

Korelacja. Regresja

W tej części kursu będziemy zajmować się korelacją oraz regresją. Na początek kilka uwag wstępnych. W statystyce zasadniczo odróżnia się te dwa pojęcia na poziomie conceptualnym. Rozróżnienie to nie jest konsekwentnie przez wszystkich stosowane, warto jednak je znać (dla spokoju ducha!).

Regresja - metoda pozwalająca na zbadanie związku pomiędzy zmiennymi i wykorzystanie tej wiedzy do przewidywania nieznanych wartości jednych wielkości na podstawie znajomości wartości innych. Poszukuje się związku między jedną (lub więcej) zmienną objaśniającą lub niezależną X a zmienną objaśnianą lub zależną Y .

W **regresji** zmienna X (objaśniająca) jest w pełni kontrolowana przez eksperymentatora i pozbawiona elementu losowości.

W **korelacji** obie zmienne są zmiennymi losowymi.

W praktyce, tak jak wspomnieliśmy, różnica ta jest trochę zatarta, ale warto o niej pamiętać.

Wykres punktowy (*scatterplot*)

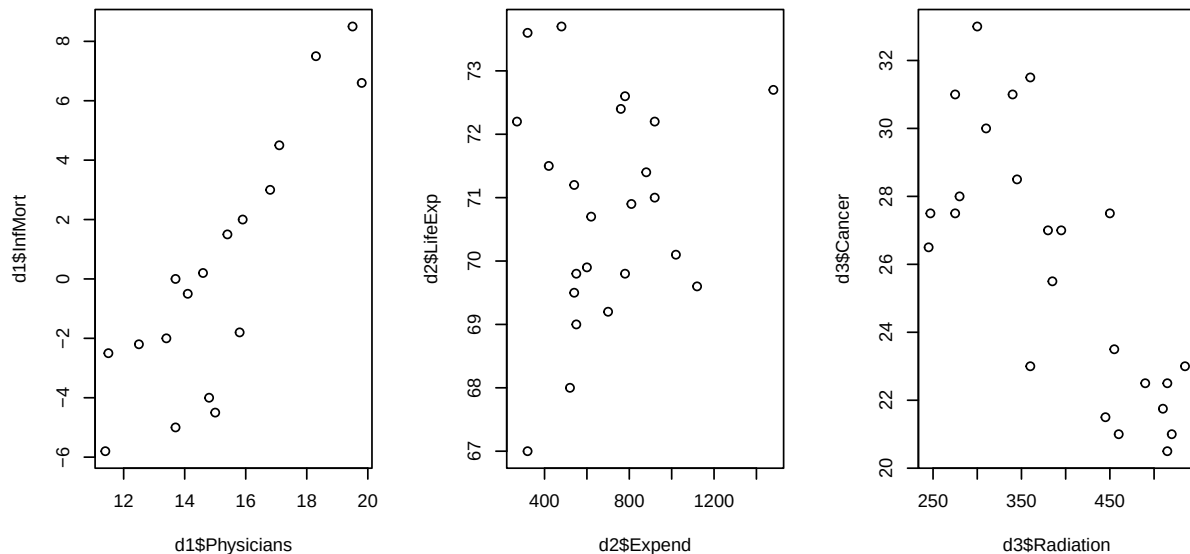
Na początek przyjrzyjmy się sposobowi wizualizacji relacji między dwiema zmiennymi. Nie zdziwi Państwa informacja, że znany i (przynajmniej przez niektórych) lubiany wykres punktowy (zwany także wykresem rozrzutu) świetnie nadaje się do ilustrowania tego typu danych. Przypomnijmy więc sobie, w jaki sposób w R możemy narysować wykres punktowy.

Podstawy

Domyślnie, jeżeli wywołamy funkcję `plot` i prześlemy jej jako argumenty dwa wektory typu `numeric`, to R narysuje wykres punktowy. W tym przypadku reprodukuje wykresy znajdujące się w rozdziale 9 podręcznika Howella. Każdy z punktów na wykresie reprezentuje jeden kraj. Pierwszy wykres przedstawia relację między liczbą lekarzy (zmienna objaśniająca) a śmiertelnością noworodków (zmienna objaśniana), drugi między wydatkami na służbę zdrowia (zmienna objaśniająca) a oczekiwaną długością życia (zmienna objaśniana), trzeci zaś

między promieniowaniem słonecznym (zmienna objaśniająca) a zachorowaniem na nowotwory (zmienna objaśniana).

```
par(mfrow = c(1,3))
d1 <- read.table('Fig9-1a.dat', header = TRUE)
plot(d1$Physicians, d1$InfMort)
d2 <- read.table('Fig9-1b.dat', header = TRUE)
plot(d2$Expend, d2$LifeExp)
d3 <- read.table('Fig9-1c.dat', header = TRUE)
plot(d3$Radiation, d3$Cancer)
```



