

Digital signal processing workbook

Andrey Akinshin

Contents

1	Introduction	2
2	Discrete-time (DT) signals	4
2.1	Discrete-time signals	5
2.1.1	The delta signal	5
2.1.2	The unit step	5
2.1.3	The exponential decay	6
2.1.4	The sinusoid	6
2.1.5	Shift of a finite-length: finite-support	7
2.1.6	Shift of a finite-length: periodic extension	9
2.1.7	Energy and power	10
2.2	The complex exponential	11
2.3	The Karplus-Strong Algorithm	12
2.3.1	Sine wave	12
2.3.2	Proto-violin	12
2.3.3	Random	13
3	Euclid and Hilbert	14
3.1	From Euclid to Hilbert	14
3.1.1	Space of square-integrable functions over $[-1, 1] : l_2([-1, 1])$	14
3.1.2	Interesting question when the space has ∞ dimensions	15
3.2	Hilbert Space, properties and bases	19
3.2.1	Formal properties of a vector space	19
3.2.2	Formal properties of the inner product	19
3.2.3	Inner product for signals	19
3.2.4	Special bases	19
3.3	Hilbert Space and approximation	20
3.3.1	Parseval's Theorem	20
3.3.2	Gram-Schmidt algorithm	20
4	Fourier Analysis	21
4.1	Exploration via a change of basis	21
4.1.1	The Fourier Basis for \mathbb{C}^N	21
4.2	The Discrete Fourier Transform	26
4.2.1	DFT of the delta signal	27
4.2.2	DFT of the uniform signal	27
4.2.3	DFT of the cos signal	28
4.2.4	DFT of length-M step in \mathbb{C}^N	30
4.3	The DFT in practice	30
4.3.1	Average Monthly Temperatures at Nottingham, 1920-1939	30
4.4	DFT, DFS, DTFT	32
4.4.1	Discrete-Time Fourier Transform (DTFT)	32
4.5	DTFT: intuition and properties	34

4.5.1	DTFT properties	34
4.5.2	Some particular cases	34
4.5.3	DTFT as basic expansion	34
4.6	Relationship between transforms	35
4.6.1	DTFT of periodic signals	35
4.6.2	32-tap sawtooth	35
4.6.3	Interval indicator signal	37
4.7	Sinusoidal modulation	38
4.7.1	Classifying signals in frequency	38
4.7.2	Sinusoidal modulation	38
4.7.3	Tuning a guitar	39
4.8	The Short-Time Fourier Transform	41
4.8.1	Example: 1-5-9	41
4.8.2	Short-Time Fourier Transform	42
4.8.3	Spectrogram	45
4.9	FFT: history and algorithm	46
4.9.1	History	46
4.9.2	The DFT matrix	46
4.9.3	Divide and Conquer for DFT — One step	46
4.9.4	Divide and Conquer for DFT — Multiple steps	46
4.9.5	Conclusion	47
4.10	Why the DFT is useful: A few examples	47
4.10.1	Overview	47

Module 1

Introduction

This document is a workbook for the “Digital signal processing” Coursera course (<https://www.coursera.org/course/dsp>). It uses the R language and the knitr package for calculation and plot rendering.

Help code:

```
library("ggplot2")

## Loading required package: methods

library("grid")
library("gridExtra")
library("phonTools")
# Discrete signal plot
dsplot <- function(x, y, xlab="", ylab "") {
  d <- data.frame(x = x, y = y, y0 = rep(0, length(y)))
  p <- ggplot() +
    geom_segment(data=d, mapping=aes(x=x, y=y0, xend=x, yend=y), size=1, color="red") +
    geom_point(data=d, mapping=aes(x=x, y=y), size=2, shape=21, fill="red") +
    scale_x_continuous(name=xlab) +
    scale_y_continuous(name=ylab)
  p
}

# Continuous signal plot
csplot <- function(x, y, xlab="", ylab "") {
  d <- data.frame(x = x, y = y)
  p <- ggplot(data=d, aes(x=x, y=y)) +
    geom_line(color="red") +
    scale_x_continuous(name=xlab) +
    scale_y_continuous(name=ylab)
  p
}

# dft plot
dftplot <- function(x, xlab="", onlyhalf=F) {
  n <- 1:length(x)
  if (onlyhalf)
    n <- 1:(length(x)/2)
  X <- fft(x)
  grid.arrange(dsplot(n, round(Re(X),4), ylab="Re(X[n])", xlab=xlab),
               dsplot(n, round(Im(X),4), ylab="Im(X[n])", xlab=xlab), ncol=2)
  grid.arrange(dsplot(n, round(Mod(X),4), ylab="Mod(X[n])", xlab=xlab),
```

```

        dsplot(n, round(Arg(X),4), ylab="Arg(X[n])", xlab=xlab), ncol=2)
}

# Discrete-Time Fourier Transform
dtft <- function(x) {
  func <- function(w) {
    sum(x * exp(-1i * w * (1:length(x))))
  }
  Vectorize(func)
}

# Discrete-Time Fourier Transform plot
dtftplot <- function(x, xlab="", ylab "") {
  func <- dtft(x)
  t <- seq(-pi, pi, by=0.01)
  value <- func(t)
  grid.arrange(csplot(t, round(Re(value),4), ylab="Re(X[n])", xlab=xlab),
               csplot(t, round(Im(value),4), ylab="Im(X[n])", xlab=xlab), ncol=2)
  grid.arrange(csplot(t, round(Mod(value),4), ylab="Mod(X[n])", xlab=xlab),
               csplot(t, round(Arg(value),4), ylab="Arg(X[n])", xlab=xlab), ncol=2)
}

```

Module 2

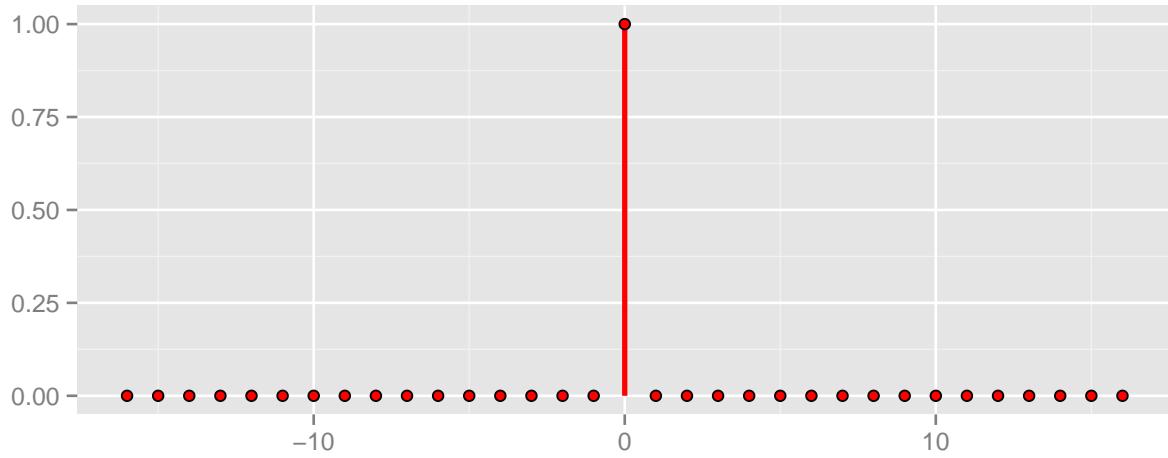
Discrete-time (DT) signals

2.1 Discrete-time signals

2.1.1 The delta signal

$$x[n] = \delta[n]$$

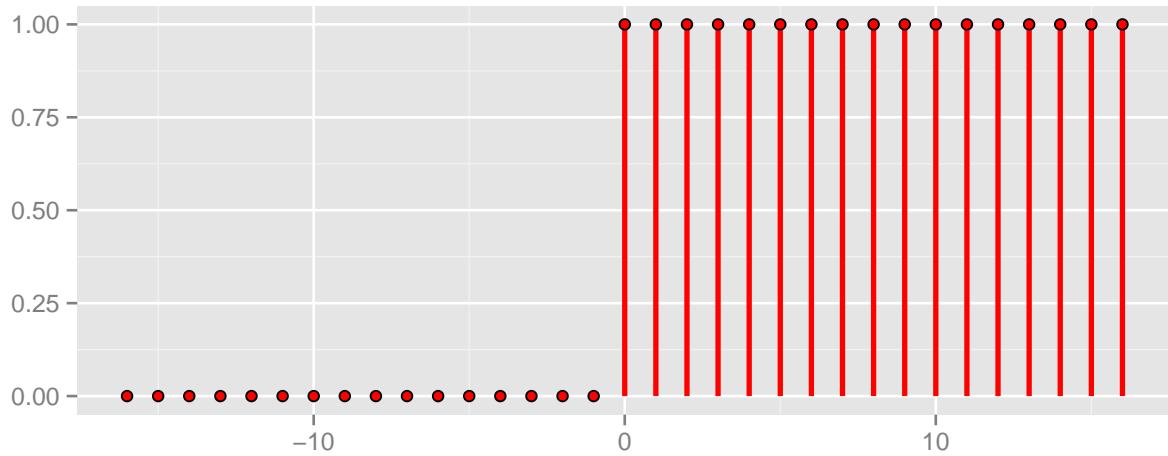
```
n <- -16:16
x <- ifelse(n == 0, 1, 0)
dsplot(n, x)
```



2.1.2 The unit step

$$x[n] = 1$$

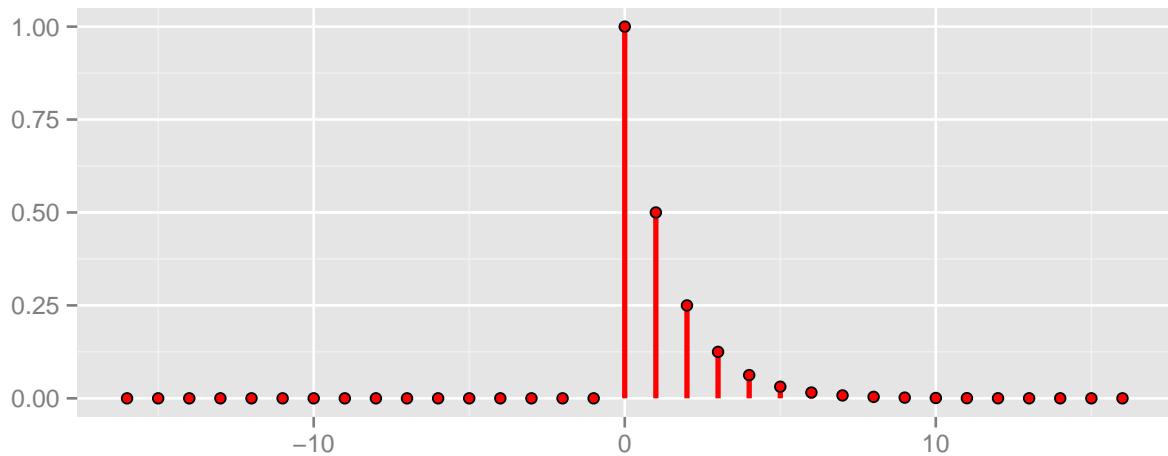
```
n <- -16:16
x <- ifelse(n >= 0, 1, 0)
dsplot(n, x)
```



2.1.3 The exponential decay

$$x[n] = |a|^n u[n], \quad |a| < 1$$

```
a <- 0.5
n <- -16:16
x <- ifelse(n >= 0, abs(a)^n, 0)
dspplot(n, x)
```



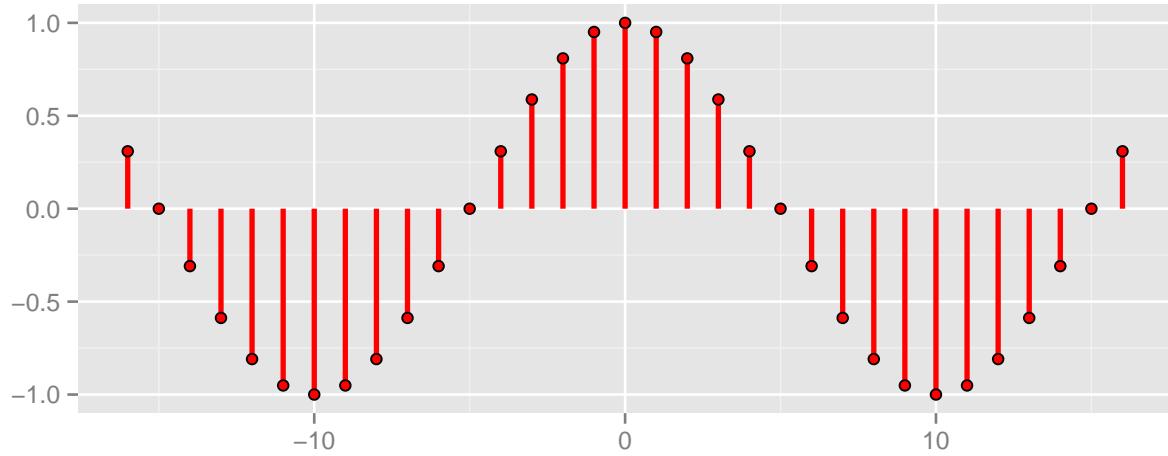
2.1.4 The sinusoid

$$x[n] = \sin(\omega_0 n + \theta)$$

```

omega0 <- pi/10; theta <- pi/2
n <- -16:16
x <- sin(omega0 * n + theta)
dsplot(n, x)

```



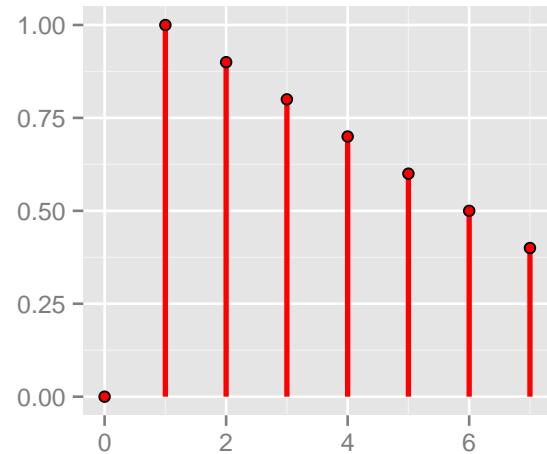
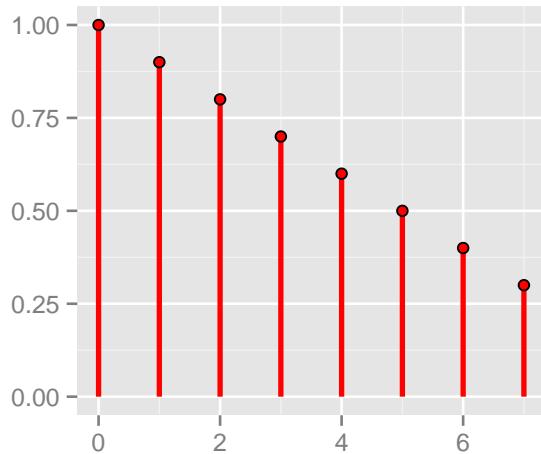
2.1.5 Shift of a finite-length: finite-support

$$\bar{x}[n] = x[n - z]$$

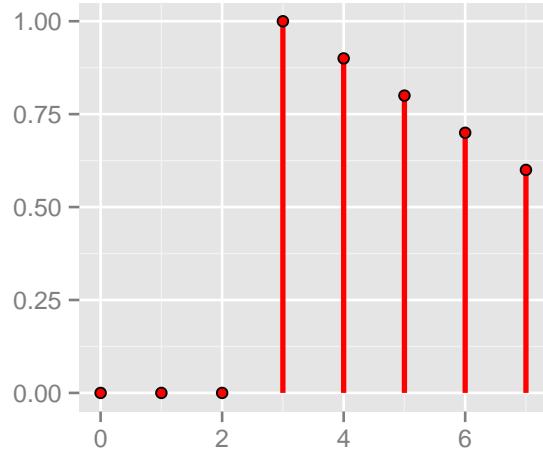
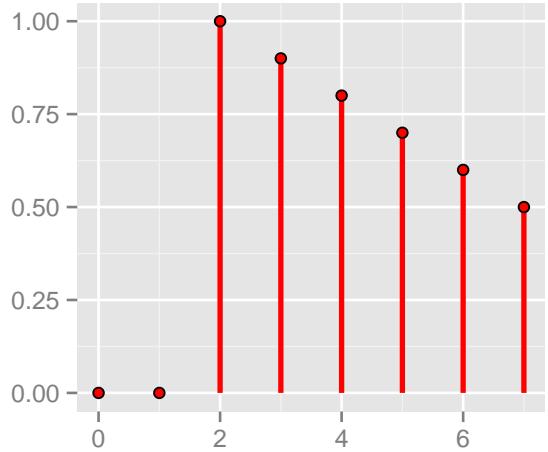
```

shift <- function(x, z) c(rep(0, z), x[1:(length(x)-z)])
n <- 0:7
x1 <- 1-n*0.1
x2 <- shift(x1, 1)
x3 <- shift(x1, 2)
x4 <- shift(x1, 3)
grid.arrange(dsplot(n, x1), dsplot(n, x2), ncol=2)

```



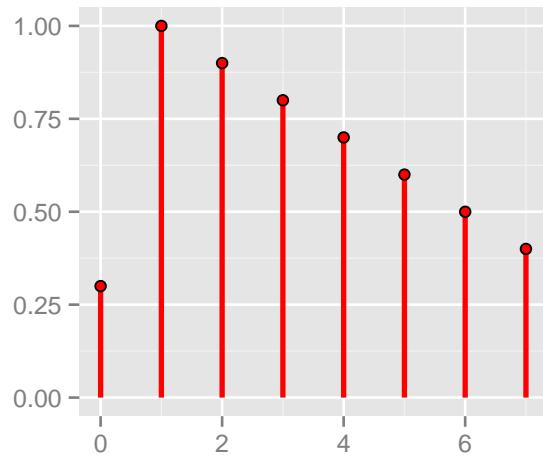
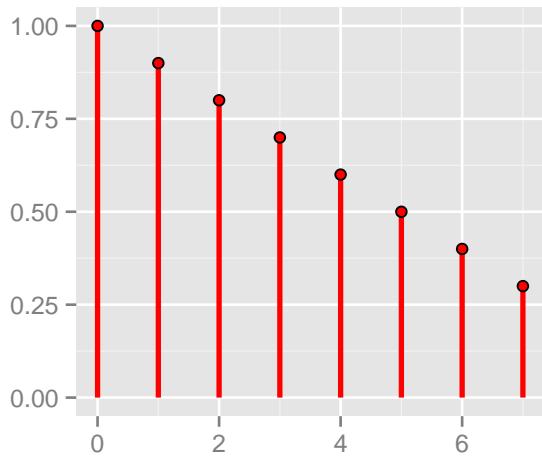
```
grid.arrange(dsplot(n, x3), dsplot(n, x4), ncol=2)
```



