Problem 2

Code ▾

Hide

```
library(quantmod)
```

```
Loading required package: xts
Loading required package: zoo

Attaching package: 'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric


Attaching package: 'xts'

The following objects are masked from 'package:dplyr':

    first, last

Loading required package: TTR
Version 0.4-0 included new data defaults. See ?getSymbols.
Learn from a quantmod author: https://www.datacamp.com/courses/importing-and-managing-financial-data-in-r
```

Problem 2 - Chepurny

1.

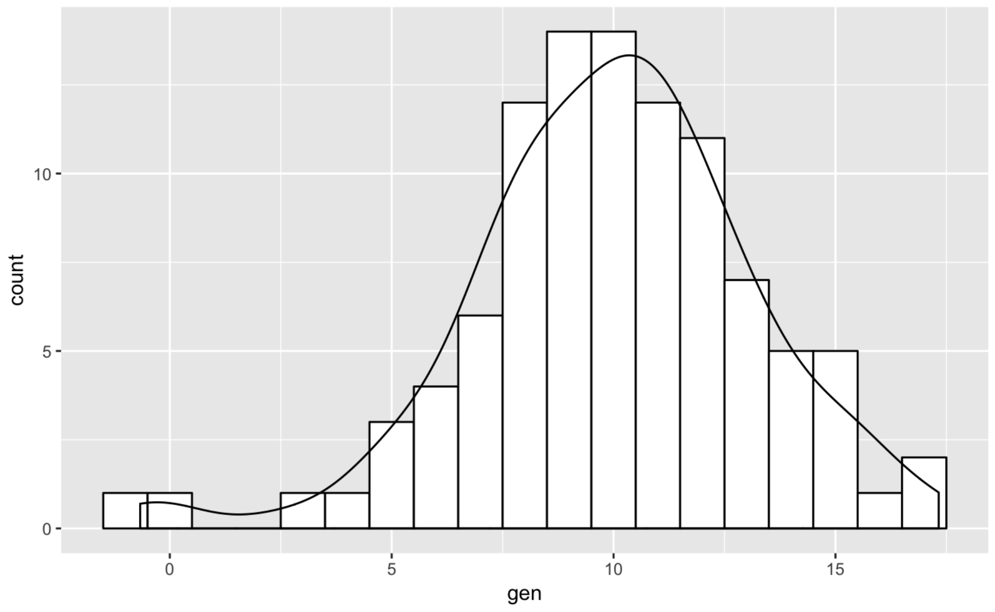
Hide

```
set.seed(17)
normal <- rnorm(100, mean=10, sd=3)
```

Histogram vs Density plot

Hide

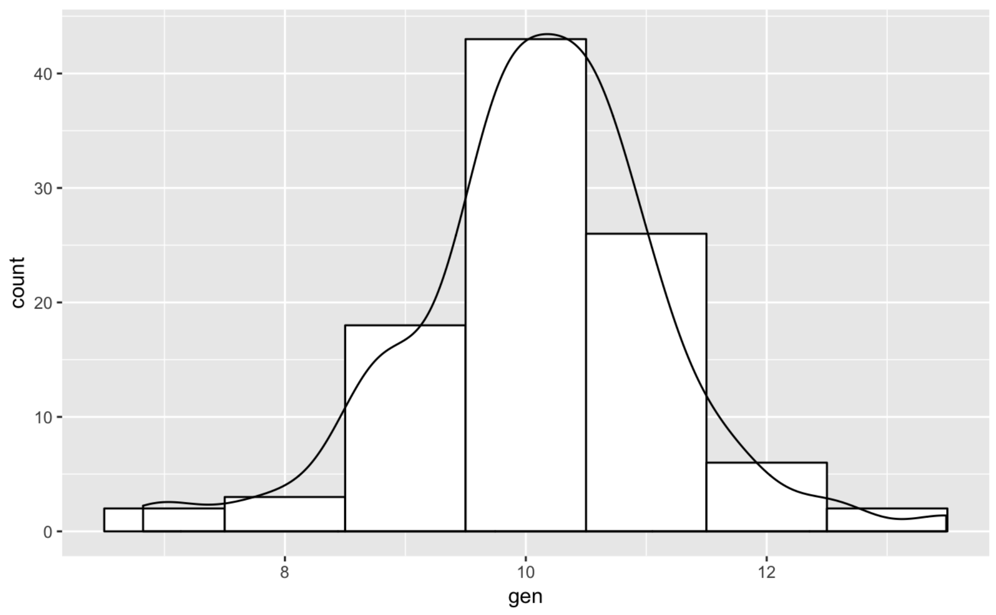
```
data <- data.frame(gen=normal)
ggplot(data, aes(x=gen)) +
  geom_histogram(binwidth=1, colour="black", fill="white") +
  geom_density(aes(y=..count..))
```



