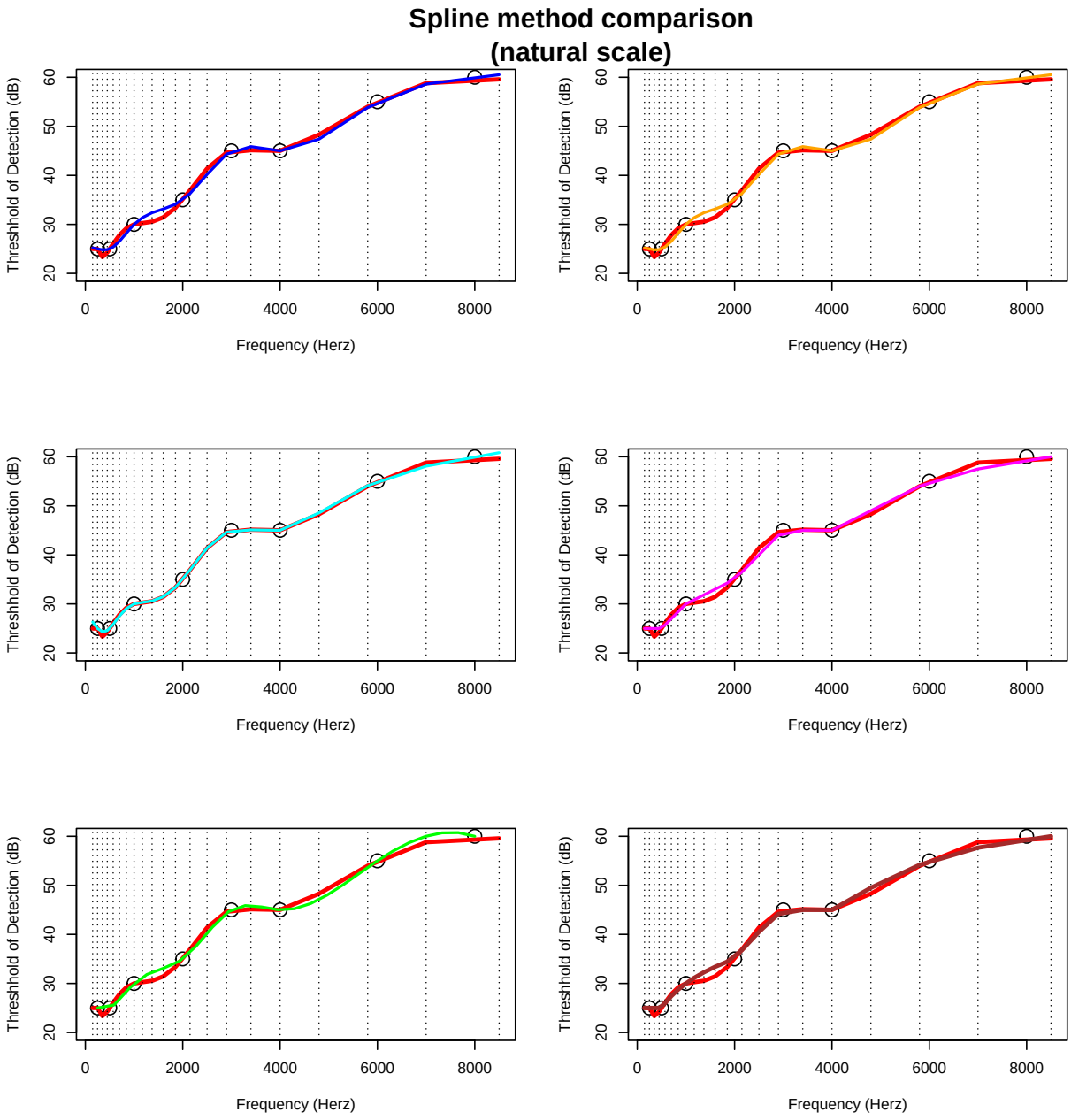
B Interpolation

Do some experimentation to determine how to best perform interpolation/extrapolation from the small set of frequencies where hearing sensitivity was measured to the set of frequencies necessary for the calculation of SII:

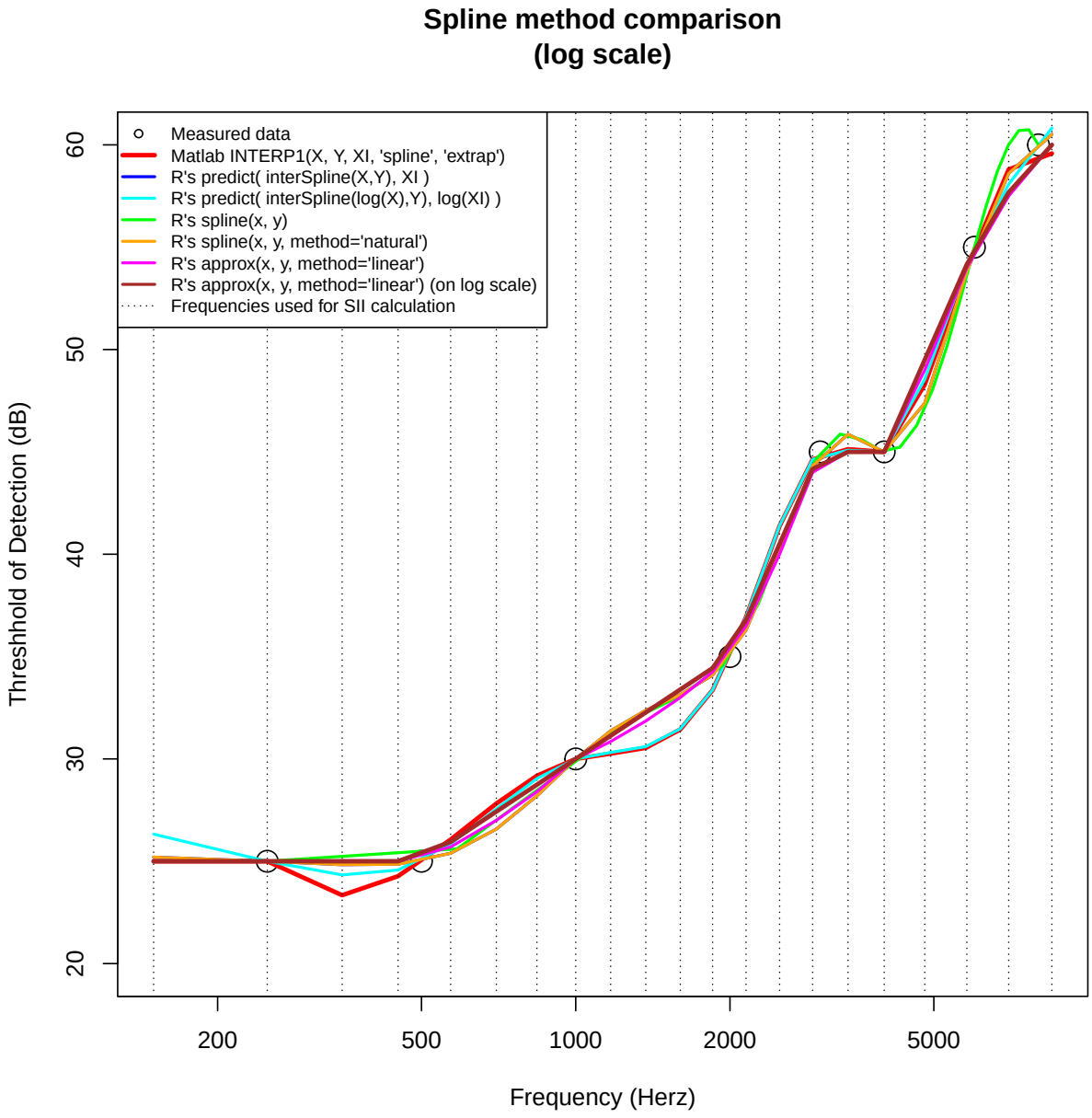
```
> THDI=c(25,25,30,35,45,45,55,60)
> freq=c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000)
> sii.freqs <- SII:::sii.constants[,1]
> xlist <- sort(c(SII:::sii.constants[, 1], freq))
> ylist <- rep(NA, length=length(xlist))
> names(ylist) <- xlist
> ylist[ as.character(freq) ] <- THDI


> library(splines)
> ispl <- interpSpline( THDI ~ freq )
> ispl <- predict(ispl, sii.freqs)$y
> ispl.l <- interpSpline( THDI ~ log(freq) )
> ispl.l <- predict(ispl.l, log(sii.freqs) )$y
> approx.l <- function(x,y,xout,...)
+   {
+     retval <- approx(log(x), y, log(xout), ...)
+     retval$x <- xout
+     retval
+   }
>


> doplot(FALSE, FALSE)
```