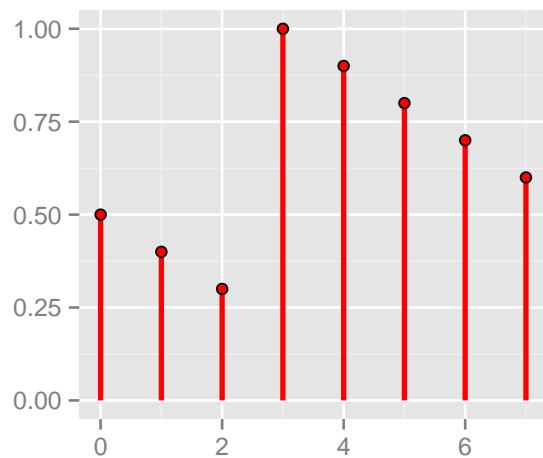
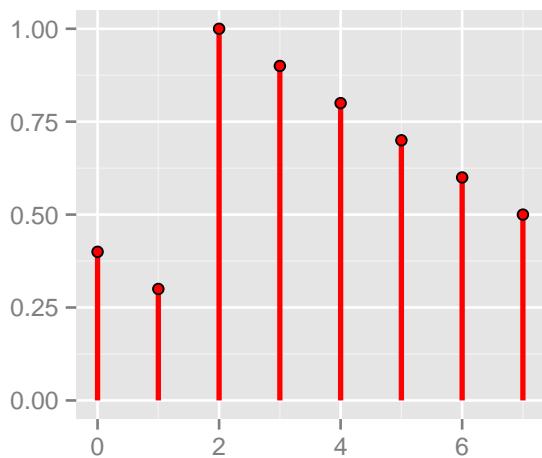
2.1.6 Shift of a finite-length: periodic extension

$$\tilde{x}[n] = x[n - z]$$

```
shift <- function(x, z) c(x[(length(x)-z+1):length(x)], x[1:(length(x)-z)])
n <- 0:7
x1 <- 1-n*0.1
x2 <- shift(x1, 1)
x3 <- shift(x1, 2)
x4 <- shift(x1, 3)
grid.arrange(dsplot(n, x1), dsplot(n, x2), ncol=2)
```



```
grid.arrange(dsplot(n, x3), dsplot(n, x4), ncol=2)
```



2.1.7 Energy and power

General formulas:

$$E_x = \sum_{n=-\infty}^{\infty} |x[n]|^2$$

$$P_x = \lim_{N \rightarrow \infty} \frac{1}{2N+1}$$

Periodic signals:

$$E_{\tilde{x}} = \infty$$

$$P_{\tilde{x}} = \sum_{n=0}^{N-1} |\tilde{x}[n]|^2$$

2.2 The complex exponential

$$x[n] = Ae^{j(\omega n + \phi)} = A(\cos(\omega n + \phi) + j \sin(\omega n + \phi))$$

$$\cos(\omega n + \phi) = a \cos(\omega n) + b \sin(\omega n), \quad a = \cos \phi, \quad b = \sin \phi$$

$$\operatorname{Re}\{e^{j(\omega n + \phi)}\} = \operatorname{Re}\{e^{j\omega n} e^{j\phi}\}$$

$$e^{j\alpha} = \cos \alpha + j \sin \alpha$$

$$\text{Rotation: } z' = ze^{j\alpha}$$

$$\text{Signal: } x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$

$$\text{Signal: } x[n] = e^{j\omega n + \phi}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$

$$e^{j\omega n} \text{ periodic} \Leftrightarrow \omega = \frac{M}{N} 2\pi, \quad M, N \in \mathbb{N}$$

$$e^{j\omega} = e^j (\omega + 2k\pi), \forall k \in \mathbb{N}$$

2.3 The Karplus-Strong Algorithm

Main equation:

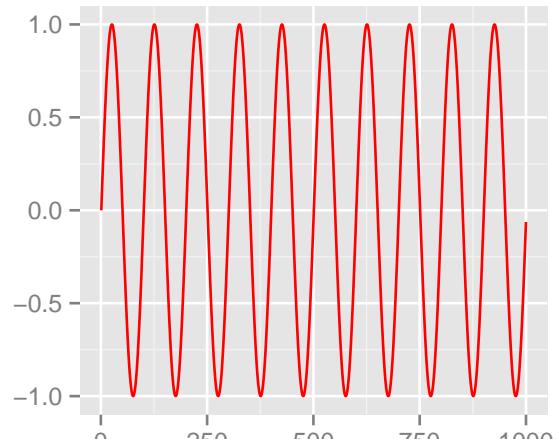
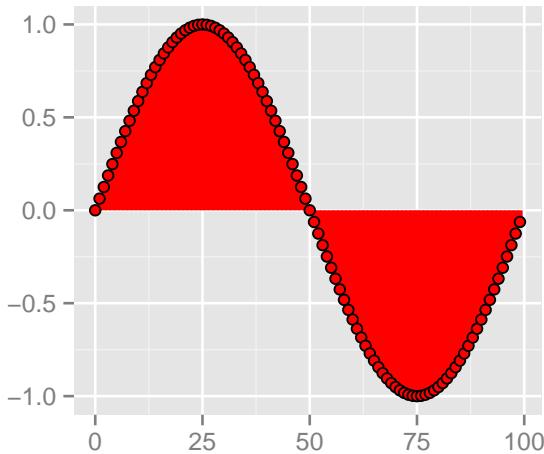
$$y[n] = \alpha y[n - M] + x[n]$$

```
ks <- function(x, M, alpha, N) {
  y <- c()
  for (n in 1:N) {
    yn <- alpha * ifelse(n > M, y[n-M], 0) +
      ifelse(n <= length(x), x[n], 0)
    y <- c(y, yn)
  }
  y
}
```

2.3.1 Sine wave

$M = 100, \alpha = 1, \bar{x}[n] = \sin(2\pi n/100)$ for $0 \leq n < 100$ and zero elsewhere

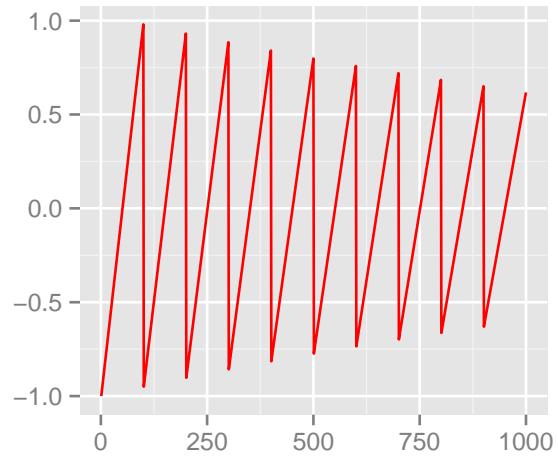
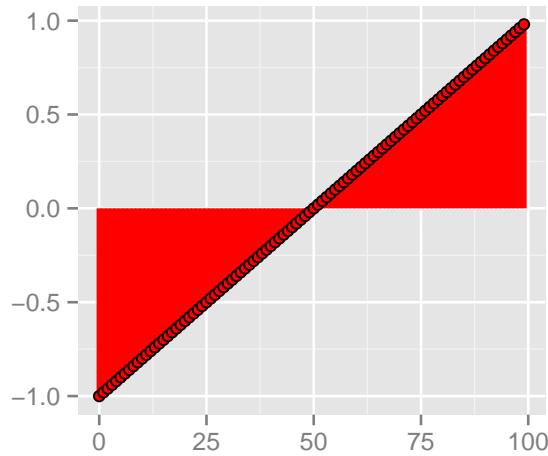
```
M <- 100; alpha <- 1; n <- 0:99
x <- sin(2*pi*n/100)
y <- ks(x, M, alpha, 1000)
grid.arrange(dsplot(n, x), csplot(1:length(y), y), ncol=2)
```



2.3.2 Proto-violin

$M = 100, \alpha = 0.95, \bar{x}[n] : \text{zero-mean sawtooth wave between 0 and 99, zero elsewhere}$

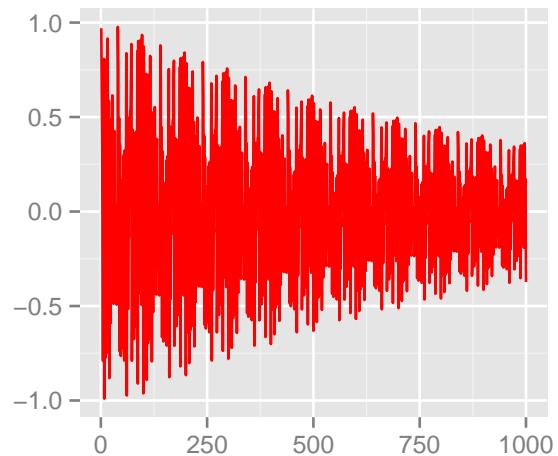
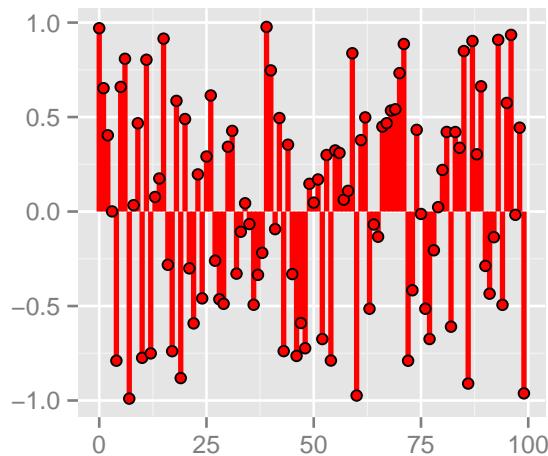
```
M <- 100; alpha <- 0.95; n <- 0:99
x <- -1 + 2*n/100
y <- ks(x, M, alpha, 1000)
grid.arrange(dsplot(n, x), csplot(1:length(y), y), ncol=2)
```



2.3.3 Random

$M = 100, \alpha = 0.9, \bar{x}[n] : 100$ random values between 0 and 99, zero elsewhere

```
M <- 100; alpha <- 0.9; n <- 0:99
x <- runif(100, -1, 1)
y <- ks(x, M, alpha, 1000)
grid.arrange(dsplot(n, x), csplot(1:length(y), y), ncol=2)
```



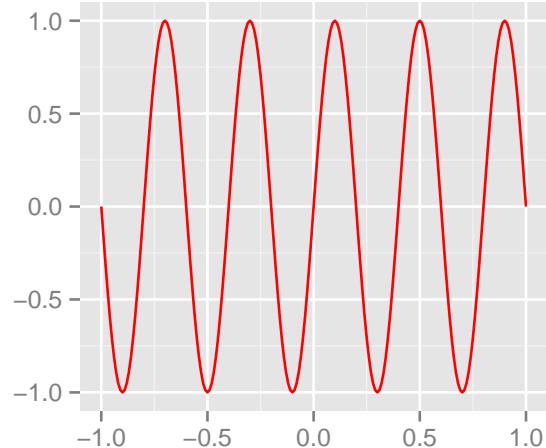
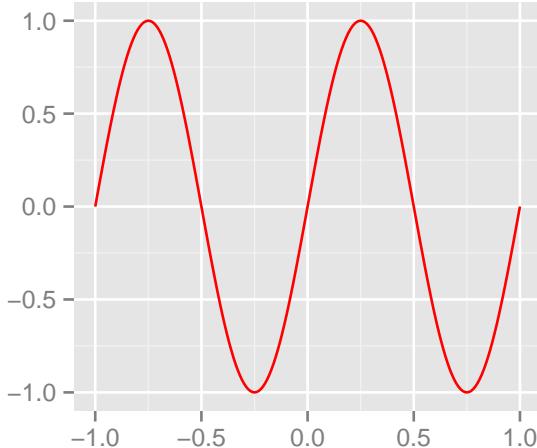
Module 3

Euclid and Hilbert

3.1 From Euclid to Hilbert

3.1.1 Space of square-integrable functions over $[-1, 1] : l_2([-1, 1])$

```
t <- seq(-1, 1, by=0.01)
f1 <- 2*pi; x1 <- sin(f1 * t)
f2 <- 5*pi; x2 <- sin(f2 * t)
grid.arrange(csplot(t, x1), csplot(t, x2), ncol=2)
```



$$x^{(1)} = \sin(f_1 t), \quad f_1 = 2\pi$$

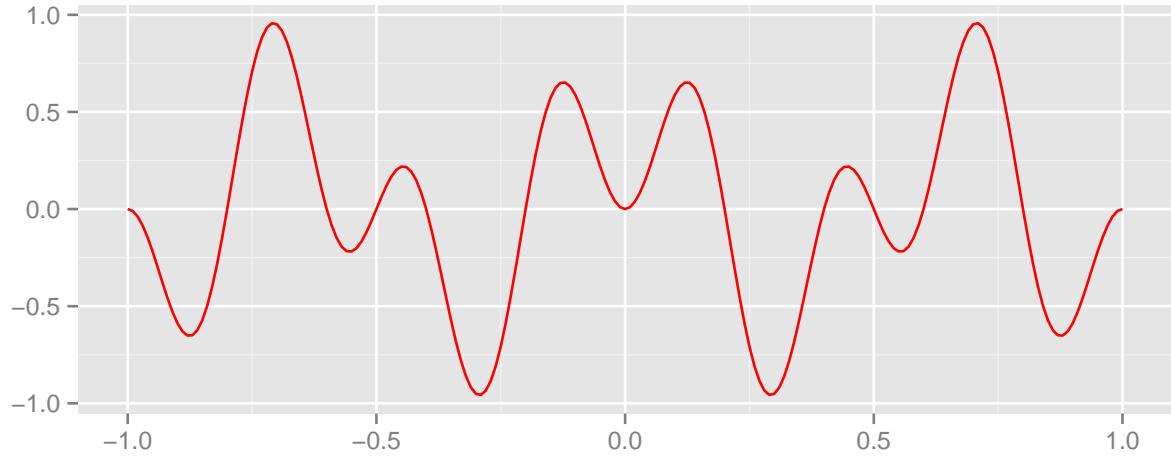
$$x^{(2)} = \sin(f_2 t), \quad f_2 = 5\pi$$

$$\langle x^{(1)}, x^{(2)} \rangle = \int_{-1}^1 \sin(f_1 t) \sin(f_2 t) dt$$

$x^{(1)} \perp x^{(2)}$ if $f_1 \neq f_2$ and f_1, f_2 integer multiples of a fundamental (harmonically related)

$$\sin(f_1 t) \sin(f_2 t), \quad f_1 = 2\pi, f_2 = 5\pi$$

```
csplot(t, x1*x2)
```

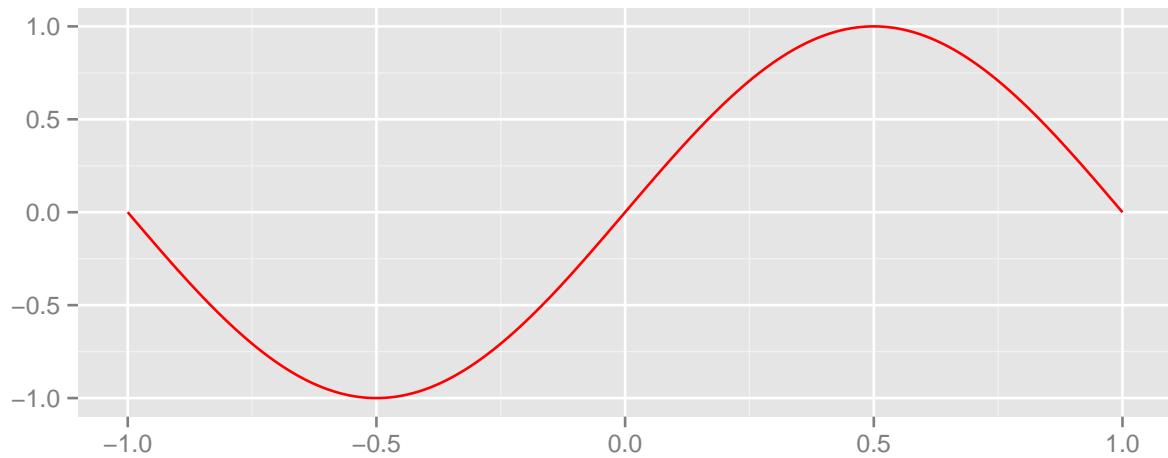


```
round(integrate(function(x) sin(f1*x)*sin(f2*x), lower=-1, upper=1)$value, 10)
## [1] 0
```

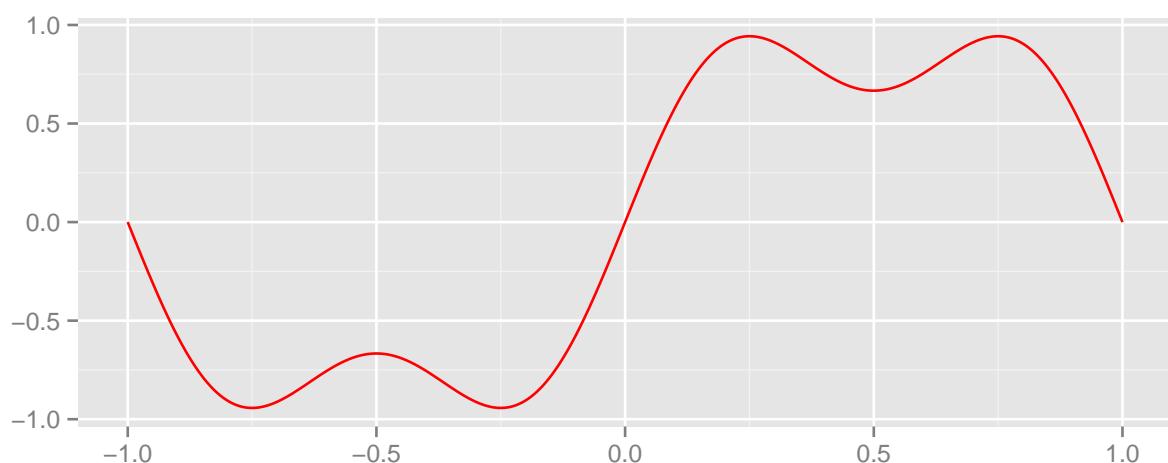
3.1.2 Interesting question when the space has ∞ dimensions

$$\sum_{k=0}^N x^{(2k+1)}, \quad x^{(n)} = \sin(\pi n t)/n, t \in [-1, 1]$$

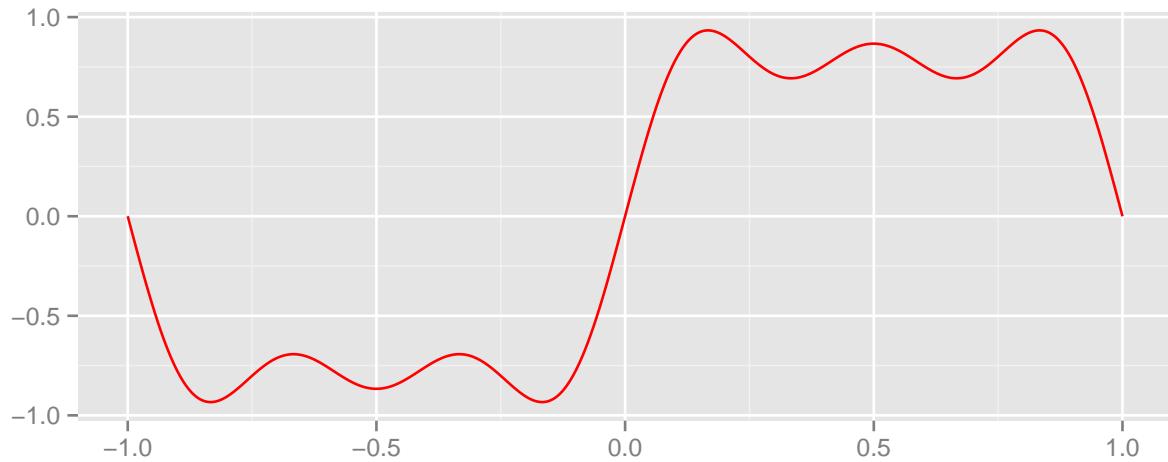
```
inftySpace <- function(N) {
  t <- seq(-1, 1, by=0.001)
  x <- rep(0, length(t))
  for (k in 0:N) {
    n <- 2 * k + 1
    x <- x + sin(pi*n*t)/n
  }
  csplot(t, x)
}
inftySpace(0)
```



