Yazılım Tedarik Zincirinde Kritiklik Haritalaması: En Çok İndirilen 1000 NPM Paketinin Bağımlılıklarının Topolojik Risk Değerlendirmesi

Yusuf Talha ARABACI
Karabük Üniversitesi
Yüksek Lisans, Yazılım Mühendisliği Öğrencisi

Ekim 2025

Özet

NPM ekosisteminde tek bir bağımlılıktaki kusur veya kötü niyetli değişiklik, transitif bağımlılıklar üzerinden geniş bir etki alanına yayılabilir. Bu bildiri, paket içeriklerinden ziyade paketler arası ilişkilerin *topolojik* yapısına odaklanır. Bağımlı → bağımlılık yönünde kurulan yönlü ağ üzerinde in-degree, out-degree ve betweenness merkeziyetleri hesaplanır; bu ölçüler min–max normalizasyonu ile bir *bileşik risk skoru*na dönüştürülür. Ayrıca, en kritik düğümlerin çıkarılmasıyla ağın bağlanırlığı üzerinden bir *sağlamlık* değerlendirmesi yapılır. Tüm görseller ve tablolar results/ dizinindeki çıktılara dayanır ve ilgili başlıklar altında sunulur.

1 Giriş

Yazılım tedarik zincirinde tek bir bağımlılıktaki hata ya da kasıtlı kötü niyetli değişiklik, transitif bağımlılıklar üzerinden yüzlerce hatta binlerce projeye yayılabilir. NPM ekosistemi; ölçek, sürüm sıklığı ve yoğun bağımlılık grafiği nedeniyle bu tür zincirleme risklere özellikle açıktır. Bu çalışma, paket içeriğinden ziyade paketler arası ilişkinin *topolojik* yapısına odaklanır: Bir paketin ağ içindeki konumu ve bu konumun sistemik etkileri nicel olarak değerlendirilir. Amaç, topolojik ölçütleri tek bir bileşik skorda birleştirerek kritik paketleri önceliklendirmek ve ağ düzeyinde sağlamlık duyarlılıklarını görünür kılmaktır.
Projenin kod ve özet sonuçları: `https://yusufarbc.github.io/npm-complex-network-analysis/`.

2 Yöntem

2.1 Çalışma Tasarımı ve Parametreler

Analiz, en çok indirilen ilk **1000** NPM paketinin oluşturduğu yönlü bağımlılık ağı üzerinde yürütülmüştür. Varsayılan ayarlar: betweenness için örnekleme ($k \approx 200$), yalnızca `dependencies` alanı (isteğe bağlı `peerDependencies` dahil edilebilir), HTTP önbelleği ve tekrar denemeleri etkindir. Kenarlar *bağımlı* $\rightarrow$ *bağımlılık* yönündedir; graf yönlüdür (DiGraph).

2.2 Veri Kaynakları ve Ön İşleme

- **Top N paket listesi:** Öncelik `ecosyste.ms` API'sinde (indirmeye göre), yedek olarak `npm registry search` ve `npms.io` kullanılır.
- **Paket adı kodlama:** Scoped paketler (`@scope/name`) için URL-güvenli kodlama uygulanır.
- **Sürüm seçimi:** `dist-tags.latest` mevcutsa o sürüm, değilse en yüksek sürüm anahtarı seçilir.
- **Önbellek ve tekrar:** Bağımlılık sorguları JSON dosyada önbelleklenir (`results/cache_deps.json`); başarısız istekler en fazla üç kez tekrar edilir.

2.3 Ağ Kurulumu

- Düğümler paketleri, kenarlar *Dependent* $\rightarrow$ *Dependency* yönünü temsil eder; self-loop yoktur, grafik yönlüdür.
- Paketlerin en güncel sürümlerinden `dependencies` alanı okunur; isteğe bağlı olarak `peerDependencies` dahil edilebilir.
- HTTP oturumu yeniden kullanılır; istekler basit tekrar (retry) ve disk önbelleği ile dayanıklı hale getirilir.

2.4 Metrikler

- **in-degree**: Bir pakete bağımlı paket sayısı — çekirdek/merkez cazibeyi yansıtır.
- **out-degree**: Bir paketin bağlı olduğu bağımlılık sayısı — kırılgan yüzeyin genişliğini yansıtır.
- **betweenness centrality**: Akışın aralarından geçtiği köprü konumlar. Büyük graflarda örnekleme (`sample-k`) ile hızlandırılır.

2.5 Normalizasyon ve Bileşik Risk

Metrikler min–max ile [0,1] aralığına ölçeklenir: $x' = (x - \min)/(\max - \min)$. Bileşik risk skoru:

$$\text{risk} = w_{in}\, in' + w_{out}\, out' + w_{btw}\, btw' \quad \text{(varsayılan: 0.5, 0.2, 0.3)}$$

Ağırlıklar komut satırı seçenekleri ile değiştirilebilir. Betweenness büyük graflarda örnekleme (k) ile hesaplanır; tipik olarak $k = 200$ (graf boyutuna göre ayarlanır).

2.6 Kaskad Etkisi (Basamaklanma) ve Sağlamlık

Kaskad etkisi. Kenarlar *Dependent* → *Dependency* yönündeyken, bir bağımlılığın ele geçirilmesi onu *kullanan* paketleri etkiler. Bu nedenle grafın tersinde (G^{rev}) bir kaynaktan erişilebilen düğüm sayısı, o paketin potansiyel *etki alanı*dır. Algoritma BFS/DFS ile $O(|V| + |E|)$: `G_rev = reverse(G); reach = BFS(G_rev, s)`.

Sağlamlık. Seçili düğümler kaldırıldıktan sonra zayıf bağlı bileşen sayısı, en büyük bileşen boyutu ve (mümkünse) LCC çapı raporlanır. Bu ölçüler, kritik düğümlerin ağ bütünlüğüne etkisini gösterir.

