Data inspection and analysis for: Thermal Comfort and Perception of Different Materials for Table Tops

Mike Burnard, Nastja Podrekar, Dean Lipovac

January 6, 2020

Contents

Introduction

This analysis investigates the temperature difference of materials before and after use. Our goal is to identify which materials are most suitable for everyday use as table top materials that could be used, on desks, conference tables, and similar surfaces. We base our analysis on objectively measured thermal properties and temperature data along with subjective evaluation by human subjects.

Data on thermal properties

The data include:

- Subject (numeric): the person that participated in the experiment
- Material (string): a code for the material of the table top in the experiment
- Stage (string): a string describing the part of the test the measurements pertain to (pre or post)
- Hand (string):the section of the table where the reading was taken from
- Reading (string): which part of the arm does the reading pertain to
- Value (numeric): the temperature in celsius.

We add name_nice which has human readable names of the material.

Reading the raw data

Reading individual excel files and merging them into a single data set, on which will base our analysis. Produces ~ 7 million records. After reading the data from many files, we write it to a single csv file to simplify sharing and importing the data. Code Not Run.

```r
# DO NOT RUN ... takes for ever.
files <- list.files("data/arms")
d.arms <- tibble(material = as.character(), subject = as.numeric(),
                 stage = as.character(), hand = as.character(),
                 reading = as.character(), value=as.numeric())

for(i in seq_along(files)) {
  f.sheets <- readxl::excel_sheets(paste("data/arms/", files[i], sep=""))
  f.info <- str_split(str_sub(files[i],1, -6), "_")
  for( j in seq_along(f.sheets)) {
    d.sheet <- readxl::read_excel(paste("data/arms/", files[i], sep=""), f.sheets[j], skip=4)
    if(f.info[[1]][3] == "pre") {
      names(d.sheet) <- c("reading", "value")
      d.sheet <- d.sheet %>% select("value") %>%
        mutate(material = f.info[[1]][1],
               subject = as.numeric(f.info[[1]][2]),
               stage = f.info[[1]][3],
               hand = "none",
               reading = f.sheets[j],
               value = value)
    }
    if(f.info[[1]][3] == "post") {
      names(d.sheet) <- c("reading", "left", "right")
      d.sheet <- d.sheet %>% select(left, right) %>%
        pivot_longer(cols = c("left", "right"),
                     names_to = "hand",
                     values_to = "value") %>%
```

```
        mutate(material = f.info[[1]][1],
               subject = as.numeric(f.info[[1]][2]),
               stage = f.info[[1]][3],
               hand = hand,
               reading = f.sheets[j],
               value = value)
    }

    d.arms <- bind_rows(d.arms, d.sheet)
  }
}
d.arms <- d.arms %>% drop_na()
write_csv(d.arms, "data/arms-complete.csv")
```

Raw data, simplified

Read the raw data from the new file and create an modified version for analysis and visualisation. We also add nice names to the data for plotting and reporting.

```
# big file! takes some time to load
d.arms <- read_csv("data/arms-complete.csv", col_types = cols())
d.arms <- d.arms %>% filter(reading == "Whole region") %>%
  mutate(material = case_when(material == "i" ~ "I",
                              material == "j" ~ "J",
                              TRUE ~ material),
         sub.fac = paste("sub",as.character(subject), sep="-"))
mats <- read_csv2("data/material_codes.csv", col_types = cols()) %>%
  mutate(name_nice = c("Oak Untreated", "Oak Veneer", "MTFC", "Oak Oiled",
                       "Spruce Untreated", "IW Laminate", "Spruce Oiled",
                       "Spruce Lacquered", "Oak Lacquered", "Glass"),
         code = str_to_upper(code))
```

```
## Using ',' as decimal and '.' as grouping mark. Use read_delim() for more control.
```

```
d.arms <- full_join(d.arms, mats, by=c("material" = "code"))
```

Basic analysis and visualisation

Raw data inspection

First, we examine the raw data grouped only by stage and material, then plot the raw data. We need to reduce the data to something more manageable.

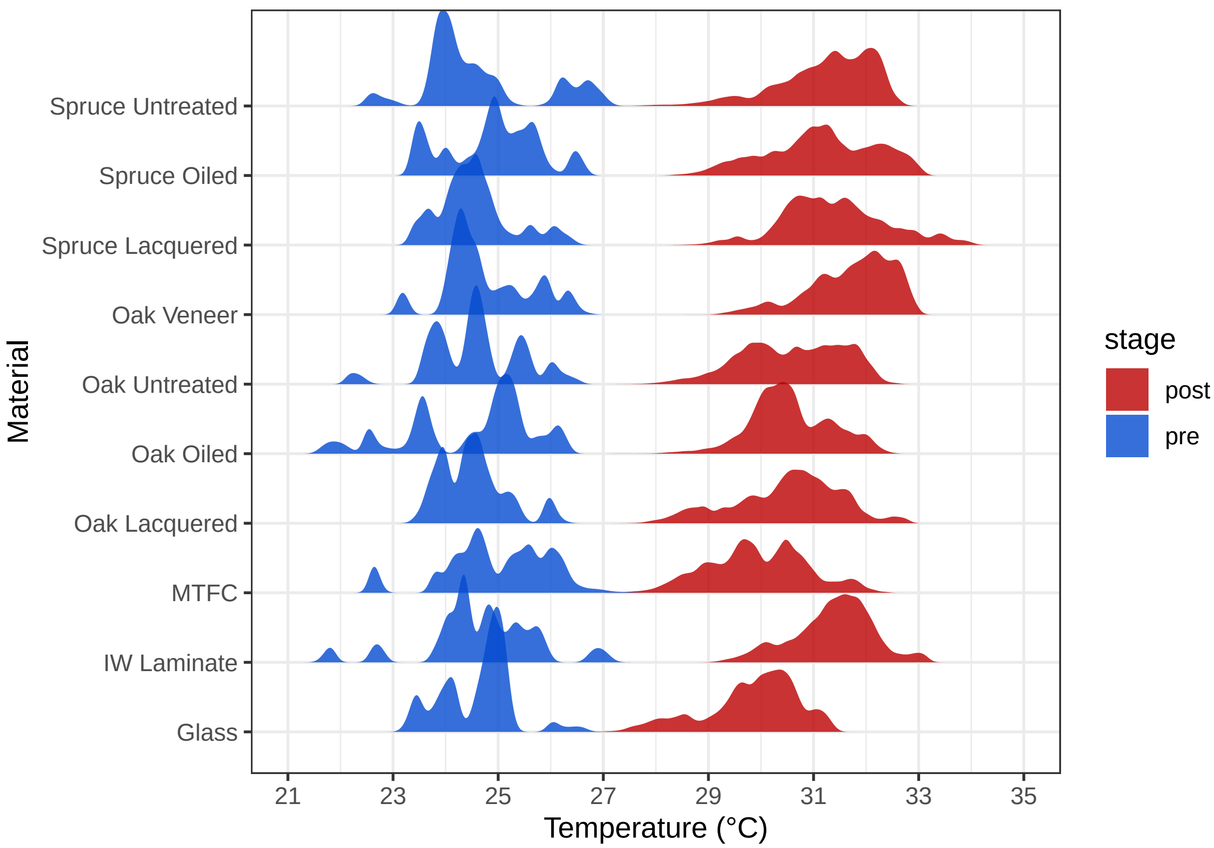
```
d.arms %>% group_by(material, stage) %>%
  summarise(mu=mean(value), med = median(value), sd=sd(value))
```

```
## # A tibble: 20 x 5
## # Groups:   material [10]
##    material stage    mu   med    sd
##    <chr>    <chr> <dbl> <dbl> <dbl>
##  1 A        post   30.6  30.6 0.958
##  2 A        pre    24.6  24.6 0.867
```

```
##   3 B         post   31.7   31.9 0.790
##   4 B         pre    24.8   24.6 0.810
##   5 C         post   29.9   29.9 0.930
##   6 C         pre    25.0   25.0 0.975
##   7 D         post   30.6   30.5 0.767
##   8 D         pre    24.5   24.9 1.19
##   9 E         post   31.2   31.4 0.905
##  10 E         pre    24.6   24.3 1.10
##  11 F         post   31.4   31.5 0.757
##  12 F         pre    24.7   24.7 1.06
##  13 G         post   31.1   31.2 1.02
##  14 G         pre    24.9   25.0 0.872
##  15 H         post   31.4   31.3 0.981
##  16 H         pre    24.6   24.5 0.691
##  17 I         post   30.5   30.6 1.01
##  18 I         pre    24.5   24.5 0.623
##  19 J         post   29.8   30.0 0.880
##  20 J         pre    24.6   24.8 0.673
```

```r
ggplot(data = d.arms, aes(x=value, y=name_nice, fill=stage)) +
  geom_density_ridges(alpha = 0.8, colour=NA) +
  scale_fill_manual(values=c("#BC0000", "#044BD1")) +
  scale_x_continuous(limits=c(21,35), breaks=seq(21,35,2)) +
  labs(y="Material",
       x="Temperature (\u00B0C)")
```

```
## Picking joint bandwidth of 0.0852
```

