Проект. Исследование рынка инвестиций

Автор: Котов Алексей (когорта da_132)

Почта: alexkotov1001@yandex.ru

Введение

Финансовая компания, работающая с венчурными инвестициями, хочет понять закономерности финансирования стартапов и оценить перспективы выхода на рынок с покупкой и развитием компаний. Для этого необходимо провести исследование на исторических данных.

Предстоить работать с информацией о компаниях, объёмах и типах привлечённых инвестиций, а также с дополнительной статистикой по возвратам средств.

Цель:

Подготовить датасет к работе, исследовать динамику и структуру финансирования стартапов и ответить на вопросы, важные для оценки инвестиционных стратегий.

Задачи:

- Проанализировать динамику предоставления финансирования по годам.
- Проанализировать динамику размера общего финансирования по массовым сегментам рынка для растущих в 2014 году сегментов.
- Проанализировать годовую динамику доли возвращённых средств по типам финансирования.
- Подготовить итоговый вывод и рекомендации заказчику.

Шаг 1. Знакомство с данными: загрузка и предобработка

Описание датасетов

Датасет получен из базы данных стартапов.

Название основного датасета — `cb_investments.zip`. Внутри архива один файл — `cb_investments.csv`.

Описание данных: <...>

Название дополнительного датасета — `cb_returns.csv`. Он содержит суммы возвратов по типам финансирования в миллионах долларов по годам.

Описание данных: <...>

Вывод общей информации

Загружаем необходимые для работы библиотеки.

In [1]:
```
# Импортируем библиотеки
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Установим настройку, чтобы показывались все столбцы датафрейма
pd.set_option("display.max_columns", None)
```

Загружаем все данные по проекту.

In [ ]:
```
# датафрейм основного датасета с затратами на финансирование
df = pd.read_csv(".../cb_investments.zip", sep=';', low_memory=False)

# датафрейм вспомогательного датасета с суммами возвратов от финансирования
df_returns = pd.read_csv(".../cb_returns.csv")
```

Выводим информацию, которая необходима для принятия решений о предобработке.

In [4]:
```
df.head()
```

Out[4]:

	name	homepage_url	category_list	market	funding_total_usd	status	country_code	state_code	region	city	funding_rounds	participants	foun
0	Harvard University	http://harvard.edu	|Education|	Education	9,00,00,000	operating	USA	MA	Boston	Cambridge	1.0	NaN	16
1	University of New Brunswick	http://www.unb.ca	NaN	NaN	20,00,000	operating	NaN	NaN	NaN	NaN	1.0	NaN	178
2	DuPont	http://www.dupont.com	|Business Services|Agriculture|Automotive|Inve...	Business Services	90,00,000	operating	USA	DE	Wilmington, Delaware	Wilmington	1.0	1.0	180
3	University of Michigan	http://www.umich.edu/	|Education|	Education	77,00,000	operating	USA	MI	Detroit	Ann Arbor	3.0	0.0	181
4	Case Western Reserve University	http://www.case.edu	|Education|	Education	5,40,000	operating	USA	OH	Cleveland	Cleveland	1.0	NaN	182

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54294 entries, 0 to 54293
Data columns (total 40 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   name                  49437 non-null  object 
 1   homepage_url          45989 non-null  object 
 2   category_list         45477 non-null  object 
 3    market               45477 non-null  object 
 4    funding_total_usd    49438 non-null  object 
 5   status                48124 non-null  object 
 6   country_code          44165 non-null  object 
 7   state_code            30161 non-null  object 
 8   region                44165 non-null  object 
 9   city                  43322 non-null  object 
 10  funding_rounds        49438 non-null  float64
 11  participants          30473 non-null  float64
 12  founded_at            38554 non-null  object 
 13  founded_month         38482 non-null  object 
 14  founded_quarter       38482 non-null  object 
 15  founded_year          38554 non-null  float64
 16  first_funding_at      49438 non-null  object 
 17  mid_funding_at        30288 non-null  object 
 18  last_funding_at       49438 non-null  object 
 19  seed                  49438 non-null  float64
 20  venture               49438 non-null  float64
 21  equity_crowdfunding   49438 non-null  float64
 22  undisclosed           49438 non-null  float64
 23  convertible_note      49438 non-null  float64
 24  debt_financing        49438 non-null  float64
 25  angel                 49438 non-null  float64
 26  grant                 49438 non-null  float64
 27  private_equity        49438 non-null  float64
 28  post_ipo_equity       49438 non-null  float64
 29  post_ipo_debt         49438 non-null  float64
 30  secondary_market      49438 non-null  float64
 31  product_crowdfunding  49438 non-null  float64
 32  round_A               49438 non-null  float64
 33  round_B               49438 non-null  float64
 34  round_C               49438 non-null  float64
 35  round_D               49438 non-null  float64
 36  round_E               49438 non-null  float64
 37  round_F               49438 non-null  float64
 38  round_G               49438 non-null  float64
 39  round_H               49438 non-null  float64
dtypes: float64(24), object(16)
memory usage: 16.6+ MB
```

In [6]:
```
# Процент пропусков каждого столбца
df.isna().mean().sort_values(ascending=False) * 100
```

Out[6]:
```
state_code              44.448742
mid_funding_at          44.214830
participants            43.874093
founded_month           29.122923
founded_quarter         29.122923
founded_at              28.990312
founded_year            28.990312
city                    20.208494
country_code            18.655837
region                  18.655837
category_list           16.239363
 market                 16.239363
homepage_url            15.296350
status                  11.364055
name                     8.945740
private_equity           8.943898
round_E                  8.943898
round_F                  8.943898
round_D                  8.943898
round_C                  8.943898
round_G                  8.943898
round_B                  8.943898
round_A                  8.943898
product_crowdfunding     8.943898
secondary_market         8.943898
post_ipo_debt            8.943898
post_ipo_equity          8.943898
venture                  8.943898
grant                    8.943898
angel                    8.943898
debt_financing           8.943898
convertible_note         8.943898
undisclosed              8.943898
equity_crowdfunding      8.943898
seed                     8.943898
last_funding_at          8.943898
first_funding_at         8.943898
funding_rounds           8.943898
 funding_total_usd       8.943898
round_H                  8.943898
dtype: float64
```

В датафрейме `df` 40 столбцов и 54294 строк.

В названии столбцов `market` и `funding_total_usd` есть лишние пробелы.

Столбец `funding_total_usd` имеет тип `object` из-за некорректной записи чисел.

Столбцы `founded_at` , `founded_month` , `founded_quarter` , `first_funding_at` , `mid_funding_at` и `last_funding_at` типа `object` , но содержат информацию о дате.

В числовых столбцах можно попробовать оптимизировать размерность.

Больше всего пропусков в столбцах `state_code` (44%), `mid_funding_at` (44%) и `participants` (43%). Большинство пропусков в регистрационной информации о компаниях, которая для данного анализа не важна.

In [7]:
```
df_returns.head()
```

Out[7]:

	year	seed	venture	equity_crowdfunding	undisclosed	convertible_note	debt_financing	angel	grant	private_equity	post_ipo_equity	post_ipo_debt	secondary_market	product_crowdfunding
0	2000	16.70	55.40	0.0	78.21	0.00	8.66	6.43	0.0	0.00	0.94	0.0	0.20	0.0
1	2001	2.88	23.49	0.0	21.50	0.01	4.49	1.18	0.0	0.00	0.46	0.0	0.46	0.0
2	2002	6.59	209.42	0.0	25.77	0.02	3.42	3.41	0.0	1.51	0.34	0.0	0.06	0.0
3	2003	7.74	233.86	0.0	9.40	0.01	1.09	3.41	0.0	1.62	2.11	0.0	0.08	0.0
4	2004	9.93	555.90	0.0	33.19	0.01	13.55	9.18	0.0	2.19	3.38	0.0	0.55	0.0

In [8]:
```
df_returns.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   year                  15 non-null     int64
 1   seed                  15 non-null     float64
 2   venture               15 non-null     float64
 3   equity_crowdfunding   15 non-null     float64
 4   undisclosed           15 non-null     float64
 5   convertible_note      15 non-null     float64
 6   debt_financing        15 non-null     float64
 7   angel                 15 non-null     float64
 8   grant                 15 non-null     float64
 9   private_equity        15 non-null     float64
 10  post_ipo_equity       15 non-null     float64
 11  post_ipo_debt         15 non-null     float64
 12  secondary_market      15 non-null     float64
 13  product_crowdfunding  15 non-null     float64
dtypes: float64(13), int64(1)
memory usage: 1.8 KB
```

В датафрейме `df_returns` с суммами возвратов от финансирования 14 столбцов и 15 строк.

Все столбцы имеют числовой тип соотвествтующий содержимому, но можно попробовать понизить размерность данных. Пропусков в данных датафрейма нет.

Данные соответствуют описанию.

Предобработка данных

В названиях столбцов в датафрейма `df` есть столбцы с лишними пробелами в названии, приведем их к единому аккуратному стилю.

In [9]:
```
# В датафрейме уберем лишние пробелы в названии столбцов
df.columns = df.columns.str.strip()
# В датафрейме все буквы в нижний регистр
df.columns = df.columns.str.lower()
# Выведем названия столбцов
df.columns
```

Out[9]:
```
Index(['name', 'homepage_url', 'category_list', 'market', 'funding_total_usd',
       'status', 'country_code', 'state_code', 'region', 'city',
       'funding_rounds', 'participants', 'founded_at', 'founded_month',
       'founded_quarter', 'founded_year', 'first_funding_at', 'mid_funding_at',
       'last_funding_at', 'seed', 'venture', 'equity_crowdfunding',
       'undisclosed', 'convertible_note', 'debt_financing', 'angel', 'grant',
       'private_equity', 'post_ipo_equity', 'post_ipo_debt',
       'secondary_market', 'product_crowdfunding', 'round_a', 'round_b',
       'round_c', 'round_d', 'round_e', 'round_f', 'round_g', 'round_h'],
      dtype='object')
```

Уберем в столбце `funding_total_usd` выделение разрядов и приведите его к числовому типу, с заменой нечисловых значений на NaN.

In [10]:
```
# Удаляем запятые
df['funding_total_usd'] = df['funding_total_usd'].str.replace(',', '')
# Приводим к числовому типу с заменой нечисловых значений на NaN
df['funding_total_usd'] = pd.to_numeric(df['funding_total_usd'], errors='coerce')
# Выводим первые строки для проверки
df['funding_total_usd'].head()
```

Out[10]:
```
0    90000000.0
1     2000000.0
2     9000000.0
3     7700000.0
4      540000.0
Name: funding_total_usd, dtype: float64
```

Обработаем типы данных в столбцах `founded_at`, `founded_month`, `founded_quarter`, `first_funding_at`, `mid_funding_at` и `last_funding_at`, которые хранят значения даты и времени.

In [11]:
```
# Список столбцов с датами
date_columns = ['founded_at', 'founded_month', 'founded_quarter', 'first_funding_at', 'mid_funding_at', 'last_funding_at']
# Выведем типы столбцов
print(df[date_columns].dtypes)
# Выведем первые столбцы с непустыми датами
df[date_columns].dropna().head()
```

```
founded_at          object
founded_month       object
founded_quarter     object
first_funding_at    object
mid_funding_at      object
last_funding_at     object
dtype: object
```

Out[11]:

	founded_at	founded_month	founded_quarter	first_funding_at	mid_funding_at	last_funding_at
74	1903-01-01	1903-01	1903-Q1	2008-08-07	2008-08-07	2008-08-07
77	1906-01-01	1906-01	1906-Q1	2012-03-15	2012-03-15	2012-03-15
78	1906-01-01	1906-01	1906-Q1	2012-08-16	2012-08-16	2012-08-16
82	1908-01-01	1908-01	1908-Q1	2013-08-25	2013-08-25	2014-02-21
84	1910-01-01	1910-01	1910-Q1	2013-04-01	2013-04-01	2013-04-01

In [12]:
```
# Приводим к типу даты и времени, используя определенный формат
for col in date_columns:
    if col in ['founded_at', 'first_funding_at', 'mid_funding_at', 'last_funding_at']:
        df[col] = pd.to_datetime(df[col], format='%Y-%m-%d', errors='coerce')
    elif col == 'founded_month':
        df[col] = pd.to_datetime(df[col], format='%Y-%m', errors='coerce')
    elif col == 'founded_quarter':
        df[col] = pd.PeriodIndex(df[col], freq='Q').to_timestamp()
    else:
        pass
# Выведем типы столбцов
print(df[date_columns].dtypes)
# Выведем первые столбцы с датами
df[date_columns].head()
```

