به نام خدا

دانشگاه صنعتی شریف - دانشکده مهندسی کامپیوتر

آمار و احتمال مهندسی

پاییز ۱۴۰۱

تمرین عملی سری سوم

نام و نام خانوادگی: امیررضا آذری
شماره دانشجویی: 99101087

طراحان: آیناز رفیعی، علیرضا حبیب‌زاده

موعد تحویل: ۲۴ آذر

همفکری در تمامی تمرین‌های درس توصیه می‌شود. در عین حال از شما انتظار می‌رود تمام پیاده‌سازی را به تنهایی و بدون مشاهده کد دیگران انجام دهید.

لطفا در فایل ارسالی تمام بلوک‌های کد اجرا شده و شامل نمودارها و خروجی‌های لازم باشند.

سوال اول

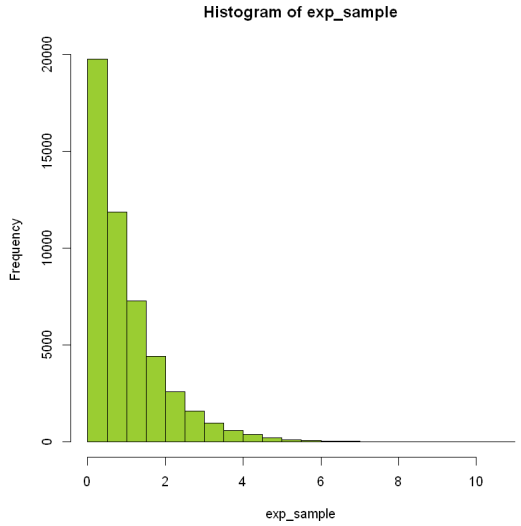
تابعی از یک نمونه از یک متغیر تصادفی uniform بین 0 و 1 به نمونه ای از متغیر تصادفی نمایی بیابید.

پاسخ: U ~ uniform(0, 1), funct = - ln(1-U)
Note that since RU=(0,1), RX=(0,∞). We will find the CDF of X. For x∈(0,∞)
,FX(x) =P(X≤x) =P(−ln(1−U)≤x) =P(1/(1−U)≤e^x) =P(U≤1−e^−x)=1−e^−x

.which is the CDF of an Exponential(1) random variable
##

50000 نمونه از متغیر تصادفی uniform بین 0 و 1 تولید کرده و تابع بالا را روی آن اعمال کنید. پارامتر توزیع نمایی را دلخواه در نظر بگیرید ولی آن را ذکر کنید. نتیجه را بر روی نمودار نمایش دهید.

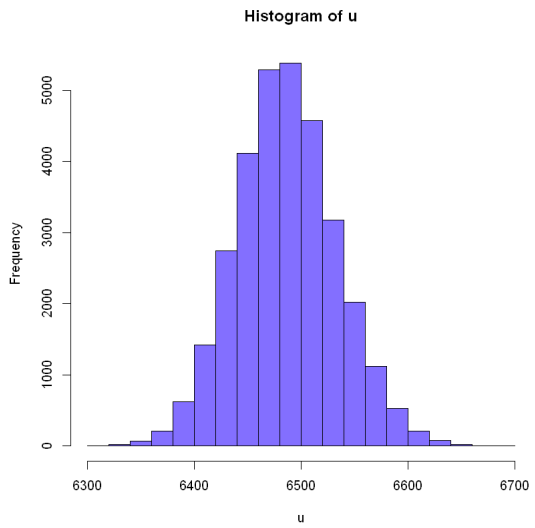
In [2]:
```
funct = function(x) -log(1 - x)
sample <- runif(50000, 0, 1)
exp_sample <- funct(sample)
hist(exp_sample, col="yellowgreen")
```



10000 نمونه از متغیر تصادفی binomial با استفاده از متغیرهای تصادفی از توزیع نمایی که در مرحله‌ی قبل به دست آورده‌اید تولید کرده و بر روی نمودار نمایش دهید.
پارامترهای توزیع را دلخواه انتخاب کنید ولی آن‌ها را ذکر کنید.

