Data analysis

"Investigation of the emission rate of particles when musicians play wind, woodwind and brass instruments'"

Lukas Schumann[1], Dorothea von Zadow[2], Alexander Schmidt[2,3], Isabell Fernholz[2,3], Anne Hartmann[1], Liliana Ifrim[2], Martin Kriegel[1], Joachim Seybold[4], Dirk Mürbe[2], Mario Fleischer[2]

[1]Technische Universität Berlin, Hermann-Rietschel-Institut, Berlin, Germany
[2]Charité – Universitätsmedizin Berlin, Corporate Member of Freie Universität Berlin and Humboldt-Universität Berlin, Department of Audiology and Phoniatrics, Berlin, Germany
[3] Kurt-Singer-Institute for Music Physiology and Musicians Health, Hanns Eisler School of Music, Berlin, Germany
[4]Charité – Universitätsmedizin Berlin, Corporate Member of Freie Universität Berlin and Humboldt-Universität Berlin, Medical Directorate, Berlin, Germany

E-mail - mario.fleischer@charite.de

Contents

1 Definitions

1.1 Read data

Set working directory according to your file hierarchy. The file "data.csv" is part of the supplemental data.

```
rm(list = ls())
winds.data <- read.csv('data.csv')
singers.data.medians <- read.csv('data_sci_reports_muerbe_2021_medians.csv')
# singers.data.medians$PVR <- NULL
system('rm -f Schumann_Fig*.jpg Schumann_Fig*.pdf Schumann_Tab*.*')
```

1.2 Load routines

```
##
## Attache Paket: 'rstatix'
```

```
## Das folgende Objekt ist maskiert 'package:stats':
##
##     filter
```

1.3 Clean data from non-respiratory particles

We defined non-respiratory particles as "bias'', which was provoked in additional tasks where the participants played the orchestral excerpts and 'Ode to the joy' by moving their fingers only and without any air flow into the instrument driven by the mouth (unplayed sequences).

1.3.1 Non-biased data

```
winds.data.ready <- dplyr::filter(winds.data,
                            (Condition == 'piano' |
                             Condition == 'forte' |
                             Condition == 'breathing' |
                             Condition == 'speaking'))
```

1.3.2 Unplayed sequences

```
winds.data.unplayed <- dplyr::filter(winds.data,
                            (Condition == 'orchestral_excerpt_unplayed' |
                             Condition == 'ode_unplayed' ))
```

1.3.3 Biased sequences

```
winds.data.played <- dplyr::filter(winds.data,
                            (Condition == 'orchestral_excerpt_played' |
                             Condition == 'ode_played'))
```

1.3.4 Compute un-biased sequences

```
winds.data.played$PM_0.3_0.5um <- winds.data.played$PM_0.3_0.5um-
                                          winds.data.unplayed$PM_0.3_0.5um
winds.data.played$PM_0.5_1.0um <- winds.data.played$PM_0.5_1.0um-
                                          winds.data.unplayed$PM_0.5_1.0um
winds.data.played$PM_1.0_3.0um <- winds.data.played$PM_1.0_3.0um-
                                          winds.data.unplayed$PM_1.0_3.0um
winds.data.played$PM_3.0_5.0um <- winds.data.played$PM_3.0_5.0um-
                                          winds.data.unplayed$PM_3.0_5.0um
winds.data.played$PM_5.0_10.0um <- winds.data.played$PM_5.0_10.0um-
                                          winds.data.unplayed$PM_5.0_10.0um


winds.data.played$PM_0.3_0.5um[winds.data.played$PM_0.3_0.5um<0] <- 0
winds.data.played$PM_0.5_1.0um[winds.data.played$PM_0.5_1.0um<0] <- 0
winds.data.played$PM_1.0_3.0um[winds.data.played$PM_1.0_3.0um<0] <- 0
winds.data.played$PM_3.0_5.0um[winds.data.played$PM_3.0_5.0um<0] <- 0
winds.data.played$PM_5.0_10.0um[winds.data.played$PM_5.0_10.0um<0] <- 0


col1 <- match("PM_0.3_0.5um",names(winds.data.played))
col2 <- match("PM_5.0_10.0um",names(winds.data.played))
winds.data.played$PM <- rowSums( winds.data.played[,col1:col2])

percentages <- colSums( winds.data.played[,col1:col2])
percentages_cumsum <- cumsum(percentages/sum(percentages)*100)


winds.data.aggregated = plyr::join(winds.data.played, winds.data.ready, type = "full")
```

```
## Joining by: ID, Instrument, Condition, PM, SPL, PM_0.3_0.5um, PM_0.5_1.0um, PM_1.0_3.0um, PM_3.0_5.0u
```

```
rm(winds.data,
   winds.data.ready,
   winds.data.played,
   winds.data.unplayed)
```