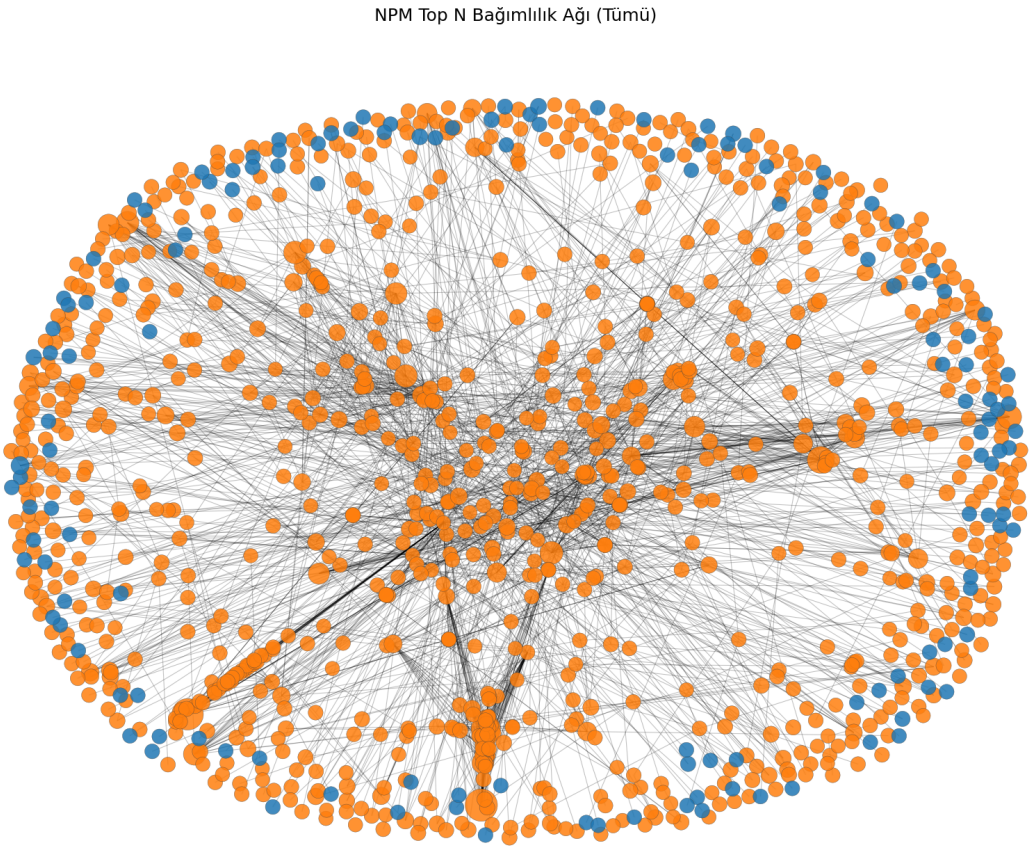
2.7 Köprü Kenarlar ve Diğer Çıktılar

Kenar betweenness. Yüksek değere sahip kenarlar alt-ağlar arasında köprü görevi görür; kopmaları parçalanmayı hızlandırır. Ayrıca `edges.csv`, `metrics.csv`, `risk_scores.csv` ve `graph_stats.json` üretilir.

3 Bulgular ve Yorum

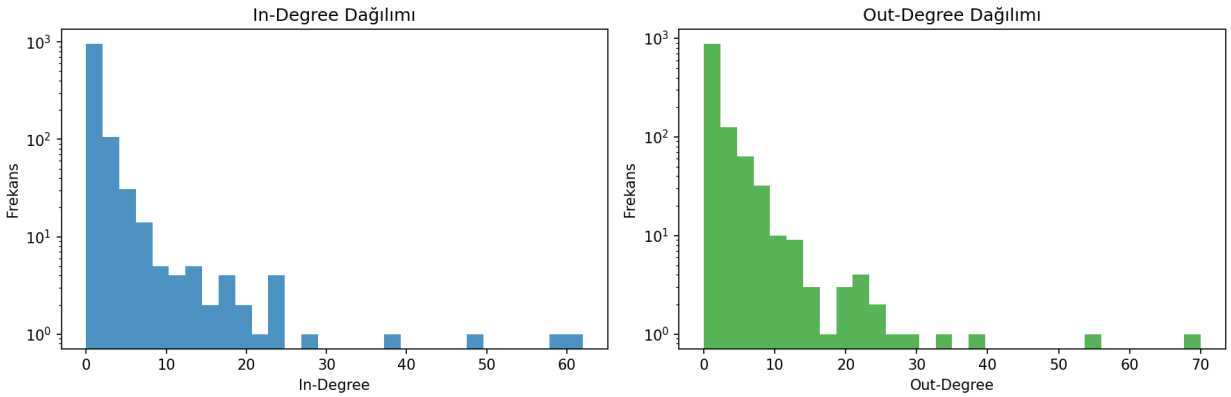
Bu bölümde bulgular görseller ve tablolarla birlikte, her birinin *yorum* kısmıyla sunulmaktadır. Tüm materyal `results/` klasöründen alınmıştır.

3.1 Ağ ve Derece Dağılımları



Şekil 1: En çok indirilen paketlerle kurulan tam ağın görselleştirmesi. Yoğun merkezî bölgeler omurga paket kümelerini, seyrek bölgeler çevreyi işaret etmektedir.