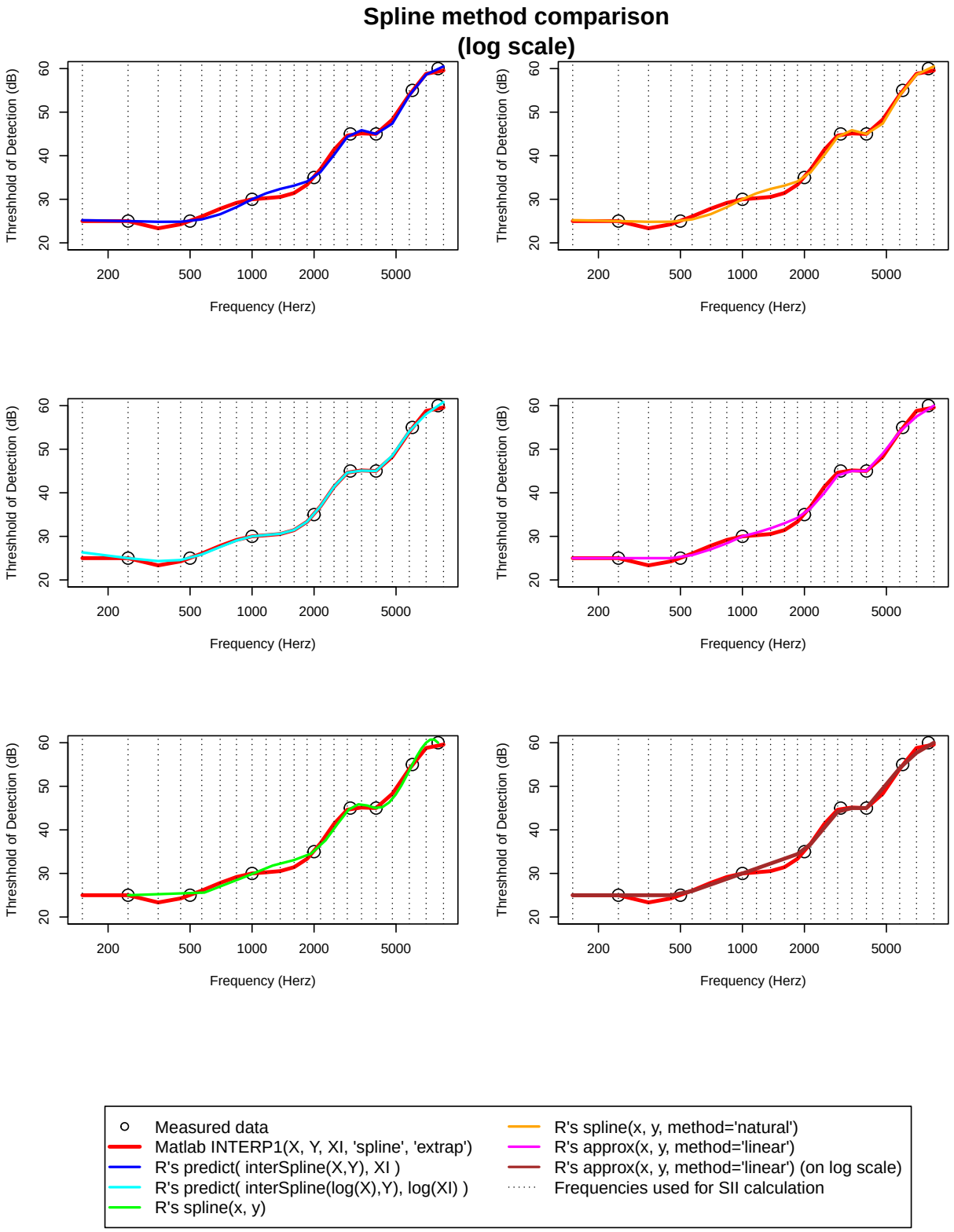

```
> doplot(FALSE, TRUE)
```


Spline method comparison
(natural scale)
Threshhold of Detection (dB)
Frequency (Herz)
20
30
40
50
60
0
2000
4000
6000
8000
Measured data
Matlab INTERP1(X, Y, XI, 'spline', 'extrap')
R's predict(interSpline(X,Y), XI)
R's predict(interSpline(log(X),Y), log(XI))
R's spline(x, y)
R's spline(x, y, method='natural')
R's approx(x, y, method='linear')
R's approx(x, y, method='linear') (on log scale)
Frequencies used for SII calculation

```
> doplot(TRUE, FALSE)
```




```
> doplot(TRUE, TRUE)
```


Spline method comparison
(log scale)
Threshhold of Detection (dB)
Frequency (Herz)
20
30
40
50
60
200
500
1000
2000
5000
Measured data
Matlab INTERP1(X, Y, XI, 'spline', 'extrap')
R's predict(interSpline(X,Y), XI)
R's predict(interSpline(log(X),Y), log(XI))
R's spline(x, y)
R's spline(x, y, method='natural')
R's approx(x, y, method='linear')
R's approx(x, y, method='linear') (on log scale)
Frequencies used for SII calculation

C SII function definition

C.1 Excel to R translation

First, I have defined an R function that simply translates the Excel code into R:

```
> args(SII:::sii.excel)
```

```
function (THDI, freq = c(250, 500, 1000, 2000, 3000, 4000, 6000,
    8000), matlab.spline = TRUE)
NULL
```

C.2 R code written from the ANSI/ASA S3.5-1997 (R2024)

Second, I implemented the SII calculation following the text of the standard. This code is more general and quite a bit more complex.

```
> args(sii)
```

```
function (speech = c("normal", "raised", "loud", "shout"), noise,
    threshold, loss, freq, method = c("critical", "equal-contributing",
        "one-third octave", "octave"), importance = c("SII",
        "NNS", "CID22", "NU6", "DRT", "ShortPassage", "SPIN",
        "CST"), interpolate = FALSE, prescription = NULL, desensitization = FALSE,
    ldl = NULL, gender = "male", experience = "experienced",
    config = "bilateral", coupling = "custom_occluded", module = "standard",
    transducer = "inserts", custom_gain = NULL, measured_wrs = NULL,
    wrs_level = NULL, distortion_category = NULL, nal_ldf = FALSE,
    ...)
NULL
```

C.3 Utility functions

I also defined a utility function to nicely print the output.

```
>
```

In order to make it easy to maintain the SII constants, a single Microsoft Excel file, `\data\SII_Constants.xls` is included in the SII `data` directory containing the complete set of SII constants, where each tab corresponds to one Table from the standard.

