Inspecting, Plotting, & Modelling Check Density in Decorative Maple Veneered Plywood Panels

Michael D. Burnard, University of Primorska, InnoRenew CoE
Lisa Ganio, Oregon State University
Scott Leavengood, Oregon State University
Lech Musynski, Oregon State University

31 July 2018

Contents

Introduction

About the project

The project this analysis is derived from explored how different manufacturing decisions related to producing decorative maple veneered plywood panels affect crack development in thin veneer overlays. Cracks in wood, often refered to as *checks*, are a common source of customer complaints to manufacturers of furniture. The project was the masters thesis of Michael Burnard at Oregon State University in 2012 (Michael D Burnard 2012). The project employed a novel application of digital image correlation to detect, count, measure, and describe check development over time. The method for observing panels was automated so that many panels could be observed nearly simultaneously (Michael David Burnard et al. 2018). This study proposed a new measure of check severity, called Check Density, that is the sum of the area of each check on a panel divided by one square meter. This measure provides a comparable value for future studies.

The study examined 96 combinations of 4 manufacturing factors: veneer type (4 levels), core type (4 levels), adhesive type (3 levels), lathe check orientation (2 levels). There were 8 replicates of each treatment combination, except in two cases where material defects caused shortages during manufacturing.

The script version (.Rmd) of this file is available.

About the analysis

This document contains the data analysis, illustrative plots, and modeling employed to explore and understand how selection of manufacturing decisions affect crack development in decorative maple veneered plywood panels. It relies on a several common R packages for data wrangling, plotting, and model fitting. The data did not fit the assumptions of a linear model based on the normal distribution. Instead, Tweedie's Compound Poisson distribution was used. The *cplm* package was used to fit this model (Zhang 2013). Bootstrapping was used to establish reliable confidence intervals associated with them. The bootstrapping code is included in this report, but it is not run here; it is also available as supplemental material to the article. The original and output data are also available as supplemental material. Most charts will get slightly modified or adjusted in Adobe Illustrator for the final submitted versions (and potentially further modified in layout), so they may not exactly the same as in the journal version.

Setting the environment

The packages used in this analysis.

```
#Data wrangling & visualisation
library(tidyverse)
library(gridExtra)
library(scales)
#Fitting the models
library(cplm)
#bootstrapping confidence intervals
library(boot)
```

Reading and organising the data

The raw data that is the subject this analysis, and contained in the associated Burnardetal2018_MapleCheckingInputData.csv file, contains information about check density for 12 points in time. This data can be used to explore check progression and development over time, but, is used here to describe and predict the severity of checking for plywood panels manufactured using the components of interest for this study.

The first step with the data is to read the raw data file, and condense it to the information we're interested in. There are two principal aspects of the data we're interested for this analysis.
1. The non-zero check density information at its peak for each panel.
2. The number of panels with and without checks.

```r
#The raw data
d <- read.csv("Burnardetal2018_MapleCheckingInputData.csv", stringsAsFactors=FALSE)

####
#Select, only the peak CD for each sample, and remove zero values,
#remove "type" field to avoid confusion.
d.nz <- d %>% group_by(name) %>%
  filter(checkDensity == max(checkDensity), checkDensity > 0)
#This leaves a few panels that have maximum check density at two or more stages,
#e.g., stage 23 and 24 are equal. In these cases, we take the earlier stage.
d.nz <- d.nz %>% group_by(name) %>% filter(stage == min(stage))

#duplicated(d.nz$name) #run this line to find out if there are duplicate names.
#There is an extreme value that could not be validated by
#reexamination of the panel in question
#We remove it. There will be only 7 observations for this treatment.
max(d.nz$checkDensity) # 18331
```

```
## [1] 18331.62
```

```r
d.nz <- d.nz %>% filter(checkDensity<18330)
glimpse(d.nz)
```

```
## Observations: 428
## Variables: 9
## $ X            <int> 11, 26, 32, 49, 65, 78, 91, 102, 111, 128, 138, 1...
## $ name         <chr> "A-001", "A-002", "A-003", "A-004", "A-005", "A-0...
## $ stage        <int> 22, 24, 17, 21, 24, 24, 24, 22, 18, 22, 19, 21, 2...
## $ checkDensity <dbl> 22.248, 846.724, 227.109, 801.633, 2011.687, 47.0...
## $ combo_string <chr> "S5-L-C-U", "S5-L-V-U", "S4-T-C-U", "P4-T-C-U", "...
## $ veneer       <chr> "S5", "S5", "S4", "P4", "S5", "S5", "P3", "S4", "...
## $ lc           <chr> "L", "L", "T", "T", "T", "L", "T", "T", "L", "L",...
## $ core         <chr> "C", "V", "C", "C", "C", "M", "M", "V", "M", "P",...
## $ glue         <chr> "U", "U", "U", "U", "U", "U", "U", "U", "U", "U",...
```

```r
######
# A data frame for counts of checks/nochecks

d.c <- d %>% group_by(name) %>% filter(checkDensity==max(checkDensity))
d.c <- d.c %>% group_by(name) %>% filter(stage==min(stage), checkDensity<188330) %>%
  mutate(checked = ifelse(checkDensity == 0, "No Checks","Checks"),
         chkd.logical = ifelse(checkDensity == 0, 0, 1))
```

Exploring the data

Tabular view

As an example, some summary of the raw data can be presented like this.

```
#summary info
d.nz %>% group_by(veneer,core) %>%
  summarise(Mean=mean(checkDensity), Median=median(checkDensity), SD=sd(checkDensity),
            Min=min(checkDensity), Max=max(checkDensity))
```

veneer	core	Mean	Median	SD	Min	Max
P3	C	606.3001	235.4040	1027.6534	9.434	4574.976
P3	M	229.9662	101.3465	306.7415	5.136	988.850
P3	P	106.0081	61.3495	116.5655	8.858	398.155
P3	V	245.3300	111.0760	405.7190	5.279	1814.481
P4	C	373.6979	98.9370	1208.1754	4.971	6917.461
P4	M	268.7002	72.0155	578.0738	7.913	2694.959
P4	P	171.0712	49.5520	427.6014	6.025	2083.980
P4	V	417.7897	159.0410	712.0923	6.229	2983.491
S4	C	254.0634	73.2280	665.6709	5.412	3457.401
S4	M	219.2876	97.7450	339.4943	6.586	1582.758
S4	P	143.7753	38.4200	238.6020	5.191	969.825
S4	V	559.3223	181.0140	1431.0514	7.696	7128.701
S5	C	429.6594	76.5700	857.2873	5.817	3726.530
S5	M	291.2649	98.3510	512.5438	5.197	2148.354
S5	P	333.4987	76.1690	593.7201	8.058	2402.929
S5	V	523.3359	209.8250	975.0117	5.516	5708.171

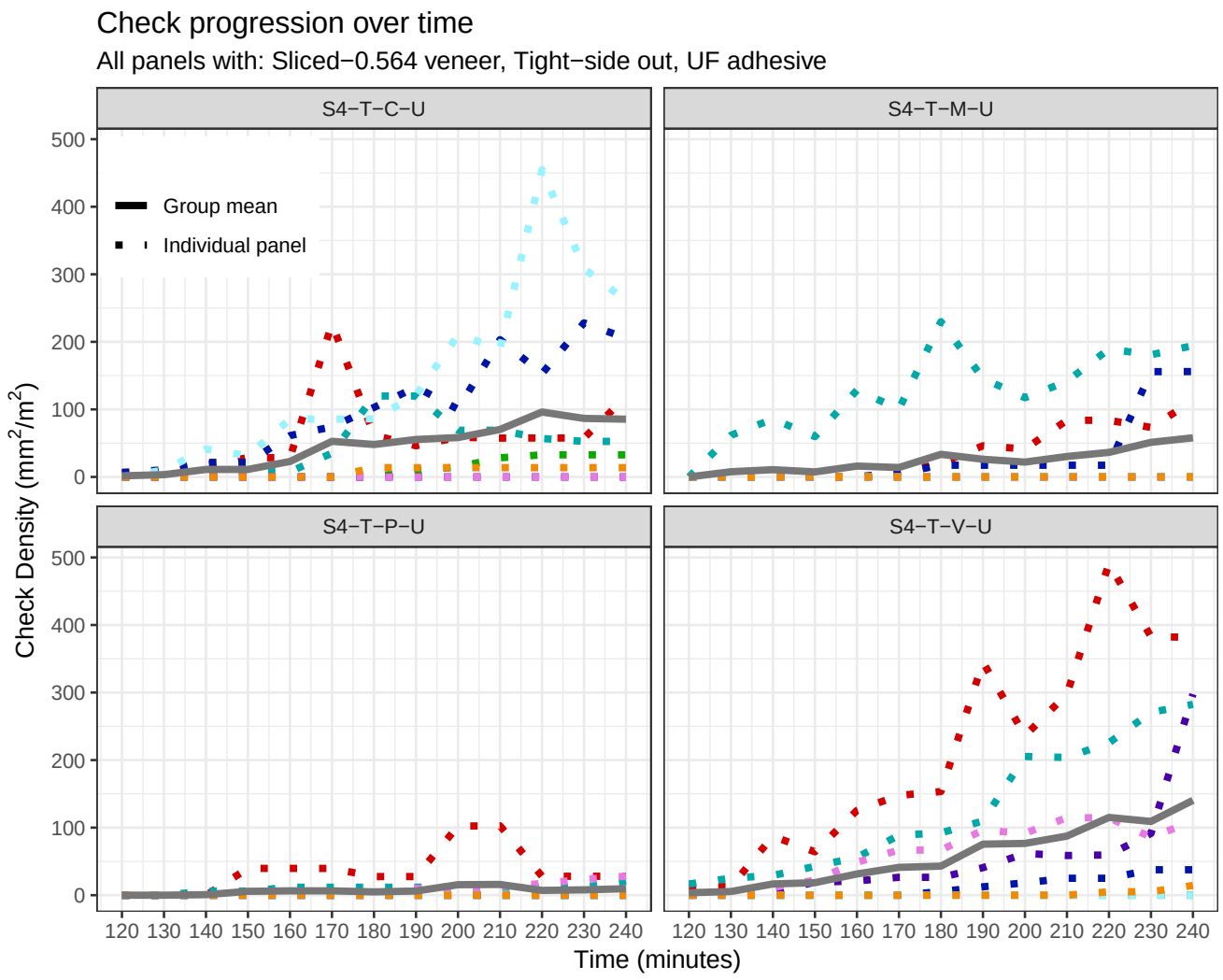
```
#all panel types where checking occured in all panels
d.c %>% group_by(combo_string) %>% summarise(cnt = sum(chkd.logical)) %>% filter(cnt == 8)
```

combo_string	cnt
P3-T-V-U	8
P4-L-V-U	8
S5-L-V-S	8
S5-T-V-U	8

Check progression over time

This is a little bit of a carry over from the previous paper, but it provides insight into the variable patterns of check progression, and that peak check density is often reached before the end of the test period. Here we use only the checking data for panels made with sliced - 0.564 mm veneer with the tight side out, using UF adhesive. Each facet is for a different core type.