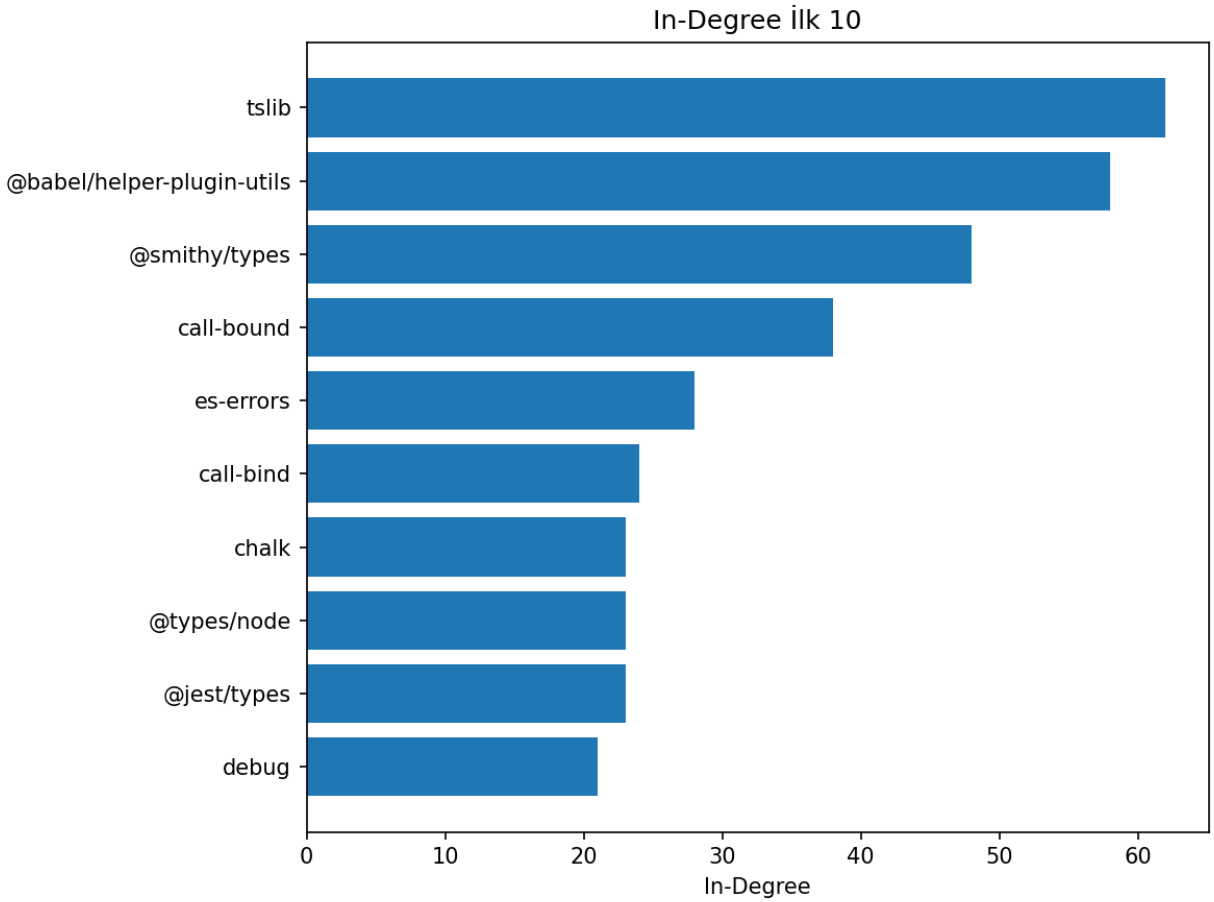
Yorum — Ağ görünümü, birkaç yüksek dereceli çekirdek etrafında yoğunlaşan küçük-dünya benzeri bir yapı sergiler. Bu, az sayıda paket üzerinde orantısız etki birikimine işaret eder.



Şekil 2: In-degree ve out-degree dağılımlarının histogramları. Her iki dağılım da ağır kuyruklu bir profil göstermektedir.

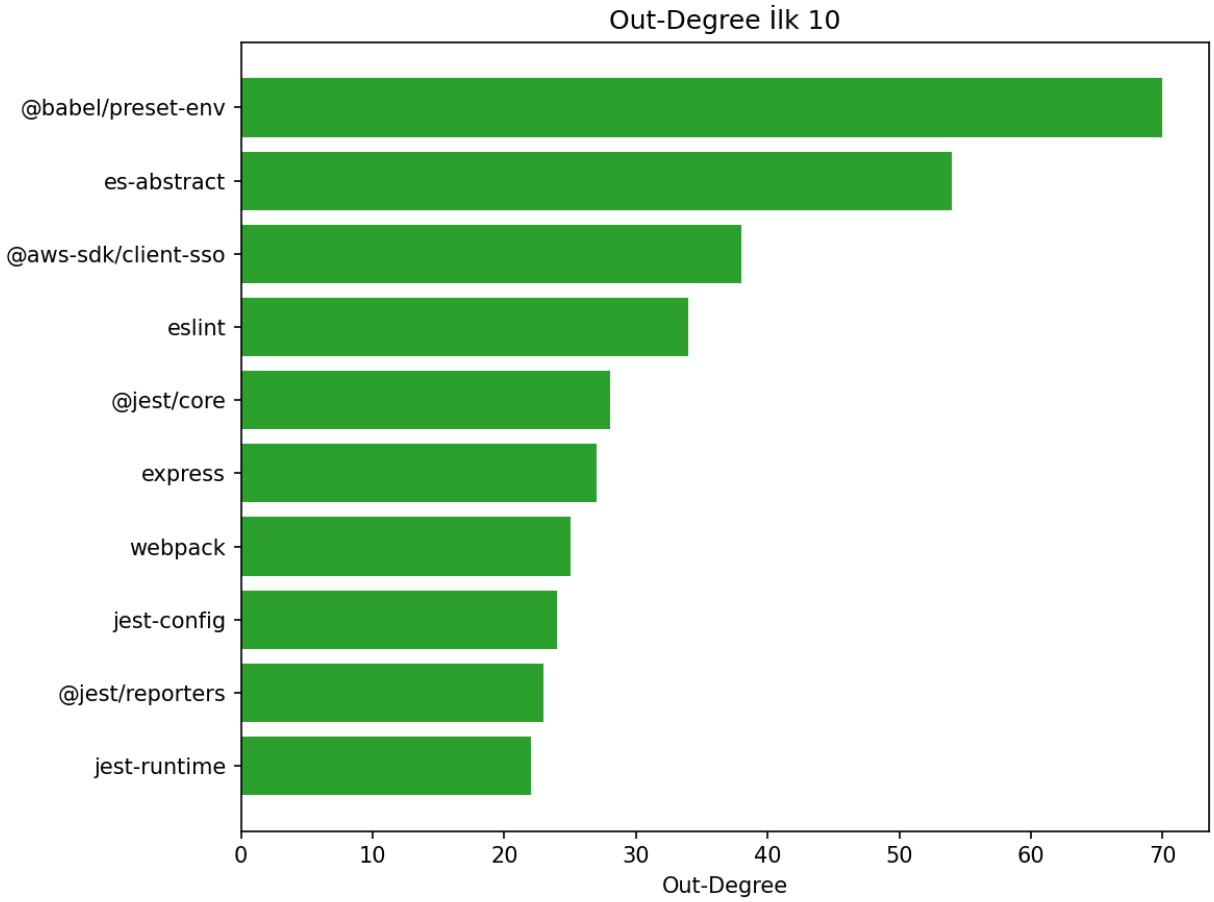
Yorum — Ağır kuyruk, çok az sayıda paketin çok yüksek dereceye sahip olduğunu, büyük çoğunluğun ise düşük derecelerde kaldığını gösterir. Bu, sistemik riskte eşitsizliği artırır.