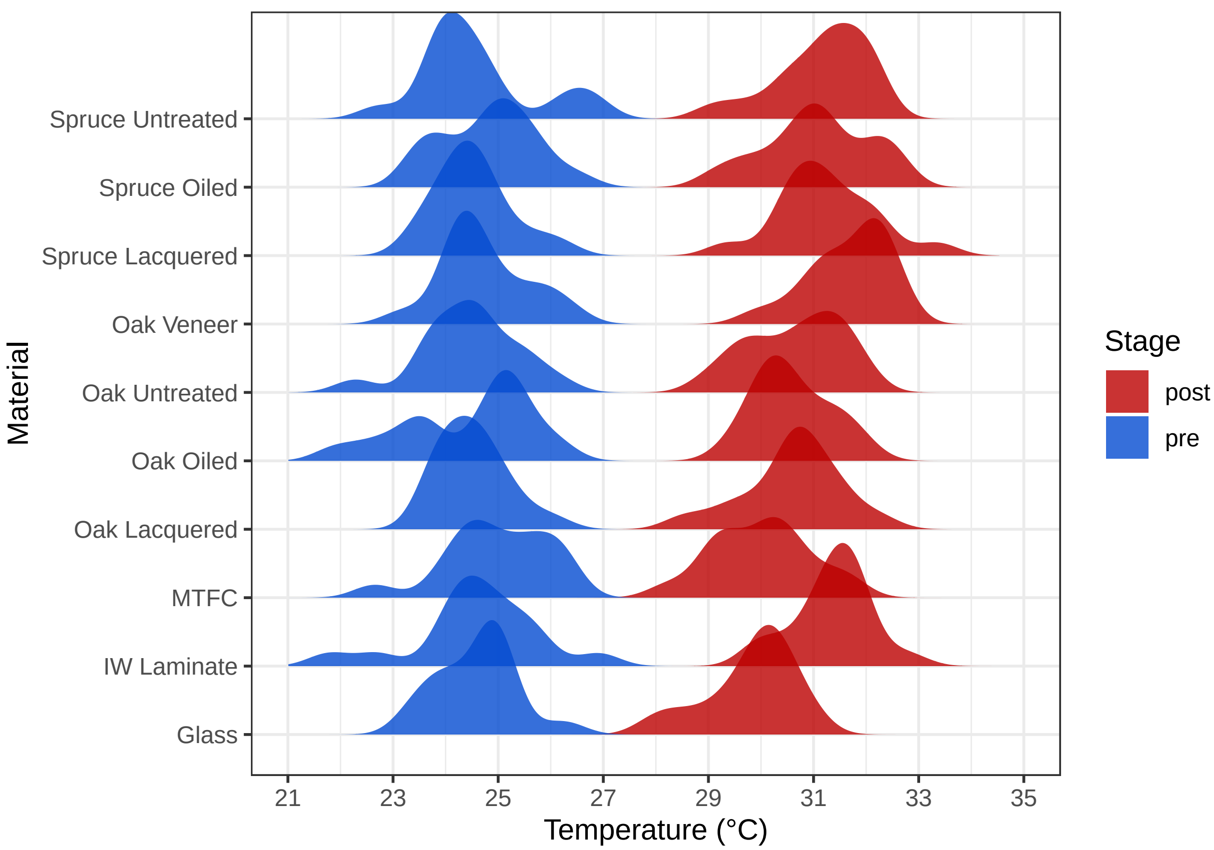


Reduced data

The reduced data set is the average temperature for each subject on each material. This makes the data more manageable for interpreation, visualisation, and modeling.

```
d.sm.arms <- d.arms %>% group_by(name_nice, stage, sub.fac) %>%
  summarise(temp.mu = mean(value), temp.min=min(value),
            temp.max = max(value), temp.sd = sd(value)) %>%
  mutate(range = temp.max - temp.min)

d.sm.sum <- d.arms %>% group_by(name_nice, stage) %>%
  summarise(temp.mu = mean(value), temp.min=min(value),
            temp.max = max(value), temp.sd = sd(value)) %>%
  mutate(range = temp.max - temp.min)
write_excel_csv(d.sm.sum, "Observed_Summary.csv")


ggplot(data = d.sm.arms, aes(x=temp.mu, y=name_nice, fill=stage)) +
  geom_density_ridges(alpha = 0.8, colour=NA) +
  scale_fill_manual(values=c("#BC0000", "#044BD1"), name="Stage") +
  scale_x_continuous(limits=c(21,35), breaks=seq(21,35,2)) +
  labs(y="Material",
       x="Temperature (\u00B0C)")
```

```
## Picking joint bandwidth of 0.396
```




```
#ggsave("figs/pre-post_ridges.pdf", device=cairo_pdf(), height=5, width=9, units="in")
#ggsave("figs/pre-post_ridges.png", device=png(), height=5, width=9, units="in")
```

Model fitting

The data include repeated measures (for each subject), so we fit a linear mixed model to the data including the repeated measure by subject. We examine each "stage" seperately.

Pre / Before stage

First we create a dataset with only the "pre" stage, then fit a model using that data. This model doesn't tell us much, but we don't expect it to. We would be surprised to see notable differences here.

```
d.smb.arms <- d.sm.arms %>% filter(stage == "pre")
arms.lmb <- lmer(data=d.smb.arms, temp.mu~name_nice+(1|sub.fac))
summary(arms.lmb)
```

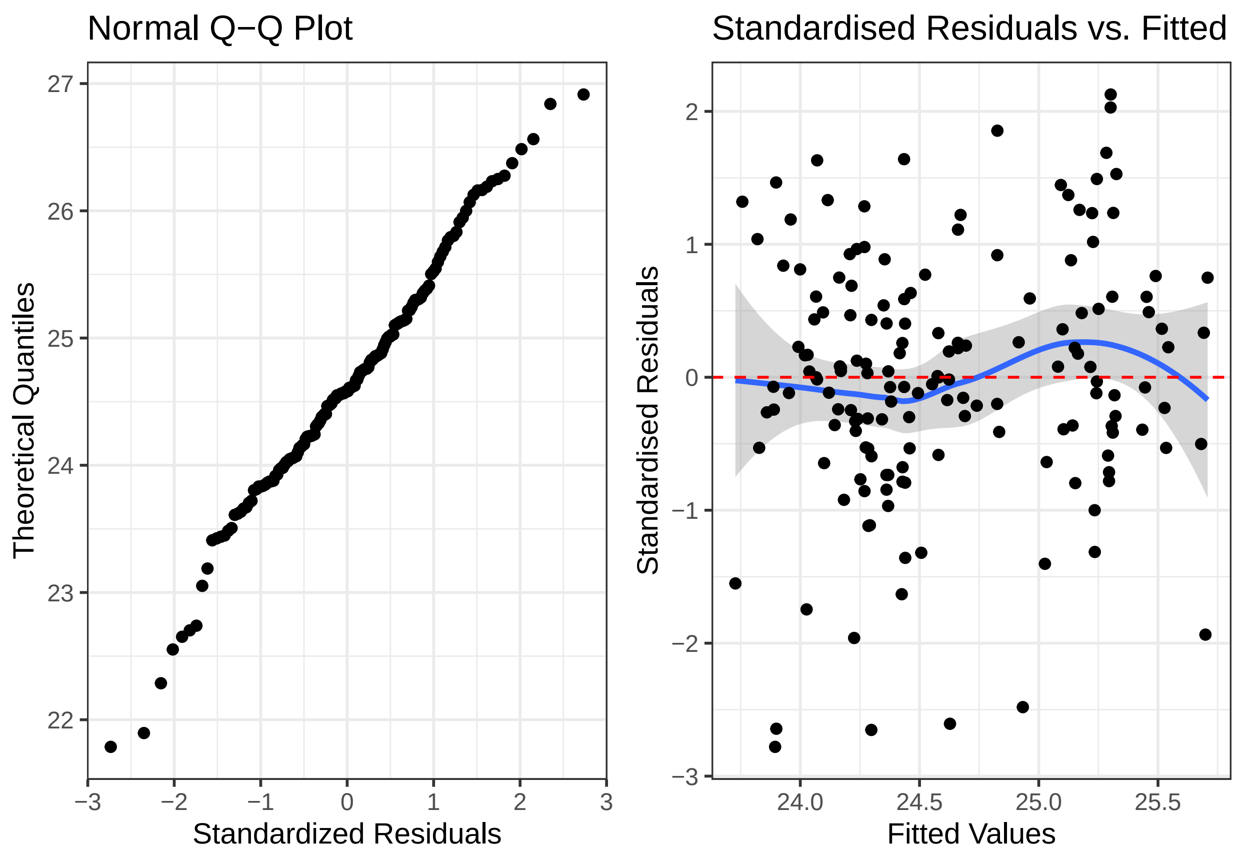
```
## Linear mixed model fit by REML ['lmerMod']
## Formula: temp.mu ~ name_nice + (1 | sub.fac)
##    Data: d.smb.arms
##
## REML criterion at convergence: 398
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.77981 -0.50822 -0.00043  0.58906  2.12682
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  sub.fac  (Intercept) 0.3061   0.5533
##  Residual             0.5748   0.7581
## Number of obs: 160, groups:  sub.fac, 16
##
## Fixed effects:
##                              Estimate Std. Error t value
## (Intercept)                 24.591913   0.234637 104.808
## name_niceIW Laminate        -0.004237   0.268038  -0.016
## name_niceMTFC                0.386357   0.268038   1.441
## name_niceOak Lacquered      -0.078183   0.268038  -0.292
## name_niceOak Oiled          -0.170762   0.268038  -0.637
## name_niceOak Untreated      -0.141562   0.268038  -0.528
## name_niceOak Veneer          0.139664   0.268038   0.521
## name_niceSpruce Lacquered -0.070924   0.268038  -0.265
## name_niceSpruce Oiled        0.221544   0.268038   0.827
## name_niceSpruce Untreated -0.011261   0.268038  -0.042
##
## Correlation of Fixed Effects:
##             (Intr) nm_IWL n_MTFC nm_nOL nm_nOO nm_nOU nm_nOV nm_nSL nm_nSO
## nm_ncIWLmnt -0.571
## name_ncMTFC -0.571  0.500
## nm_ncOkLcqr -0.571  0.500  0.500
## nam_ncOkOld -0.571  0.500  0.500  0.500
## nm_ncOkUntr -0.571  0.500  0.500  0.500  0.500
## nam_ncOkVnr -0.571  0.500  0.500  0.500  0.500  0.500
## nm_ncSprcLc -0.571  0.500  0.500  0.500  0.500  0.500  0.500
## nm_ncSprcOl -0.571  0.500  0.500  0.500  0.500  0.500  0.500  0.500
```

```
## nm_ncSprcUn -0.571  0.500  0.500  0.500  0.500  0.500  0.500  0.500  0.500
```