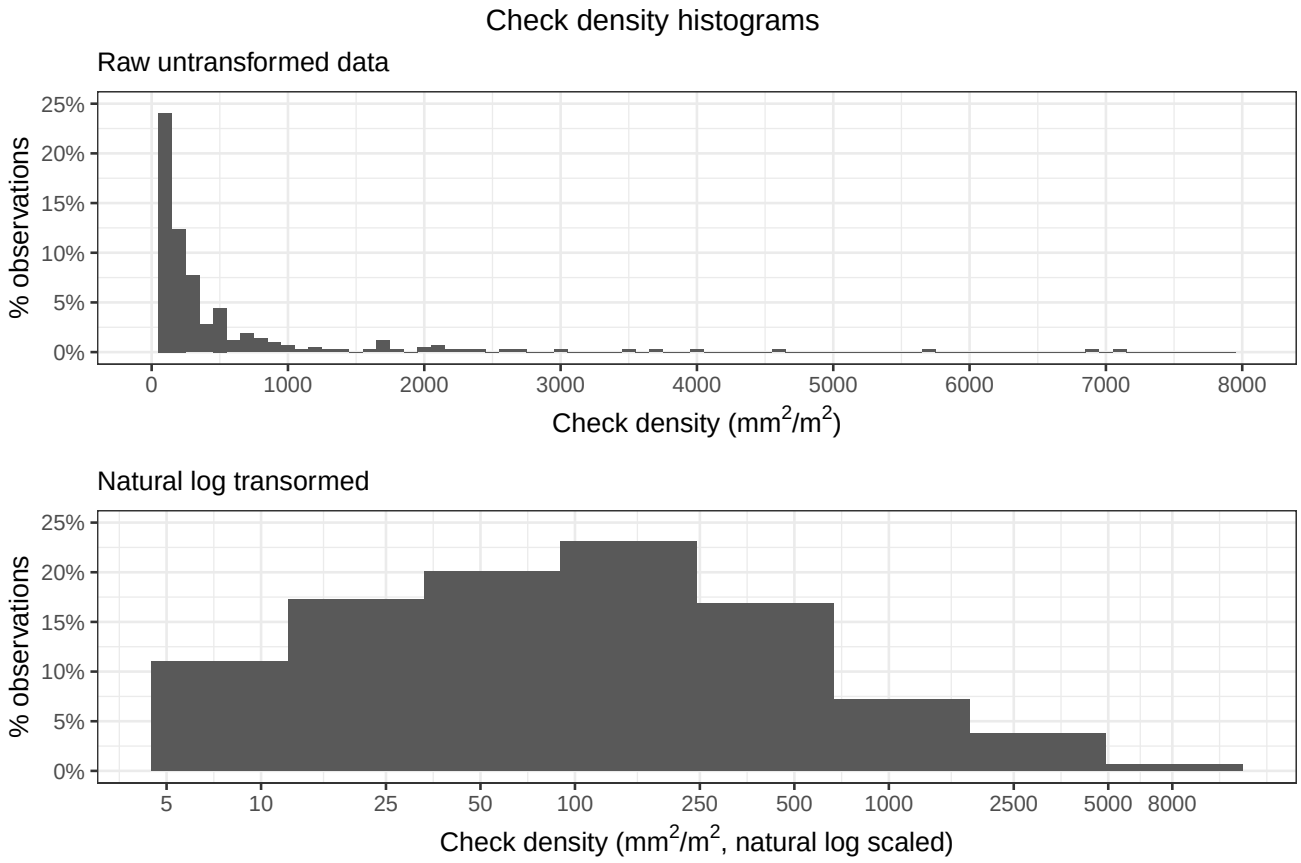
```
cp <- read.csv("Burnardetal2018_checkProgression.csv", stringsAsFactors=FALSE)
ggplot(data=cp, aes(x=minutes, y=checkDensity, colour=colour, linetype=type)) + theme_bw() +
  geom_line(size=1.5) +
  facet_wrap(~combo_string, nrow=2) +
  scale_linetype_manual(values=c(1,3), name="") +
  scale_colour_manual(values=c("#D30000", "#0EA800", "#00A8A8", "#0014A8",
                               "#4900A8", "#E679E2", "#97F3FF", "#F28900",
                               "#777777")) +
  scale_x_continuous(breaks=seq(120,240,10)) +
  guides(colour=FALSE) +
  labs(title="Check progression over time",
       subtitle="All panels with: Sliced-0.564 veneer, Tight-side out, UF adhesive",
       x="Time (minutes)",
       y=expression(paste("Check Density ", "(", mm^2, "/", m^2, ")"))) +
  theme(legend.position=c(.1,.9))
```



#Plots to show the raw data in summary
Before fitting the model, we need to learn a bit about the data. We can see it's quite skewed, so we include a transformation (natural log).

Histograms to show the distribution of observations pre- and post-transformation

```
#produces an ignorable error.
raw <- ggplot(d.nz, aes(checkDensity)) + theme_bw() +
  geom_histogram(aes(y = (..count..)/sum(..count..)), binwidth=100) +
  scale_x_continuous(limits=c(0,8000), breaks=seq(0,8000,1000)) +
  scale_y_continuous(labels=scales::percent, limits=c(0,.25)) +
  labs(subtitle="Raw untransformed data", y="% observations",
       x=expression(paste("Check density (", mm^{2}, "/", m^{2}, ")")))

 tfd <- ggplot(d.nz, aes(checkDensity)) + theme_bw() +
  geom_histogram(aes(y = (..count..)/sum(..count..)), binwidth=1) +
  scale_x_continuous(trans="log", breaks=c(0,1,5,10,25,50,100,250,500,1000,2500,5000,8000)) +
  scale_y_continuous(labels=scales::percent, limits=c(0,.25)) +
  labs(subtitle="Natural log transormed", y="% observations",
       x=expression(paste("Check density (", mm^{2}, "/", m^{2}, ", natural log scaled)")))
grid.arrange(raw, tfd, nrow=2,top="Check density histograms")
```



Observed panel checking state and check density by factor and level

Here we are looking for a good way to show the fraction of panels that checked within each level of each factor, and the share of the overall observed check density for each level by each factor. We start with creating dataframes for each factor, then will plot them all together.

```
cd.sum <- sum(d.c$checkDensity)
d.c.c <- d.c %>% group_by(core) %>% summarise(PercOfType = sum(chkd.logical)/n(),
                                    PercentOfAll=sum(chkd.logical)/766,
                                    Count.Checked=sum(chkd.logical),
                                    PercOfCD=sum(checkDensity)/cd.sum,
                                    n=n())
d.c.c
```

core	PercOfType	PercentOfAll	Count.Checked	PercOfCD	n
C	0.5989583	0.1501305	115	0.2863488	192
M	0.4869110	0.1214099	93	0.1429007	191
P	0.4450262	0.1109661	85	0.1022453	191
V	0.7083333	0.1775457	136	0.4685052	192

```
d.c.v <- d.c %>% group_by(veneer) %>% summarise(PercOfType = sum(chkd.logical)/n(),
                                    PercentOfAll=sum(chkd.logical)/766,
                                    Count.Checked=sum(chkd.logical),
                                    PercOfCD=sum(checkDensity)/cd.sum,
                                    n=n())
d.c.v
```

veneer	PercOfType	PercentOfAll	Count.Checked	PercOfCD	n
P3	0.5156250	0.1292428	99	0.1890282	192
P4	0.5729167	0.1436031	110	0.3236497	192
S4	0.5468750	0.1370757	105	0.2004727	192
S5	0.6052632	0.1501305	115	0.2868494	190

```
d.c.g <- d.c %>% group_by(glue) %>% summarise(PercOfType = sum(chkd.logical)/n(),
                                    PercentOfAll=sum(chkd.logical)/766,
                                    Count.Checked=sum(chkd.logical),
                                    PercOfCD=sum(checkDensity)/cd.sum,
                                    n=n())
d.c.g
```