3.2 Öncü Paketler ve Korelasyonlar



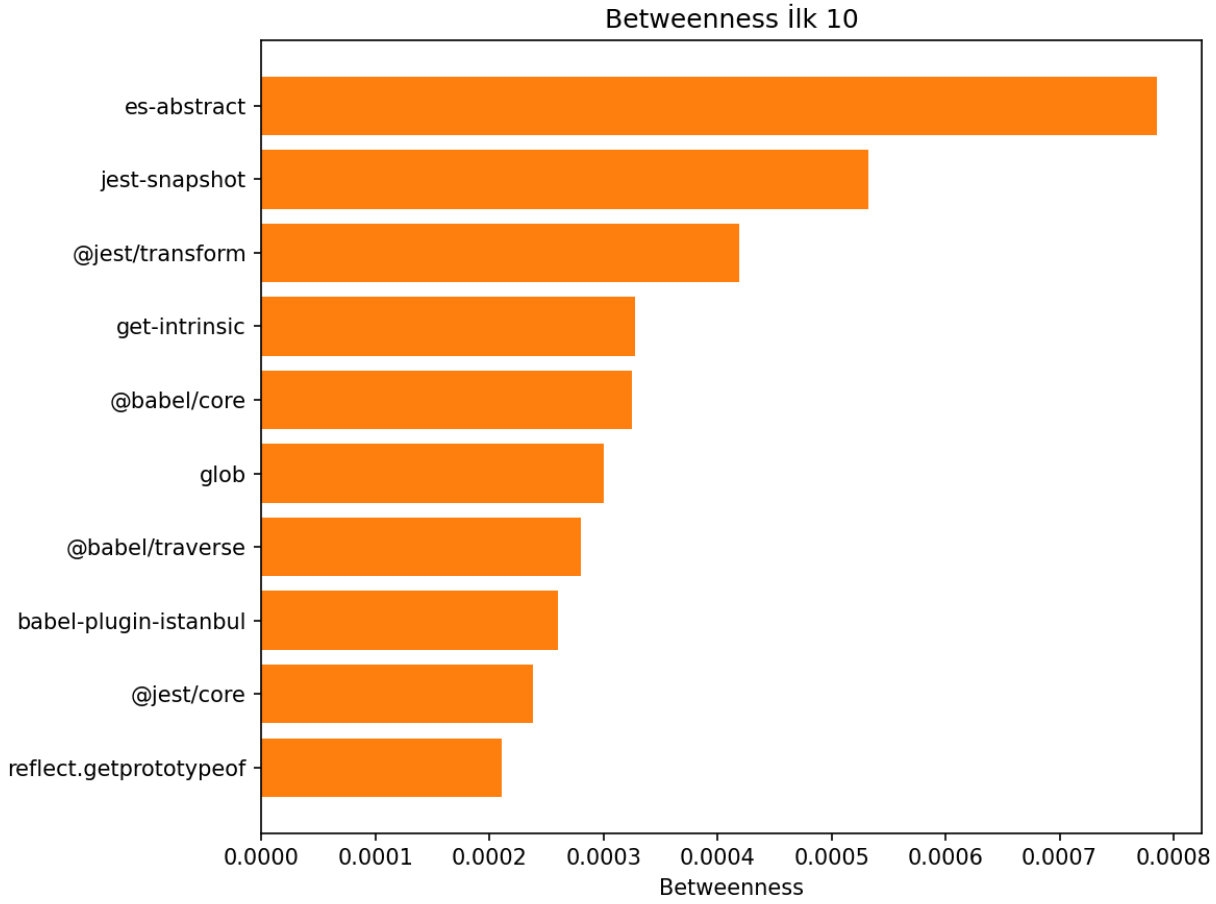
Şekil 3: In-degree açısından ilk 10 paket. Omurga niteliğindeki çekirdek bağımlılıklar.

Yorum — Bu paketlerdeki bir bozulma, bağımlıların çokluğu nedeniyle zincirleme etkiler yaratabilir; bakım ve izleme önceliği yüksektir.



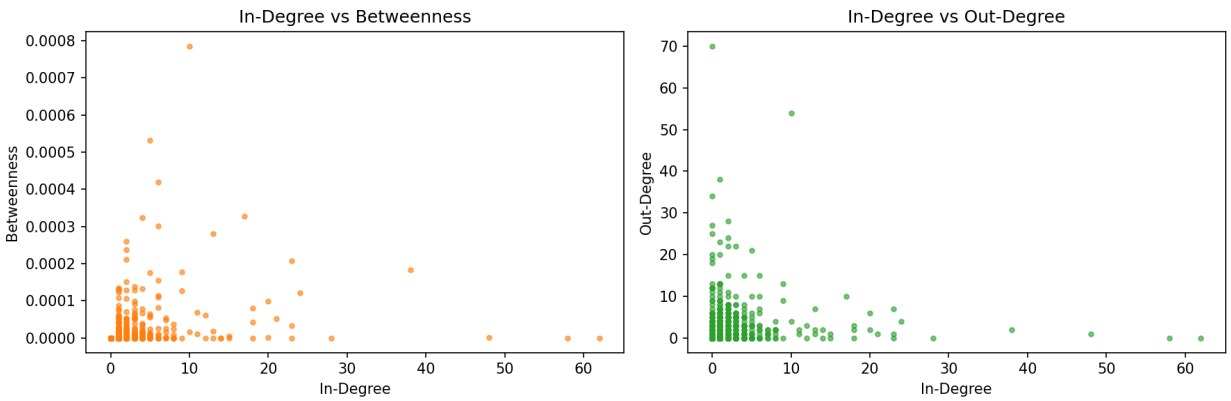
Şekil 4: Out-degree açısından ilk 10 paket. Geniş bağımlılık yüzeyine sahip bileşenler.