2 Calculate extended measures

```
# shift PM (avoid NaN after log-transform)

winds.data.aggregated$shift.PM <-
                          winds.data.aggregated$PM+
                          winds.data.aggregated$Accuracy.PM

# log-transform PM
winds.data.aggregated$log.PM <- log10(winds.data.aggregated$shift.PM)


# compute acoustic pressure p
winds.data.aggregated$p <- (10^((winds.data.aggregated$SPL)/20))*2e-5
```

```
# compute PM/p
winds.data.aggregated$PM2p <- winds.data.aggregated$PM/
                                  winds.data.aggregated$p
```

3 Rename and re-order entries

```
winds.data.aggregated$Condition[
                winds.data.aggregated$Condition ==
                  'ode_played'] <- 'Ode'
winds.data.aggregated$Condition[
                winds.data.aggregated$Condition ==
                  'orchestral_excerpt_played'] <- 'orchestral_excerpt'

winds.data.aggregated$Condition <- factor(winds.data.aggregated$Condition,
                                levels = c('breathing',
                                           'speaking',
                                           'piano',
                                           'forte',
                                           'Ode',
                                           'orchestral_excerpt'))
winds.data.aggregated$Instrument <- factor(winds.data.aggregated$Instrument,
                                levels = c('flute',
                                           'oboe',
                                           'trumpet',
                                           'clarinette'))

write.csv(winds.data.aggregated,"data_aggregated.csv", row.names = FALSE)
```

4 Descriptive analysis

4.1 Particle distributions

```
size_classes <- c(0.3, 0.5, 1.0, 3.0, 5.0, 10., 25.)
no_of_trials <- 5
dlogD = data.frame(log10(size_classes[2:6])-log10(size_classes[1:5]))
dlogD <- dlogD*length(unique(winds.data.aggregated$ID))*no_of_trials
size_categories <- c('C1','C2','C3','C4','C5') %>% data.frame()
names(dlogD) <- 'dlogD'
names(size_categories) <- 'size_categories'


winds.data.aggregated.classes <- winds.data.aggregated %>% dplyr::select(
   Instrument,
                                      Condition,
                                      PM_0.3_0.5um,
                                      PM_0.5_1.0um,
                                      PM_1.0_3.0um,
                                      PM_3.0_5.0um,
                                      PM_5.0_10.0um
)
```

```
winds.data.aggregated.classes.mean_eb.flute <- calc_mean_eb(list(
                                                    'data'=winds.data.aggregated.classes,
                                                    'size_categories'=size_categories),
                                                    'flute')

winds.data.aggregated.classes.mean_eb.oboe <- calc_mean_eb(list(
                                                    'data'=winds.data.aggregated.classes,
                                                    'size_categories'=size_categories),
                                                    'oboe')

winds.data.aggregated.classes.mean_eb.clarinette <- calc_mean_eb(list(
                                                    'data'=winds.data.aggregated.classes,
                                                    'size_categories'=size_categories),
                                                    'clarinette')

winds.data.aggregated.classes.mean_eb.trumpet <- calc_mean_eb(list(
                                                    'data'=winds.data.aggregated.classes,
                                                    'size_categories'=size_categories),
                                                    'trumpet')


p01 <- plotID_dist_bar(winds.data.aggregated.classes.mean_eb.flute,
                  "Flute","$P_N$ in P/s",
                  "",
                  c(1,10000))
```

```
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```

```
p02 <- plotID_dist_bar(winds.data.aggregated.classes.mean_eb.oboe,
                  "Oboe","",
                  "",
                  c(1,10000))
```

```
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```

```
p03 <- plotID_dist_bar(winds.data.aggregated.classes.mean_eb.clarinette,
                  "Clarinette","$P_N$ in P/s",
                  "Particle diameter $(D_p)$ in $\\mu$m",
                  c(1,10000))
```

```
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```

```
p04 <- plotID_dist_bar(winds.data.aggregated.classes.mean_eb.trumpet,
                  "Trumpet","",
                  "Particle diameter $(D_p)$ in $\\mu$m",
                  c(1,10000))
```

```
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```

```
p0 <- gridExtra::grid.arrange(p01,p02,p03,p04,nrow=2)
```