In [4]:
```
u <- qbinom(exp_sample, 10000, 0.65)
hist(u, col="slateblue1")
```

```
Warning message in qbinom(exp_sample, 10000, 0.65):
"NaNs produced"
```



سوال دوم

در اتاقی یک بلندگو (فرستنده) قرار دارد که سیگنالی ارسال می‌کند و همچنین سه میکروفون (گیرنده) در فواصل و جاهای مختلف قرار داده شده‌اند که این سیگنال را دریافت می‌کنند. اطلاعات سیگنال ارسال شده و دریافتی بر حسب زمان در دیتاست data.csv به شما داده شده است.
داده‌ها را در یک دیتافریم لود کنید.

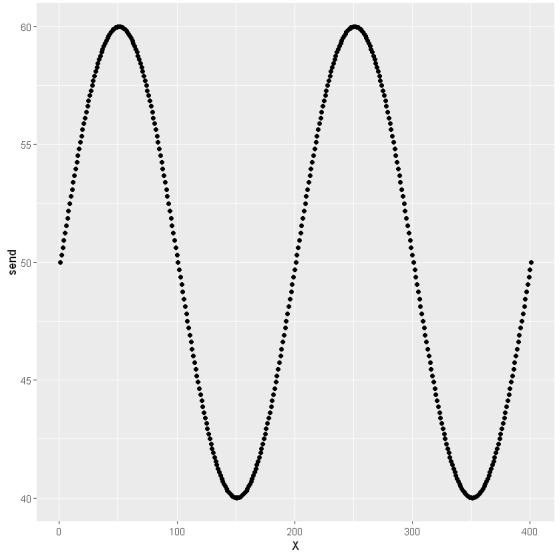
In [1]:
```
data <- read.csv("data.csv")
head(data)
```

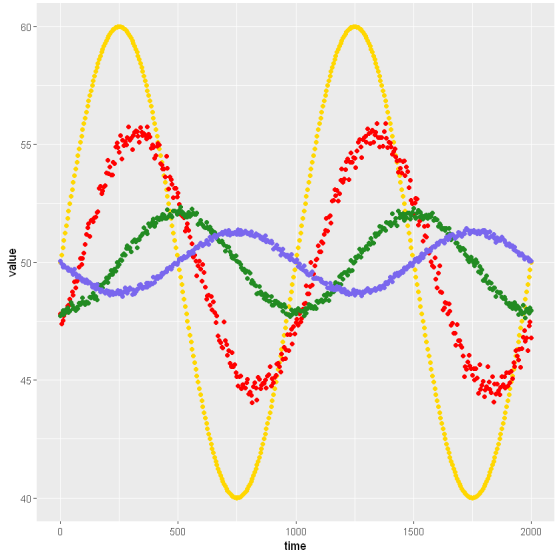
A data.frame: 6 × 6

	X	time	send	recieve1	receive2	receive3
	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
1	1	0	50.00000	47.82258	47.71930	50.03958
2	2	5	50.31411	47.38460	47.82535	49.92389
3	3	10	50.62791	47.53287	47.95819	49.78453
4	4	15	50.94108	47.74043	47.77721	49.87762
5	5	20	51.25333	47.82595	47.88115	49.80018
6	6	25	51.56434	47.89652	48.00486	49.77694

نمودار اطلاعات فرستنده و سه گیرنده بر حسب زمان را در یک نمودار بکشید.

In [3]:
```
library("ggplot2")

ggplot(data, aes(x=X, y=send)) +
  geom_point()

ggplot(data) +
  geom_point(aes(x=time, y=send), col="gold") +
  geom_point(aes(x=time, y=recieve1), col = "red") +
  geom_point(aes(x=time, y=receive2), col = "forestgreen") +
  geom_point(aes(x=time, y=receive3), col = "slateblue2") +
  ylab("value")
```





داده‌های اندازه‌گیری شده به دلیل تفاوت و فاصله‌ی گیرنده‌ها و عوامل محیطی ممکن است دچار تغییراتی شده باشند. به همین منظور لازم است تا داده‌ها را نرمالایز کنیم. یعنی تبدیل خطی‌ای روی آن‌ها اعمال کنیم که توزیع نهایی دارای امید ریاضی صفر و انحراف معیار واحد باشد.

$$x' = \frac{x - \bar{x}}{\sigma_x}$$

داده‌های نرمالایز شده‌ی فرستنده و سه گیرنده را به دیتافریم اضافه کنید. از این به بعد تنها با داده‌های نرمالایز شده کار خواهیم کرد.