D Tests

D.1 Compare interpolation details for example data

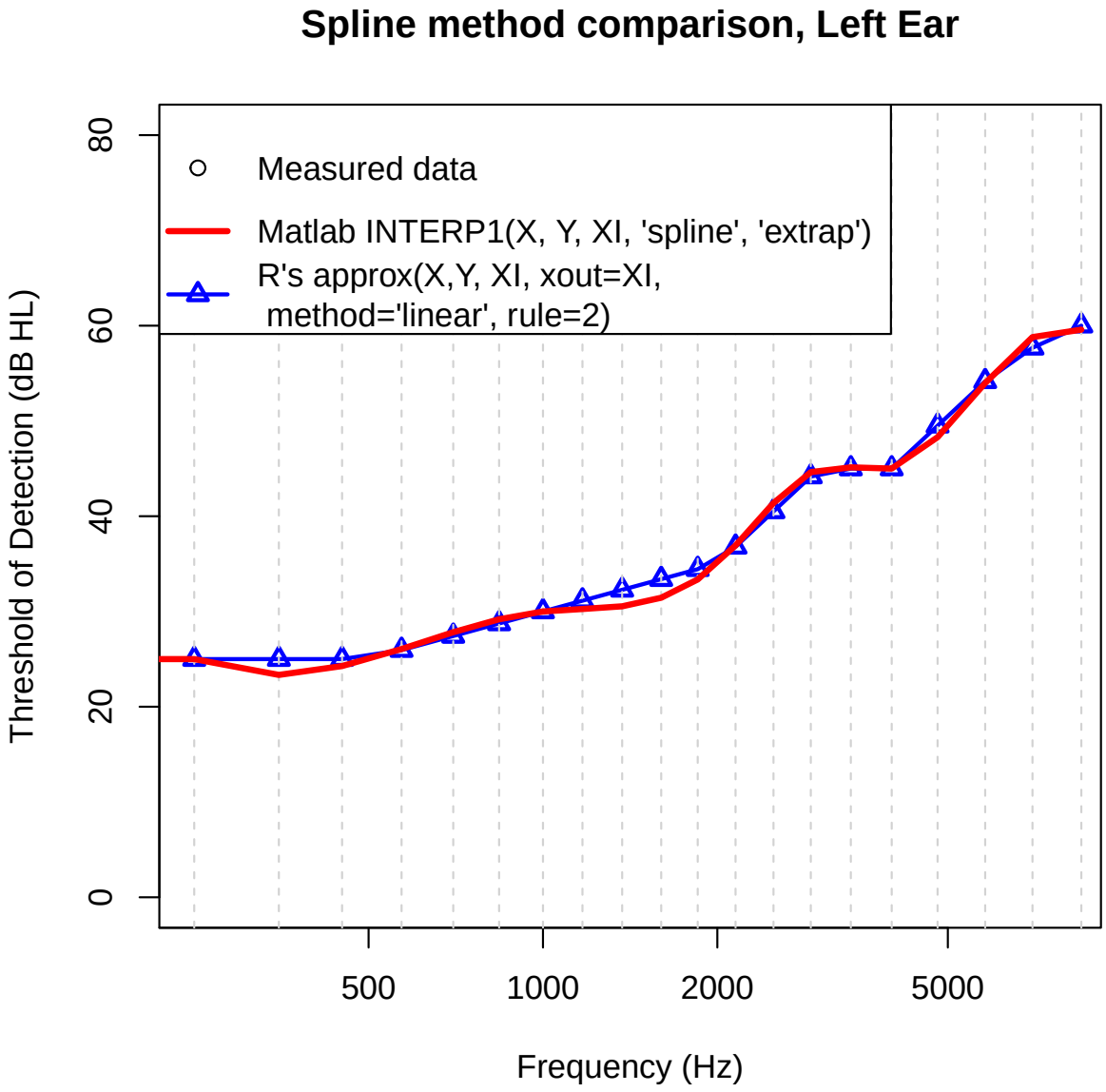
D.1.1 Left Ear

```
> sii.left <- sii(
+                 speech="raised",
+                 threshold=c(25,25,30,35,45,45,55,60),
+                 freq=c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000),
+                 importance="NU6",
+                 interpolate=TRUE
+                 )
> ## comparison of our interpolation and matlab's
> tab <- cbind(
+              matlab=SII:::sii.constants[,"Ti'.THDN"],
+              R     =sii.left$table[,"T'i"],
+              delta =NA
+              )
> tab[,3] <- tab[,1] - tab[,2]
> rownames(tab) <- SII:::sii.constants[,".NFreqLin"]
> t(round(tab,2))
```

```
       150 250   350   450   570   700   840 1000  1170  1370  1600  1850  2150
matlab  25  25 23.34 24.27 26.06 27.82 29.18   30 30.26 30.54 31.44 33.36 36.93
R       25  25 25.00 25.00 25.95 27.43 28.74   30 31.13 32.27 33.39 34.44 36.78
delta    0   0 -1.66 -0.73  0.11  0.39  0.44    0 -0.87 -1.73 -1.95 -1.08  0.15
        2500  2900  3400 4000  4800  5800  7000  8500
matlab 41.38 44.63 45.13   45 48.29 53.98 58.80 59.58
R      40.50 44.16 45.00   45 49.50 54.16 57.68 60.00
delta   0.88  0.47  0.13    0 -1.21 -0.18  1.12 -0.42
```

```
> compare.plot <- function(x, matlab, title)
+   {
+     plot(x)
+     lines(SII:::sii.constants[,".NFreqLin"],
+           matlab,
+           type="l", col="red", lwd=3)
+
+     legend("topleft",
+            legend=c(
+              "Measured data",
+              "Matlab INTERP1(X, Y, XI, 'spline', 'extrap')",
+              "R's approx(X,Y, XI, xout=XI,\n method='linear', rule=2)"
+              ),
+            col=c("black", "red","blue","green","orange","magenta"),
+            pch=c(      1,    NA,    2,     NA,      NA,       NA),
+            lty=c(     NA,     1,    1,      1,       1,        1),
+            lwd=c(     NA,     3,    2,      2,       2,        2),
+            bg="white"
+            )
```

```
+     title(title)
+   }


> compare.plot(sii.left, matlab=SII:::sii.constants[, "Ti'.THDN"], title="Spline method comparison, Left
>
```