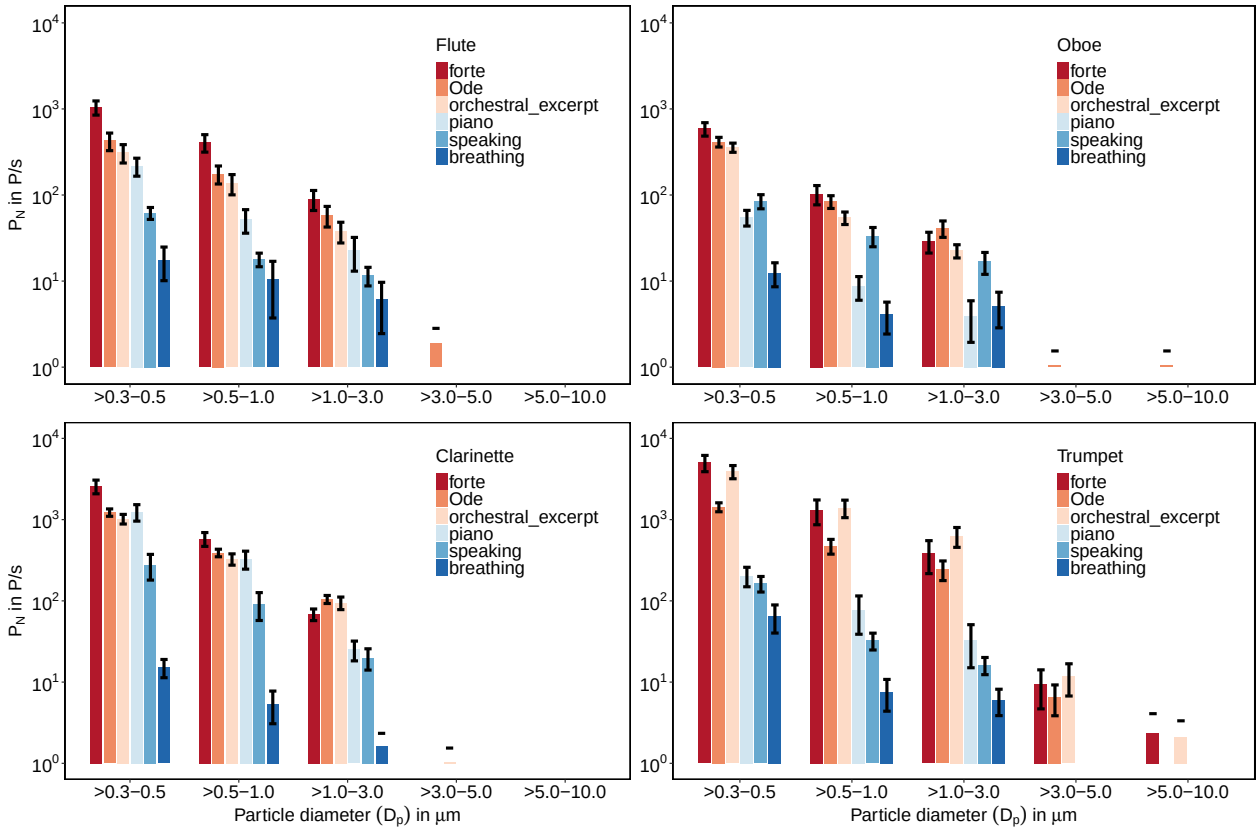

Figure 1: Particle distributions.

```
ggsave(
   "./Schumann_Fig2.pdf",
   device=cairo_pdf,
   plot = p0,
   width = plot_x,
   height = plot_y,
   units = c("in"),
   dpi = 600,
)

ggsave(
   "./Schumann_Fig2.jpg",
   plot = p0,
   width = plot_x,
   height = plot_y,
   units = c("in"),
   dpi = 600,
)
```

4.2 Compute medians (of medians)

```
library(dplyr)
```

```
##
## Attache Paket: 'dplyr'
```

```
## Die folgenden Objekte sind maskiert von 'package:stats':
##
##     filter, lag
```

```
## Die folgenden Objekte sind maskiert von 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
winds.data.aggregated.medians.1 = aggregate(SPL ~
                                            ID+Condition, winds.data.aggregated, median)
winds.data.aggregated.medians.2 = aggregate(Instrument ~
                                            ID+Condition, winds.data.aggregated, first)


winds.data.aggregated.medians.5 = aggregate(PM ~
                                            ID+Condition, winds.data.aggregated, median)
winds.data.aggregated.medians.6 = aggregate(PM2p ~
                                            ID+Condition, winds.data.aggregated, median)

winds.data.aggregated.medians.7 = aggregate(PVR ~
                                            ID+Condition, winds.data.aggregated, median)

winds.data.aggregated.medians = plyr::join(winds.data.aggregated.medians.1,
                                           winds.data.aggregated.medians.2,
                                           type = "inner")
```

```
## Joining by: ID, Condition
```

```
winds.data.aggregated.medians = plyr::join(winds.data.aggregated.medians,
                                           winds.data.aggregated.medians.5,
                                           type = "inner")
```

```
## Joining by: ID, Condition
```

```
winds.data.aggregated.medians = plyr::join(winds.data.aggregated.medians,
                                           winds.data.aggregated.medians.6,
                                           type = "inner")
```

```
## Joining by: ID, Condition
```

```
winds.data.aggregated.medians = plyr::join(winds.data.aggregated.medians,
                                           winds.data.aggregated.medians.7,
                                           type = "inner")
```

```
## Joining by: ID, Condition
```

```
rm(winds.data.aggregated.medians.1,
   winds.data.aggregated.medians.2,
   winds.data.aggregated.medians.5,
   winds.data.aggregated.medians.6,
   winds.data.aggregated.medians.7
   )
```