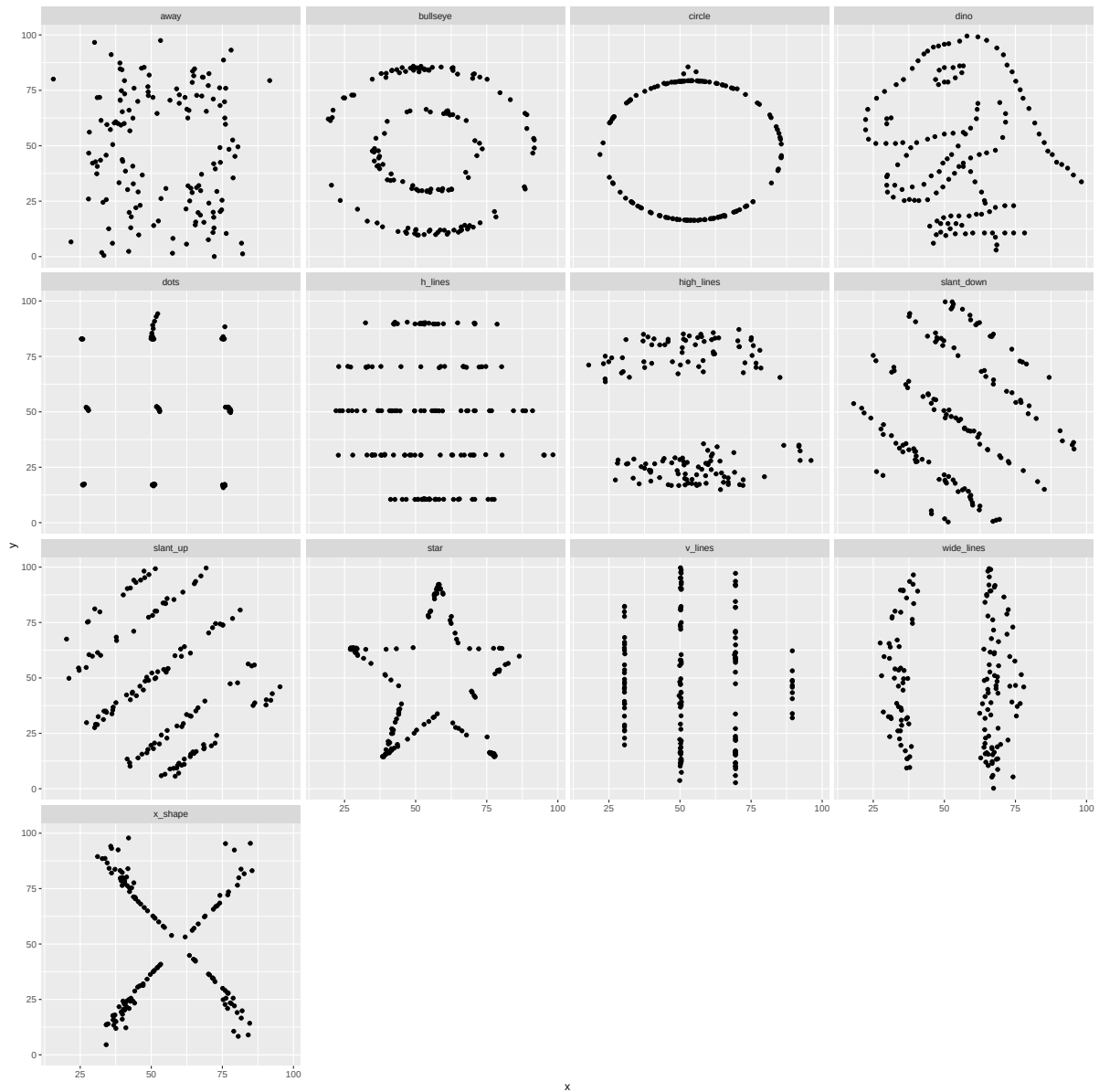
Dlaczego zawsze należy wizualizować dane? Datasaurus

Datasaurus (czyli dinozaur ułożony z punków stworzony przez badaczy z firmy Autocad; więcej informacji na: <https://www.autodeskresearch.com/publications/samestats>) jest dowodem na to, że zawsze powinniśmy wizualizować nasze dane przed przeprowadzeniem analiz statystycznych. Poniżej znajduje się ilustracja 13 rozkładów, które mają takie same: - średnie X - średnie Y - odchylenie standardowe X - odchylenie standardowe Y - współczynnik korelacji między X a Y - współczynnik kowariancji między X a Y

A mimo to są diametralnie różnymi rozkładami! Gdybyśmy patrzyli wyłącznie na gołe statystyki liczbowe, to nigdy byśmy nie zobaczyli, że te zbiory danych są diametralnie różne.

Oto wykresy:

```
options(repr.plot.width=15, repr.plot.height=15)
library(ggplot2)
ggplot(data = datasauRus::datasaurus_dozen) +
  geom_point(aes(x = x, y = y)) +
  facet_wrap(~dataset, ncol=4)
```



A tutaj znajdą Państwo statystyki deskryptywne!

```
library(dplyr)
group_by(datasauRus::datasaurus_dozen, dataset) %>%
  summarise(
    "Korelacja" = cor(x, y),
    "Kowariancja" = cov(x, y),
    "Średnia X" = mean(x),
    "Średnia Y" = mean(y),
    "Odchylenie standardowe X" = sd(x),
    "Odchylenie standardowe Y" = sd(y),
  )
```

A tibble: 13 x 7

	dataset	Korelacja	Kowariancja	Średnia X	Średnia Y	Odchylenie
		standardo~1				
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	
	<dbl>					
1	away	-0.0641	-29.0	54.3	47.8	
	↳ 16.8					
2	bullseye	-0.0686	-31.0	54.3	47.8	
	↳ 16.8					
3	circle	-0.0683	-30.8	54.3	47.8	
	↳ 16.8					
4	dino	-0.0645	-29.1	54.3	47.8	
	↳ 16.8					
5	dots	-0.0603	-27.2	54.3	47.8	
	↳ 16.8					
6	h_lines	-0.0617	-27.9	54.3	47.8	
	↳ 16.8					
7	high_li~	-0.0685	-30.9	54.3	47.8	
	↳ 16.8					
8	slant_d~	-0.0690	-31.2	54.3	47.8	
	↳ 16.8					
9	slant_up	-0.0686	-31.0	54.3	47.8	
	↳ 16.8					
10	star	-0.0630	-28.4	54.3	47.8	
	↳ 16.8					
11	v_lines	-0.0694	-31.4	54.3	47.8	
	↳ 16.8					
12	wide_li~	-0.0666	-30.1	54.3	47.8	
	↳ 16.8					
13	x_shape	-0.0656	-29.6	54.3	47.8	
	↳ 16.8					

```
# i abbreviated name: 1: `Odchylenie standardowe X`  
# i 1 more variable: `Odchylenie standardowe Y` <dbl>
```

Regresja liniowa

Regresję przeprowadzamy w R za pomocą funkcji `lm`. Funkcja ta zwraca obiekt, którego metoda `print` wyświetla wyraz wolny regresji, współczynnik kierunkowy oraz informacje o wywołaniu funkcji. Jeżeli chcemy się dowiedzieć czegoś więcej, musimy obiekt ten przekazać funkcji `summary`.