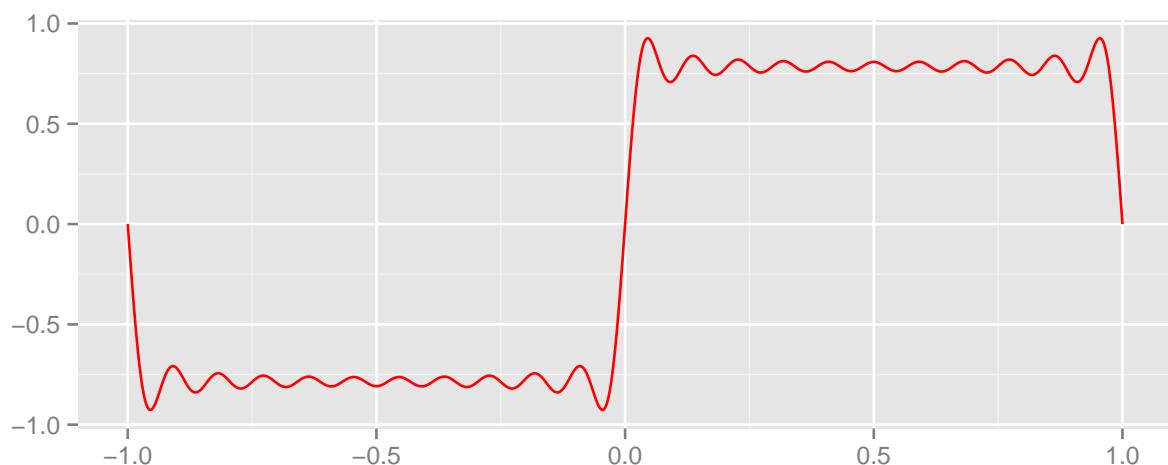
inftySpace(1)



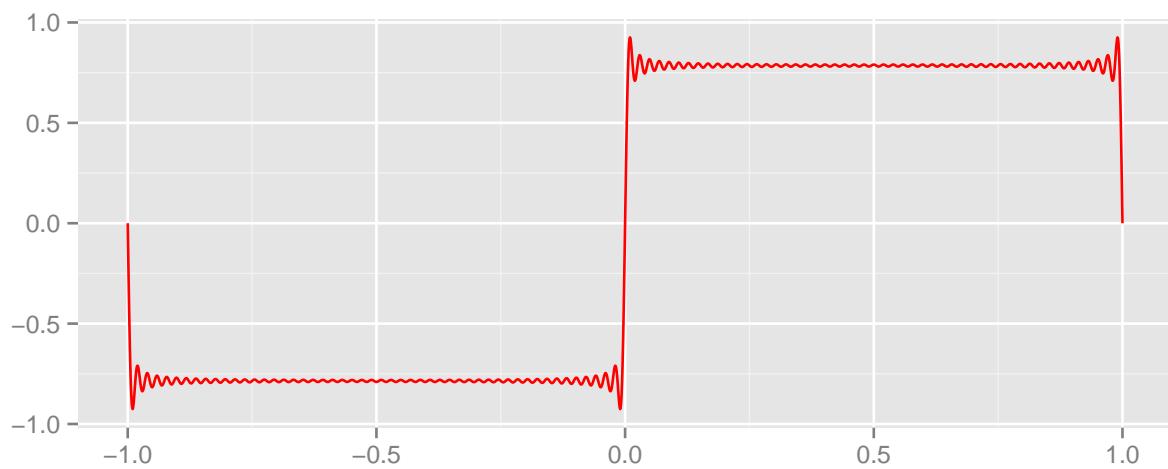
inftySpace(2)



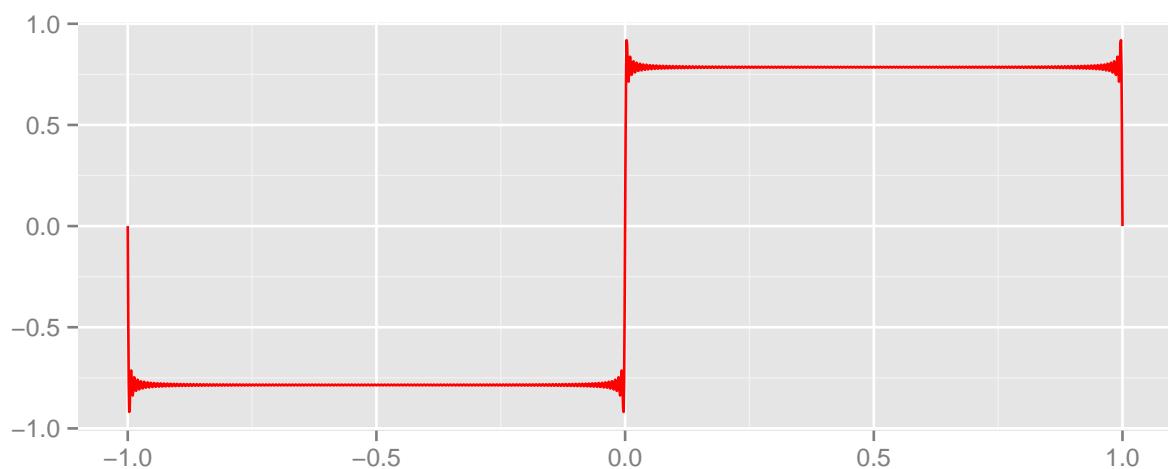
inftySpace(10)



inftySpace(50)



inftySpace(150)



3.2 Hilbert Space, properties and bases

3.2.1 Formal properties of a vector space

For $x, y, z \in V$ and $\alpha, \beta \in \mathbb{C}$:

- $x + y = y + x$
- $(x + y) + z = x + (y + z)$
- $\alpha(x + y) = +\alpha y$
- $(\alpha + \beta)x = \alpha x + \beta x$
- $\alpha(\beta x) = (\alpha\beta)x$
- $\exists 0 \in V \quad | \quad x + 0 = 0 + x = x$
- $\forall x \in V \exists (-x) \quad | \quad x + (-x) = 0$

3.2.2 Formal properties of the inner product

For $x, y, z \in V$ and $\alpha \in \mathbb{C}$:

- $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$
- $\langle x, y \rangle = \langle y, x \rangle^*$
- $\langle \alpha x, y \rangle = \alpha^* \langle x, y \rangle$
- $\langle x, \alpha y \rangle = \alpha \langle x, y \rangle$
- $\langle x, x \rangle \geq 0$
- $\langle x, x \rangle = 0 \Leftrightarrow x = 0$
- if $\langle x, y \rangle = 0$ and $x, y \neq 0$ then x and y are called orthogonal

3.2.3 Inner product for signals

$$\langle x, y \rangle = \sum_{n=0}^{N-1} x^*[n]y[n]$$

$$\langle x, y \rangle = \sum_{n=-\infty}^{\infty} x^*[n]y[n]$$

We require sequences to be *square-summable*: $\sum |x[n]|^2 < \infty$.
Space of square-summable sequences: $l_2(\mathbb{Z})$.

3.2.4 Special bases

Orthogonal basis: $\langle w^{(k)}, w^{(n)} \rangle = 0$ for $k \neq n$.
Orthonormal basis: $\langle w^{(k)}, w^{(n)} \rangle = \delta[n - k]$.

3.3 Hilbert Space and approximation

3.3.1 Parseval's Theorem

$$x = \sum_{k=0}^{K-1} \alpha_k w^{(k)}$$

For an orthonormal basis:

$$\|x\|^2 = \sum_{k=0}^{K-1} |\alpha_k|^2$$

3.3.2 Gram-Schmidt algorithm

The Gram-Schmidt algorithm leads to an orthonormal basis for $P_N([-1, 1])$:

$$\begin{aligned} u^{(0)} &= \sqrt{1/2} \\ u^{(1)} &= \sqrt{3/2}t \\ u^{(2)} &= \sqrt{5/8}(3t^2 - 1) \\ u^{(3)} &= \dots \end{aligned}$$

$$\alpha_k = \langle u^{(k)}, x \rangle$$

Module 4

Fourier Analysis

4.1 Exploration via a change of basis

4.1.1 The Fourier Basis for \mathbb{C}^N

In vector notation:

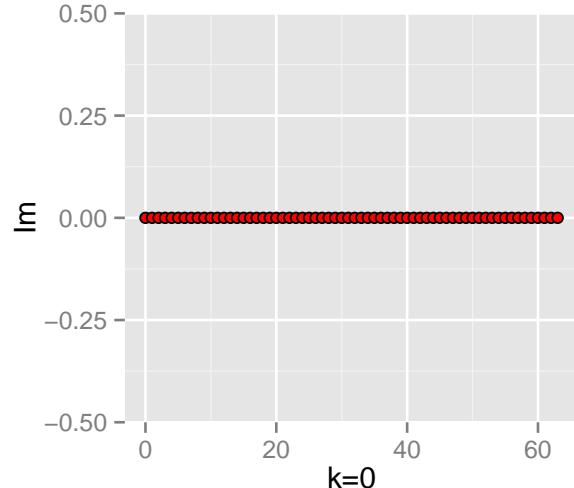
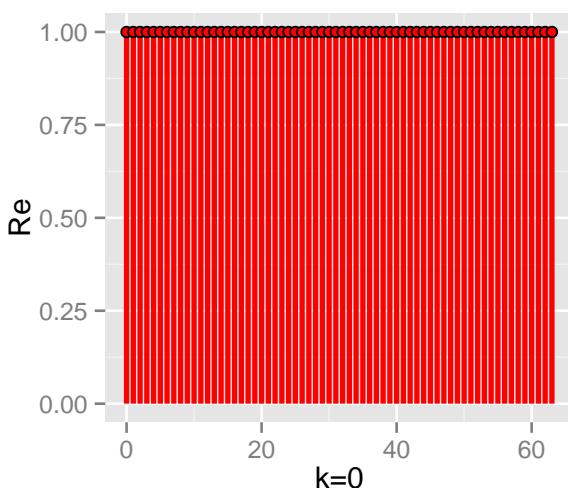
$$\{w_{k=0,1,\dots,N-1}^{(k)}\}$$

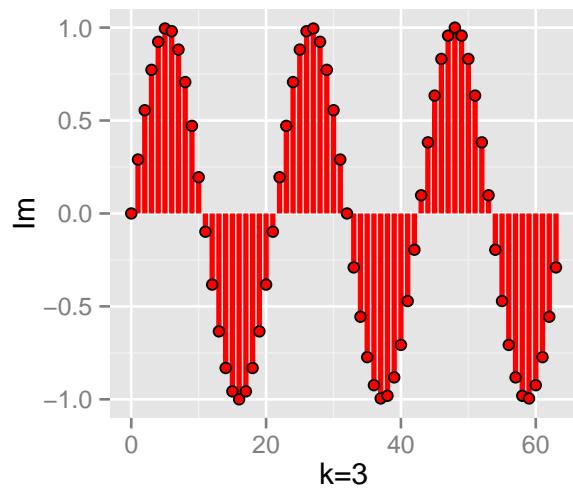
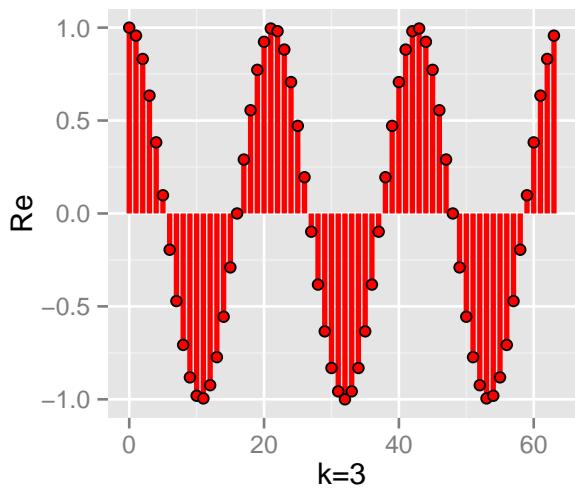
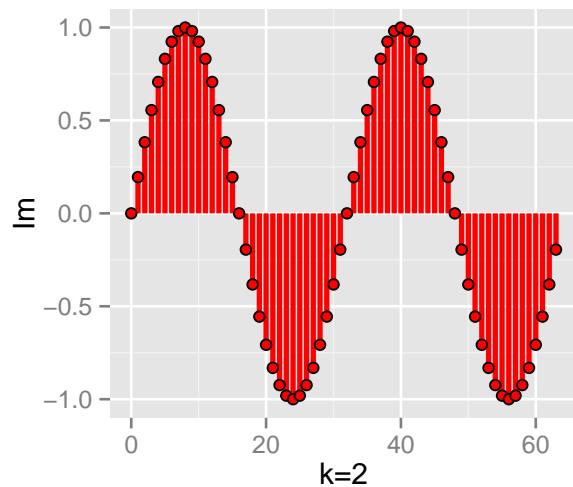
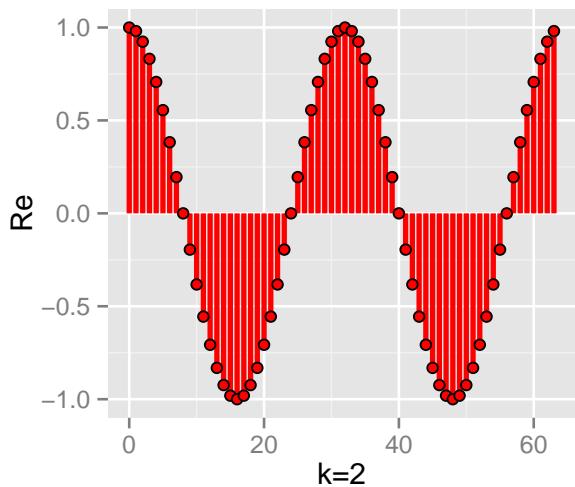
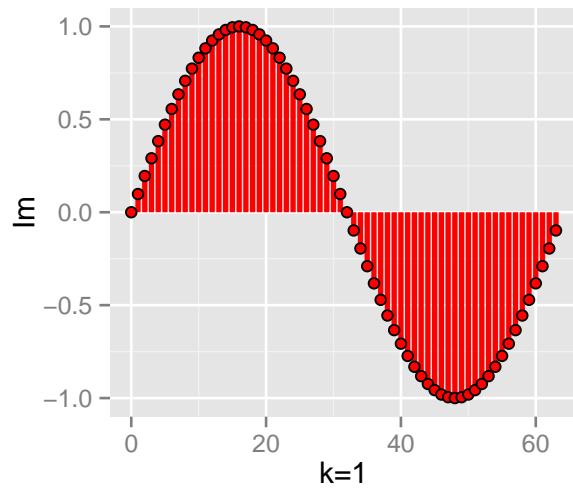
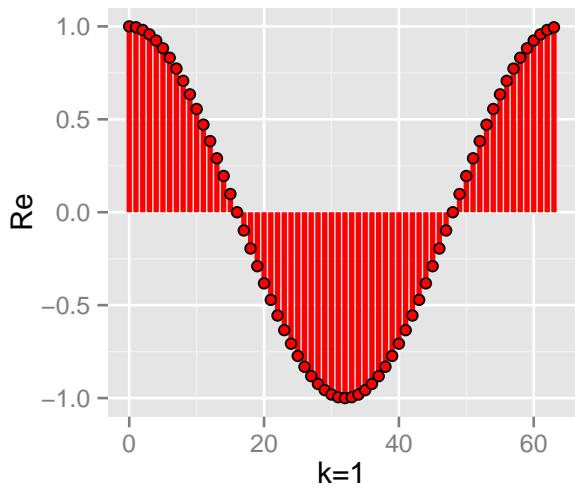
with