T5 distribution vs density plot

Hide

```
t_dist <- data.frame(gen=rt(100, 5)) %>% mutate(gen=10 + gen * (3/5)^(1/3))
ggplot(t_dist, aes(x=gen)) +
  geom_histogram(binwidth=1, colour="black", fill="white") +
  geom_density(aes(y=..count..))
```

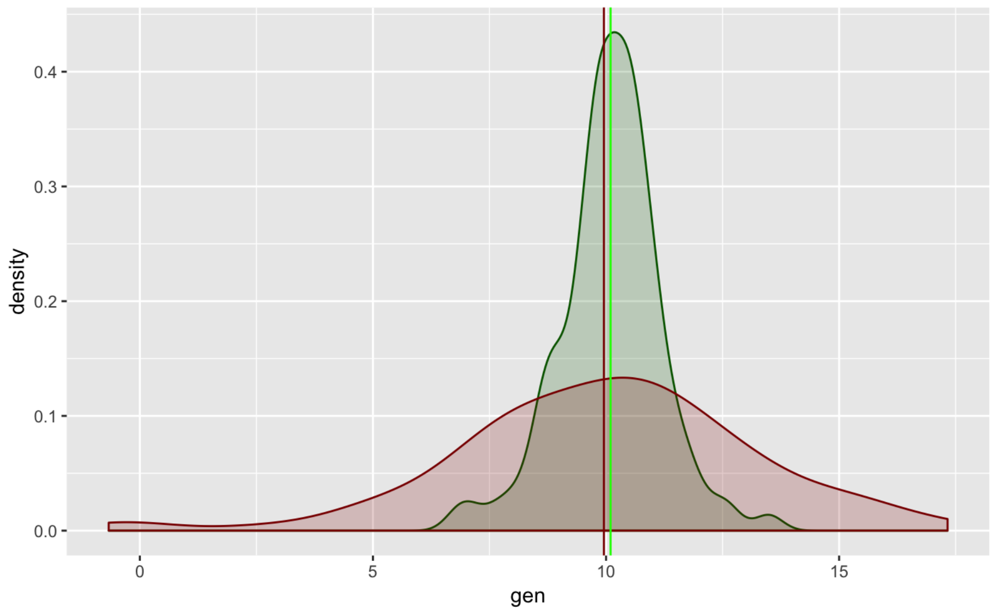


T Density plot (green) vs N(10, 9) density plot

As we can see, N distribution has higher range of the values and lower density around mean value, although mean values of both distributions are close. Modifying the T distribution (scaling along y axis and shifting along x axis) helped to adjust T distribution mean closer to mean of N distribution. Applying modifications to the distributions can help us to vary mean, density and other measures.