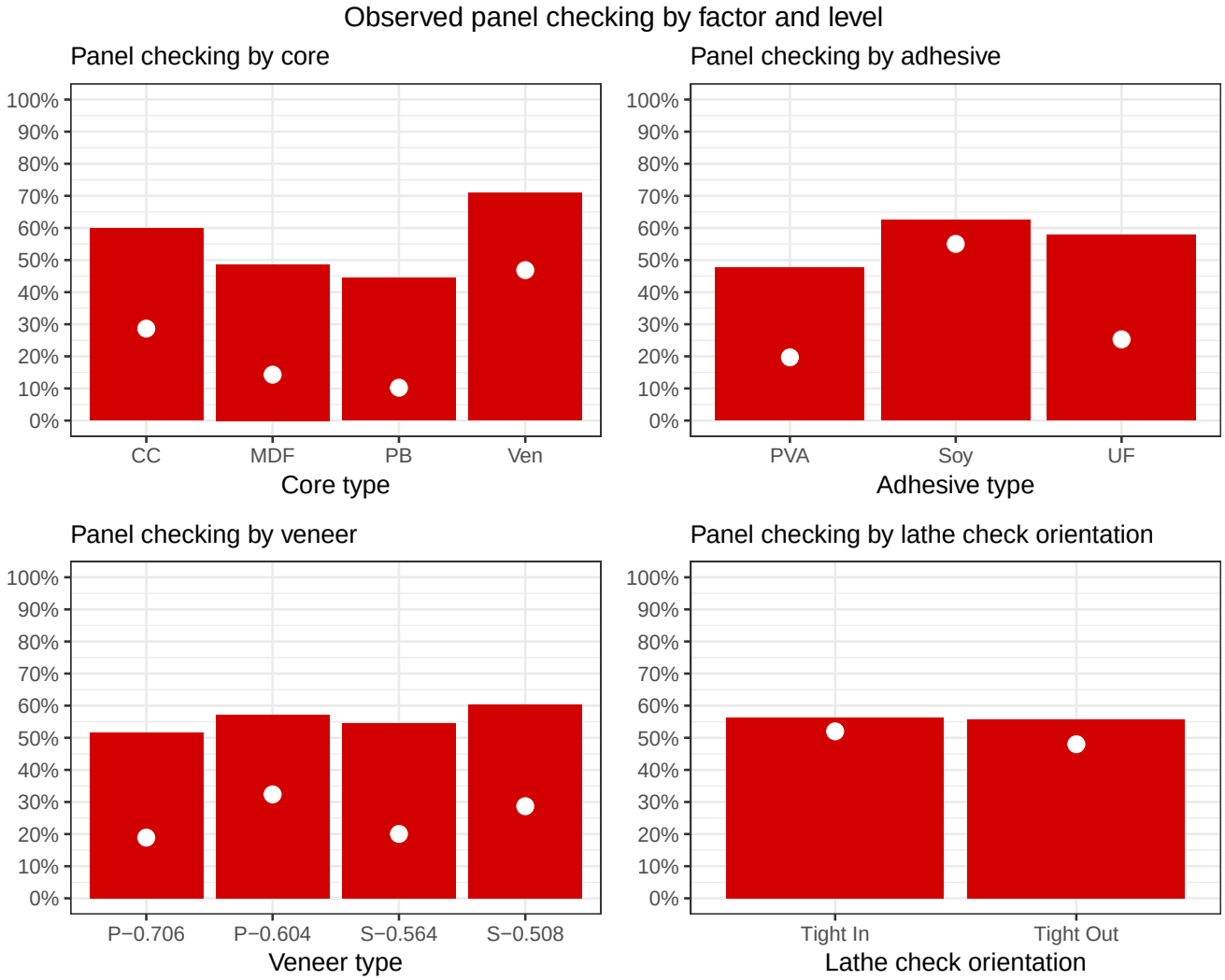
glue	PercOfType	PercentOfAll	Count.Checked	PercOfCD	n
P	0.4765625	0.1592689	122	0.1968355	256
S	0.6250000	0.2088773	160	0.5502507	256
U	0.5787402	0.1919060	147	0.2529137	254

```
d.c.l <- d.c %>% group_by(lc) %>% summarise(PercOfType = sum(chkd.logical)/n(),
                                    PercentOfAll=sum(chkd.logical)/766,
                                    Count.Checked=sum(chkd.logical),
                                    PercOfCD=sum(checkDensity)/cd.sum,
                                    n=n())
d.c.l
```

lc	PercOfType	PercentOfAll	Count.Checked	PercOfCD	n
L	0.5639687	0.2819843	216	0.5200482	383
T	0.5561358	0.2780679	213	0.4799518	383

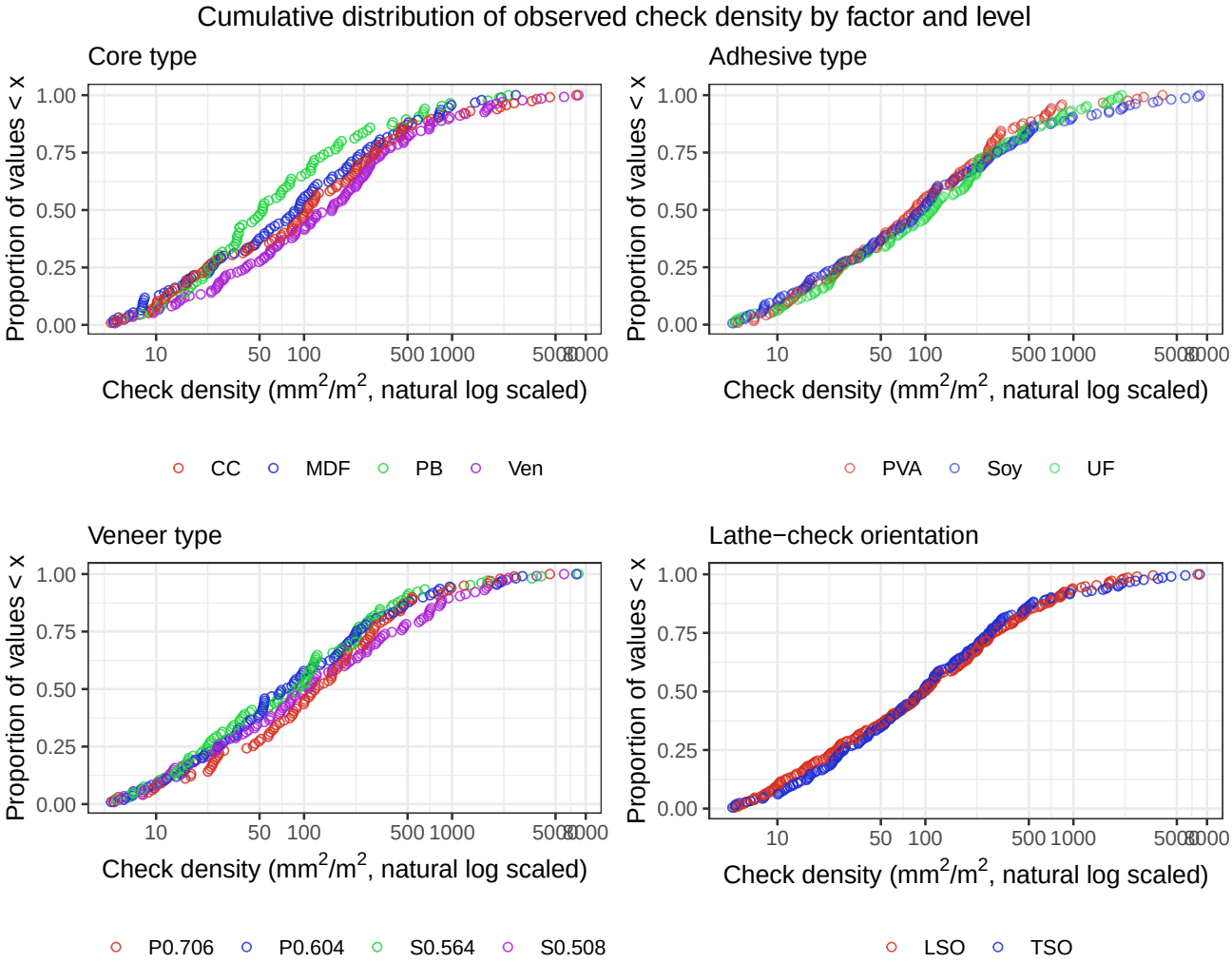
Here we display the number of panels with observed checking and without observed checking as bar charts.

```
pc <- ggplot(d.c.c, aes(y=PercOfType, x=core)) + theme_bw() +
  geom_bar(stat="identity", fill="#D30000") +
  geom_point(data=d.c.c, aes(y=PercOfCD, x=core), colour="#FFFFFF", size=3) +
  scale_y_continuous(labels=percent, breaks=seq(0,1,.1), limits=c(0,1)) +
  scale_x_discrete(labels=c("C"="CC", "M"="MDF",  "P"="PB", "V"="Ven")) +
  labs(subtitle="Panel checking by core", x="Core type", y=NULL) +
  guides(fill=FALSE, size=FALSE, colour=FALSE)

pv <- ggplot(d.c.v, aes(y=PercOfType, x=veneer)) + theme_bw() +
  geom_bar(stat="identity", fill="#D30000") +
  geom_point(data=d.c.v, aes(y=PercOfCD, x=veneer), colour="#FFFFFF", size=3) +
  scale_y_continuous(labels=percent, breaks=seq(0,1,.1), limits=c(0,1)) +
  scale_x_discrete(labels=c("P3"="P-0.706", "P4"="P-0.604",
                            "S4"="S-0.564", "S5"="S-0.508")) +
  labs(subtitle="Panel checking by veneer", x="Veneer type", y=NULL) +
  guides(fill=FALSE, size=FALSE, colour=FALSE)

pg <- ggplot(d.c.g, aes(y=PercOfType, x=glue)) + theme_bw() +
  geom_bar(stat="identity", fill="#D30000") +
  geom_point(data=d.c.g, aes(y=PercOfCD, x=glue), colour="#FFFFFF", size=3) +
  scale_y_continuous(labels=percent, breaks=seq(0,1,.1), limits=c(0,1)) +
  scale_x_discrete(labels=c("P"="PVA", "S"="Soy",  "U"="UF")) +
  labs(subtitle="Panel checking by adhesive", x="Adhesive type", y=NULL) +
  guides(fill=FALSE, size=FALSE, colour=FALSE)

pl <- ggplot(d.c.l, aes(y=PercOfType, x=lc)) + theme_bw() +
  geom_bar(stat="identity", fill="#D30000") +
  geom_point(data=d.c.l, aes(y=PercOfCD, x=lc), colour="#FFFFFF", size=3) +
  scale_y_continuous(labels=percent, breaks=seq(0,1,.1), limits=c(0,1)) +
  scale_x_discrete(labels=c("T"="Tight Out", "L"="Tight In")) +
  labs(subtitle="Panel checking by lathe check orientation",
       x="Lathe check orientation", y=NULL) +
  guides(fill=FALSE, size=FALSE, colour=FALSE)
grid.arrange(pc, pg, pv, pl, ncol=2, nrow=2, top="Observed panel checking by factor and level")
```


Observed panel checking by factor and level
Panel checking by core
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
CC
MDF
PB
Ven
Core type
Panel checking by adhesive
PVA
Soy
UF
Adhesive type
Panel checking by veneer
P−0.706
P−0.604
S−0.564
S−0.508
Veneer type
Panel checking by lathe check orientation
Tight In
Tight Out
Lathe check orientation