Wywołanie funkcji `lm`

Wywołanie funkcji `lm` na danych zwraca nam obiekt klasy `lm`. Możemy go przypisać do zmiennej (często spotykaną w R konwencją jest nazwa `fit` dla modelu) albo po prostu wyświetlić.

```
# Wywołanie funkcji `lm` zwraca nam odpowiednio dopasowany model.  
lm(d1$Physicians ~ d1$InfMort)
```

Call:

```
lm(formula = d1$Physicians ~ d1$InfMort)
```

Coefficients:

```
(Intercept)    d1$InfMort  
    15.0346         0.4868
```

Wywołanie funkcji `summary` na obiekcie zwracanym przez `lm`

Tak, jak powiedzieliśmy sobie wcześniej, funkcja `summary` wywołana na obiekcie zwróconym przez funkcję `lm` pozwala nam się dowiedzieć więcej o stworzonym przez nas modelu. W szczególności zaś pozwala nam poznać różne jego parametry oraz przeprowadza szereg testów statystycznych (na istotność poszczególnych współczynników w regresji oraz na dopasowanie całego modelu).

```
summary(lm(d1$Physicians ~ d1$InfMort))
```

Call:

```
lm(formula = d1$Physicians ~ d1$InfMort)
```

Residuals:

Min	1Q	Median	3Q	Max
-2.3176	-0.6836	-0.2450	0.9065	2.1561

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	15.03459	0.29866	50.340	< 2e-16 ***
d1\$InfMort	0.48681	0.07072	6.884	3.68e-06 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.264 on 16 degrees of freedom

Multiple R-squared: 0.7476, Adjusted R-squared: 0.7318

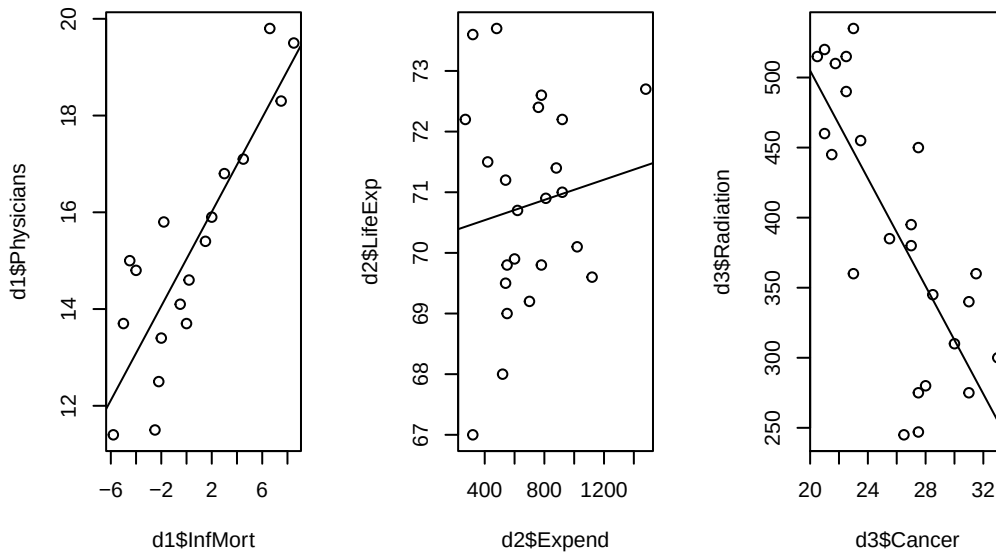
F-statistic: 47.39 on 1 and 16 DF, p-value: 3.675e-06

Dodawanie linii regresji

Moglibyśmy ręcznie „wydobyć” wartości niezbędne do dodania linii regresji (są w końcu drukowane przez R!), ale możemy skorzystać z faktu, że nasz model stworzony za pomocą funkcji `lm` możemy bezpośrednio przekazać jako argument do funkcji `abline`. Funkcja ta zaś automatycznie dorysuje do istniejącego wykresu linię regresji. Należy jednak pamiętać o tym, żeby tworząc wykres punktowy używać składni formuły (z `~`). Na chwile obecną należy pamiętać, że formuły wyglądają mniej więcej tak: $Y \sim X$.

```
# Ustawiamy parametry graficzne R tak, aby narysować trójpanelowy wykres
par(mfrow = c(1,3))
# Tworzymy pierwszy wykres za pomocą składni formuły (tzn. składni z tyldą: Y
  ↪ ~ X)
plot(d1$Physicians ~ d1$InfMort)
# Przypisujemy model do zmiennej `fit`
fit <- lm(d1$Physicians ~ d1$InfMort)
# Model można przekazać jako argument dla funkcji `abline` i ona będzie
  ↪ wiedziała, co i gdzie narysować
abline(fit)

# Pozostałe dwa przypadki analogicznie
plot(d2$LifeExp ~ d2$Expend)
abline(lm(d2$LifeExp ~ d2$Expend))
plot(d3$Radiation ~ d3$Cancer)
abline(lm(d3$Radiation ~ d3$Cancer))
```



Kowariancja i współczynnik korelacji r Pearsona

Kowariancja

Wzór na współczynnik kowariancji między dwiema zmiennymi wygląda następująco:

$$Cov(X, Y) = E[(X - E(X))(Y - E(Y))]$$

- bardzo podobna do wariancji (gdyby za Y podstawić X byłby to wzór na wariancję)
- miara współzmienności dwóch zmiennych
- jeżeli dużym X (w sensie odległości od średniej) towarzyszą duże Y , to kowariancja będzie wysoka i dodatnia, jeśli małe Y to będzie ujemna, jeśli raz takie, a raz takie (brak korelacji) to będą się znosić a kowariancja będzie wynosić około 0

Korelacja

Problem z kowariancją jest taki, że jej wielkość zależy od tego, jakie jednostki mają nasze zmienne oraz w szczególności od tego, jakie mają odchylenie standardowe. Może to utrudnić porównania i interpretacje takiej wartości. Obliczenie współczynnika korelacji jest sposobem na poradzenie sobie z tym problemem. Można o nim myśleć jako o “wystandaryzowanym” współczynnikiem kowariancji. Wzór na korelację w populacji (ρ) wygląda tak:

$$\rho = \frac{Cov(X, Y)}{\sqrt{Var(X)Var(Y)}}$$

- przyjmuje wartości z przedziału $[-1, 1]$
- jest miarą liniowej zależności między zmiennymi losowymi X i Y
- jeżeli dysponujemy próbą, to możemy wyliczyć współczynnik korelacji Pearsona (r):

$$r = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2 (Y_i - \bar{Y})^2}}$$

albo prościej:

$$r = \frac{cov_{XY}}{s_x s_y}$$

- na podstawie jego wartości możemy ocenić siłę związku prostoliniowego między cechami X a Y :