Hide

```
ggplot() +
  geom_density(data=t_dist, aes(x=gen), color='darkgreen', fill='darkgreen', alpha=0.2) +
  geom_density(data=data, aes(x=gen), color='darkred', fill='darkred', alpha=0.2) +
  geom_vline(xintercept = mean(t_dist$gen), color='green') +
  geom_vline(xintercept = mean(data$gen), color='darkred')
```

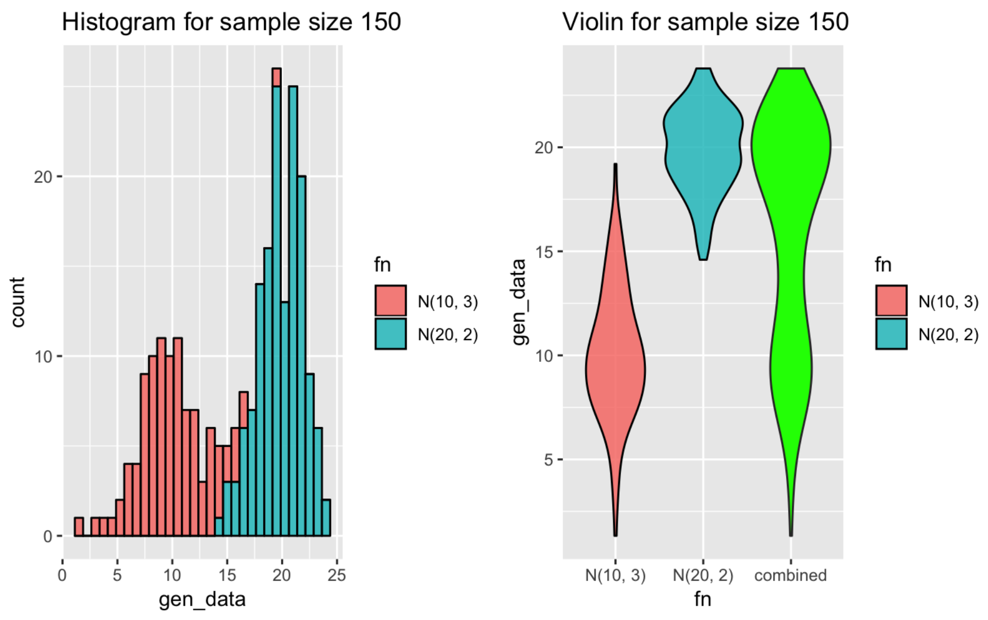


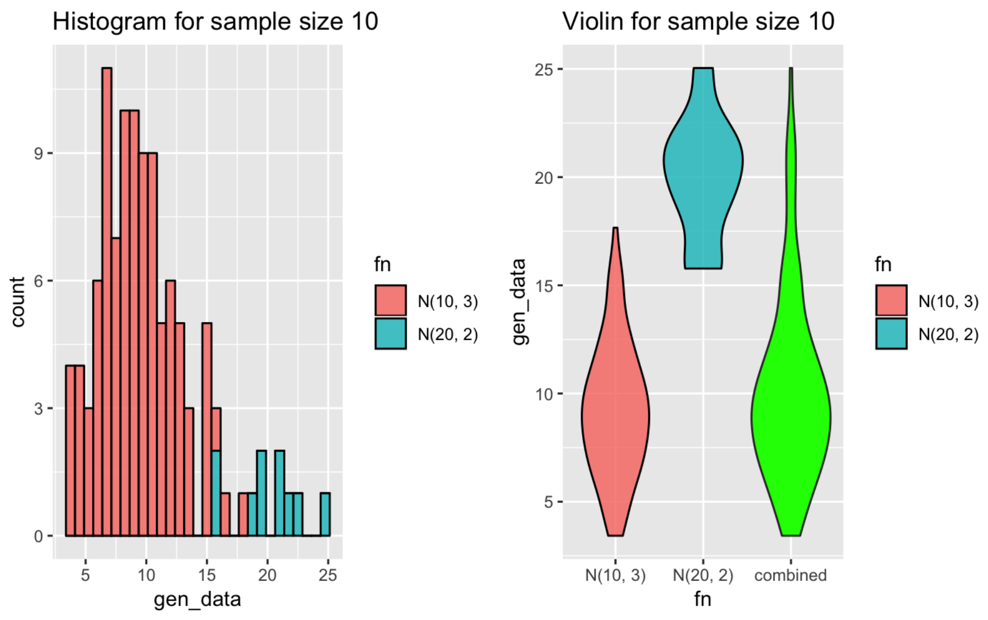
2.