In [4]:
```
data$norm_send <- (data$send - mean(data$send)) / sd(data$send)
data$norm_r1 <- (data$recieve1 - mean(data$recieve1)) / sd(data$recieve1)
data$norm_r2 <- (data$receive2 - mean(data$receive2)) / sd(data$receive2)
data$norm_r3 <- (data$receive3 - mean(data$receive3)) / sd(data$receive3)
head(data)
tail(data)
```

A data.frame: 6 × 10

	X	time	send	recieve1	receive2	receive3	norm_send	norm_r1	norm_r2	norm_r3
	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	1	0	50.00000	47.82258	47.71930	50.03958	0.00000000	-0.5661115	-1.523544	0.04769859
2	2	5	50.31411	47.38460	47.82535	49.92389	0.04442152	-0.6806540	-1.452570	-0.07810391
3	3	10	50.62791	47.53287	47.95819	49.78453	0.08879920	-0.6418786	-1.363669	-0.22963938
4	4	15	50.94108	47.74043	47.77721	49.87762	0.13308925	-0.5875953	-1.484789	-0.12841165
5	5	20	51.25333	47.82595	47.88115	49.80018	0.17724796	-0.5652290	-1.415226	-0.21262445
6	6	25	51.56434	47.89652	48.00486	49.77694	0.22123174	-0.5467747	-1.332436	-0.23789314

A data.frame: 6 × 10

	X	time	send	recieve1	receive2	receive3	norm_send	norm_r1	norm_r2	norm_r3
	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
396	396	1975	48.43566	46.65165	47.63494	50.19865	-0.22123174	-0.8723422	-1.579998	0.22066765
397	397	1980	48.74667	46.77738	48.00830	50.20030	-0.17724796	-0.8394605	-1.330134	0.22245995
398	398	1985	49.05892	46.25627	47.95978	50.19051	-0.13308925	-0.9757432	-1.362604	0.21181770
399	399	1990	49.37209	47.31246	47.83297	50.03916	-0.08879920	-0.6995211	-1.447470	0.04723622
400	400	1995	49.68589	47.46225	48.06766	50.10046	-0.04442152	-0.6603474	-1.290407	0.11389870
401	401	2000	50.00000	46.78344	47.95065	50.04847	0.00000000	-0.8378760	-1.368714	0.05736733

برای اطمینان میانگین و واریانس داده‌های نرمالایز شده را حساب کنید.
آیا نتیجه با آنچه انتظار دارید یکسان است؟

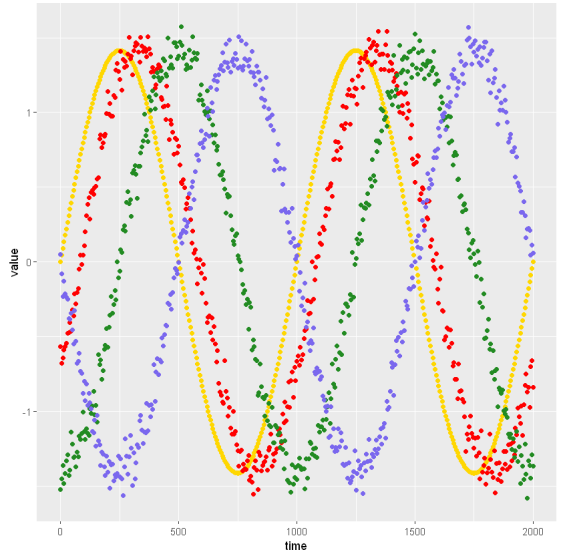
In [5]:
```
print(mean(data$norm_send))
print(var(data$norm_send))
print(mean(data$norm_r1))
print(var(data$norm_r1))
print(mean(data$norm_r2))
print(var(data$norm_r2))
print(mean(data$norm_r3))
print(var(data$norm_r3))
```

```
[1] -1.01522e-19
[1] 1
[1] -7.879346e-16
[1] 1
[1] 6.643623e-16
[1] 1
[1] 2.542777e-15
[1] 1
```

توضیح
:
Double precision floats have about 15 decimal digits of precision, so this is the expected behavior
that the mean is not zero.