4.3 Combine group medians with that from professional singers

```
winds.data.aggregated.medians.wisi  <- winds.data.aggregated.medians
winds.data.aggregated.medians.wisi$SPL <- NULL
winds.data.aggregated.medians.wisi$PM2p <- NULL
winds.data.aggregated.medians.wisi <- rbind(winds.data.aggregated.medians.wisi,
                                            singers.data.medians)


winds.data.aggregated.medians.wisi$Condition <-
   as.character(winds.data.aggregated.medians.wisi$Condition)
winds.data.aggregated.medians.wisi$Condition[
      winds.data.aggregated.medians.wisi$Condition ==
                                          'orchestral_excerpt'] <- 'o.e./Song'

winds.data.aggregated.medians.wisi$Instrument <-
   as.character(winds.data.aggregated.medians.wisi$Instrument)
winds.data.aggregated.medians.wisi$Instrument[
      winds.data.aggregated.medians.wisi$Instrument !='Singer' &
           (winds.data.aggregated.medians.wisi$Condition == 'breathing' |
            winds.data.aggregated.medians.wisi$Condition == 'speaking')
                                             ] <- 'All participants'

winds.data.aggregated.medians.wisi$Condition <-
   factor(winds.data.aggregated.medians.wisi$Condition,
                               levels = c('breathing',
                                          'speaking',
                                          'piano',
                                          'forte',
                                          'Ode',
```

```
                                          'o.e./Song'))
winds.data.aggregated.medians.wisi$Instrument <-
   factor(winds.data.aggregated.medians.wisi$Instrument,
                                levels = c('flute',
                                           'oboe',
                                           'trumpet',
                                           'clarinette',
                                           'Singer',
                                           'All participants'))
```

4.4 P_N for all test conditions (incl. data from professional singers)

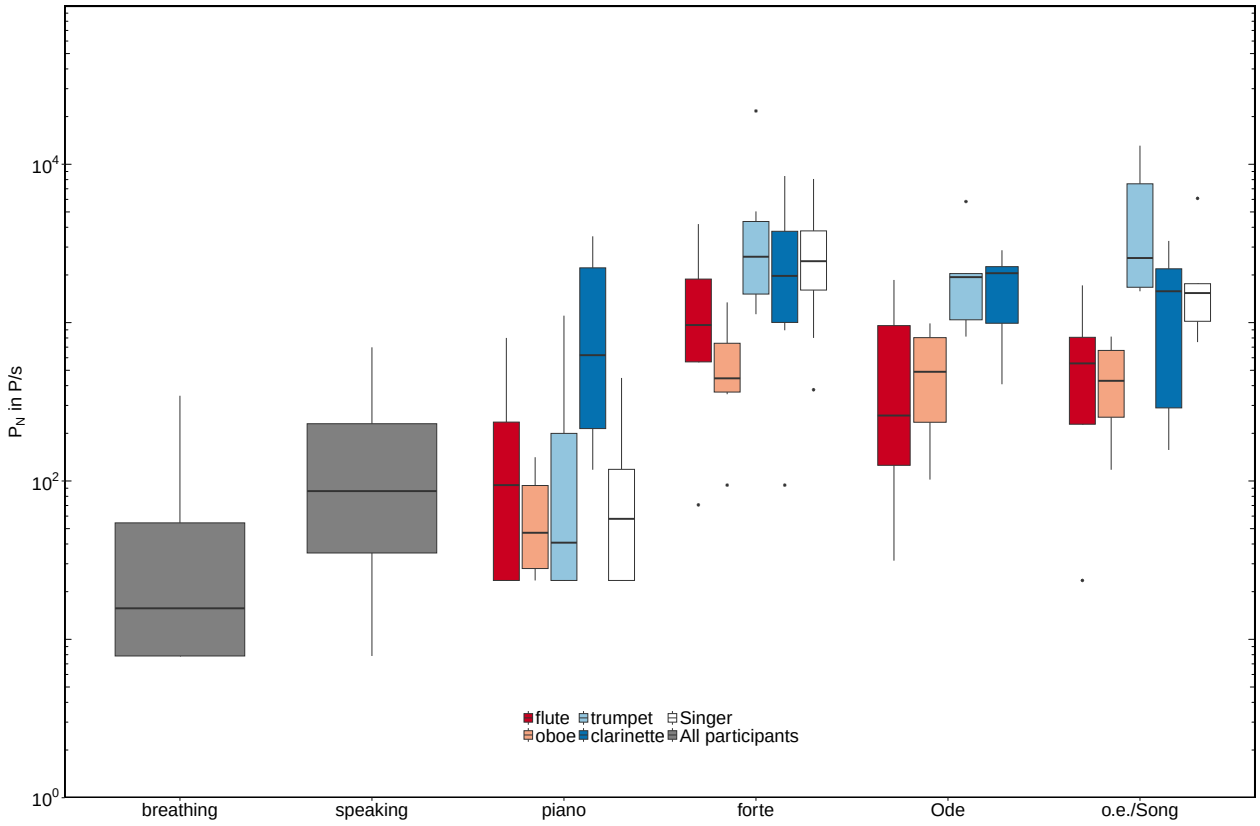

Figure 2: Boxplot of P_N for all participants (based on group medians of (mostly) five replications of each task (respiratoric emitted particles only)). Values for singers were adapted from the supplemental information from Mürbe et al (2021), https://doi.org/10.1038/s41598-021-93281-x