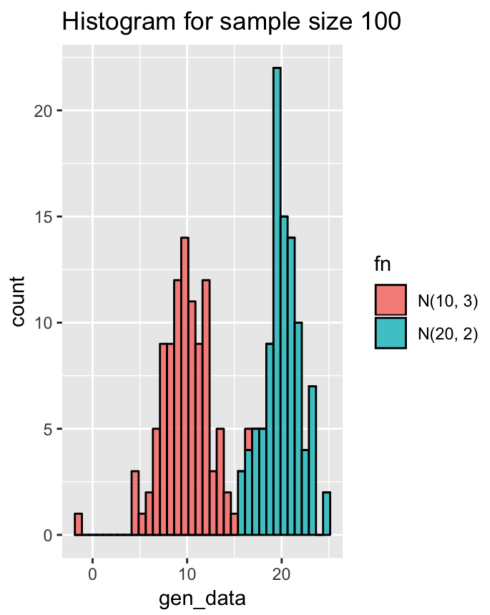
Box plots and Violin plots of combinations N(10, 3) and N(20, 2) with different size of m (number of samples from N(20, 2))

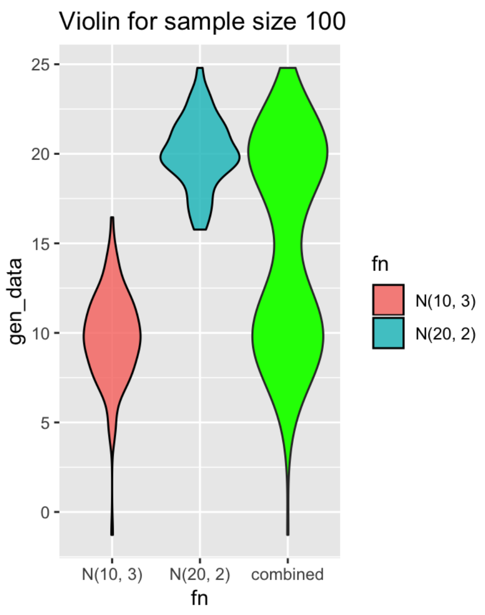
Hide

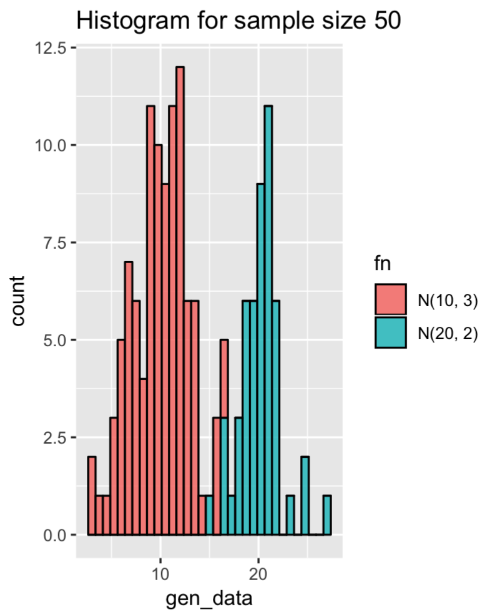
```
#Initial normal dist
build_plots_for_m <- function(m) {
  n1 <- data.frame(gen_data = rnorm(100, mean=10, sd=3), fn='N(10, 3)')
  mix <- data.frame(gen_data = rnorm(m, mean=20, sd=2), fn='N(20, 2)')

  combined <- rbind(n1, mix)

  hist = ggplot(data=combined, aes(x=gen_data, fill=fn)) +
    ggtitle(paste("Histogram for sample size", m)) +
    geom_histogram(binwidth = 0.75, color='black', alpha=0.8)

  violin = ggplot(data=combined, aes(x=fn,y=gen_data, fill=fn)) +
    ggtitle(paste("Violin for sample size", m)) +
    geom_violin(color='black', alpha=0.8) +
    geom_violin(aes(x='combined', y=combined$gen_data), fill='green')

  return(list(hist=hist, violin=violin))
}
for (m in c(10, 50, 100, 150, 200)) {
  plts = build_plots_for_m(m)
  grid.arrange(plts$hist, plts$violin, nrow=1)
}
```