Inspect model fit (pre)

Fit isn't perfect, but is acceptable to continue.

```
#fortify(arms.lmb)
p1 <-ggplot(fortify(arms.lmb), aes(sample=temp.mu)) +
  stat_qq() +
  labs(title="Normal Q-Q Plot", x="Standardized Residuals", y="Theoretical Quantiles")
#residuals vs. fitted
p2 <- ggplot(fortify(arms.lmb), aes(x=.fitted, y=.scresid)) +
  geom_smooth(method="loess") +
  geom_point() +
  geom_hline(yintercept=0, col="red", linetype=2) +
  labs(title="Standardised Residuals vs. Fitted",
y="Standardised Residuals", x="Fitted Values")

grid.arrange(p1, p2, ncol=2)
```



Post / After stage

```
d.sma.arms <- d.sm.arms %>% filter(stage == "post")
arms.lma <- lmer(data=d.sma.arms, temp.mu~name_nice+(1|sub.fac))
summary(arms.lma)
```

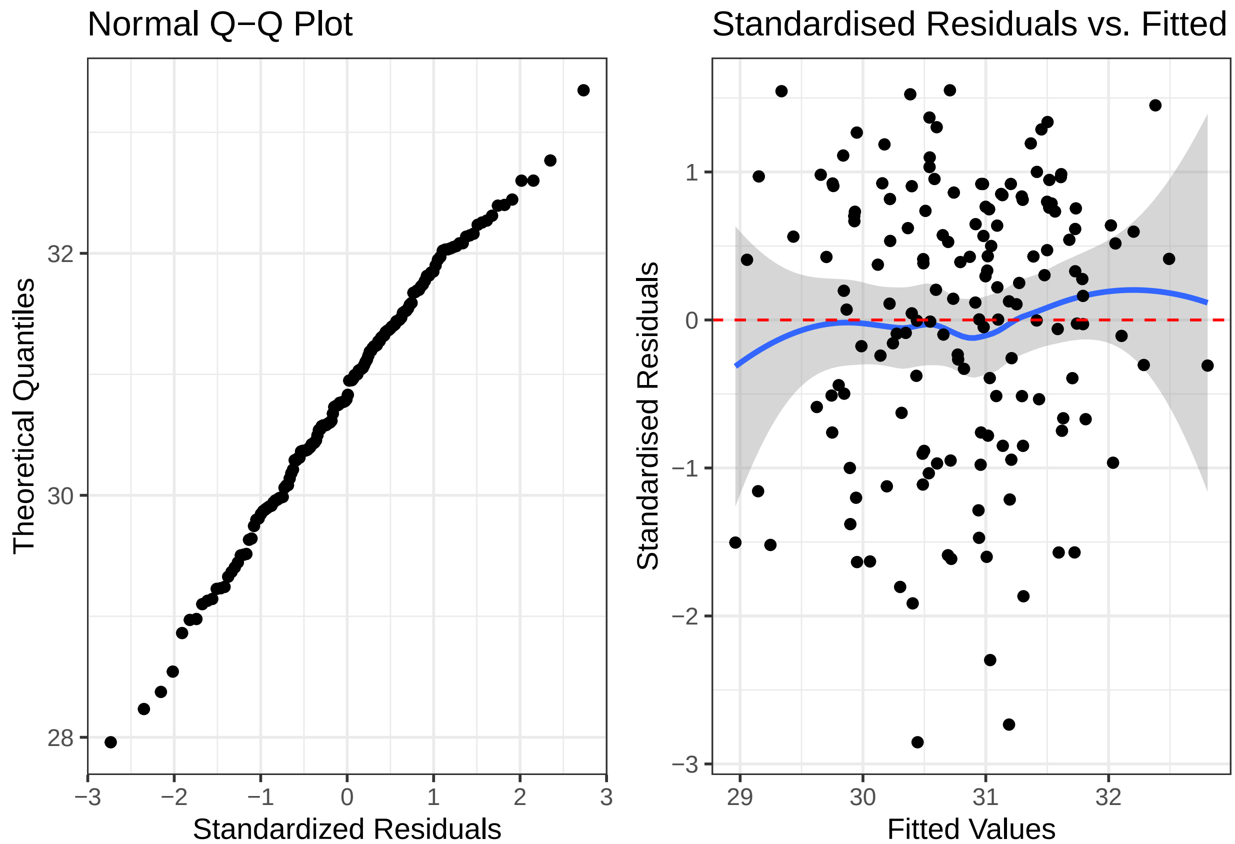
```
## Linear mixed model fit by REML ['lmerMod']
## Formula: temp.mu ~ name_nice + (1 | sub.fac)
##    Data: d.sma.arms
##
## REML criterion at convergence: 362.1
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.8533 -0.6367  0.1530  0.7398  1.5520
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  sub.fac  (Intercept) 0.2938   0.5420
##  Residual             0.4442   0.6665
## Number of obs: 160, groups:  sub.fac, 16
##
## Fixed effects:
##                            Estimate Std. Error t value
## (Intercept)                 29.8084     0.2148 138.795
## name_niceIW Laminate         1.5473     0.2356   6.566
## name_niceMTFC                0.1907     0.2356   0.809
## name_niceOak Lacquered       0.7882     0.2356   3.345
## name_niceOak Oiled           0.7830     0.2356   3.323
## name_niceOak Untreated       0.8411     0.2356   3.569
## name_niceOak Veneer          1.8609     0.2356   7.897
## name_niceSpruce Lacquered    1.4360     0.2356   6.094
## name_niceSpruce Oiled        1.2584     0.2356   5.340
## name_niceSpruce Untreated    1.3412     0.2356   5.692
##
## Correlation of Fixed Effects:
##             (Intr) nm_IWL n_MTFC nm_nOL nm_nOO nm_nOU nm_nOV nm_nSL nm_nSO
## nm_ncIWLmnt -0.549
## name_ncMTFC -0.549  0.500
## nm_ncOkLcqr -0.549  0.500  0.500
## nam_ncOkOld -0.549  0.500  0.500  0.500
## nm_ncOkUntr -0.549  0.500  0.500  0.500  0.500
## nam_ncOkVnr -0.549  0.500  0.500  0.500  0.500  0.500
## nm_ncSprcLc -0.549  0.500  0.500  0.500  0.500  0.500  0.500
## nm_ncSprcOl -0.549  0.500  0.500  0.500  0.500  0.500  0.500  0.500
## nm_ncSprcUn -0.549  0.500  0.500  0.500  0.500  0.500  0.500  0.500  0.500
```

Inspect model fit (post)

Fit is acceptable.

```
p1 <-ggplot(fortify(arms.lma), aes(sample=temp.mu)) +
  stat_qq() +
  labs(title="Normal Q-Q Plot", x="Standardized Residuals", y="Theoretical Quantiles")
#residuals vs. fitted
p2 <- ggplot(fortify(arms.lma), aes(x=.fitted, y=.scresid)) +
  geom_smooth(method="loess") +
  geom_point() +
  geom_hline(yintercept=0, col="red", linetype=2) +
  labs(title="Standardised Residuals vs. Fitted",
```

```r
y="Standardised Residuals", x="Fitted Values")

grid.arrange(p1, p2, ncol=2)
```