Yorum — Yüksek out-degree, tedarik riskine duyarlılığı artırır; bu paketlerin alt bağımlılıklarında oluşan sorunlar doğrudan etkiler yaratır.



Şekil 5: Betweenness merkeziyeti açısından ilk 10 paket. Köprü/ana arter konumlar.

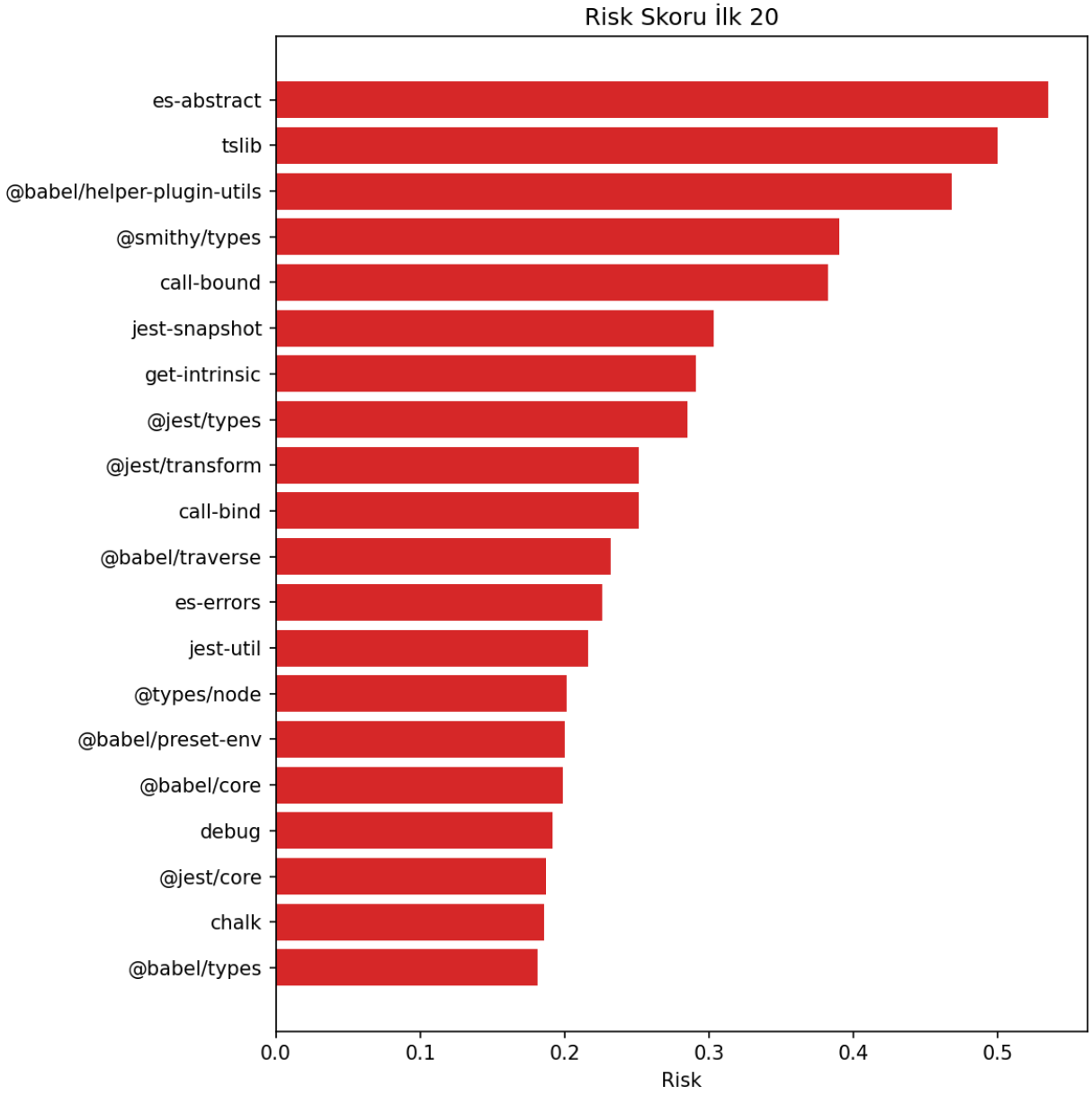
Yorum — Köprü konumlar, alt-ağlar arasında akışın zorunlu geçtiği düğümler olup, hedefli saldırılara karşı hassastır.



Şekil 6: Metrikler arası korelasyonlara ilişkin saçılım grafikleri. In-degree ile betweenness çoğu zaman birlikte yükselir; out-degree ilişkisi bağlama bağlıdır.