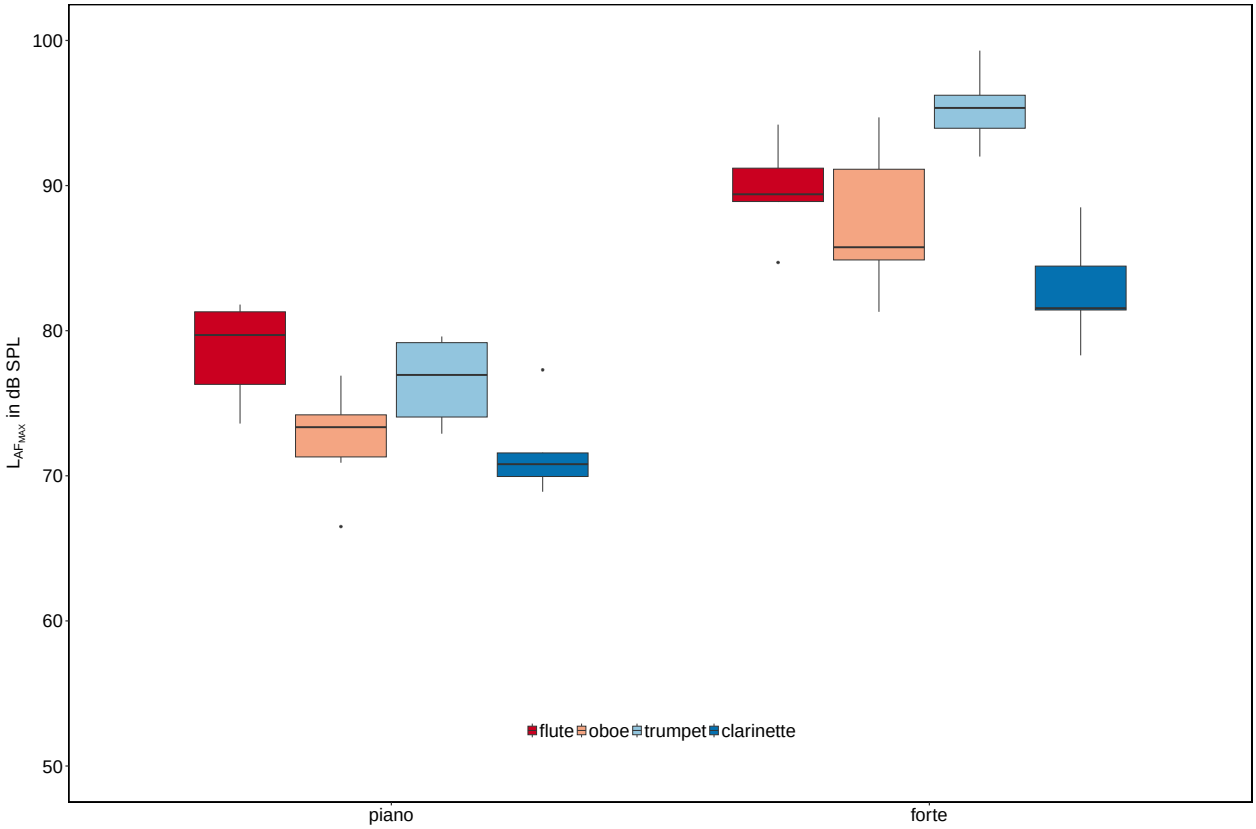

Figure 3: Boxplot of the sound pressure level $L_{AF_{MAX}}$ for all participants (based on medians of five replications of each task).