```
founded_at          datetime64[ns]
founded_month       datetime64[ns]
founded_quarter     datetime64[ns]
first_funding_at    datetime64[ns]
mid_funding_at      datetime64[ns]
last_funding_at     datetime64[ns]
dtype: object
```

Out[12]:

	founded_at	founded_month	founded_quarter	first_funding_at	mid_funding_at	last_funding_at
0	NaT	NaT	NaT	2014-01-06	NaT	2014-01-06
1	1785-01-01	NaT	NaT	2014-05-15	NaT	2014-05-15
2	1802-07-19	NaT	NaT	2009-07-02	2009-07-02	2009-07-02
3	1817-01-01	NaT	NaT	2013-11-21	2013-11-21	2014-11-03
4	1826-01-01	NaT	NaT	2014-01-14	NaT	2014-01-14

Для датасета `cb_returns` в датафрейме `df_returns` сделаем столбец `year` индексом.

In [13]:
```
# В датафрейме сделаем столбец year индексом
df_returns.set_index('year', inplace=True)
# Выводим первые строки
df_returns.head()
```

Out[13]:

year	seed	venture	equity_crowdfunding	undisclosed	convertible_note	debt_financing	angel	grant	private_equity	post_ipo_equity	post_ipo_debt	secondary_market	product_crowdfunding
2000	16.70	55.40	0.0	78.21	0.00	8.66	6.43	0.0	0.00	0.94	0.0	0.20	0.0
2001	2.88	23.49	0.0	21.50	0.01	4.49	1.18	0.0	0.00	0.46	0.0	0.46	0.0
2002	6.59	209.42	0.0	25.77	0.02	3.42	3.41	0.0	1.51	0.34	0.0	0.06	0.0
2003	7.74	233.86	0.0	9.40	0.01	1.09	3.41	0.0	1.62	2.11	0.0	0.08	0.0
2004	9.93	555.90	0.0	33.19	0.01	13.55	9.18	0.0	2.19	3.38	0.0	0.55	0.0

Обработаем текстовые данные. Пропуски в текстовых столбцах заполним заглушками `unknown`.

In [14]:
```
columns_list = [] # список текстовых столбцов
for col in df.columns: # идем по всем столбцам
    if df[col].dtype == 'object': # условие, что текстовый столбец
        columns_list.append(col) # добавляем название столбца в список столбцов
        df[col] = df[col].str.strip() # удаление пробелов
        df[col] = df[col].str.lower() # в нижний регистр
        df[col] = df[col].fillna('unknown') # заполнение пропусков
print(f'Обработанные столбцы: {columns_list}')
# Выведем первые строки
df[columns_list].head()
```

```
Обработанные столбцы: ['name', 'homepage_url', 'category_list', 'market', 'status', 'country_code', 'state_code', 'region', 'city']
```

Out[14]:

	name	homepage_url	category_list	market	status	country_code	state_code	region	city
0	harvard university	http://harvard.edu	|education|	education	operating	usa	ma	boston	cambridge
1	university of new brunswick	http://www.unb.ca	unknown	unknown	operating	unknown	unknown	unknown	unknown
2	dupont	http://www.dupont.com	|business services|agriculture|automotive|inve...	business services	operating	usa	de	wilmington, delaware	wilmington
3	university of michigan	http://www.umich.edu/	|education|	education	operating	usa	mi	detroit	ann arbor
4	case western reserve university	http://www.case.edu	|education|	education	operating	usa	oh	cleveland	cleveland

Обработаем полные дубликаты в данных и пропуски в `funding_total_usd`. Избавимся от тех строк, которые не несут какой-либо информации либо не содержат данных о финансировании.

In [15]:
```
# Количество полных дубликатов
dub_count = df.duplicated().sum()
# Количество строк
rows_count = df.shape[0]
# Выводим количество и процент полных дубликатов
print(f'Полных дубликатов: {dub_count} из {rows_count} строк ({dub_count / rows_count * 100:.2f}%)')
# Удаляем дубликаты, сохраняя первый экземпляр
df.drop_duplicates(keep='first', inplace=True)
new_rows_count = df.shape[0]
# Выводим информацию после удаления дубликатов
print(f'Дубликаты удалены: осталось {new_rows_count} из {rows_count} строк ({new_rows_count / rows_count * 100:.2f}%)')
```

```
Полных дубликатов: 4855 из 54294 строк (8.94%)
Дубликаты удалены: осталось 49439 из 54294 строк (91.06%)
```

Создадим список столбцов `funding_columns`, содержащих инвестиции. Воспользуемся тем, что названия таких столбцов в двух датафреймов совпадают.

In [16]:
```
# Создаем список столбцов с инвестициями
funding_columns = list(df_returns.columns)
# Выведем список
funding_columns
```

Out[16]:
```
['seed',
 'venture',
 'equity_crowdfunding',
 'undisclosed',
 'convertible_note',
 'debt_financing',
 'angel',
 'grant',
 'private_equity',
 'post_ipo_equity',
 'post_ipo_debt',
 'secondary_market',
 'product_crowdfunding']
```

Найдем строки с пропусками или нулями в столбце `funding_total_usd`.

In [17]:
```
# Создадим маску на условие пропусков или нулей
mask = df['funding_total_usd'].isna() | (df['funding_total_usd'] == 0)

# Результат проверки на пропуски или нули столбца funding_total_usd
print(f'Строк с пропусками или нулями в столбце "funding_total_usd" было: {mask.sum()}')
```

```
Строк с пропусками или нулями в столбце "funding_total_usd" было: 8532
```

Попробуем восстановить пропуски или нули суммой столбцов из списка `funding_columns` с отдельными инвестициями.

In [18]:
```
# Присвоим пропускам или нулям сумму столбцов из списка funding_columns
df.loc[mask, 'funding_total_usd'] = df.loc[mask, funding_columns].sum(axis=1)

# Обновим маску на условие пропусков или нулей
mask = df['funding_total_usd'].isna() | (df['funding_total_usd'] == 0)

# Результат проверки на пропуски или нули столбца funding_total_usd
print(f'Строк с пропусками или нулями в столбце "funding_total_usd" стало: {df[mask].shape[0]}')
```

```
Строк с пропусками или нулями в столбце "funding_total_usd" стало: 8532
```

Получается, что строки с пропусками или нулями в поле `funding_columns` не получилось заполнить значениями, т. к. остальные столбцы с отдельными инвестициями тоже были пустые или нулевые. Данные строки не несут информации о финансировании, поэтому их можно удалить.

In [19]:
```
# Выводим сколько строк было
rows_count = df.shape[0]
print(f'Было: {rows_count} строк')
```

```
# Удаляем строки с нулевым значением в столбце funding_total_usd
df = df[~mask]

# Выводим сколько строк стало
new_rows_count = df.shape[0]
print(f'Удалено: {rows_count - new_rows_count} ({(rows_count - new_rows_count) / rows_count * 100:.2f}%) строк')
print(f'Осталось: {new_rows_count} строк')
```

```
Было: 49439 строк
Удалено: 8532 (17.26%) строк
Осталось: 40907 строк
```

Удалили 8532 (17% от 49439) строк без данных о финансировании.

Заполним пропуски в значениях `mid_funding_at` на основании значений в столбцах `first_funding_at` и `last_funding_at`. В качестве нового значения вместо пропусков возьмем приблизительно середину интервала между этими двумя датами.

In [20]:
```
# Сколько пропусков в поле mid_funding_at было
mask = df['mid_funding_at'].isna()
rows_count = mask.sum()
print(f'Пропусков в поле "mid_funding_at" было: {rows_count} ({rows_count / df.shape[0] * 100:.2f}% всех строк)')

# Заполняем пропуски с помощью значений first_funding_at и last_funding_at
diff_date = (df.loc[mask, 'last_funding_at'] - df.loc[mask, 'first_funding_at'])
df.loc[mask, 'mid_funding_at'] = df.loc[mask, 'first_funding_at'] + (diff_date // 2)

# Сколько пропусков в поле mid_funding_at стало
mask = df['mid_funding_at'].isna()
new_rows_count = mask.sum()
print(f'Пропусков в поле "mid_funding_at" стало: {new_rows_count}')

# Проверим незаполненные пустые
df.loc[mask, ['first_funding_at', 'mid_funding_at', 'last_funding_at']]
```

```
Пропусков в поле "mid_funding_at" было: 13676 (33.43% всех строк)
Пропусков в поле "mid_funding_at" стало: 1
```

Out[20]:

	first_funding_at	mid_funding_at	last_funding_at
33041	NaT	NaT	2014-09-25

Строки с пропусками в значениях `mid_funding_at` составляли треть всех строк. После заполнения пропусков остался только 1 пропуск, из-за того, что у этой записи пустое значение `first_funding_at`, присвоим `mid_funding_at` непустое значение `last_funding_at`.

In [21]:
```
# Присвоим пустому mid_funding_at непустое значение last_funding_at.
df.loc[mask, 'mid_funding_at'] = df.loc[mask, 'last_funding_at']
# Проверим
df.loc[mask, ['first_funding_at', 'mid_funding_at', 'last_funding_at']]
```

Out[21]:

	first_funding_at	mid_funding_at	last_funding_at
33041	NaT	2014-09-25	2014-09-25

Результаты предобработки:

- В датафрейме `df` 40 столбцов и 54294 строк.
- В датафрейме `df_returns` с суммами возвратов от финансирования 14 столбцов и 15 строк.
- Нормализованы названия столбцов датафрейма `df`.
- Исправлены данные в столбце `funding_total_usd` и переведены в числовой тип.
- Данные с информацией о дате переведены в соответствующий тип.
- В датафрейме `df_returns` индексами сделали поле `year`.
- Нормализовали данные в текстовых столбцах. Пропуски в текстовых столбцах заполним заглушками `unknown`.
- Удалили 4855 полных дубликатов из 54294 строк (8.94%), осталось 49439 строк.
- Удалили 8532 строк (17% от 49439 строк) без данных о финансировании в столбце `funding_total_usd` и столбцах с отдельными инвестициями, осталось 40907 строк от начального количества.
- Строки с пропусками в значениях `mid_funding_at` составляли треть всех строк и успешно были заполнены по столбцам `first_funding_at` и `last_funding_at`.
- Итого из 54294 строк всего была удалена четверть - 13387 строк (24.7%). Оставшихся данных достаточно для анализа.

Шаг 2. Инжиниринг признаков

Группы по срокам финансирования

Разделим все компании на три группы:

- Единичное финансирование — был всего один раунд финансирования.
- Срок финансирования до года — между первым и последним раундом финансирования прошло не более года.
- Срок финансирования более года.

In [22]:
```
# Создаем новый столбец с группой финансирования и присваиваем значение по умолчанию "более года"
df['funding_type'] = 'более года'
# Найдем группу "единичное"
df.loc[df['funding_rounds'] == 1, 'funding_type'] = 'единичное'
# Найдем группу "до года"
df.loc[(df['funding_rounds'] > 1) & ((df['last_funding_at'] - df['first_funding_at']) <= pd.Timedelta(days=365)), 'funding_type'] = 'до года'
```

Визуализируем соотношение этих групп, создав два графика:

- По количеству компаний: покажем, какой процент от общего числа компаний относится к каждой из трёх групп.
- По объёму инвестиций: отобразим, какую долю от общего объёма привлечённых средств получила каждая группа.

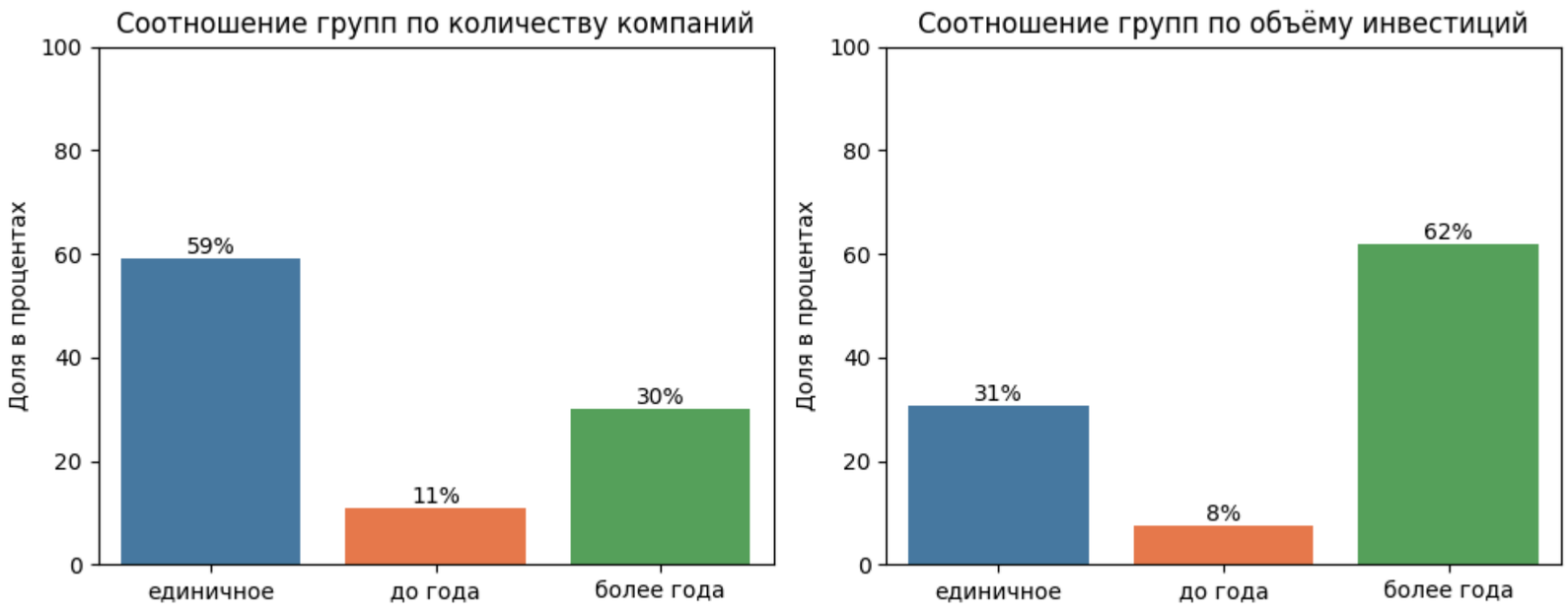
Для ясности и согласованности используем единую цветовую палитру для всех графиков, чтобы каждая категория всегда отображалась одним и тем же цветом.

In [23]:
```
# Палитра: синий, оранжевый, зелёный
colors_list = colors = ['#377EB0', '#FF6F33', '#4CAF50']
# Порядок групп
category_order = ['единичное', 'до года', 'более года']

# Считаем и количество, и сумму инвестиций по категориям
df_agg = df.groupby('funding_type').agg(
    count=('funding_type', 'size'),                # количество компаний
    total_investment=('funding_total_usd', 'sum')  # сумма инвестиций
).reset_index()