Cumulative distribution of observed check density by factor

```
d.nzx <- d.nz %>% group_by(core) %>% mutate(cumdist = cume_dist(checkDensity))
cdc <- ggplot(d.nzx, aes(x=checkDensity, y=cumdist, colour=core)) + theme_bw() +
  geom_point(alpha=.8, shape=1) +
  scale_colour_manual(values=c("#D62F1F","#1E2DD6", "#1ED63D", "#AB1ED6"),
    labels=c("CC", "MDF", "PB", "Ven"), name="") +
  scale_x_continuous(trans="log", breaks=c(10,50,100,500,1000,5000,8000)) +
  theme(legend.position = "bottom") +
  labs(x=expression(paste("Check density (", mm^{2}, "/", m^{2}, ", natural log scaled)")),
       subtitle="Core type",
       y="Proportion of values < x")
d.nzx <- d.nz %>% group_by(glue) %>% mutate(cumdist = cume_dist(checkDensity))
cdg <- ggplot(d.nzx, aes(x=checkDensity, y=cumdist, colour=glue)) + theme_bw() +
  geom_point(alpha=.6, shape=1) +
  scale_colour_manual(values=c("#D62F1F","#1E2DD6", "#1ED63D"),
    labels=c("PVA", "Soy", "UF"), name="") +
  scale_x_continuous(trans="log", breaks=c(10,50,100,500,1000,5000,8000)) +
  theme(legend.position = "bottom") +
  labs(x=expression(paste("Check density (", mm^{2}, "/", m^{2}, ", natural log scaled)")),
       subtitle="Adhesive type",
       y="Proportion of values < x")
d.nzx <- d.nz %>% group_by(veneer) %>% mutate(cumdist = cume_dist(checkDensity))
cdv <- ggplot(d.nzx, aes(x=checkDensity, y=cumdist, colour=veneer)) + theme_bw() +
  geom_point(alpha=.8, shape=1) +
  scale_colour_manual(values=c("#D62F1F","#1E2DD6", "#1ED63D", "#AB1ED6"),
    labels=c("P0.706", "P0.604", "S0.564", "S0.508"), name="") +
  scale_x_continuous(trans="log", breaks=c(10,50,100,500,1000,5000,8000)) +
  theme(legend.position = "bottom") +
  labs(x=expression(paste("Check density (", mm^{2}, "/", m^{2}, ", natural log scaled)")),
       subtitle="Veneer type",
       y="Proportion of values < x")
d.nzx <- d.nz %>% group_by(lc) %>% mutate(cumdist = cume_dist(checkDensity))
cdl <- ggplot(d.nzx, aes(x=checkDensity, y=cumdist, colour=lc)) + theme_bw() +
  geom_point(alpha=.8, shape=1) +
  scale_colour_manual(values=c("#D62F1F","#1E2DD6"),
    labels=c("LSO", "TSO"), name="") +
  scale_x_continuous(trans="log", breaks=c(10,50,100,500,1000,5000,8000)) +
  theme(legend.position = "bottom", legend.box.spacing = unit(.3,"cm")) +
  labs(x=expression(paste("Check density (", mm^{2}, "/", m^{2}, ", natural log scaled)")),
       subtitle="Lathe-check orientation",
       y="Proportion of values < x")
grid.arrange(cdc, cdg, cdv, cdl, nrow=2,
    top="Cumulative distribution of observed check density by factor and level")
```


Cumulative distribution of observed check density by factor and level
Core type
Adhesive type
Veneer type
Lathe−check orientation
Proportion of values < x
1.00
0.75
0.50
0.25
0.00
10
50
100
500
1000
5000
8000
Check density (mm2/m2, natural log scaled)
CC
MDF
PB
Ven
PVA
Soy
UF
P0.706
P0.604
S0.564
S0.508
LSO
TSO

Fitting the model

Here we fit the data to a full model using all two, three, and four-way interactions.

Data

First, we start with the data set. We need to add a few features, including block & whole plot. We start with the *d.c* data frame. We'll copy it to new data frame and add the columns we need (*d.m*).
1. **block** is already contained in the *name* column, we just need to separate the letter value from the column.
2. **wholePlot** is the *glue* * *block* combination.

There should be 765 observations in the dataset and 12 variables.

```
d.m <- d.c %>% filter(checkDensity<18330) %>%
  separate(name, c("block"), remove=FALSE, extra="drop") %>% # create block colum
  unite(wholePlot, block, glue, sep="", remove=FALSE) # create wholePlot column
glimpse(d.m)
```

```
## Observations: 765
## Variables: 13
## $ X            <int> 11, 26, 32, 49, 65, 78, 91, 102, 111, 128, 138, 1...
## $ name         <chr> "A-001", "A-002", "A-003", "A-004", "A-005", "A-0...
## $ wholePlot    <chr> "AU", "AU", "AU", "AU", "AU", "AU", "AU", "AU", "...
## $ block        <chr> "A", "A", "A", "A", "A", "A", "A", "A", "A", "A",...
## $ stage        <int> 22, 24, 17, 21, 24, 24, 24, 22, 18, 22, 19, 21, 2...
## $ checkDensity <dbl> 22.248, 846.724, 227.109, 801.633, 2011.687, 47.0...
## $ combo_string <chr> "S5-L-C-U", "S5-L-V-U", "S4-T-C-U", "P4-T-C-U", "...
## $ veneer       <chr> "S5", "S5", "S4", "P4", "S5", "S5", "P3", "S4", "...
## $ lc           <chr> "L", "L", "T", "T", "T", "L", "T", "T", "L", "L",...
## $ core         <chr> "C", "V", "C", "C", "C", "M", "M", "V", "M", "P",...
## $ glue         <chr> "U", "U", "U", "U", "U", "U", "U", "U", "U", "U",...
## $ checked      <chr> "Checks", "Checks", "Checks", "Checks", "Checks",...
## $ chkd.logical <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1...
```

The model

We will create a full model (all four-way interactions), a three-way model, and a four-way model with out the blocking random effect for the bootstrap model.Importantly, the model is paramterised on the observed means, not the transformed means, and uses a logarithmic link function to the mean instead. Character vectors are automatically coerced to factors.

```
#Each may take up to 1 minute to run.
#full two, three, and four-way interactions
#both random effects
m.full<-cpglmm(checkDensity~glue+veneer+core+lc+
  glue:veneer+glue:core+glue:lc+
  veneer:core+veneer:lc+core:lc+
  glue:veneer:core+glue:veneer:lc+
  glue:core:lc+veneer:core:lc+
  glue:core:lc:veneer+
  (1|block)+(1|wholePlot),link="log",d.m)

#3-way and two-interactions, both random effecits
m.3w <- cpglmm(checkDensity~glue+veneer+core+lc+
  glue:veneer+glue:core+glue:lc+
  veneer:core+veneer:lc+core:lc+
  glue:veneer:core+glue:veneer:lc+
  glue:core:lc+veneer:core:lc+
  (1|block)+(1|wholePlot),link="log",d.m)

#Full without the block random effect. Will be used for bootstrapping.
```

```
m.nb<-cpglmm(checkDensity~glue+veneer+core+lc+
  glue:veneer+glue:core+glue:lc+
  veneer:core+veneer:lc+core:lc+
  glue:veneer:core+glue:veneer:lc+
  glue:core:lc+veneer:core:lc+
  glue:core:lc:veneer+
  (1|wholePlot),link="log",d.m)
```