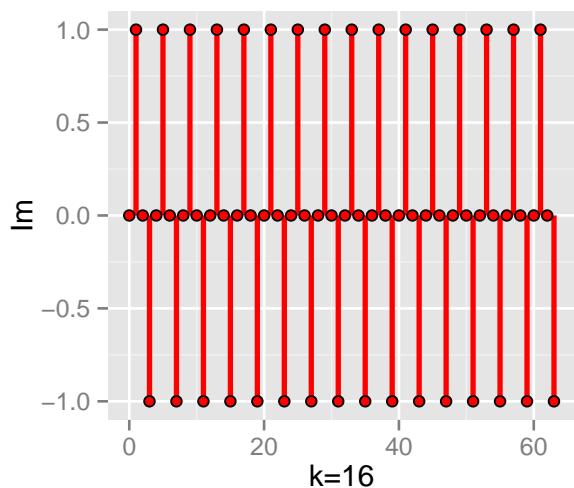
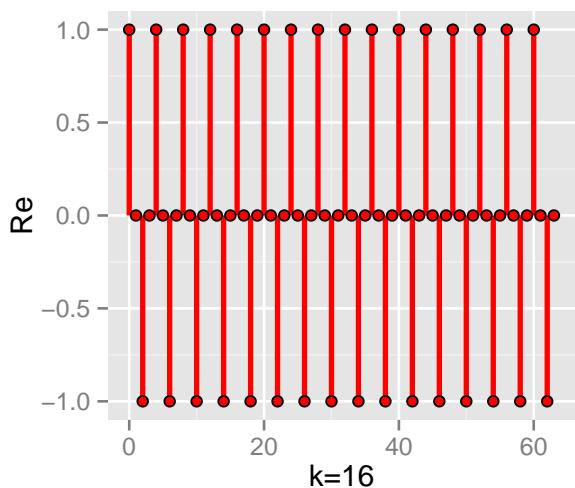
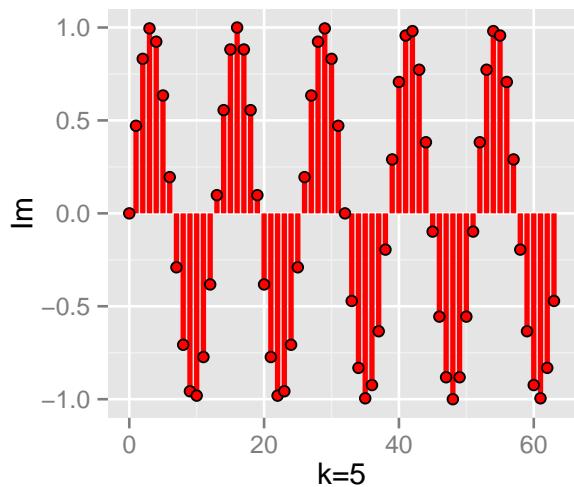
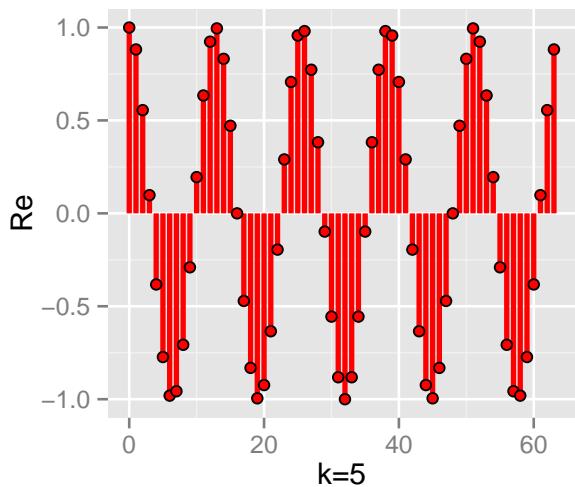
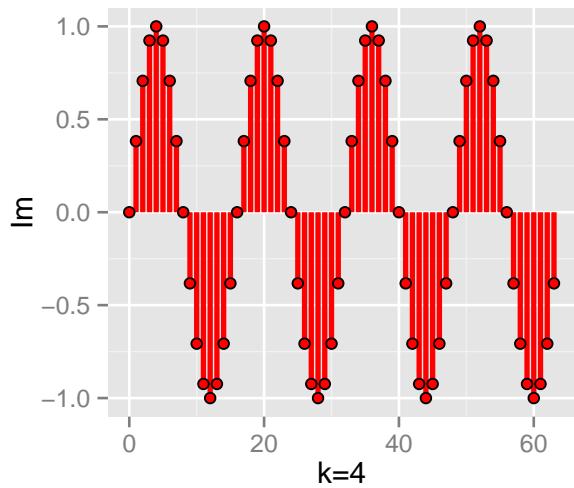
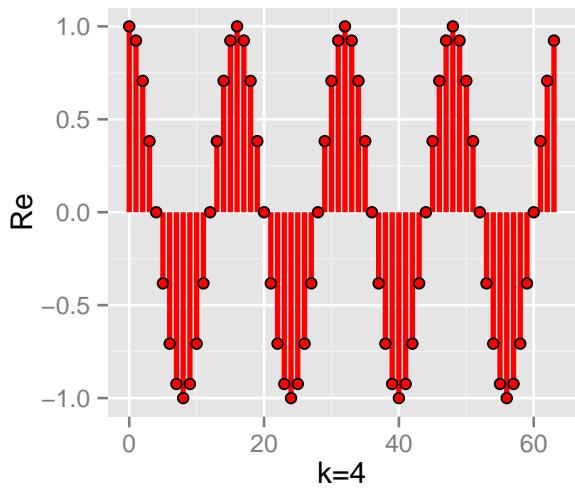
$$w_n^k = e^{j \frac{2\pi}{N} nk}$$

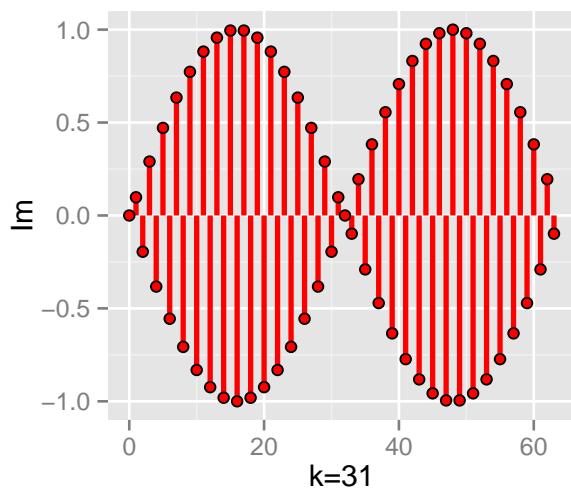
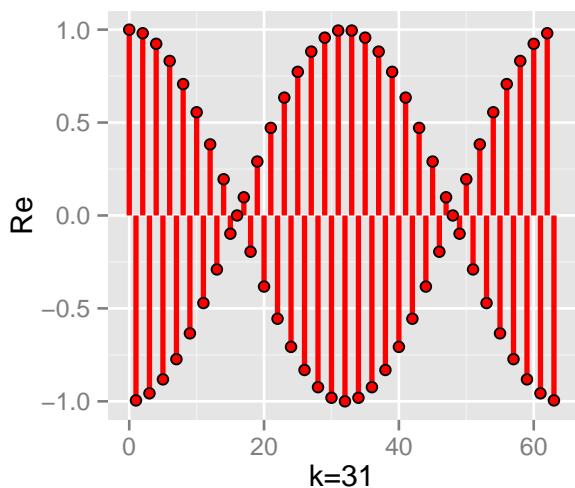
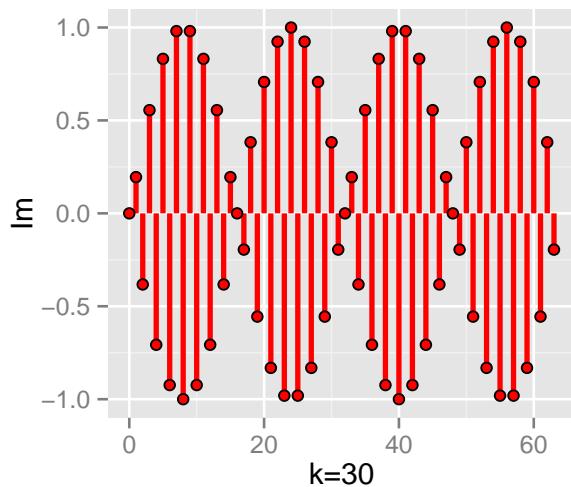
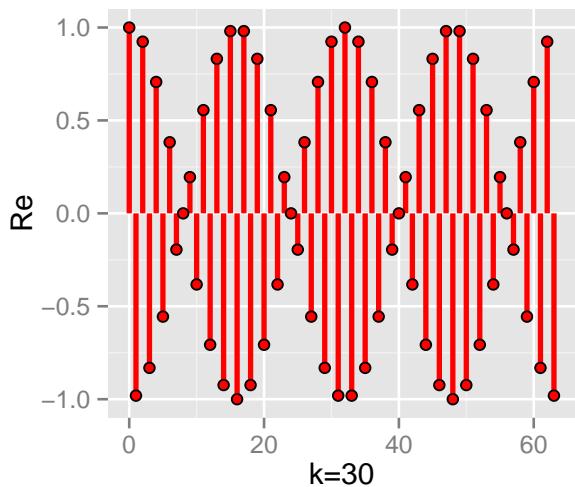
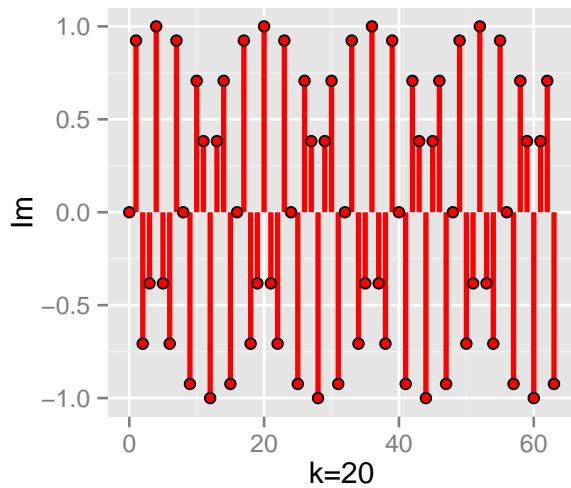
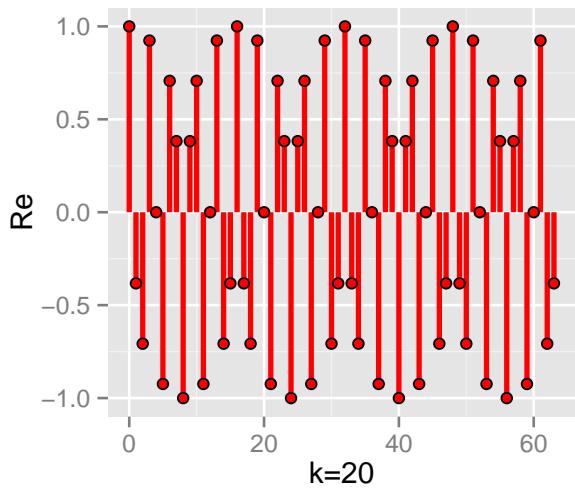
is an orthogonal basis in \mathbb{C}^N .

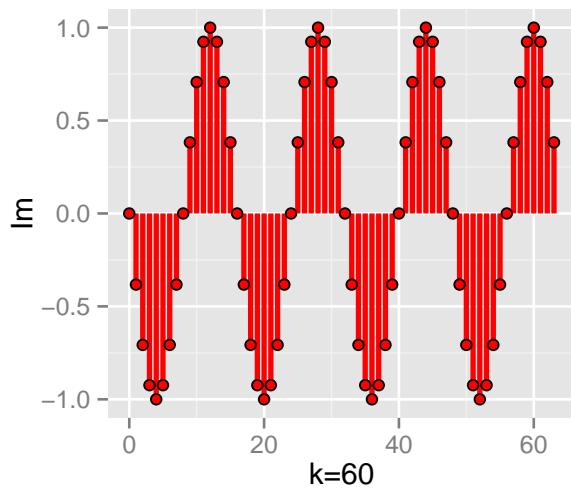
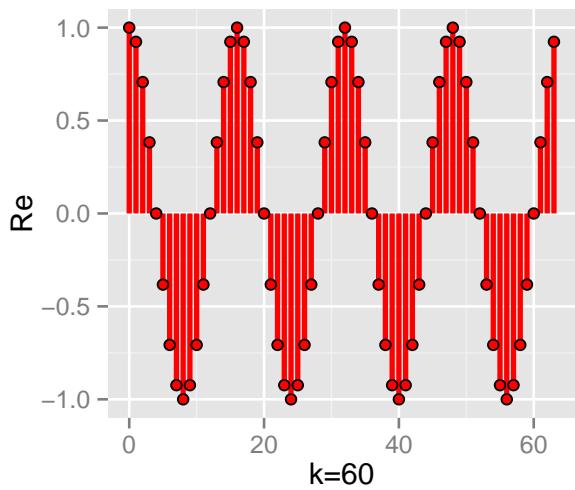
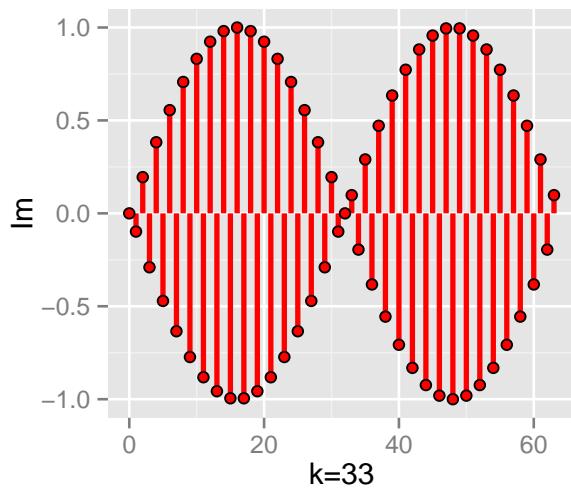
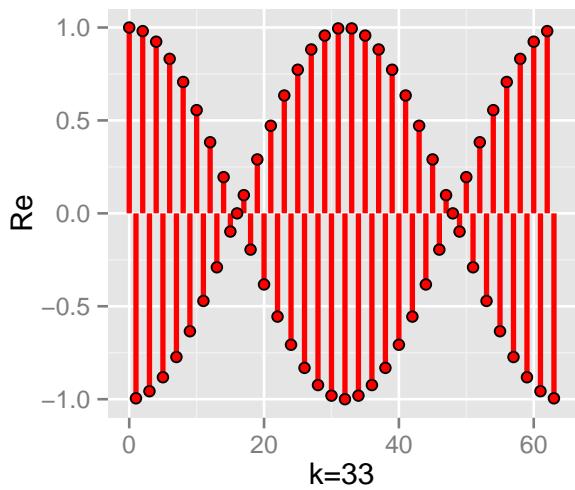
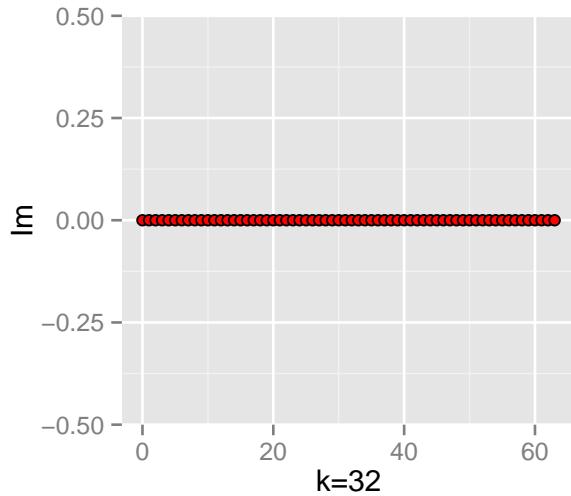
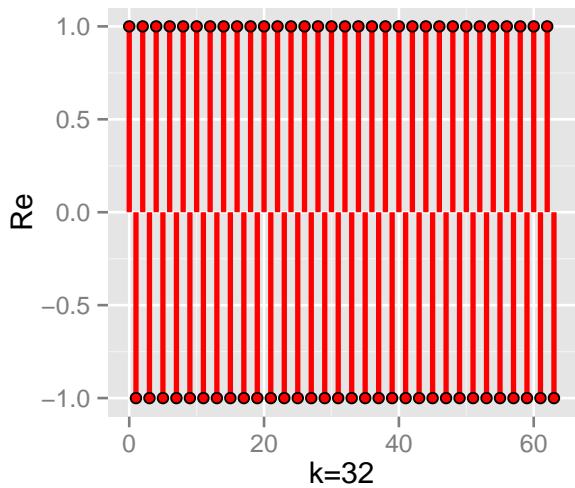
```
N <- 64
plotW <- function(k) {
  n <- 0:(N-1)
  w <- exp(1i * 2*pi/N * n * k)
  grid.arrange(dsplot(n, Re(w), xlab=paste0("k=", k), ylab="Re"),
               dsplot(n, round(Im(w),5), xlab=paste0("k=", k), ylab="Im"), ncol=2)
}
for (k in c(0,1,2,3,4,5,16,20,30,31,32,33,60,62,63))
  plotW(k)
```

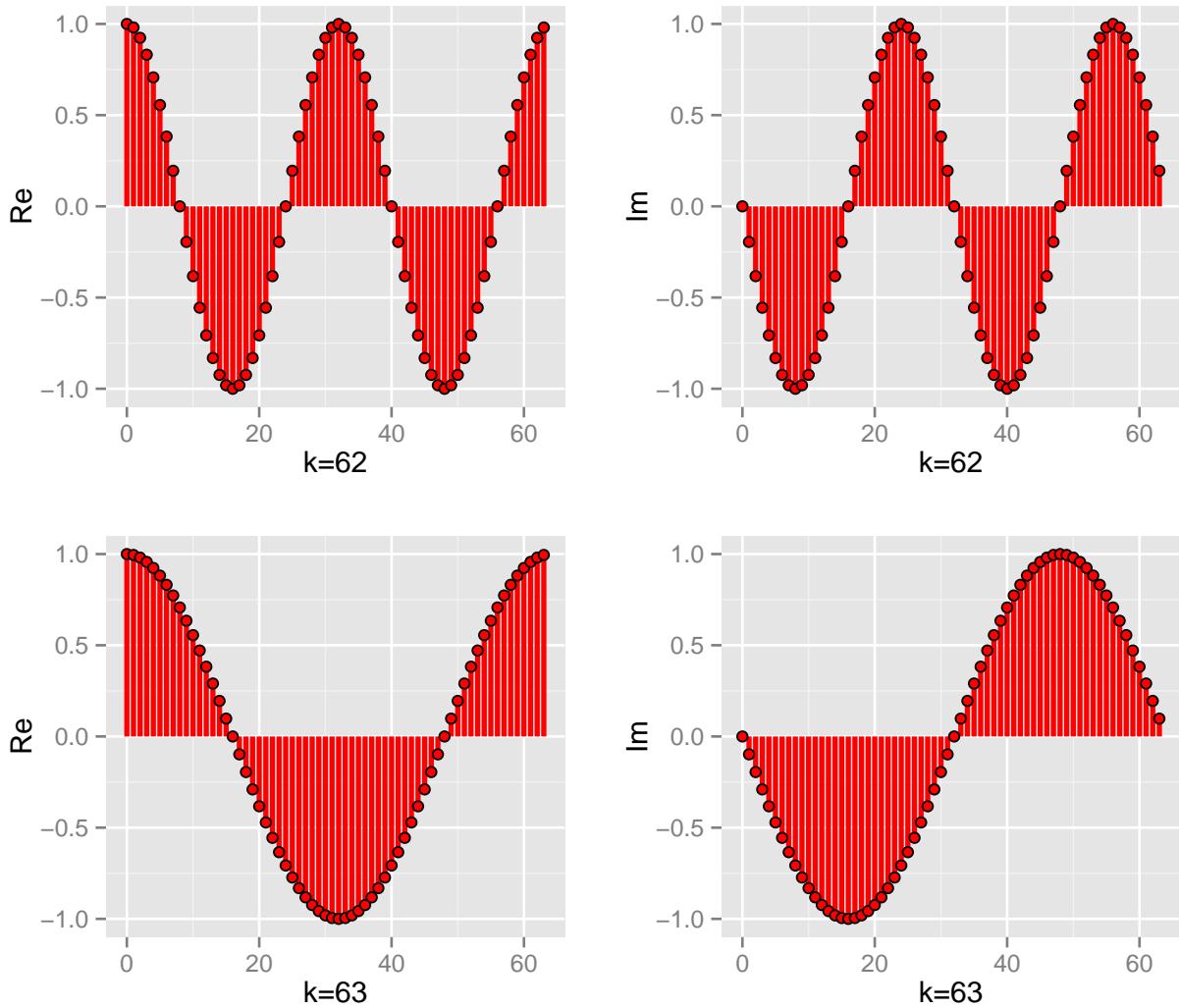












4.2 The Discrete Fourier Transform

Analysis formula:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi}{N} nk}, \quad k = 0, 1, \dots, N-1$$

N-point signal in the *frequency domain*.