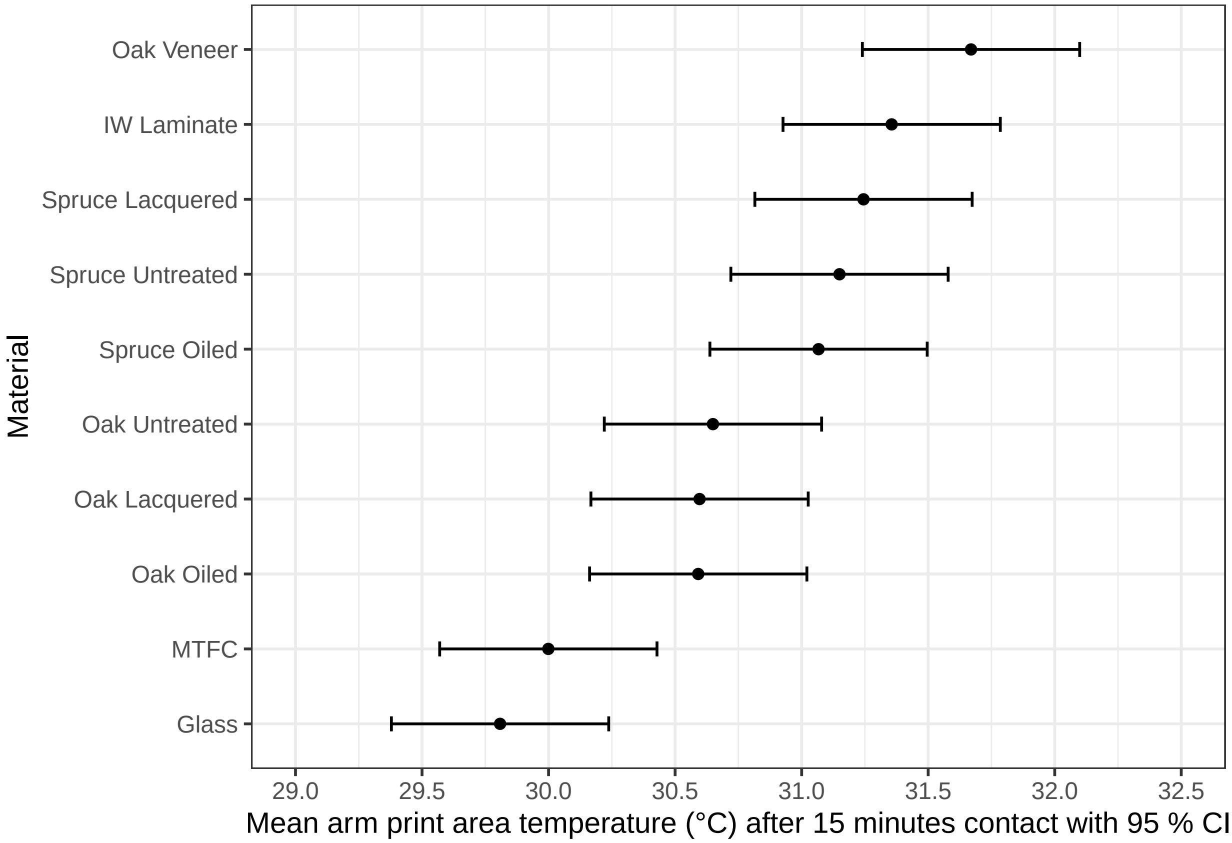
Summary of model data (post)

```r
arms.ema <- as_tibble(emmeans(arms.lma, ~ name_nice)) %>%
  mutate(name_nice = fct_reorder(name_nice, upper.CL))
arms.ema %>% arrange(upper.CL)
```

```
## # A tibble: 10 x 6
##    name_nice         emmean    SE    df lower.CL upper.CL
##    <fct>              <dbl> <dbl> <dbl>    <dbl>    <dbl>
##  1 Glass               29.8 0.215  61.8     29.4     30.2
##  2 MTFC                30.0 0.215  61.8     29.6     30.4
##  3 Oak Oiled           30.6 0.215  61.8     30.2     31.0
##  4 Oak Lacquered       30.6 0.215  61.8     30.2     31.0
##  5 Oak Untreated       30.6 0.215  61.8     30.2     31.1
##  6 Spruce Oiled        31.1 0.215  61.8     30.6     31.5
##  7 Spruce Untreated    31.1 0.215  61.8     30.7     31.6
##  8 Spruce Lacquered    31.2 0.215  61.8     30.8     31.7
##  9 IW Laminate         31.4 0.215  61.8     30.9     31.8
## 10 Oak Veneer          31.7 0.215  61.8     31.2     32.1
```

```r
ggplot(data = arms.ema, aes(x=name_nice, y=emmean,
                            ymin=lower.CL, ymax=upper.CL)) +
```

```r
  geom_point() +
  geom_errorbar(width=0.2) +
  coord_flip() +
  scale_y_continuous(limits=c(29, 32.5), breaks=seq(29,32.5,0.5)) +
  labs(y="Mean arm print area temperature (\u00B0C) after 15 minutes contact with 95 % CI",
       x="Material")
```




```r
#ggsave("figs/post-stage_temp.pdf", device=cairo_pdf(), height=5, width=9, units="in")
```

Contrasts & Pairwise comparisons (post)

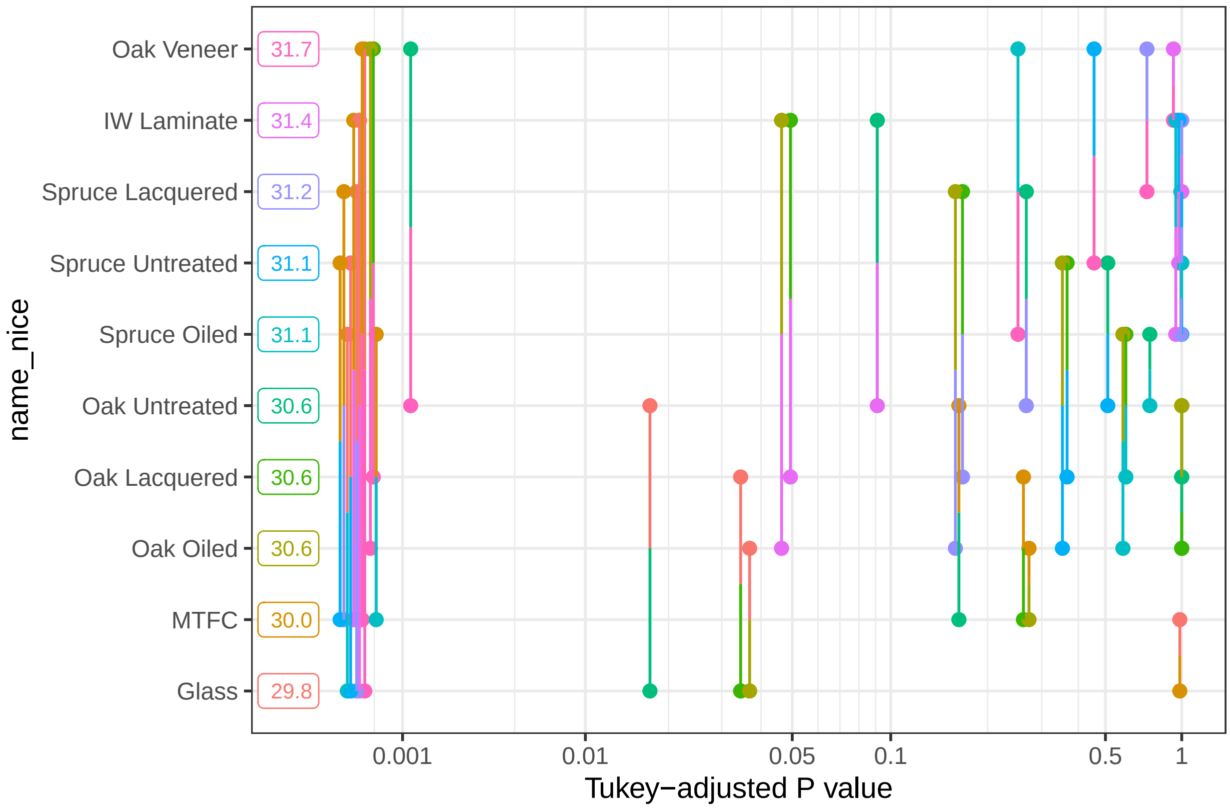
```r
# DF's are calculated using the kenward-rogers method
# P-values are adjsuted with the FDR method ffor 10 tests
arms.cont <- as_tibble(contrast(emmeans(arms.lma, ~ name_nice)))
#arms.cont
# DF are calculated using Kenward-rogers
# P-value adjusted using the tukey method for a family of 10 estimates
arms.pairs <- as_tibble(pairs(emmeans(arms.lma, ~ name_nice)))
arms.pairs.ci <- as_tibble(confint(pairs(emmeans(arms.lma, ~ name_nice))))
arms.pairs <- full_join(arms.pairs, arms.pairs.ci %>%
                          select(contrast, upper.CL, lower.CL), by="contrast")
arms.pairs <- arms.pairs %>%
  separate(contrast, c("Material1", "Material2"), sep="-") %>%
  mutate(Material1 = trimws(Material1),
         Material2 = trimws(Material2))
```

```r
#write_excel_csv(arms.pairs, "post_temp_pairs.csv")
```