# Рассчитываем доли в процентах
total_count = df_agg['count'].sum()
total_invest = df_agg['total_investment'].sum()
df_agg['percent_count'] = (df_agg['count'] / total_count) * 100
df_agg['percent_invest'] = (df_agg['total_investment'] / total_invest) * 100

# Сортируем датафрейм по нужному порядку категорий
df_agg = df_agg.set_index('funding_type').loc[category_order].reset_index()

# Выводим
df_agg
```

Out[23]:

	funding_type	count	total_investment	percent_count	percent_invest
0	единичное	24113	1.993044e+11	58.945902	30.618237
1	до года	4501	4.888598e+10	11.003007	7.510132
2	более года	12293	4.027433e+11	30.051092	61.871631

In [24]:
```
# Создаём полотно с двумя подграфиками (2 графика по горизонтали)
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 4))

# Первый график: Соотношение групп по количеству компаний
sns.barplot(
    data=df_agg,
    x='funding_type',
    y='percent_count',
    palette=colors_list,
    order=category_order,
    ax=ax1  # указываем ось для первого графика
)

# Добавляем подписи к первому графику
for idx, value in enumerate(df_agg['percent_count']):
    ax1.text(idx, value, f'{value:.0f}%', ha='center', va='bottom',)

# Оформление первого графика
ax1.set_title('Соотношение групп по количеству компаний')
ax1.set_ylabel('Доля в процентах')
ax1.set_xlabel('')
ax1.set_ylim(0, 100)

# Второй график: соотношение по объёму инвестиций
sns.barplot(
    data=df_agg,
    x='funding_type',
    y='percent_invest',
    palette=colors_list,
    order=category_order,
    ax=ax2  # указываем ось для второго графика
)

# Добавляем подписи ко второму графику
for idx, value in enumerate(df_agg['percent_invest']):
    ax2.text(idx, value, f'{value:.0f}%', ha='center', va='bottom')

# Оформление второго графика
ax2.set_title('Соотношение групп по объёму инвестиций')
ax2.set_ylabel('Доля в процентах')
ax2.set_xlabel('')
ax2.set_ylim(0, 100)

# Общая настройка и отображение
plt.tight_layout()
plt.show()
```



Распределение групп финансирования по количеству компаний / по объему инвестиций:

- Единичное - 59% / 31%.
- До года - 11% / 8%.
- Более года - 30% / 62%.

Выделение средних и нишевых сегментов рынка

Компании указывают свой сегмент рынка в столбце `market`. Рассчитаем, как часто в датасете встречается каждый из сегментов. Сегменты, к которым относится более 120 компаний, отнесите к массовым, сегменты, в которые входит от 35 до 120 включительно, отнесите к средним, а сегменты до 35 компаний отнесите к нишевым.

In [25]:
```
# Распределение количества компаний по сегменту
df_segment = df['market'].value_counts(ascending=False).reset_index()
df_segment.columns = ['market', 'count']

# Назначаем категории
df_segment['category'] = 'средний'
df_segment.loc[df_segment['count'] > 120, 'category'] = 'массовый'
df_segment.loc[df_segment['count'] < 35, 'category'] = 'нишевый'

# Рассчитаем, сколько сегментов попадает в каждую из категорий.
df_segment['category'].value_counts(ascending=False).reset_index()
```

Out[25]:

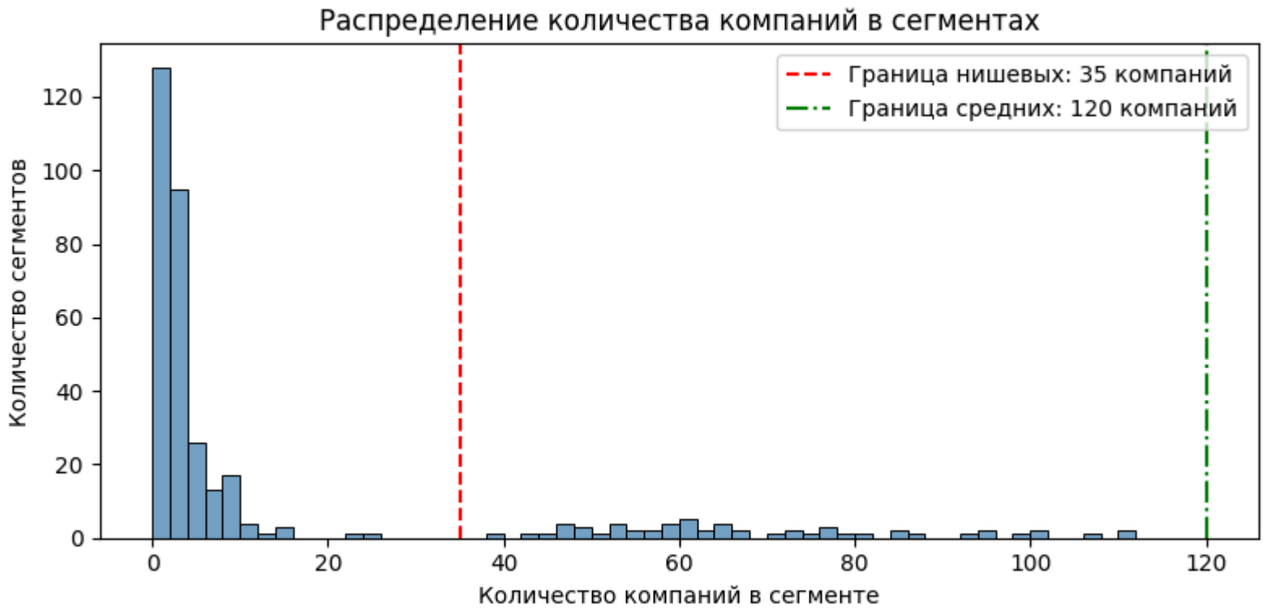
	index	category
0	нишевый	289
1	средний	57
2	массовый	49

1. Нишевых сегментов - 289.
2. Средних сегментов - 57.
3. Массовых сегментов - 49.

Построим график распределения количества компаний в сегментах и отобразите на нём разделение на нишевые и средние сегменты.

In [26]:
```
# Размер графика
plt.figure(figsize=(8, 4))

# Гистограмма
ax = sns.histplot(
    data=df_segment['count'],
```

```
    bins=range(0,121,2),
    color='steelblue',
)

plt.axvline(35, color='r', linestyle='--', label='Граница нишевых: 35 компаний')
plt.axvline(120, color='g', linestyle='-.', label='Граница средних: 120 компаний')


# Оформление
plt.title('Распределение количества компаний в сегментах')
plt.ylabel('Количество сегментов')
plt.xlabel('Количество компаний в сегменте')
plt.legend()

# Отобразить график
plt.tight_layout()
plt.show()
```



Вывод: Подавляющее большинство сегментов - нишевые, содержащие до 5 компаний.

Оставим в столбце `market` только массовые сегменты. Для остальных сегментов заменим значения на заглушки — `niche` для нишевых и `mid` для средних.

Дальнейшие исследования выполним с учётом этой замены. Индивидуальные сегменты внутри средней и нишевой групп рассматривать не нужно — они объединяются в два общих сегмента.

In [27]:
```
# создаем df_cor – копию датафрейма df с присоединением столбца category из датафрейма df_segment
df_corr = df.copy().merge(df_segment[['market', 'category']], on='market', how='left')
# замена средних сегментов
df_corr.loc[df_corr['category'] == 'средний', 'market'] = 'mid'
# замена нишевых сегментов
df_corr.loc[df_corr['category'] == 'нишевый', 'market'] = 'niche'
# выводим для проверки
df_corr[['market', 'category']]

# Рассчитаем, сколько компаний попадает в каждый сегмент
df_corr['market'].value_counts(ascending=False).reset_index()
```

Out[27]:

	index	market
0	software	4812
1	mid	3841
2	biotechnology	3590
3	unknown	2503
4	mobile	2344
5	e-commerce	1866
6	curated web	1693
7	enterprise software	1381
8	health care	1185
9	clean technology	1180
10	games	1117
11	advertising	1107
12	hardware + software	1062
13	social media	1003
14	health and wellness	873
15	education	844
16	niche	830
17	finance	828
18	analytics	667
19	manufacturing	596
20	security	567
21	semiconductors	484
22	web hosting	424
23	consulting	349
24	hospitality	336
25	travel	330
26	fashion	303
27	news	301
28	messaging	295
29	search	291
30	real estate	279
31	saas	272
32	music	264
33	internet	241
34	technology	238
35	apps	223
36	photography	204
37	sports	204
38	marketplaces	196
39	video	188
40	social network media	158
41	automotive	155
42	cloud computing	152
43	medical	151
44	entertainment	150
45	big data	150
46	networking	143
47	design	135
48	nonprofits	135
49	public relations	134
50	startups	133

Шаг 3. Работа с выбросами и анализ

Анализируем и помечаем выбросы в каждом из сегментов

Заказчика интересует обычный для рассматриваемого периода размер средств, который предоставлялся компаниям.

По предобработанному столбцу `funding_total_usd` графическим образом оценим, какой размер общего финансирования для одной компании будет типичным, а какой — выбивающимся. Укажем интервал, в котором лежат типичные значения.

Для удобства в столбце `funding_total_usd` изменим единицы измерения с долларов на млн долларов.

In [28]:
```
# В столбце `funding_total_usd` изменим единицы измерения с долларов на млн долларов
df_corr['funding_total_usd'] = df_corr['funding_total_usd'] / 1000000
```

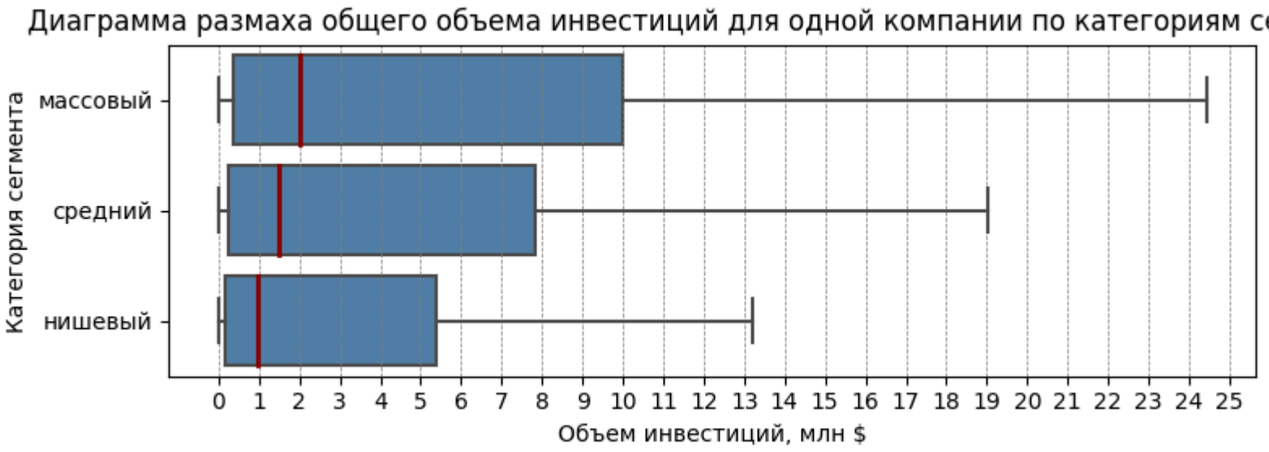
In [29]:
```
# Размер графика
plt.figure(figsize=(8, 2))

# Диаграмма размаха
ax = sns.boxplot(
    data=df_corr,
    orient='h',
    x='funding_total_usd',
    showfliers=False,
    color='steelblue',
    medianprops={'color': 'darkred', 'linewidth': 2}
    )

# Оформление
plt.title('Диаграмма размаха общего объема инвестиций для одной компании')
plt.ylabel('')
plt.xlabel('Объем инвестиций, млн $')
```

```
# Добавляем метки с шагом 1
plt.xticks(range(0, 26, 1))
# Убираем научный формат
plt.ticklabel_format(axis='x', style='plain')
# Вертикальные линии сетки
ax.grid(axis='x', color='gray', linestyle='--', linewidth=0.5)

# Отобразить график
plt.tight_layout()
plt.show()
```

Вывод:

- По диаграмме размаха видно, что распеределение асимметричное правостороннее.
- Медиана около 2 млн USD.
- Типичные значения в пределах до 24.5 млн USD.

In [30]:
```
# Размер графика
plt.figure(figsize=(8, 3))

# Диаграмма размаха
ax = sns.boxplot(
    data=df_corr,
    orient='h',
    x='funding_total_usd',
    y='category',
    showfliers=False,
    color='steelblue',
    medianprops={'color': 'darkred', 'linewidth': 2}
    )

# Оформление
plt.title('Диаграмма размаха общего объема инвестиций для одной компании по категориям сегментов')
plt.ylabel('Категория сегмента')
plt.xlabel('Объем инвестиций, млн $')

# Добавляем метки с шагом 1
plt.xticks(range(0, 26, 1))
# Убираем научный формат
plt.ticklabel_format(axis='x', style='plain')
# Вертикальные линии сетки
ax.grid(axis='x', color='gray', linestyle='--', linewidth=0.5)

# Отобразить график
plt.tight_layout()
plt.show()
```



Вывод:

- По диаграмме размаха по категориям сегментов видно, что у всех категорий также распеределение асимметричное правостороннее.
- Медиана у нишевых сегментов около 1 млн USD, у средних - 1.5 млн USD, у массовых - 2 млн USD.
- Предел типичных значений у нишевых сегментов до 13.2 млн USD, у средних - до 19 млн USD, у массовых - до 24.5 млн USD.

Определим компании с аномальным объёмом общего финансирования, используя метод IQR отдельно по каждому сегменту. Все нишевые сегменты были объединены в одну группу, а средние — в другую.

In [31]:
```
# Создаем столбец-индикатор аномального значения
df_corr['anomaly'] = False
# Список уникальных названий сегментов
market_list = df_corr['market'].unique()