Synthesis formula:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j \frac{2\pi}{N} nk}, \quad n = 0, 1, \dots, N-1$$

N-point signal in the “*time*” domain.

DFT is obviously linear:

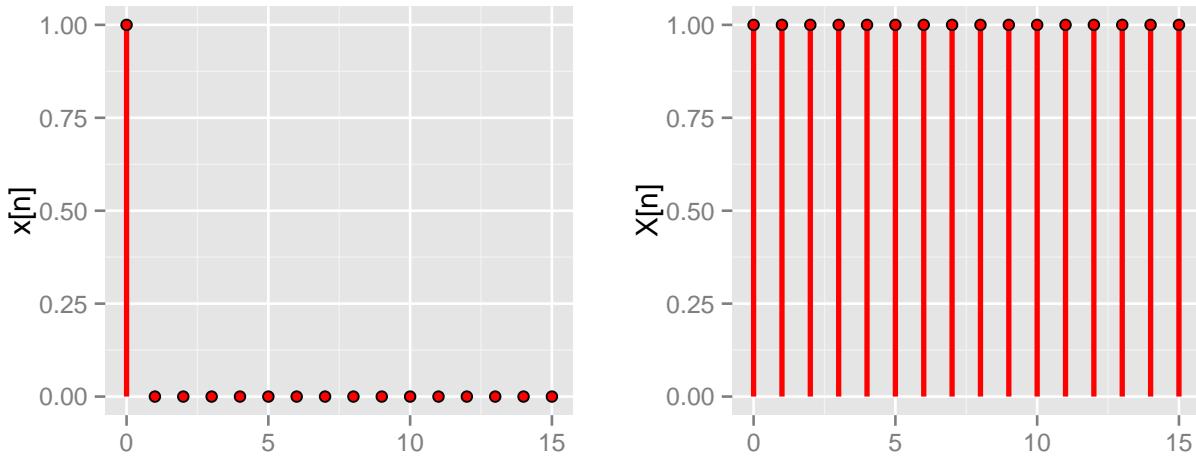
$$DFT\{\alpha x[n] + \beta y[n]\} = \alpha DFT\{x[n]\} + \beta DFT\{y[n]\}$$

4.2.1 DFT of the delta signal

$$DFTofx[n] = \sigma[n], \quad x[n] \in \mathbb{C}^N$$

$$X[k] = 1$$

```
n <- 0:15
x <- c(1, rep(0, 15))
X <- fft(x)
grid.arrange(dsplot(n, x, ylab="x[n]"), dsplot(n, Re(X), ylab="X[n]"), ncol=2)
```

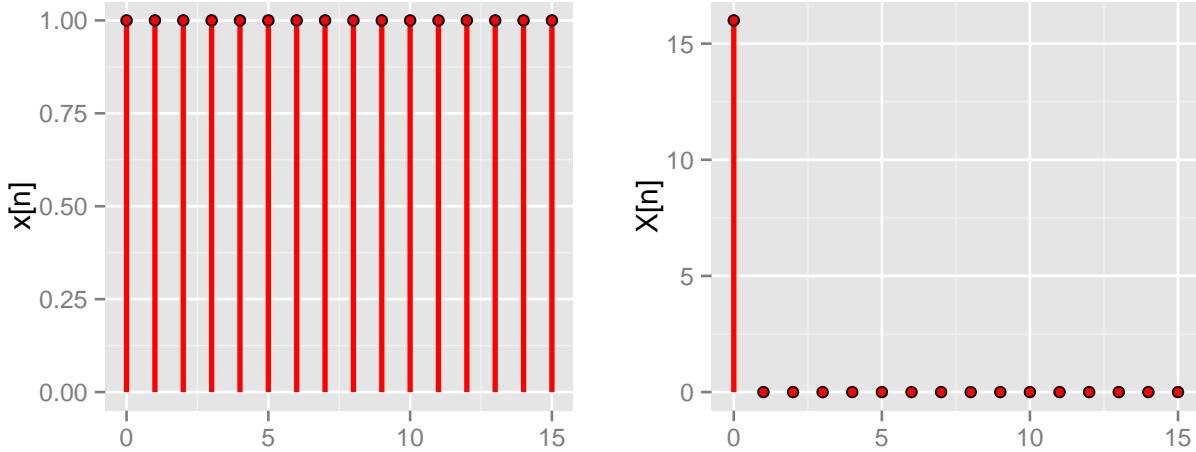


4.2.2 DFT of the uniform signal

$$DFTofx[n] = 1, \quad x[n] \in \mathbb{C}^N$$

$$X[k] = N\sigma[k]$$

```
n <- 0:15
x <- rep(1, 16)
X <- fft(x)
grid.arrange(dsplot(n, x, ylab="x[n]"), dsplot(n, Re(X), ylab="X[n]"), ncol=2)
```

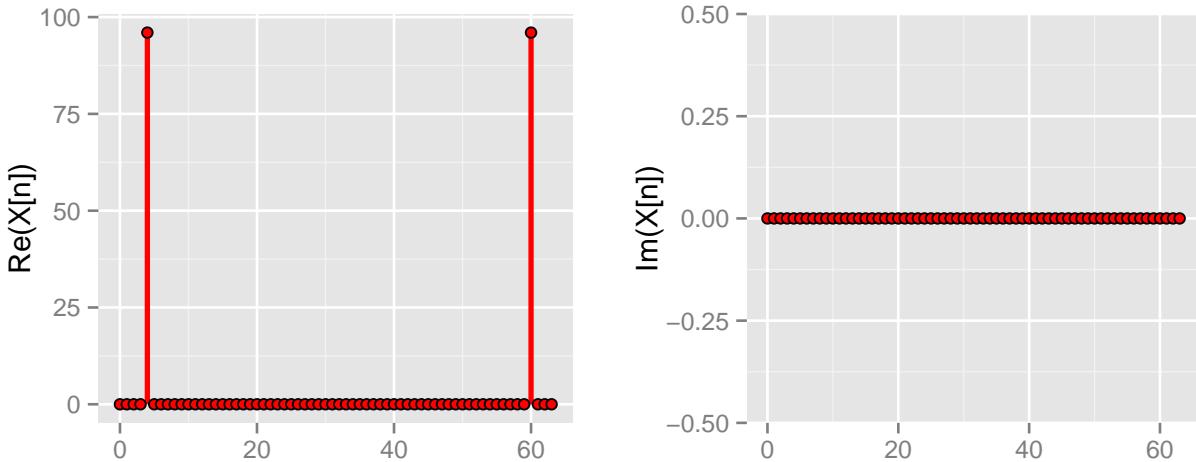


4.2.3 DFT of the cos signal

$$DFTofx[n] = 3 \cos(2\pi/16n), \quad x[n] \in \mathbb{C}^{64}$$

$$X[k] = \begin{cases} 96 & \text{for } k = 4, 60 \\ 0 & \text{otherwise} \end{cases}$$

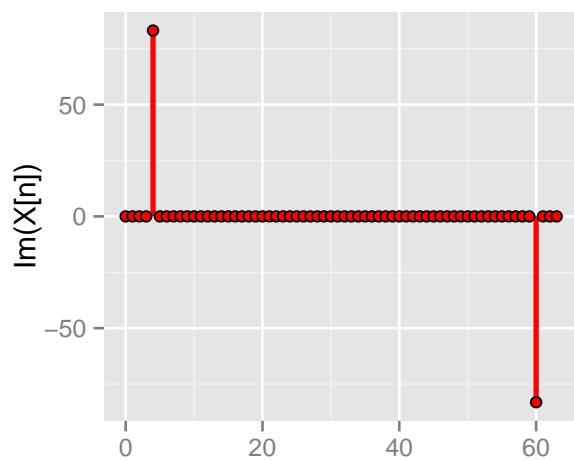
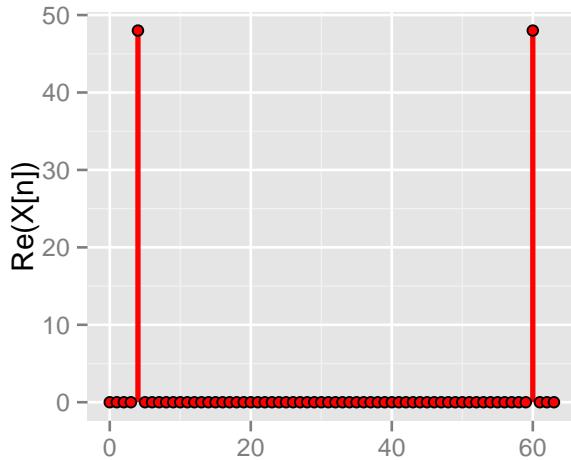
```
n <- 0:63
x <- 3 * cos(2*pi/16*n)
X <- fft(x)
grid.arrange(dsplot(n, Re(X), ylab="Re(X[n])"),
             dsplot(n, round(Im(X),4), ylab="Im(X[n])"), ncol=2)
```



$$DFTofx[n] = 3 \cos(2\pi/16n + \pi/3), \quad x[n] \in \mathbb{C}^{64}$$

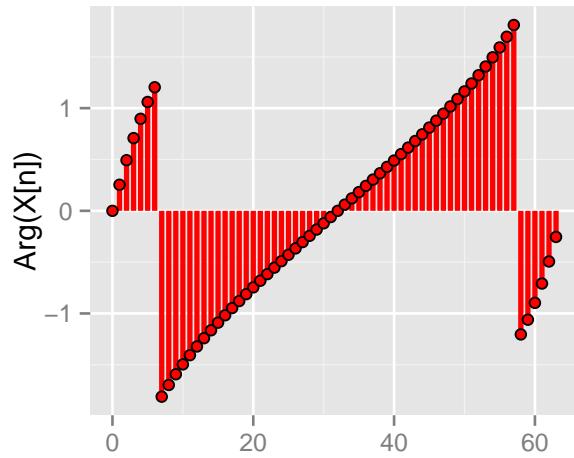
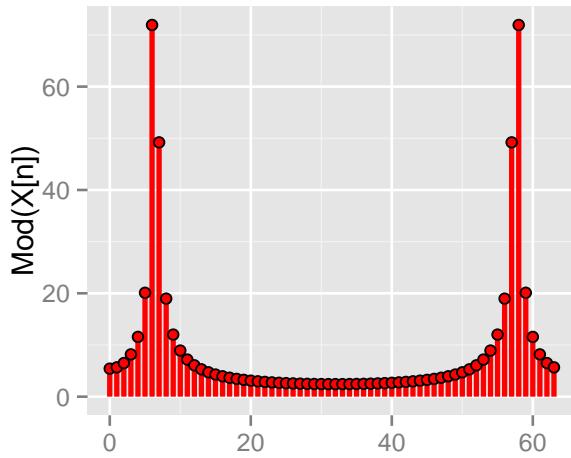
$$X[k] = \begin{cases} 96e^{j\frac{\pi}{3}} & \text{fork } 4 \\ 96e^{-j\frac{\pi}{3}} & \text{fork } 60 \\ 0 & \text{otherwise} \end{cases}$$

```
n <- 0:63
x <- 3 * cos(2*pi/16*n + pi/3)
X <- fft(x)
grid.arrange(dsplot(n, Re(X), ylab="Re(X[n])"),
             dsplot(n, round(Im(X),4), ylab="Im(X[n])"), ncol=2)
```



$$DFTofx[n] = 3 \cos(2\pi/10n), \quad x[n] \in \mathbb{C}^{64}$$

```
n <- 0:63
x <- 3 * cos(2*pi/10*n)
X <- fft(x)
grid.arrange(dsplot(n, Mod(X), ylab="Mod(X[n])"),
             dsplot(n, Arg(X), ylab="Arg(X[n])"), ncol=2)
```

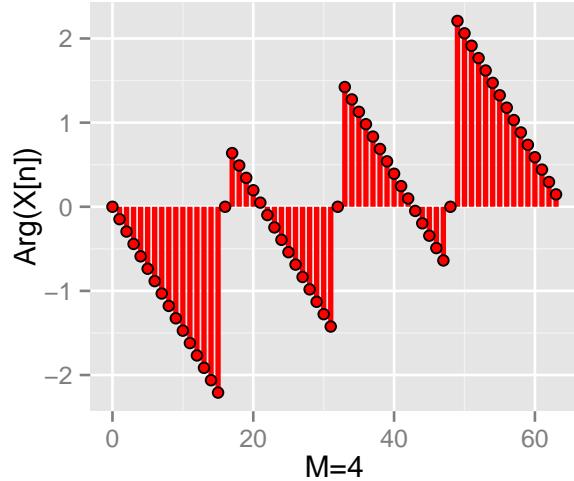
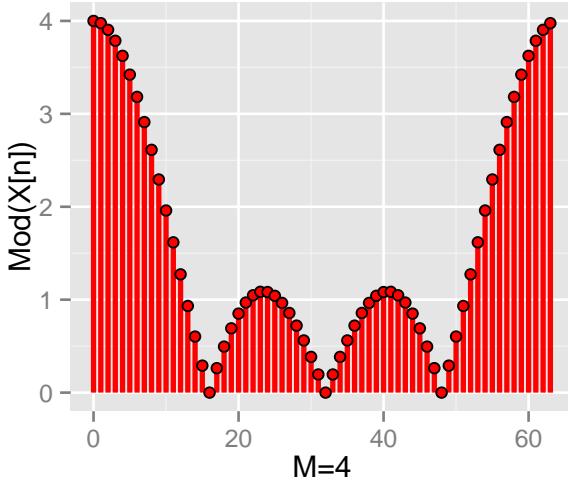


4.2.4 DFT of length-M step in \mathbb{C}^N

$$x[n] = \sum_{h=0}^{M-1} \sigma[n-h], \quad n = 0, 1, \dots, N-1$$

$$X[k] = \frac{\sin(\frac{\pi}{N} Mk)}{\sin(\frac{\pi}{N} k)} e^{-j\frac{pi}{N}(M-1)k}$$

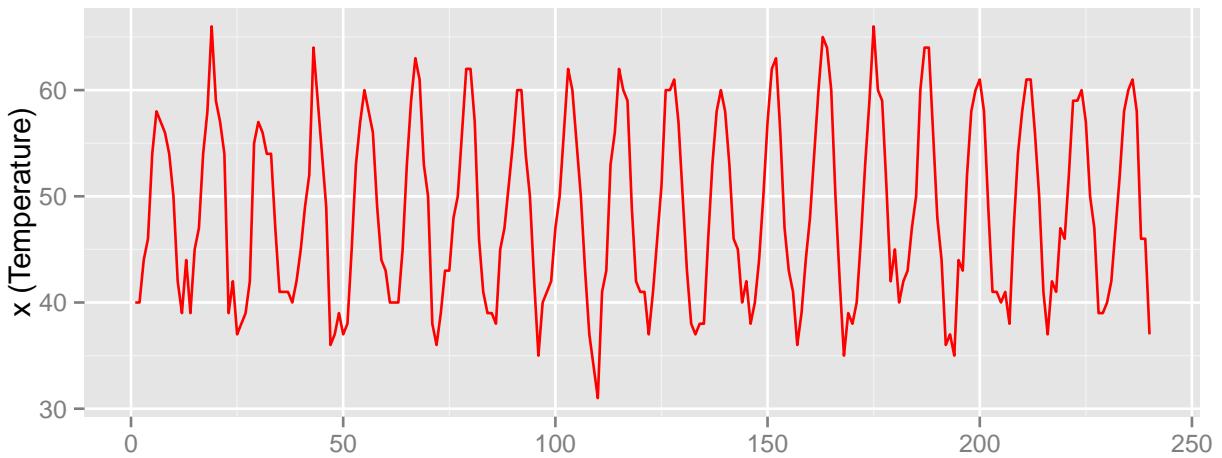
```
plotStep <- function(M) {
  n <- 0:63
  x <- c(rep(1, M), rep(0, 64-M))
  X <- fft(x)
  grid.arrange(dsplot(n, Mod(X), ylab="Mod(X[n])", xlab="M=4"),
               dsplot(n, Arg(X), ylab="Arg(X[n])", xlab="M=4"), ncol=2)
}
plotStep(4)
```