Visualising contrasts

The plot concept is nice, but its very busy. We include it here, but it is not reported in the associated publication.

```r
pwpp(emmeans(arms.lma, ~ name_nice))
```



Difference between Post - Pre temperature

First we need to create a data frame with the difference between Post and Pre stages. Then we can fit the model as before. Here we expect large effect sizes and see them.

```r
top_n(d.sm.arms, 5)
```

```
## Selecting by range

## # A tibble: 100 x 8
## # Groups:   name_nice, stage [20]
##    name_nice stage sub.fac temp.mu temp.min temp.max temp.sd range
##    <chr>     <chr> <chr>     <dbl>    <dbl>    <dbl>   <dbl> <dbl>
##  1 Glass     post  sub-1      30.8     29.8     31.5   0.384 1.71
##  2 Glass     post  sub-11     30.0     28.8     30.7   0.440 1.91
##  3 Glass     post  sub-12     28.0     26.8     28.7   0.345 1.91
##  4 Glass     post  sub-13     30.4     28.9     31.3   0.491 2.41
##  5 Glass     post  sub-8      30.1     29.2     30.8   0.293 1.63
```

```
##   6 Glass     pre   sub-10     24.8     24.4     25.3   0.135 0.901
##   7 Glass     pre   sub-11     26.2     25.8     26.8   0.240 1.00
##   8 Glass     pre   sub-4      24.1     23.4     24.7   0.273 1.25
##   9 Glass     pre   sub-5      24.7     24.4     25.4   0.187 0.939
## 10 Glass     pre   sub-8      25.0     24.4     25.5   0.206 1.08
## # ... with 90 more rows
```

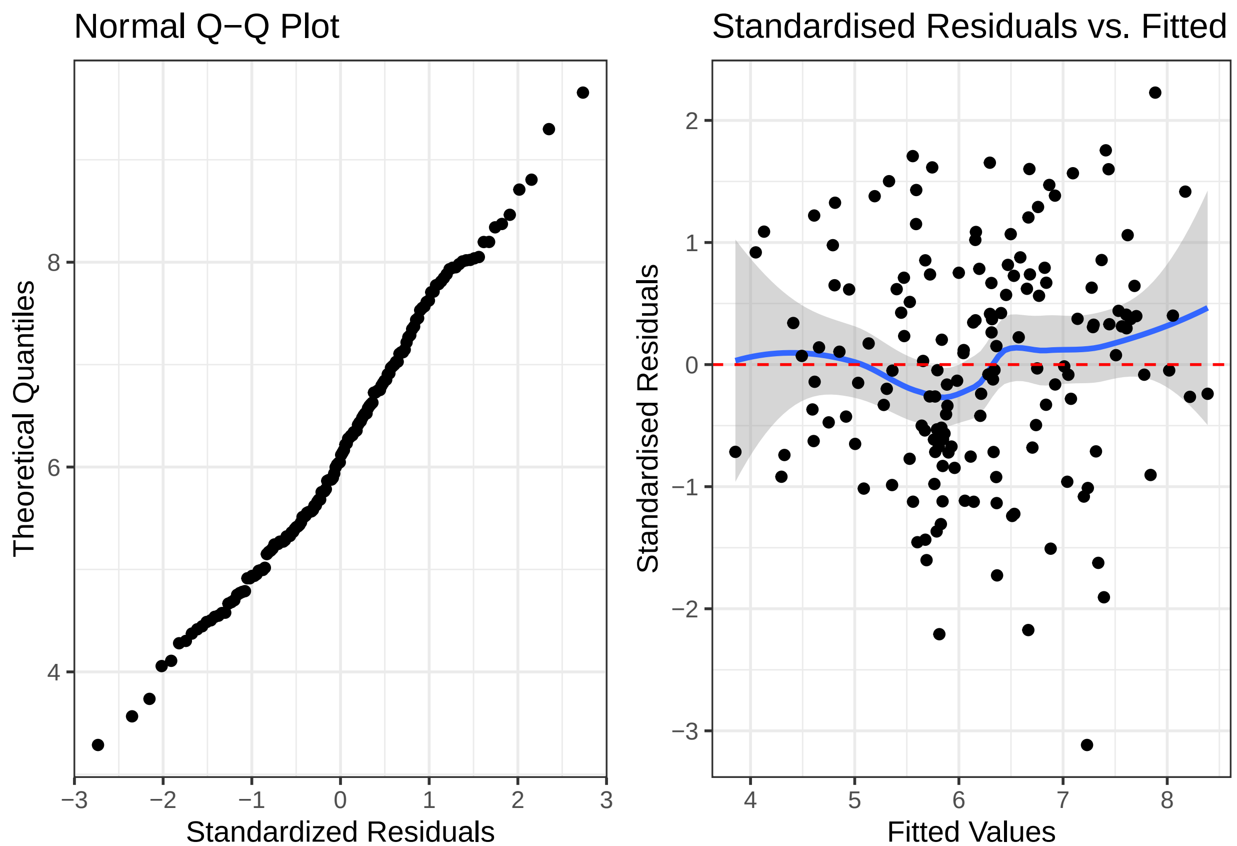
```
d.arms_w <- d.sm.arms %>% select(name_nice, sub.fac, stage, temp.mu) %>%
  pivot_wider(names_from=stage, values_from=temp.mu) %>%
  mutate(temp.dif = post - pre)

dif.lm <- lmer(data=d.arms_w, temp.dif ~ name_nice + (1|sub.fac))
summary(dif.lm)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: temp.dif ~ name_nice + (1 | sub.fac)
##    Data: d.arms_w
##
## REML criterion at convergence: 420.9
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -3.11587 -0.67186 -0.02329  0.64567  2.22760
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  sub.fac  (Intercept) 0.6488   0.8055
##  Residual             0.6322   0.7951
## Number of obs: 160, groups:  sub.fac, 16
##
## Fixed effects:
##                           Estimate Std. Error t value
## (Intercept)                 5.2165     0.2830  18.436
## name_niceIW Laminate        1.5515     0.2811   5.519
## name_niceMTFC              -0.1957     0.2811  -0.696
## name_niceOak Lacquered      0.8664     0.2811   3.082
## name_niceOak Oiled          0.9538     0.2811   3.393
## name_niceOak Untreated      0.9827     0.2811   3.496
## name_niceOak Veneer         1.7213     0.2811   6.123
## name_niceSpruce Lacquered   1.5069     0.2811   5.361
## name_niceSpruce Oiled       1.0368     0.2811   3.688
## name_niceSpruce Untreated   1.3525     0.2811   4.811
##
## Correlation of Fixed Effects:
##             (Intr) nm_IWL n_MTFC nm_nOL nm_nOO nm_nOU nm_nOV nm_nSL nm_nSO
## nm_ncIWLmnt -0.497
## name_ncMTFC -0.497  0.500
## nm_ncOkLcqr -0.497  0.500  0.500
## nam_ncOkOld -0.497  0.500  0.500  0.500
## nm_ncOkUntr -0.497  0.500  0.500  0.500  0.500
## nam_ncOkVnr -0.497  0.500  0.500  0.500  0.500  0.500
## nm_ncSprcLc -0.497  0.500  0.500  0.500  0.500  0.500  0.500
## nm_ncSprcOl -0.497  0.500  0.500  0.500  0.500  0.500  0.500  0.500
## nm_ncSprcUn -0.497  0.500  0.500  0.500  0.500  0.500  0.500  0.500  0.500
```

Fit assessment differences model

Fit is again acceptable.