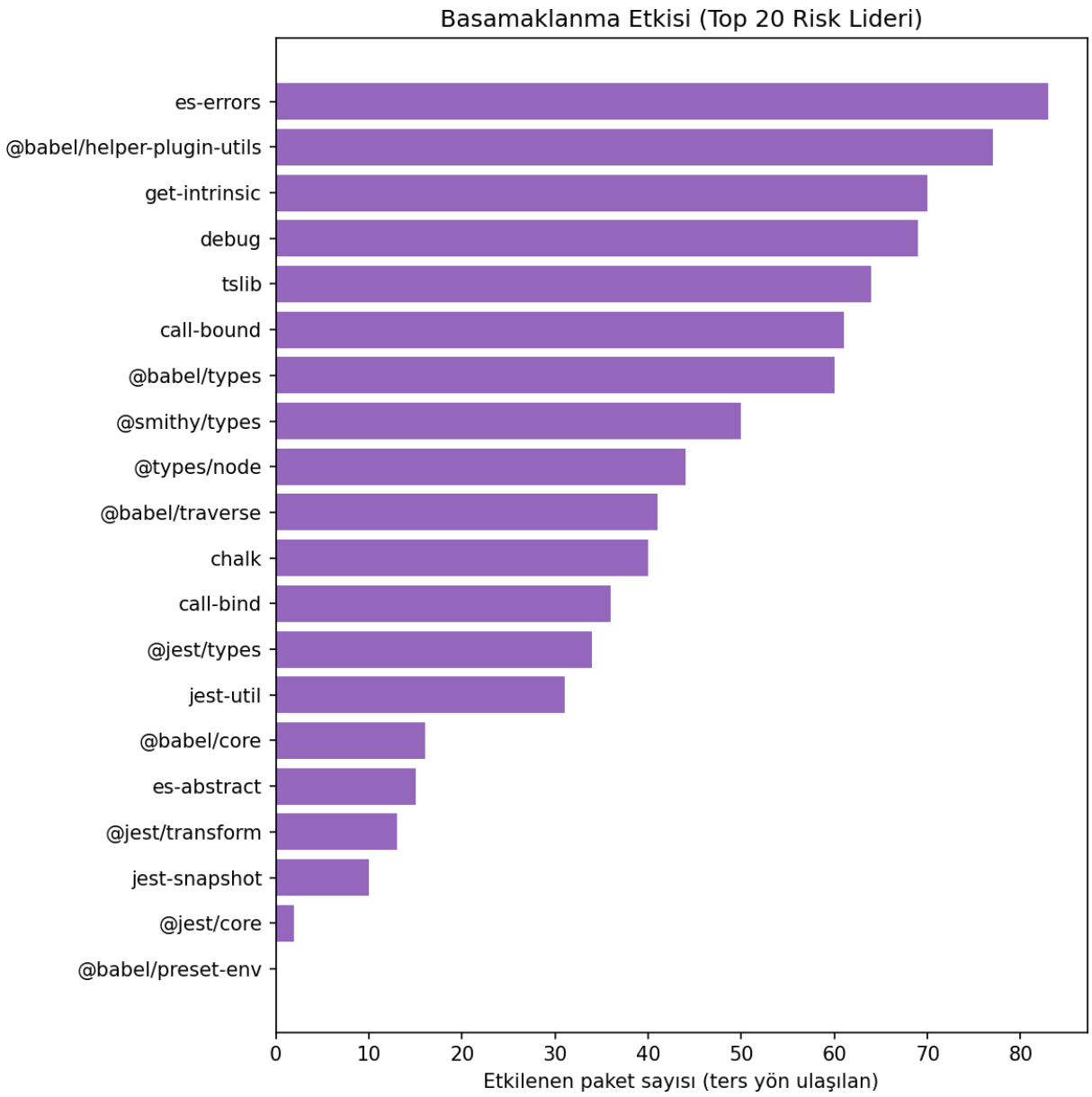
Yorum — Korelasyonlar, tekil metriklerin farklı boyutları yakaladığını; bileşik skora ihtiyaç olduğunu doğrular.

3.3 Bileşik Risk ve Kaskad Etki



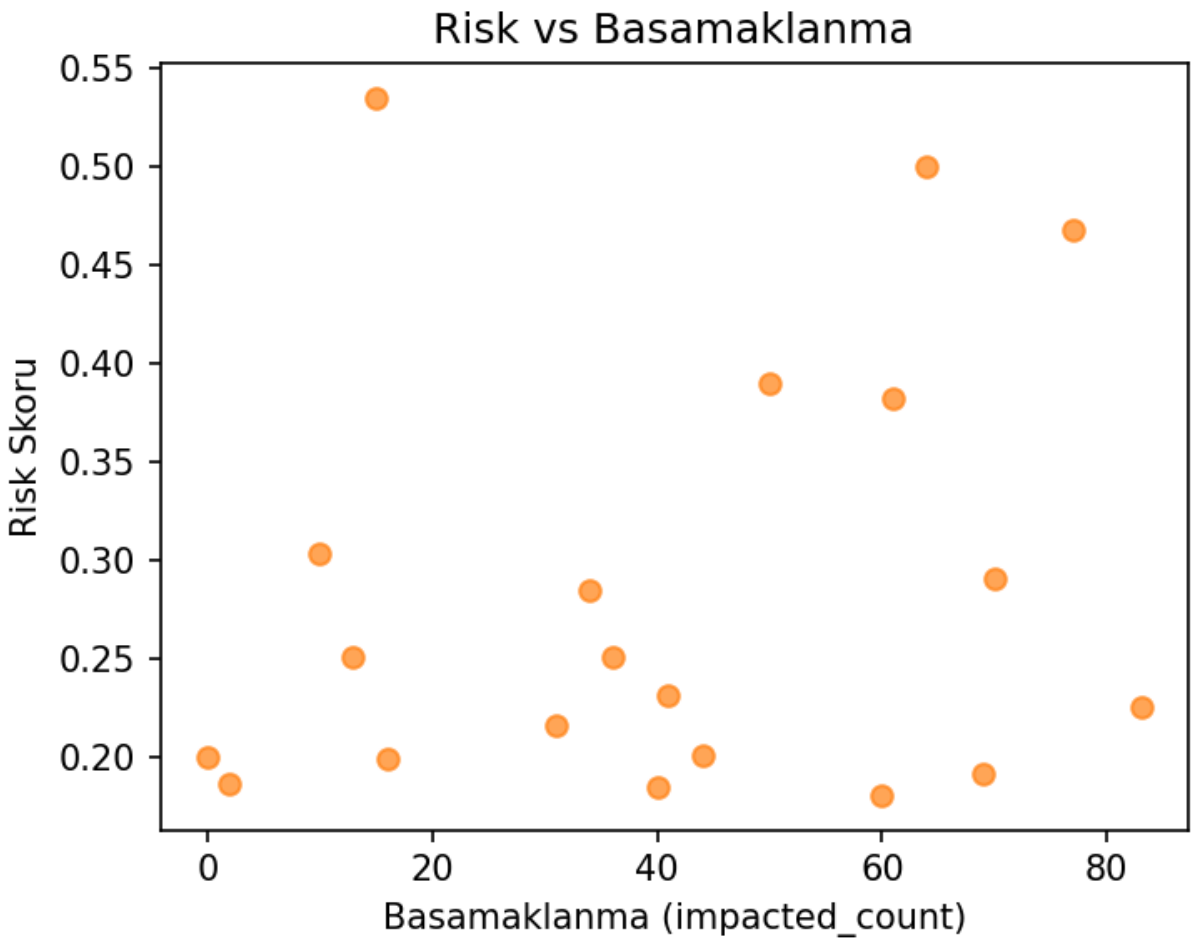
Şekil 7: Bileşik risk skoruna göre ilk 20 paket. Ağırlıklar: in=0.5, out=0.2, btw=0.3.