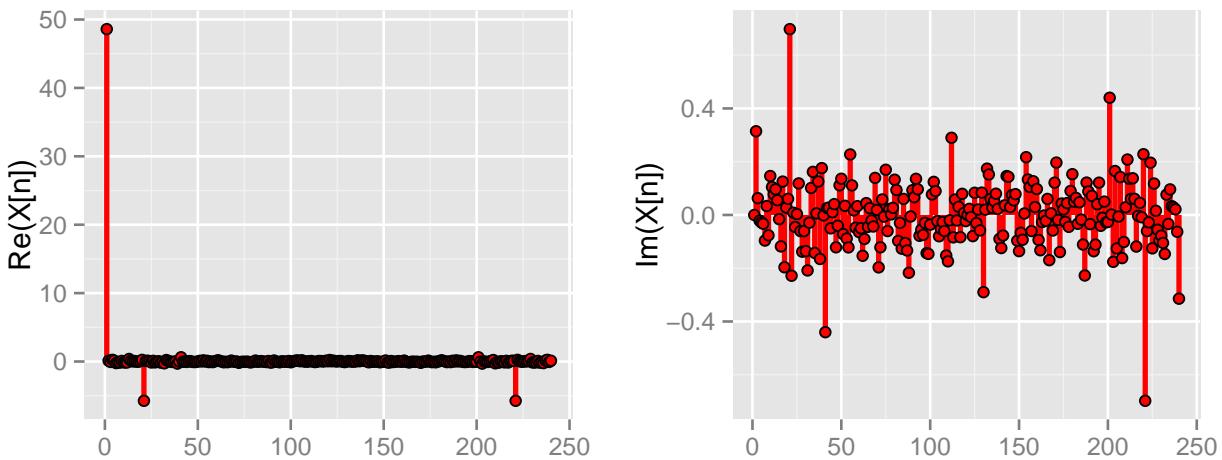
4.3 The DFT in practice

4.3.1 Average Monthly Temperatures at Nottingham, 1920-1939

```
x <- as.integer(nottem)
n <- 1:length(x)
X <- fft(x) / length(x)
csplot(n, x, ylab="x (Temperature)")
```



```
grid.arrange(dsplot(n, Re(X), ylab="Re(X[n])"),
             dsplot(n, Im(X), ylab="Im(X[n])"), ncol=2)
```



```
MainPeak = which.max(abs(Re(X[-1])))
cat("Average value:", Re(X[1]))
## Average value: 48.59583
cat("DFT main peak for k = ", MainPeak, ", value = ", abs(Re(X[MainPeak+1])))
## DFT main peak for k = 20 , value = 5.742034
cat("Month count:", length(n))
## Month count: 240
cat("Frequency: ", length(n) , "/", MainPeak, "= ", length(n)/MainPeak, "months")
## Frequency: 240 / 20 = 12 months
cat("Temperature excursion: ", Re(X[1]), "+-", 2 * abs(Re(X[MainPeak+1])))
## Temperature excursion: 48.59583 +- 11.48407
```

4.4 DFT, DFS, DTFT

4.4.1 Discrete-Time Fourier Transform (DTFT)

Formal definition:

- $x[n] \in l_2(\mathbb{Z})$
- define the *function* of $\omega \in \mathbb{R}$

$$F(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

- inversion (when $F(\omega)$ exists):

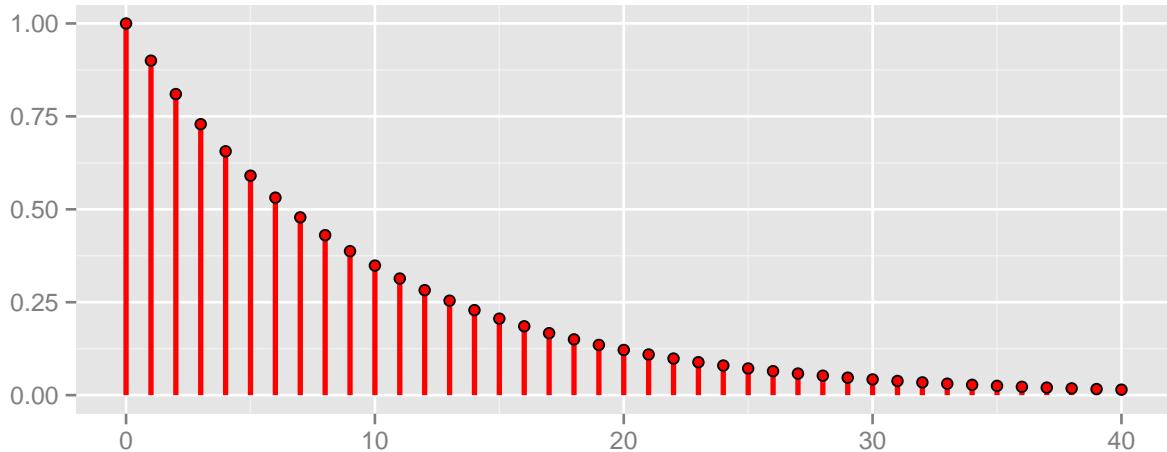
$$x[n] = \frac{1}{2\pi} \int_{-pi}^{pi} F(\omega)e^{j\omega n} d\omega$$

- $F(\omega)$ is 2π -periodic
- to stress periodicity (and for other reasons) we will write

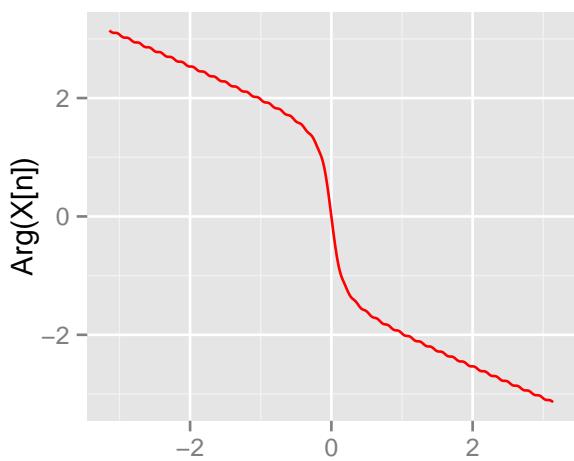
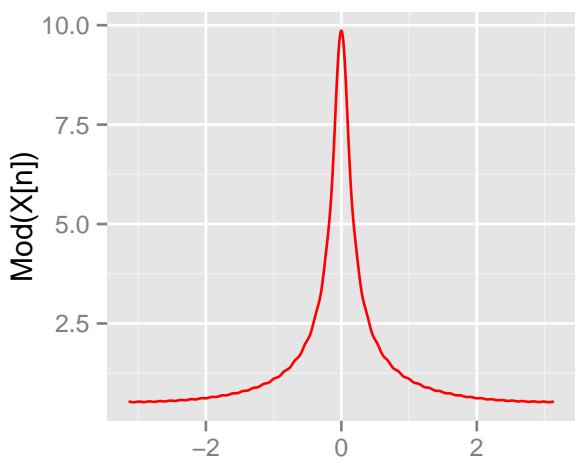
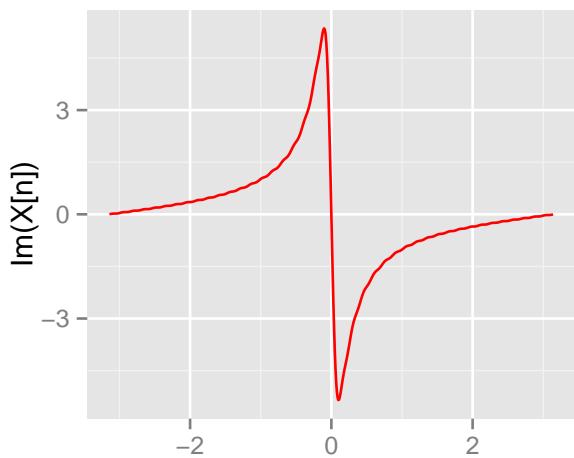
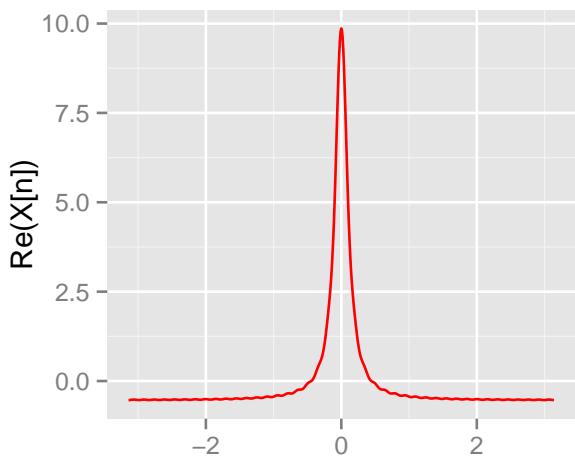
$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

- by convention, $X(e^{j\omega})$ is represented over $[-pi; pi]$

```
alpha <- 0.9; n <- 0:40
x <- alpha^n
dsplot(n, x)
```



```
dtftplot(x)
```



4.5 DTFT: intuition and properties

4.5.1 DTFT properties

- Linearity

$$DTFT\{\alpha x[n] + \beta y[n]\} = \alpha X(e^{j\omega}) + \beta Y(e^{j\omega})$$

- Time shift

$$DTFT\{x[n - M]\} = e^{-j\omega M} X(e^{j\omega})$$

- Modulation (dual)

$$DTFT\{e^{j\omega_0 n} x[n]\} = X(e^{j(\omega - \omega_0)})$$

- Time reversal

$$DTFT\{x[-n]\} = X(e^{-j\omega})$$

- Conjugation

$$DTFT\{x^*[n]\} = X^*(e^{-j\omega})$$

4.5.2 Some particular cases

- if $x[n]$ is symmetric, the DTFT is symmetric:

$$x[n] = x[-n] \Leftrightarrow X(e^{j\omega}) = X(e^{-j\omega})$$

- if $x[n]$ is real, the DTFT is Hermitian-symmetric:

$$x[n] = x^*[n] \Leftrightarrow X(e^{j\omega}) = X^*(e^{-j\omega})$$

- special case: if $x[n]$ is real, the magnitude of the DTFT is symmetric:

$$x[n] \in \mathbb{R} \Rightarrow |X(e^{j\omega})| = |X(e^{-j\omega})|$$

- more special case: if $x[n]$ is real and symmetric, $X(e^{j\omega})$ is also real and symmetric

4.5.3 DTFT as basic expansion

- $DFT\{\delta[n]\} = 1$
- $DTFT\{\delta[n]\} = 1$
- $DFT\{1\} = N\delta[k]$
- $DTFT\{1\} = \tilde{\delta}(\omega)$
- $DTFT\{e^{j\omega_0 n}\} = \tilde{\delta}(\omega - \omega_0)$
- $DTFT\{\cos(\omega_0 n)\} = (\tilde{\delta}(\omega - \omega_0) + \tilde{\delta}(\omega + \omega_0))/2$
- $DTFT\{\sin(\omega_0 n)\} = -j(\tilde{\delta}(\omega - \omega_0) - \tilde{\delta}(\omega + \omega_0))/2$

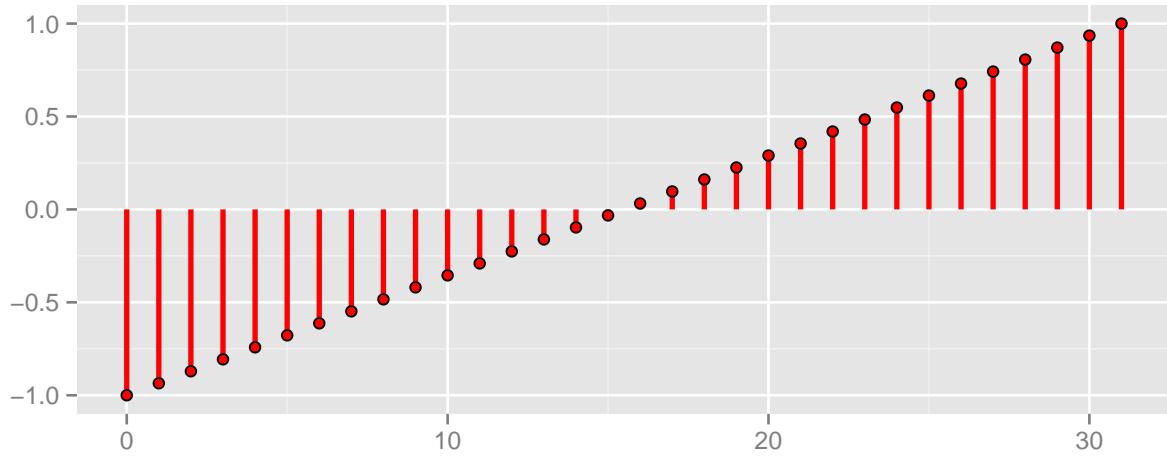
4.6 Relationship between transforms

4.6.1 DTFT of periodic signals

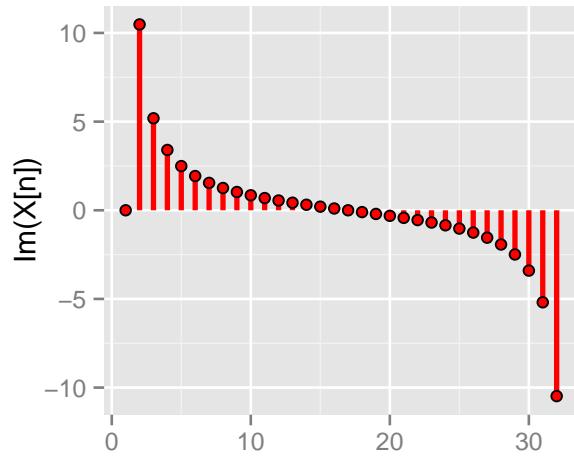
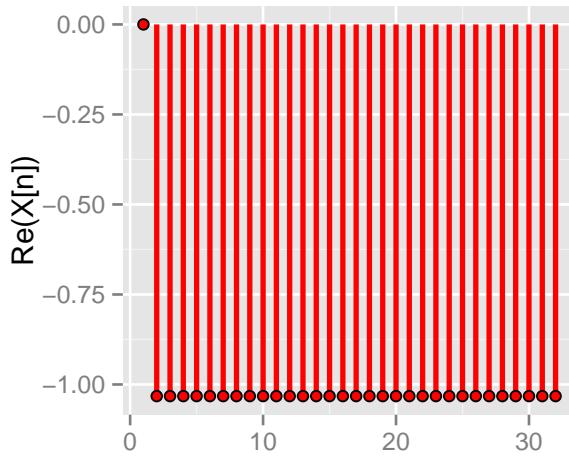
$$\tilde{X}(e^{j\omega}) = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \delta(\omega - \frac{2\pi}{N}k)$$

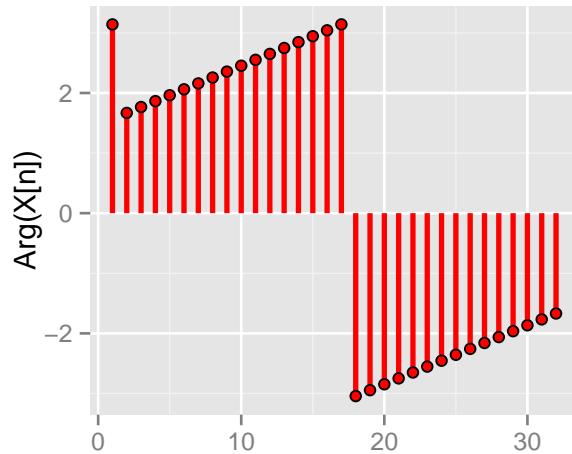
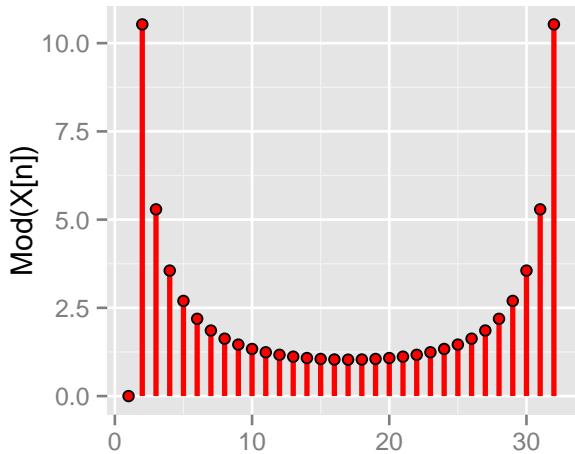
4.6.2 32-tap sawtooth

```
n <- 0:31
x <- -1 + 2*n/31
dspplot(n, x)
```

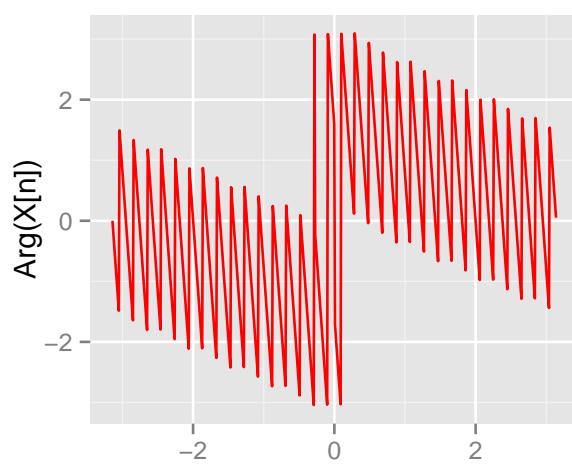
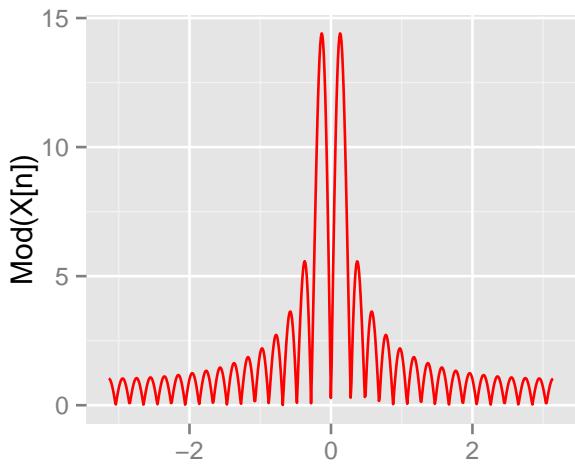
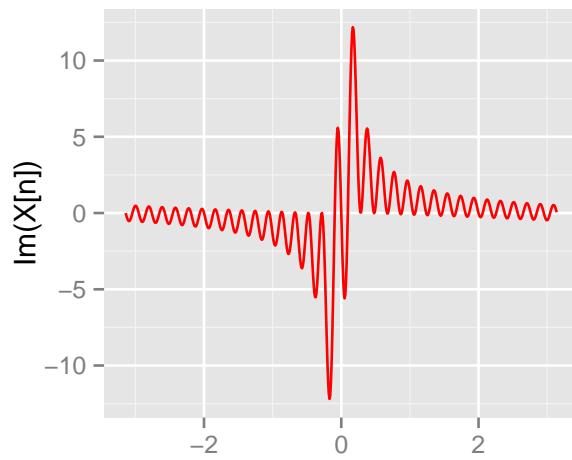
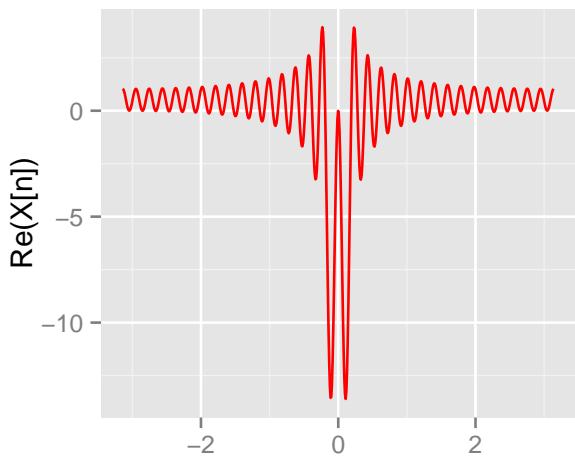


```
dftplot(x)
```



