RandomByAlpha

Load libraries

```
library("ggplot2") # general plotting
library("ggpubr") # combining plots
```

Specify the experiment parameters

```
ntimes <- 5000 # number of noise replicates to investigate probabilities and expectations
maxdim <- 10000 # maximal dimension of the data set to be investigated
dims <- round(exp(seq(log(2), log(maxdim), length.out=10))) # dimensions to study from log-scale
a <- 1.25 # magnitude of noise: per dimension we sample noise uniformly from [-a, a]
alphas <- c(seq(2, 6, by=1), Inf) # factors controling the growth rate of the ground truth diameters
```

Specify the ground truth sequences according to growth rates

```
x <- numeric(maxdim) # we fix x as the zero vector
y <- numeric(maxdim) # we fix y as the zero vector
zs <- lapply(alphas, function(alpha){ # we use z to control the diameter growth rate
  if(alpha==Inf) rep(1, maxdim) else (1:maxdim)**(-1 / alpha)
})
```

Investigate the various diameter growth rates

```
diams <- data.frame(alpha=integer(), dim=integer(), diam=numeric())
for(idx in 1:length(alphas)){
  diam <- sapply(dims, function(this_dim) norm(zs[[idx]][1:this_dim], type="2"))
  diams[(1:length(dims)) + nrow(diams),] <- data.frame(alpha=alphas[idx], dim=dims, diam=diam)
}

set.seed(42)
diam_noise <- apply(sapply(1:ntimes, function(n){ # expected diameter of noise
  N <- matrix(runif(maxdim * 2, min=-a, max=a), ncol=2)
  sapply(dims, function(this_dim){
    norm(N[1:this_dim, 1] - N[1:this_dim, 2], type="2")
  })
}), 1, mean)
diam_noise <- data.frame(dim=dims, diam=diam_noise)

P1 <- ggplot(data=diams, aes(x=dim, y=diam)) +
  geom_line(aes(color=factor(alpha))) +
  geom_point(aes(color=factor(alpha))) +
  geom_line(data=diam_noise, aes(linetype="noise")) +
  scale_x_log10() +
  ylab("diameter") +
  labs(col="alpha") +
  labs(linetype="") +
  scale_linetype_manual(name="noise", labels=c("expected"), values=2) +
  theme_classic() +
  theme(text=element_text(size=10))
```

Investigate the relative discrimination between furthest and closest points

```
ratios <- data.frame(alpha=integer(), dim=integer(), discrimination=numeric())
for(idx in 1:length(alphas)){
  this_ratios <- apply(sapply(1:ntimes, function(n){ # expected discrimination
    N <- matrix(runif(maxdim * 3, min=-a, max=a), ncol=3)
    sapply(dims, function(this_dim){
      DNxy <- norm(N[1:this_dim, 1] - N[1:this_dim, 2], type="2")
      DNxz <-norm(N[1:this_dim, 1] - zs[[idx]][1:this_dim] - N[1:this_dim, 3], type="2")
      DNyz <-norm(N[1:this_dim, 2] - zs[[idx]][1:this_dim] - N[1:this_dim, 3], type="2")
      DNmax <- max(DNxy, DNxz, DNyz)
      DNmin <- min(DNxy, DNxz, DNyz)
      (DNmax - DNmin) / DNmin
    })
  }), 1, mean)
  ratios[(1:length(dims)) + nrow(ratios),] <- data.frame(alpha=alphas[idx], dim=dims,
                                                         discrimination=this_ratios)
}

P2 <- ggplot(data=ratios, aes(x=dim, y=discrimination)) +
  geom_line(aes(color=factor(alpha))) +
  geom_point(aes(color=factor(alpha))) +
  scale_x_log10() +
  ylab("relative contrast") +
  labs(col="alpha") +
  theme_classic() +
  theme(text=element_text(size=10))
```

We study how high-dimensional noise results in closest/furthest neighbors becoming random

```
P  <- Reduce("+", lapply(1:ntimes, function(n){
  N <- matrix(runif(maxdim * 3, min=-a, max=a), ncol=3)
  sapply(zs, function(z){
    sapply(dims, function(this_dim){
      norm(N[1:this_dim, 1] - N[1:this_dim, 2], type="2") <
        norm(N[1:this_dim, 1] - z[1:this_dim] - N[1:this_dim, 3], type="2")
    })
  })
})) / ntimes

P <- data.frame(alpha=rep(alphas, each=length(dims)),
                dim=rep(dims, length(alphas)),
                P=as.numeric(P))

sigma2 <- a^2 / 3 # variance of uniform distribution over [-a, a]
mu4 <- a^4 / 5 # fourth moment of uniform distribution over [-a, a]
Pat4 <- pnorm(sqrt(2 / (mu4 + 3 * sigma2^2))) # asymptotic probability for alpha = 4

P3 <- ggplot(data=P, aes(x=dim, y=P, color=factor(alpha))) +
  geom_line()+
  geom_point() +
  scale_x_log10() +
  geom_hline(yintercept=Pat4, color="red", linetype=2) +
  annotate("text", x=2, y=Pat4 + 0.025, label="~Phi[4]", color="red", parse=TRUE) +
  ylab("probability") +
```

```
  labs(col="alpha") +
  theme_classic() +
  theme(text=element_text(size=10), plot.title=element_text(hjust=0.5, size=12))
```

Finally, we visually compare the obtained plots

```
ggarrange(P1, P2, P3, ncol=3, common.legend=TRUE)
```