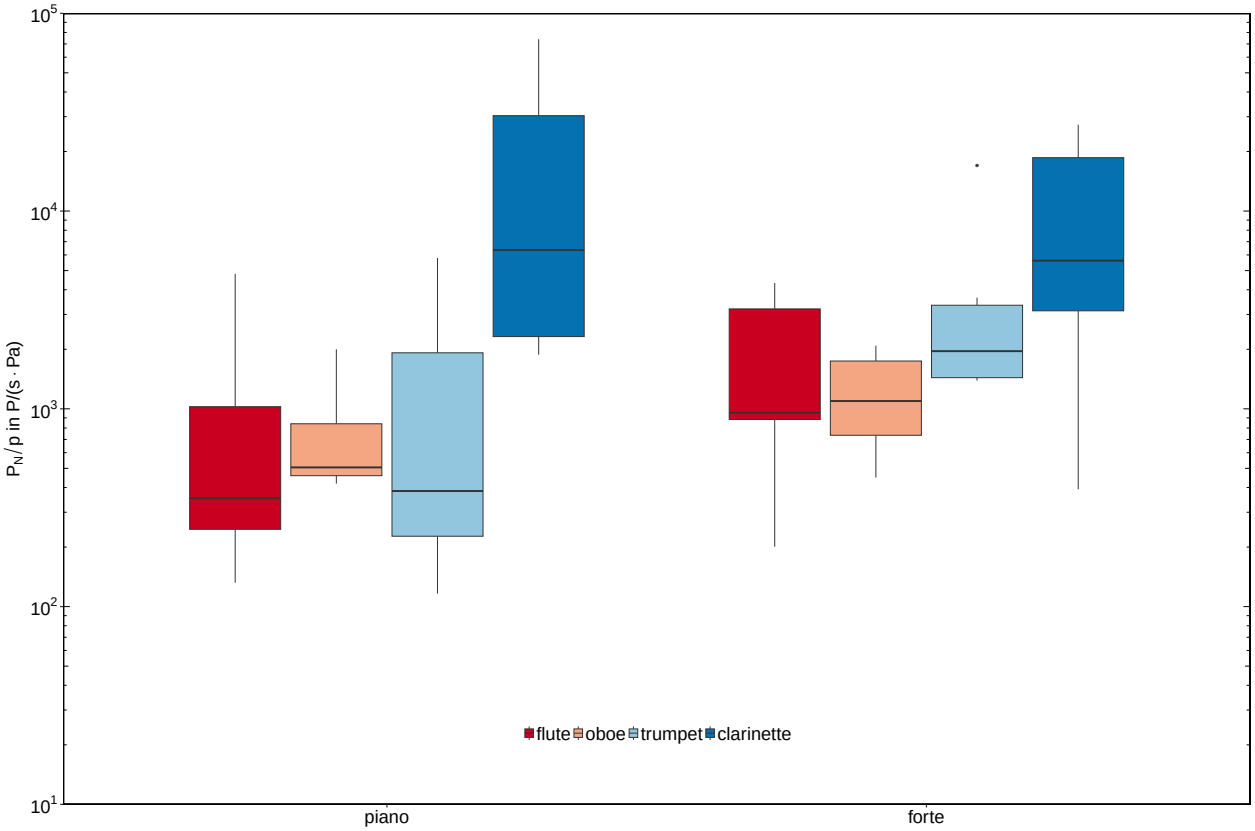

Figure 4: Boxplot of P_N related to the acoustic pressure $p = 10^{L_{AF_{MAX}}/20} \cdot 2 \times 10^{-5}$ Pa for all participants (based on medians of five replications of each task).