Spróbujmy sprawdzić teraz, czy R oblicza oba współczynniki zgodnie ze wzorami! Zaczniemy od kowariancji:

```
print('Kowariancja')
```

```
[1] "Kowariancja"
```

```
sum((d1$Physicians - mean(d1$Physicians)) * (d1$InfMort - mean(d1$InfMort))) /  
  (nrow(d1)-1) # wzór kowariancji wpisany ręcznie
```

```
[1] 9.14598
```

```
cov(d1$Physicians, d1$InfMort) # funkcja wbudowana w R
```

```
[1] 9.14598
```

A teraz sprawdzimy to samo dla korelacji:

```
print('Korelacja')
```

```
[1] "Korelacja"
```

```
cor(d1$Physicians, d1$InfMort) # funkcja wbudowana w R
```

```
[1] 0.8646336
```

```
(sum((d1$Physicians - mean(d1$Physicians))*(d1$InfMort - mean(d1$InfMort)))/
  (nrow(d1)-1))/(sd(d1$Physicians)*sd(d1$InfMort))
```

```
[1] 0.8646336
```

Interpretacja wartości r

Interpretując współczynnik korelacji Pearsona musimy pamiętać, że właściwa interpretacja będzie zależec od tego, w jakiej dziedzinie się aktualnie znajdujemy. Dla fizyka inny współczynnik korelacji będzie uznany za “duży” niż dla psychologa społecznego. Poniżej znajdują się jednak pewne “zasady kciuka” dotyczące interpretacji współczynnika r w naukach psychologicznych:

- $r = 0$ - współzależność nie występuje, brak korelacji, zmienne są nieskorelowane
- $0 < |r| < 0.3$ - słaby stopień współzależności
- $0.3 \leq |r| < 0.5$ - średni stopień współzależności
- $0.5 \leq |r| < 0.7$ - znaczny stopień współzależności
- $0.7 \leq |r| < 0.9$ - wysoki stopień współzależności
- $|r| \geq 0.9$ - bardzo wysoki stopień współzależności
- $|r| = 1$ - współzależność całkowita (ściskość)

Macierze korelacyjne - wizualizacje

Czasami mamy do czynienia z więcej niż dwiema zmiennymi. W takim przypadkach zdarza się, że chcielibyśmy zbadać korelacje między wszystkimi kombinacjami dwóch zmiennych jednocześnie. W tym celu możemy stworzyć macierz korelacyjną (tak samo tworzy się macierz kowariancji - przyda się Państwu na Statystyce II przy PCA). Tutaj zrobimy to (może nieco nudno) na losowych danych pochodzących z rozkładu normalnego.

```
# Wylosujemy 350 obserwacji z rozkładu normalnego a następnie podzielimy je
  ↪ na 10 kolumn
# W ten sposób zasymulujemy sytuację zbioru danych z 10 zmiennymi
m <- matrix(rnorm(350), 35, 10)
# Przekształcamy macierz w ramkę danych
m <- as.data.frame(m)
# Tworzymy macierz korelacyjną, dla kowariancji będzie to `cov`
cor(m)
```

	V1	V2	V3	V4	V5	V6
V1	1.00000000	-0.30722819	0.17620959	-0.06282882	-0.01272841	0.27934814
V2	-0.30722819	1.00000000	-0.03765733	0.01843325	0.09696172	-0.01376366

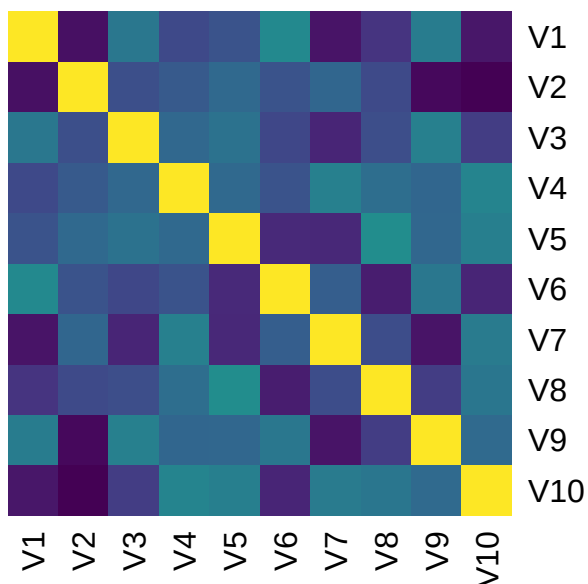
V3	0.17620959	-0.03765733	1.00000000	0.09256867	0.14855978	-0.07264537
V4	-0.06282882	0.01843325	0.09256867	1.00000000	0.09790751	-0.01682502
V5	-0.01272841	0.09696172	0.14855978	0.09790751	1.00000000	-0.20825088
V6	0.27934814	-0.01376366	-0.07264537	-0.01682502	-0.20825088	1.00000000
V7	-0.29043662	0.08352219	-0.22186139	0.22542149	-0.21200592	0.04053582
V8	-0.15616629	-0.05857142	-0.03973281	0.12350003	0.29865071	-0.25203337
V9	0.20511762	-0.33263324	0.22382153	0.07963568	0.08871653	0.17476783
V10	-0.28076740	-0.36320330	-0.11987723	0.25009461	0.21907761	-0.21980305
	V7	V8	V9	V10		
V1	-0.29043662	-0.15616629	0.20511762	-0.2807674		
V2	0.08352219	-0.05857142	-0.33263324	-0.3632033		
V3	-0.22186139	-0.03973281	0.22382153	-0.1198772		
V4	0.22542149	0.12350003	0.07963568	0.2500946		
V5	-0.21200592	0.29865071	0.08871653	0.2190776		
V6	0.04053582	-0.25203337	0.17476783	-0.2198031		
V7	1.00000000	-0.04848030	-0.28968233	0.1961338		
V8	-0.04848030	1.00000000	-0.12105225	0.1706965		
V9	-0.28968233	-0.12105225	1.00000000	0.1041229		
V10	0.19613380	0.17069655	0.10412288	1.00000000		

Jednym ze sposobów wizualizacji takich danych jest “mapa cieplna”, w której intensywność koloru (albo jakiś jego inny parametr) oznacza siłę korelacji (tutaj - im jaśniejszy tym wyższy współczynnik korelacji, im ciemniejszy tym niższy).

```
library('viridis')
```