داده‌های نرمالایز شده‌ی گیرنده‌ها و فرستنده بر حسب زمان را در یک نمودار نشان دهید.

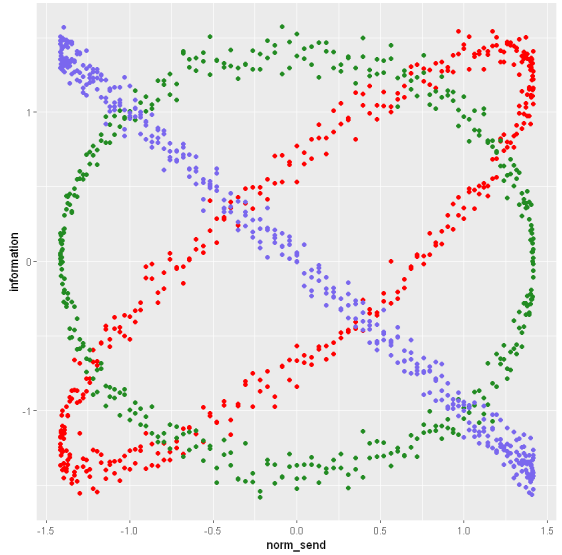
In [6]:
```
ggplot(data) +
  geom_point(aes(x=time, y=norm_send), col = "gold") +
  geom_point(aes(x=time, y=norm_r1), col = "red") +
  geom_point(aes(x=time, y=norm_r2), col = "forestgreen")  +
  geom_point(aes(x=time, y=norm_r3), col = "slateblue2") +
  ylab("value")
```

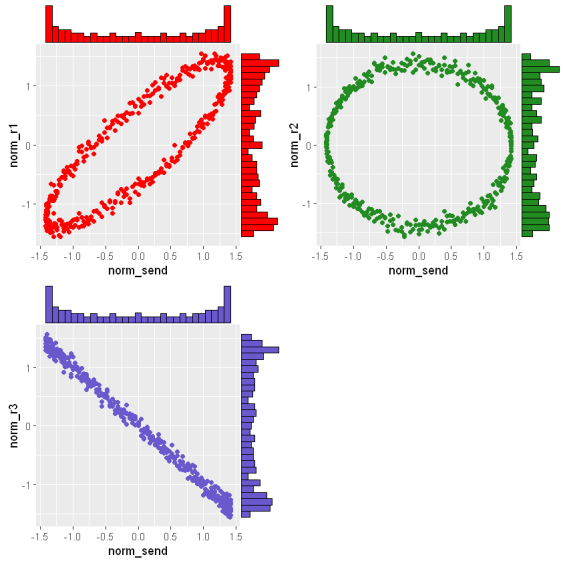


توزیع توام داده‌های نرمالایز شده‌ی سه گیرنده با فرستنده را در نمودار یا نمودارهای مناسبی نمایش دهید.
(راهنمایی: جواب‌ها همگی نوعی بیضی هستند)

توزیع حاشیه‌ای هر کدام از متغیرهای نرمالایز شده‌ی send, receive1, receive2, receive3 را به صورت جداگانه یا روی همان توزیع توام نمایش دهید.
برای این کار می‌توانید از ggMarginal از کتابخانه‌ی ggExtra استفاده کنید. (یا نکنید)