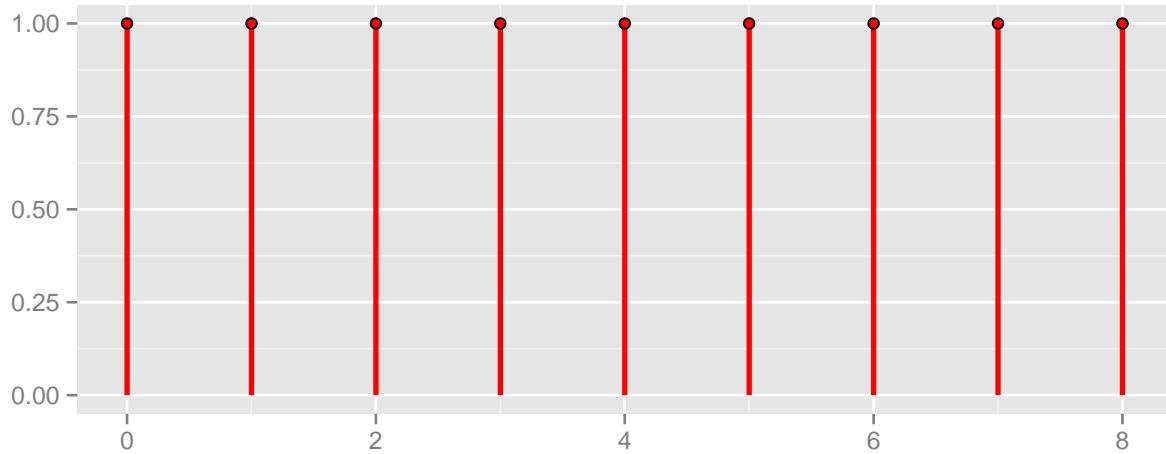


```
dtftplot(x)
```

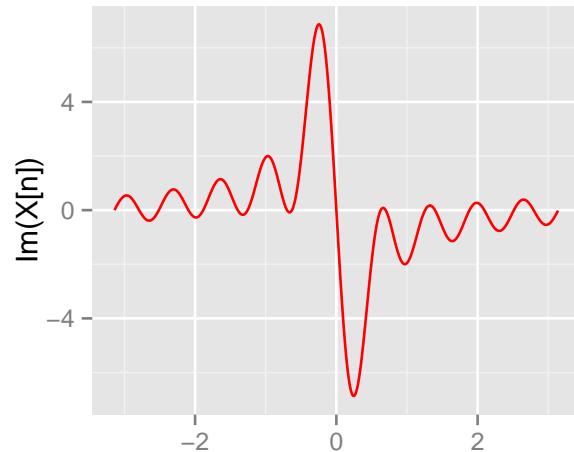
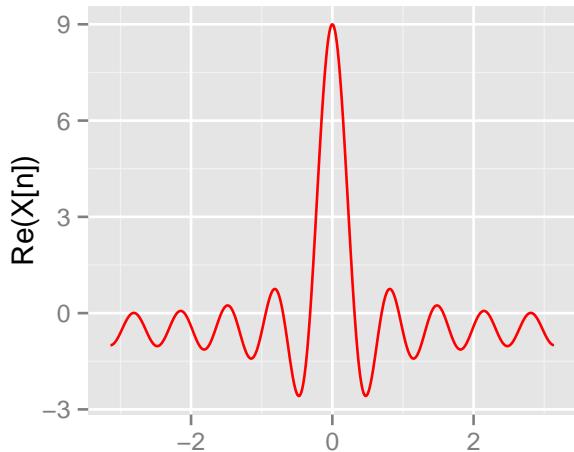


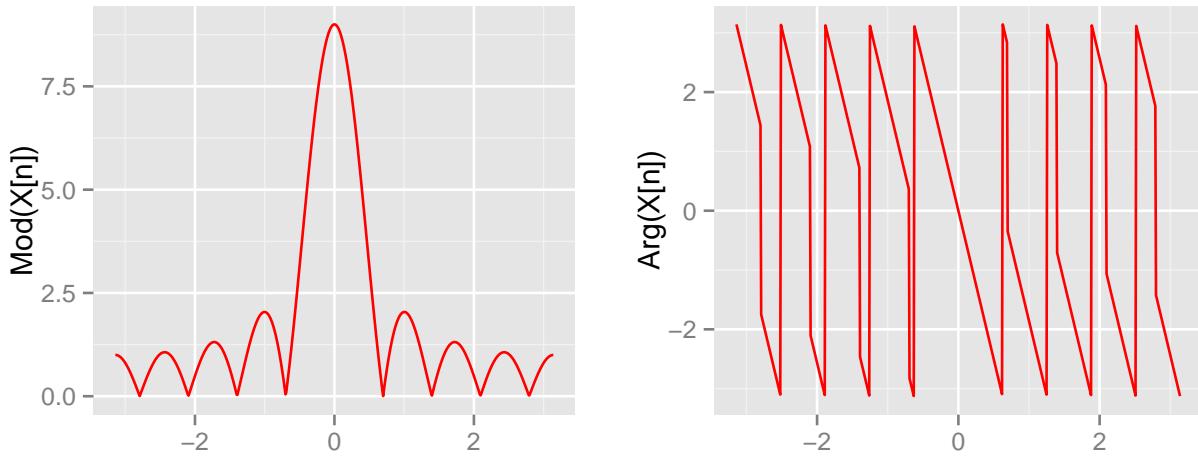
4.6.3 Interval indicator signal

```
N <- 9  
n <- 0:(N-1)  
x <- rep(1, N)  
dsplot(n, x)
```



```
dtftplot(x)
```





4.7 Sinusoidal modulation

4.7.1 Classifying signals in frequency

Three broad categories according to where most of the spectral energy resides:

- lowpass signals (also known as “baseband” signals)
- highpass signals
- bandpass signals

4.7.2 Sinusoidal modulation

$$\text{DTFT}\{x[n] \cos(\omega_c n)\} = \text{DTFT}\left\{\frac{1}{2}e^{j\omega_c n}x[n] + \frac{1}{2}e^{-j\omega_c n}x[n]\right\} = \frac{1}{2}\left(X(e^{j(\omega-\omega_c)}) + X(e^{j(\omega+\omega_c)})\right)$$

- usually $x[n]$ baseband
- ω_c is the *carrier* frequency

Applications:

- voice and music are lowpass signals
- radio channels are bandpass, in much higher frequencies
- modulation brings the baseband signal in the transmission band
- demodulation at the receiver brings it back

Demodulation in the frequency domain:

- we recovered the baseband signal exactly...
- but we have some spurious high-frequency components
- in the next Module we will learn how to get rid of them!

4.7.3 Tuning a guitar

Problem (abstraction):

- reference sinusoid at frequency ω_0
- tunable sinusoid of frequency ω
- make $\omega = \omega_0$ "by ear"

The procedure:

1. bring ω close to ω_0 (easy)
2. when $\omega \approx \omega_0$ play both sinusoid together
3. trigonometry comes to the rescue:

$$x[n] = \cos(\omega_0 n) + \cos(\omega n) = 2 \cos\left(\frac{\omega_0 + \omega}{2}n\right) \cos\left(\frac{\omega_0 - \omega}{2}n\right) \approx 2 \cos(\Delta\omega n) \cos(\omega_0 n)$$

Let's see what's happening:

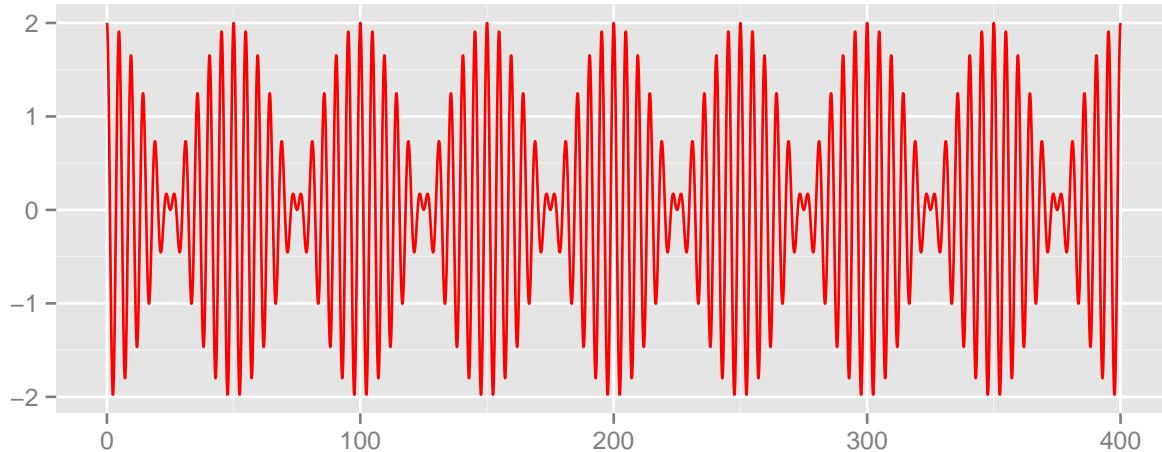
$$x[n] \approx 2 \cos(\Delta\omega n) \cos(\omega_0 n)$$

- "error" signal: $2 \cos(\Delta\omega n)$
- modulation at ω_0 : $\cos(\omega_0 n)$
- when $\omega \approx \omega_0$, the error signal is too low to be heard; modulation brings it up to hearing range and we perceive it as amplitude oscillations of the carrier frequency

Examples in the time domain :

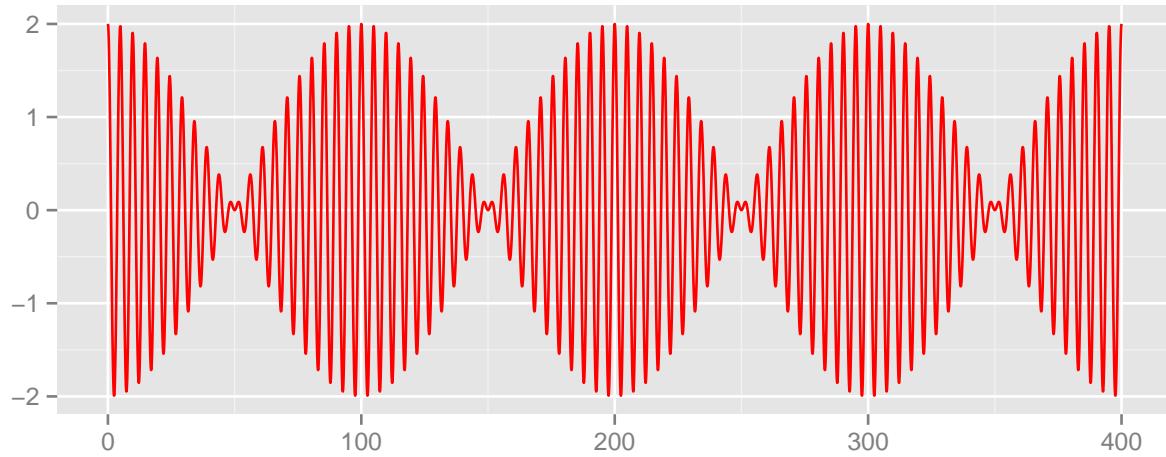
$$\omega_0 = 2\pi \cdot 0.2, \omega = 2\pi \cdot 0.22, \Delta\omega = 2\pi \cdot 0.0100$$

```
omega0 <- 2*pi*0.2
omega <- 2*pi*0.22
t <- seq(0, 400, by=0.1)
x <- cos(omega0*t) + cos(omega*t)
csplot(t, x)
```



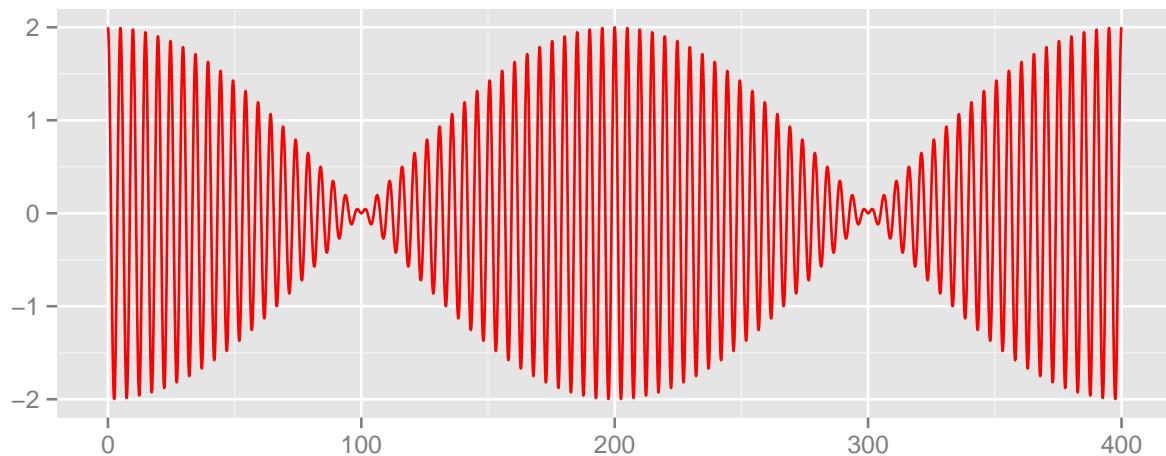
$$\omega_0 = 2\pi \cdot 0.2, \omega = 2\pi \cdot 0.21, \Delta\omega = 2\pi \cdot 0.0050$$

```
omega0 <- 2*pi*0.2
omega <- 2*pi*0.21
t <- seq(0, 400, by=0.1)
x <- cos(omega0*t) + cos(omega*t)
cspolt(t, x)
```



$$\omega_0 = 2\pi \cdot 0.2, \omega = 2\pi \cdot 0.205, \Delta\omega = 2\pi \cdot 0.0025$$

```
omega0 <- 2*pi*0.2
omega <- 2*pi*0.205
t <- seq(0, 400, by=0.1)
x <- cos(omega0*t) + cos(omega*t)
cspolt(t, x)
```

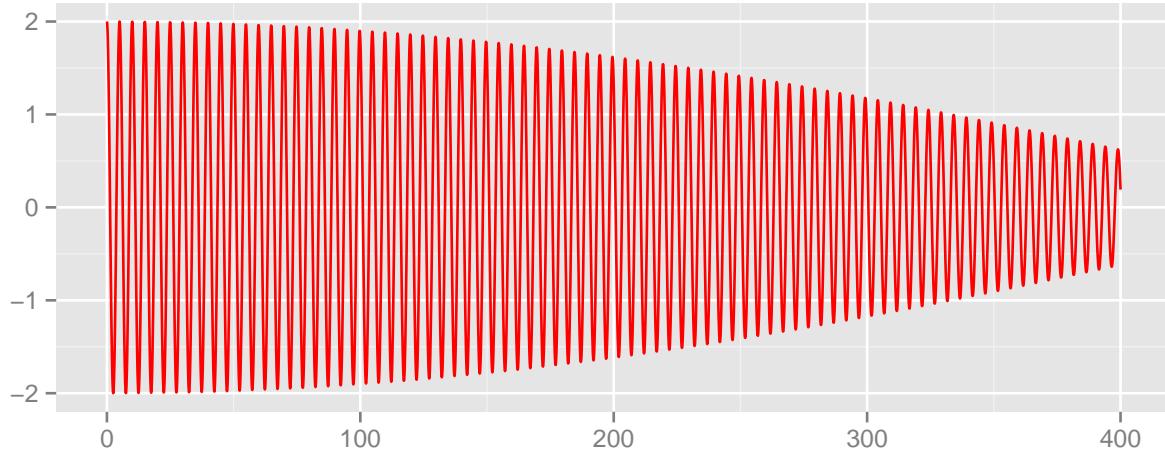


$$\omega_0 = 2\pi \cdot 0.2, \omega = 2\pi \cdot 0.201, \Delta\omega = 2\pi \cdot 0.0005$$

```

omega0 <- 2*pi*0.2
omega <- 2*pi*0.201
t <- seq(0, 400, by=0.1)
x <- cos(omega0*t) + cos(omega*t)
csplot(t, x)

```



4.8 The Short-Time Fourier Transform

Dual-Tone Multi Frequency (DTMF) dialing:

```

f123 <- 697
f456 <- 770
f789 <- 852
fs0s <- 941
f147s <- 1209
f2580 <- 1336
f369s <- 1477

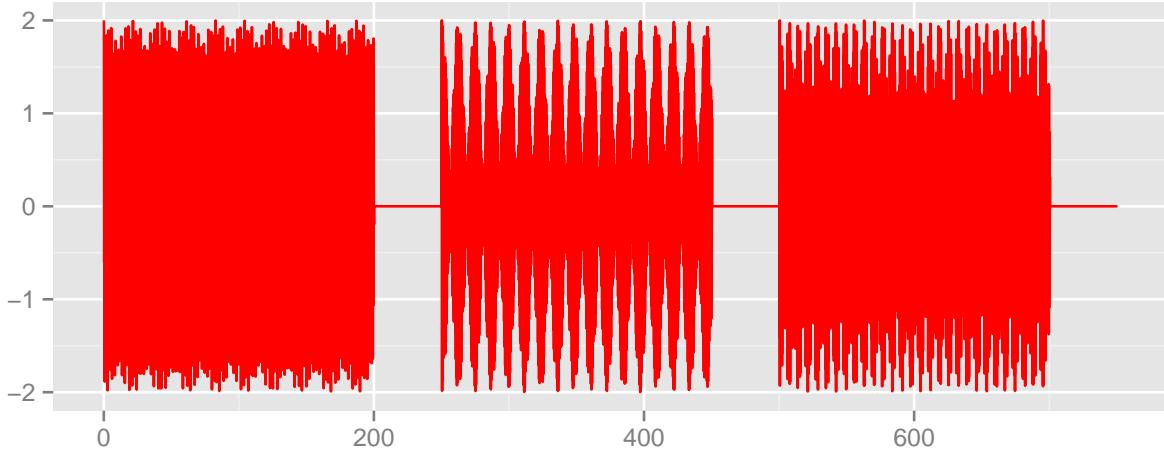
```