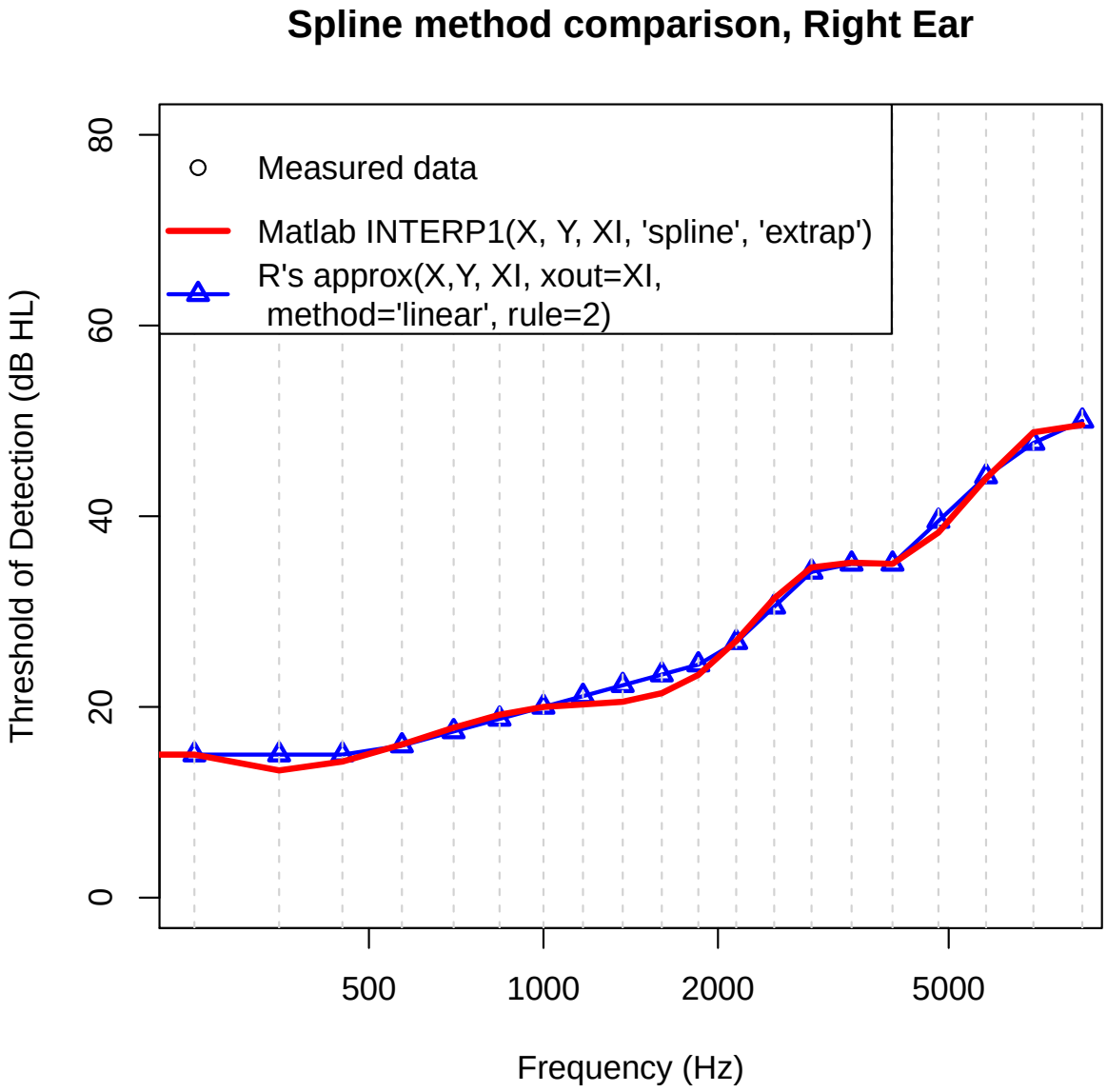
D.1.2 Right Ear

```
> # comparison of our interpolation and matlab's
>
> matlab <- c(15.00, 15.00, 13.34, 14.27, 16.06, 17.82, 19.18, 20.00,
```

```
+             20.26, 20.54, 21.44, 23.36, 26.93, 31.38, 34.63, 35.13,
+             35.00, 38.29, 43.98, 48.80, 49.58)
> sii.right <- sii(
+                  speech="raised",
+                  threshold=c(15,15,20,25,35,35,45,50),
+                  freq=c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000),
+                  importance="NU6",
+                  interpolate=TRUE
+                  )
> tab <- cbind(
+              matlab=matlab,
+              R     =sii.right$table    [,"T'i"],
+              delta =NA
+              )
> tab[,3] <- tab[,1] - tab[,2]
> rownames(tab) <- SII:::sii.constants[,".NFreqLin"]
> t(round(tab,2))
```

```
       150 250   350   450   570   700   840 1000  1170  1370  1600  1850  2150
matlab  15  15 13.34 14.27 16.06 17.82 19.18   20 20.26 20.54 21.44 23.36 26.93
R       15  15 15.00 15.00 15.95 17.43 18.74   20 21.13 22.27 23.39 24.44 26.78
delta    0   0 -1.66 -0.73  0.11  0.39  0.44    0 -0.87 -1.73 -1.95 -1.08  0.15
        2500  2900  3400 4000  4800  5800  7000  8500
matlab 31.38 34.63 35.13   35 38.29 43.98 48.80 49.58
R      30.50 34.16 35.00   35 39.50 44.16 47.68 50.00
delta   0.88  0.47  0.13    0 -1.21 -0.18  1.12 -0.42
```

```
> compare.plot(sii.right, matlab=matlab, title="Spline method comparison, Right Ear")
```