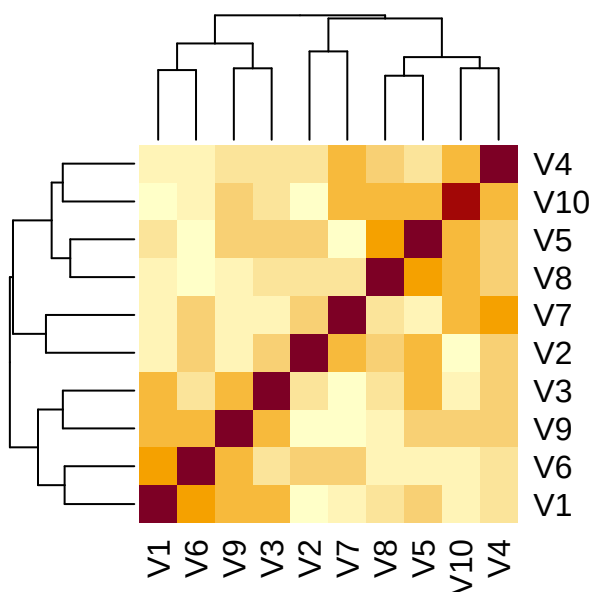
Loading required package: viridisLite

```
heatmap(cor(m), symm = TRUE, Rowv = NA, col=viridis(256))
```



Wersja podstawowa produkowana przez funkcję `heatmap` wygląda w tym przypadku tak.

```
heatmap(cor(m)) # wersja z klastrami, ale one nam są niepotrzebne
```



Pewną ciekawą odmianą mapy cieplnej jest wykres z elipsami, na którym widać siłę (jak bardzo spłaszczona jest elipsa) oraz kierunek (w którą stronę „kopnięta” jest elipsa) korelacji między zmiennymi.

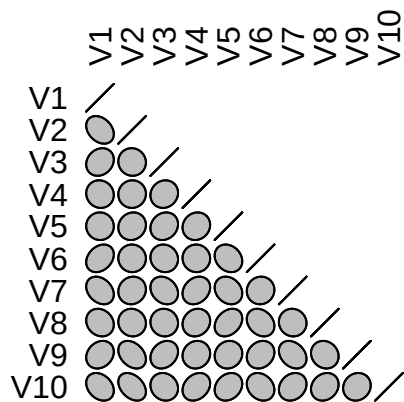
```
library(ellipse)
```

Attaching package: 'ellipse'

The following object is masked from 'package:graphics':

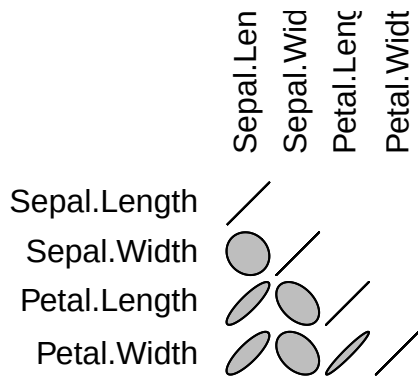
pairs

```
# `type` mówi o tym, czy chcemy umieszczać na wykresie całą macierz czy tylko  
  ↳ jej część  
# `diag` mówi, czy rysować przekątną (na przekątnej r = 1, więc nie jest za  
  ↳ bardzo ciekawa...)  
plotcorr(cor(m), type = 'lower',  
          diag = TRUE)
```



W przypadku wylosowanych obserwacji może być trudno zrozumieć, jak działać mają rysowane przez `plotcorr` elipsy. Być może lepiej widać to na znanym Państwu już zbiorze dotyczącym irysów (`iris`).

```
plotcorr(cor(iris[,1:4]), type = 'lower', diag = TRUE)
```

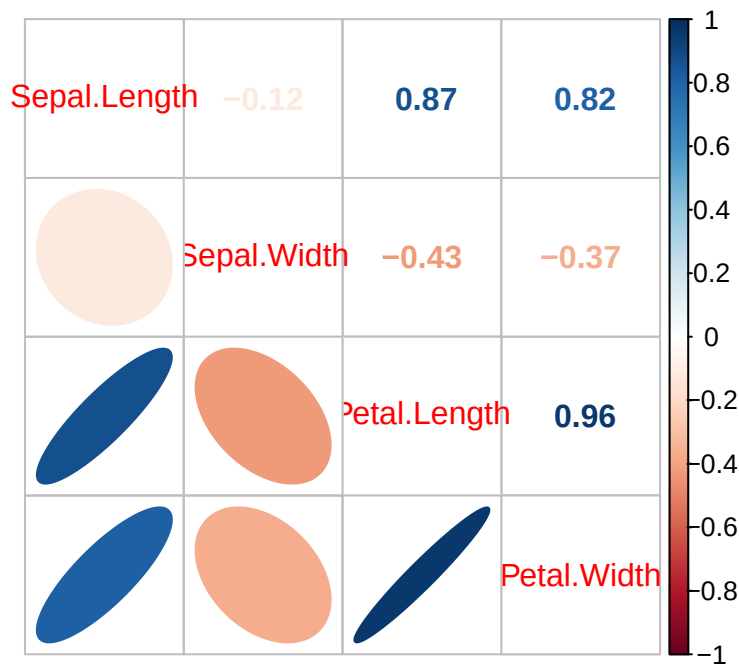


Ostatnim sposobem generowania ładnie wyglądających wizualizacji korelacji między wieloma zmiennymi, który omówimy, są funkcje `corrplot` i `corrplot.mixed` z pakietu (!) `corrplot`. Funkcje ta ma bardzo wiele możliwości dostosowania wyglądu wykresu - zachęcam do samodzielnej eksploracji!

```
library(corrplot)
```

```
corrplot 0.95 loaded
```

```
corrplot.mixed(cor(iris[,1:4]),
               lower = "ellipse",
               upper = "number")
```