# Определяем компании с аномальным финансированием отдельно по каждому сегменту
for market in market_list:
    ser = df_corr.loc[df_corr['market'] == market, 'funding_total_usd']
    q_1 = ser.quantile(0.25)
    q_3 = ser.quantile(0.75)
    iqr = q_3 - q_1
    lower =  q_1 - 1.5 * iqr
    upper =  q_3 + 1.5 * iqr
    mask = (df_corr['market'] == market) & ((df_corr['funding_total_usd'] > upper) | (df_corr['funding_total_usd'] < lower))
    df_corr.loc[mask, 'anomaly'] = True

# Выведем количество компаний с аномальным финансированием
print(f'Количество компаний с аномальным финансированием: {df_corr["anomaly"].sum()} ({df_corr["anomaly"].sum() / df_corr.shape[0] * 100:.1f}%)')
print(f'Общее количество компаний в датафрейме: {df_corr.shape[0]}')
```

```
Количество компаний с аномальным финансированием: 5244 (12.8%)
Общее количество компаний в датафрейме: 40907
```

Компаний с аномальным финансированием: 5244 (12.8%) из 40907.

Определим сегменты рынка с наибольшей долей компаний, получивших аномальное финансирование, и выведем топ таких сегментов.

In [32]:
```
# Группируем по сегменту и найдем долю аномальных компаний в процентах
df_grouped = df_corr.groupby('market')['anomaly'].mean() * 100
# Сортируем по убыванию доли аномальных и сбросим индексы
df_grouped = df_grouped.sort_values(ascending=False).reset_index()
df_grouped.columns = ['market', 'percent']
# Выведем топ-10
df_grouped.head(10)
```

Out[32]:

	market	percent
0	real estate	17.204301
1	entertainment	16.666667
2	consulting	16.618911
3	search	16.494845
4	cloud computing	16.447368
5	saas	16.176471
6	photography	16.176471
7	technology	15.966387
8	video	15.957447
9	niche	15.903614

Топ-10 сегментов компаний с аномальным финансированием:

1. real estate - 17.2%
2. entertainment - 16.7%
3. consulting - 16.6%
4. search - 16.5%
5. cloud computing - 16.4%
6. saas - 16.2%
7. photography - 16.2%
8. technology - 16.0%
9. video - 16.0%
10. niche - 15.9%

Определяем границы рассматриваемого периода, отбрасываем аномалии

Проверим по датасету, можно ли считать, что нам предоставили полные данные за 2014 год.

In [33]:
```
# Оставим только даты последнего финансирования за 2014 год
df_last_date = df_corr.loc[df_corr['last_funding_at'].dt.year == 2014, 'last_funding_at']
# Подсчитаем количество последних финансирований в каждый день
df_last_date = df_last_date.value_counts().reset_index()
df_last_date.columns = ['last_funding_at', 'count']
# Отсортируем по дате
df_last_date = df_last_date.sort_values('last_funding_at')

df_last_date
```

Out[33]:

	last_funding_at	count
0	2014-01-01	274
137	2014-01-02	41
241	2014-01-03	15
269	2014-01-04	7
258	2014-01-05	10
...	...	...
332	2014-11-29	1
215	2014-12-01	26
225	2014-12-02	24
329	2014-12-24	1
331	2014-12-31	1

335 rows × 2 columns

In [34]:
```
# Добавим столбец месяца 2014 года
df_last_date['last_funding_at'] = pd.to_datetime(df_last_date['last_funding_at'])
df_last_date['month'] = df_last_date['last_funding_at'].dt.month
# Подсчитаем количество последних финансирований по месяцам 2014 года
df_last_date.groupby('month')['count'].sum()
```

Out[34]:
```
month
1     1178
2      947
3     1072
4     1077
5     1036
6     1229
7     1254
8     1064
9     1135
10    1139
11     732
12      52
Name: count, dtype: int64
```

Вывод:

- За 2014 год приведены данные, включающие 335 дней от 1 янв 2014 г. до 31 дек 2014 г.
- Отмечается заметное снижение финансирований в ноябре, а в декабре финансирований крайне мало (примерно в 20 раз меньше любого месяца с января по октябрь) - что указывает на потенциальную неполноту данных за конец 2014 г.

Исключим из датасета компании, которые ранее посчитали получившими аномальное финансирование.

In [35]:
```
# Создает датафрейм без аномальных финансирований
df_clean = df_corr.loc[df_corr['anomaly'] == False].copy()
# Выведем количество строк
print(f'Строк было: {df_corr.shape[0]}')
print(f'Удалено аномальных финансирований: {df_corr.shape[0] - df_clean.shape[0]} ({(df_corr.shape[0] - df_clean.shape[0]) / df_corr.shape[0] * 100:.1f}%)')
print(f'Строк стало: {df_clean.shape[0]}')
```
```
Строк было: 40907
Удалено аномальных финансирований: 5244 (12.8%)
Строк стало: 35663
```

Из 40907 удалено 5244 (12.8%) компаний с аномальным финансированием, осталось 35663 компаний.

После исключения аномальных записей, на основе столбцов `mid_funding_at` и `funding_rounds` оставим в датасете данные только о компаниях, которые получали финансирование в годы, когда было зафиксировано 50 или более раундов финансирования.

In [36]:
```
# Добавим новый столбец mid_funding_at_year с годом столбца mid_funding_at
df_clean['mid_funding_at_year'] = df_clean['mid_funding_at'].dt.year
```

```
# Выведем интересующие нас строки
df_clean[['mid_funding_at', 'mid_funding_at_year', 'funding_rounds']].head()
```

Out[36]:

	mid_funding_at	mid_funding_at_year	funding_rounds
1	2014-05-15	2014	1.0
2	2009-07-02	2009	1.0
3	2013-11-21	2013	3.0
4	2014-01-14	2014	1.0
7	2014-01-12	2014	1.0

In [37]:
```
# Считаем количество раундов в год
rounds = df_clean['mid_funding_at_year'].value_counts().sort_index(ascending=False)

# Годы с 50 и более раундами
years = rounds[rounds >= 50].index.tolist()
print(f'Годы, когда было 50 и более раундов:', years)

# Фильтруем записи в годы с 50 и более раундами
df_filtered = df_clean.loc[df_clean['mid_funding_at_year'].isin(years)].copy()

# Выводим количество записей
print(f'Строк было: {df_clean.shape[0]}')
print(f'Строк отфильтровано: {(df_clean.shape[0] - df_filtered.shape[0])} ({(df_clean.shape[0] - df_filtered.shape[0]) / df_clean.shape[0] * 100:.1f}%)')
print(f'Строк стало: {df_filtered.shape[0]}')
```

```
Годы, когда было 50 и более раундов: [2014, 2013, 2012, 2011, 2010, 2009, 2008, 2007, 2006, 2005, 2004, 2003, 2000]
Строк было: 35663
Строк отфильтровано: 159 (0.4%)
Строк стало: 35504
```

После фильтрации 159 (0.4%) строк в датафрейме осталось 35503 записей.

Анализ типов финансирования по объёму и популярности

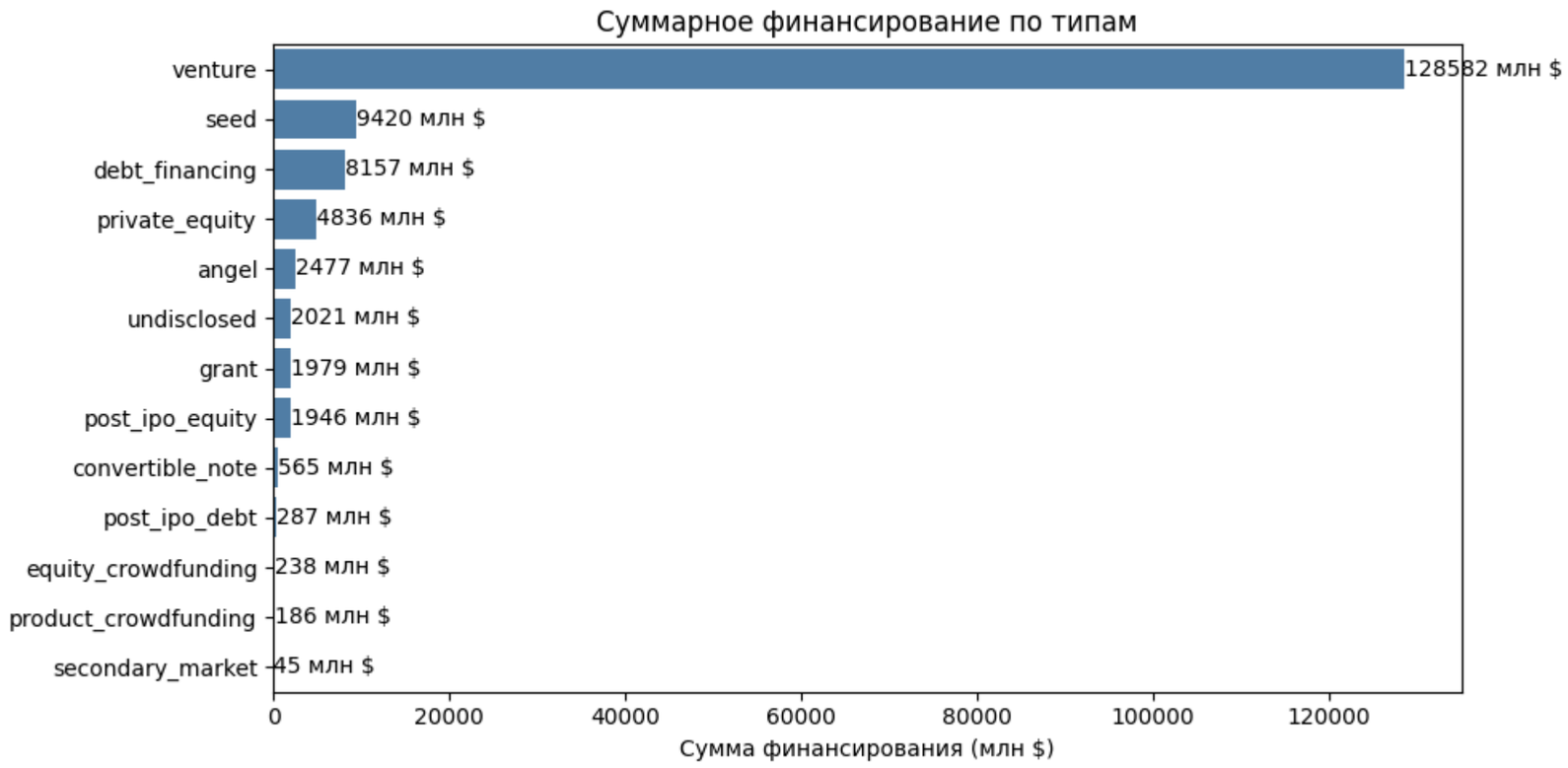
Построим график, который покажет, какие типы финансирования в сумме привлекли больше всего денег. Переведем суммы инвестиций по различным типам финансирования в млн USD.

In [38]:
```
# Сформируем список названий данных столбцов, заметив, что названия можно взять у другого датафрейм
funding_columns_list = df_returns.columns.tolist()
# Переведем суммы различных типов финансирования в млн USD
df_filtered[funding_columns_list] = df_filtered[funding_columns_list] / 1000000
```

In [39]:
```
# Найдем сумму по каждому типу финансирования
df_total_funding = df_filtered[funding_columns_list].sum()

# Сбрасываем индексы
df_total_funding = df_total_funding.reset_index()

# Переименуем названия столбцов
df_total_funding.columns = ['funding_type', 'total_sum']

# Сортируем по убыванию
df_total_funding = df_total_funding.sort_values('total_sum', ascending=False).reset_index(drop=True)

# Суммы финансирования переведем в млн долларов
df_total_funding['total_sum'] = df_total_funding['total_sum']

# Выведем результат с форматированием
display(df_total_funding.style.format({'total_sum': '{:,.0f} млн $'}))
```

	funding_type	total_sum
0	venture	128,582 млн $
1	seed	9,420 млн $
2	debt_financing	8,157 млн $
3	private_equity	4,836 млн $
4	angel	2,477 млн $
5	undisclosed	2,021 млн $
6	grant	1,979 млн $
7	post_ipo_equity	1,946 млн $
8	convertible_note	565 млн $
9	post_ipo_debt	287 млн $
10	equity_crowdfunding	238 млн $
11	product_crowdfunding	186 млн $
12	secondary_market	45 млн $

In [40]:
```
# Строим стобчатую диаграмму
plt.figure(figsize=(10, 5))

ax = sns.barplot(
    data=df_total_funding,
    y='funding_type',
    x='total_sum',
    color='steelblue'
)

# Настраиваем оформление
plt.title('Суммарное финансирование по типам')
plt.xlabel('Сумма финансирования (млн $)')
plt.ylabel('')
# Добавляем подписи к столбцам
for idx, value in enumerate(df_total_funding['total_sum']):
    ax.text(value + 0.5, idx, f'{value:.0f} млн $', ha='left', va='center')