Some model diagnostics

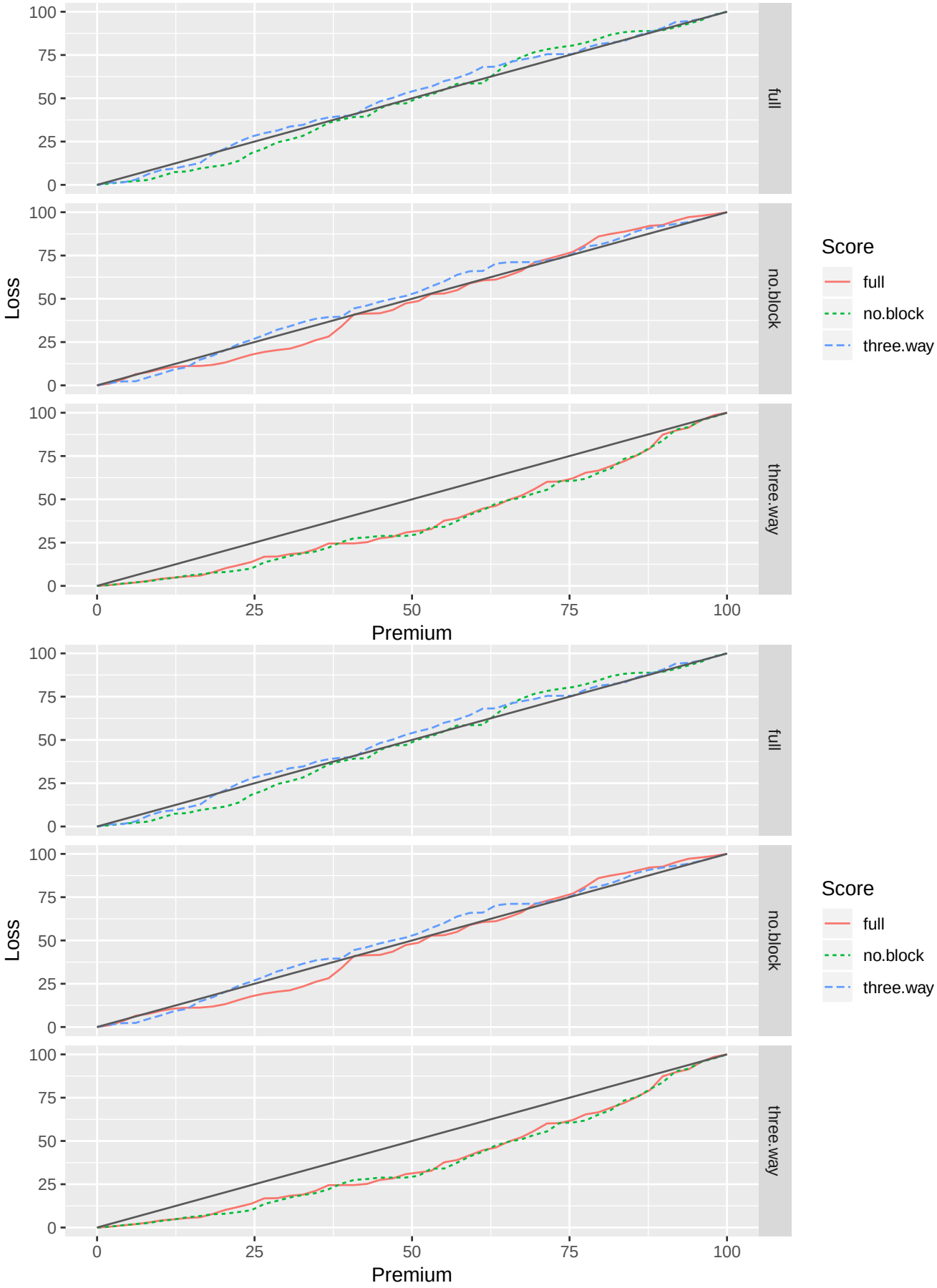
```
cplm::anova(m.full, m.3w, m.nb)
```

	Df	AIC	BIC	logLik	Chisq	Chi Df	Pr(>Chisq)
m.3w	81	6491.295	6867.125	-3164.647	NA	NA	NA
m.nb	98	6477.140	6931.848	-3140.570	48.15441	17	0.0000813
m.full	99	6467.074	6926.422	-3134.537	12.06660	1	0.0005133

```
#summary(m.full) # deviance 6269
#summary(m.3w) # deviance 6329
#drop in deviance 6329-6269
#drop in devience test 1-pchisq(60, 18)
#summary(m.nb) #Suppressed for reporting, takes many pages.

#gini indexes, and plotting the ordered Lorenz curve
d.t <- cbind(cd = d.m$checkDensity, full = fitted(m.full),
             no.block = fitted(m.nb), three.way=fitted(m.3w))
gg <- gini(loss="cd", score=c("full", "no.block", "three.way"), data=d.t)
gg;
```

```
##
## Call:
## gini(loss = "cd", score = c("full", "no.block", "three.way"),
##     data = d.t)
##
## Gini indices:
##             full    no.block  three.way
## full        0.000    1.884    -3.161
## no.block    2.848    0.000    -3.512
## three.way  22.231   23.620     0.000
##
## Standard errors:
##             full   no.block  three.way
## full       0.000  5.301     5.362
## no.block   5.224  0.000     5.219
## three.way  4.805  4.670     0.000
##
## The selected score is full.
```

```
#ordered lorezne curves / loss-premium.
plot(gg);
```



The four-way interactions appear to be significant, and the gini index favors the full model. However, we will use the "no block" model for the bootsrap approach we will later take to allow for resampling.

Extract and calculate details from the model

This is the info based on the observed data and the model fit to the observed data (rather than the bootstrap).

```
d.m$pred <- predict(m.full) #predicted data
#model matrix of fixed effects only
mm <- model.matrix(m.full)
#variances of fixed effects
d.m$var.f <- diag(mm %*% tcrossprod(vcov(m.full),mm))
#d.sum is the summary data frame we can use to plot the data
d.sum <- d.m %>% select(combo_string, checkDensity, var.f, pred) %>%
  group_by(combo_string) %>%
  summarise(med.o = median(checkDensity), #observed
            mu.o = mean(checkDensity),
            sd.o = sd(checkDensity),
            n.obs = n(),
            med.p = median(pred), #predicted
            mu.p = mean(pred),
            se.p = sqrt(sum(var.f) / n())) #unexpectedly and unbelievably small

#calculate the hw of the 95% CI
#Also, order by predicted mean, and add factor labels back in
d.sum <- d.sum %>% arrange(mu.p) %>%
  separate(combo_string, c("veneer", "lc", "core", "glue"),
           remove=FALSE)  %>%
  mutate(hw = 1.96 * se.p)
#using the hw, calculate the upper and lower bounds of the CI
#limit the lower bound to zero, add an index
d.sum <- d.sum %>% mutate(upr = mu.p+hw, lwr = ifelse(mu.p-hw < 0, 0, mu.p-hw),
                          index = 1:nrow(d.sum))
glimpse(d.sum)
```

```
## Observations: 96
## Variables: 16
## $ combo_string <chr> "P3-L-P-S", "P4-T-M-P", "S5-T-P-U", "P4-T-M-U", "...
## $ veneer       <chr> "P3", "P4", "S5", "P4", "P3", "S4", "P4", "P4", "...
## $ lc           <chr> "L", "T", "T", "T", "L", "L", "L", "L", "T", "L",...
## $ core         <chr> "P", "M", "P", "M", "M", "C", "P", "V", "P", "P",...
## $ glue         <chr> "S", "P", "U", "U", "P", "P", "U", "P", "U", "P",...
## $ med.o        <dbl> 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0...
## $ mu.o         <dbl> 0.00000, 2.90325, 10.21750, 7.10650, 14.72963, 4....
## $ sd.o         <dbl> 0.000000, 8.211631, 28.899454, 14.776273, 41.6616...
## $ n.obs        <int> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8...
## $ med.p        <dbl> 6.972597e-11, 5.067636e-01, 3.811722e+00, 4.16742...
## $ mu.p         <dbl> 1.932787e-10, 1.387614e+00, 6.371043e+00, 6.96556...
## $ se.p         <dbl> 181.9367665, 1.5328455, 1.0863052, 1.0682502, 1.1...
## $ hw           <dbl> 356.596062, 3.004377, 2.129158, 2.093770, 2.18916...
## $ upr          <dbl> 356.596062, 4.391991, 8.500201, 9.059339, 9.22959...
## $ lwr          <dbl> 0.000000, 0.000000, 4.241885, 4.871798, 4.851261,...
## $ index        <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15...
```

The confidence intervals simply don't make sense from this method, so we won't use them (they are far too small). In previous versions of the *cplm* package, this method extracted reasonable, but still to small, confidence intervals.

Bootstrap model parameters, values, and confidence intervals for them

We need to try another method to extract confidence intervals. We select bootstrapping because we don't have to make assumptions about the form of any distribution and the intervals we create will be based on the data we observe.

This code is not run here. Running 10,000 simulations (R=10000), may take more than 24 hours.

Changes to multicore may be required to run on Windows. The following was run as a shell script as with a shell argument of 10000:

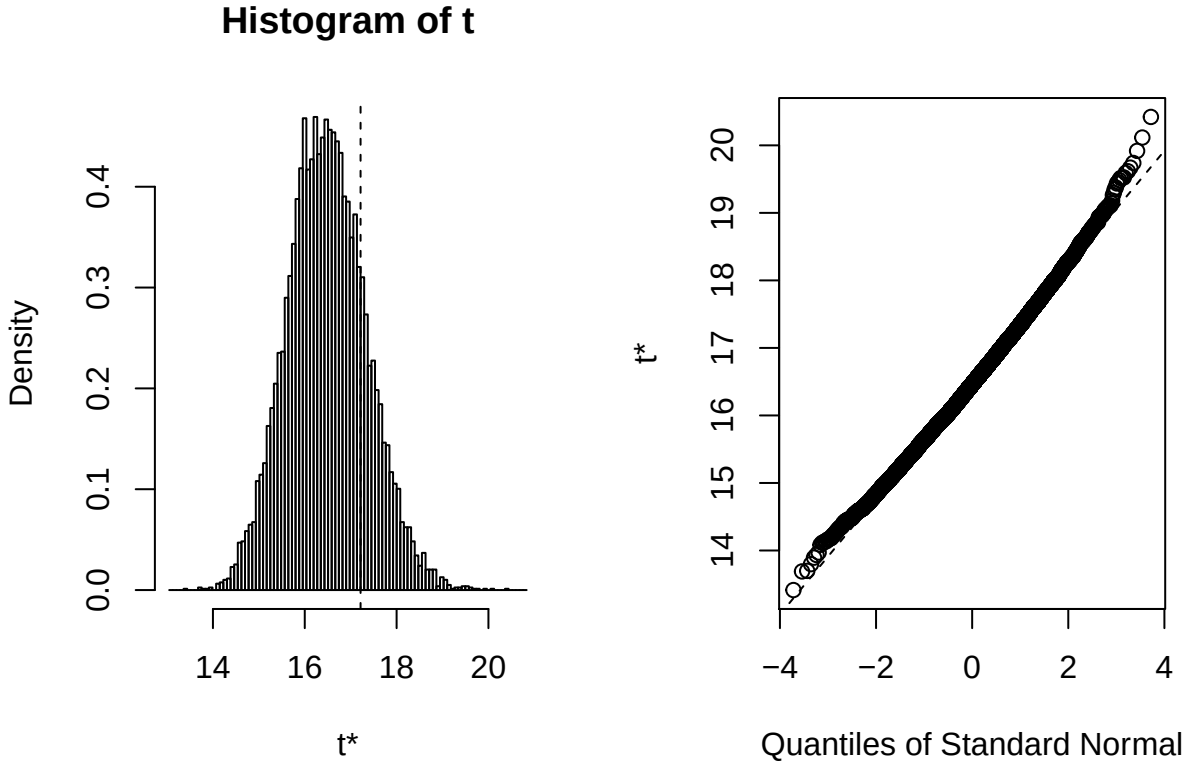
Rscript mapBoot.r 10000

```
library(boot)
library(cplm)
library(dplyr)

#arguments from the command line, they are read as strings!
args <- commandArgs(TRUE) #args[1] = number of reps -- anything else is ignored.

#timing
start.time <- Sys.time()

#get data
d.boot <- read.csv("Burnardetal2017_dataForBootstrap.csv",
                   stringsAsFactors=FALSE, header=TRUE)

#the function for boot that returns the statistic of interest
#the stat of interest is the predicted mean for each treatment combo
#we retrieve it by refitting the model for each sample of the observed data
#then predicting values for new observations, and taking the mean of those
#predictions for each treatment combo.
boot.f <- function(data, i) {
  dxb <- data[i,]
  #fit the model with the resampled data
  fit <- cpglmm(checkDensity~glue+veneer+core+lc+
                  glue:veneer+glue:core+glue:lc+
                  veneer:core+veneer:lc+core:lc+
                  glue:veneer:core+glue:veneer:lc+
                  glue:core:lc+veneer:core:lc+
                  (1|wholePlot), link="log", data=dxb)
  #add predictions based on the newly fit model
  dxb$pred <- cplm::predict(fit)
  #calculate means for each combo
  x <- dxb %>% select(combo_string, pred)
           %>% group_by(combo_string) %>% summarise(val.mu = mean(pred))
  #model info. we include the index parameter, the dispersion parameter,
  #residual variance, and wholePlot variance
  ret <- c(x$val.mu, fit$p, fit$phi,
           VarCorr(fit)$wholePlot[1], attr(VarCorr(fit), "sc")^2)
  #return the model info and mean combo values
  return(ret)
}
#the call to the bootstrap function
#Using strata to sample within treatment combinations
bt <- boot(data=d.boot, statistic=boot.f, R=as.numeric(args[1]),
           strata=factor(d.boot$combo_string), parallel = "multicore", ncpus=8)
#save the R boot object. This allows you to import the object into a new session
saveRDS(bt, "btObj-y.rds")
```