Yorum — Liste, yüksek in-degree ve betweenness kombinasyonuna sahip paketlerin üst sıraları domine ettiğini gösterir.



Şekil 8: En riskli 20 paket için kaskad etki (ters yön dependents) büyüklükleri. Değerler, ters yön erişilebilir düğüm sayısını temsil eder.

Yorum — Yüksek risk çoğunlukla daha geniş kaskadla örtüşse de, topolojiye bağlı istisnalar görülebilir.



Şekil 9: Risk skoru ile kaskad etki (dependents yönü) ilişkisi. Doğrusal olmayan yapı, tek metrikle açıklamanın yetersizliğine işaret eder.

Yorum — Kaskad etkisi, komşuluk yapısına duyarlıdır; benzer riskli düğümler farklı kaskad profilleri üretebilir.

3.4 Tablolar: Derece ve Betweenness

Yorum — In-degree tablosu, omurga paketleri; out-degree tablosu, geniş bağımlılık yüzeyini; betweenness tablosu ise köprü konumları nicel olarak özetler.

Tablo 1: Top 20 In-Degree (Toplam Dugumler)

Paket	In-Degree	Out-Degree	Betweenness	TopN?
tslib	62	0	0.000000	True
@babel/helper-plugin-utils	58	0	0.000000	True
@smithy/types	48	1	0.000001	True
call-bound	38	2	0.000183	True
es-errors	28	0	0.000000	True
call-bind	24	4	0.000121	True
@jest/types	23	7	0.000207	True
@types/node	23	1	0.000034	True
chalk	23	0	0.000000	True
debug	21	1	0.000051	True
@aws-sdk/types	20	2	0.000002	True
jest-util	20	6	0.000099	True

Paket	In-Degree	Out-Degree	Betweenness	TopN?
@babel/types	18	2	0.000079	True
define-properties	18	3	0.000043	True
graceful-fs	18	0	0.000000	True
get-intrinsic	17	10	0.000328	True
es-object-atoms	15	1	0.000006	True
gopd	15	0	0.000000	True
@smithy/protocol-http	14	2	0.000000	True
semver	14	0	0.000000	True

Tablo 2: Top 20 Out-Degree (Toplam Dugumler)

Paket	Out-Degree	In-Degree	Betweenness	TopN?
@babel/preset-env	70	0	0.000000	True
es-abstract	54	10	0.000785	True
@aws-sdk/client-sso	38	1	0.000075	True
eslint	34	0	0.000000	True
@jest/core	28	2	0.000238	True
express	27	0	0.000000	True
webpack	25	0	0.000000	True
jest-config	24	2	0.000128	True
@jest/reporters	23	1	0.000039	True
jest-runner	22	2	0.000042	True
jest-runtime	22	3	0.000129	True
jest-snapshot	21	5	0.000532	True
jest-circus	20	1	0.000040	True
jsdom	20	0	0.000000	True
eslint-plugin-import	19	0	0.000000	True
eslint-plugin-react	18	0	0.000000	True
@babel/core	15	4	0.000324	True
@jest/transform	15	6	0.000419	True
babel-preset-current-node-syntax	15	2	0.000151	True
@aws-sdk/core	13	9	0.000127	True

Tablo 3: Top 20 Betweenness (Toplam Dugumler)

Paket	Betweenness	In-Degree	Out-Degree	TopN?
es-abstract	0.000785	10	54	True
jest-snapshot	0.000532	5	21	True
@jest/transform	0.000419	6	15	True
get-intrinsic	0.000328	17	10	True
@babel/core	0.000324	4	15	True
glob	0.000301	6	6	True
@babel/traverse	0.000280	13	7	True
babel-plugin-istanbul	0.000260	2	5	True
@jest/core	0.000238	2	28	True
reflect.getprototypeof	0.000211	2	8	True
@jest/types	0.000207	23	7	True
call-bound	0.000183	38	2	True
jest-message-util	0.000178	9	9	True
@babel/helper-compilation-targets	0.000176	5	5	True

Paket	Betweenness	In-Degree	Out-Degree	TopN?
jest-haste-map	0.000156	6	10	True
babel-preset-current-node-syntax	0.000151	2	15	True
@babel/generator	0.000139	3	5	True
which-builtin-type	0.000134	1	13	True
browserslist	0.000133	4	5	True
jackspeak	0.000132	1	1	True

3.5 Tablolar: Risk ve Kaskad

Yorum — Risk sıralaması, çok boyutlu kritiklik sinyalini tek bir ölçekte verir; kaskad tablosu, etki alanının büyüklüğünü gösterir.