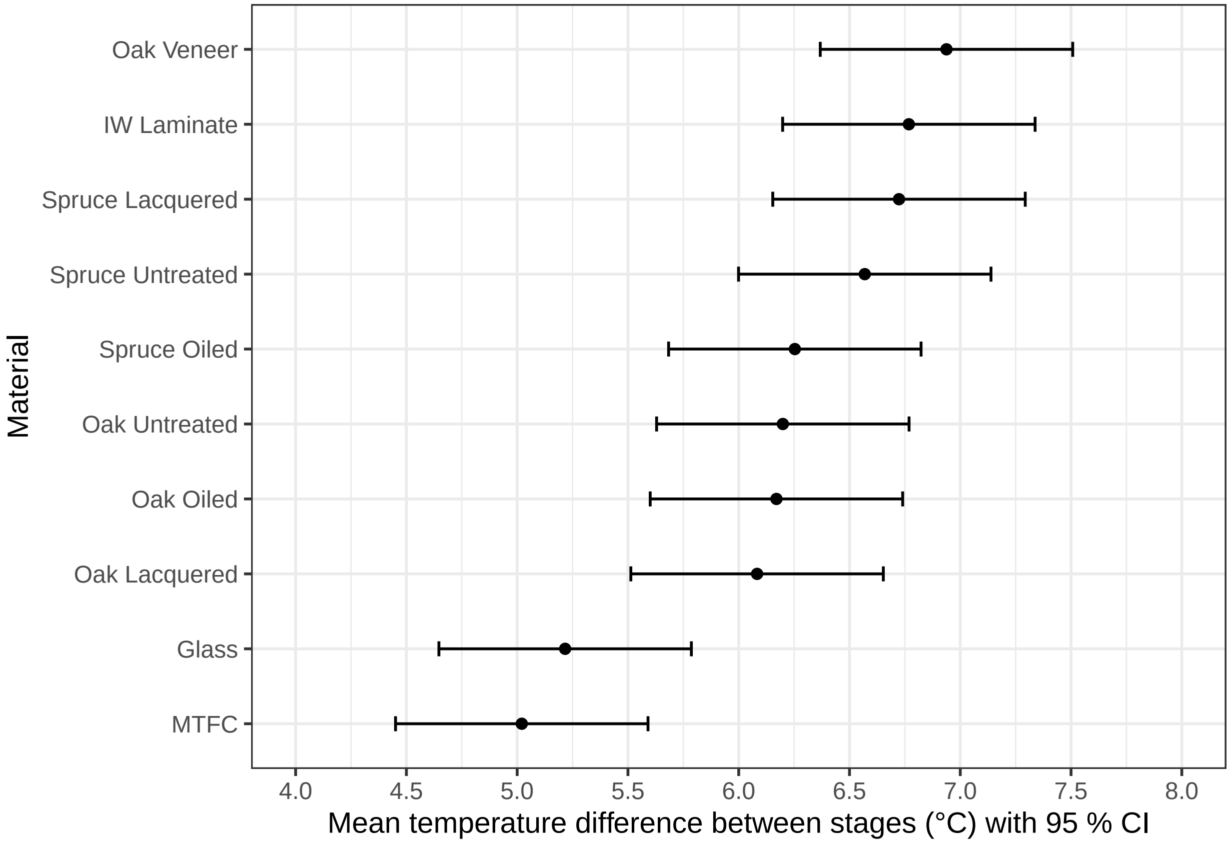
```
p1 <-ggplot(fortify(dif.lm), aes(sample=temp.dif)) +
  stat_qq() +
  labs(title="Normal Q-Q Plot", x="Standardized Residuals", y="Theoretical Quantiles")
#residuals vs. fitted
p2 <- ggplot(fortify(dif.lm), aes(x=.fitted, y=.scresid)) +
  geom_smooth(method="loess") +
  geom_point() +
  geom_hline(yintercept=0, col="red", linetype=2) +
  labs(title="Standardised Residuals vs. Fitted",
y="Standardised Residuals", x="Fitted Values")

grid.arrange(p1, p2, ncol=2)
```



Summary data and visualisation of differences

```
difs.em <- as_tibble(emmeans(dif.lm, ~ name_nice)) %>%
  mutate(name_nice = fct_reorder(name_nice, upper.CL))
ggplot(data = difs.em, aes(x=name_nice, y=emmean,
                              ymin=lower.CL, ymax=upper.CL)) +
  geom_point() +
  geom_errorbar(width=0.2) +
  coord_flip() +
```

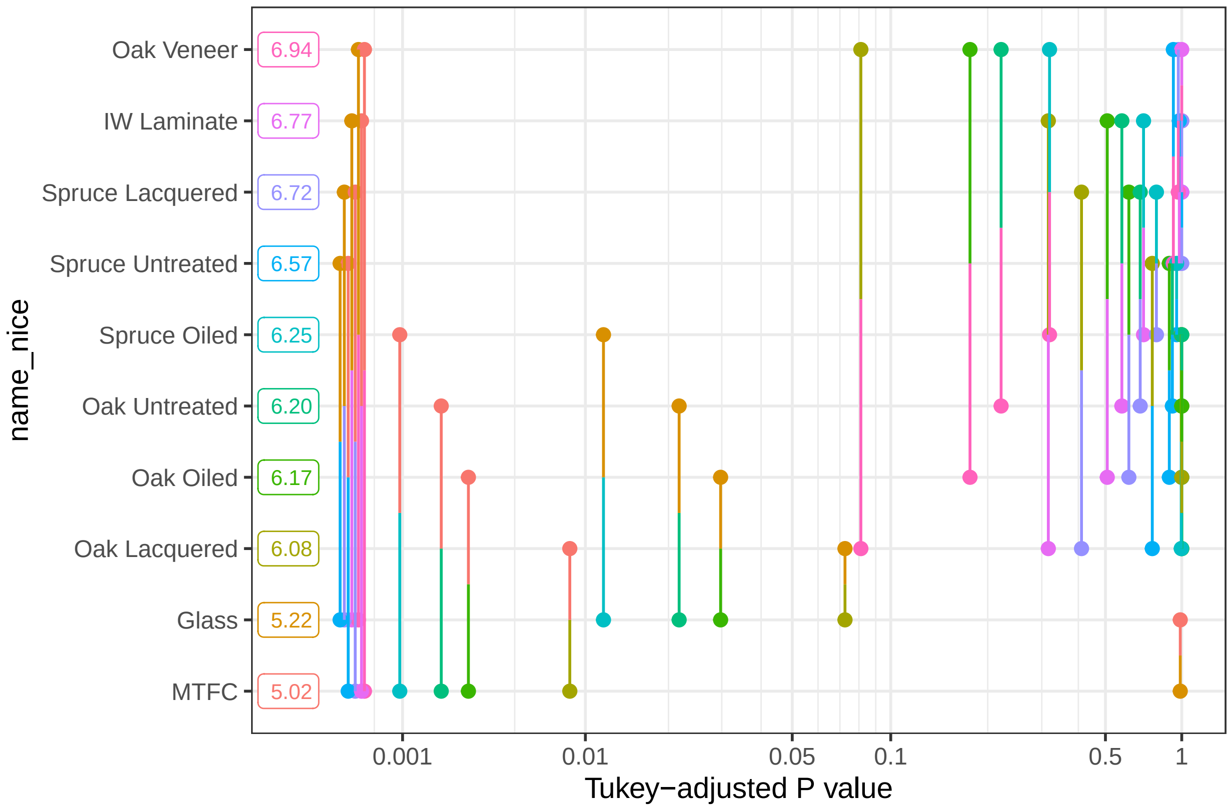
```r
  scale_y_continuous(limits=c(4,8), breaks=seq(4,8,.5)) +
  labs(y="Mean temperature difference between stages (\u00B0C) with 95 % CI",
       x="Material")
```




```r
#ggsave("figs/post-pre_difs.pdf", device=cairo_pdf(), width=9, height=5, units="in")
```

```r
# DF are calculated using Kenward-rogers
# P-value adjusted using the tukey method for a family of 10 estimates
difs.pairs <- as_tibble(pairs(emmeans(dif.lm, ~ name_nice)))
difs.pairs.ci <- as_tibble(confint(pairs(emmeans(dif.lm, ~ name_nice))))
difs.pairs <- full_join(difs.pairs, difs.pairs.ci %>%
                    select(contrast, upper.CL, lower.CL), by="contrast")
difs.pairs <- difs.pairs %>% separate(contrast,
                                  c("Material1", "Material2"),
                                  sep="-") %>%
  mutate(Material1 = trimws(Material1),
         Material2 = trimws(Material2))
#write_excel_csv(difs.pairs, "pre-post_pairs.csv")
```

```r
pwpp(emmeans(dif.lm, ~ name_nice))
```



Data on subjective evaluation of materials

The data include:

- Session (numeric): session number (the order in which materials were tested)
- Visit (numeric): each new day of test sessions counts as a single visit from a participant
- Subject (numeric): the person that participated in the experiment
- Material (string): the code for the material of the table top in the experiment
- Property (string): a string describing the material property that was rated by participants
- Score (numeric): the rating participants gave to materials on a given property (from 1 - especially dislike to 9 - especially like)

Importing data

```
desks_assessment <- read_csv("data/subjective_evaluation.csv", col_types=cols())
```

Due to the non-normal distribution of data, we computed bootstrapped medians and percentile confidence intervals for each material and each subjectively rated property.