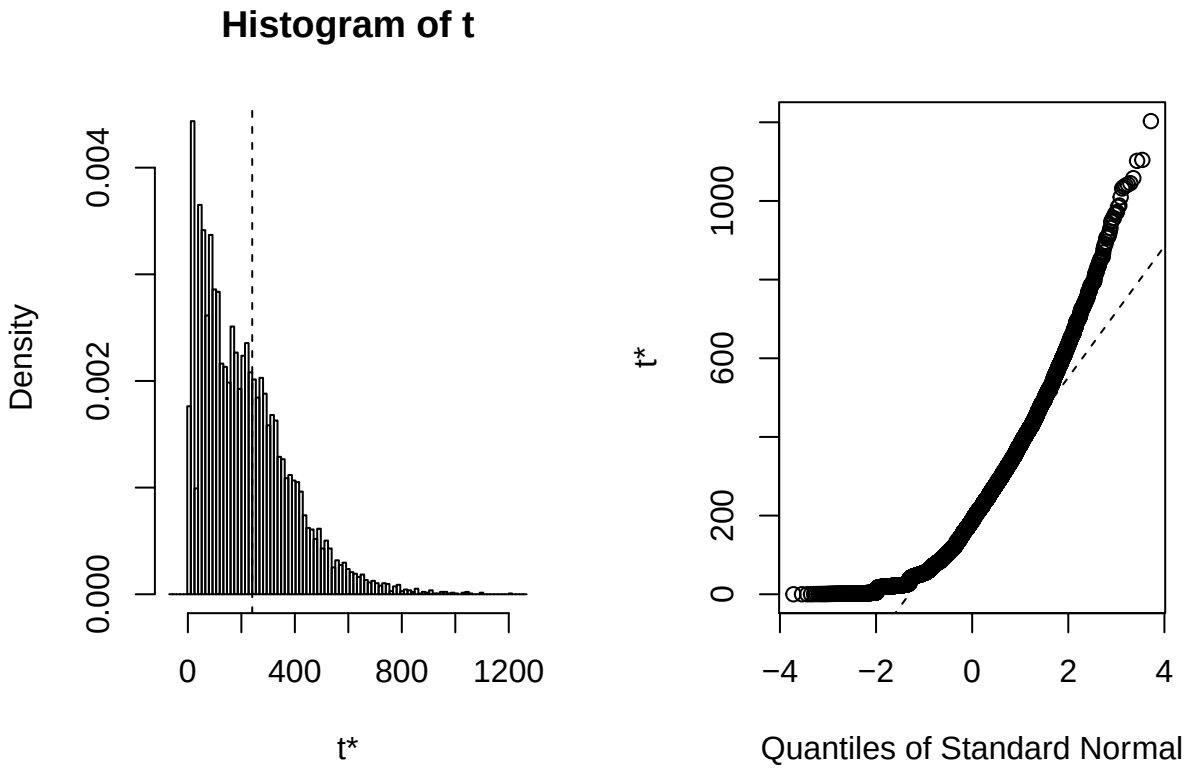
```
#create a data frame to store the results
z <- data_frame(boot.mu=1, bca.lwr=1, bca.upr=1)
#loop through all the store statistics (means from each bootstrapped model)
#extract lower and upper CIs
for(i in 1:length(bt$t0)) {
  y <- boot.ci(bt, conf=0.95, type=c("bca"), index=i)
  z <- bind_rows(z, data_frame(boot.mu=bt$t0[i], bca.lwr=y$bca[4], bca.upr=y$bca[5]))
}
z <-z [-1,]
#split model parameters out, then add names
z.model <- z[97:100,]
z.model$Parameter <- c("Index", "Dispersion", "WholePlot", "Residual")
#split predictions out, add names, factor levels, order by prediction, an index
z.pred <- z[1:96,]
z.pred$combo_string <- levels(factor(bt$strata))
z.pred <- z.pred %>% separate(combo_string, c("veneer", "lc", "core", "glue"),
           remove=FALSE)  %>% arrange(boot.mu) %>% mutate(index = seq(1,96,1))

#write the files for later use
write.csv(file="BootPredictions-y.csv", z.pred)
write.csv(file="BootModel-y.csv", z.model)

#timing
end.time <- Sys.time();
elapsed <- end.time - start.time;
tm <- data_frame(start=start.time, end=end.time, run=elapsed, sims=as.numeric(args[1]))
write.csv(file="runInfo-y.csv", tm)
```

Examining the bootstrap data

```
#read the data
bt <- readRDS("Burnardetal2018_bootObject10k.rds") #boostrap object returned by boot function
boot.model <- read.csv("Burnardetal2018_bootModel10k.csv") #model data
boot.pred <- read.csv("Burnardetal2018_bootPredictions10k.csv") #prediction data
#as examples, look at a few items from the boostrap model
plot(bt, index=100) #residual variance of the model
```




```
plot(bt, index=17) #treatment P3-T-M-S (to find out: levels(factor(bt$strata))[17] )
```



```
levels(factor(bt$strata))[17]
```

```
## [1] "P3-T-M-S"
```

```
print(bt, index=17)
```

```
##
## STRATIFIED BOOTSTRAP
##
##
## Call:
## boot(data = d.boot, statistic = boot.f, R = as.numeric(args[1]),
##     strata = factor(d.boot$combo_string), parallel = "multicore",
##     ncpus = 8)
##
##
## Bootstrap Statistics :
##      original    bias    std. error
## t17* 240.6482 -22.32632    166.5947
```

```
#add observed means
boot.pred <- bind_cols(boot.pred, as.data.frame(d.sum %>%
                                  select(mu.o), by="combo_string"))

#identify specimens with observed means outside the 95 % CI of the mean
boot.pred %>% filter(mu.o < bca.lwr | mu.o > bca.upr)
```

X	boot.mu	bca.lwr	bca.upr	combo_string	veneer	lc	core	glue	index	mu.o
1	0.0000	0.0000	0.000	P3-L-P-S	P3	L	P	S	1	0.0000
90	564.6004	297.3739	1624.755	S5-L-V-S	S5	L	V	S	90	230.3705

```
#create some columns to help us summarise the data.
boot.pred <- boot.pred %>% mutate(bins4 = ntile(boot.mu, 4), bins5 = ntile(boot.mu, 5),
                                  bin10 = ntile(boot.mu, 10))

#tables of quartile membership for each factor by level as a fraction
table(boot.pred$veneer,boot.pred$bins4)/24
```

```
##
##              1         2         3         4
##   P3 0.3333333 0.2500000 0.1250000 0.2916667
##   P4 0.2916667 0.2083333 0.2916667 0.2083333
##   S4 0.2083333 0.3750000 0.2500000 0.1666667
##   S5 0.1666667 0.1666667 0.3333333 0.3333333
```

```
table(boot.pred$core,boot.pred$bins4)/24
```

```
##
##              1          2          3          4
##   C 0.16666667 0.20833333 0.29166667 0.33333333
##   M 0.29166667 0.33333333 0.29166667 0.08333333
##   P 0.50000000 0.29166667 0.16666667 0.04166667
##   V 0.04166667 0.16666667 0.25000000 0.54166667
```

```
table(boot.pred$glue,boot.pred$bins4)/32
```

```
##
##           1       2       3       4
##   P 0.37500 0.28125 0.18750 0.15625
##   S 0.03125 0.25000 0.37500 0.34375
##   U 0.34375 0.21875 0.18750 0.25000
```

```
table(boot.pred$lc,boot.pred$bins4)/48
```

```
##
##             1         2         3         4
##   L 0.2708333 0.2500000 0.2291667 0.2500000
##   T 0.2291667 0.2500000 0.2708333 0.2500000
```

```
boot.pred %>% group_by(bins4) %>% summarise(maxCDperBin = max(boot.mu))
```