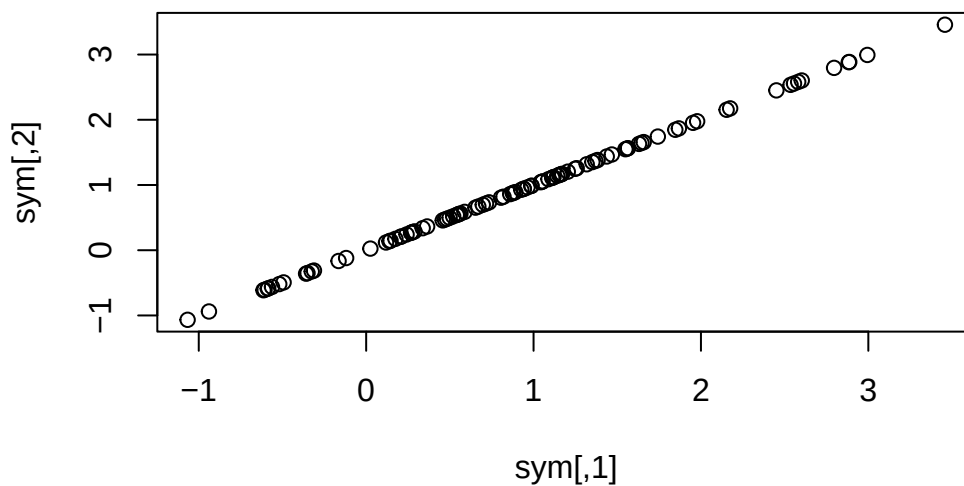
Gra - zgadnij korelację

Jeśli będziemy w przyszłości pracować z danymi, to warto nabyć pewną intuicję dotyczącą tego, jaki wizualny “rozrzut” punktów odpowiada jakiemu współczynnikowi korelacji. Dobrym ćwiczeniem jest dostępna tutaj gra, w której możemy przetestować i “poprawić” swoją intuicję.

<http://guessthecorrelation.com/>

Możemy podobną grę przeprowadzić za pomocą R. Dobrym punktem wyjścia jest funkcja `rmvnorm` z pakietu `mvtnorm`, która pozwala nam losować skorelowane ze sobą zmienne.

```
library(mvtnorm)
sym <- rmvnorm(100, mean = c(1,1), sigma = matrix(c(1,1,1,1),2,2))
plot(sym)
```



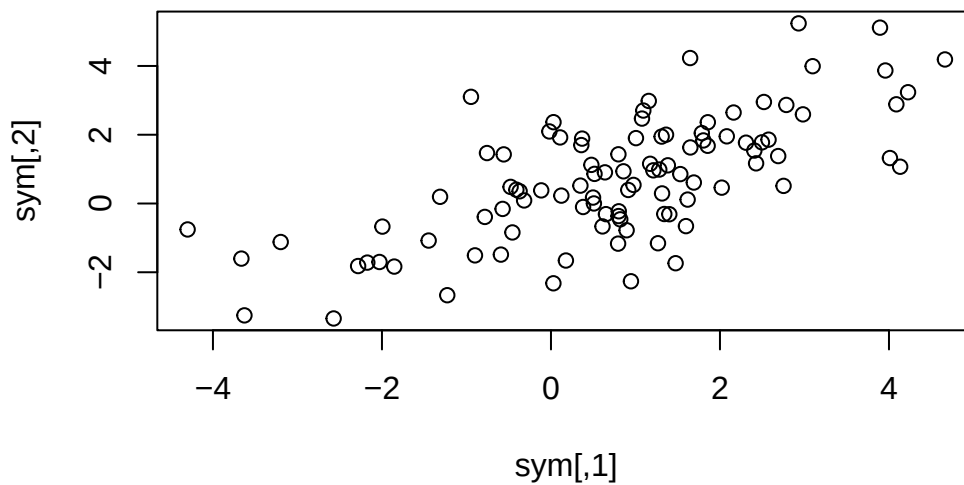
```
cov(sym)
```

```
      [,1]      [,2]
[1,] 0.8905963 0.8905963
[2,] 0.8905963 0.8905963
```

```
cor(sym)
```

```
      [,1] [,2]
[1,]    1    1
[2,]    1    1
```

```
sym <- rmvnorm(100, mean = c(1,1), sigma = matrix(c(3,2,2,3),2,2))
plot(sym)
```



```
cov(sym)
```

```
      [,1]      [,2]
[1,] 3.174526 2.160883
[2,] 2.160883 3.152556
```

```
cor(sym)
```

```
      [,1]      [,2]
[1,] 1.0000000 0.6830624
[2,] 0.6830624 1.0000000
```

Linia regresji

Na koniec powróćmy do regresji liniowej. Jak powszechnie wiadomo, równanie prostej dopasowywanej do danych w regresji liniowej ma postać:

$$\hat{Y} = bX + a$$

gdzie:

- $\hat{Y}$ - przewidywana wartość Y

- b - współczynnik regresji (*slope* - współczynnik kierunkowy)
- a - wyraz wolny (*intercept*)
- X - wartość zmiennej predyktora

Zadanie

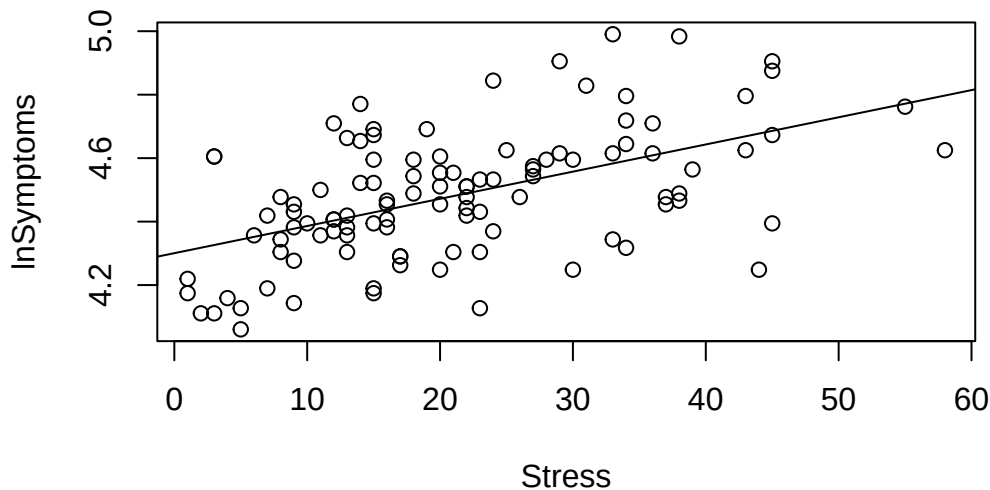
Wagner, Compas i Howell (1988) badali związek między stresem a zdrowiem psychicznym wśród studentów pierwszego roku koledżu. Używając opracowanego przez siebie narzędzia do mierzenia częstotliwości, odczuwanej wagi i pożądaności niedawnych wydarzeń życiowych, stworzyli miarę negatywnych zdarzeń w życiu. Miara ta służyła do pomiaru środowiskowego i społecznego stresu odczuwanego przez badanych. Oprócz tego poprosili swoich badanych (studentów), aby wypełnili *Hopkins Symptom Checklist*, która to służy do pomiaru występowania lub brak występowania 57 psychologicznych symptomów zaburzeń zdrowia psychicznego.