# Выводим график
plt.tight_layout()
plt.show()
```

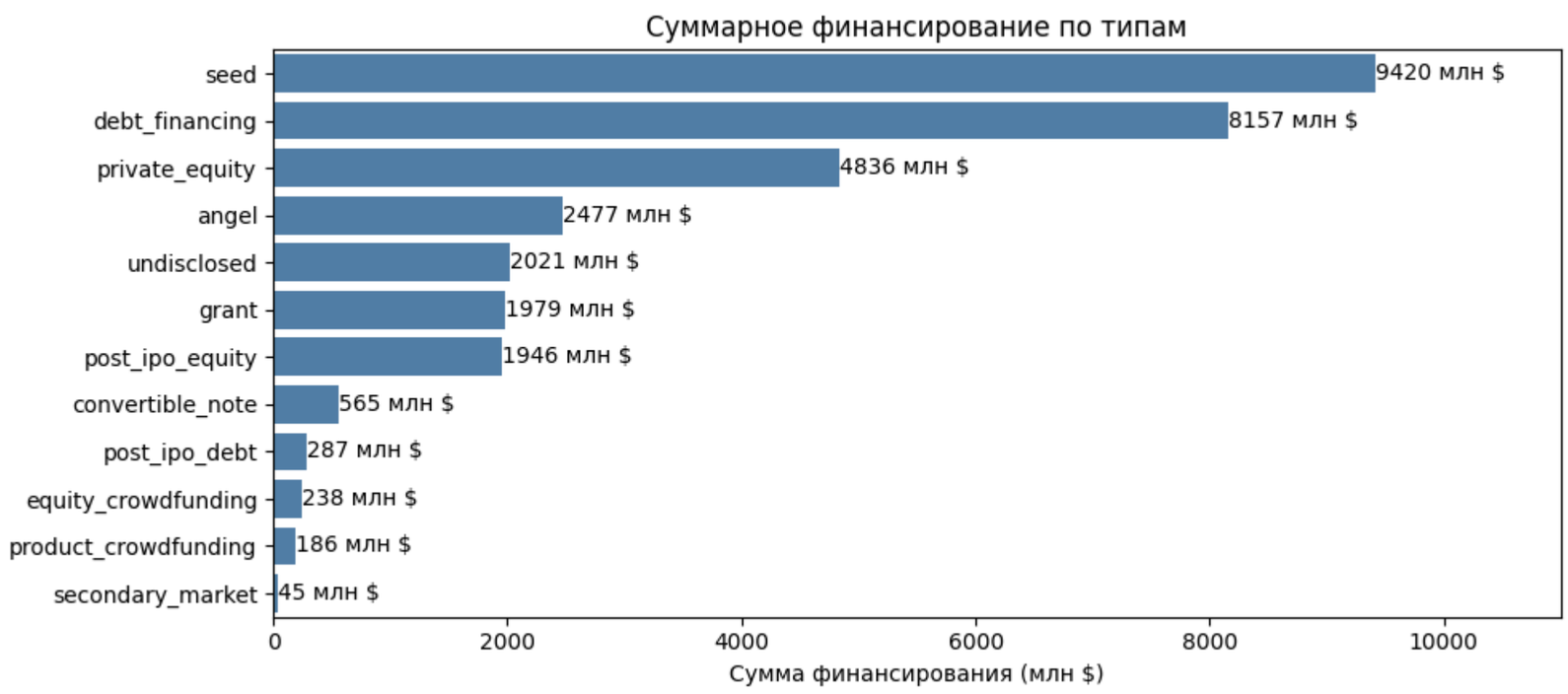


Вывод:

- Венчурные инвестиции значительно превышают остальные вместе взятые.

Построим график без венчурных инвестиций.

In [41]:
```
# Строим стобчатую диаграмму
plt.figure(figsize=(10, 4.5))

ax = sns.barplot(
    data=df_total_funding[df_total_funding['funding_type'] != 'venture'],
    y='funding_type',
    x='total_sum',
    color='steelblue'
)

# Настраиваем оформление
plt.title('Суммарное финансирование по типам')
plt.xlabel('Сумма финансирования (млн $)')
plt.ylabel('')
plt.xlim(0, 11000)

# Добавляем подписи к столбцам
filtered_data = df_total_funding[df_total_funding['funding_type'] != 'venture'].reset_index(drop=True)
for idx, value in enumerate(filtered_data['total_sum']):
    ax.text(value + 0.5, idx, f'{value:.0f} млн $', ha='left', va='center')

# Выводим график
plt.tight_layout()
plt.show()
```

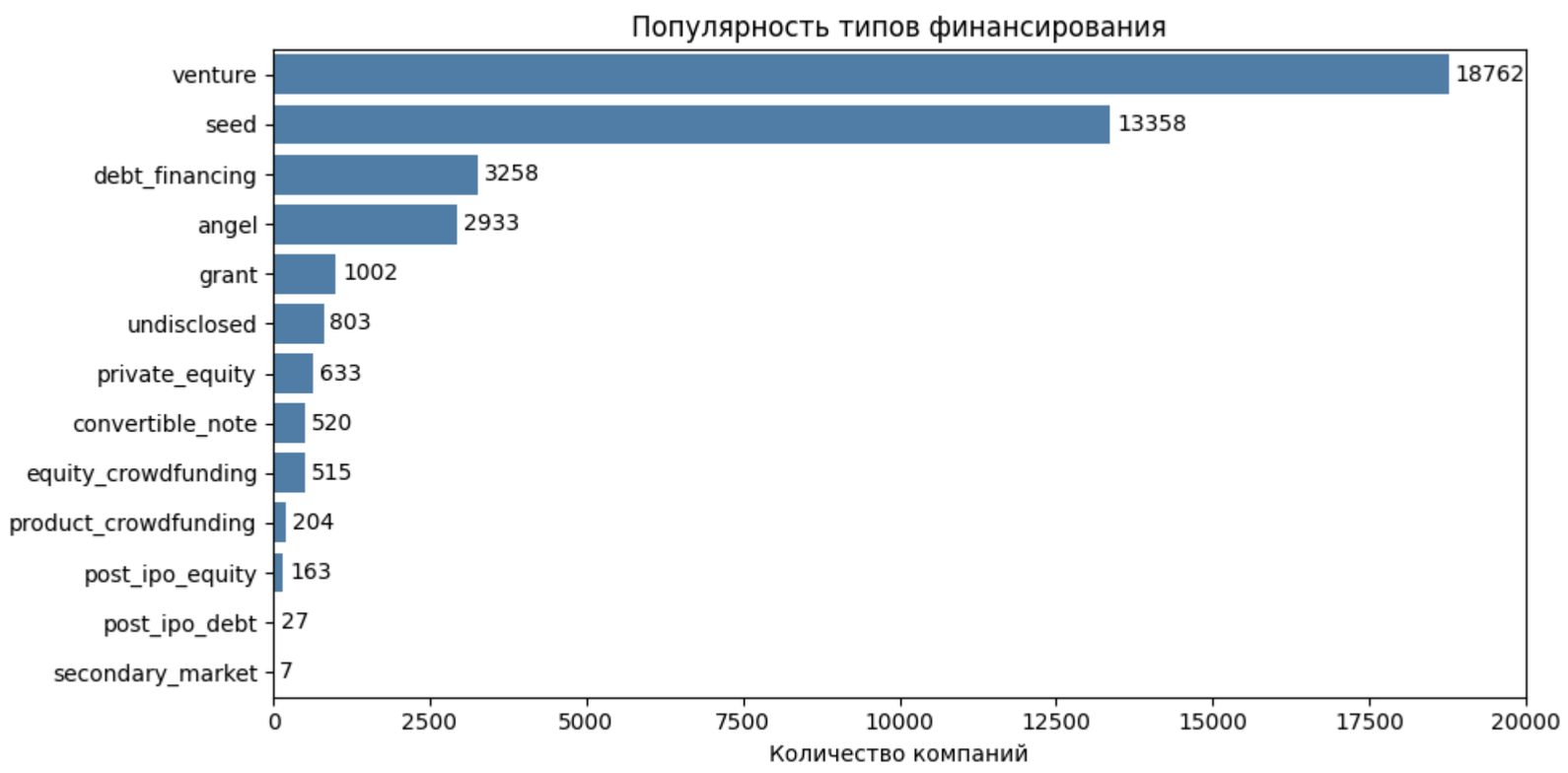


Также построим график, который покажет популярность разных типов финансирования — какие типы финансирования чаще всего используются компаниями, то есть встречаются в датасете наибольшее количество раз.

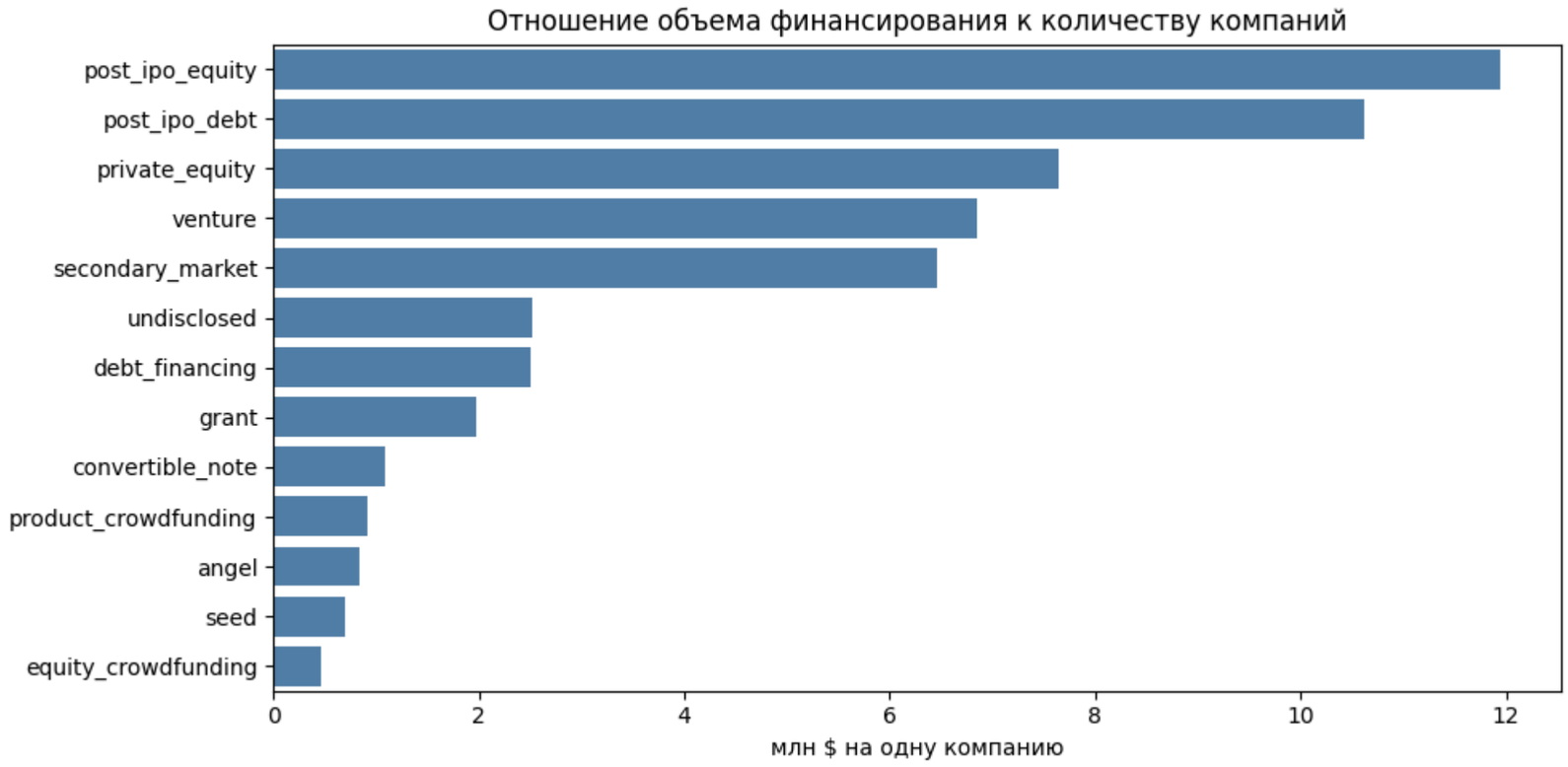
In [42]:
```
# Найдем популярность каждого типа финансирования
df_popular = df_filtered[funding_columns_list].astype(bool).sum()

# Сбрасываем индексы
df_popular = df_popular.reset_index()

# Переименуем названия столбцов
df_popular.columns = ['funding_type', 'popular']

# Сортируем по убыванию
df_popular = df_popular.sort_values('popular', ascending=False).reset_index(drop=True)

# Выведем результат с форматированием
display(df_popular)
```

	funding_type	popular
0	venture	18762
1	seed	13358
2	debt_financing	3258
3	angel	2933
4	grant	1002
5	undisclosed	803
6	private_equity	633
7	convertible_note	520
8	equity_crowdfunding	515
9	product_crowdfunding	204
10	post_ipo_equity	163
11	post_ipo_debt	27
12	secondary_market	7

In [43]:
```
# Строим стобчатую диаграмму
plt.figure(figsize=(10, 5))

ax = sns.barplot(
    data=df_popular,
    y='funding_type',
    x='popular',
    color='steelblue'
)

# Настраиваем оформление
plt.title('Популярность типов финансирования')
plt.xlabel('Количество компаний')
plt.ylabel('')
plt.xlim(0, 20000)
# Добавляем подписи к столбцам
for idx, value in enumerate(df_popular['popular']):
    ax.text(value + 100, idx, f'{value:.0f}', ha='left', va='center')

# Выводим график
plt.tight_layout()
plt.show()
```



In [44]:
```
# Соединим два датафрейма
df_analysis = df_total_funding.merge(df_popular, on='funding_type', how='inner')
# Среднее финансирование одной компании
df_analysis['avg_sum'] = df_analysis['total_sum'] / df_analysis['popular']
# Отсортируем
df_analysis = df_analysis.sort_values('avg_sum', ascending=False)

# Строим стобчатую диаграмму
plt.figure(figsize=(10, 5))

ax = sns.barplot(
    data=df_analysis,
    y='funding_type',
    x='avg_sum',
    color='steelblue'
)

# Настраиваем оформление
plt.title('Отношение объема финансирования к количеству компаний')
plt.xlabel('млн $ на одну компанию')
plt.ylabel('')