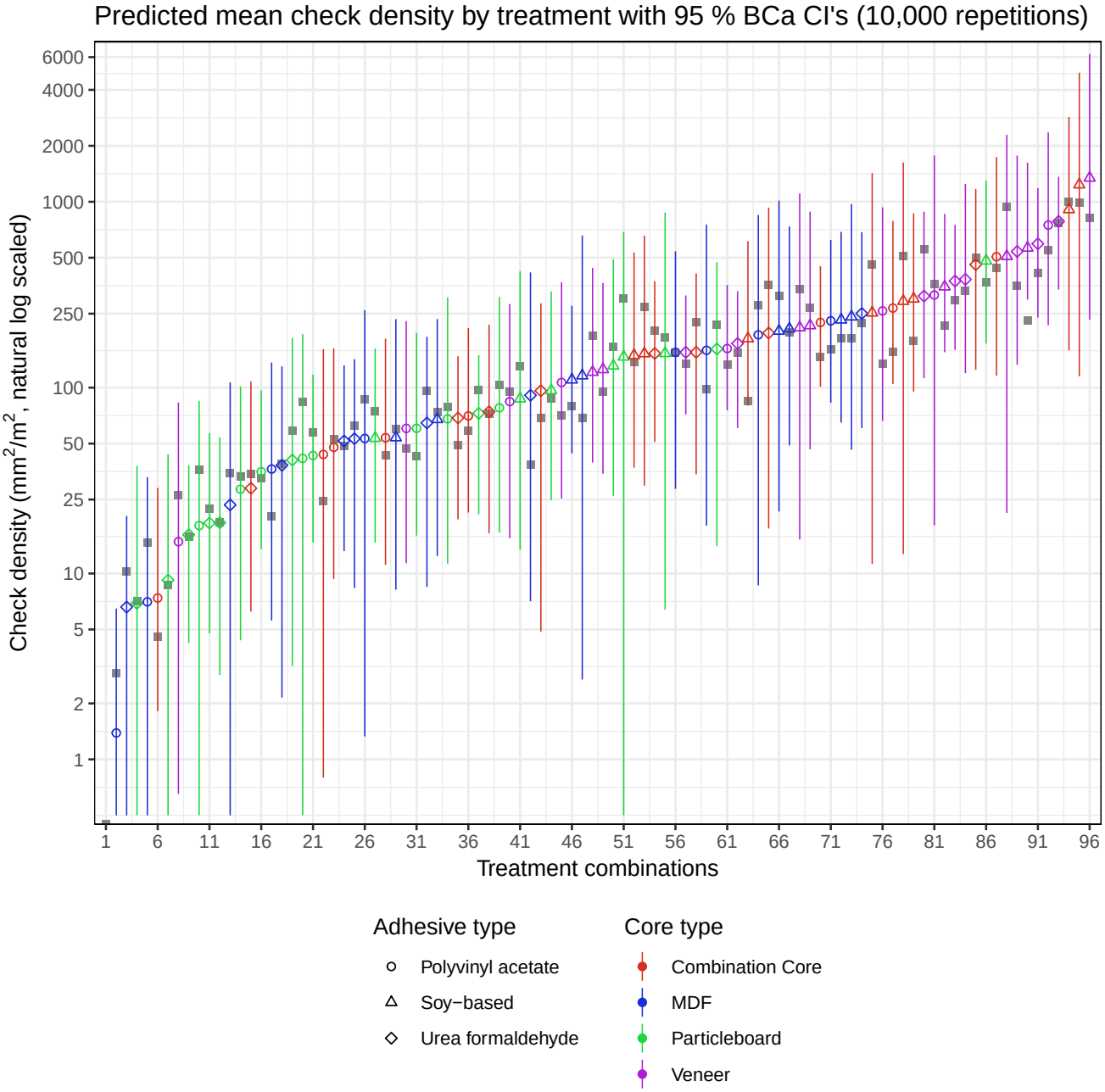
bins4	maxCDperBin
1	51.67433
2	121.19518
3	231.63613
4	1343.19049

```
write.csv(file="Burnardetal2018_MapleCheckingResultsData.csv", boot.pred)
```

Plot the ordered predicted means with 95 % confidence intervals (a)

Log scaled on the y-axis to "zoom in" on the smaller values in the distribution. We add a small amount to the 0 value lower confidence inverval bounds to get them to draw on this chart. We explain that in the figure caption. [produces some errors because of the transformation of 0 values: log(0) = -inf.]

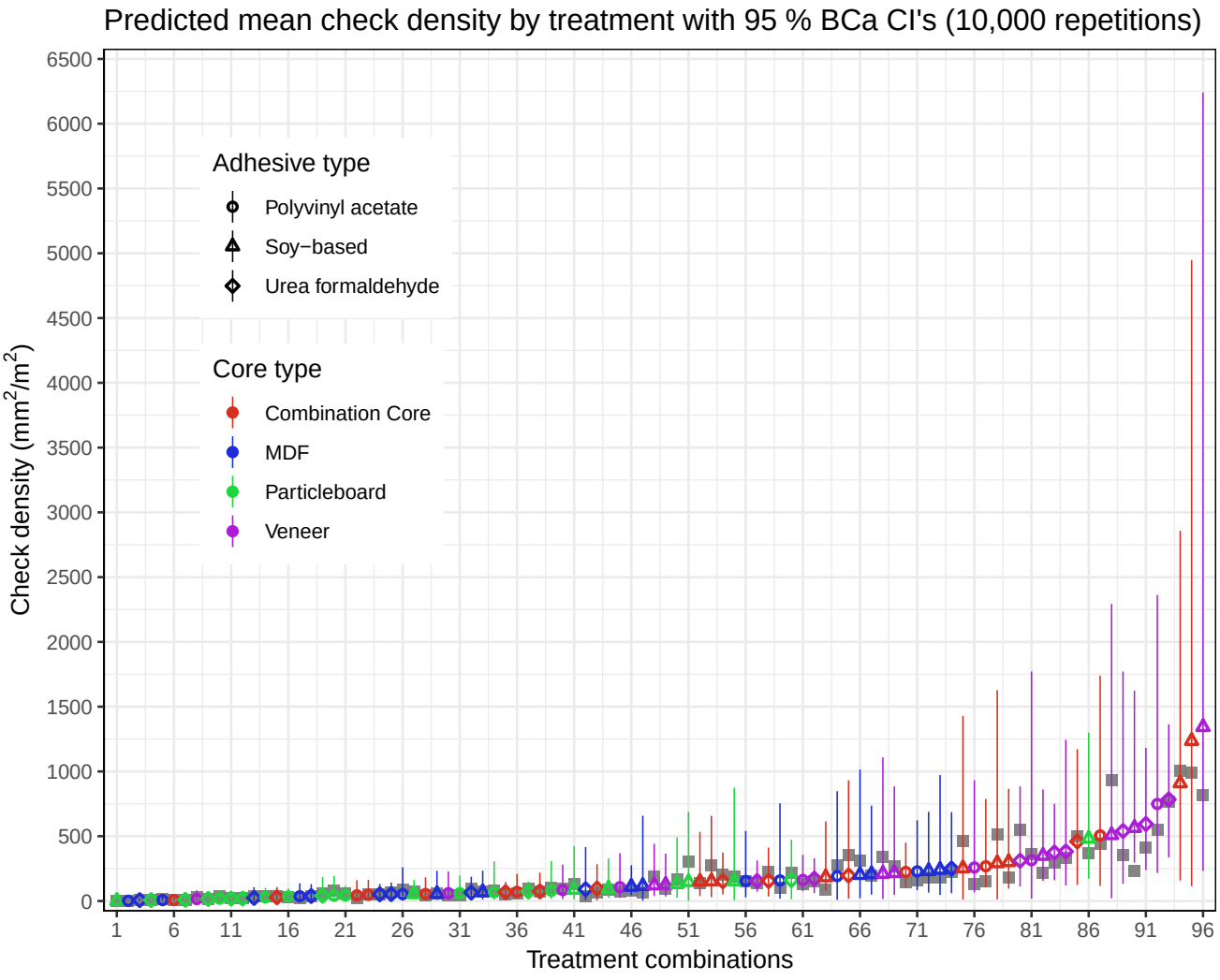
```
ggplot(boot.pred, aes(y=boot.mu, x=index, ymax=bca.upr,
                      ymin=ifelse(bca.lwr < 0.5, bca.lwr+0.5, bca.lwr),
                      shape=glue, colour=core)) + theme_bw() +
  geom_point(data=boot.pred, aes(y=mu.o, x=index),
             alpha=.9, size=1.5, shape=15, colour="#777777") +
  geom_point(size=1.5) +
  geom_linerange(size=.3) +
  scale_y_continuous(trans="log", limits=c(.5,6500), expand=c(0.012,0),
                     breaks=c(0,1,2,5,10,25,50,100,250,500,1000,2000,4000,6000)) +
  scale_x_continuous(breaks=seq(1,96,5), limits=c(1,96), expand=c(0.012,0)) +
  scale_shape_manual(values=c(1, 2, 5),
      labels=c("Polyvinyl acetate", "Soy-based", "Urea formaldehyde"),
      name="Adhesive type")  +
  scale_colour_manual(values=c("#D62F1F","#1E2DD6", "#1ED63D", "#AB1ED6"),
      labels=c("Combination Core", "MDF", "Particleboard", "Veneer"),
      name="Core type") +
  scale_linetype_manual(values=c(1,2,3),
      labels=c("Polyvinyl acetate", "Soy-based", "Urea formaldehyde"),
      name="Adhesive type") +
  labs(
    title="Predicted mean check density by treatment with 95 % BCa CI's (10,000 repetitions)",
    x="Treatment combinations",
    y=expression(paste("Check density (", mm^{2}, "/", m^{2}, ", natural log scaled)"))) +
  theme(legend.key=element_rect(fill="#FFFFFF"), legend.position="bottom",
        legend.direction="vertical", panel.border=element_rect(colour="#000000", fill=NA))
```


Predicted mean check density by treatment with 95 % BCa CI's (10,000 repetitions)
Check density (mm2/m2, natural log scaled)
6000
4000
2000
1000
500
250
100
50
25
10
5
2
1
1
6
11
16
21
26
31
36
41
46
51
56
61
66
71
76
81
86
91
96
Treatment combinations
Adhesive type
Polyvinyl acetate
Soy−based
Urea formaldehyde
Core type
Combination Core
MDF
Particleboard
Veneer

Plot the ordered predicted means with 95 % confidence intervals (b)

Natural scaled chart for comparison. We will overlay them in the text for close comparison.