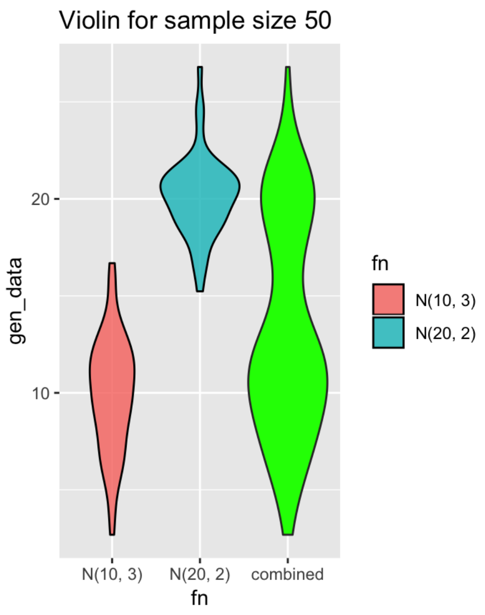


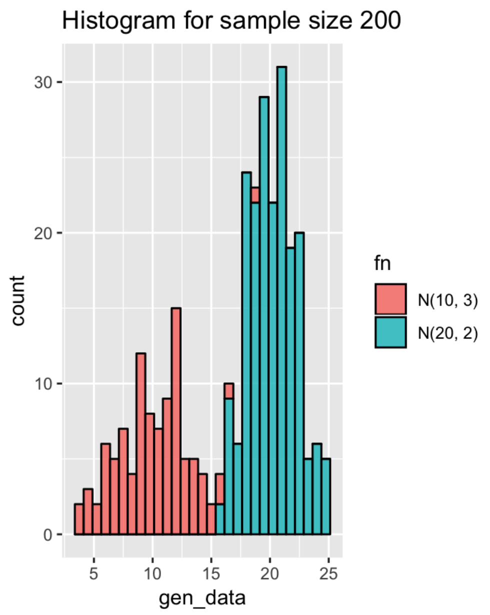


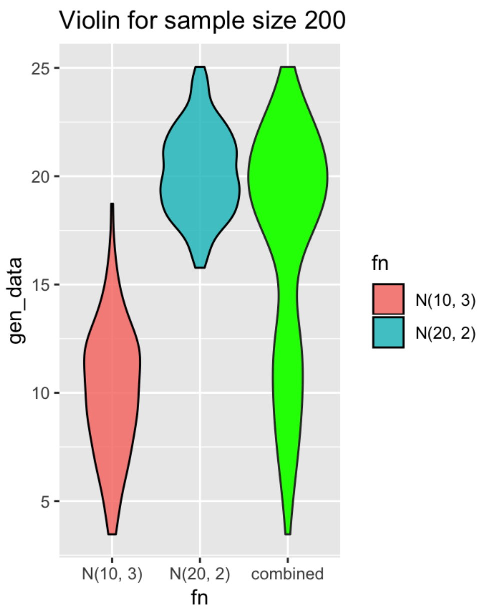












Interactive plot link: https://chepurny.shinyapps.io/prob2/ (https://chepurny.shinyapps.io/prob2/)

3.