# Выводим график
plt.tight_layout()
plt.show()
```



Вывод:

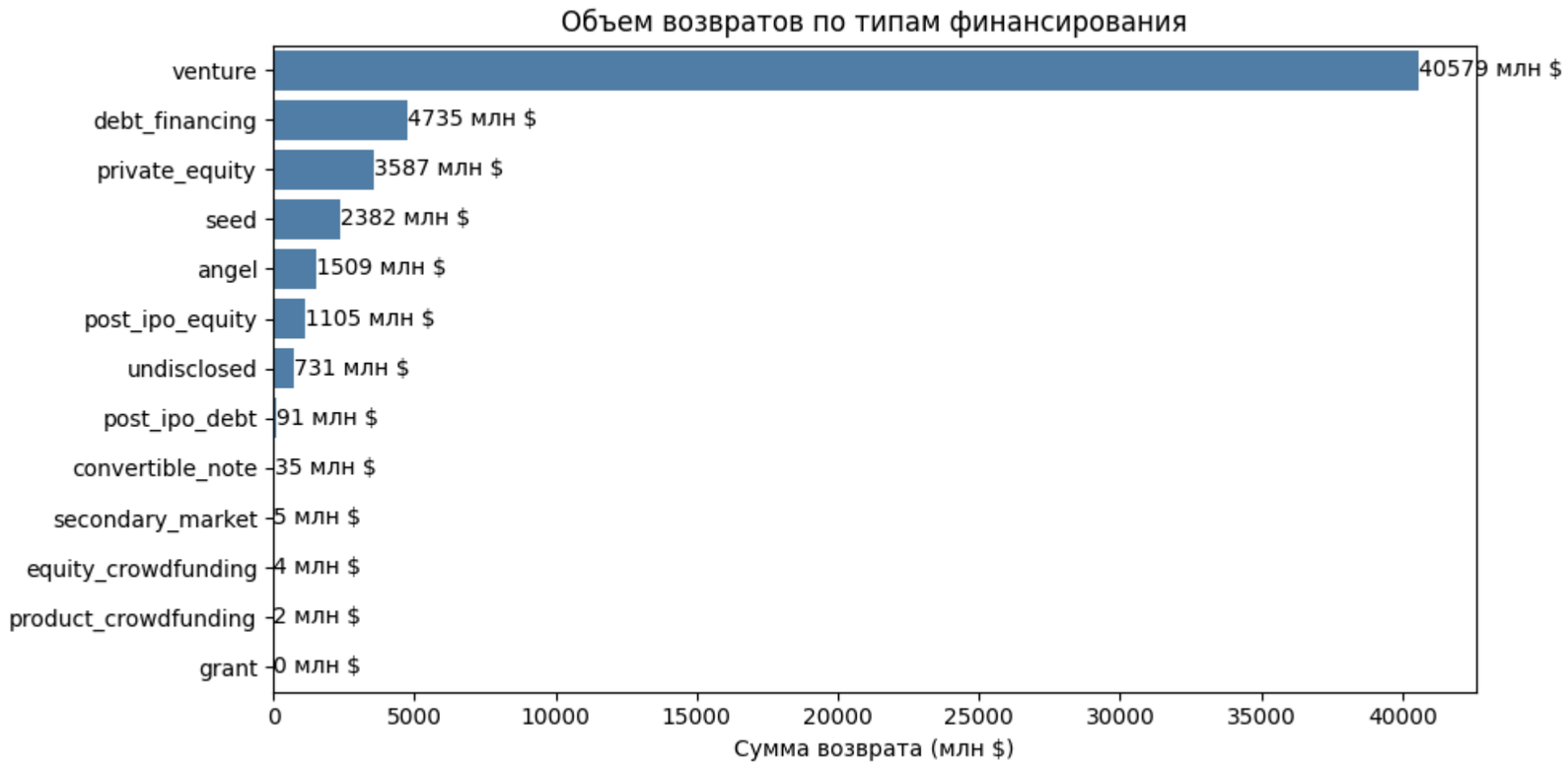
- часто используемые типы финансирования, которые при этом характеризуются небольшими объёмами финансирования: `seed`, `angel`, `equity_crowdfunding`.
- редко используемые типы финансирования, которые при этом характеризуются значительным объёмом финансирования: `private_equity`, `post_ipo_equity`, `post_ipo_debt`.

Построим график суммарных объёмов возвратов от разных типов финансирования за весь период на основе дополнительного датасета.

In [45]:
```
# Найдем сумму по каждому типу финансирования
df_returns_total = df_returns.sum()

# Сбрасываем индексы
df_returns_total = df_returns_total.reset_index()

# Переименуем названия столбцов
df_returns_total.columns = ['funding_type', 'total_sum_returns']

# Сортируем по убыванию
df_returns_total = df_returns_total.sort_values('total_sum_returns', ascending=False).reset_index(drop=True)

# Выведем результат с форматированием
display(df_returns_total.style.format({'total_sum_returns': '{:,.0f} млн $'}))
```

	funding_type	total_sum_returns
0	venture	40,579 млн $
1	debt_financing	4,735 млн $
2	private_equity	3,587 млн $
3	seed	2,382 млн $
4	angel	1,509 млн $
5	post_ipo_equity	1,105 млн $
6	undisclosed	731 млн $
7	post_ipo_debt	91 млн $
8	convertible_note	35 млн $
9	secondary_market	5 млн $
10	equity_crowdfunding	4 млн $
11	product_crowdfunding	2 млн $
12	grant	0 млн $

In [46]:
```
# Строим стобчатую диаграмму
plt.figure(figsize=(10, 5))

ax = sns.barplot(
    data=df_returns_total,
    y='funding_type',
    x='total_sum_returns',
    color='steelblue'
)

# Настраиваем оформление
plt.title('Объем возвратов по типам финансирования')
plt.xlabel('Сумма возврата (млн $)')
plt.ylabel('')

# Добавляем подписи к столбцам
for idx, value in enumerate(df_returns_total['total_sum_returns']):
    ax.text(value + 0.5, idx, f'{value:.0f} млн $', ha='left', va='center')

# Выводим график
plt.tight_layout()
plt.show()
```

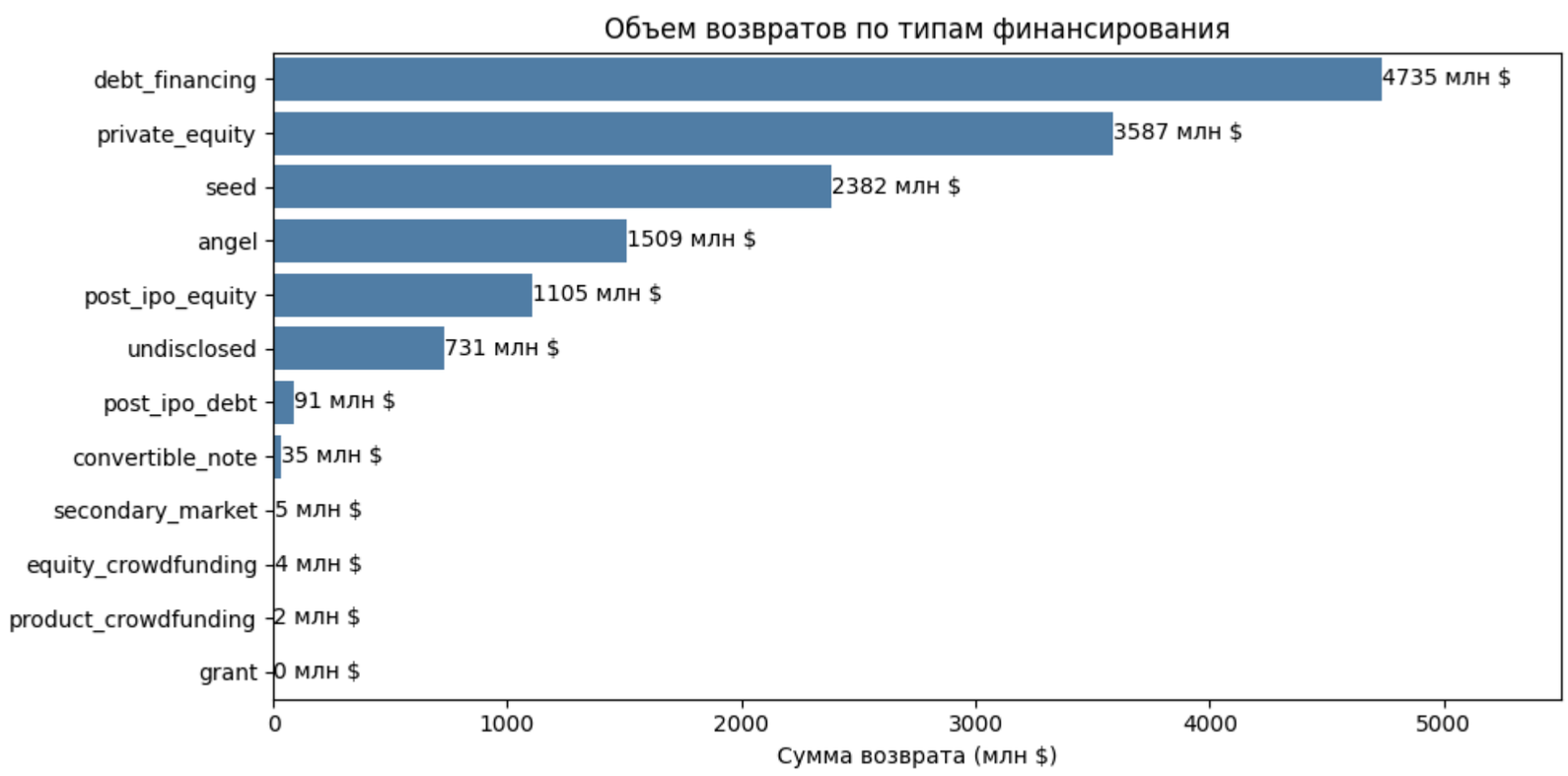


Вывод:

- Сумма возврата по венчурным инвестициям также превышает вместе взятые остальные типы финансирования.

Построим график без венчурных инвестиций.

In [47]:
```
# Строим стобчатую диаграмму
plt.figure(figsize=(10, 5))

ax = sns.barplot(
    data=df_returns_total[df_returns_total['funding_type'] != 'venture'],
    y='funding_type',
    x='total_sum_returns',
    color='steelblue'
)

# Настраиваем оформление
plt.title('Объем возвратов по типам финансирования')
plt.xlabel('Сумма возврата (млн $)')
plt.ylabel('')
plt.xlim(0, 5500)

# Добавляем подписи к столбцам
filtered_data = df_returns_total[df_returns_total['funding_type'] != 'venture']
for idx, value in enumerate(filtered_data['total_sum_returns']):
    ax.text(value + 0.5, idx, f'{value:.0f} млн $', ha='left', va='center')

# Выводим график
plt.tight_layout()
plt.show()
```



`venture` лидирует по объёму возвращённых средств, а вот находившийся в тройке лидеров и по объёму, и по количеству предоставленных средств `seed` перешёл на четвёртое место по возвратам.

Шаг 4. Анализ динамики

Динамика предоставления финансирования по годам

Строим графики в этом задании и следующих, используя данные только по тем компаниям, которые остались в датасете после предыдущих фильтраций.

Используя столбцы `funding_total_usd` и `funding_rounds`, рассчитаем для каждой компании средний объём одного раунда финансирования.

In [48]:
```
# Добавим новый столбец со средним объемом одного раунда
df_filtered['avg_funding_round'] = df_filtered['funding_total_usd'] / df_filtered['funding_rounds']
# Выведем
df_filtered[['funding_total_usd', 'funding_rounds', 'avg_funding_round']].head()
```

Out[48]:

	funding_total_usd	funding_rounds	avg_funding_round
1	2.00	1.0	2.000000
2	9.00	1.0	9.000000
3	7.70	3.0	2.566667
4	0.54	1.0	0.540000
7	8.70	1.0	8.700000

Построим график, отражающий динамику типичного размера средств, которые стартапы получали в рамках одного раунда финансирования.

In [49]:
```
# Сгруппируем по году mid_funding_at_year и найдем среднее финансирование одного раунда финансирования
df_agg = df_filtered.groupby('mid_funding_at_year')['avg_funding_round'].mean()
df_agg = df_agg.reset_index()
df_agg.columns = ['mid_funding_at_year', 'avg_funding_round']

# Размер графика
plt.figure(figsize=(8, 4))

# Строим диаграмму
ax = sns.lineplot(
    data=df_agg,
    x='mid_funding_at_year',
    y='avg_funding_round',
    marker='o',
    color='steelblue',
)
# Оформление
plt.ylabel('Среднее финансирование 1 раунда (млн $)')
plt.xlabel('')
plt.title('Динамика среднего финансирования одного раунда')
plt.ylim(0, 6.5)
plt.xticks(range(2000, 2015, 1))

# Добавляем подписи к каждой точке
for x, y in zip(df_agg['mid_funding_at_year'], df_agg['avg_funding_round']):
    plt.annotate(
        f'{y:.1f}',           # текст подписи
        xy=(x, y),          # координаты точки
        xytext=(0, 6),     # отступ от точки (8 пунктов вверх)
        textcoords='offset points',
        ha='center', # горизонтальное выравнивание: по центру точки
)

# Отображаем график
plt.tight_layout()
plt.show()
```

Построим график, отражающий динамику общего количества раундов за каждый год, то есть насколько активно происходили инвестиции на рынке (чем больше раундов, тем выше активность).

In [50]:
```
# Сгруппируем по году mid_funding_at_year и найдем среднее финансирование одного раунда финансирования
df_agg = df_filtered.groupby('mid_funding_at_year')['funding_rounds'].sum()
df_agg = df_agg.reset_index()
df_agg.columns = ['mid_funding_at_year', 'funding_rounds']