D.2 Comparison with Excel implementation

Test the SII function using the example data from the Excel worksheet:

- Left ear example
  Excel code: 0.43.
  "Translated" R code:

```
> SII:::sii.excel(
+           c(25,25,30,35,45,45,55,60),
+           c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000)
+           )
```

```
SII: 0.373
```

New R code:

```
> sii.left <- sii(
+                 speech="raised",
+                 threshold=c(25,25,30,35,45,45,55,60),
+                 freq=c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000),
+                 importance="NU6",
+                 interpolate=TRUE
+                 )
> sii.left
```

```
SII: 0.77
```

- Right ear example

  Excel worksheet: 0.72.

  "Translated" R code:

```
> SII:::sii.excel(
+           c(15,15,20,25,35,35,45,50),
+           c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000)
+           )
```

```
SII: 0.717
```

  New R code:

```
> sii.right <- sii(
+                  speech="raised",
+                  threshold=c(15,15,20,25,35,35,45,50),
+                  freq=c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000),
+                  importance="NU6",
+                  interpolate=TRUE
+                  )
> sii.right
```

```
SII: 0.906
```

- Best possible score (No detectible threshold)

  Excel Spreadsheet: 0.9887

  "Translated" R code:

```
> SII:::sii.excel(
+           rep(0,8),
+           freq=c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000)
+           )
```

```
SII: 0.989
```

  New R code:

```
> sii.best <- sii(
+                 threshold=rep(0,8),
+                 freq=c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000),
+                 interpolate=TRUE
+                 )
> sii.best
```

```
SII: 0.996
```

- Worst possible score (No detectible threshold)

  Excel Spreadsheet: 0.00

  "Translated" R code:

```
> SII:::sii.excel(
+           rep(100,8),
+           c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000)
+           )

SII: 0
```

  New R code:

```
> sii.worst <- sii(
+                  threshold=rep(100,8),
+                  freq=c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000),
+                  interpolate=TRUE
+                 )
> sii.worst

SII: 0
```

D.3 Test handling of missing values

```
> sii.worst <- sii(
+                  threshold=c(NA, rep(100,7)),
+                  freq=c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000),
+                  interpolate=TRUE
+                 )
> sii.worst


SII: 0


> sii.right <- sii(
+                  speech="raised",
+                  threshold=c(0,15,15,20,25,35,35,45,50),
+                  freq=c(NA, 250, 500, 1000, 2000, 3000, 4000, 6000, 8000),
+                  importance="NU6",
+                  interpolate=TRUE
+                  )
> sii.right


SII: 0.906


> ## This should fail, because there is no data!
> sii.NONE <- try(
+              sii(
+                  threshold=rep(NA,8),
```

```
+                  freq=c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000),
+                  interpolate=TRUE
+                 )
+             )
> sii.NONE
```

```
SII: NA
```

```
> sii.right <- sii(
+                  speech="raised",
+                  threshold=c(15,15,20,NA,35,35,45,50),
+                  freq=c(250, 500, 1000, 2000, 3000, 4000, 6000, 8000),
+                  importance="NU6",
+                  interpolate=TRUE
+                  )
> sii.right
```

```
SII: 0.895
```

```
> sii.C1.NA <- sii(
+               speech   = c(50.0, 40.0, 40.0, NA,   20.0,  0.0),
+               noise    = c(70.0, 65.0, 45.0, 25.0,  1.0,-15.0),
+               threshold= c( 0.0,  0.0,  0.0,  0.0,   NA,  0.0),
+               freq     = c( 250,  500, 1000, 2000, 4000, 8000),
+               method="octave",
+               importance="CST",
+               interpolate=TRUE
+               )
> sii.C1
```

```
SII: 0.487
```

```
> sii.C1.NA
```

```
SII: 0.487
```

E Final cleanup

Save the data we created here for debugging purposes:

```
> save.image("SII-Code.Rda")
```