Summary statistics

```
# Summary statistics
desks_assessment %>%
  select(-subject) %>%
```

```r
  group_by(material, property) %>%
  get_summary_stats()
```

```
## # A tibble: 120 x 15
##    material property variable     n   min   max median    q1    q3   iqr   mad
##    <chr>    <chr>    <chr>    <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 glass    everyda... score       16     1     7    3.5  2.75  4.25  1.5   1.48
##  2 glass    everyda... session     16     1    10    4.5  2     7.25  5.25  3.71
##  3 glass    everyda... visit       16     1     4    2    1     3     2     1.48
##  4 glass    touch_p... score       16     1     7    5    3     6     3     1.48
##  5 glass    touch_p... session     16     1    10    4.5  2     7.25  5.25  3.71
##  6 glass    touch_p... visit       16     1     4    2    1     3     2     1.48
##  7 glass    vision_... score       16     2     8    6.5  2.75  7     4.25  1.48
##  8 glass    vision_... session     16     1    10    4.5  2     7.25  5.25  3.71
##  9 glass    vision_... visit       16     1     4    2    1     3     2     1.48
## 10 glass    writing... score       16     3     8    6    4     7.25  3.25  2.96
## # ... with 110 more rows, and 4 more variables: mean <dbl>, sd <dbl>, se <dbl>,
## #   ci <dbl>
```

```r
# Summary statistics with boostraping
boot_assessment <- groupwiseMedian(
  score ~ property + material,
  data = desks_assessment,
  bca = FALSE,
  percentile = TRUE,
  R = 10000
)
```

Friedman rank sum tests and pairwise comparisons with Wilcoxon tests

We use Friedman rank sum tests to calculate if the ratings on each subjectively rated property differ between any two rated materials. Where statistically signifnicant differences are detected, post-hoc comparisons are calculated with two sample Wilcoxon tests, comparing all possible pairs of materials.

```r
##########################

# Pleasant to the touch
desks_assessment %>%
  filter(property == "touch_pleasant") %>%
  friedman_test(score ~ material | subject)
```

```
## # A tibble: 1 x 6
##   .y.       n statistic    df         p method
## * <chr> <int>     <dbl> <dbl>     <dbl> <chr>
## 1 score    16      38.3     9 0.0000156 Friedman test
```

```r
desks_assessment %>%
  filter(property == "touch_pleasant") %>%
  pairwise_wilcox_test(score ~ material, paired = TRUE, p.adjust.method = "fdr") %>%
  filter(p.adj.signif != "ns")
```

```
## # A tibble: 5 x 9
##   .y.   group1 group2            n1    n2 statistic        p p.adj p.adj.signif
##   <chr> <chr>  <chr>          <int> <int>     <dbl>    <dbl> <dbl> <chr>
## 1 score glass  melamine          16    16         0 0.000974 0.031 *
```

```
## 2 score glass  oak_lacquer       16    16       9.5 0.003    0.038 *
## 3 score glass  oak_oil           16    16       4   0.001    0.031 *
## 4 score glass  spruce_lacquer    16    16      10   0.005    0.045 *
## 5 score kerrok melamine          16    16       7.5 0.005    0.045 *
```

```
##########################

# Pleasant to the eye
desks_assessment %>%
  filter(property == "vision_pleasant") %>%
  friedman_test(score ~ material | subject)
```

```
## # A tibble: 1 x 6
##   .y.       n statistic    df          p method
## * <chr> <int>     <dbl> <dbl>      <dbl> <chr>
## 1 score    16      41.6     9 0.00000392 Friedman test
```

```
desks_assessment %>%
  filter(property == "vision_pleasant") %>%
  pairwise_wilcox_test(score ~ material, paired = TRUE, p.adjust.method = "fdr") %>%
  filter(p.adj.signif != "ns")
```

```
## # A tibble: 7 x 9
##   .y.   group1 group2         n1    n2 statistic     p p.adj p.adj.signif
##   <chr> <chr>  <chr>       <int> <int>     <dbl> <dbl> <dbl> <chr>
## 1 score glass  oak_oil        16    16      12   0.004 0.034 *
## 2 score kerrok melamine       16    16       0   0.002 0.018 *
## 3 score kerrok oak_lacquer    16    16      15.5 0.007 0.045 *
## 4 score kerrok oak_oil        16    16       0   0.001 0.018 *
## 5 score kerrok oak_raw        16    16       0   0.002 0.018 *
## 6 score kerrok spruce_oil     16    16       6.5 0.007 0.045 *
## 7 score kerrok veneer         16    16       1.5 0.001 0.018 *
```

```
##########################

# Suitable for writing
desks_assessment %>%
  filter(property == "writing_suitable") %>%
  friedman_test(score ~ material | subject)
```

```
## # A tibble: 1 x 6
##   .y.       n statistic    df      p method
## * <chr> <int>     <dbl> <dbl>  <dbl> <chr>
## 1 score    16      16.7     9 0.0530 Friedman test
```

```
# desks_assessment %>%
#   filter(property == "writing_suitable") %>%
#   pairwise_wilcox_test(score ~ material, paired = TRUE, p.adjust.method = "fdr") %>%
#   filter(p.adj.signif != "ns")

##########################

# Suitable for everyday use
desks_assessment %>%
  filter(property == "everyday_use") %>%
  friedman_test(score ~ material | subject)
```

```
## # A tibble: 1 x 6
##   .y.       n statistic    df           p method
## * <chr> <int>     <dbl> <dbl>       <dbl> <chr>
## 1 score    16      46.4     9 0.000000503 Friedman test
```

```
desks_assessment %>%
  filter(property == "everyday_use") %>%
  pairwise_wilcox_test(score ~ material, paired = TRUE, p.adjust.method = "fdr") %>%
  filter(p.adj.signif != "ns")
```

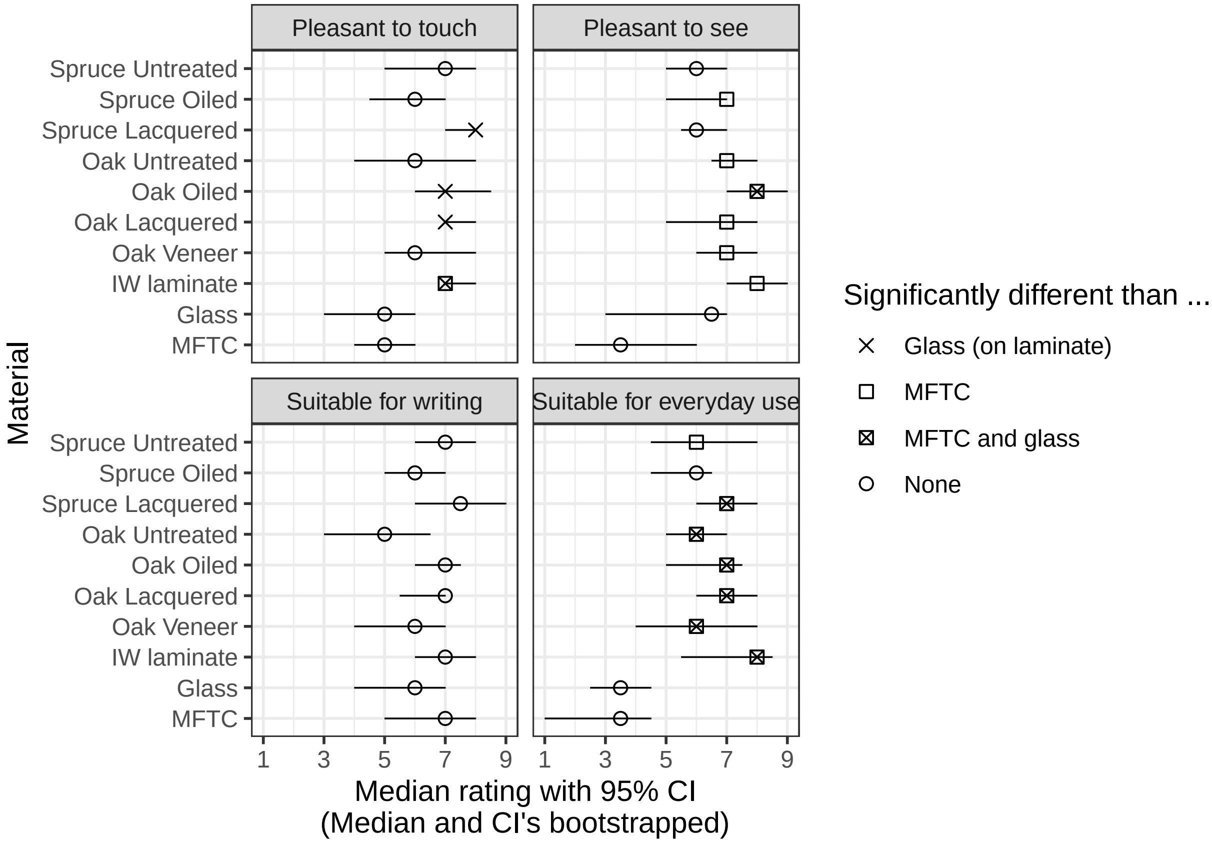
```
## # A tibble: 13 x 9
##    .y.   group1 group2            n1    n2 statistic     p p.adj p.adj.signif
##    <chr> <chr>  <chr>          <int> <int>     <dbl> <dbl> <dbl> <chr>
##  1 score glass  melamine          16    16       0   0.001 0.017 *
##  2 score glass  oak_lacquer       16    16       0   0.004 0.03  *
##  3 score glass  oak_oil           16    16      15.5 0.012 0.045 *
##  4 score glass  oak_raw           16    16      10   0.014 0.048 *
##  5 score glass  spruce_lacquer    16    16      14.5 0.006 0.03  *
##  6 score glass  veneer            16    16      12   0.007 0.03  *
##  7 score kerrok melamine          16    16       0   0.001 0.017 *
##  8 score kerrok oak_lacquer       16    16       4   0.002 0.017 *
##  9 score kerrok oak_oil           16    16      11   0.006 0.03  *
## 10 score kerrok oak_raw           16    16      12.5 0.004 0.03  *
## 11 score kerrok spruce_lacquer    16    16      13   0.005 0.03  *
## 12 score kerrok spruce_raw        16    16      18   0.01  0.042 *
## 13 score kerrok veneer            16    16       6   0.001 0.017 *
```

Plot (subjective assessment of materials)