4.8.1 Example: 1-5-9

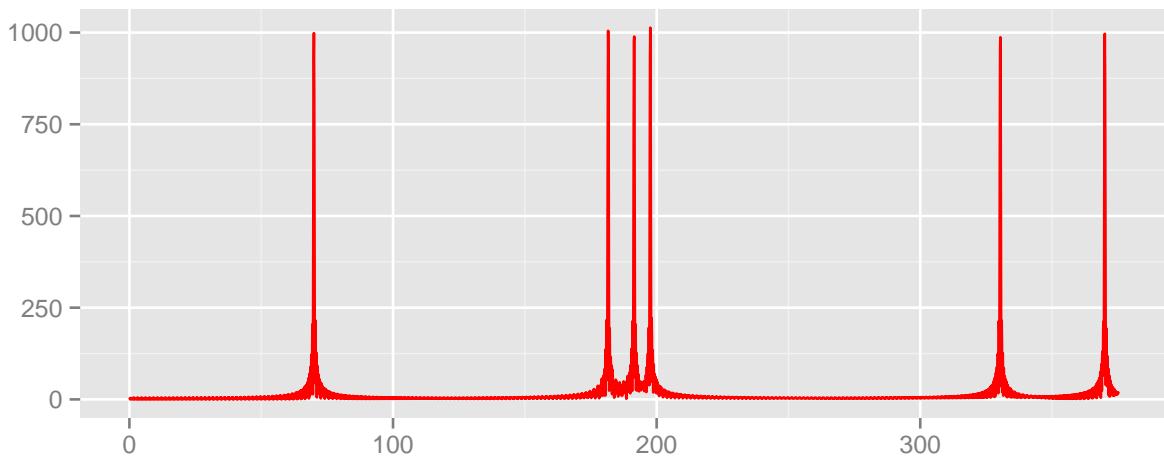
```

step <- 0.1
t1 <- seq(0, 200, by=step)
t2 <- seq(0, 50, by=step)
x <- c(
  cos(t1 * f123) + cos(t1 * f147s), # 1
  t2 * 0,
  cos(t1 * f456) + cos(t1 * f2580), # 5
  t2 * 0,
  cos(t1 * f789) + cos(t1 * f369s), # 9
  t2 * 0)
N <- length(x)
t <- (1:N)*step
csplot(t, x)

```



```
X <- fft(x)
csplot(t[1:(N/2)], Mod(X)[1:(N/2)])
```



4.8.2 Short-Time Fourier Transform

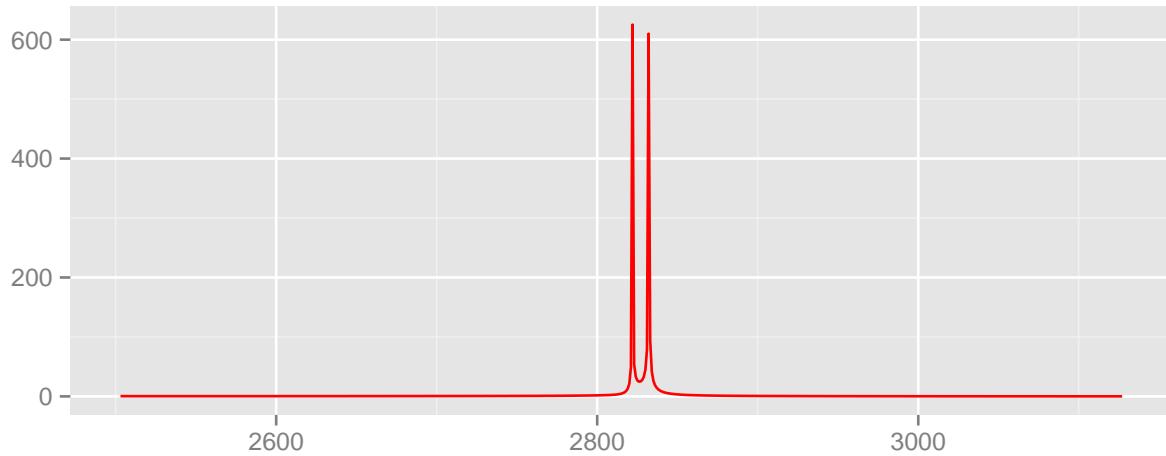
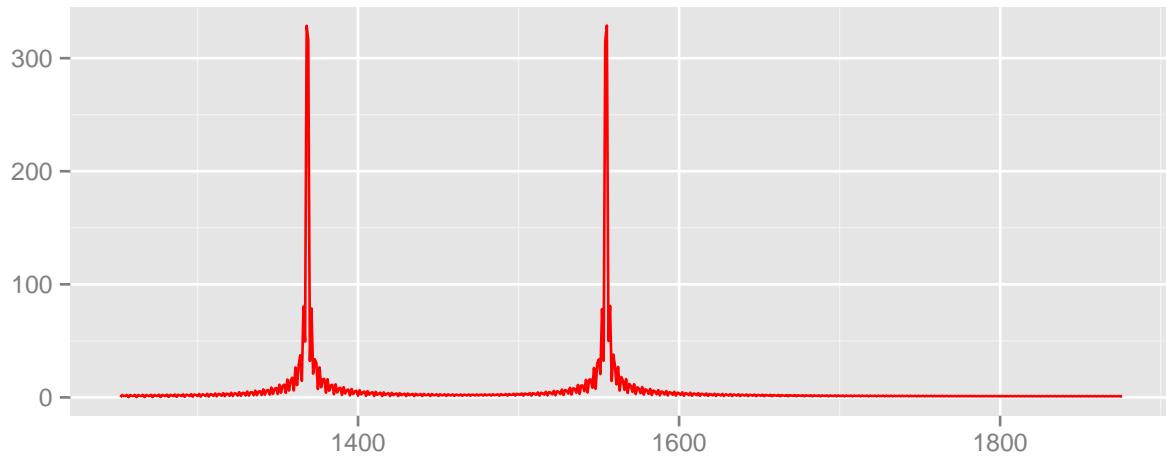
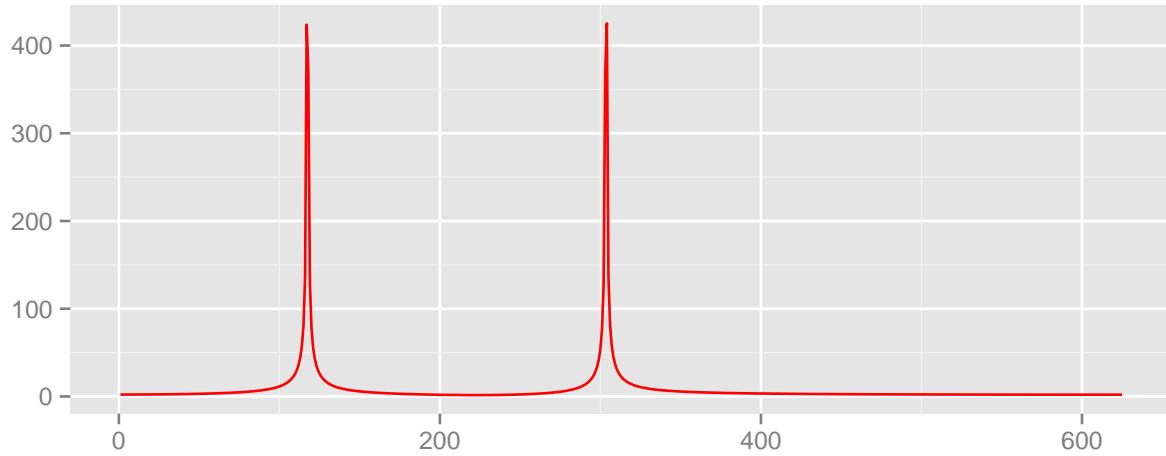
Idea:

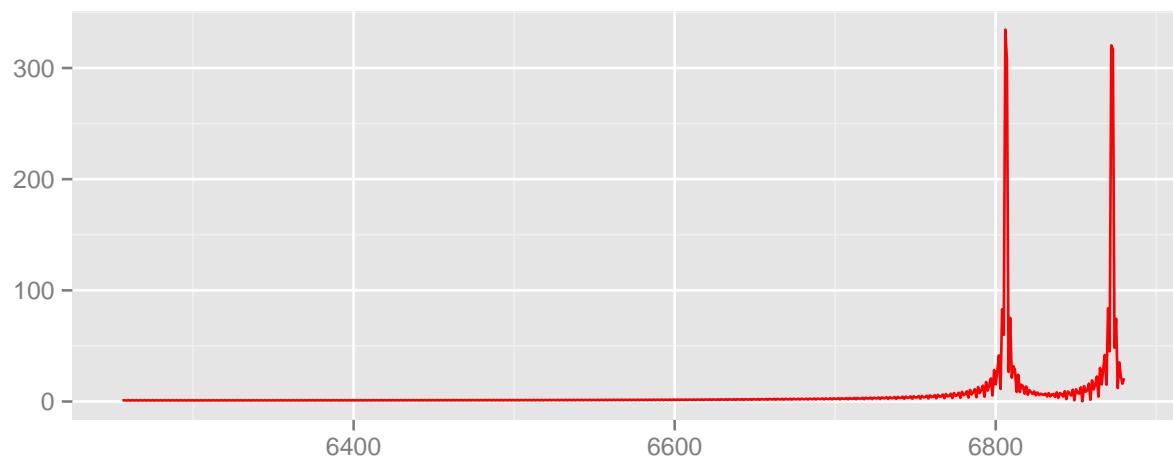
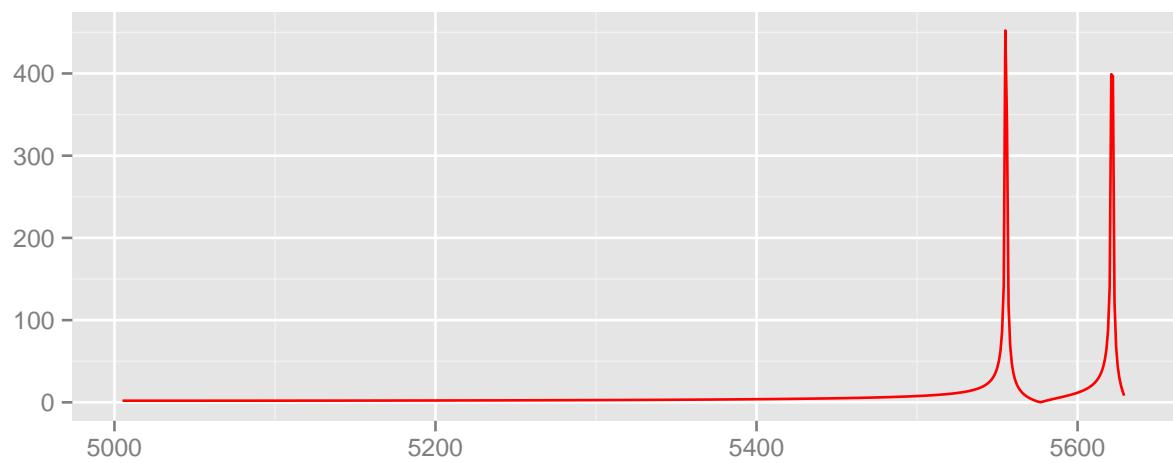
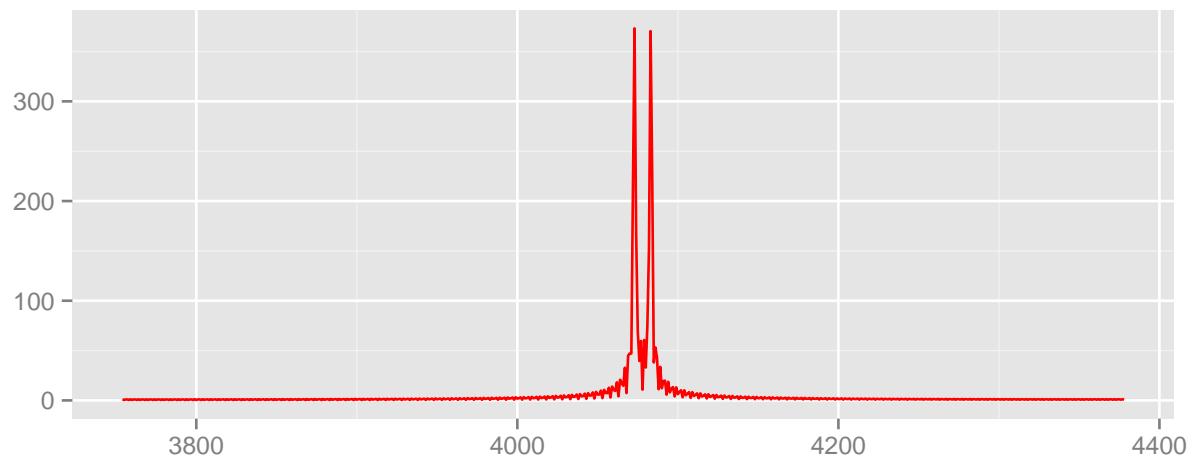
- take small signal pieces of length L
- look at the DFT of each piece:

$$X[m; k] = \sum_{n=0}^{L-1} x[m+n] e^{-j \frac{2\pi}{L} nk}$$

```
count <- 6
L <- N / count
for (i in 1:count) {
  subt <- (L*(i - 1)+1):(L*i)
  X <- fft(x[subt])
```

```
    print(cspplot(subt[1:(L/2)], Mod(X)[1:(L/2)]))  
}
```



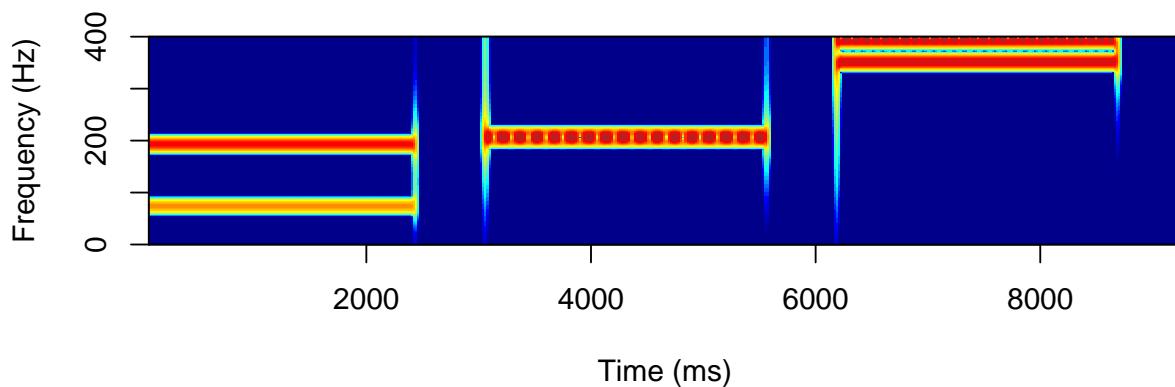


4.8.3 Spectrogram

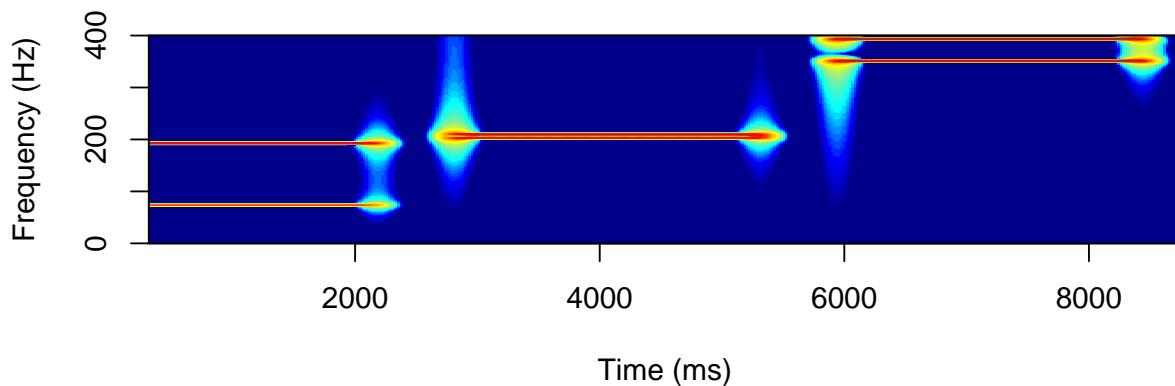
Idea:

- color-code the magnitude: dark is small, white is large
- use $10\log_{10}(|X[m;k]|)$ to see better (power in dBs)
- plot spectral slices one after another

```
spectrogram(x, fs=800, windowlength=N/60)
```



```
spectrogram(x, fs=800, windowlength=N/12)
```



4.9 FFT: history and algorithm

4.9.1 History

- Gauss computes trigonometric series efficiently in 1805
- Fourier invents Fourier series in 1807
- People start computing Fourier series, and develop tricks
- Good comes up with an algorithm in 1958
- Cooley and Tukey (re)-discover the fast Fourier transform algorithm in 1965 for N a power of a prime
- Winograd combines all methods to give the most efficient FFTs

4.9.2 The DFT matrix

- $W_N = e^{-j\frac{2\pi}{N}}$ (or simply W when N is clear from the context)
- power of N can be taken modulo N , since $W^N = 1$.
- DFT Matrix of size N by N :

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W^1 & W^2 & W^3 & \dots & W^{N-1} \\ 1 & W^2 & W^4 & W^6 & \dots & W^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W^{N-1} & W^{2(N-1)} & W^{3(N-1)} & \dots & W^{(N-1)^2} \end{bmatrix}$$

4.9.3 Divide and Conquer for DFT — One step

Recall: computing $X = W_n x$ has complexity $O(N^2)$

Idea:

- Take a problem of size N where N is a power of 2.
- Cut into two problems of size $N/2$ that use complexity $N^4/4$ each
- There might be some complexity to recover the full solution, say N .
- The divide-and-conquer solution has complexity $N^2/2 + N$ for one step
- For $N \geq 4$ this is much better than N^2 !

4.9.4 Divide and Conquer for DFT — Multiple steps

Divide and conquer can be reapplied

- If it worked once, it will work again (recall, $N = 2^K$)
- the two problems of size $N/2$ into 4 problems of size $N/4$
- There might be some complexity to recover the full solution, say N at each step
- You can do this $\log_2(N - 1) = K - 1$ times, until problem of size 2 is obtained
- This requires order N complexity each time
- The divide-and-conquer solution has therefore complexity of order $N \log_2 N$
- For $N \geq 4$ this is much better than N^2 !

4.9.5 Conclusion

Don't worry, be happy!

- The Cooley-Tukey is the most popular algorithm, mostly for $N = 2^N$
- Note that there is always a good FFT algorithm around the corner
- Do not zero-pad to lengthen a vector to have a size equal to a power of 2
- There are good packages out there (e.g. Fastest Fourier Transform in the West, SPIRAL)
- It does make a BIG difference!

4.10 Why the DFT is useful: A few examples

4.10.1 Overview

- Analysis of musical instruments
- Approximation of periodical phenomena: the case of tides in Venice
- The secret of MP3 compression
- Magnetic Resonance Imaging