In [8]:
```
library(ggExtra)
library(ggpubr)

p1 <- ggplot(data) +
  geom_point(aes(x=norm_send, y=norm_r1), col = "red") +
  geom_point(aes(x=norm_send, y=norm_r2), col = "forestgreen")  +
  geom_point(aes(x=norm_send, y=norm_r3), col = "slateblue2") +
  ylab("information")

p2 <- ggplot(data) +
  geom_point(aes(x=norm_send, y=norm_r1), col = "red")
p3 <- ggMarginal(p2, type="histogram", fill="red")

p4 <- ggplot(data) +
  geom_point(aes(x=norm_send, y=norm_r2), col = "forestgreen")
p5 <- ggMarginal(p4, type="histogram", fill="forestgreen")

p6 <- ggplot(data) +
  geom_point(aes(x=norm_send, y=norm_r3), col = "slateblue3")
p7 <- ggMarginal(p6, type="histogram", fill="slateblue3")
p1
ggarrange(p3,p5,p7)
```





کوواریانس اطلاعات نرمالایز شده‌ی گیرنده‌ها با فرستنده را حساب کنید. (سه مقدار)
مقادیر به دست‌آمده را با توزیع‌های توام که در قسمت قبل کشیدید مقایسه کنید.
(به کوواریانس بین توزیع‌های نرمالایز شده کوریلیشن نیز می‌گویند که مقداری بین منفی یک و یک است.)

In [9]:
```
print(cov(data$norm_r1, data$norm_send))
print(cor(data$norm_r1, data$norm_send))
print(cov(data$norm_r2, data$norm_send))
print(cor(data$norm_r2, data$norm_send))
print(cov(data$norm_r3, data$norm_send))
print(cor(data$norm_r3, data$norm_send))
# همانطور که مشاهده می شود، طبق صورت سوال این مقادیر با هم برابرند

# norm_r1:
# همانطور که در نمودار های کشیده شده در قسمت قبل مشاهده می شود،
# حالت بیضی برای این مقدار، نشان دهنده کوریلیشن مثبت می باشد که با محاسبات هم ثابت شده است

# norm_r2:
# همانطور که در نمودار های کشیده شده در قسمت قبل مشاهده می شود،
# حالت بیضی برای این مقدار، نشان دهنده کوریلیشن تقریبا 0 می باشد که با محاسبات هم ثابت شده است

# norm_r3:
# همانطور که در نمودار های کشیده شده در قسمت قبل مشاهده می شود،
# حالت بیضی برای این مقدار، نشان دهنده کوریلیشن منفی می باشد که با محاسبات هم ثابت شده است
```

```
[1] 0.8628303
[1] 0.8628303
[1] -0.00724553
[1] -0.00724553
[1] -0.9967574
[1] -0.9967574
```

امتیازی

همان‌طور که احتمالا حدس زدید گیرنده‌ها علاوه بر آن که اطلاعات را با مقداری نویز و تضعیف شده دریافت می‌کنند آن را با تاخیر هم می‌بینند. یعنی نمودار داده‌های گیرنده‌ها در مقایسه با داده‌های فرستنده شیفت خورده است. به دلیل نویزی که در اطلاعات وجود دارد هیچوقت نقاط با شیفت دادن دقیقا روی هم قرار

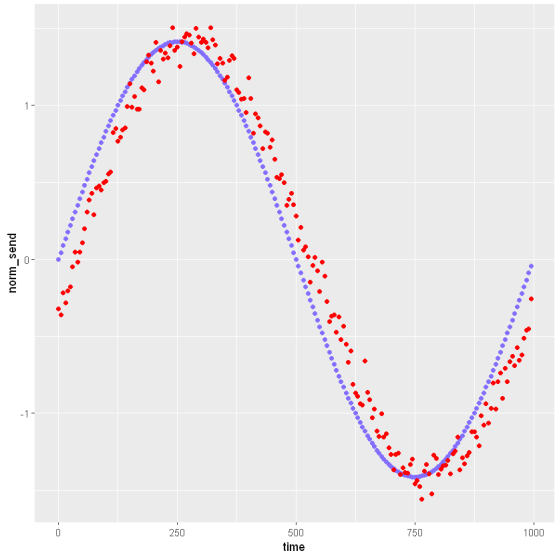
نمی‌گیرند. یک روش خوب برای پیدا کردن تاخیر بین این دو موج شیفت دادن یک موج و محاسبه‌ی کوواریانس بین این دو است تا جایی که این کوواریانس بیشینه شود.
(اگر نویزی وجود نداشت این مقدار برای داده‌های نرمالایز شده‌ی یک می‌بود چرا که کوواریانس یک متغیر نرمالایز شده با خودش یک است)