Variance of N(0, 1) is equal to 1 because parameters of N distribution is mean(0) and variance(1)

We need to proove that $U^* = 1$, $Var(\rho U + \sqrt{1-\rho^2}V) = 1$ and $Corr(U^*, V^*) = \rho$

$Var(U^*) = Var(U) = 1$ because $U = U^*$

To compute $Var(V^*)$ we will apply the follwing rule - $Var(\alpha X + b) = \alpha^2 + Var(X)$RVs are independent, so $Var(A + B) = Var(A) + Var(B)$

1. $Var(\rho U + \sqrt{1-\rho^2}V) = (\rho^2 * 1) + (1 - \rho^2 * 1) = 1$
2. $Corr(U^*, V^*) = \frac{Cov(U^*,V^*)}{\sqrt{Var(U^*)Var(V^*)}} = \frac{E(U^* - E(U^*)(V^* - E(V^*)))}{1}$

As $Var(X) = E([X-\mu]^2)$ often denoted as σ^2 then

$$\frac{E(U^* - E(U^*)(V^* - E(V^*)))}{1} = E((U^* - 0) * (V^* - 0)) = E(\rho U^2 + \sqrt{1-\rho^2}VU) = E(\rho U^2) + E(\sqrt{1-\rho^2}VU) = \rho E(U^2) + 0 = \rho$$

To double check it - we will compute the correlation

Hide

```
u <- rnorm(100, mean=0, sd=1)
v <- rnorm(100, mean=0, sd=1)
p = 0.3
vs = p * u + sqrt(1 - (p * p)) * v
pearson = cor(u, vs, method='pearson')
sprintf("Pearson: %.3f. P value: %.3f", pearson, p)
```

```
[1] "Pearson: 0.303. P value: 0.300"
```

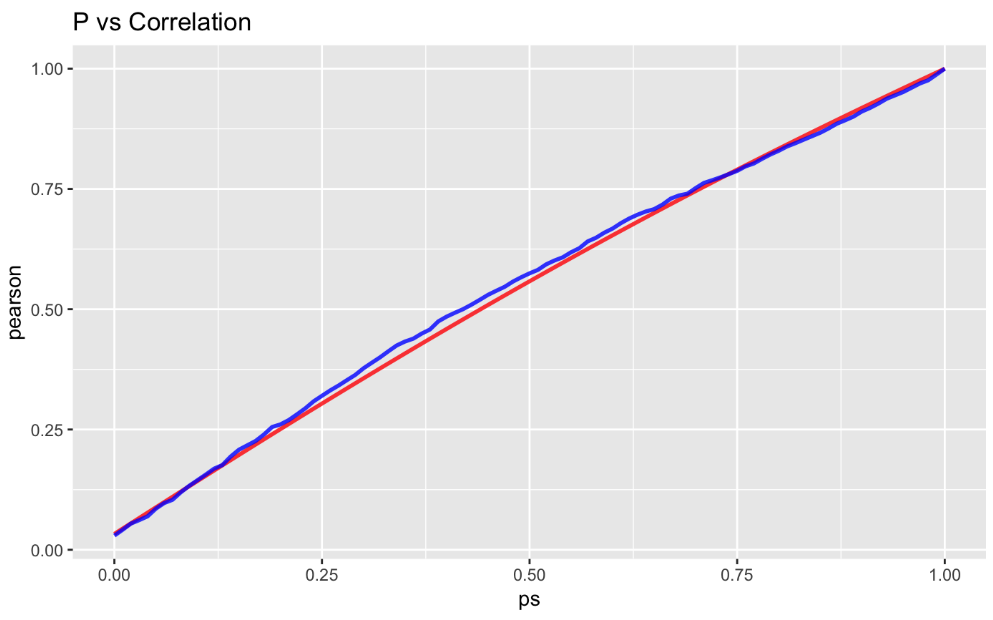
Hide

```
u <- rnorm(100, mean=0, sd=1)
v <- rnorm(100, mean=0, sd=1)
p = 0
vs = p * u + sqrt(1 - (p * p)) * v
pearson = cor(u, vs, method='pearson')
spearman = cor(u, vs, method='spearman')
data <- data.frame(
  ps = 0:100/100
) %>% rowwise() %>%
  mutate(pearson = cor(u, ps * u + sqrt(1 - (ps * ps)) * v, method='pearson'),
         spearman = cor(u, ps * u + sqrt(1 - (ps * ps)) * v, method='spearman'))
data_lt <- data.frame(
  ps = 0:100/100
) %>% rowwise() %>%
  mutate(pearson = cor(u, exp(ps * u + sqrt(1 - (ps * ps)) * v), method='pearson'),
         spearman = cor(u, exp(ps * u + sqrt(1 - (ps * ps)) * v), method='spearman'))
```