Rozpocniemy od wczytania oraz wizualizacji danych. Za pomocą funkcji `abline` dodamy do wykresu punktowego linię regresji.

```
df <- read.table('Tab9-2.dat', header = TRUE)
head(df)
```

	ID	Stress	Symptoms	lnSymptoms
1	1	30	99	4.595120
2	2	27	94	4.543295
3	3	9	80	4.382027
4	4	20	70	4.248495
5	5	3	100	4.605170
6	6	15	109	4.691348

```
plot(lnSymptoms ~ Stress, data = df)
abline(lm(lnSymptoms ~ Stress, data = df))
```



Następnie przyjrzymy się bliżej stworzonemu przez nas modelowi.

```
summary(lm(lnSymptoms ~ Stress, data = df))
```

Call:

```
lm(formula = lnSymptoms ~ Stress, data = df)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.42889	-0.13568	0.00478	0.09672	0.40726

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	4.300537	0.033088	129.974	< 2e-16 ***
Stress	0.008565	0.001342	6.382	4.83e-09 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1726 on 105 degrees of freedom

Multiple R-squared: 0.2795, Adjusted R-squared: 0.2726

F-statistic: 40.73 on 1 and 105 DF, p-value: 4.827e-09

(Dygresja: w pakiecie `rms` znajduje się funkcja do przeprowadzania regresji liniowej zwracająca nieco inne informacje niż ta domyślnie wbudowana w R. Można ją przetestować po wczytaniu pakietu `rms` (`library(rms)`) wywołując polecenie `ols(lnSymptoms ~ Stress, data = df)`)

Wyraz wolny (*intercept*)

- **definicja:** wartość $\hat{Y}$ kiedy X przyjmuje wartość 0
- jego interpretacja zależy od tego, czy $X = 0$ ma jakąkolwiek sensowną interpretację
- zwykle nie ma sensownej interpretacji i ma tylko tę matematyczną (ogromna i niepraktyczna ekstrapolacja z naszych danych - pomyśl o wadze 0kg!)
- zawsze można **wycentrować** nasz predyktor wokół średniej - wtedy uzyskujemy sensowną interpretację - $\hat{Y}$ dla wartości oczekiwanej X
- **wycentrowanie** nie ma żadnych skutków dla współczynnika kierunkowego oraz współczynnika korelacji

Współczynnik kierunkowy (*slope*)

- zmiana $\hat{Y}$ związana ze zmianą X o jedną jednostkę
- definicja ta mówi nam że ma on sensowną interpretację (np. jeżeli mówimy o regresji dochodu z liczby lat edukacji, to współczynnik kierunkowy powie nam jaka różnica w dochodzie jest związana z każdym dodatkowym rokiem edukacji)

Standaryzowany współczynnik regresji

- to co współczynnik kierunkowy, tylko że obie zmienne są wystandaryzowane
- możemy policzyć za pomocą `lm.beta` z pakietu `QuantPsyc`, ale równie dobrze możemy zrobić to sami mnożąc przez iloraz wariancji

Korelacja a standaryzowany współczynnik regresji

- jeżeli mamy jeden predyktor (tak jak w tych przykładach, których się zajmujemy) to jest to ta sama wartość

Testowanie hipotez dla współczynnika korelacji r Pearsona

Najczęściej będziemy sprawdzać czy zmienne X i Y są skorelowane. W takim wypadku nasze hipotezy będą przedstawiać się w następujący sposób. Zaczniemy od hipotezy alternatywnej.

$$H_A : \rho \neq 0$$

To znaczy, że hipoteza alternatywna głosi, że korelacja w populacji (ρ) jest różna od zera. Hipoteza zerowa (tę będziemy obalać!) głosi więc, że korelacja w populacji wynosi 0:

$$H_0 : \rho = 0$$

Statystyka testowa wygląda w takim przypadku tak:

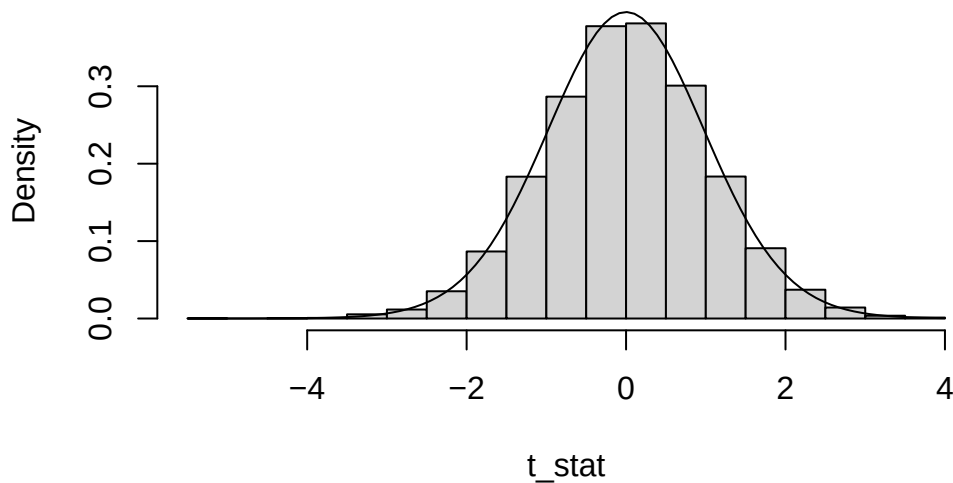
$$T = \frac{r}{\sqrt{1-r^2}} \sqrt{n-2}$$

która ma przy prawdziwości H_0 rozkład t o $n-2$ stopniach swobody.

Możemy to z łatwością sprawdzić za pomocą prostego eksperymentu symulacyjnego.