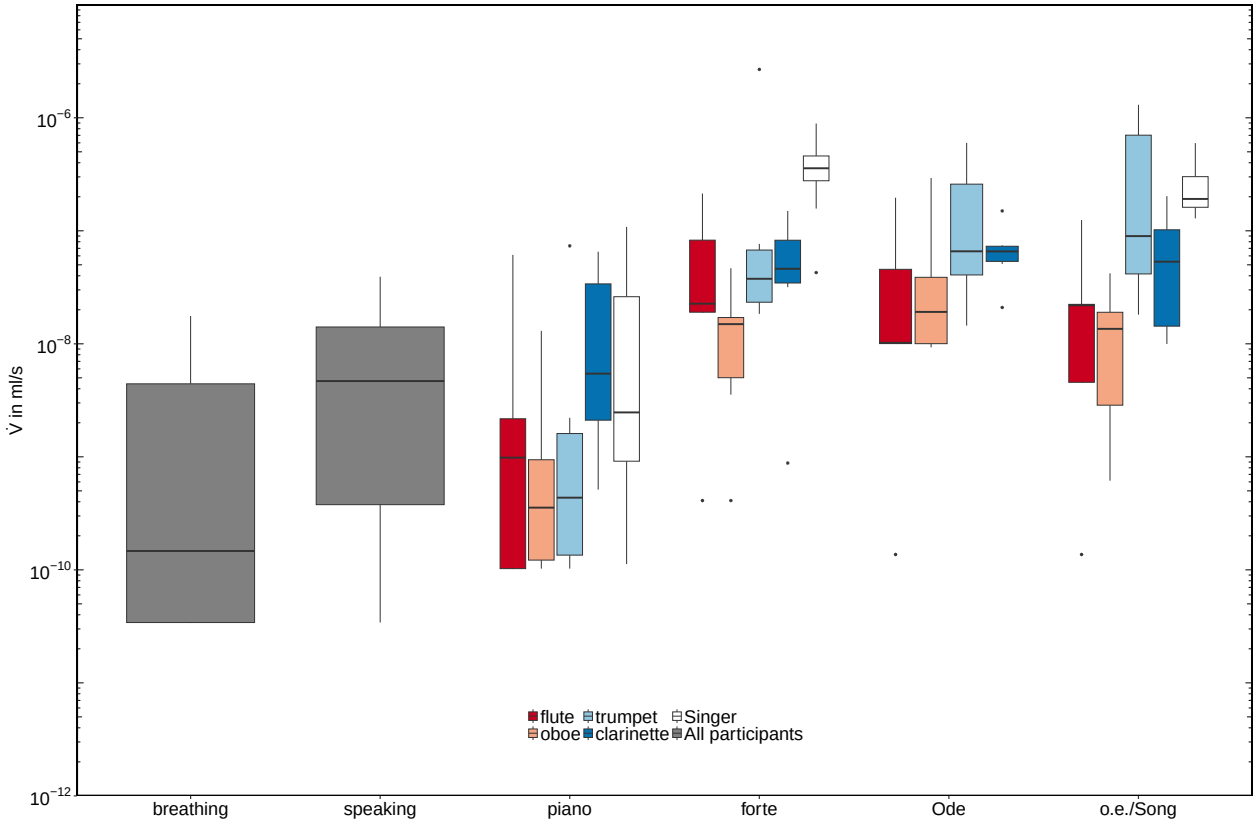

Figure 5: Boxplot of Particle Volume Rate $\dot{V}$ for all participants (based on group medians of (mostly) five replications of each task (respiratoric emitted particles only)). Values for singers were adapted from the supplemental information from Mürbe et al (2021), https://doi.org/10.1038/s41598-021-93281-x

4.5 Sound pressure level

4.6 P_N/p

4.7 Particle volume rate for all test conditions (incl. data from professional singers)

5 Confirmatory analysis (Linear-Mixed-Effect-Modeling)

5.1 Experiment 1: Piano and forte

```
model.winds.data.aggregated.log.PM.pifo <-
                  lmerTest::lmer(log.PM~Instrument+Condition
                   +(1|ID),
                  winds.data.aggregated.pifo,
                  REML=TRUE)
jtools::summ(model.winds.data.aggregated.log.PM.pifo, digits = 3)
```

Observations	230
Dependent variable	log.PM
Type	Mixed effects linear regression

AIC	267.317
BIC	291.384
Pseudo-R^2 (fixed effects)	0.431
Pseudo-R^2 (total)	0.795

Fixed Effects					
	Est.	S.E.	t val.	d.f.	p
(Intercept)	2.107	0.221	9.515	19.441	0.000
Instrumentoboe	-0.265	0.298	-0.888	19.000	0.386
Instrumenttrumpet	0.286	0.298	0.958	19.000	0.350
Instrumentclarinette	0.488	0.298	1.637	19.000	0.118
Conditionforte	0.862	0.047	18.220	206.000	0.000

p values calculated using Satterthwaite d.f.

Random Effects		
Group	Parameter	Std. Dev.
ID	(Intercept)	0.479
Residual		0.359

Grouping Variables		
Group	# groups	ICC
ID	23	0.641

5.2 Experiment 2: Playing orchestral excerpt and 'Ode to the joy'

```
model.winds.data.aggregated.log.PM.pl <-
                  lmerTest::lmer(log.PM~Instrument+Condition
                   +(1|ID),
                  winds.data.aggregated.pl,
                  REML=TRUE)

jtools::summ(model.winds.data.aggregated.log.PM.pl, digits = 3)
```

Observations	230
Dependent variable	log.PM
Type	Mixed effects linear regression

AIC	164.806
BIC	188.873
Pseudo-R^2 (fixed effects)	0.364
Pseudo-R^2 (total)	0.842

Fixed Effects

	Est.	S.E.	t val.	d.f.	p
(Intercept)	2.285	0.222	10.293	19.265	0.000
Instrumentoboe	0.309	0.300	1.030	19.000	0.316
Instrumenttrumpet	1.125	0.300	3.754	19.000	0.001
Instrumentclarinette	0.772	0.300	2.578	19.000	0.018
Conditionorchestral_excerpt	-0.040	0.037	-1.085	206.000	0.279

p values calculated using Satterthwaite d.f.