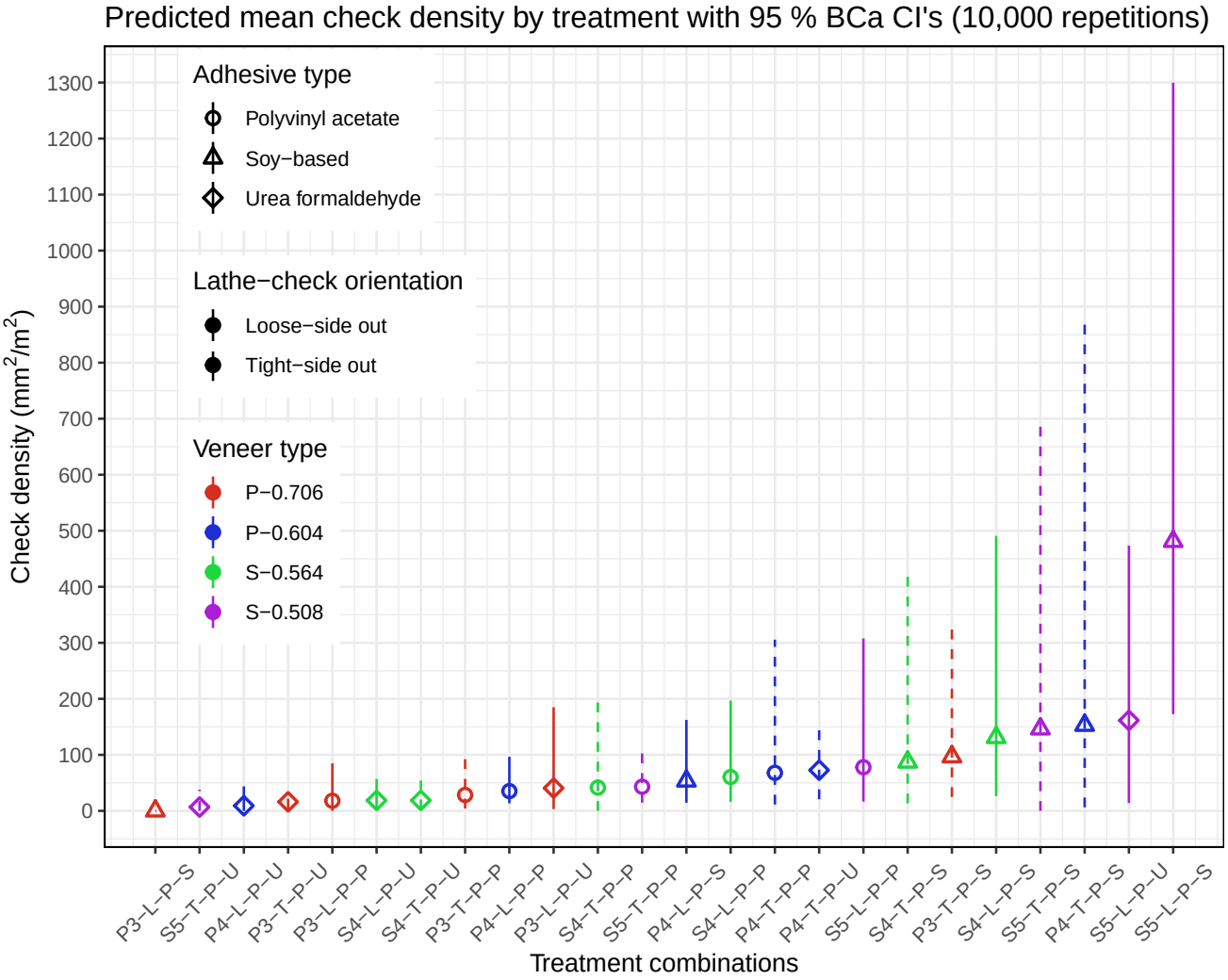
```
ggplot(boot.pred, aes(y=boot.mu, x=index, ymax=bca.upr, ymin=bca.lwr,
                      shape=glue, colour=core)) + theme_bw() +
  geom_point(data=boot.pred, aes(y=mu.o, x=index),
             alpha=.9, size=2, shape=15, colour="#777777") +
  geom_pointrange(size=0.3) +
  scale_y_continuous(limits=c(0,6500), breaks=seq(0,6500,500), expand=c(0.012,0)) +
  scale_x_continuous(breaks=seq(1,96,5), limits=c(1,96), expand=c(0.012,0)) +
  scale_shape_manual(values=c(1, 2, 5),
      labels=c("Polyvinyl acetate", "Soy-based", "Urea formaldehyde"),
      name="Adhesive type")  +
  scale_colour_manual(values=c("#D62F1F","#1E2DD6", "#1ED63D", "#AB1ED6"),
      labels=c("Combination Core", "MDF", "Particleboard", "Veneer"),
      name="Core type") +
  scale_linetype_manual(values=c(1,2,3),
      labels=c("Polyvinyl acetate", "Soy-based", "Urea formaldehyde"),
      name="Adhesive type") +
  labs(
    title="Predicted mean check density by treatment with 95 % BCa CI's (10,000 repetitions)",
    x="Treatment combinations",
    y=expression(paste("Check density (", mm^{2}, "/", m^{2}, ")"))) +
  theme(legend.key=element_rect(fill="#FFFFFF"), legend.position=c(0.2,0.65),
        panel.border=element_rect(colour="#000000", fill=NA))
```



Plot the ordered predicted means with 95 % confidence intervals for particleboard core

We will now use only the particleboard panels as an example. First we will create a new data frame and modify it with a new index.

```
#create subsets for each core type with numeric indexes for each
pred.pb <- boot.pred %>% filter(core == "P") %>% mutate(index = seq(1:24))
pred.mdf <- boot.pred %>% filter(core == "M") %>% mutate(index = seq(1:24))
pred.v <- boot.pred %>% filter(core == "V") %>% mutate(index = seq(1:24))
pred.cc <- boot.pred %>% filter(core == "C") %>% mutate(index = seq(1:24))
#plot for particleboard -- adjust y limits for each,
#or make 0 to 6300 for full range
ggplot(pred.pb, aes(y=boot.mu, x=index, ymax=bca.upr, ymin=bca.lwr,
                      shape=glue, colour=veneer, linetype=lc)) + theme_bw() +
  geom_pointrange(size=0.5) +
  scale_y_continuous(limits=c(0,1300), breaks=seq(0,1300, 100)) +
  scale_x_continuous(breaks=seq(1,24,1), limits=c(1,24),
                     labels=pred.pb$combo_string) +
  scale_shape_manual(values=c(1, 2, 5),
      labels=c("Polyvinyl acetate", "Soy-based", "Urea formaldehyde"),
      name="Adhesive type")  +
  scale_color_manual(values=c("#D62F1F","#1E2DD6", "#1ED63D", "#AB1ED6"),
    labels=c("P-0.706", "P-0.604", "S-0.564", "S-0.508"),
    name="Veneer type") +
  scale_linetype_manual(values=c(1,2,3),
      labels=c("Loose-side out", "Tight-side out"),
      name="Lathe-check orientation") +
  labs(
    title="Predicted mean check density by treatment with 95 % BCa CI's (10,000 repetitions)",
    x="Treatment combinations",
    y=expression(paste("Check density (", mm^{2}, "/", m^{2}, ")"))) +
  theme(legend.key=element_rect(fill="#FFFFFF"), legend.position=c(0.2,0.625),
        panel.border=element_rect(colour="#000000", fill=NA),
        axis.text.x=element_text(angle=45, vjust=.5))
```


Predicted mean check density by treatment with 95 % BCa CI's (10,000 repetitions)
Adhesive type
Polyvinyl acetate
Soy−based
Urea formaldehyde
Lathe−check orientation
Loose−side out
Tight−side out
Veneer type
P−0.706
P−0.604
S−0.564
S−0.508
Check density (mm²/m²)
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
P3−L−P−S
S5−T−P−U
P4−L−P−U
P3−T−P−U
P3−L−P−P
S4−L−P−U
S4−T−P−U
P3−T−P−P
P4−L−P−P
P3−L−P−U
S4−T−P−P
S5−T−P−P
P4−L−P−S
S4−L−P−P
P4−T−P−P
P4−T−P−U
S5−L−P−P
S4−T−P−S
P3−T−P−S
S4−L−P−S
S5−T−P−S
P4−T−P−S
S5−L−P−U
S5−L−P−S
Treatment combinations

Environment

```
sessionInfo()
```

```
## R version 3.5.1 (2018-07-02)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS High Sierra 10.13.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/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] splines   stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] bindrcpp_0.2.2  boot_1.3-20     cplm_0.7-7      Matrix_1.2-14
##  [5] coda_0.19-1     scales_0.5.0    gridExtra_2.3   forcats_0.3.0
##  [9] stringr_1.3.1   dplyr_0.7.6     purrr_0.2.5     readr_1.1.1
## [13] tidyr_0.8.1     tibble_1.4.2    ggplot2_3.0.0   tidyverse_1.2.1
##
## loaded via a namespace (and not attached):
##  [1] statmod_1.4.30   biglm_0.9-1      tidyselect_0.2.4 reshape2_1.4.3
##  [5] haven_1.1.2      lattice_0.20-35  colorspace_1.3-2 stats4_3.5.1
##  [9] htmltools_0.3.6  yaml_2.2.0       rlang_0.2.1      pillar_1.3.0
## [13] glue_1.3.0       withr_2.1.2      tweedie_2.3.2    modelr_0.1.2
## [17] readxl_1.1.0     bindr_0.1.1      plyr_1.8.4       munsell_0.5.0
## [21] gtable_0.2.0     cellranger_1.1.0 rvest_0.3.2      evaluate_0.11
## [25] labeling_0.3     knitr_1.20       highr_0.7        broom_0.5.0
## [29] Rcpp_0.12.18     backports_1.1.2  jsonlite_1.5     hms_0.4.2
## [33] digest_0.6.15    stringi_1.2.4    grid_3.5.1       rprojroot_1.3-2
## [37] cli_1.0.0        tools_3.5.1      magrittr_1.5     lazyeval_0.2.1
## [41] crayon_1.3.4     pkgconfig_2.0.1  xml2_1.2.0       lubridate_1.7.4
## [45] minqa_1.2.4      assertthat_0.2.0 rmarkdown_1.10   httr_1.3.1
## [49] rstudioapi_0.7   R6_2.2.2         nlme_3.1-137     compiler_3.5.1
```

References

Burnard, Michael D. 2012. "Key Factors Influencing Checking in Maple Veneered Decorative Hardwood Plywood."

Burnard, Michael David, Lech Muszyński, Scott Leavengood, Lisa Ganio, and Michael David Burnard. 2018. "An optical method for rapid examination of check development in decorative plywood panels." *Eur. J. Wood Wood Prod.* 0 (0). Springer Berlin Heidelberg: 0. doi:10.1007/s00107-018-1327-7.

Zhang, Yanwei. 2013. "Likelihood-based and Bayesian methods for Tweedie compound Poisson linear mixed models." *Stat. Comput.* 23 (6): 743–57. doi:10.1007/s11222-012-9343-7.