```
n <- 35
t_stat = replicate(10000, {
  r = cor(rnorm(n), rnorm(n))
  (r/sqrt(1-r^2))*sqrt(n-2)})
hist(t_stat, freq = FALSE)
curve(dt(x,n-2), add = TRUE)
```

Histogram of t_stat



Zróbmy proste ćwiczenie. Na początek stwórzmy macierz z losowymi liczbami pochodzącymi z rozkładu normalnego (10 kolumn po 35 liczb) a następnie obliczmy statystykę testową t dla korelacji między nimi.

```
# Generujemy zbiór danych
m <- matrix(rnorm(350), 35, 10)

# Tworzymy macierz korelacji
c_m <- cor(m)

# Tworzymy macierz statystyk t
t_m <- (c_m/sqrt(1-c_m^2)) * (sqrt(35-2))
t_m
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	Inf	0.4800288	-0.3451965	-1.3496842	-2.39003013	0.7258763
[2,]	0.4800288	Inf	-0.1103936	-0.4594719	-0.84329020	-1.2575014
[3,]	-0.3451965	-0.1103936	Inf	-0.8606814	2.16153313	-0.1291704
[4,]	-1.3496842	-0.4594719	-0.8606814	Inf	0.73569217	-1.1378901
[5,]	-2.3900301	-0.8432902	2.1615331	0.7356922	Inf	0.4030385
[6,]	0.7258763	-1.2575014	-0.1291704	-1.1378901	0.40303851	Inf
[7,]	-0.7158483	-0.6252840	1.1884167	0.3987274	1.57740653	0.2121234
[8,]	-0.8544879	1.4092020	-0.0569788	0.3768702	0.57189838	-0.7226753
[9,]	-0.9529914	1.0687030	-0.5507346	0.8986912	0.09198021	-0.4861767
[10,]	-0.6204653	-0.5465096	0.7599606	0.6351513	0.04681898	-1.0682165
	[,7]	[,8]	[,9]	[,10]		
[1,]	-0.7158483	-0.8544879	-0.95299137	-0.62046527		
[2,]	-0.6252840	1.4092020	1.06870296	-0.54650964		
[3,]	1.1884167	-0.0569788	-0.55073457	0.75996056		
[4,]	0.3987274	0.3768702	0.89869122	0.63515134		
[5,]	1.5774065	0.5718984	0.09198021	0.04681898		
[6,]	0.2121234	-0.7226753	-0.48617671	-1.06821646		
[7,]	Inf	-1.4772019	-0.16814007	0.27471857		
[8,]	-1.4772019	Inf	0.15082111	0.30639429		
[9,]	-0.1681401	0.1508211	Inf	-0.62967455		
[10,]	0.2747186	0.3063943	-0.62967455	Inf		

Mając taką macierz jesteśmy w stanie dla każdej komórki (= dla każdej kombinacji dwóch zmiennych) ocenić, czy odrzucamy hipotezę zerową czy nie. Wystarczy posłużyć się odpowiednim rozkładem t o $n - 2$ stopniach swobody. Za pomocą funkcji poniżej możemy łatwo stwierdzić, które hipotezy zerowe odrzucamy.

```
t_m <= qt(0.025, 33) | t_m >= qt(0.975, 33)
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]
[1,]	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE

```
[2,] FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[3,] FALSE FALSE TRUE FALSE TRUE FALSE FALSE FALSE FALSE FALSE
[4,] FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
[5,] TRUE FALSE TRUE FALSE TRUE FALSE FALSE FALSE FALSE FALSE
[6,] FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE
[7,] FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE
[8,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE
[9,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE
[10,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE
```

A nawet policzyć, ile razy odrzuciliśmy hipotezę zerową!

```
print('Liczba statystycznie istotnych korelacji między wygenerowanymi
↪ próbami')
```

```
[1] "Liczba statystycznie istotnych korelacji między wygenerowanymi próbami"
```

```
sum(t_m <= qt(0.025, 33) | t_m >= qt(0.975, 33)) - ncol(t_m)
```

```
[1] 4
```

(Uwaga! Test statystyczny dla współczynnika korelacji możemy również wykonać za pomocą wbudowanej w R funkcji `cor.test`. Funkcja ta jednak przyjmuje tylko dwa wektory!

```
cor.test(m[, 1], m[, 2])
```

Pearson's product-moment correlation

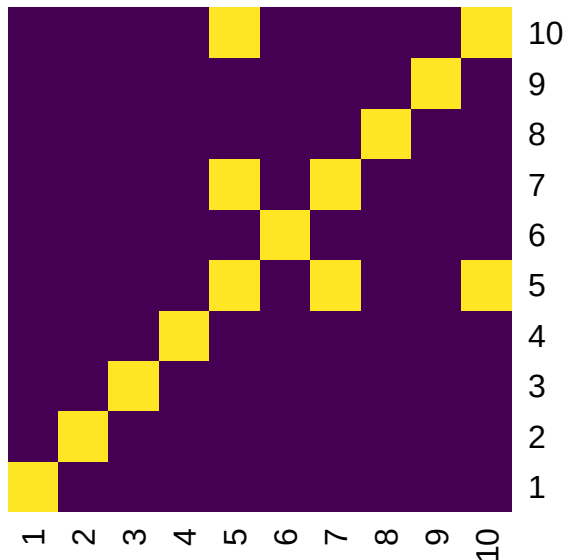
```
data: m[, 1] and m[, 2]
t = 0.48003, df = 33, p-value = 0.6344
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.2571093 0.4052722
sample estimates:
      cor
0.08327206
)
```

Poniżej widzą Państwo wykres ilustrujący nasz eksperyment z losowaniem. Pod wykresem wypisujemy liczbę istotnych statystycznie współczynników korelacji poza przekątną (czyli pomijamy korelacje zmiennych z samymi sobą). Wokół jakiej liczby oscylowałaby ta wartość, gdybyśmy powtarzali nasz eksperyment? Dlaczego?

```

m <- matrix(rnorm(350), 35, 10)
c_m <- cor(m)
t_m <- (c_m/sqrt(1-c_m^2)) * (sqrt(35-2))
reject <- (t_m <= qt(0.025, 33) | t_m >= qt(0.975, 33))
heatmap(matrix(as.numeric(reject), 10,10), Rowv = NA, Colv = NA, col =
  ↪ viridis(2))

```



```

print(sum(reject) - 10) # 10 leży na przekątnej

```

```
[1] 4
```