Random Effects

Group	Parameter	Std. Dev.
ID	(Intercept)	0.487
Residual		0.280

Grouping Variables

Group	# groups	ICC
ID	23	0.752

Table 1: P_N - Values for piano (left) and forte (right) in P/s

ID	Min	Median	Max
flute	23.54	94.170	800.47
oboe	23.54	47.090	141.26
trumpet	23.54	47.085	1106.53
clarinette	117.72	647.435	3507.92

ID	Min	Median	Max
flute	70.63	965.270	4190.67
oboe	94.17	447.320	1341.96
trumpet	1130.07	2613.285	21706.75
clarinette	94.17	2083.565	8428.43

Table 2: P_N - Values for Ode to joy (left) and orchestral excerpt (right) in P/s

ID	Min	Median	Max
flute	31.39	258.970	1859.91
oboe	102.03	490.485	988.80
trumpet	816.17	1938.380	5815.15
clarinette	408.08	2052.175	2864.41

ID	Min	Median	Max
flute	0.00	486.560	1718.65
oboe	117.71	431.635	816.16
trumpet	1577.39	2648.595	13105.67
clarinette	156.96	1640.170	3280.34

Table 3: P_N - Values for breathing (left) and speaking (right) in P/s

ID	Min	Median	Max
All participants	7.85	15.7	345.3

ID	Min	Median	Max
All participants	7.85	86.32	698.45

Table 4: P_N/p - Values for piano (left) and forte (right) in P/(s·Pa)

ID	Min	Median	Max
flute	132.062	353.150	4811.883
oboe	418.696	507.995	1999.527
trumpet	116.353	396.100	5793.396
clarinette	1879.270	8123.827	73963.880

ID	Min	Median	Max
flute	200.890	956.353	4328.343
oboe	449.658	1096.508	2084.889
trumpet	1389.045	2030.090	17004.537
clarinette	391.636	6154.405	27305.710

6 Appendix

6.1 Minimum, maximum, and median values for all conditions (based on group medians)

6.1.1 P_N values

6.1.2 P_N/p values

6.2 Packages used for this study

```
sessionInfo()
```

```
## R version 4.3.1 (2023-06-16)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.6 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0
##
## locale:
##  [1] LC_CTYPE=de_DE.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_GB.UTF-8        LC_COLLATE=de_DE.UTF-8
##  [5] LC_MONETARY=en_GB.UTF-8    LC_MESSAGES=de_DE.UTF-8
##  [7] LC_PAPER=en_GB.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
##
## time zone: Europe/Berlin
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] dplyr_1.1.2     rstatix_0.7.2   xtable_1.8-4    forcats_1.0.0
## [5] latex2exp_0.9.6 ggplot2_3.4.2   scales_1.2.1    tidyr_1.3.0
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.2.0      viridisLite_0.4.2     farver_2.1.1
##  [4] fastmap_1.1.1         janitor_2.2.0         jtools_2.2.2
##  [7] promises_1.2.0.1      digest_0.6.33         timechange_0.2.0
## [10] mime_0.12             lifecycle_1.0.3       ellipsis_0.3.2
## [13] processx_3.8.2        magrittr_2.0.3        compiler_4.3.1
## [16] rlang_1.1.1           tools_4.3.1           plotrix_3.8-2
## [19] utf8_1.2.3            yaml_2.3.7            data.table_1.14.8
## [22] knitr_1.43            prettyunits_1.1.1     labeling_0.4.2
## [25] htmlwidgets_1.6.2     pkgbuild_1.4.2        curl_5.0.1
## [28] xml2_1.3.5            plyr_1.8.8            pkgload_1.3.2.1
## [31] abind_1.4-5           miniUI_0.1.1.1        numDeriv_2016.8-1.1
## [34] withr_2.5.0           purrr_1.0.1           grid_4.3.1
## [37] fansi_1.0.4           urlchecker_1.0.1      profvis_0.3.8
## [40] colorspace_2.1-0      MASS_7.3-60           cli_3.6.1
## [43] rmarkdown_2.23        crayon_1.5.2          ragg_1.2.5
```

```
## [46] generics_0.1.3       remotes_2.4.2.1      rstudioapi_0.15.0
## [49] httr_1.4.6           sessioninfo_1.2.2    minqa_1.2.5
## [52] cachem_1.0.8         pander_0.6.5         stringr_1.5.0
## [55] splines_4.3.1        rvest_1.0.3          vctrs_0.6.3
## [58] devtools_2.4.5       webshot_0.5.5        Matrix_1.6-0
## [61] boot_1.3-28          carData_3.0-5        car_3.1-2
## [64] callr_3.7.3          systemfonts_1.0.4    glue_1.6.2
## [67] nloptr_2.0.3         ps_1.7.5             lubridate_1.9.2
## [70] stringi_1.7.12       gtable_0.3.3         later_1.3.0
## [73] lme4_1.1-34          lmerTest_3.1-3       munsell_0.5.0
## [76] tibble_3.2.1         pillar_1.9.0         htmltools_0.5.5
## [79] R6_2.5.1             textshaping_0.3.6    kableExtra_1.3.4.9000
## [82] evaluate_0.21        shiny_1.7.3          lattice_0.21-8
## [85] highr_0.10           backports_1.4.1      memoise_2.0.1
## [88] broom_1.0.5          snakecase_0.11.0     httpuv_1.6.6
## [91] Rcpp_1.0.11          svglite_2.1.1        gridExtra_2.3
## [94] nlme_3.1-162         xfun_0.39            fs_1.6.3
## [97] usethis_2.2.2        pkgconfig_2.0.3
```