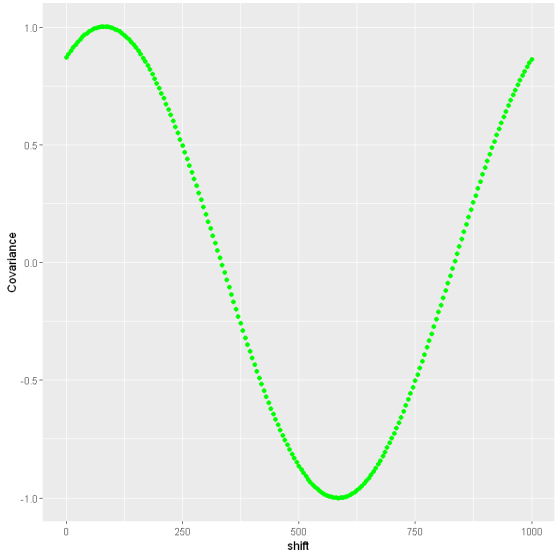
داده‌های گیرنده‌ی اول را به ازای جابه‌جایی‌های مضرب 50 از صفر تا 1000 میکروثانیه (نصف کل بازه‌ی زمانی) به سمت چپ شیفت دهید. سپس داده‌های 1000 میکروثانیه ابتدایی از فرستنده و گیرنده باقی می‌ماند را جدا کنید و کوواریانس این دو را محاسبه کنید. حال نمودار کوواریانس بر حسب شیفت را رسم کنید.

(راهنمایی: همیشه باید 200 داده از میان ستون receive1 نرمالایز شده بردارید و کوواریانس آن را با 200 داده‌ی ابتدایی send نرمالایز شده حساب کنید.)

In [10]:
```
# for shift = 50
reciever1_data <- data$norm_r1
reciever1_data <- reciever1_data[11:210]
data2 <- data[1:200, ]
data2$norm_r1 <- reciever1_data
ggplot(data2) +
  geom_point(aes(x=time, y=norm_send), col = "slateblue1") +
  geom_point(aes(x=time, y=norm_r1), col = "red")
cov(data2$norm_send, data2$norm_r1)
# shift = 0, 5, 10, ..., 10000
funct <- function(x){
  reciever1_data <- data$norm_r1
  reciever1_data <- reciever1_data[(x/5 + 1): (x/5 + 200)]
  data2 <- data[1:200, ]
  (cov(data2$norm_send, reciever1_data))
}

x <- seq(0,1000, by = 5)
df <- data.frame(x,sapply(x, funct))

ggplot(df) +
  geom_point(aes(x=x, y=sapply.x..funct.), col = "green") +
  ylab("Covariance") +
  xlab("shift")
```

0.981454959787633





شیفتی که در آن بیشترین کوواریانس بین دادهای گیرنده‌ی شیفت خورده و فرستنده ایجاد می‌شود و کوواریانس بیشینه را پیدا کنید.

In [11]:
```
cat("Shift = ")
cat(df[which.max(df$sapply.x..funct.), 1])
cat("\nCovariance = ")
cat(max(df$sapply.x..funct.))
```

```
Shift = 80
Covariance = 1.002021
```

توزیع توام 1000 میکروثانیه اول گیرنده‌ی شیفت خورده با فرستنده را رسم کنید.

In [12]:
```
reciever1_data <- data$norm_r1
reciever1_data <- reciever1_data[17: 216]
data2 <- data[1:200, ]
data2$norm_r1 <- reciever1_data
p1 <- ggplot(data2) +
  geom_point(aes(x=norm_send, y=norm_r1), col = "forestgreen")
p1
```