Plot displays boostrapped median ratings and percentile confidence intervals for each material on each subjectively rated property. Statistically significant comparisons are marked.

```
material_labels <- rev(c("Spruce Untreated", "Spruce Oiled", "Spruce Lacquered",
                         "Oak Untreated", "Oak Oiled", "Oak Lacquered",
                         "Oak Veneer", "IW laminate", "Glass", "MFTC"))

boot_assessment %>%
  mutate(
    significant = case_when(
      material == "oak_raw" & property == "vision_pleasant" ~ "vs_kerrok",
      material == "veneer" & property == "vision_pleasant" ~ "vs_kerrok",
      material == "oak_oil" & property == "vision_pleasant" ~ "vs_both",
      material == "oak_lacquer" & property == "vision_pleasant" ~ "vs_kerrok",
      material == "spruce_oil" & property == "vision_pleasant" ~ "vs_kerrok",
      material == "melamine" & property == "vision_pleasant" ~ "vs_kerrok",
      material == "oak_oil" & property == "touch_pleasant" ~ "vs_glass",
      material == "oak_lacquer" & property == "touch_pleasant" ~ "vs_glass",
      material == "spruce_lacquer" & property == "touch_pleasant" ~ "vs_glass",
      material == "melamine" & property == "touch_pleasant" ~ "vs_both",
      material == "oak_raw" & property == "everyday_use" ~ "vs_both",
      material == "veneer" & property == "everyday_use" ~ "vs_both",
      material == "oak_oil" & property == "everyday_use" ~ "vs_both",
      material == "oak_lacquer" & property == "everyday_use" ~ "vs_both",
      material == "spruce_raw" & property == "everyday_use" ~ "vs_kerrok",
```

```r
    material == "spruce_lacquer" & property == "everyday_use" ~ "vs_both",
    material == "melamine" & property == "everyday_use" ~ "vs_both"
  ),
  significant = ifelse(is.na(significant), "vs_none", significant)
) %>%
ggplot(aes(factor
(
  material,
  levels = rev(c("spruce_raw", "spruce_oil", "spruce_lacquer",
                 "oak_raw", "oak_oil", "oak_lacquer", "veneer",
                 "melamine", "glass", "kerrok"))
),
Median,
ymin = Percentile.lower,
ymax = Percentile.upper,
shape = significant
)) +
geom_point(size = 2, position = position_dodge(width = 0.6)) +
geom_errorbar(size = 0.3, width = 0, position = position_dodge(width = 0.6)) +
facet_wrap(~ factor(property, levels = c("touch_pleasant", "vision_pleasant",
                                        "writing_suitable", "everyday_use"),
                    labels = c("Pleasant to touch", "Pleasant to see",
                               "Suitable for writing",
                               "Suitable for everyday use")),
           nrow = 2) +
labs(
  x = "Material",
  y = "Median rating with 95% CI\n(Median and CI's bootstrapped)"
) +
scale_shape_manual(
  name = "Significantly different than ...",
  breaks = c("vs_glass", "vs_kerrok", "vs_both", "vs_none"),
  labels = c("Glass (on laminate)", "MFTC", "MFTC and glass", "None"),
  values = c(7, 4, 0, 1)
) +
scale_x_discrete(labels = material_labels) +
scale_y_continuous(limits=c(1,9), breaks=seq(1,9,2)) +
coord_flip()
```




```
# ggsave("material_assessment_v3.png", height = 10, width = 20, unit = "cm")
```

Relationship between thermal conductivity and subjective ratings

We calculate Spearman rank correlation coefficients between material thermal conductivity, temperature, and subjectively rated material properties.

```
# Import data
thermal <- read_csv2("data/thermal_conductivity.csv", col_types=cols())
```

```
## Using ',' as decimal and '.' as grouping mark. Use read_delim() for more control.
```

```
# Calculate correlation coefficients
thermal_correlation <- thermal %>%
  select(-Material) %>%
  cor_mat(method = "spearman")

# Extract p values
thermal_correlation_pvalues <- cor_get_pval(thermal_correlation)
```

Power analyses (post-hoc)

We use simulation to do post-hoc / observed power analysis using the simr package of the lmm's. Using the powerSim() fuction with 1000 sims takes some time (~ 2 minutes)

```r
set.seed(54321)
dif.pwr <- powerSim(dif.lm, nsim=1000, progress=FALSE)
post.pwr <- powerSim(arms.lma, nsim=1000, progress=FALSE)
pre.pwr <- powerSim(arms.lmb, nsim=1000, progress=FALSE)
```

Pre / Before test state power

```r
print(pre.pwr)
```

```
## Power for predictor 'name_nice', (95% confidence interval):
##       43.40% (40.30, 46.54)
##
## Test: Kenward Roger (package pbkrtest)
##
## Based on 1000 simulations, (6 warnings, 0 errors)
## alpha = 0.05, nrow = 160
##
## Time elapsed: 0 h 1 m 23 s
##
## nb: result might be an observed power calculation
```

Post / After test state power

```r
print(post.pwr)
```

```
## Power for predictor 'name_nice', (95% confidence interval):
##       100.0% (99.63, 100.0)
##
## Test: Kenward Roger (package pbkrtest)
##
## Based on 1000 simulations, (3 warnings, 0 errors)
## alpha = 0.05, nrow = 160
##
## Time elapsed: 0 h 1 m 19 s
##
## nb: result might be an observed power calculation
```

Post - Pre power

```r
print(dif.pwr)
```

```
## Power for predictor 'name_nice', (95% confidence interval):
##       100.0% (99.63, 100.0)
##
## Test: Kenward Roger (package pbkrtest)
##
## Based on 1000 simulations, (6 warnings, 0 errors)
## alpha = 0.05, nrow = 160
##
## Time elapsed: 0 h 1 m 23 s
```

```
##
## nb: result might be an observed power calculation
```

Environment

Information about the R environment where the analyses were conducted.

```r
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Catalina 10.15.2
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] rcompanion_2.3.7 rstatix_0.3.1    simr_1.0.5       Cairo_1.5-10
##  [5] gridExtra_2.3    ggridges_0.5.1   emmeans_1.4.3.01 lme4_1.1-21
##  [9] Matrix_1.2-18    forcats_0.4.0    stringr_1.4.0    dplyr_0.8.3
## [13] purrr_0.3.3      readr_1.3.1      tidyr_1.0.0      tibble_2.1.3
## [17] ggplot2_3.2.1    tidyverse_1.3.0
##
## loaded via a namespace (and not attached):
##  [1] TH.data_1.0-10     minqa_1.2.4        colorspace_1.4-1   ellipsis_0.3.0
##  [5] modeltools_0.2-22  rio_0.5.16         estimability_1.3   fs_1.3.1
##  [9] rstudioapi_0.10    farver_2.0.1       fansi_0.4.0        mvtnorm_1.0-11
## [13] lubridate_1.7.4    coin_1.3-1         xml2_1.2.2         codetools_0.2-16
## [17] splines_3.6.1      libcoin_1.0-5      knitr_1.26         zeallot_0.1.0
## [21] jsonlite_1.6       nloptr_1.2.1       pbkrtest_0.4-7     broom_0.5.3
## [25] binom_1.1-1        dbplyr_1.4.2       compiler_3.6.1     httr_1.4.1
## [29] backports_1.1.5    assertthat_0.2.1   lazyeval_0.2.2     cli_2.0.0
## [33] htmltools_0.4.0    tools_3.6.1        coda_0.19-3        gtable_0.3.0
## [37] glue_1.3.1         Rcpp_1.0.3         carData_3.0-3      cellranger_1.1.0
## [41] vctrs_0.2.1        nlme_3.1-143       iterators_1.0.12   lmtest_0.9-37
## [45] xfun_0.11          openxlsx_4.1.4     rvest_0.3.5        lifecycle_0.1.0
## [49] MASS_7.3-51.4      zoo_1.8-6          scales_1.1.0       hms_0.5.2
## [53] parallel_3.6.1     sandwich_2.5-1     expm_0.999-4       yaml_2.2.0
## [57] curl_4.3           EMT_1.1            stringi_1.4.3      nortest_1.0-4
## [61] plotrix_3.7-7      boot_1.3-23        zip_2.0.4          rlang_0.4.2
## [65] pkgconfig_2.0.3    matrixStats_0.55.0 evaluate_0.14      lattice_0.20-38
## [69] labeling_0.3       tidyselect_0.2.5   plyr_1.8.5         magrittr_1.5
## [73] R6_2.4.1           multcompView_0.1-7 DescTools_0.99.31  generics_0.0.2
## [77] multcomp_1.4-11    RLRsim_3.1-3       DBI_1.1.0          pillar_1.4.2
## [81] haven_2.2.0        foreign_0.8-73     withr_2.1.2        mgcv_1.8-31
```

```
## [85] survival_3.1-8     abind_1.4-5        modelr_0.1.5       crayon_1.3.4
## [89] car_3.0-5          utf8_1.1.4         rmarkdown_2.0      readxl_1.3.1.9000
## [93] data.table_1.12.8  reprex_0.3.0       digest_0.6.23      xtable_1.8-4
## [97] stats4_3.6.1       munsell_0.5.0
```