Tablo 4: Top 20 Risk Skoru

Paket	Risk	In-Degree	Out-Degree	Betweenness	TopN?
es-abstract	0.534931	10	54	0.000785	True
tslib	0.500000	62	0	0.000000	True
@babel/helper-plugin-utils	0.467742	58	0	0.000000	True
@smithy/types	0.390249	48	1	0.000001	True
call-bound	0.382129	38	2	0.000183	True
jest-snapshot	0.303511	5	21	0.000532	True
get-intrinsic	0.290993	17	10	0.000328	True
@jest/types	0.284735	23	7	0.000207	True
@jest/transform	0.251456	6	15	0.000419	True
call-bind	0.251171	24	4	0.000121	True
@babel/traverse	0.231984	13	7	0.000280	True
es-errors	0.225806	28	0	0.000000	True
jest-util	0.216164	20	6	0.000099	True
@types/node	0.201331	23	1	0.000034	True
@babel/preset-env	0.200000	0	70	0.000000	True
@babel/core	0.199075	4	15	0.000324	True
debug	0.191845	21	1	0.000051	True
@jest/core	0.187008	2	28	0.000238	True
chalk	0.185484	23	0	0.000000	True
@babel/types	0.181088	18	2	0.000079	True

Tablo 5: Basamaklanma Etkisi: Top 20 (Ters yonde etkilenebilecek paket sayisi)

Paket	Etkilenen Paket Sayisi
es-errors	83
@babel/helper-plugin-utils	77
get-intrinsic	70
debug	69
tslib	64
call-bound	61
@babel/types	60
@smithy/types	50
@types/node	44
@babel/traverse	41
chalk	40
call-bind	36

Paket	Etkilenen Paket Sayisi
@jest/types	34
jest-util	31
@babel/core	16
es-abstract	15
@jest/transform	13
jest-snapshot	10
@jest/core	2
@babel/preset-env	0

3.6 Tablolar: Köprü Kenarlar

Yorum — Kenar betweenness sıralaması, parçalanmaya duyarlı bağlantıların tespitinde yol göstericidir.

Tablo 6: Edge Betweenness Ilk 10 (Yuksek kopru kenarlar)

U	V	Edge Betweenness
@jest/transform	babel-plugin-istanbul	0.000222
@jest/expect	jest-snapshot	0.000212
jest-snapshot	@jest/transform	0.000206
jest	@jest/core	0.000150
call-bound	get-intrinsic	0.000150
glob	jackspeak	0.000147
reflect.getprototypeof	which-builtin-type	0.000146
jackspeak	@isaacs/cliui	0.000140
babel-plugin-istanbul	test-exclude	0.000139
@babel/core	@babel/helper-compilation-targets	0.000138

4 İlgili Çalışmalar ve Konumlandırma

Tedarik zinciri tehdit taksonomileri, kötüye kullanım teknik haritaları ve bağımlılık ağlarının yapısal özelliklerini inceleyen zengin bir literatür bulunmaktadır. Örneğin *Backstabbers Knife Collection* saldırı taksonomisi (Backstabbers Knife Collection Project, 2019), *Hitchhiker's Guide* ekosistemler arası kurulum/çalışma zamanı tekniklerini (Hitchhiker's Guide Authors, 2020), *OSCAR* gibi dinamik analiz hattı çalışmaları ise zehirleme tespitinde pratik iyileştirmeleri rapor eder (OSCAR Research Group, 2023). NPM bağımlılık ağlarının küçük-dünya özellikleri ve hedefli düğüm çıkarımlarındaki kırılganlık da daha önce sayısallaştırılmıştır (Hafner & Others, 2018; Oldnall & Others, 2020; Authors, 2020). Bu çalışma, söz konusu bulguları operasyonel bir *yapısal risk* metriğinde birleştirip önceliklendirmeye odaklanmaktadır (Authors, 2021a, 2021b).

5 Tartışma ve Sınırlılıklar

Betweenness merkeziyeti büyük graflarda hesaplama açısından pahalıdır; bu nedenle örnekleme kullanımı pratikte gereklidir. HTTP API yanıtlarındaki geçici hatalar eksik veri doğurabilir; önbellekleme ve tekrar denemeleri bu etkiyi azaltır. Risk ağırlıkları alan-bağımlıdır; bağlamınıza göre w_{in}, w_{out}, w_{btw} değerlerini yeniden kalibre etmeniz önerilir.

6 Sonuç

Bağımlılık topolojisinin merkeziliği ve köprü konumları, paket içi zafiyet göstergelerinin ötesinde yapısal bir risk perspektifi sunar. Burada sunulan basit ama yorumlanabilir bileşik skor, sınırlı analiz kapasitesini

yüksek etkili paketlere yönlendirmek için kullanılabilir. Gelecek çalışmalar, sayfa-derecesi (PageRank), k-çekirdek ya da topluluk/çekirdek ayrışımı gibi ek metriklerle skoru zenginleştirebilir.

Çıktılar ve Tekrarlanabilirlik

Tam çıktı listesi ve görseller için `results/` klasörüne bakınız. CLI tekrarı:

```
python -m analysis.run -topN 1000 -sample-k 200 -classify
```

Kaynaklar

Authors, V. (2020). *Demystifying vulnerability propagation via dependency trees in npm.* Preprint. Retrieved from `../academic/08-Demystifyingvulnerabilitypropagationviadependencytreesinnpm.pdf`

Authors, V. (2021a). *Practical automated detection of malicious npm packages.* Conference Paper. Retrieved from `../academic/19-PracticalAutomatedDetectionofMaliciousnpmPackages.pdf`

Authors, V. (2021b). *Towards measuring supply chain attacks on package managers.* Conference Paper. Retrieved from `../academic/28-TowardsMeasuringSupplyChainAttacksonPackageManagers.pdf`

Backstabbers Knife Collection Project. (2019). *Backstabbers knife collection: A review of open source software supply chain attacks.* Whitepaper. Retrieved from `../academic/04-BackstabbersKnifeCollectionAReviewofOpenSourceSoftwareSupplyChainAttacks.pdf`

Hafner, N., & Others. (2018). *Node package manager's dependency network robustness.* Conference Paper. Retrieved from `../academic/16-Nodepackagemanager'sdependencynetworkrobustness.pdf`

Hitchhiker's Guide Authors. (2020). *The hitchhiker's guide to malicious third-party dependencies.* Report. Retrieved from `../academic/24-TheHitchhikersGuidetoMaliciousThird-PartyDependencies.pdf`

Oldnall, N., & Others. (2020). *The web of dependencies: A complex network analysis of the npm.* Thesis/Report. Retrieved from `../academic/25-ThewebofdependenciesacomplexnetworkanalysisoftheNPM.pdf`

OSCAR Research Group. (2023). *Towards robust detection of oss supply chain poisoning (oscar).* Preprint. Retrieved from `../academic/29-TowardsRobustDetectionofOSSSupplyChainPoisoning(OSCAR).pdf`