# Размер графика
plt.figure(figsize=(8, 4))

# Строим диаграмму
ax = sns.lineplot(
    data=df_agg,
    x='mid_funding_at_year',
    y='funding_rounds',
    marker='o',
    color='steelblue',
)
# Оформление
plt.ylabel('Общее количество раундов финансирования')
plt.xlabel('')
plt.title('Динамика общего количества раундов финансирования')
plt.ylim(0, 14000)

plt.xticks(range(2000, 2015, 1))

# Добавляем подписи к каждой точке
for x, y in zip(df_agg['mid_funding_at_year'], df_agg['funding_rounds']):
    plt.annotate(
        f'{y:.0f}',           # текст подписи
        xy=(x, y),          # координаты точки
        xytext=(0, 6),     # отступ от точки (8 пунктов вверх)
        textcoords='offset points',
        ha='center', # горизонтальное выравнивание: по центру точки
)

# Отображаем график
plt.tight_layout()
plt.show()
```

Выводы:

- В 2005 году размер средств, собранных в рамках одного раунда, был максимальным (5.8 млн USD). После 2005 года размер средств на один раунд уменьшался, хотя количество раундов увеличивалось.
- В 2014 году размер средств на один раунд (2.2 млн USD) перестал уменьшаться после минимальных значений в 2012-2013 гг. (2.0 млн USD). При этом общее количество раундов финансирования с максимального количества 12 880 в 2013 г. резко уменьшилось до 7 124 в 2014 г. Но надо учитывать, что данные за декабрь 2014 года у нас неполные.
- Тенденция по типичному размеру финансирования в 2014 году - сохраняется на уровне двух предыдущих лет, но не хватает данных, чтобы утверждать, что закрепилась тенденция на увеличение среднего размера средств.

Динамика размера общего финансирования по массовым сегментам рынка для растущих в 2014 году сегментов

Составим сводную таблицу, в которой указывается суммарный размер общего финансирования `funding_total_usd` по годам и сегментам рынка. Отберем из неё только те сегменты, которые показывали рост размера суммарного финансирования в 2014 году по сравнению с 2013.

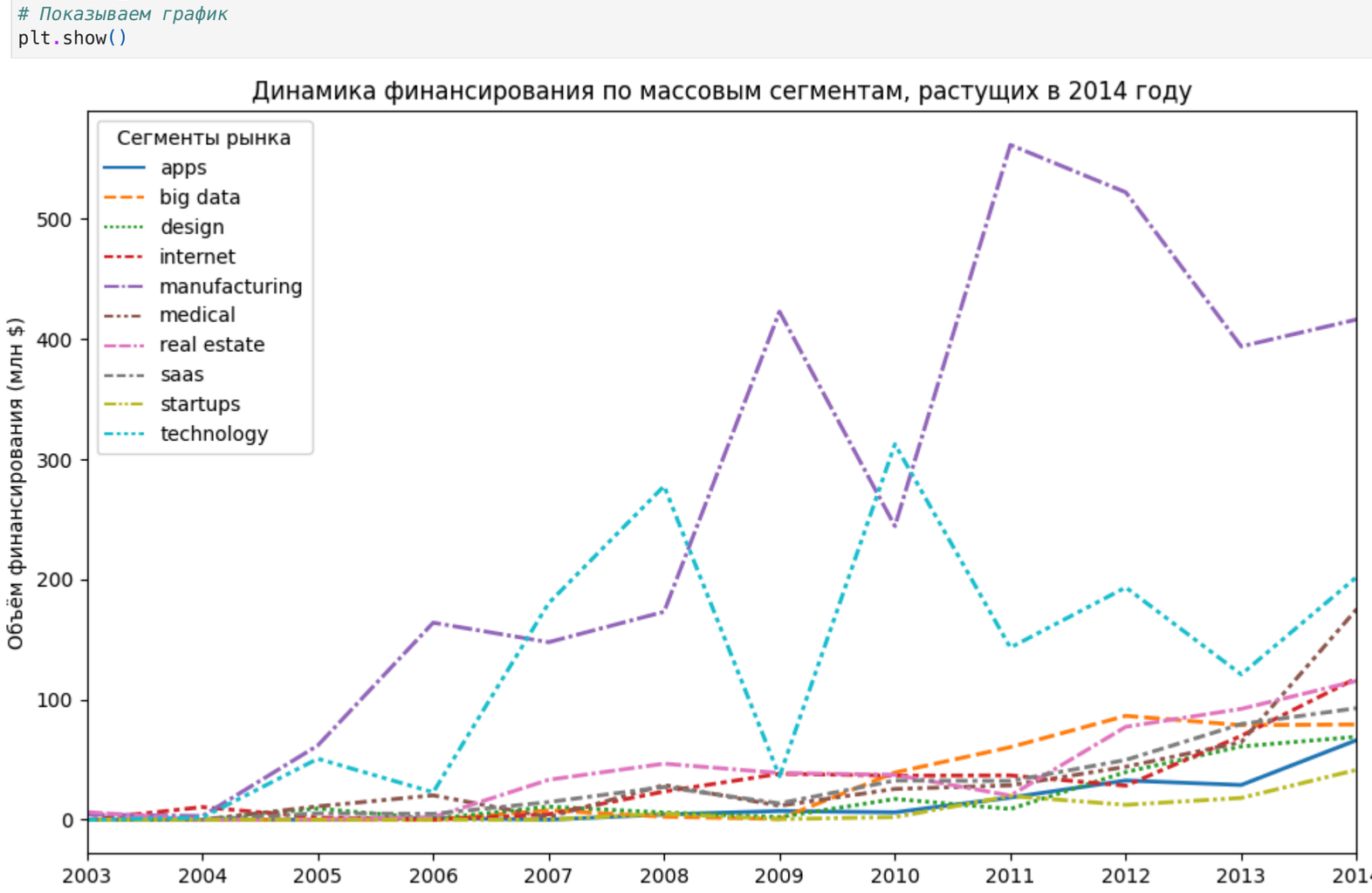
На графике отразим, как менялся суммарный размер общего финансирования в каждом из отобранных сегментов по годам, за которые у нас достаточно данных. Рассматриваем только массовые сегменты, а средние и нишевые исключаем. Также с неизвестным сегментом тоже исключаем.

In [51]:
```
# Найдем сумму общего финансирования отдельно по сегментам
df_market_table = df_filtered.pivot_table(
    index='market',
    columns='mid_funding_at_year',
    values='funding_total_usd',
    aggfunc='sum',
    fill_value=0
).drop(['mid', 'niche', 'unknown']) # исключаем средние и нишевые, а также неизвестные

# Оставим только те сегменты, где был рост в 2014 году относительно 2013 года
df_market_table = df_market_table[df_market_table[2014] > df_market_table[2013]]

# Выведем подходящие сегменты
df_market_table.index
```

Out[51]:
```
Index(['apps', 'big data', 'design', 'internet', 'manufacturing', 'medical',
       'real estate', 'saas', 'startups', 'technology'],
      dtype='object', name='market')
```

Сегменты, показавшие рост суммарного финансирования в 2014 году относительно 2013: `apps`, `big data`, `design`, `internet`, `manufacturing`, `medical`, `real estate`, `saas`, `startups`, `technology`.

In [52]:
```
# Строим линейный график
plt.figure(figsize=(10, 6))
ax = sns.lineplot(
    # Трнаспонируем таблицу для построения графиков
    data=df_market_table.T,
    linewidth=2,
)

# Оформление
plt.title('Динамика финансирования по массовым сегментам, растущих в 2014 году')
plt.ylabel('Объём финансирования (млн $)')
plt.xlabel('')
plt.xlim(2003, 2014) # Оставим данные с 2003 года, т. к. в 2001 и 2002 нет данных
plt.xticks(range(2003, 2015, 1))
plt.legend(title='Сегменты рынка', loc='upper left')

# Оптимизируем макет
plt.tight_layout()
# Показываем график
plt.show()
```



Слишком много сегментов, оставим только 6 сегментов, у которых самый большой относительный прирост объема финансирования в 2014 году относительно 2013 года.

In [53]:
```
# Ищем разницу между 2014 годом и 2013 относительно 2013 года
df_top_segment = (df_market_table[2014] - df_market_table[2013]) / df_market_table[2013]
# Оставим топ-6 компаний
df_top_segment = df_top_segment.sort_values(ascending=False).head(6)
# Результат
df_top_segment
```

Out[53]:
```
market
medical        1.718127
startups       1.295386
apps           1.294454
internet       0.689764
technology     0.671320
real estate    0.253338
dtype: float64
```

In [54]:
```
# Строим линейный график
plt.figure(figsize=(8, 5))
ax = sns.lineplot(
    # Трнаспонируем таблицу для построения графиков
    data=df_market_table.loc[df_top_segment.index].T,
    linewidth=2,
)

# Оформление
```

```
plt.title('Динамика финансирования по массовым сегментам, растущих в 2014 году')
plt.ylabel('Объём финансирования (млн $)')
plt.xlabel('')
plt.xlim(2003, 2014) # Оставим данные с 2003 года, т. к. в 2001 и 2002 нет данных
plt.xticks(range(2003, 2015, 1))
plt.legend(title='Сегменты рынка', loc='upper left')

# Оптимизируем макет
plt.tight_layout()
# Показываем график
plt.show()
```

На основе графика сделайте вывод о том, какие сегменты показывают наиболее быстрый и уверенный рост. Выводы:

- Наиболее быстрый и уверенный рост в последние годы демонстрируют: `medical`, `internet`, `apps`, `real estate`.
- Стоит дополнительно отметить, что для технологического сегмента `technology` характерны самые большие объемы инвестиций, но и динамика волнообразная с самым большим размахом и периодическими резкими падениями.
- Видно смещение приоритетов инвесторов в сторону цифровых технологий и анализа данных.

Годовая динамика доли возвращённых средств по типам финансирования

Заказчик хочет знать, какая часть вложенных или выданных денег со временем возвращается обратно инвесторам или финансистам.

Для каждого года и каждого вида финансирования рассчитаем нормированные значения возврата средств: какую долю возвращённые средства составляют от предоставленных. При этом слишком большие аномальные значения, то есть неадекватные выбросы, нужно заменить на пропуски.

In [55]:
```
# Найдем сумму финансирования отдельно по типам финансирования
df_funding_table = df_filtered.pivot_table(
    index='mid_funding_at_year',
    values=funding_columns_list,
    aggfunc='sum',
    fill_value=0
)
# Название идекса поменяем
df_funding_table.index.name = 'year'
# Проверим, что получилось
df_funding_table
```

Out[55]:

year	angel	convertible_note	debt_financing	equity_crowdfunding	grant	post_ipo_debt	post_ipo_equity	private_equity	product_crowdfunding	secondary_market	seed	undis
2000	24.086333	0.000000	14.000000	0.000000	0.293114	0.000000	3.467747	0.000000	0.000000	7.718867	16.759140	112.2
2003	5.629661	0.000000	1.050000	0.000000	16.850717	0.000000	0.000000	0.000000	0.000000	0.000000	15.318355	10.2
2004	11.013741	0.000000	30.816623	0.000000	10.363600	0.000000	0.000000	0.000000	0.000000	0.000000	18.104728	62.5
2005	60.914621	0.000000	101.720656	0.000000	6.266481	0.000000	4.796022	5.000000	0.000000	0.000000	39.425198	8.
2006	70.756153	10.702385	140.848395	0.933057	6.147500	0.000000	0.000000	17.820244	0.000000	0.000000	66.794774	61.5
2007	201.152441	14.116788	208.601638	0.000000	34.237779	0.000000	12.000000	153.370229	0.000000	0.000000	192.296481	109.6
2008	249.299613	28.600902	428.622061	0.000000	23.485347	0.000000	36.000000	191.756331	1.000000	0.000000	302.003020	119.9
2009	157.822426	41.257969	1015.071410	2.049671	300.835487	0.000000	36.628903	258.118979	0.303050	19.300000	284.351394	99.6
2010	245.725462	46.247821	1131.602620	7.400000	226.905678	0.000000	85.510500	524.927867	3.500000	0.910000	503.808937	210.4
2011	393.511877	78.613209	1144.121986	3.091684	193.791434	3.100000	193.398404	910.997761	18.142441	12.500000	1166.553328	259.1
2012	385.383621	105.770341	1375.685503	15.068930	154.094626	77.500000	199.703350	821.143538	13.136864	0.000000	1916.136879	449.9
2013	417.044638	122.238088	1745.256931	46.383011	534.066826	44.800000	392.012155	1605.596587	82.289662	0.000000	2994.847714	311.5
2014	254.923249	116.991933	819.616023	163.034292	471.410099	161.318349	982.634987	346.876968	67.721903	4.856929	1903.799080	204.9

Удалим 2000 год в `df_funding_table`, т. к. нет данных за 2001 и 2002 года. Также удалим 2000, 2001 и 2002 года в `df_returns`.

In [56]:
```
# Удалим 2000 год в df_funding_table
df_funding_table = df_funding_table.drop(2000)