As we can see from the plots - before transformation dependency is almost linear for pearson. Spearman is somewhat linear with some noise. After non-linear trasformation applied to the data we can see that pearson correlation hasn't changed (because we have applied monotone but non-linear transformation). Pearson line changed a lot because this coefficient tracks linear dependencies (while spearman tracks monotone dependencies)

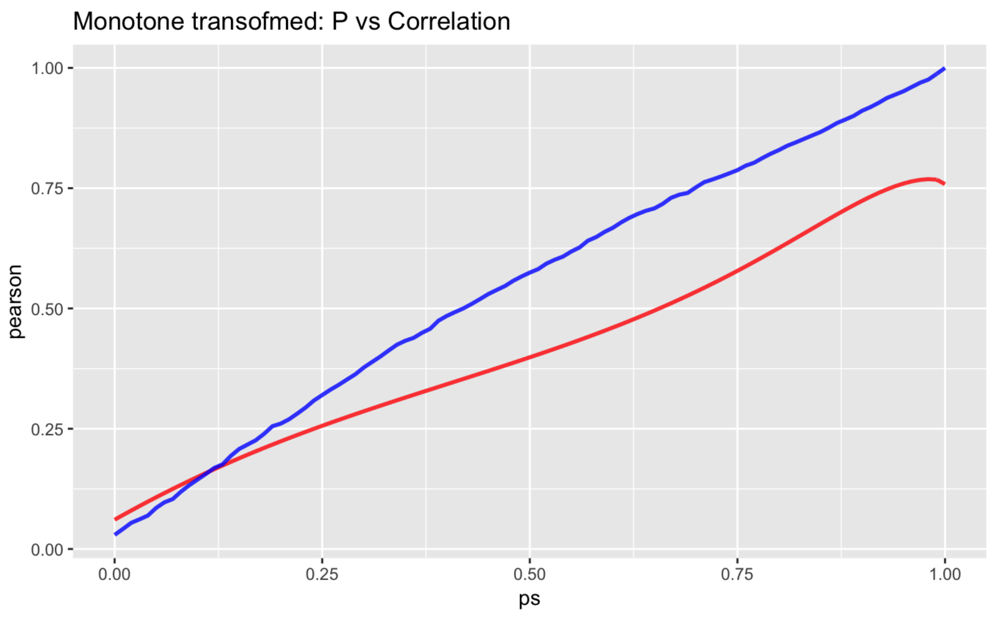
Hide

```
ggplot(data) +
  ggtitle("P vs Correlation") +
  geom_line(aes(x=ps, y=pearson), color='red', size=1, alpha=0.8) +
  geom_line(aes(x=ps, y=spearman), color='blue', size=1, alpha=0.8)
```



Hide

```
ggplot(data_lt) +
  ggtitle("Monotone transofmed: P vs Correlation") +
  geom_line(aes(x=ps, y=pearson), color='red', size=1, alpha=0.8) +
  geom_line(aes(x=ps, y=spearman), color='blue', size=1, alpha=0.8)
```


Monotone transofmed: P vs Correlation
pearson
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
ps