# Удалим 2000-2002 гг в df_returns
df_returns = df_returns.drop([2000, 2001, 2002])
```

In [57]:
```
# Посмотрим на возвраты
df_returns
```

Out[57]:

year	seed	venture	equity_crowdfunding	undisclosed	convertible_note	debt_financing	angel	grant	private_equity	post_ipo_equity	post_ipo_debt	secondary_market	product_crowdfunding
2003	7.74	233.86	0.00	9.40	0.01	1.09	3.41	0.0	1.62	2.11	0.00	0.08	0.00
2004	9.93	555.90	0.00	33.19	0.01	13.55	9.18	0.0	2.19	3.38	0.00	0.55	0.00
2005	26.60	2628.92	0.00	9.51	0.02	35.09	31.06	0.0	2.40	3.51	0.00	0.05	0.00
2006	61.81	3100.18	0.19	46.74	1.78	113.21	47.75	0.0	16.67	20.58	0.00	0.12	0.00
2007	70.41	3585.37	0.01	55.37	3.22	125.68	164.51	0.0	88.81	24.36	0.00	0.57	0.00
2008	89.72	2717.02	0.03	41.02	1.71	397.54	102.83	0.0	130.38	84.28	0.00	0.47	0.00
2009	160.21	2501.29	0.18	37.50	2.25	394.10	97.21	0.0	203.70	76.76	0.00	0.12	0.02
2010	265.33	3299.35	0.39	52.15	3.14	503.73	157.21	0.0	339.08	115.72	0.00	0.18	0.24
2011	317.29	3919.38	0.30	59.36	3.96	572.92	172.17	0.0	666.03	129.01	3.04	0.16	0.26
2012	373.31	4727.74	0.44	71.04	4.79	694.97	193.03	0.0	762.40	160.88	20.34	0.34	0.28
2013	442.97	5749.29	1.14	84.40	6.00	848.65	225.87	0.0	988.53	219.93	30.45	0.81	0.34
2014	530.75	7272.01	1.15	105.72	7.87	1017.75	293.98	0.0	384.01	262.70	37.20	1.03	0.72

Разделим один датафрейм на другой, чтобы найти долю возвращенных средств. Долю выразим в процентах. При этом слишком большие аномальные значения, то есть неадекватные выбросы, нужно заменить на пропуски.

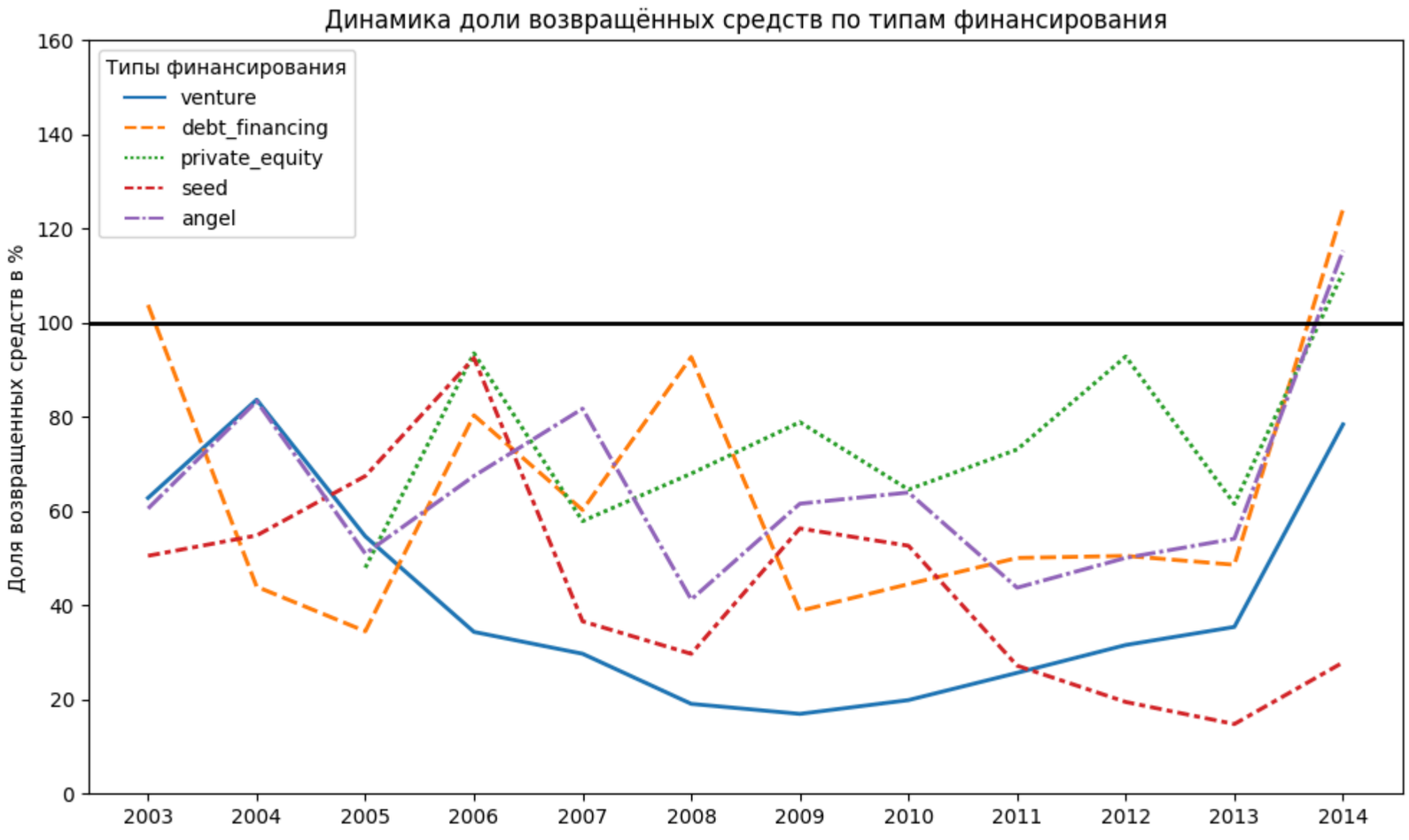
In [58]:
```
# Находим долю возврата отдельно по каждому году и типу финансирование в процентах
df_funding_re_table = df_returns / (df_funding_table + 1e-60) * 100
# Слишком большие аномальные значения более 10 000%, заменяем на пропуски
df_funding_re_table = df_funding_re_table.where(df_funding_re_table < 10000, float('nan'))
# Вывод
df_funding_re_table
```

Out[58]:

year	angel	convertible_note	debt_financing	equity_crowdfunding	grant	post_ipo_debt	post_ipo_equity	private_equity	product_crowdfunding	secondary_market	seed	undisclosed
2003	60.572031	NaN	103.809524	0.000000	0.0	0.000000	NaN	NaN	0.000000	NaN	50.527619	91.439689
2004	83.350426	NaN	43.969776	0.000000	0.0	0.000000	NaN	NaN	0.000000	NaN	54.847551	52.755930
2005	50.989400	NaN	34.496435	0.000000	0.0	0.000000	73.185653	48.000000	0.000000	NaN	67.469541	107.199235
2006	67.485297	16.631807	80.377203	20.363172	0.0	0.000000	NaN	93.545296	0.000000	NaN	92.537180	75.943816
2007	81.783745	22.809721	60.248808	NaN	0.0	0.000000	203.000000	57.905632	0.000000	NaN	36.615335	50.497081
2008	41.247557	5.978832	92.748376	NaN	0.0	0.000000	234.111111	67.992540	0.000000	NaN	29.708312	34.206041
2009	61.594542	5.453492	38.824855	8.781897	0.0	0.000000	209.561286	78.917095	6.599571	0.621762	56.342259	37.617657
2010	63.977904	6.789509	44.514743	5.270270	0.0	0.000000	135.328410	64.595542	6.857143	19.780220	52.664806	24.785076
2011	43.752174	5.037321	50.075080	9.703450	0.0	98.064516	66.706859	73.109949	1.433104	1.280000	27.198928	22.906284
2012	50.087754	4.528680	50.518087	2.919915	0.0	26.245161	80.559490	92.846130	2.131407	NaN	19.482429	15.787913
2013	54.159670	4.908454	48.626078	2.457796	0.0	67.968750	56.102852	61.567769	0.413175	NaN	14.791069	27.056680
2014	115.320984	6.726960	124.174000	0.705373	0.0	23.059993	26.734240	110.704957	1.063172	21.206816	27.878467	51.593768

Построим график, на котором отобразим нормированные значения возврата средств для типов финансирования `venture`, `debt_financing`, `private_equity`, `seed` и `angel`.

In [59]:
```
# Строим линейный график
plt.figure(figsize=(10, 6))

ax = sns.lineplot(
    data=df_funding_re_table[['venture', 'debt_financing', 'private_equity', 'seed', 'angel']],
    linewidth=2,
)
# Горизонтальная линия на уровне 100%
plt.axhline(y=100, color='k', linestyle='-', linewidth=2)

# Оформление
plt.title('Динамика доли возвращённых средств по типам финансирования')
plt.ylabel('Доля возвращенных средств в %')
plt.xlabel('')
plt.ylim(0, 160)
plt.xticks(range(2003, 2015, 1))
plt.legend(title='Типы финансирования', loc='upper left')

# Оптимизируем макет
plt.tight_layout()
# Показываем график
plt.show()
```



Выводы:

- `venture` демонстрирует стабильное и предсказуемое поведение: сначала медленное снижение, затем медленный рост, который по-видимому еще сохранится, и в перспективе превысит 100%.
- `private_equity`, `debt_financing` и `angel` демонстрируют резкие колебания, поэтому последний период роста может смениться резким падением.
- `seed` демонстрирует самую плохую динамику.

Шаг 5. Итоговый вывод и рекомендации

Итоги проекта

Выполненные шаги

В ходе исследования были последовательно выполнены следующие этапы работы.

Загрузка и предобработка данных:

- Импортированы основные датасеты (`cb_investments.zip` и `cb_returns.csv`).
- Приведены к единому формату названия столбцов, устранены лишние пробелы и текстовые поля преобразованы в нижний регистр.
- Обработаны типы данных: столбцы с датами переведены в `datetime`, `funding_total_usd` — в числовой формат.
- Заполнены пропуски в текстовых столбцах значением `unknown`.
- Удалены полные дубликаты и строки без данных о финансировании.
- Восстановлены пропуски в `mid_funding_at` на основе `first_funding_at` и `last_funding_at`.

Инжиниринг признаков:

- Созданы группы по срокам финансирования (`единичное`, `до года`, `более года`).
- Сегменты рынка классифицированы на `массовые`, `средние` и `нишевые` по количеству компаний.
- Для анализа оставлены только массовые сегменты, остальные объединены в `mid` и `niche`.

Работа с выбросами и анализ:

- Визуализировано распределение финансирования с помощью гистограмм и диаграммы размаха.
- Определены аномальные значения финансирования методом IQR по сегментам.
- Исключены компании с аномальным финансированием (12,8 % от общего числа).
- Проанализированы типы финансирования по объёму, популярности и среднему чеку.

Анализ динамики:

- Исследована динамика среднего размера раунда и общего количества раундов по годам.
- выделены массовые сегменты с ростом финансирования в 2014 г.
- Рассчитана и визуализирована доля возвращённых средств по типам финансирования.

Итоги проекта:

- Выполненные шаги
- какие выводы удалось сделать;
- насколько выводы согласуются между собой или, наоборот, вызывают сомнения.

Представьте, что на календаре 2015 год. Опираясь на результаты анализа, дайте рекомендацию заказчику:

- в какую отрасль стоит инвестировать;
- какой тип финансирования при этом будет наиболее уместным.

Основные выводы

Распределение финансирования:

- Большинство сегментов — нишевые (до 35 компаний), но массовые сегменты аккумулируют наибольшую долю объема инвестиций.
- Медиана общего финансирования — около 2 млн USD, типичные значения — до 24,5 млн USD.
- Венчурные инвестиции доминируют как по объёму финансирования, так и по объему возвратов.

Типы финансирования:

- `venture` — самый объёмный и стабильный источник финансирования.
- `seed`, `angel`, `equity_crowdfunding` — частые, но с небольшими объемами инвестиций.
- `private_equity`, `post_ipo_equity`, `post_ipo_debt` — редкие, но с большими объемами инвестиций.

Динамика по годам:

- Пик среднего объема финансирования одного раунда — в 2005 г. (5,8 млн USD), затем постепенное снижение до 2,2 USD.
- Максимальное количество раундов — в 2013 г. (12 880), в 2014 г. резкое падение (до 7 124), что может быть связано с неполнотой данных за декабрь 2014 г.
- В 2014 г. рост финансирования показали сегменты: `medical`, `internet`, `apps`, `real estate`, `technology`.

Возвраты средств:

- Венчурные инвестиции демонстрируют стабильную динамику возврата, приближающуюся к 100 %.
- `private_equity`, `debt_financing`, `angel` — высокая волатильность, риски резких падений.
- `seed` — наихудшая динамика возвратов.

Согласованность выводов

Выводы в целом согласуются:

- Венчурные инвестиции — ключевой драйвер рынка, сочетающий масштаб и относительную стабильность.
- Массовые сегменты (`medical`, `internet`, `apps`, `real estate`) показывают устойчивый рост, что подтверждает их инвестиционную привлекательность.
- Аномалии в финансировании (12,8 %) не искажают общую картину, так как были корректно исключены.

Сомнения вызывает:

- Неполнота данных за декабрь 2014 г., что может повлиять на оценку динамики 2014 г.
- Высокая волатильность возвратов по некоторым типам финансирования (`private_equity`, `debt_financing`), что усложняет прогнозирование.

Рекомендации на 2015 год

Отрасль для инвестирования

Приоритетные сегменты с учётом роста и объёма финансирования:

- `Medical` — стабильный рост, высокий потенциал инноваций.
- `Internet` — масштабируемость, устойчивая динамика.
- `Apps` — быстрый рост, низкая стоимость входа.
- `Real estate` — рост в 2014 г., потенциал монетизации.

Данные сегменты показали увеличение объёма финансирования в 2014 г. и обладают достаточной ликвидностью для инвесторов.

Тип финансирования

Оптимальный вариант — **венчурные инвестиции (`venture`)**:

- Крупнейший объём привлечённых средств.
- Относительно стабильная динамика возвратов (в перспективе приближается к 100%).
- Подходит для масштабируемых проектов в выбранных сегментах.
- Баланс между риском и доходностью.

Итог

В 2015 г. целесообразно фокусироваться на **венчурных инвестициях** в сегментах `medical` , `internet` , `apps` и `real estate` . Это сочетание обеспечивает баланс роста, ликвидности и управляемый риск.